A novel neurophysiologically-inspired self-organizing cerebellar memory framework

Sintiani Dewi Teddy

2008

Sintiani, D. T. (2008). A novel neurophysiologically-inspired self-organizing cerebellar memory framework. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41119>

<https://doi.org/10.32657/10356/41119>

A Novel Neurophysiologically-Inspired Self-Organizing Cerebellar Memory Framework

Sintiani Dewi Teddy

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2008

Abstract

The human cerebellum is a major brain construct that facilitates the learning and acquisition of motor and procedural skills. Computationally, the cerebellum functions as an associative memory with stable, fast and efficient learning based on supervised error-correction. The multi-layered *Cerebellar Model Articulation Controller* (CMAC) neural network is a classical computational model of the human cerebellum. CMAC possesses strengths such as fast training, local generalization and ease of hardware implementations. This subsequently motivates its prevalent use in engineering applications. However, several drawbacks are associated with the CMAC network. They are: (1) the curse of input dimensionality; (2) a constant output resolution; (3) the generalization–accuracy dilemma; and (4) convoluted network computations. These drawbacks are fundamentally due to the uniform quantization of the CMAC memory surface, where the CMAC computing cells are regularly spaced (allocated). Two main approaches have been used to resolve these deficiencies: multi-resolution discrete and fuzzy quantization of the CMAC memory space. However, the solutions are suboptimal and they introduced high operational complexity to the CMAC network.

In this Thesis, a neurophysiologically-inspired multi-resolution (i.e. nonlinearly quantized) cerebellar learning memory framework is presented to address the above-mentioned problems encountered in the CMAC network. Neuroscience research on the human brain development and the cerebellar learning and adaptation processes has motivated the construction of a data-driven approach for computing (memory) cell allocation in the proposed cerebellar memory framework. This culminates to the development of two novel computational architectures: (1) the HCAQ-CMAC network that incorporates the brain’s neuronal competition mechanism and (2) the PSECMAC network that synthesizes the cerebellum’s structural plasticity phenomenon into their respective computing cell allocation processes. In addition, the learning processes of these networks are shown to be functionally stable. The Thesis also proposes a three-stage learning paradigm with novel memory patching and upsizing algorithms to directly address the long training time and the undesirable system performances due to the partial or insufficient training of a large CMAC network. The novel *Three-Stage Learning CMAC* paradigm (TSL-CMAC) functionally correlates to a three-stage psychological learning model for the acquisition of human skills. The TSL-CMAC paradigm is subsequently evaluated with the control of a hypothetical plant while the proposed HCAQ-CMAC and PSECMAC networks are employed for automated car driving, intelligent insulin regulation for diabetes management, option pricing and arbitrage-trading as well as speaker verification. The experimental results are encouraging.

Acknowledgements

My PhD has been the fruit of hard labours of many people. First and foremost, I would like to express my deepest gratitudes and sincere appreciations to my two wonderful Thesis supervisors, Dr. Quek Hiok Chai and Dr. Edmund Lai Ming-Kit, who had played the biggest roles in the completion of this Thesis. They have constantly motivated me with their numerous encouragements and personal guidances, and have imparted their knowledge in research to me enthusiastically and whole-heartedly. I would also like to thank them for their valuable financial supports for my multiple conference trips overseas, which allowed me to interact with talented researchers world-wide, and at the same time providing me with the opportunities to visit the numerous beautiful countries. Indeed, these were breaths of fresh air and I had enjoyed the much-needed breaks during my hectic research schedule.

Special thanks to my fellow comrades at the Center for Computational Intelligence (C2I): Ang Kai Keng, Guo Zaiyi, Huang Hao Ming, Khoo Sui Yang, Liu Feng, Ma Li, Richard Jayadi Oentaryo, Tony Tan Tuan Zea, and Ting Chan Wai, for creating a warm and vibrant research environment. I am proud to be part of the center with you guys. My sincere gratitude also goes out to the technicians at C2I, Mr. Lau Boon Chee and Mr. Tan Swee Huat, for providing the prompt and excellent technical support and equipment maintenance to the computing system that I had used for my research.

I am also grateful to my mom, dad, brother and sister, who had provided the much-needed emotional support and encouragements during the course of my research. Big hugs go to Deny Sentosa, my bestest friend and fellow PhD student, who is always there to listen and to share the weal and woes of PhD life. Without them, I would not have persevered to the end. I would also like to say a big loving thank you to my husband, Tung Whye Loon, without whom the publication of this Thesis will not be possible. His love, understanding and continuous emotional support are the only reasons that I had managed to pull through those darkest days of Thesis writing. I am very thankful for his incisive feedbacks and countless proof-readings that had greatly helped me in polishing up this Thesis.

Last but not least, I would like to thank those whose names I had not mentioned but nevertheless contributed to the successful conclusion of this Thesis in one way or another and the Nanyang Technological University (NTU) for providing the scholarship and the opportunity for me to undertake this research.

Contents

Abstract	i
Acknowledgments	ii
Table of Contents	iii
List of Figures	ix
List of Tables	xiv
List of Abbreviations	xvii
1 Introduction	1
1.1 Brain-Inspired Computing	2
1.2 The Human Cerebellum	3
1.2.1 The Cerebellar Model Articulation Controller	4
1.2.2 The CMAC Variants	5
1.3 The Proposed Cerebellar Learning Memory Framework	5
1.4 Organization of The Thesis	10
2 Literature Review	12
2.1 Cerebellum and Human Procedural Memory System	12
2.1.1 Memory Formation in the Cerebellum – The Anatomy of the Cerebellar Circuit	13
2.1.2 Learning in the Cerebellum – The Physiological Aspects of the Cerebellar Circuit	16

2.2	Computational Models of The Human Cerebellum	19
2.3	The CMAC Network Architecture	20
2.3.1	Basics of CMAC Neural Network	20
2.3.2	The CMAC Network Operations	21
2.3.3	The CMAC Network Learning Process	23
2.3.4	Strengths and Weaknesses of The CMAC Network	24
2.4	The CMAC Variants	27
2.5	Soft Computing Techniques	30
2.5.1	Artificial Neural Networks	31
2.5.2	Fuzzy Systems and Hybrid Neural-Fuzzy Systems	31
2.5.3	Clustering Techniques	33
2.6	Summary	33
3	The HCAQ-CMAC Architecture	35
3.1	HCAQ-CMAC's Neurophysiological Motivations	36
3.2	The HCAQ-CMAC Network	37
3.2.1	The HCAQ-CMAC Structural Learning Process	39
3.2.2	The HCAQ-CMAC Network Computational Process	45
3.3	The HCAQ-CMAC Neural Correlates	48
3.4	Learning Convergence of the HCAQ-CMAC Network	49
3.4.1	Mathematical Perspective of The HCAQ-CMAC Network	50
3.4.2	Learning Convergence of The HCAQ-CMAC Network	55
3.5	Case Studies	56
3.5.1	Case I: Automatic Control of Car Maneuver	56
3.5.2	Case II: Modeling The Dynamics of The Human Glucose Metabolic Process	61
3.6	Summary	69

4	The PSECMAC Architecture	70
4.1	PSECMAC's Neurophysiological Motivations	72
4.2	The PSECMAC Network	73
4.2.1	The PSECMAC Structural Learning Process	74
4.2.2	The PSECMAC Network Computational Process	81
4.2.3	The PSECMAC Learning Process	85
4.3	The PSECMAC Neural Correlates	87
4.4	The PSECMAC Learning Convergence	87
4.4.1	Mathematical Perspective of The PSECMAC Network	87
4.4.2	Learning Convergence of The PSECMAC Network	95
4.5	Case Study I: Option Pricing and Arbitrage Trading	96
4.5.1	A PSECMAC based Option Pricing Model	99
4.5.2	A Cerebellar Associative Memory Approach to Arbitrage Trading	106
4.6	Case Study II: Banking Failure Classification	114
4.7	Summary	119
5	TSL-CMAC: A Neurologically-Inspired CMAC Learning Framework	121
5.1	The Control of A Second Order Hypothetical Plant : A Study of The CMAC Surface Traversal	123
5.2	C-Patch – To Generate A Plausible Memory Surface for the Untrained CMAC Cells	128
5.3	Performance of the C-Patch Technique	133
5.4	TSL-CMAC: A CMAC Learning Framework	137
5.4.1	The Initial Learning Stage	138
5.4.2	The Network Generalization Stage	139
5.4.3	The Continuous Fine-Tuning Stage	140
5.5	Performance of the TSL-CMAC Framework	142
5.6	Summary	148

6	A Cerebellar-based Approach to Text-Dependent Speaker Verification	150
6.1	The Human Speech Production System	154
6.2	The PSECMAC-based Speaker Verification System	156
6.2.1	Preprocessing of Speech Signal	157
6.2.2	Feature Extraction	159
6.2.3	PSECMAC Modeling of Speaker-Specific Characteristics	163
6.3	Experiments and Results	164
6.3.1	Dataset	164
6.3.2	Experimental Results and Analysis	166
6.4	Summary	172
7	Cerebellar-Based Insulin Pump Control for the Personalized Treatment of Diabetes	173
7.1	Closed-Loop Glycaemic Controls for Diabetes Treatment	176
7.2	The Physiological Process of Pancreatic Insulin Secretion	181
7.3	The Patient Profile – A Simulated Human Subject	183
7.4	The PSECMAC Insulin Model	186
7.5	Modeling the Insulin Response under Different Dietary Profiles	193
7.6	PSECMAC-based Control of Insulin Infusion for Diabetes Treatment	205
7.7	Summary	216
8	Conclusions and Future Research	220
8.1	Major Contributions of The Thesis	221
8.1.1	Theoretical Research Contributions	222
8.1.2	Empirical Research Contributions	223
8.2	Constraints of the Reported Work	226
8.3	Future Research Directions	227
8.3.1	Extensions to the Cerebellar Memory Framework	227

8.3.2 Application Areas of the Cerebellar Memory Models	229
Bibliography	231
A Major Development Milestones in ANN Research	257
B Single Layer Model of The CMAC Network	259
C The HCAQ-CMAC Proof of Learning Convergence	264
D Modified PSEC Density Clustering Algorithm	270
E The PSECMAC Proof of Learning Convergence	277
F Financial Covariates for Banking Failure Classification	288
G The MZCR Algorithm	290
H ROC Curves of the PSECMAC- based Speaker Verification System	291
H.1 Unbalanced Training Scenario	291
H.2 Balanced Training Scenario	293
I Insulin and Its Delivery Systems – A Review	296
J Current Glucose-Insulin Regulatory Systems	299
K The GlucoSim Simulator	301
K.1 Puckett’s Model of the Human Metabolic Process	301
K.2 Sorensen’s Model of the Human Metabolic Process	302
K.3 The GlucoSim Model of the Human Metabolic Process	303
L Asian Diet – Sample Menu	306
M The Extracted Glucose Variables for Modeling of Insulin Response	307

N	The MCES Algorithm	311
O	Results of the PSECMAC Static Closed-Loop Insulin Control	314
P	Publication List	317

List of Figures

1.1	Motivations and research outline of the proposed cerebellar memory framework	6
2.1	An atlas of the human brain	13
2.2	A diagram of the cerebellar circuitry: GC - Granule Cell; PC - Purkinje Cell; CN - Deep Cerebellar Nuclei; IO - Inferior Olive; • - inhibitory connection; o - excitatory connection (Adapted from (Doya, 1999))	15
2.3	Schematic diagram of the cerebellum (Adapted from (Albus, 1989))	15
2.4	Schematic diagram of the CMAC neural network (Adapted from (Albus, 1989))	20
2.5	An example of a 2D CMAC network (m_1^* refers to the set of quantization functions along the S_1 dimension and m_2^* refers to the set of quantization functions along the S_2 dimension)	21
2.6	A classification of CMAC variants in the literature	27
3.1	Comparison of CMAC and HCAQ-CMAC memory structure for a given input-output characteristic surface	39
3.2	Illustration of the HCAQ-CMAC non-uniform quantization process for an arbitrary input dimension j where $\hat{M} = 6$	43
3.3	The flowchart and computational complexity of the HCAQ-CMAC structural learning process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; S – number of training samples; $\hat{M}$ – number of HCAQ-CMAC memory cells per dimension	44
3.4	The flowchart and computational complexity of the HCAQ-CMAC computational process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of HCAQ-CMAC memory cells per dimension; N – HCAQ-CMAC neighborhood size	48
3.5	An example of the 2-input HCAQ-CMAC memory array	50

3.6	2D HCAQ-CMAC $\mathbf{Z} \rightarrow \mathbf{W}$ mapping	50
3.7	Driving simulation environment	57
3.8	Driving tracks	58
3.9	Sample glucose metabolism data output from the GlucoSim simulator . . .	64
3.10	Soft-windowing weighting functions to compute the carbohydrate content of meal(s) in the segmented windows of the 6-hours food history (In the figure, the current time is 1840 hrs and there are three instances of food intakes in the previous 6-hours; that is, lunch at 1245 hrs, afternoon snack at 1524 hrs and dinner at 1833 hrs respectively.)	65
3.11	Modeling results of the CMAC and HCAQ-CMAC network for the glucose metabolic process of Subject A	68
4.1	Comparison of CMAC and PSECMAC memory surface with neighborhood activation of the computing cells for a 2-dimensional input example	71
4.2	An example of data-driven quantization of 2D PSECMAC memory cells . .	74
4.3	A sample output of the MPSEC clustering technique	78
4.4	Adaptive memory cell distribution in cluster $C_{j,n}^{(\tau_{\text{end}})}$	79
4.5	The flowchart and computational complexity of the PSECMAC structural learning process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; S – number of training samples; $\hat{M}$ – number of PSECMAC memory cells per dimension; τ_{max} – maximum number of LVQ iteration; $n_{c,j}^{(-1)}$ – initial number of density clusters; $n_{c,j}^{\text{end}}$ – final number of density clusters	82
4.6	An example of a 2D PSECMAC neighborhood	83
4.7	The flowchart and computational complexity of the PSECMAC network computational process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of PSECMAC memory cells per dimension; N – PSECMAC neighborhood size	86
4.8	An example of the 2-input PSECMAC memory contents	89
4.9	2D PSECMAC $\mathbf{Y} \rightarrow \mathbf{W}$ mapping	89
4.10	General framework of the proposed mis-priced option arbitrage trading system	109
4.11	Price plot of the GBP vs USD futures for the period of the study	110
4.12	ROC curves of the PSECMAC banking failure classification	116

5.1	The Simulink diagram of the CMAC control loop	124
5.2	The CMAC characteristic surface after training with step functions	126
5.3	The devised training sequence for full training of the CMAC controller ($f_{full}(t)$)	127
5.4	The CMAC characteristic surface after training with $f_{full}(t)$	128
5.5	The random training sequences used to train the CMAC network	129
5.6	The corresponding CMAC characteristic surfaces after training with the randomly generated training profiles	130
5.7	The workings of the proposed C-Patch algorithm (2-input CMAC example)	131
5.8	The C-Patch Algorithm	131
5.9	The flowchart and computational complexity of the C-Patch Algorithm: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of CMAC memory cells per dimension	132
5.10	The characteristic surfaces of the two randomly-trained CMAC networks after the application of the proposed C-Patch technique	133
5.11	The plots of control performances of a fully-trained CMAC network as compared to a partially-trained CMAC network and its C-Patched counterpart for various test sequences	137
5.12	The TSL-CMAC Learning Framework	139
5.13	The C-Upsize Algorithm	140
5.14	The flowchart and computational complexity of the C-Upsize Algorithm: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; OldCMAC – number of the original CMAC memory cells per dimension; NewCMAC – number of the up-sized CMAC memory cells per dimension; N – CMAC neighborhood size	141
5.15	Simulation results under different CMAC configurations for staircase reference	144
5.16	The random training sequence used to partially-train the CMAC networks .	145
5.17	Observed learning (closed-loop) error curves and controlled plant outputs for a 32×32 CMAC with TSL learning (denoted as CMAC _{TSL}) and a 32×32 CMAC with no TSL learning (denoted as CMAC _{Direct}) (CMAC _{TSL} refers to the 32×32 CMAC which was created from the partially-trained, patched and enlarged 8×8 CMAC network in the previous experiments. CMAC _{Direct} refers to the partially-trained 32×32 CMAC from Table 5.3.)	147
6.1	Classification of speaker recognition system	152

6.2	The human speech production system: (1) nasal cavity; (2) hard palate; (3) alveolar ridge; (4) velum; (5) apex; (6) dorsum; (7) uvula; (8) radix; (9) pharynx; (10) epiglottis; (11) false vocal cords; (12) vocal cords; (13) larynx; (14) esophagus, and (15) trachea (Adapted from (Lemmetty, 1999))	155
6.3	The proposed PSECMAC-based speaker verification system	156
6.4	Examples of the voiced, unvoiced and silence speech frames	158
6.5	The cepstrum of the speech signal (Adapted from (Gold and Morgan, 2000))	161
6.6	The Mel-frequency bandpass filters	161
6.7	The partitioning of the three CV groups for the MFCCs dataset of a speaker	166
7.1	Taxonomy of Glucose-Insulin Regulatory Systems	177
7.2	The human model of energy and glucose homeostasis. Adapted from (Porte Jr. et al., 2005)	182
7.3	Sample glucose metabolism data output from the GlucoSim simulator . . .	185
7.4	Measurement of prediction performance using the MSE and MAE indicators	191
7.5	3-days generalization performances of the CMAC and PSECMAC networks in modeling the insulin profile of a healthy person	194
7.6	3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the undereat diet	197
7.7	3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the overeat dietary profile .	199
7.8	3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the irregular dietary profile	200
7.9	3-days generalization performances of the CMAC and PSECMAC networks (after-tuning) in modeling the insulin profile of Subject A for the overeat dietary profile	203
7.10	3-days generalization performances of the CMAC and the PSECMAC networks (after-tuning) in modeling the insulin profile of Subject A for the irregular dietary profile	204
7.11	Static closed-loop control setup for PSECMAC-based glucose-insulin regulation	207
7.12	3-days control performances of the PSECMAC-based static closed-loop glucose-insulin regulatory system for different dietary profiles	211

7.13	Adaptive closed-loop control setup for PSECMAC-based glucose–insulin regulation	213
7.14	3-days diabetic blood glucose control performances by the PSECMAC-based adaptive closed-loop glucose–insulin regulatory system for the different dietary profiles	218
B.1	An example of a single layer implementation of the 2D CMAC network example shown in Figure 2.5 of Chapter 2	260
D.1	The flowchart and computational complexity of the MPSEC algorithm: $O(\cdot)$ – the Big-O notation; S – number of training samples; $\tau_{\max}$ – maximum number of LVQ iteration; $n_{c,j}^{(-1)}$ – initial number of density clusters	276
H.1	The ROC curves of the PSECMAC-based speaker verification system across the ten speakers for the "unbalanced" training scenario	293
H.2	The ROC curves of the PSECMAC-based speaker verification system across the ten speakers for the "balanced" training scenario	295
I.1	Various sites of insulin injection	297
K.1	Pharmacokinetics diagrams of the glucose and insulin models of GlucoSim. Adapted from (GlucoSim, Online)	304
N.1	Feature selection process of the MCES algorithm	313
O.1	3-days control performances of the PSECMAC-based static closed-loop glucose–insulin regulatory system under different dietary profiles	316

List of Tables

2.1	Various computational models of the human cerebellum	18
2.2	Application areas of the CMAC network	25
2.3	Existing FCMAC Architectures	28
2.4	Existing Discrete CMAC Architectures with Adaptive Quantization	29
3.1	Correspondence between the neurophysiological aspects of the human cerebellum and the functionalities of the proposed HCAQ-CMAC network	49
3.2	The sensors definition of the Car Simulator	58
3.3	Comparison of driving results for the various benchmarked networks using the Autopilot System*	60
3.4	Comparison of Cell Occupancy Rates (COR) for the various benchmarked networks*	62
3.5	The profile of the simulated healthy person (Subject A)	63
3.6	Comparison of results for the various benchmarked networks on the modeling of the human glucose metabolism process	66
4.1	Correspondence between the neurophysiological aspects of the human cerebellum and the functionalities of the proposed PSECMAC network	88
4.2	Simulation set-ups based on permutations of the three sub-groups A, B and C to define the training and testing sets of the proposed PSECMAC option pricing model	101
4.3	Performances of the proposed PSECMAC option pricing model	102
4.4	Benchmarking results for various option pricing model	103

4.5	Comparison of the Cell Occupancy Rate of the PSECMAC and CMAC Option Pricing Model	105
4.6	Performances of the Black-Scholes option pricing model (All parameters are derived directly from the spot market)	106
4.7	Arbitrage performances of the proposed PSECMAC-based option trading system. (Note: UO is option under-priced arbitrage opportunity; OO is option over-priced arbitrage opportunity; and ROI denotes the return on investment)	111
4.8	Comparison of arbitrage performances by the proposed PSECMAC-based arbitrage trading system (PSECMAC), the BS-based arbitrage trading system (Black-Scholes), simple Buy and Hold Strategy (BHS) and Buy and Hold Strategy with Hedging (BHSH)	111
4.9	EER readings extracted from the ROC curves - Unbalanced training scenario	117
4.10	EER readings extracted from the ROC curves - Balanced training scenario	118
4.11	Comparison of the Cell Occupancy Rate of the PSECMAC and CMAC Banking Failure Classification	118
5.1	Comparison of the control performances of the various networks*	135
5.2	The functional correlates of the proposed TSL-CMAC learning framework and the three-stage model of skill acquisition by Fitts and Posner	142
5.3	Comparison of the control performances of the various networks*	143
6.1	The performances of the PSECMAC-based speaker verification system . . .	168
6.2	The performances of the benchmarked speaker verification systems	170
7.1	The profile of the simulated healthy Subject A	184
7.2	Computations for the generation of GlucoSim input parameters for the normal, under and over dietary profiles. (Note: 100 sets of input parameters are generated. $U(x,y)$: a uniformly distributed random number between x and y inclusively; and $N(\mu,\sigma)$: a normally distributed random number with mean μ and standard deviation σ .)	187
7.3	Simulation results for the various insulin models	191
7.4	Simulation results of the insulin models for the various dietary profiles . . .	195
7.5	Performance comparisons between the PSECMAC and CMAC insulin models for the various dietary profiles	195

7.6	Simulation results of the tuned insulin models for the overeat and irregular dietary profiles	202
7.7	Performance comparisons between the tuned PSECMAC and CMAC insulin models for the various dietary profiles	202
7.8	The control performances of the PSECMAC-based static closed-loop control system for various dietary profiles	209
7.9	The control performances of the PSECMAC-based adaptive closed-loop control system for the various dietary profiles	215
7.10	Performance comparisons between the static and the adaptive PSECMAC insulin model for the various dietary profiles	215
8.1	Estimated worst-case computational complexity of the various algorithms presented in the Thesis	228
A.1	Important milestones in ANN research presented in chronological order . . .	257
F.1	Definition of covariates and their expected impact on failure (Numbers in brackets are the identification of the data elements from the Call Reports) .	288
I.1	Various types of insulin and their actions	297
J.1	Existing Closed-Loop Glucose-Insulin Regulatory Systems	299
K.1	The definitions of the parameters used in the pharmacokinetics diagrams of GlucoSim	305
L.1	Typical daily meals (and carbohydrate contents) of Subject A based on the Asian diet	306
M.1	The glucose variables extracted to model the healthy insulin response . . .	308
M.2	The results of the MCES algorithm	310

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
APG	Adjustable Pattern Generator
BMI	Body Mass Index
BS	Black-Scholes
CMAC	Cerebellar Model Articulation Controller, or Cerebellar Model Arithmetic Computer
CME	Chicago Mercantile Exchange
COR	Cell Occupancy Rate
CV	Cross Validation
DCT	Discrete Cosine Transform
DHTS	Delta Hedge Trading Strategy
EER	Equal Error Rate
FCMAC	Fuzzy CMAC
FFT	Fast Fourier Transform
GenSoFNN	Generic Self-Organizing Fuzzy Neural Network
HCAQ-CMAC	Hierarchically Clustered Adaptive Quantization CMAC
I/O	Input/Output
IDTM	Inducers of Decision Table Majority
ITS	Intelligent Transportation System
LTD	Long Term Depression
LTP	Long Term Potentiation
LVQ	Learning Vector Quantization
MAE	Mean Absolute Error

MCES	Monte-Carlo Evaluative Selection
MFCC	Mel-Frequency Cepstral Coefficient
MIMO	Multi Input Multi Output
MISO	Multi Input Single Output
MLP	Multi-Layered Perceptron
MMR-CMAC	Moody's Multi Resolution CMAC
MPSEC	Modified Self-Evolving Cerebellar
MSE	Mean Squared Error
MSFB	Mel-Scale Filter Bank
MRAC	Model Reference Adaptive Control
MZCR	Modified Zero Crossing Rate
NFS	Neural Fuzzy System
PI	Performance Index
PID	Proportional-Integral-Derivative
PSEC	Pseudo Self-Evolving Cerebellar
PSECMAC	Pseudo Self-Evolving CMAC
TMR-CMAC	Tree-based Multi Resolution CMAC
TSL-CMAC	Three-Stage-Learning CMAC
RBF	Radial Basis Function
RDA	Recommended Daily Allowance
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
ROI	Return On Investment
RSPOP	Rough Set-based Pseudo Outer Product
WGNO	Weighted Gaussian Neighborhood Output
WGNU	Weighted Gaussian Neighborhood Update
ZCR	Zero-Crossing Rate

"The beginning is the most important part of the work"
–Plato

Chapter 1

Introduction

The brain is the biological organ responsible for the human intelligence, where complex networks of neurons collaborated to create a massive information computing structure that forms the neural basis for thoughts, emotions, perceptions, learning and memory, and consciousness. The most profound and fundamental aspects of the human intelligence are arguably the capacity for learning and memory (Kandel et al., 2000a; Okano et al., 2000), without which one is capable of nothing but simple reflexes and stereotyped behaviors. *Learning* is defined as a process that results in a consistent change in an individual's behavioral responses due to the proficiency derived through the repeated exposures to the environmental stimuli (Tomprowski, 2003) and *memory* is the coding, storage and subsequent recall of this acquired knowledge (Sweatt, 2003).

From a scientific perspective, the ability to comprehend the functional principles of the human brain associated with its role as a biological information mediator can provide much insight to advance artificial intelligence, computer science and information technology research. Following that, as part of the ongoing research efforts to develop intelligent computing systems that exhibit brain-like information processing capabilities (Amari and Kasabov, 1998; Doya, 1999; Kawato and Samejima, 2007; Tan et al., 2005; Teddy et al., 2007a; Tung and Quek, 2006b), there is an intense interest in the study and development of functional models of the various brain parts that facilitate the human ability to learn and retain knowledge for future references (Atallah et al., 2004; Gluck et al., 2005; Kitamura, 2001; O'Reilly, 2006; Salu, 2001; Sim et al., 2006; Wang and Wang, 2006).

1.1 Brain-Inspired Computing

The traditional approach to machine intelligence pursued by the artificial intelligence community has provided many achievements, but has generally fallen short of the vision of developing versatile human-like intelligent systems. Revolutionary advances may be possible by building upon new approaches inspired by cognitive psychology and neuroscience. Such approaches have the potential to help researchers understand and model significant aspects of intelligence thus far not attained by classic formal knowledge modeling technology (DARPA, Online). With respect to this notion, *neurocognitive informatics* is an actively pursued topical research (Amari, 2005; Oentaryo and Pasquier, 2006; Shi and Shi, 2003; Sim et al., 2006; Tung and Quek, 2006a, 2007; Wang, 2003, 2007; Wang and Kinsner, 2006) and its primary focus is to acquire a deeper understanding of how information is represented, organized, manipulated and used within the neural structures of the human brain system; and how these various brain information processing mechanisms facilitate the high-level cognitive capabilities that are amply demonstrated in the prolific displays of human intelligence such as consciousness (Thompson and Madigan, 2005), decision-making (Deppe et al., 2005; Marschner et al., 2005), thinking and reasoning (Holyoak and Spellman, 1993; Rips, 1990) and problem solving (Newell and Simon, 1972) in everyday life.

From a computational perspective, the human brain is regarded as a complex information processing system (Amari, 2005; Churchland and Sejnowski, 1994; Grossberg, 2000; Taylor, 2005) and a reductionist approach is generally adopted in neurocognitive informatics research to study the various information processing tracts of the biological brain via simplified abstractions of the functionalities of the major brain constructs. A significant fundamental belief of neurocognitive informatics research is that the major underlying support to the displayed and perceived human intelligence is the ubiquitous multi-faceted memory structures in the human brain. Neurocognitive informatics research thus focuses on the study and development of functional models of the brain systems that exhibit the ability to construct knowledge and to store and retrieve the acquired information for subsequent use. This leads to the construction of computational models of learning memory systems (Barto et al., 1999; Gluck et al., 2003; Moren, 2002).

It is important to note that these models generally do not depict every physiological detail of the corresponding memory systems in the biological brain. Instead, their computing

structures and information processing mechanisms are often inspired by the neurophysiologic understanding of the underlying neural organisation, physiological functions and signal pathways of the brain. Hence, they are also referred to as *brain-inspired computing architectures* (Amari, 2005; Duch, 2005). The objective of neurocognitive informatics research is therefore to emulate and replicate facets of the human intelligence via a functional description of the mechanisms and processes involved in human learning and memory formation using computational intelligence and soft computing techniques such as neural networks, fuzzy systems, evolutionary computations and machine learning models. It is hoped that this will bring forth the development of more powerful, flexible and intelligent brain-inspired computing systems that are capable of human-like information processing to solve ill-defined, complex and dynamic problems.

1.2 The Human Cerebellum

Of the various major brain constructs (i.e. hippocampus, amygdala, prefrontal cortex etc.) studied by the neurocognitive informatics research fraternity, the cerebellum is one of the most extensively explored brain region. The cerebellum is one brain region in which the neuronal connectivity is sufficiently regular to facilitate a comprehensive understanding of its functional properties. It is located at the bottom rear of the head (the hind-brain) and constitutes a vital part of the brain system that mediates motor movement control and a number of sub-conscious cognitive functions (Middleton and Strick, 1998a), including the learning and memory of procedural and motor skills. The human cerebellum functions as a motor movement calibrator (Albus, 1989) and possesses the capability to model highly complex and nonlinear physical dynamics to facilitate the precise and rapid executions of dexterous movements and fluid motor reflexes (Eichenbaum, 2002). Hence, it is highly desirable to construct a computational model of the human cerebellum in order to capture and to emulate its rapid and nonlinear function learning capability. Such a computational tool has diverse use in applications such as autonomous control and pattern recognition where there are generally no precise mathematical descriptions of the problem's characteristics and the inherent process behavior can only be inferred from measurable physical observations.

1.2.1 The Cerebellar Model Articulation Controller

The *Cerebellar Model Articulation Controller* (CMAC) (Albus, 1975a) is a neural network inspired by the neurophysiological properties of the human cerebellum and is widely recognized for its localized generalization and rapid algorithmic computations. As a computational model of the human cerebellum, CMAC¹ manifests as an associative memory network (Albus, 1975b), and employs error correction signals to drive the network learning and memory formation processes. This allows for simple computation, fast training, local generalization and ease of hardware implementation (Albus, 1989), which subsequently motivates the prevalent use of CMAC-based systems for process control and optimizations (Cetinkunt and Donmez, 1993; Imamizu et al., 2000; Ku et al., 1998; Larsen et al., 1995; Lin and Hyongsuk, 1991; Yamamoto and Kaneda, 1999), modeling and control of robotic manipulators (Commuri et al., 1997; Kano and Takayama, 1990), as well as various signal processing and pattern-recognition tasks (Huang et al., 1997; Ker et al., 1997; Wahab et al., 2001). The learning convergence of the CMAC network has also been established in (He et al., 2001; Lin and Chiang, 1997; Wong and Sideris, 1992).

However, there are several major computational limitations associated with the CMAC network that arise from the rigidness of its computing structure. The CMAC associative memory network employs a highly regularized grid-like computing structure (i.e. equally spaced memory cells along each input dimension) that indirectly enforces the uniform quantization of a problem's input-output (I/O) mapping space. On the other hand, meaningful real-life applications are generally *heteroskedastic*, where the problems are often characterized by highly nonlinear I/O trends and statistically varying data patterns. Such observations implied that specific regions of these problems' I/O associative spaces are more informative (and therefore demand a higher modeling resolution) than others. For such an application, the simplistic approach of adopting a uniformly quantized I/O mapping space (as in the memory space of CMAC) to model the problem's input-output data characteristics may not be adequate as it often leads to: (1) a suboptimal system where there is a lack of modeling accuracy at the important regions of the I/O mapping space; (2) poor memory utilization as characterized by a large number of untrained memory (computing) cells assigned to the sparsely populated data regions; and (3) a trade-off between the generalization capability

¹CMAC is also known as the *Cerebellar Model Arithmetic Computer*. The CMAC network was first proposed as a function modeler for robotic controllers by Albus in 1975 (hence the name), but has also been extensively used in reinforcement learning and also as a computing system for automated classification in machine learning.

and the modeling fidelity of the network. That is, a small-sized CMAC with fewer memory cells is able to better generalize the characteristics of the training data, but a large-sized CMAC network with fine modeling resolution produces more accurate outputs.

1.2.2 The CMAC Variants

In the existing literature, there has been a number of attempts to address the above-mentioned limitations of the CMAC network. Generally, these efforts can be broadly classified into two major approaches: the multi-resolution discrete CMAC and the fuzzy CMAC variants. The multi-resolution discrete CMAC variants employ computing cells with crisp boundaries and attempt to produce a more efficient mapping of the I/O associative space via the optimization of the network quantization decision functions (Gao et al., 1996; Kim and Lin, 1992; Lee et al., 2003; Yeh and Lu, 2002) or with the use of multi-layered CMAC networks of increasing resolutions (Menozzi and Chow, 1997; Moody, 1989). The fuzzy CMAC systems, on the other hand, employ fuzzified cell boundaries of varying sizes to enhance memory efficiency (Jou, 1992; Ker et al., 1997; Kim, 2002; Lane et al., 1992; Nie and Linkens, 1993; Ozawa et al., 1992; Zhang and Qian, 2000) as well as ensuring network interpretability via the incorporation of formalized fuzzy inference schemes (Jiang et al., 2007; Quek and Guo, 2007; Quek et al., 2006; Sim et al., 2006; Ting and Quek, 2006). These CMAC extensions are discussed in Chapter 2 of the Thesis. In general, current attempts have limited success in addressing CMAC's computational shortcomings and often do so at the expense of introducing a high operational complexity. Most of the CMAC variants also do not attempt to establish mathematical proof of the system's learning convergence, which is often crucial for control and function approximation tasks. Moreover, since human behavioral studies have established that learning stability in the human cerebellum is central to the acquisition and the subsequent execution of smooth and precise motor movements (Kleim et al., 1998a; Medina and Mauk, 1999; Steele and Mauk, 1999), any credible computational model of the cerebellum should therefore guarantee a stable learning process.

1.3 The Proposed Cerebellar Learning Memory Framework

The primary objective of this research is to develop a cerebellar associative memory framework to address the computational deficiencies encountered in the CMAC network.

Figure 1.1: Motivations and research outline of the proposed cerebellar memory framework

Figure 1.1 depicts the blueprint of the research efforts reported in this Thesis. Fundamentally, the computational limitations of the CMAC network can be analyzed from both its architectural and training principles. From the architectural perspective, the CMAC network is a static associative memory model whose structure epitomizes the nonlinear mapping between the cerebellar afferent inputs and the Purkinje cell outputs in the human cerebellum (see Chapter 2 for details). Computationally, CMAC is an associative memory-based neural network that performs functional mapping of multi-dimensional input-output data tuples. However, the uniform partitioning of the CMAC associative space and the subsequent linear mapping of the partitioned subspaces as the CMAC network cells lead to three major architectural drawbacks as outlined in Section 1.2.1 and Figure 1.1.

In addition, the learning principle of the CMAC network also poses several fundamental problems. As a computational model of the human cerebellum (Albus, 1972, 1989), CMAC operates based on the principle that similar inputs should produce similar outputs, while inputs that are dissimilar should invoke nearly independent outputs. This translates to the localized generalization property of CMAC learning, whereby only a localized subset of CMAC computing cells (i.e. a localized region of input-output (I/O) mappings) are trained for each training pattern. While it equips the network with a certain degree of robustness against learning interference, the localized learning principle saddles CMAC with several significant training issues (see Figure 1.1): (1) Insufficient training. It is difficult to comprehensively-train the CMAC network as meticulous planning (if possible) is required to design a training profile to ensure that all the CMAC computing cells are trained; (2) An exponential increase in the training time of a CMAC network with an increased network size; and (3) Generalization-accuracy dilemma. Due to the localized training of the CMAC computing cells, smaller network size increases the generalization ability of the CMAC network while reducing the accuracy of the computed output. On the other hand, a large network size increases the output resolution at the expense of reduced generalization performance.

The computational deficiencies depicted in Figure 1.1 often result in sub-optimal performances of a CMAC-based system. Currently, there has been little attempt in the literature to address these observed problems in the construction and training of a CMAC network. In this Thesis, a novel neurophysiologically-inspired self-organizing cerebellar associative memory framework is presented to address the computational limitations of the CMAC network. The proposed cerebellar framework consists of four primary approaches (see Fig-

ure 1.1), namely: (1) *adaptive memory quantization*; (2) *neighborhood-based training*; (3) *memory patching*; and (4) *memory upsizing*.

Adaptive memory allocation refers to the non-uniform quantization of the cerebellar computing cells to cover the entire input–output mapping space. This sought to address the problems introduced by the rigid uniform partitioning of the CMAC network cells. The objective of the proposed adaptive memory allocation scheme is to assign more memory cells to model the data regions with higher information throughput. This enhances the memory utilization rate and improves the modeling accuracy of the cerebellar memory models. In addition, the flexibility of the adaptive memory allocation scheme also partially addresses the generalization–accuracy dilemma as the modeling resolution is varied according to the information characteristics of the training data across the input–output associative mapping space.

To further address the generalization–accuracy dilemma of the CMAC network, a neighborhood-based training algorithm is proposed. In the proposed neighborhood-based training scheme, each input vector to the network activates a region of computing (memory) cells. Such a neighborhood-based activation of computing cells facilitates a distributed learning paradigm in the cerebellar memory model that enhances the generalization ability of the model. This training scheme also contributes to the smoothing of the computed cerebellar output since the neighborhood-based activation process results in continuity of the network output. In addition, a Gaussian activation function is adopted to weight the activation level of each computing cell in the neighborhood in order to achieve the desired output accuracy. The neighborhood-based training scheme proposed in this work is inspired by the physiological properties of the cerebellar neural connectivity, where the existence of *microzones* (see Chapter 2 for details) in the human cerebellum signifies the simultaneous activation and adaptation of highly correlated Purkinje cells (Doya, 1999; Voogd and Glickstein, 1998).

Furthermore, the memory patching (C-Patch) and memory upsizing (C-Upsize) algorithms are proposed to resolve the issues related to the training of a CMAC-based system. The patching technique addresses the problem of insufficient training in a CMAC network by constructing a plausible memory surface for the untrained CMAC memory cells, while the upsizing technique reduces the CMAC training time by first training a small-sized CMAC network and then scaling it up to a large-sized CMAC to obtain a finer output resolution. The application of the upsizing algorithm also resolves the generalization-accuracy trade-off

in training the CMAC network as the network training is performed in two stages. More details will be presented in Chapter 5 of the Thesis.

The proposed approaches described above cumulate to three major theoretical research contributions as shown in Figure 1.1, which are summarized as follows:

- The Hierarchically Clustered Adaptive Quantization CMAC (HCAQ-CMAC).
The HCAQ-CMAC network (Teddy et al., 2007a) is a new adaptively-quantized cerebellar-based computational architecture that adopts a hierarchical clustering technique to non-uniformly allocate its computing cells. The memory allocation process in the HCAQ-CMAC network is designed to adapt to the observed characteristics of the training data such that more memory cells are allocated to model the data regions where rapid fluctuations of the output values are observed while fewer cells are used to characterize the data segments with relatively monotonic outputs. This memory allocation process is inspired by the two-stage physiological development process observed in the human brain, where the excessive neurons and synaptic connections created during the formation of the infant brain are gradually pruned and refined to form the precise wirings of the developed adult brain (Jessell, 1996; Shaw et al., 2006).
- The Pseudo Self-Evolving CMAC (PSECMAC) network.
The PSECMAC network (Teddy et al., 2007c) constitutes a self-organizing multi-resolution cerebellar associative memory architecture that employs a data-driven memory allocation scheme to adaptively quantize its computing cells. The adaptive memory allocation process in the PSECMAC network is inspired by the neurophysiological aspects of the cerebellar learning phenomenon, where it has been observed that significantly higher densities of the cerebellar synaptic connections are located at the frequently-accessed regions of the cerebellum that are activated by repeated learning episodes (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). This cerebellar-based experience-driven synaptic plasticity phenomenon is emulated in the PSECMAC network by employing a data-driven adaptive memory quantization scheme (i.e. MPSEC) for the derivation of its computing structure. Please refer to Chapter 4 for the details. In addition, the PSECMAC network extends from the HCAQ-CMAC network by adopting a neighborhood-based learning principle to address the generalization-accuracy dilemma encountered in the CMAC network.
- The Three-Stage CMAC Learning (TSL-CMAC) framework.

The TSL-CMAC framework (Teddy et al., 2006a, 2007d) is a paradigm for training a CMAC-based system and it consists of three stages: (1) the *initial learning*; (2) *network generalization*; and (3) *fine tuning* stages. The three training stages functionally correlates to the three stages of a psychological learning model suggested for the acquisition of human motor and cognitive skills (Fitts and Posner, 1967; Tomporowski, 2003). The proposed TSL-CMAC framework is built upon the patching and upsizing algorithms (i.e. C-Patch and C-Upsize) and sought to address the computational deficiencies associated with the training of a CMAC network.

Finally, the proposed cerebellar memory framework and the two resultant novel cerebellar-based computing architectures are applied to a wide range of characteristically different applications that consists of automated car driving, modeling of the glucose–insulin dynamics of the human metabolic process, option pricing and arbitrage trading, banking failure classification, automated speaker verification, personalized regulation of blood glucose for the treatment of Type-1 diabetes as well as the control of a hypothetical plant. Detailed analysis of the experimental results are presented in the respective chapters of the Thesis.

1.4 Organization of The Thesis

This Thesis is organized into six parts and consists of eight chapters. Chapter 1 constitutes Part 1 of the Thesis, which describes the problem statements and motivations behind the research as well as the organization of the Thesis.

Part 2 consists of Chapter 2 that presents a review of the research fields related to the reported work. This includes an overview of the neuro-physiological aspects together with a comparative study of the existing computational models of the human cerebellum, particularly the Cerebellar Model Articulation Controller (CMAC) network and its variants. In addition, a brief review of the soft computing techniques, specifically neural networks, fuzzy systems, neuro-fuzzy systems and data clustering techniques associated with the research on the proposed cerebellar associative memory framework is provided.

In Part 3 of the Thesis, the proposed neuro-physiologically-inspired cerebellar memory framework and the two resultant learning memory architectures are introduced. Chapter 3 presents the novel *Hierarchically Clustered Adaptive Quantization CMAC* (HCAQ-CMAC)

network and the theoretical proof of convergence of its learning process. The HCAQ-CMAC network is subsequently evaluated on two complex case-studies, namely: (1) automatic control of car maneuver; and (2) modeling the dynamics of the human glucose metabolic process. In Chapter 4, the newly-developed *Pseudo Self-Evolving CMAC* (PSECMAC) network is described together with a comprehensive proof of its learning convergence. The proposed PSECMAC network is applied to two real-life financial applications, namely: (1) option pricing and arbitrage trading; and (2) US banking failure classifications.

Part 4 investigates the associative learning process of the CMAC network. In Chapter 5, a three-stage computational paradigm for CMAC learning named the *Three-Stage Learning CMAC* (TSLCMAC) framework is proposed to resolve the deficiencies associated with the training of a CMAC-based system (see Section 1.3). The TSL-CMAC trained cerebellar model is subsequently applied to the control of a hypothetical second-ordered plant.

Part 5 describes the applications of the HCAQ-CMAC and PSECMAC cerebellar computational models to solve real-life problems. In Chapter 6, the two proposed networks are employed in a text dependent speaker verification system to automatically authenticate the purported identity of an unknown speaker. Following that, Chapter 7 presents a novel personalized approach to the treatment of Type-1 diabetes mellitus, where the HCAQ-CMAC and PSECMAC networks are employed as computational β -cells to capture the complex dynamics of the insulin responses of a healthy person in order to replicate the desired insulin profile in a diabetic subject to regulate his blood glucose levels.

Part 6 consists of Chapter 8 that concludes the research reported in this Thesis and highlights the research contributions as well as the possible directions for further investigations. The future works include the lateral and vertical extensions to the proposed cerebellar computational models and their application domains.

*"All truths are easy to understand once they are discovered;
 the point is to discover them"*
 –Galileo Galilei

Chapter 2

Literature Review

This chapter presents an overview of the human cerebellum, with a focus on the anatomical and physiological aspects of the cerebellum that inspired the work in this Thesis. The chapter also covers the fundamentals of the CMAC neural network, which is a well established computational model of the human cerebellum, and reviews some of the CMAC variants proposed in the literature. In addition, a brief overview on some of the neural/fuzzy architectures and clustering techniques related to the research in this Thesis is also provided.

2.1 Cerebellum and Human Procedural Memory System

The human procedural memory system is a facet of the brain's information computing capacity that represents a memory tract for the acquisition of skills and procedures (Eichenbaum, 2002). As the recall from this particular memory pathway is generally subconscious, the procedural memory system is also referred to as *implicit knowledge memory* (Squire et al., 1993). The human procedural memory system consists of the cerebellum and the striatum that is part of the basal ganglia formation (Eichenbaum, 2002). Due to its structural neuronal organization and anatomic simplicity, the cerebellum is one of the few brain constructs where the patterns of intrinsic synaptic connections are known in considerable details (Middleton and Strick, 1998a). The fact that the cerebellar cortex has a highly regularized anatomy has enabled brain researchers to derive a number of important neurophysiological relationships regarding the working mechanisms of the human cerebellum.

The human cerebellum, or *little brain* in Latin, is a brain construct that is important for a

Figure 2.1: An atlas of the human brain

number of motor and cognitive functions, including learning and memory (Middleton and Strick, 1998b; Thach, 1998). Although the cerebellum functions primarily as a movement regulator (Albus, 1981), there have been studies suggesting that it also plays an active role in purely cognitive tasks (Thach, 1996). Functional neuroimaging studies (Desmond and Fiez, 1998) have presented evidences for cerebellar involvements in the activation of the working memory, implicit and explicit learning and memory formation, as well as language processing. Further supports of the cognitive functions provided by the cerebellum come from lesion studies, where patients with a blocked posterior inferior cerebellar artery are observed to encounter difficulties in learning word association tasks (Kandel et al., 2000b).

This section presents the underlying cerebellar anatomical and physiological characteristics that facilitate and expedite the learning (knowledge acquisition) and memory (information retention) capabilities of the human cerebellum.

2.1.1 Memory Formation in the Cerebellum – The Anatomy of the Cerebellar Circuit

Located at the bottom rear-end of the brain (see Figure 2.1), the cerebellum constitutes only ten percent of the total volume of the brain but contains more than half of all its neurons. The most striking feature of the human cerebellum is the near-crystalline structure of its anatomical layout. However, despite its remarkably uniform anatomical structure, the

cerebellum is divided into several distinct regions. Each of these regions receives sensory information from different parts of the brain as well as the spinal cord and projects to different motor systems. Such physical connectivity suggests that different regions of the human cerebellum perform similar computational operations but on different sensory inputs (Kandel et al., 2000b). In order to effectively accomplish its motor regulatory functions, the cerebellum is provided with an extensive repertoire of information about the objectives (intentions), actions (motor commands) and outcomes (feedback signals) associated with a physical movement. There are three sets of extra cerebellar afferents: the *mossy* and the *climbing* fibers, both of which carry sensory inputs from the periphery as well as sets of motor commands-related information from the cerebral cortex; and a set of *mono-armigenic* and *cholinergic* afferents that is postulated to signal rewards (Tyrrell and Willshaw, 1992). The mossy fibers carry information originating from the spinal cord and the brainstem, while the climbing fibers originate from the inferior olivary in the medulla oblongata.

These cerebellar afferent inputs flow into the *granule cell* layer of the cerebellar cortex. The mossy fiber inputs, which carry both sensory afferent and cerebral efferent signals, are relayed by a massive number of granule cells. These granule cells work as expansion encoders by combining different mossy fiber inputs. Subsequently, each of the granule cells extends an ascending axon that rises up to the molecular layer of the cerebellar cortex as *parallel fiber*. These parallel fibers in turn serve as the inputs to the *Purkinje cells* at the cerebellar cortex. The Purkinje cells are the main computational units of the cerebellar cortex. Each Purkinje cell draws its inputs from the parallel and the climbing fibers. The parallel fiber inputs to the Purkinje cells provide large vectors of sensory and command-related information, while the climbing fibers are thought to function as training signals that regulate the modifications of the Purkinje cells' synaptic weights. The parallel fibers run perpendicularly to the flat fan-like dendritic arborization of the Purkinje cells, enabling the greatest possible number of parallel fibers and Purkinje cells contacts per unit volume. The Purkinje cells perform combinations of the synaptic inputs, and their axons carry the output of the cerebellar cortex downwards into the underlying white matter and subsequently to the *deep cerebellar nuclei*. The outputs of the deep cerebellar nuclei form the overall processed output of the cerebellum. Figure 2.2 depicts a diagram of the cerebellar circuitry.

In contrast to the massive synaptic connectivity of the parallel fibers to the Purkinje cells, each of the Purkinje neurons receives input from exactly *one* climbing fiber. There exists a *topographical mapping* in the synaptic connections between the cerebellar cortex, the deep

Figure 2.2: A diagram of the cerebellar circuitry: GC - Granule Cell; PC - Purkinje Cell; CN - Deep Cerebellar Nuclei; IO - Inferior Olive; • - inhibitory connection; ○ - excitatory connection (Adapted from (Doya, 1999))

Figure 2.3: Schematic diagram of the cerebellum (Adapted from (Albus, 1989))

cerebellar nuclei and the inferior olivary from which the climbing fibers originate (Doya, 1999; Voogd and Glickstein, 1998). This topological mapping results in a modular structure or clustering known as *microzones*, where each microzone involves approximately 3000 Purkinje cells. The Purkinje cells of a microzone is connected to one corresponding deep cerebellar nucleus. The olivocerebellar projections to the cerebellar cortex are also arranged in a similar manner: a sub-nuclei of the inferior olive projects to a microzone of Purkinje cells sharing the same target nucleus (Strata and Rossi, 1998). Therefore, the output of the cerebellar cortex is organized as a series of *discrete modules*, where each of them is provided with a private connection to the inferior olive (Voogd and Glickstein, 1998). Such an arrangement suggests a pattern of neighborhood activation of the Purkinje cells belonging to the same microzone during the computation of the processed output in the human cerebellum.

Memory formation in the cerebellum is facilitated by the storage of information embedded in its synaptic connections. The cerebellum corresponds to an associative memory system that performs a nonlinear mapping from the mossy fiber inputs to the Purkinje cells' outputs. This mapping is functionally depicted as Figure 2.3. The granule cell layer acts as an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells forms an array of modifiable synaptic weights of the cerebellar computing system. The Purkinje cell array subsequently forms the knowledge base of the cerebellum and the output of the cerebellar memory system is generated by integrating the contents of the activated Purkinje cells.

2.1.2 Learning in the Cerebellum – The Physiological Aspects of the Cerebellar Circuit

As the regulator of physiological movements, the cerebellum evaluates the disparities between the formulated intention and the executed action and subsequently adjusts the operations of the motor centers to affect and regulate the ongoing movement (Rapoport et al., 2000; Salman, 2002). Studies in neuroscience have established that the cerebellum performs an associative mapping from the input sensory afferent and cerebral efferent signals to the cerebellar output, which is subsequently transmitted back to the cerebral cortex and spinal cord through the thalamus (Albus, 1971, 1989; Houk et al., 1996; Ito, 1984; Marr, 1969). The physiological process of constructing an associative pattern map constitutes the underlying neuronal mechanism of learning in the human cerebellum.

Neuroscience research has established that the cerebellum adopts an error-correction-driven supervised learning paradigm (Kandel et al., 2000b). This implies that cerebellar learning requires extended trials with repeated exposures to similar sequence of movements in order to achieve a finely calibrated mapping between the intended and actual execution of the motor movements. The existence of microzones (Doya, 1999; Voogd and Glickstein, 1998), as well as the established role of the climbing fibers as the teaching signals to the Purkinje cells, suggest that the cerebellar circuitry performs neighborhood-based training of the synaptic weights. That is, the cerebellar input (motor commands and sensory signals) and output (corrective error signals) pairing of a learning episode alters the synaptic weights of a cluster of Purkinje cells that is topographically defined by the corresponding microzones.

This neighborhood-based learning mechanism enables a faster convergence of the cerebellar learning process, and underlies the generalization of skill learning in everyday life.

The cerebellar learning mechanism is facilitated by the modifiable synaptic transmissions (cerebellar synaptic plasticity) and the synaptic reorganization capability (cerebellar structural plasticity) of the neuronal connections. Research into the physiology of the human cerebellum has sufficiently demonstrated that the Long Term Depression (LTD) of the Purkinje cells' firing potentials in response to synaptic inputs from the parallel fibers is the underlying cellular mechanism responsible for cerebellar synaptic plasticity (Albus, 1971; Houk et al., 1996; Ito, 2000; Kandel et al., 2000b; Schutter, 1997; Tyrrell and Willshaw, 1992). Although both the mossy fibers and the climbing fibers carry sensory afferent and cerebral efferent signals, they respond differently to sensory stimulations and thus result in distinct responses of the Purkinje cells. The activation of the mossy fiber inputs of the Purkinje cells produces a steady stream of regular impulses called *simple spikes*. In contrast, the climbing fibers fire spontaneously at very low rates, creating a burst of *complex spikes* and these spontaneous firing rates are changed only modestly by sensory stimuli. The parallel fiber inputs to the Purkinje cells provide large vectors of sensory information, thereby transmitting a diverse array of signals. The climbing fibers, on the other hand, function as training signals that regulate the Purkinje cells' responses to specific patterns of their parallel fiber inputs by modifying the synaptic weights of the neuronal connections via the learning mechanism of LTD. The effect of such LTD modulations vary from minutes to hours, depending on the degree of depolarization and the quantity of calcium produced by the regulatory climbing fibers in the Purkinje cells' dendrites (Kandel et al., 2000b).

However, scientific studies have suggested that synaptic depression alone may not be adequate for forming permanent, long term memories of motor programs (Houk et al., 1996). Instead, there are evidences of morphological alterations of the cerebellar cortex following extensive cerebellar learning. These studies on the experience-driven cerebellar structural plasticity phenomenon have demonstrated that complex motor skill learning actually leads to an increase in the number of synapses within the cerebellar cortex (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). In such studies, rats were given acrobatic training by challenging them to acquire complex motor skills necessary to traverse a series of obstacles. It was discovered that the rats with such training developed an increased density of the parallel fibers to Purkinje cells synapses per unit volume. The increased synaptic density was facilitated by increased dendritic arborization and increased dendritic spine

densities along the Purkinje cells' spiny branchlets (Kleim et al., 1998b). Such an observation constitutes a learning-driven biological manifestation of the multi-resolution nature of the cerebellar circuitry. Similar plasticity characteristics were also observed along the olivocerebellar pathway. The olivocerebellar axons and climbing fibers are capable of remarkable structural plasticity that is regulated via their interactions with the Purkinje cells (Strata and Rossi, 1998). This neurobiological adaptation process underlies the formation of long term procedural memory at the cerebellum. These neurophysiological properties of cerebellar learning and memory formation subsequently provides the biological inspirations for the construction of the proposed cerebellar models presented in the later chapters.

Table 2.1: Various computational models of the human cerebellum

Marr's Model (Marr, 1969)	This model advocated that the human cerebellum functions as a pattern associator, where each output cell of the cerebellum controls an elemental movement of the body in response to the specific states in which the physical movement occurs. It is suggested that the cerebellum learns the subconscious execution of movements through pattern association between the mossy fiber inputs and the activations of the Purkinje cells. Marr's cerebellar model was subsequently implemented by Willshaw and his colleagues (Tyrrell and Willshaw, 1992) via the use of an associative net (Willshaw et al., 1969).
Albus' Model (Albus, 1971, 1981)	Albus' cerebellar model is similar to the one proposed by Marr. The cerebellum is modelled as an associative memory that implements locally-generalizing, nonlinear maps between the mossy fiber inputs and Purkinje cell outputs. The model was subsequently developed into a computational architecture named Cerebellar Model Articulation Controller (CMAC).
Adjustable Pattern Generator (Barto et al., 1995; Fagg et al., 1997)	The cerebellar model consists of a group of independent submodules, referred to as the Adjustable Pattern Generator (APG). Each APG is made up of a set of Purkinje cell modules and a single cerebellar nuclear cell module. An APG is then responsible for the learning of the appropriate state-dependent activation level of a subset of muscles that it controls.
Cerebellar Feedback Error Learning Model (Kawato and Gomi, 1992; Kawato et al., 1987)	This model contemplated that the human cerebellum functions as an <i>inverse model</i> of the limb control locomotive system. This inverse model performs the transformation of the desired movement trajectory of the limb to the motor commands required to attain the movement goal.
Spiking Cerebellar Model (Bezzi et al., 2004; Coenen et al., 2004)	This computational model consists of four types of interacting spiking neural networks that respectively model the granular layer, Purkinje layers, cerebellar nuclei, and inferior olives of the human cerebellar circuit.

2.2 Computational Models of The Human Cerebellum

The extensive knowledge gathered on the anatomical properties and the physiological characteristics of the human cerebellum has motivated the construction of various computational models that attempt to explain and emulate the working mechanisms of the human cerebellum. Table 2.1 outlines an overview of some of the well-known cerebellar computational models. Among all these models, Marr's and Albus' models are the first and most comprehensive mathematical models of the human cerebellum (Kanerva, 1993). Marr's theoretical model was subsequently implemented by Willshaw and his colleagues (Tyrrell and Willshaw, 1992) using their proposed associative net (Willshaw et al., 1969) structure. On the other hand, Albus developed a dedicated computing structure named CMAC to emulate the computations of the human cerebellum. To date, CMAC is the most widely used computational model of the human cerebellum. This is due to the computational simplicity and the straight forward structural design of the model, which enables CMAC to be readily adopted by the AI community to solve real-world engineering problems (Ang et al., 2002; Geng and Shen, 1997; Huang et al., 1997; Ku et al., 1998; Wahab et al., 2001; Yamamoto and Kaneda, 1999).

Due to its computations, CMAC manifests as an associative memory network (Miller et al., 1990). There are numerous types of associative memories that have been proposed in the literatures, such as the associative nets (Graham and Willshaw, 1997; Willshaw et al., 1969), auto-associative memory (Hopfield network) (Abu-Mostafa and St. Jaques, 1985; Chen and Amari, 2001; Hopfield, 1982), self-organizing map (SOM) (Kohonen, 1989; Oja et al., 2002), schema-associative memory (Botelho and Coelho, 1995), and bidirectional associative Memory (BAM) (Kosko, 1988; Wu and Pados, 2000) etc. In general, these memory models possess similar operational principles to the CMAC network; that is, they define a feedforward mapping function that associates a set of input patterns to a set of output patterns. However, these networks are essentially not developed based on the neurophysiological understanding of the human cerebellum, and their computational similarities to CMAC are purely coincidental as the primary information processing mechanism of the human cerebellum is also associative-based. These models, therefore, are not considered in this Thesis. The interested reader may refer to (Hassoun, 1993; Hinton and Anderson, 1989) for an in-depth study on the subject.

Figure 2.4: Schematic diagram of the CMAC neural network (Adapted from (Albus, 1989))

2.3 The CMAC Network Architecture

The Cerebellar Model Articulation Controller (CMAC) associative memory network is a well-established functional model of the human cerebellum (Albus, 1975a,b) that was constructed to explain the information-processing characteristics of its biological counterpart. This section presents the basic computational principles of the CMAC neural network.

2.3.1 Basics of CMAC Neural Network

The CMAC network functions as a static associative memory that models the non-linear mapping between the mossy fiber inputs and the Purkinje cell outputs. The massive mesh of granule cell encoders in the cerebellum corresponds to an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells form an array of modifiable synaptic weights that motivates the grid-like CMAC computing structure. In the human cerebellum, the outputs of the activated Purkinje cells are combined by the deep cerebellar nuclei to form the cerebellar output. In CMAC, the network output is computed by aggregating the memory contents of the active computing cells.

Figure 2.4 illustrates the functional characteristics of the CMAC network as formalized by Albus (Albus, 1989). Notice the close similarities to the computational mechanisms of the cerebellum as depicted in Figure 2.3. The CMAC network is essentially a multi-dimensional

Figure 2.5: An example of a 2D CMAC network (m_1^* refers to the set of quantization functions along the S_1 dimension and m_2^* refers to the set of quantization functions along the S_2 dimension)

memory array, in which an input vector acts as the address decoder to access the respective memory cells containing the adjustable weight parameters that constitute the corresponding output. CMAC learns the correct output response to each input vector by modifying the contents of the selected memory locations via learning by error-correction. For each input, the difference between the CMAC response and the known target response is computed and the weight values of the selected memory cells in the network are adjusted accordingly.

2.3.2 The CMAC Network Operations

As a functional model of the human cerebellum, CMAC operates based on the principle that similar inputs should produce similar outputs, while inputs that are dissimilar should invoke nearly independent outputs. As an associative memory, CMAC stores information locally and computes by employing a table-lookup operation, where the contents of the network are indexed by the inputs to the network. In a CMAC network, the network computing (memory) cells are uniformly partitioned to cover the entire input space. The inputs are then *quantized* to one of the discrete network cells to compute the indices to retrieve the network output. This process is referred to as the *uniform quantization* of the input space.

The computing structure of a CMAC network consists of layers of uniformly-quantized, multi-dimensional memory array (Albus, 1981). The CMAC computing structure is defined by a set of *quantization functions* that maps each input dimension to the corresponding physical memory axes of the network. The number of layers in a CMAC network is determined by the number of (user-defined) quantization functions. That is, each quantization function corresponds to one layer of memory array. This also implicitly means that all the input dimensions have the same number of quantization functions. Figure 2.5 depicts an example of a 2-input CMAC network with four quantization functions (and therefore four layers) in each of the input dimensions. The resultant two-dimensional grid in Figure 2.5 corresponds to the input space of the CMAC network that is used to learn the associative mapping patterns for 256 input–output vector combinations. The quantization functions of the network are defined as follows:

Along dimension S_1 :

$$\begin{aligned} S_1 Q_1 &\rightarrow \{A, B, C, D\} \\ S_1 Q_2 &\rightarrow \{E, F, G, H, I\} \\ S_1 Q_3 &\rightarrow \{J, K, L, M, N\} \\ S_1 Q_4 &\rightarrow \{O, P, Q, R, S\} \end{aligned} \quad (2.1)$$

Along dimension S_2 :

$$\begin{aligned} S_2 Q_1 &\rightarrow \{a, b, c, d\} \\ S_2 Q_2 &\rightarrow \{e, f, g, h, i\} \\ S_2 Q_3 &\rightarrow \{j, k, l, m, n\} \\ S_2 Q_4 &\rightarrow \{o, p, q, r, s\} \end{aligned} \quad (2.2)$$

where $^j Q_k$ denotes the quantization function for the k^{th} layer of the j^{th} dimension and $\{\cdot\}$ denotes the set of quantization levels. Each of the quantization levels (i.e. A, B, C, etc.) denotes a memory axis in the respective layer. The intersections of the memory axes of a layer constitute the memory locations for that layer. For instance, with respect to Figure 2.5, there are 16 memory cells corresponding to the pair of quantization functions $S_1 Q_1$ and $S_2 Q_1$ in layer 1. During the operations of the CMAC network, the input vector is first quantized into the corresponding quantization level at each layer, and this forms the address index used to access the memory location in that layer. Each input vector selects *one* memory cell from a layer. In the CMAC of Figure 2.5, the input vector of (6, 6) selects a total of four memory cells (one from each layer). That is, the C1, C2, C3 and C4 cells that correspond to the quantization points of Bb, Gg, Ll, and Pp respectively. The output of the CMAC network is subsequently computed by the linear combination of the memory contents (weights) of the selected cells.

The computation of the CMAC output is mathematically described as follows. Let J denote the total number of inputs to a CMAC network and $j = \{1 \cdots J\}$ be the index to the j^{th}

input dimension. Let $k = \{1 \cdots K\}$ be the index to the k^{th} layer of the CMAC network, where K is the total number of layers in the network; and jQ_k be the quantization function for the k^{th} layer of the j^{th} dimension. The output Y_s of the CMAC network to the s^{th} input $\mathbf{X}_s = [x_{s,1} \cdots x_{s,j} \cdots x_{s,J}]$ is computed as follows:

$$Y_s = \sum_{k=1}^K \left[{}^k W_{\bar{\mathbf{X}}_{k,s}} \right] \quad (2.3)$$

where ${}^k W_{\bar{\mathbf{X}}_{k,s}}$ denotes the weight of the CMAC memory location indexed by $\bar{\mathbf{X}}_{k,s}$ in the k^{th} layer of the CMAC network. The quantized index $\bar{\mathbf{X}}_{k,s}$ is computed as:

$$\bar{\mathbf{X}}_{k,s} = \left[{}^1\bar{Q}_k[x_{s,1}] \quad {}^2\bar{Q}_k[x_{s,2}] \quad \cdots \quad {}^j\bar{Q}_k[x_{s,j}] \quad \cdots \quad {}^J\bar{Q}_k[x_{s,J}] \right] \quad (2.4)$$

where ${}^j\bar{Q}_k[\cdot]$ denotes the quantization operation in the j^{th} dimension of the k^{th} layer that maps the input point $x_{s,j}$ onto one of the quantization levels in the jQ_k .

2.3.3 The CMAC Network Learning Process

CMAC adopts an error correction learning paradigm, where the correct input to output mapping patterns are acquired by comparing the computed outputs with the desired outcomes, and subsequently using the observed disparities to update the contents of the activated memory cells. Let $\hat{Y}_s$ denotes the expected output of the CMAC network for the s^{th} input $\mathbf{X}_s$. The CMAC network learning equation for the s^{th} training tuple $(\mathbf{X}_s, \hat{Y}_s)$ at the $(i)^{th}$ training iteration is as follows:

$${}^k W_{\bar{\mathbf{X}}_{k,s}}^{(i+1)} = {}^k W_{\bar{\mathbf{X}}_{k,s}}^{(i)} + \alpha E_s^{(i)}, \quad \forall k \in \{1 \cdots K\} \quad (2.5)$$

$$E_s^{(i)} = \hat{Y}_s - Y_s^{(i)} \quad (2.6)$$

where ${}^k W_{\bar{\mathbf{X}}_{k,s}}^{(i)}$ denotes the weight of the CMAC memory location indexed by $\bar{\mathbf{X}}_{k,s}$ in the k^{th} layer of the CMAC network at the $(i)^{th}$ training iteration, $\bar{\mathbf{X}}_{k,s}$ is the quantized index of the input $\mathbf{X}_s$ for the k^{th} layer, $Y_s^{(i)}$ is the computed CMAC output at the $(i)^{th}$ training iteration, α is the learning constant, and $E_s^{(i)}$ denotes the training error of the CMAC network for the s^{th} training tuple at the $(i)^{th}$ iteration.

The stability of the CMAC learning process has been extensively studied over the years. The

theoretical proof of convergence for the CMAC learning process has been established in (He et al., 2001; Lin and Chiang, 1997; Wong and Sideris, 1992). Essentially, a CMAC learning process is guaranteed to converge to a limit cycle, provided that the learning constant α is within the range of $0 < \alpha < 2$ (Lin and Chiang, 1997).

2.3.4 Strengths and Weaknesses of The CMAC Network

The structure and operating principles of the CMAC network results in various advantages such as simple computation, fast training, local generalization and ease of hardware implementation. The CMAC network is essentially a multi-dimensional memory array which employs a table lookup access of its memory cells. Such an operational characteristic allows for fast and efficient computations, which subsequently motivate the prevalent usage of CMAC in real-time applications. In addition, due to its associative memory structure, the CMAC network possesses a *localized learning* characteristic, whereby only a localized region of input-output (I/O) mappings (i.e. weight values) are adjusted during each training instance. This provides CMAC with robustness against learning interferences.

The layered structure of the CMAC computing cells has also equipped CMAC network with a good *generalization* ability. Each input vector to the CMAC network activates a set of adjacent computing (memory) cells in the I/O mapping space, whose contents are aggregated to form the network output. In this Thesis, this activation principle is referred to as the *neighborhood activation* of the CMAC computing cells. As such, a CMAC network does not require training at all points on its I/O mapping space before it is able to generate a satisfactory output. CMAC can respond adequately to an unlearned set of inputs as long as there has been training in the neighboring I/O mapping space. Such a generalization ability also contributes to the smoothing of the computed CMAC output. In addition, the neighborhood activation of the CMAC computing cells implies that a set of memory cells is updated during each training iteration. This in turn facilitates a faster convergence of the network learning process.

The advantages of the CMAC network have motivated the prevalent use of CMAC-based systems for real-time system control and optimizations, modeling and control of robotic manipulators, as well as various signal processing and pattern recognition tasks. Table 2.2 lists some of the applications of the CMAC network.

Table 2.2: Application areas of the CMAC network

System Control	<ul style="list-style-type: none"> • In (Ang et al., 2002), CMAC is employed in a Continuous Variable Transmission (CVT) control system. • (Yamamoto and Kaneda, 1999) used multiple CMAC networks for intelligent control of process system. • In (Ku et al., 1998), (Larsen et al., 1995) and (Cetinkunt and Donmez, 1993), CMAC is used as a servo controller for high-precision machineries. • (Lin and Hyongsuk, 1991) investigated the use of the CMAC network in adaptive critic self-learning control.
Robotics	<ul style="list-style-type: none"> • In (Kim and Lewis, 2000), (Commuri et al., 1997), and (Kano and Takayama, 1990), CMAC is employed for control of robot manipulators. • (Miller, 1989) and (III, 1987) implemented a sensor-based control of robots with vision using the CMAC network. • In (Hu et al., 1999) and (Hu and Pratt, 1999), a stable adaptive control of a bipedal walking robot is implemented using CMAC networks.
Pattern Classification	<ul style="list-style-type: none"> • In (Huang et al., 1997), CMAC is applied for the color scanner to printer calibration. • (Ker et al., 1997) investigates color reproduction using a fuzzy CMAC network. • (Xu et al., 2004) applied CMAC networks for medical image classification. • (Geng and Shen, 1997) proposed a fingerprint classification system using fuzzy CMAC networks.
Signal Processing	<ul style="list-style-type: none"> • In (Wahab et al., 2001), CMAC is employed for noise cancellation. • (Iiguni, 1996) presented an image coding approach using CMAC. • (Glanz and Miller, 1988) developed a signal deconvolution technique via the CMAC neural network.

However, despite the numerous advantages of the CMAC network, there are several drawbacks associated with the organization of the CMAC computing structure. They are:

1. Curse of dimensionality

Due to the table lookup computing mechanism of the CMAC network, the network size increases exponentially with the increase in the number of input variables.

2. Constant output resolution associated with the entire input space

CMAC employs a uniform partitioning of its available memory cells to cover the entire

I/O mapping space. The uniform quantization of each input to define the receptive fields of the CMAC network results in a static output resolution associated with the entire input range. The accuracy of the CMAC output thus depends on the network size.

3. Generalization–accuracy dilemma

Due to the uniformity in the CMAC memory allocation process, there is a trade-off between the generalization capability and the modeling fidelity of the CMAC network. That is, a small-sized CMAC network is able to better generalize the characteristics of the training data, while a large-sized CMAC network produces more accurate outputs.

4. Comprehensibility of the network computation

The layered structure of the CMAC computing cells often leads to limited comprehensibility of the network computation.

The above-mentioned limitations can be attributed to the uniform quantization of the CMAC input space and the subsequent linear mapping of these partitioned subspaces as the CMAC network cells. The fact that the CMAC input space to network cells mapping process is not tailored to the characteristics of the problem at hand subsequently leads to suboptimal cell allocation and poor memory utilization of the CMAC network. In the current literature, there is a number of previous attempts to resolve the uniform quantization issue in the CMAC network. These attempts are briefly discussed in the next section.

In addition, there are several shortcomings associated with the localized learning characteristics of the CMAC network. Due to the fact that only a localized region of the I/O mapping space is adapted during each training instance, it is difficult to fully train a CMAC network. A comprehensive and meticulous planning is required to design a training profile to ensure all the CMAC memory cells are trained. Such a deliberate planning is not always feasible, especially in cases involving a highly complex, nonlinear and high-dimensional problem. The localized learning principle also implies that the time required to (fully) train a CMAC network increases exponentially with the network size. Finally, the learning performance of the CMAC network is saddled by the generalization–accuracy dilemma arising from the localized generalization property of the network.

Figure 2.6: A classification of CMAC variants in the literature

2.4 The CMAC Variants

The importance of an efficient memory allocation scheme to the performance of CMAC is evidenced by the number of CMAC variants proposed to address the issue in the literature. Figure 2.6 depicts a topology of these CMAC variants based on the approach employed to resolve the uniform-quantization problem. Generally, the solutions offered by the CMAC variants are of two main approaches: (1) multi-resolution based discrete quantization; and (2) fuzzy quantization of the input space. The motivation for the former is to derive an efficient mapping of the memory cells onto the input-output associative space to enhance memory usage via increasingly finer output resolutions without inducing high computational complexity of the resultant network. The fuzzified CMAC variants, on the other hand, attempt to address the rigidity of the crisp boundaries and the problem of a constant output resolution imposed by employing discrete quantization functions with the use of fuzzy membership for the quantization process. In addition, the mapping of a formalized reasoning process improves the computational interpretability of the opaque CMAC network.

Table 2.3 presents a brief review of the various fuzzified CMAC (FCMAC) architectures proposed in the literature. The FCMAC network structures are essentially constructed by incorporating fuzzy membership functions as their quantization functions. This attempts to minimize the quantization error introduced by the crisp quantization boundaries in CMAC. A comprehensible computational process is achieved by the integration of various fuzzy inference schemes into the hybrid system. While these variants offered the computational interpretability lacking in the black-box CMAC model, there is a significant increase in the computational complexity of the hybrid network. This increase in the computational complexity can be attributed to the fuzzification and the defuzzification techniques, as well as the computations of the inference scheme adopted by the hybrid network.

Table 2.3: Existing FCMAC Architectures

Model	Receptive Field	Quantization	Rule Base Model	Inference Scheme	Weight Stored	Update Rule	Defuzzification Method
CMAC-fuzzy (Ozawa et al., 1992)	Triangular membership function	Uniform quantization	TSK	Not mentioned	Real numbers	Direct error update on weight	Not applicable
Fuzzified CMAC (Nie and Linkens, 1993)	Calculate matching degree by neighborhood	Not applicable	Mamdani	Not mentioned	Triangular membership function	Direct error update on weight	Centre of gravity
Fuzzy CMAC (Zhang and Qian, 2000)	Not applicable	Uniform quantization	Not mentioned	Not mentioned	Gaussian function	Gradient descent update for centre and variance of the Gaussian function	Not applicable
Fuzzy CMAC (Jou, 1992)	Bell-shaped membership function	Uniform quantization	Mamdani	CRI	No preference	Direct error update on weight	Centroid
CBLFC (Kim, 2002)	Bell-shaped membership function	Uniform quantization	Mamdani	CRI	Bell-shaped membership function	Generalized delta and chain-rules of differentiations for centre and width of the membership function	Centre of gravity
BMAC (Lane et al., 1992)	B-Spline function	Uniform quantization	Not mentioned	Not mentioned	Real number	Direct error update on weight	Not applicable
Fuzzy CMAC (Ker et al., 1997)	Bell-shaped membership function	Uniform quantization	Mamdani	Not mentioned	Bell-shaped membership function	Maximum gradient method to adjust the mean for bell shaped membership function	Centre of area
FCMAC-CRI (Jiang et al., 2007)	Trapezoidal membership function	Fuzzy quantization	Mamdani	CRI	Trapezoidal membership function	Direct error update on weight	Centre of area
FCMAC-TSK (Ting and Quek, 2006)	Trapezoidal membership function	Fuzzy quantization	TSK	TSK	Real number	Direct error update on weight	Not applicable

continued on next page

continued from previous page

Model	Receptive Field	Quantization	Rule Base Model	Inference Scheme	Weight Stored	Update Rule	Defuzzification Method
FCMAC-Yager (Sim et al., 2006)	Trapezoidal membership function	Fuzzy quantization	Mamdani	Yager	Trapezoidal membership function	Direct error update on weight	Modified centre of averaging
FCMAC-AARS (Quek and Guo, 2007)	Trapezoidal membership function	Fuzzy quantization	Mamdani	AARS	Real number or trapezoidal membership function	Direct error update on weight	Modified centre of area
FCMAC-TVR (Quek et al., 2006)	Trapezoidal membership function	Fuzzy quantization	TSK	TVR	Real number	Direct error update on weight	Not applicable

Table 2.4: Existing Discrete CMAC Architectures with Adaptive Quantization

Model	Adaptive Quantization Method	Number of Cells	Update Rule	Neighborhood Training	Output Retrieval
Multi-layer CMAC (Moody, 1989)	Layered CMACs with different resolutions	Variable	Squared error minimization	Yes	Neighbor-hood function with graded response
Tree-based Multi-layer CMAC (Menozzi and Chow, 1997)	Tree-based extension of Multi-layered CMAC	Variable	Normalized gradient descent	Not Mentioned	Summed output from each layer
SO-HCMAC (Lee et al., 2003)	Golden-section search and Shannon's entropy measure	Variable	Differentiable gradient descent	Not Mentioned	Summation of output with Gaussian basis function
Adaptive Quantization CMAC (Yeh and Lu, 2002)	Grey relational analysis	Fixed	Error correction learning	Not Applicable	Summation of selected weights
Adaptive Resolution CMAC (Kim and Lin, 1992)	Target derivative analysis to determine quantization decision function	Fixed	Error correction learning	Not Mentioned	Not Mentioned
Adaptive Quantization CMAC (Gao et al., 1996)	Target derivative-based competitive learning using Self-Organizing Map	Fixed	Error correction learning	Not Applicable	Summation of selected weight

The crisp CMAC variants, on the other hand, can be classified into three subgroups: i.e. multi-layered multi-resolution CMACs, CMACs with mathematically-induced quantization decision functions, and CMACs with information entropy-based quantization decision functions. Table 2.4 briefly outlines a summary of some of the crisp CMAC variants reported in the literature. The multi-layered multi-resolution CMACs utilize a hierarchy of CMAC networks of different modeling resolutions to achieve a variable quantization of the input space. From a coarsely defined (small) CMAC, additional layers of CMACs with higher modeling resolutions are created when the modeling error is deemed too large. While this approach can effectively capture the general trends as well as the minute details of the target function to be learnt, the total number of memory cells required for the system cannot be computed in advance. In most multi-layered multi-resolution CMAC systems, much more memory cells are required to achieve the same accuracy level as compared to the uniformly-quantized CMAC network. The CMACs with mathematically-induced adaptive quantization decision functions, on the other hand, generally assumed prior knowledge of the derivatives of the target function to be learnt. As a result, these CMAC variants have not been extensively evaluated with real-life applications. The third subgroup of CMAC variants employs information entropy-based approaches to compute the optimal quantization of the input space using the information measures derived from the training data. However, these CMAC systems are computationally more suited for pattern classification problems, in which information entropy is readily measurable.

2.5 Soft Computing Techniques

As described earlier, CMAC is a neural network model of the human cerebellum, while the fuzzy CMAC variants are essentially soft computing models constructed from the integration of the CMAC neural network and fuzzy system which employs a fuzzy inference scheme. Therefore, this section presents a brief overview of the current popular soft computing techniques, in particular the neural networks, neural fuzzy systems and various clustering algorithms related to the research in the subsequent chapters.

2.5.1 Artificial Neural Networks

Artificial Neural Networks (ANN) (Grossberg, 1988; Hornik et al., 1989; Specht, 1990; Wang, 1994; Xia and Wang, 2000) refers to a class of computing systems that are loosely modeled on the architecture of the human brain, with the primary objective of imitating the human learning capabilities. It consists of a large number of highly interconnected processing units termed *artificial neurons*, which employ various mathematical functions to emulate the operations of the biological neurons. The connections of these neurons (i.e. weights) in the neural network model hold the knowledge of the system. As a computational model of the brain's synaptic connectivity, ANN is therefore able to learn and model the complex relationships (pattern mappings) between the inputs and outputs from a set of training samples by adjusting its weight parameters, and to subsequently generalize the acquired knowledge to previously unseen samples (Tung, 2003). For this reason, ANNs are good at tasks such as pattern matching and classification (Carpenter and Grossberg, 1991; Lippmann, 1989), function approximation (Chung and Scarselli, 1998; Park and Sandberg, 1991), optimization (Hu and Wang, 2006; Liang and Wang, 2000), vector quantization (VQ) (Krishnamurthy et al., 1990; Martinetz et al., 1993), and data clustering (Gabrys and Bargiela, 2000; Herrero et al., 2001).

The research on ANN (Lin and Lee, 1996) is dated back to 1940, with the motivation to emulate the human intelligence and implement such synthetic intelligence in a new generation of "thinking" machines and intelligent systems driving a myriad of productive efforts. Due to the intense interest in this area, many forms of learning algorithms and network architectures have been proposed in the literatures over the years. Essentially, the various ANN models differ in three aspects, namely: (1) the computing (i.e. aggregation and activation) functions of the neurons; (2) the network architecture; and (3) the learning (training) algorithm. Some of the major development milestones in ANN research are listed as Appendix A of the Thesis. Please refer to (Haykin, 1994) for a detailed introduction to neural network models.

2.5.2 Fuzzy Systems and Hybrid Neural-Fuzzy Systems

Fuzzy system (Huang and Shen, 2006; Kosko, 1994; Marin-Blazquez and Shen, 2002) is a computational paradigm that employs linguistic IF-THEN fuzzy rules to model a process

or a problem that are too complex to describe using precise mathematical models. In such cases, fuzzy systems employ fuzzy sets and the associated fuzzy rules to emulate the decisions or reasoning of a human expert. Fuzzy logic (Zadeh, 1965) constitutes the basis of the fuzzy systems, where it provides a mathematical framework to compute the uncertainties associated with the human cognitive processes such as thinking and reasoning. In fuzzy systems, these uncertainties and vague concepts (e.g. those associated with the concept of "tall" or "heavy") are represented by fuzzy sets.

Fuzzy systems possess the ability to model the dynamics of a problem domain using high-level IF-THEN fuzzy rules that are intuitive to the understanding of the human user. This enables the user to easily comprehend and learn the knowledge extracted from a fuzzy system. However, traditional fuzzy systems such as (Mamdani, 1977; Sugeno and Murakami, 1984, 1985; Sugeno and Nishida, 1985) are limited by the manual formulation and tuning processes of their fuzzy rule-bases. The fuzzy rules are usually derived through interviews with experts and they may be inaccurate and biased as opinions may differ with different experts. Such a manual crafting of the fuzzy rule-base may therefore be too difficult or impossible to perform for a complex problem.

Neural Fuzzy Systems (NFS) (Harris et al., 2002; Kasabov, 2003; Mitra and Hayashi, 2000; Nauck et al., 1997) are the realizations of the functionality of fuzzy systems using neural techniques, which aims to resolve the limitations in the construction of a fuzzy system. In an NFS, the operations of the fuzzy system are fused and realized by the neural structure and the functionalities of the processing elements or nodes. When provided with numerical training data, the hybrid network can generalize from the training data, learn/tune the parameters and formulate the fuzzy rules to create a linguistic model of the problem. On the other hand, the connectionist structure of an NFS becomes transparent and the network weights can be interpreted by means of the IF-THEN fuzzy rules. There are three different types of NFS according to the type of IF-THEN fuzzy rules formulated, namely: (1) Tsukamoto (Tsukamoto, 1979); (2) Mamdani (Lee, 1990); and (3) Takagi-Sugeno-Kang (TSK) (Takagi and Sugeno, 1985) based systems. Amongst the three, Mamdani-based neural fuzzy systems possess the highest degree of transparency and interpretability of the fuzzy rule-bases. This is due to the fact that semantic meanings can be attached to the output fuzzy sets of the fuzzy rules. A detailed review on the three types of fuzzy systems can be found in (Tung, 2003).

2.5.3 Clustering Techniques

Clustering is a process to partition a given data set into different classes so that similar data points are grouped together within the same class. Cluster analysis, therefore, attempts to discover the inherent organizational structure in a set of data points. Existing clustering techniques proposed in the literature can be categorized into two main groups: (1) prototype-based; and (2) hierarchical-based clustering techniques. In prototype-based clustering techniques, the data space is grouped into k classes with the value of k known a priori. The k -means (Forgy, 1965; MacQueen, 1967) and Linear Vector Quantization (LVQ) (Kohonen, 1989, 1982) algorithms are classic examples of prototype-based clustering techniques. More recent research efforts on prototype-based clustering techniques (Lee, 1998; Zahid et al., 1999) are concentrated on the prediction/estimation of the number of clusters k before embarking on the clustering process.

In hierarchical-based clustering (Ward, 1963), a data set of N data points is repeatedly partitioned into finer regions or clusters based on some objective functions and/or criteria. This results in a hierarchy (or inverted tree) of clusters being formed. At each level, if a cluster is not the leaf node of the partitioning tree, then the cluster is further partitioned into smaller clusters. Single link (Sneath and Sokal, 1973), complete link (Johnson, 1967; King, 1967) and the algorithms proposed in (Bajcsy and Ahuja, 1998; Basak and Krishnapuram, 2005; Castro et al., 2004; Geva, 1999), are examples of hierarchical-based clustering techniques. The main drawback of hierarchical clustering is that it is static, and points committed to a given cluster in the early stages cannot be moved to a different cluster. This is in contrast to the prototype-based clustering techniques, where the data points can be redistributed under varying conditions. Please refer to (Baraldi and Blonda, 1998; Fasulo, 1999; Jain et al., 1999) for the comprehensive reviews on the various clustering algorithms in the literatures.

2.6 Summary

This chapter covers the overview of the anatomical and physiological properties of the human cerebellum. The cerebellum is a part of the human brain that is important for motor control as well as various cognitive tasks. Neurobiological and neurophysiological research

has provided an insight into the cerebellum's adaptive mechanisms to learning and motor skill acquisition. The cerebellar learning mechanism is facilitated by the synaptic plasticity and the structural plasticity of its neuronal connections. Two important conclusions can be derived from these studies: (1) cerebellar learning leads to an enduring functional adaptation of the cerebellar cortex; and (2) the acquisition of learning experiences alters the neuronal connectionist structure of the cerebellum. Such experience-driven plasticity constitutes a part of the neurobiological substrates underlying the formation of long term procedural memory in the human cerebellum and provides the neurophysiological inspirations to the development of the cerebellar memory models proposed in this Thesis.

This chapter also presents the basics of the CMAC network. The CMAC network is a well-established computational model of the human cerebellum. The associative memory structure of the CMAC network allows for numerous advantages that motivated the prevalent usage of CMAC in real-life applications. However, the uniform quantization of the memory cells in the CMAC network often leads to suboptimal memory allocation, poor memory utilization, and the generalization-accuracy dilemma. Attempts to address these limitations can be broadly categorized into two main approaches: the multi-resolution discrete CMAC approach and the fuzzified CMAC implementations. However, the success of these attempts in addressing the above-mentioned CMAC problems has been limited.

In addition, a brief overview on neural networks, fuzzy systems and clustering technologies has also been presented to complement the analysis on brain-inspired computing. The next two chapters present the novel neurophysiologically-inspired cerebellar associative memory models proposed to effectively and efficiently address the computational and learning limitations of the basic CMAC network.

"Everything should be made as simple as possible,
but not simpler"
–Albert Einstein

Chapter 3

The HCAQ-CMAC Architecture

CMAC is an associative memory network that computes by employing table-lookup operations, in which the memory contents are indexed by the inputs to the network. The basic CMAC network employs a uniform quantization of its input-output (I/O) mapping space, where the network computing (memory) cells are equally divided to cover the whole input space. During its operation, each input vector to the CMAC network is *quantized* to one of the discrete network cells to compute the indices to retrieve the network output.

As described in Section 1.2.1, there are three major drawbacks associated with the uniform quantization scheme of the CMAC network. They are, namely: (1) a constant output resolution associated with the entire input space; (2) poor memory utilization; and (3) the generalization–accuracy dilemma. It is therefore judicious to devise an efficient memory allocation scheme for the CMAC network to assign more memory cells to the input regions that require higher output resolution to: (1) enhance the CMAC memory utilization rate and (2) to provide more accurate outputs with a reasonable degree of data generalization.

One approach is to non-uniformly quantize the input space. For existing non-uniformly quantized CMAC systems, there is a trade-off between memory efficiency and computational complexity. In this chapter, a non-uniformly quantized CMAC network based on hierarchical clustering is proposed. The resultant architecture is named the *Hierarchically Clustered Adaptive Quantization CMAC* (HCAQ-CMAC) network. The objective of the HCAQ-CMAC network is to formulate a memory allocation procedure to enhance the storage efficiency, as well as to alleviate the generalization–accuracy dilemma of the CMAC network. The proposed network is inspired by the neurophysiology of the human brain,

where the excess neurons and connections of the infant brain are gradually pruned and refined to form the precise wirings of the adult brain (Shaw et al., 2006). In addition, the stability of the HCAQ-CMAC network is theoretically guaranteed by the proof of its learning convergence. The learning process of the proposed HCAQ-CMAC network always converges when the learning rate is within a theoretical range.

The rest of this chapter is organized as follows. Section 3.1 presents the neurophysiological motivations that inspire the development of the HCAQ-CMAC network. Section 3.2 details the structural learning and parameter tuning processes of the proposed HCAQ-CMAC network. In Section 3.3, the HCAQ-CMAC neural correlates are presented. The proof of learning convergence of the HCAQ-CMAC network is established in Section 3.4. Section 3.5 evaluates the performances of the proposed HCAQ-CMAC network in two real-life case-studies, namely: (1) automatic control of car maneuver and (2) modeling the dynamics of the human glucose metabolic process. Section 3.6 summarizes this chapter.

3.1 HCAQ-CMAC's Neurophysiological Motivations

Each of the diverse functions performed by the mature human brain, including the cerebellum, depends on the *precise* interconnections of its neurons that are formed through two stages of development, namely: (1) the embryonic, and (2) the postnatal development stage. The *embryonic development* of the brain system involves the *genesis* of an overabundance of neurons and synaptic connections, followed by the programmed death of superfluous cells (Jessell, 1996). The synaptic elimination process involves *competition* among the neuronal cells (Nicholls et al., 2001), and as many as half of the initially generated neuronal connections are eliminated during the embryonic development stage. At the same time, the surviving neurons grow to be more complex and take over the functions of the eliminated ones.

This initial coarse pattern of neuronal connections is further refined during the *postnatal development* stage. In this stage, sensory information plays a critical role in strengthening and eliminating synaptic connections through a competitive process. Findings from neuroscience research (Kaas et al., 1979, 1981) have shown that learning as well as exposures to various stimuli greatly affect the patterns of the internal connections of the brain. This observation emphasizes the notion that the structural organizations and the neuronal mappings

of the various brain regions are subjected to constant modifications based on adaptations (e.g. learning) and autobiographic experiences. Experiences, however, function more than merely altering the synaptic connections. There are also evidences of an on-going *synaptic reallocation* process (Hubel and Weisel, 1962), where neural cells are competitively shared by the brain's diverse functions.

With respect to the human cerebellum, there are evidences supporting the role of *experience-dependent plasticity* in cerebellar learning and memory formation. Essentially, two types of experience-dependent plasticity are observed in the cerebellum, namely: *synaptic plasticity* and *structural plasticity*. Synaptic plasticity refers to the modifiable synaptic strengths of the cerebellar circuitry that are achieved through the *Long Term Depression* (LTD) mechanism of the cerebellar learning process (Houk et al., 1996; Ito, 2000; Schutter, 1997). LTD alone, however, is not adequate for forming long term memories of procedural skills. Structural plasticity, on the other hand, refers to the alteration of the morphological structure of the neuronal interconnections in the cerebellum. Cerebellar structural plasticity studies have demonstrated that complex motor skill learning actually leads to an increase in the number of synapses within the cerebellar cortex (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). It is primarily responsible for the formation of persistent and long-lasting memory traces in the human cerebellum. Currently, findings from physiological and biological brain studies have converged on the notion of experience being the key driving factor responsible for the specialization of the neuronal interconnectivity patterns in the human cerebellum.

3.2 The HCAQ-CMAC Network

The two stages in the development of the human brain system outlined in the previous section constitute a process of selective allocation of neurons in the human brain. The embryonic stage of this process encompasses the formation of the basic architecture of the brain system, in which coarse connection patterns emerged as a result of the genesis and death of the brain cells during prenatal development. Subsequently, in the postnatal development stage, the initial architecture is refined and extraneous synaptic connections are pruned throughout an individual's life-span by repeated exposures to various activity-dependent experiences. These processes are incorporated into the HCAQ-CMAC memory allocation procedure as a mechanism for non-uniform quantization. In HCAQ-CMAC, the available

memory cells are distributed on the basis of the observed characteristics of the training data, which is defined by the data distribution in the input space as well as the variation of the target output value. For this reason, the HCAQ-CMAC architecture presented in this chapter is a Multi-Input-Single-Output (MISO) CMAC variant. This is because in a multi-output domain, the variation of each output variable may not be correlated to one another. Thus it is difficult to formulate a quantization function that is optimal for all output dimensions. A Multi-Input-Multi-Output (MIMO) problem can instead be modelled by a combination of several MISO HCAQ-CMAC networks.

Unlike the existing non-uniform quantization approaches outlined in Section 2.4 of Chapter 2, HCAQ-CMAC allocates the number of available memory cells based on the characteristics/information contents of the input-output mappings of the training data. The neuronal competition for survival in the human brain is emulated by the hierarchical clustering technique, where similar data points are clustered together to eliminate redundancy in data representation. This similarity is measured by the observed degree of variation of the target output. In the HCAQ-CMAC network, memory efficiency is achieved by allocating more memory cells to the input regions where rapid fluctuations of the output values are observed, and less memory cells to regions with relatively monotonic output values. As a result, a finer quantization level is obtained for the regions of the input space which contain more information.

In addition, unlike the basic CMAC architecture proposed by Albus (Albus, 1975a), the HCAQ-CMAC network presented in this chapter employs *one* layer of network cells. This is because the optimization of the memory allocation process of a multi-layered, multi-input CMAC neural network is not only tedious, but can also be computationally expensive, especially for high-dimensional input problems with a large number of quantization functions. Moreover, the multi-layered computing structure also limits the interpretability of the CMAC network. Therefore in this Thesis, a single-layer model of the CMAC network is proposed. Such a single layer implementation maintained the computational principles of its multi-layered counterpart via neighborhood activation of its computing cells. Please refer to Appendix B for the details on the single-layered model of the CMAC network.

To increase the accuracy of its computed output, the HCAQ-CMAC network adopts a winner-take-all computation, where each input vector to the network activates/updates exactly *one* computing cells. The non-uniform quantization process of the HCAQ-CMAC

Figure 3.1: Comparison of CMAC and HCAQ-CMAC memory structure for a given input-output characteristic surface

network is illustrated in Figure 3.1. Figure 3.1(a) shows an example of a target output surface of a 2-input problem to be modelled. Figure 3.1(b) depicts the corresponding uniformly-quantized single-layered CMAC memory structure, and Figure 3.1(c) illustrates the non-uniform HCAQ-CMAC for the same target output surface.

3.2.1 The HCAQ-CMAC Structural Learning Process

In the proposed HCAQ-CMAC network, the non-uniform quantization process and the subsequent memory allocation procedure is performed on a per-dimension basis. There are two stages to the training of the HCAQ-CMAC network: (1) structural learning, and (2)

parameter tuning. The purpose of the structural learning stage is to define the quantization function at each of the input dimensions, while the parameter tuning stage is employed to learn the memory contents of the HCAQ-CMAC network. The agglomerative hierarchical clustering technique (Ward, 1963) is employed to identify the optimal quantization decision function for each of the input dimensions. A *quantization cluster* is defined as the span of a memory quantization level in a given input dimension. Starting with an initial set of quantization clusters, the two clusters with the smallest merging cost in an input dimension are combined in each iteration of the hierarchical clustering process until the number of quantization clusters equals the number of available (predefined) memory cells for that input dimension (see Figure 3.2).

Let J denote the total number of input dimensions for a given problem. Assume that a training dataset of $\mathbf{U} = \{(\mathbf{X}_1, \hat{Y}_1), (\mathbf{X}_2, \hat{Y}_2), \dots, (\mathbf{X}_s, \hat{Y}_s), \dots, (\mathbf{X}_S, \hat{Y}_S)\}$ is used to train the HCAQ-CMAC network, where $\mathbf{X}_s = [x_{s,1} \ x_{s,2} \ \dots \ x_{s,J}]^T$ denotes the s^{th} input vector to the network, and $\hat{Y}_s$ denotes the expected scalar output of HCAQ-CMAC. Let $\hat{M}$ denote the total number of available memory cells per dimension, and $\mathbf{Q}_j^{(i)} = \{Q_{j,1}^{(i)}, Q_{j,2}^{(i)}, \dots, Q_{j,m}^{(i)}, \dots, Q_{j,M_j}^{(i)}\}$ denote a set of M_j quantization clusters in the j^{th} input dimension at the i^{th} iteration of the hierarchical clustering process. The HCAQ-CMAC memory allocation process is described as follows:

Step 1 *Perform data preparation.*

For each input dimension $j \in \{1 \dots J\}$, the input training samples $x_{s,j}$'s are sorted in ascending order such that $\{x_{1,j}, x_{2,j}, \dots, x_{s,j}, \dots, x_{S,j}\}$, where $x_{s,j} \leq x_{s+1,j}$.

Step 2 *Define the initial set of quantization clusters.*

Let $Q_{j,m}^{(i)}$ denote the m^{th} quantization cluster in the j^{th} input dimension at the i^{th} iteration of the hierarchical clustering process, and $n_{Q_{j,m}^{(i)}}$ as the number of data points in $Q_{j,m}^{(i)}$. Then, the quantization cluster $Q_{j,m}^{(i)}$ is defined as the set of input-output data pairs such that $Q_{j,m}^{(i)} = \{(x_{j,m-1}^{(i)}, y_{j,m-1}^{(i)}), (x_{j,m-2}^{(i)}, y_{j,m-2}^{(i)}), \dots, (x_{j,m-n_{Q_{j,m}^{(i)}}}^{(i)}, y_{j,m-n_{Q_{j,m}^{(i)}}}^{(i)})\}$. For each input dimension j , where $j \in \{1 \dots J\}$, the initial set of quantization clusters $\mathbf{Q}_j^{(0)}$ is derived from the input training data points $x_{s,j}$, $s \in \{1 \dots S\}$ in the j^{th} input dimension. Each *distinct* input value $x_{s,j}$ from the training set constitutes *one* quantization cluster in the j^{th} dimension. Training data points with the same input value are combined together as a cluster such

that

$$Q_{j,m}^{(0)} = \{(x_{j,m,1}^{(0)}, y_{j,m,1}^{(0)}), (x_{j,m,2}^{(0)}, y_{j,m,2}^{(0)}), \dots, (x_{j,m,n_{Q_{j,m}}}^{(0)}, y_{j,m,n_{Q_{j,m}}}^{(0)})\} \quad (3.1)$$

where $x_{j,m,1}^{(0)} = x_{j,m,2}^{(0)} = \dots = x_{j,m,n_{Q_{j,m}}}^{(0)} = x_{j,m}^{(0)}$. Each quantization cluster $Q_{j,m}^{(i)}$ is defined by a characteristic value $V_{j,m}^{(i)}$ and its centroid $P_{j,m}^{(i)}$. The characteristic value $V_{j,m}^{(i)}$ of a quantization cluster $Q_{j,m}^{(i)}$ at the i^{th} iteration of the hierarchical clustering process is computed as the statistical mean of the output values $y_{j,m,k}^{(i)}$ associated to the input points $x_{j,m,k}^{(i)}$ and is described by eq. (3.2).

$$V_{j,m}^{(i)} = \frac{\sum_{k=1}^{n_{Q_{j,m}}} y_{j,m,k}^{(i)}}{n_{Q_{j,m}}} \quad (3.2)$$

where $n_{Q_{j,m}}$ is the total number of data points in the quantization cluster $Q_{j,m}^{(i)}$ at the i^{th} iteration. $P_{j,m}^{(i)}$, on the other hand, denotes the centroid of the quantization cluster $Q_{j,m}^{(i)}$ and is described by eq. (3.3).

$$P_{j,m}^{(i)} = \frac{\max_{k \in \{1 \dots n_{Q_{j,m}}\}} (x_{j,m,k}^{(i)}) + \min_{k \in \{1 \dots n_{Q_{j,m}}\}} (x_{j,m,k}^{(i)})}{2} \quad (3.3)$$

With respect to the input dimension j , the initial set of quantization clusters is defined as: $\mathbf{Q}_j^{(0)} = \{Q_{j,1}^{(0)}, Q_{j,2}^{(0)}, \dots, Q_{j,M_j}^{(0)}\}$ where $V_{j,m}^{(0)} = \frac{\sum_k y_{j,m,k}^{(0)}}{n_{Q_{j,m}}}$, $k \in \{1 \dots n_{Q_{j,m}}\}$ and $P_{j,m}^{(0)} = x_{j,m}^{(0)}$ (since $x_{j,m,1}^{(0)} = x_{j,m,2}^{(0)} = \dots = x_{j,m,n_{Q_{j,m}}}^{(0)} = x_{j,m}^{(0)}$).

Step 3 Iteratively merge similar quantization clusters.

The clusters in the initial set of quantization clusters $\mathbf{Q}_j^{(0)}$ are iteratively merged until the number of quantization clusters in the j^{th} dimension M_j equals the number of predefined available memory cells, i.e. $M_j = \hat{M}$. In HCAQ-CMAC, memory cells are allocated based on the observed characteristics of the input-output mappings of the training data. This is achieved by defining similarity measures and the grouping together of similar clusters in the initial set of quantization clusters. The similarity between two HCAQ-CMAC quantization clusters is evaluated via the characteristic values and the centroid of the corresponding clusters. During a cluster-merging iteration, only the two most similar adjacent clusters are combined. Following this, the characteristic value and centroid of the resultant cluster are computed for the next cluster-merging iteration.

The cluster-merging decision is based on a *cost function*. The merging cost function is defined as the weighted combination of the distances between the characteristic values and the centroids of two adjacent clusters, and is described mathematically in eq. (3.4).

$$\mathfrak{F}(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)}) = \beta_1 \left(\|V_{j,m_1}^{(i)} - V_{j,m_2}^{(i)}\| \right) + \beta_2 \left(\|P_{j,m_1}^{(i)} - P_{j,m_2}^{(i)}\| \right) \quad (3.4)$$

$$\beta_1 + \beta_2 = 1.0 \quad (3.5)$$

where $\mathfrak{F}(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)})$ is the merging cost of the two adjacent clusters $Q_{j,m_1}^{(i)}$ and $Q_{j,m_2}^{(i)}$ (i.e. $m_1, m_2 \in \{1 \cdots M_j\}$ and $m_2 = m_1 + 1$) at the i^{th} iteration, and β_1 and β_2 are user-defined parameters.

The parameters β_1 and β_2 weigh the respective importance of the measured differences in the output (characteristic values) and the input (quantization points) dimensions as the total cost of merging two adjacent clusters. The weighting parameter β_1 is concerned with the similarity of the outputs of the two clusters. β_2 , on the other hand, controls the importance of the similarity of the inputs in the two clusters. As such, the selection of β_1 and β_2 parameters generally varies greatly with different applications and may be guided by the relevant prior knowledge about the applications or the training data. An application with slow changing input but fast changing output may require a bigger β_1 than β_2 and vice-versa. In each iteration, the two adjacent clusters with the smallest merging cost are combined as in eq. (3.6).

$$\begin{aligned} Q_{j,m'}^{(i+1)} &= Q_{j,\tilde{m}_1}^{(i)} \cup Q_{j,\tilde{m}_2}^{(i)}, \text{ iff} \\ \mathfrak{F}(Q_{j,\tilde{m}_1}^{(i)}, Q_{j,\tilde{m}_2}^{(i)}) &= \min_{m_1, m_2 \in \{1 \cdots M_j\} \text{ and } m_2 = m_1 + 1} \mathfrak{F}(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)}) \end{aligned} \quad (3.6)$$

The characteristic value $V_{j,m'}^{(i+1)}$ and the centroid $P_{j,m'}^{(i+1)}$ of the merged cluster $Q_{j,m'}^{(i+1)}$ are recomputed using eq. (3.2) and eq. (3.3). The cluster-merging process continues until the number of clusters in the input dimension j reaches the predefined memory size $\hat{M}$ (see Figure 3.2). This computation is analogical to the activity-dependent pruning of the extraneous synaptic connections in the development of the human brain. Weak (or non-active) neurons are eliminated, and their functions annexed by the winning (or more active) neurons. In HCAQ-CMAC, similar clusters are merged and represented by larger/expanded clusters to reduce data re-

Figure 3.2: Illustration of the HCAQ-CMAC non-uniform quantization process for an arbitrary input dimension j where $\hat{M} = 6$

dundancy. Specifically, HCAQ-CMAC allocates more memory cells to the densely data-populated areas with higher degrees of output variation.

Step 4 *Construct the quantization decision function.*

A set of $\hat{M}$ quantization clusters is obtained at the end of the hierarchical clustering process for each input dimension j . Let I denote the last cluster-merging iteration for input j . Thus the final set of quantization clusters for input j is defined as: $\mathbf{Q}_j^{(I)} = \{Q_{j,1}^{(I)}, Q_{j,2}^{(I)}, \dots, Q_{j,\hat{M}}^{(I)}\}$. Subsequently, the quantization decision function in the j^{th} input dimension $\mathbf{Q}_j[\cdot]$ is determined from $\mathbf{Q}_j^{(I)}$, as described by eq. (3.7).

$$\mathbf{Q}_j[\cdot] \rightarrow \{P_{j,1}^{(I)}, P_{j,2}^{(I)}, \dots, P_{j,\hat{M}}^{(I)}\} \quad (3.7)$$

where $\mathbf{Q}_j[\cdot]$ denotes the quantization mapping function in the j^{th} input dimension. $P_{j,m}^{(I)}$ is the centroid of the m^{th} quantization cluster of the j^{th} input dimension after the cluster-merging process; and $\hat{M}$ is the number of predefined memory cells in each input dimension. The quantization decision points derived for each input dimension subsequently form the memory axes of the HCAQ-CMAC network and define its overall computing structure.

Figure 3.3: The flowchart and computational complexity of the HCAQ-CMAC structural learning process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; S – number of training samples; $\hat{M}$ – number of HCAQ-CMAC memory cells per dimension

Figure 3.3 depicts a flowchart summarizing the HCAQ-CMAC structural learning process. The four major steps in the HCAQ-CMAC memory allocation process described previously are highlighted as the numbered boxes ❶–❹ in Figure 3.3. The figure also presents a brief outline of the estimated worst-case computational complexity (in Big-O notation) of the HCAQ-CMAC network formation. As shown in Figure 3.3, the total worst-case complexity of constructing a J -dimensional HCAQ-CMAC network with a memory size of $\hat{M}$ cells per dimension based on a training set of S samples is in the order of $O(JS^2) + O(\hat{M}^J)$.

3.2.2 The HCAQ-CMAC Network Computational Process

The HCAQ-CMAC network learns a correct response to an input vector by modifying the contents of the selected memory cells. The Widrow-Hoff learning rule (Widrow and Stearns, 1985) is adopted in the training of the HCAQ-CMAC network to implement an efficient error-correction-based learning algorithm. Let I be the maximum number of training iterations. The HCAQ-CMAC memory learning process for the s^{th} training sample is described as follows:

Step 1 *Determine the winner neuron for input $\mathbf{X}_s$ at the i^{th} iteration, $i \in \{1 \cdots I\}$.*

For each input $\mathbf{X}_s$, the index $\bar{\mathbf{X}}_s$ of the winner neuron in HCAQ-CMAC is computed via the quantization mapping functions $\mathbf{Q}[\cdot]$. That is, given $\mathbf{X}_s = [x_{s,1} \ x_{s,2} \ \cdots \ x_{s,j} \ \cdots \ x_{s,J}]^T$, the winner neuron $\bar{\mathbf{X}}_s$ is computed as in eq. (3.8),

$$\bar{\mathbf{X}}_s = \mathbf{Q}[\mathbf{X}_s] = \begin{bmatrix} \mathbf{Q}_1[x_{s,1}] & \mathbf{Q}_2[x_{s,2}] & \cdots & \mathbf{Q}_j[x_{s,j}] & \cdots & \mathbf{Q}_J[x_{s,J}] \end{bmatrix}^T \quad (3.8)$$

where $\bar{\mathbf{X}}_s$ denotes the quantized input $\mathbf{X}_s$ and J is the number of input dimensions.

Step 2 *Retrieve the network output.*

The output of the HCAQ-CMAC network to the input $\mathbf{X}_s$ at the i^{th} training iteration is the memory content at the location $\bar{\mathbf{X}}_s$. This is described by eq. (3.9).

$$Y_s^{(i)} = \mathbf{Z}_{\bar{\mathbf{X}}_s} \quad (3.9)$$

where $Y_s^{(i)}$ denotes the HCAQ-CMAC output for the input $\mathbf{X}_s$ during the i^{th} training iteration and $\mathbf{Z}$ is the HCAQ-CMAC hypercube memory array.

Step 3 *Compute the network output error.*

The learning error $Err_s^{(i)}$ corresponding to the input $\mathbf{X}_s$ at the i^{th} training iteration is defined as the difference between the network output $Y_s^{(i)}$ and the expected output $\hat{Y}_s$ as given in eq. (3.10).

$$Err_s^{(i)} = \hat{Y}_s - Y_s^{(i)} = \hat{Y}_s - \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} \quad (3.10)$$

Step 4 *Update the HCAQ-CMAC memory.*

The update equations for the activated cell at index $\bar{\mathbf{X}}_s$ are given by eq. (3.11) and eq. (3.12), where α denotes the learning constant.

$$\mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i+1)} = \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} + \Delta \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} \quad (3.11)$$

$$\Delta \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} = \alpha Err_s^{(i)} = \alpha (\hat{Y}_s - \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)}) \quad (3.12)$$

The learning constant α governs the learning speed of the HCAQ-CMAC network by modulating the magnitude of the learning error that is used to update the respective HCAQ-CMAC memory cell. A larger α implies a faster learning rate since a large error-correction term will be used to drive the HCAQ-CMAC output closer to the desired output in a fewer number of training iterations. However, a large learning constant α may cause the network to overshoot the optimal weights. There is also a risk of learning instability due to the large oscillation in the HCAQ-CMAC memory contents across the different training iterations. On the other hand, a small α updates the HCAQ-CMAC slowly, but it is also more likely to result in the network converging to the optimal weights.

During the testing phase of the HCAQ-CMAC network, a neighborhood-based activation of the network cells is employed to smoothen the computed output. Given an input stimulus $\mathbf{X}_s = [x_{s,1} \ x_{s,2} \ \cdots \ x_{s,j} \ \cdots \ x_{s,J}]^T$ to the HCAQ-CMAC network, the network output during the testing phase is derived as follows:

Step 1 *Determine the region of activation.*

The computed output of the HCAQ-CMAC network corresponding to an input stimulus $\mathbf{X}_s$ is defined as the mean of the memory contents (values) of the activated cells in the neighborhood vicinity of $\mathbf{X}_s$. This averaging process is employed to smooth the network output so that abrupt changes in the computed output for characteristically similar inputs can be avoided. The neighborhood-based compu-

tation also facilitates the generalization capability of the HCAQ-CMAC network against inputs for which the network has never been trained for. The neighborhood of $\mathbf{X}_s$ is defined by a neighborhood constant N , which determines the relative size of the neighborhood with respect to the input domain. For an input stimulus $\mathbf{X}_s$, its activation neighborhood is defined by eq. (3.13) and (3.14).

$$lb_{s,j} = x_{s,j} - 0.5 \cdot N \cdot range_{s,j} \quad (3.13)$$

$$rb_{s,j} = x_{s,j} + 0.5 \cdot N \cdot range_{s,j} \quad (3.14)$$

where $j \in \{1, 2, \dots, J\}$ denotes the j^{th} input dimension; N is the neighborhood constant; $range_{s,j}$ is the input domain for the j^{th} dimension; $lb_{s,j}$ is the left boundary of the neighborhood in the j^{th} dimension; and $rb_{s,j}$ is the right boundary of the neighborhood in the j^{th} dimension. Consequently, the memory cells within the neighborhood constitute the set of activated computing cells for the input stimulus $\mathbf{X}_s$. The size of the neighborhood affects the accuracy of the computed HCAQ-CMAC output. The larger the neighborhood size, the more generalized is the output of the HCAQ-CMAC network. Conversely, a smaller neighborhood size results in a more accurate output computation. Therefore, a larger neighborhood size is suitable for a dataset that is sparse in the input space as this increases the generalization ability of HCAQ-CMAC. A smaller neighborhood size, on the other hand, is suitable for a compact dataset so as to produce more accurate results.

Step 2 Compute the HCAQ-CMAC output.

The output of the HCAQ-CMAC network with respect to the input $\mathbf{X}_s$ is defined by eq. (3.15).

$$Y_s = \frac{\sum_{k \in \mathbf{K}_s} \mathbf{Z}_k}{n_{\mathbf{K}_s}} \quad (3.15)$$

where $\mathbf{K}_s$ denotes the set of indices of the activated neighborhood cells corresponding to the input $\mathbf{X}_s$; $\mathbf{Z}_k$ is the memory content of the activated cell with index k ; $n_{\mathbf{K}_s}$ is the cardinality of $\mathbf{K}_s$; and Y_s is the output of HCAQ-CMAC with respect to the input stimulus $\mathbf{X}_s$.

Figure 3.4 depicts a flowchart of the HCAQ-CMAC computational process for both the training and testing phases. A brief analysis of the computational complexity of the two phases

Figure 3.4: The flowchart and computational complexity of the HCAQ-CMAC computational process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of HCAQ-CMAC memory cells per dimension; N – HCAQ-CMAC neighborhood size

for a J -dimensional HCAQ-CMAC network employing $\hat{M}$ memory cells per dimension are also shown in the figure. From Figure 3.4, one can observe that the worst-case complexity of the HCAQ-CMAC training process for the s^{th} sample is in the order of $O(J\hat{M})$, which arises from the estimated worst-case computation time of the indices to the winner neuron in all the J dimensions of the HCAQ-CMAC network. The worst-case testing complexity of the HCAQ-CMAC network for a particular input $\mathbf{x}_s$, on the other hand, is in the order of $O(J\hat{M}) + O((N\hat{M})^J)$, where N denote the neighborhood size of the HCAQ-CMAC network. Hence, a larger neighborhood increases the generalization capability of the HCAQ-CMAC network at the expense of higher computational complexity.

3.3 The HCAQ-CMAC Neural Correlates

As a computational model of the human cerebellum, the proposed HCAQ-CMAC network possesses characteristics analogical to the neurobiological and neurophysiological aspects of its biological counterpart. Table 3.1 lists the neural correlates between the human cerebel-

Table 3.1: Correspondence between the neurophysiological aspects of the human cerebellum and the functionalities of the proposed HCAQ-CMAC network

Characteristics	The Cerebellum	The HCAQ-CMAC Network
Physical connectivity	Purkinje cells are the main computational units of the human cerebellum and the parallel fiber inputs run perpendicularly to the flat fan-like dendritic arborization of the Purkinje cells.	Memory cells of the HCAQ-CMAC network are analogous to the Purkinje cells in the human cerebellum and the grid-like organization of these memory cells is inspired by the anatomy of the biological interconnections of the Purkinje cells and the parallel fibers.
Output computation	The output of the cerebellum originates from the deep cerebellar nuclei, which combine the outputs of the activated Purkinje cells.	The output of the HCAQ-CMAC network is an aggregate of the weighted contents of the activated memory cells.
Functionality	The cerebellum performs associative mapping from the input sensory afferent and cerebral efferent signals to the output of the brain construct.	The HCAQ-CMAC network performs associative mapping from the input vector of the network to the output response.
Synaptic organization	Research into the neurophysiological aspects of the human brain have established that the precise wirings in an adult human brain are laid out as the results of competition-based neuronal selection process.	The HCAQ-CMAC network employs the hierarchical clustering technique to perform the memory allocation procedure in which more memory cells are allocated to the input segments that contain more information.
Learning principle	The human cerebellum adopts an error-correction-driven supervised learning paradigm.	The HCAQ-CMAC network adopts the modified Widrow-Hoff training algorithm, which is essentially an error-correction based supervised learning scheme.
Learning convergence	Learning stability is vital since the cerebellum is responsible for smooth and precise coordination of the motor movements.	Learning stability (convergence) in the HCAQ-CMAC network has been established (Refer to Section 3.4).

lum and the HCAQ-CMAC network.

3.4 Learning Convergence of the HCAQ-CMAC Network

This section presents the mathematical proof of the learning convergence of the proposed HCAQ-CMAC network. Figure 3.5 depicts an example of the memory surface $\mathbf{Z}$ of a 2-input HCAQ-CMAC network. With respect to Figure 3.5, the quantization points along the X_1 dimension are $\{P_{1,1}, P_{1,2}, P_{1,3}, \dots, P_{1,\hat{M}}\}$ and along the X_2 dimension are $\{P_{2,1}, P_{2,2}, P_{2,3}, \dots, P_{2,\hat{M}}\}$ respectively. $\mathbf{Z}_{(p1,p2)}$ denotes the network cell with the address index $(p1, p2)$.

Figure 3.5: An example of the 2-input HCAQ-CMAC memory array

Figure 3.6: 2D HCAQ-CMAC $\mathbf{Z} \rightarrow \mathbf{W}$ mapping

3.4.1 Mathematical Perspective of The HCAQ-CMAC Network

The HCAQ-CMAC network employs a winner-take-all learning principle where each input training tuple accesses and modifies the memory content of *one* winner neuron. Each input vector to the network is quantized to the nearest quantization level in each dimension to identify the index of the winner neuron. The HCAQ-CMAC network output is derived from the winner network cell. Consequently, the network learning process is performed on this winner cell.

The conceptual memory surface $\mathbf{Z}$ of a multi-input HCAQ-CMAC network can be expressed as a one-dimensional weight array $\mathbf{W}$. Figure 3.6 illustrates the *linearization* of the conceptual memory surface $\mathbf{Z}$ to the physically implemented one-dimensional weight array $\mathbf{W}$ for a two-dimensional HCAQ-CMAC. With respect to the HCAQ-CMAC network, the computed

output for the s^{th} input vector (stimulus) $\mathbf{X}_s$ is defined in eq. (3.16).

$$Y_s^{(i)} = \mathbf{Z}_{\mathbf{Q}[\mathbf{X}_s]} \quad (3.16)$$

where i is the training iteration number; $\mathbf{Q}[\cdot]$ is the quantization mapping function of HCAQ-CMAC; and $\mathbf{Q}[\mathbf{X}_s]$ is the index to the winner neuron corresponding to the input $\mathbf{X}_s$.

The HCAQ-CMAC network presented here is a multi-input single-output system. Let the total number of memory cells in the HCAQ-CMAC network be $\hat{M}^J$ and the column vector $\mathbf{C}_s$ (see eq. (3.17)) denote the activation mask of the HCAQ-CMAC memory cells with respect to the s^{th} input training sample. That is,

$$\mathbf{C}_s^T = \underbrace{[c_{s,1} \quad c_{s,2} \quad \cdots \quad c_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \quad (3.17)$$

$$c_{s,j} = \begin{cases} 1, & \text{if the } j^{th} \text{ memory cell is activated} \\ 0, & \text{otherwise} \end{cases} \quad (3.18)$$

Note that the winner-take-all learning algorithm of the HCAQ-CMAC network implies that for all $s \in \{1 \cdots S\}$, only one element of $\mathbf{C}_s$ is non-zero. The scalar output of the HCAQ-CMAC network can thus be formulated as a vector product described by eq. (3.19).

$$Y_s = \underbrace{[c_{s,1} \quad c_{s,2} \quad \cdots \quad c_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \mathbf{W}_s = \mathbf{C}_s^T \mathbf{W}_s \quad (3.19)$$

where $\mathbf{W}_s$ is the memory content of the entire HCAQ-CMAC network structure when the s^{th} input training sample is presented. The memory update equation of the HCAQ-CMAC network for the s^{th} input training sample is subsequently defined as eq. (3.20).

$$\mathbf{W}_{s+1}^{(i)} = \mathbf{W}_s^{(i)} + \underbrace{\Delta \mathbf{W}_s^{(i)}}_{\alpha \times \text{local error}} = \mathbf{W}_s^{(i)} + \alpha \underbrace{\mathbf{C}_s \{\hat{Y}_s - \mathbf{C}_s^T \mathbf{W}_s^{(i)}\}}_{\text{local error}} \quad (3.20)$$

where:

- $\mathbf{W}_{s+1}^{(i)}$ the memory content of the entire HCAQ-CMAC network structure when the $(s+1)^{th}$ training sample is presented in the i^{th} training iteration;
- α the learning constant;
- $\mathbf{C}_s$ the activation mask of the HCAQ-CMAC memory cells; and

$\hat{Y}_s$ the desired (expected) HCAQ-CMAC output for the s^{th} input training vector.

The difference of the HCAQ-CMAC memory contents between two successive iterations for the s^{th} input training sample (denoted as $\mathbf{Dw}_s^{(i)}$) is therefore as defined in eq. (3.21).

$$\begin{aligned}
 \mathbf{Dw}_s^{(i)} &= \mathbf{W}_s^{(i+1)} - \mathbf{W}_s^{(i)} \\
 &= \underbrace{\mathbf{W}_{s-1}^{(i+1)} + \Delta \mathbf{W}_{s-1}^{(i+1)}}_{\mathbf{W}_s^{(i+1)}} - \underbrace{(\mathbf{W}_{s-1}^{(i)} + \Delta \mathbf{W}_{s-1}^{(i)})}_{\mathbf{W}_s^{(i)}} \\
 &= \underbrace{\mathbf{Dw}_{s-1}^{(i)}}_{\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)}} + \underbrace{\alpha \mathbf{C}_{s-1} \{\hat{Y}_{s-1} - \mathbf{C}_{s-1}^T \mathbf{W}_{s-1}^{(i+1)}\}}_{\Delta \mathbf{W}_{s-1}^{(i+1)}} - \underbrace{\alpha \mathbf{C}_{s-1} \{\hat{Y}_{s-1} - \mathbf{C}_{s-1}^T \mathbf{W}_{s-1}^{(i)}\}}_{\Delta \mathbf{W}_{s-1}^{(i)}} \\
 &= \mathbf{Dw}_{s-1}^{(i)} - \alpha \mathbf{C}_{s-1} \mathbf{C}_{s-1}^T \underbrace{(\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)})}_{\mathbf{Dw}_{s-1}^{(i)}} \\
 &= (\mathbf{I} - \alpha \mathbf{C}_{s-1} \mathbf{C}_{s-1}^T) \mathbf{Dw}_{s-1}^{(i)} \tag{3.21}
 \end{aligned}$$

Note that the activation mask $\mathbf{C}_s$ is a constant for an arbitrary input training sample s across different training iterations. This is because the HCAQ-CMAC network structure is static after the structural learning phase. Following eq. (3.21), the delta memory contents for a sequence of training data $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_s, \dots, \mathbf{X}_S)$ is described by eq. (3.23).

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T), \text{ and} \tag{3.22}$$

$$\mathbf{Dw}^{(i)} \equiv \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \dots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \tag{3.23}$$

where S denotes the total number of training samples.

The learning convergence of the HCAQ-CMAC network is established via the convergence of the network memory contents as training approaches infinity. In this case, the sufficient and necessary condition for the HCAQ-CMAC learning process to convergence can be expressed in eq. (3.24) and (3.25).

$$\lim_{i \rightarrow \infty} \mathbf{Dw}_s^{(i)} = 0, \quad \forall s \in \{1 \dots S\} \quad \text{or} \tag{3.24}$$

$$\lim_{i \rightarrow \infty} \mathbf{Dw}^{(i)} = [0] \tag{3.25}$$

where $[0]$ is the null matrix. Substituting eq. (3.22) into eq. (3.21),

$$\mathbf{Dw}_s^{(i)} = \mathbf{E}_{s-1} \mathbf{Dw}_{s-1}^{(i)} \tag{3.26}$$

HCAQ-CMAC is iteratively trained on a set of S samples. When $s = 1$, from eq. (3.26)

$$\mathbf{Dw}_1^{(i)} = \mathbf{E}_0 \mathbf{Dw}_0^{(i)}, \text{ such that} \quad (3.27)$$

$$\mathbf{Dw}_0^{(i)} = \mathbf{Dw}_S^{(i-1)} \text{ and} \quad (3.28)$$

$$\mathbf{E}_0 = \mathbf{E}_S \Rightarrow \mathbf{C}_0 = \mathbf{C}_S \text{ (From eq. (3.22))} \quad (3.29)$$

Following the results of eqs. (3.26 – 3.29), $\mathbf{Dw}^{(i)}$ (see eq. (3.23)) can be re-expressed as

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \dots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\ &= \begin{bmatrix} \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \underbrace{\mathbf{E}_1 \mathbf{Dw}_1^{(i)}}_{\mathbf{Dw}_2^{(i)}} & \dots & \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i)}}_{\mathbf{Dw}_S^{(i)}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}_S \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i-1)}}_{\mathbf{Dw}_S^{(i-1)}} & \mathbf{E}_1 \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \dots & \mathbf{E}_{S-1} \underbrace{\mathbf{E}_{S-2} \mathbf{Dw}_{S-2}^{(i)}}_{\mathbf{Dw}_{S-1}^{(i)}} \end{bmatrix} \end{aligned} \quad (3.30)$$

Decomposing the $\mathbf{Dw}$ terms on the right hand side repeatedly results in eq. (3.31).

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} (\mathbf{E}_S \mathbf{E}_{S-1} \dots \mathbf{E}_1) \mathbf{Dw}_1^{(i-1)} & (\mathbf{E}_1 \mathbf{E}_S \dots \mathbf{E}_2) \mathbf{Dw}_2^{(i-1)} & \dots \\ (\mathbf{E}_{S-1} \mathbf{E}_{S-2} \dots \mathbf{E}_S) \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \end{aligned} \quad (3.31)$$

Following eq. (3.31),

$$\mathbf{G}_s \equiv \mathbf{E}_{s-1} \mathbf{E}_{s-2} \dots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \dots \mathbf{E}_s, \quad s \in \{1 \dots S\} \quad (3.32)$$

Therefore, eq. (3.31) can be re-expressed as:

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \dots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\ &= \begin{bmatrix} \underbrace{(\mathbf{E}_S \mathbf{E}_{S-1} \dots \mathbf{E}_1)}_{\mathbf{G}_1} \mathbf{Dw}_1^{(i-1)} & \underbrace{(\mathbf{E}_1 \mathbf{E}_S \dots \mathbf{E}_2)}_{\mathbf{G}_2} \mathbf{Dw}_2^{(i-1)} & \dots & \underbrace{(\mathbf{E}_{S-1} \mathbf{E}_{S-2} \dots \mathbf{E}_S)}_{\mathbf{G}_S} \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \dots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \end{aligned} \quad (3.33)$$

It can be observed that

$$\mathbf{Dw}_s^{(i)} = \mathbf{G}_s \mathbf{Dw}_s^{(i-1)}, \quad s \in \{1 \dots S\} \quad (3.34)$$

Consequently, it follows that

$$\begin{aligned}
\mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \dots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \dots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{G}_1 \underbrace{\mathbf{G}_1 \mathbf{Dw}_1^{(i-2)}}_{\mathbf{Dw}_1^{(i-1)}} & \mathbf{G}_2 \underbrace{\mathbf{G}_2 \mathbf{Dw}_2^{(i-2)}}_{\mathbf{Dw}_2^{(i-1)}} & \dots & \mathbf{G}_S \underbrace{\mathbf{G}_S \mathbf{Dw}_S^{(i-2)}}_{\mathbf{Dw}_S^{(i-1)}} \end{bmatrix} \\
&= \begin{bmatrix} (\mathbf{G}_1)^2 \mathbf{Dw}_1^{(i-2)} & (\mathbf{G}_2)^2 \mathbf{Dw}_2^{(i-2)} & \dots & (\mathbf{G}_S)^2 \mathbf{Dw}_S^{(i-2)} \end{bmatrix} \quad (3.35)
\end{aligned}$$

Further repeated decomposition of the $\mathbf{Dw}$ terms on the right hand side results in the following expression of eq. (3.36).

$$\mathbf{Dw}^{(i)} = \begin{bmatrix} (\mathbf{G}_1)^i \mathbf{Dw}_1^{(0)} & (\mathbf{G}_2)^i \mathbf{Dw}_2^{(0)} & \dots & (\mathbf{G}_S)^i \mathbf{Dw}_S^{(0)} \end{bmatrix} \quad (3.36)$$

With respect to eq. (3.36), the memory difference matrix $\mathbf{Dw}^{(i)}$ must approach a null matrix as training tends to infinity (i.e. $i \rightarrow \infty$) in order to establish the learning convergence of the proposed HCAQ-CMAC network. Hence, the HCAQ-CMAC learning process converges if and only if eq. (3.37) holds.

$$\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{Dw}_s^{(0)} = 0, \quad \forall s \in \{1 \dots S\} \quad (3.37)$$

By definition, the difference vector $\mathbf{Dw}_s^{(0)}$ can be expressed as eq. (3.38).

$$\begin{aligned}
\mathbf{Dw}_s^{(0)} &= \mathbf{W}_s^{(1)} - \mathbf{W}_s^{(0)} = \underbrace{\mathbf{W}_{s-1}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)}}_{\mathbf{W}_s^{(1)}} - \mathbf{W}_s^{(0)} \\
&= \underbrace{\mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-2}^{(1)}}_{\mathbf{W}_{s-1}^{(1)}} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \quad (3.38)
\end{aligned}$$

Decomposing the $\mathbf{W}_s^{(i)}$ terms on the right hand side repeatedly produces eq. (3.39).

$$\begin{aligned}
\mathbf{Dw}_s^{(0)} &= \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_2^{(1)} + \dots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\
&= \underbrace{\mathbf{W}_S^{(0)} + \Delta \mathbf{W}_S^{(0)}}_{\mathbf{W}_1^{(1)}} + \Delta \mathbf{W}_1^{(1)} + \dots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\
&= \underbrace{\mathbf{W}_s^{(0)} + \Delta \mathbf{W}_s^{(0)}}_{\mathbf{W}_{s+1}^{(0)}} + \Delta \mathbf{W}_{s+1}^{(0)} + \dots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \dots + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)}
\end{aligned}$$

$$= \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} \quad (3.39)$$

From eq. (3.20), the HCAQ-CMAC memory update due to the s^{th} input training sample at the i^{th} iteration ($\Delta \mathbf{W}_s^{(i)}$) is computed via eq. (3.40).

$$\Delta \mathbf{W}_s^{(i)} = \alpha \underbrace{\mathbf{C}_s \{\hat{Y}_s - \mathbf{C}_s^T \mathbf{W}_s^{(i)}\}}_{\text{local error}}^{\text{learning error}} \quad (3.40)$$

where $\{\hat{Y}_s - \mathbf{C}_s^T \mathbf{W}_s^{(i)}\}$ is a scalar value and it is the learning (training) error of HCAQ-CMAC for the s^{th} input training vector at the i^{th} iteration. If

$$u_s^{(i)} = (\hat{Y}_s - \mathbf{C}_s^T \mathbf{W}_s^{(i)}) \quad (3.41)$$

then eq. (3.40) can be re-expressed as eq. (3.42).

$$\Delta \mathbf{W}_s^{(i)} = \alpha \mathbf{C}_s u_s^{(i)} \quad (3.42)$$

From eqs. (3.36), (3.39) and (3.42),

$$\begin{aligned} (\mathbf{G}_s)^i \mathbf{D} \mathbf{w}_s^{(0)} &= (\mathbf{G}_s)^i \{\Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)}\} \\ &= (\mathbf{G}_s)^i \{\alpha \mathbf{C}_s u_s^{(0)} + \alpha \mathbf{C}_{s+1} u_{s+1}^{(0)} + \cdots + \alpha \mathbf{C}_{s-1} u_{s-1}^{(1)}\} \\ &= \alpha (\mathbf{G}_s)^i \{\mathbf{C}_s u_s^{(0)} + \mathbf{C}_{s+1} u_{s+1}^{(0)} + \cdots + \mathbf{C}_{s-1} u_{s-1}^{(1)}\}, \quad s \in \{1 \cdots S\} \end{aligned} \quad (3.43)$$

Therefore, if $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [0]$ for all $a \in \{1 \cdots S\}$, $(\mathbf{G}_s)^i \mathbf{D} \mathbf{w}_s^{(0)}$ in eq. (3.43) evaluates as null. From eq. (3.36), the matrix $\mathbf{D} \mathbf{w}^{(i)} = [0]$ as $i \rightarrow \infty$ follows. Consequently, the learning process of the proposed HCAQ-CMAC network converges.

3.4.2 Learning Convergence of The HCAQ-CMAC Network

Theorem 3.1 *The training process of the HCAQ-CMAC network converges if and only if the learning constant α is such that $0 < \alpha < 2$.*

Proof: It can be shown that for all $a \in \{1 \cdots S\}$, when $0 < \alpha < 2$, $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [0]$ (Refer to Appendix C for a detailed proof). Therefore, the training process of the HCAQ-CMAC network converges if and only if the learning constant α satisfies the condition

$0 < \alpha < 2$. In another words, the learning constant α must satisfy the singular condition of $0 < \alpha < 2$ in order for the HCAQ-CMAC learning process to converge to a limit cycle.

3.5 Case Studies

This section presents the experiments that have been conducted to evaluate the performance of the proposed HCAQ-CMAC network. The experiments are based on two real-life applications. They are: (1) automatic control of car maneuver and (2) modeling of the human glucose metabolic process. The performances of the HCAQ-CMAC network are benchmarked against: (1) the basic CMAC network to demonstrate the memory efficiency achievable by the proposed HCAQ-CMAC network; (2) Moody's multi-resolution CMAC network (Moody, 1989) to assess the generalization and learning ability of the proposed network; and (3) Menozzi's tree-based multi-resolution CMAC (Menozzi and Chow, 1997) to assess both the modeling performance and memory efficiency achievable by HCAQ-CMAC.

3.5.1 Case I: Automatic Control of Car Maneuver

The *Intelligent Vehicle Project* is part of an ongoing research effort to develop an Intelligent Transportation System (ITS) at the Centre for Computational Intelligence (C2i) (C2iWeb, Online). The objective of the project is to realize the brain-inspired intelligence-based technologies required for the automation of control, routing and navigation of land vehicles. In this chapter, the proposed HCAQ-CMAC network is employed for the construction of an autopilot system for car maneuver.

Albeit its complexity, driving a vehicle is a motor task that humans are able to perform relatively well. It has been well-established that the learning of motor skills is mediated by the human *procedural memory system* (Eichenbaum, 2002), which consist of the cerebellum and the striatum (part of basal ganglia formation). The human procedural memory system is a facet of the brain's information processing capacity specifically for the acquisition of skilled behaviors and habits. Vehicle-driving comprises of finely-tuned sets of sensory-feedback to control action mappings that are accumulated through experiences and repeated practices. Although humans are quite adept at mastering complex skills, it is difficult to formalize these behaviors into mathematical algorithms. In such cases, the construction

Figure 3.7: Driving simulation environment

of a computational model that is able to emulate the functionality of the human brain is required (Nechyba and Xu, 1997). This subsequently motivates the use of HCAQ-CMAC to model and emulate the human driving expertise.

In this experiment, a driving simulator (as developed in (Pasquier et al., 2001)) is employed to capture the behavioral response of a human driver. The simulator consists of a 3D virtual driving environment that integrates a detailed model of the vehicle dynamics and engine characteristics together with the environmental parameters such as road profiles. The approach to the experiment is to capture and record the driving data of a human driver, which consists of a set of distance-sensor feedback information and the corresponding steering control performed by the human driver as he maneuvers the simulated car around a specified track. The feedback signals provide information such as the distance of the vehicle from the road boundaries. The simulator allows for vehicle control via steering adjustment. An overview of the driving control sequence is given in Figure 3.7(a). These feedback-control records are then used to train the HCAQ-CMAC-based autopilot system.

The objective of the autopilot system is to control the vehicle to follow a particular lane in a multi-lane circuit track. The simulated vehicle model is equipped with eight directional sensors as shown in Figure 3.7(b). The semantics of the sensor readings are tabulated as Table 3.2. For the autopilot system, only the front four sensors are utilized (i.e. SFLSTB, FLSTB, FRSTB, SFRSTB) as the inputs to the HCAQ-CMAC network. As an output,

Table 3.2: The sensors definition of the Car Simulator

Symbol	Explanation
FLSTB	Front Left Sensor to Barrier
FRSTB	Front Right Sensor to Barrier
SFLSTB	Side Front Left Sensor to Barrier
SFRSTB	Side Front Right Sensor to Barrier
SBLSTB	Side Back Left Sensor to Barrier
SBRSTB	Side Back Right Sensor to Barrier
BLSTB	Back Left Sensor to Barrier
BRSTB	Back Right Sensor to Barrier

(a) Training Track (Track 1)

(b) Testing Track (Track 2)

Figure 3.8: Driving tracks

the network responds with the appropriate steering angle. Two tracks are used in this experiment. The HCAQ-CMAC autopilot system is first trained on Track 1 (Figure 3.8(a)), with the car traveling in both clockwise and anti-clockwise direction. Subsequently, Track 2 (Figure 3.8(b)) is used for testing. The simulated track is five meters wide.

The training dataset (recorded from a human driver) contains 1018 samples. HCAQ-CMAC and the benchmarked systems were trained within 500 training epochs with a network learning constant of 0.1. Table 3.3 outlines the performances of the HCAQ-CMAC autopilot system as compared to: (1) the basic CMAC network; (2) Moody's Multi-Resolution CMAC network; and (3) the Tree-based Multi-Resolution CMAC network. Each simulation test result was collected over 100 seconds of driving-time, with a maximum driving speed of 100 km/h. The driving performances of the various networks were measured by the average deviation of the controlled car from the center of the lane (ACD) and the average deviation

tion of the car orientation from the desired orientation (AOD). Both the ACD and AOD measurements are subsequently normalized and reported as Normalized ACD (NACD) and Normalized AOD (NAOD). NACD denotes the ACD with respect to the half track width, which is the maximum leeway available before the car collides with the road boundaries. The AOD values are normalized with respect to π radians. A *Performance Index* (PI_1) is used to combine the NACD and NAOD measures as described in eq. (3.44).

$$PI_1 = (100 - NACD) + (100 - NAOD) \quad (3.44)$$

$$\widehat{PI}_1 = PI_1/200 \quad (3.45)$$

where $\widehat{PI}_1$ is the normalized PI_1 . Thus, a higher PI_1 value corresponds to a better driving performance.

For the simulation, a neighborhood constant of 0.2 was empirically determined for the evaluated networks. The network size is also varied to determine the optimal performance for each of the networks. Based on the PI_1 values in Table 3.3, the basic CMAC network achieved an optimal performance with a memory size of 10 memory cells per-dimension. For CMAC, a small network size results in a coarse partitioning of the input space, and hence the network suffers from an *averaging effect*. A large CMAC network, however, fails to generalize from the available training data and thus performs poorly on the testing track. The multi-resolution CMAC architectures, on the other hand, employ layers of overlapping CMAC networks with different resolutions to address the generalization-accuracy dilemma. Both Moody's Multi-Resolution CMAC (MMR-CMAC) and the Tree-based Multi-Resolution CMAC (TMR-CMAC) architectures were benchmarked using two and three-layered implementations. These overlay implementations improve the generalization ability of the finer-resolution CMACs while simultaneously maintaining their output accuracy, but at the expense of higher memory requirements. Best PI_1 values were observed for a two-layer MMR-CMAC with memory size of 5 cells per dimension for the first layer and 10 cells per-dimension for the secondary layer, and a two-level TMR-CMAC employing a CMAC of size 8 cells per-dimension at the first level and 26 CMACs of size 2 per-dimension at the second level.

From the results presented in Table 3.3, one can observe that the proposed HCAQ-CMAC-based autopilot system consistently outperformed the other three CMAC architectures. An optimal HCAQ-CMAC network performance is obtained with a memory size of only 5

Table 3.3: Comparison of driving results for the various benchmarked networks using the Autopilot System*

Networks	Memory size (4 input dimensions)	Train Time [ms]	ACD [m]	NACD [%]	AOD [rads]	NAOD [%]	PI_1	$\widehat{PI}_1$
CMAC	7^4	8749	0.2694	10.78	0.7388	23.53	165.60	0.8280
	8^4	9031	0.3497	13.99	0.7272	23.16	162.85	0.8143
	9^4	collided						
	10^4	14453	0.2068	8.27	0.7594	24.18	167.55	0.8378
	11^4	21687	0.3131	12.52	0.7407	23.59	163.89	0.8195
HCAQ-CMAC	12^4	87531	0.4071	16.28	0.7016	22.34	161.38	0.8069
	5^4	7562	0.1501	6.00	0.6908	22.00	172.00	0.8600
	6^4	9109	0.1591	6.36	0.7145	22.75	170.89	0.8545
	7^4	14234	0.1557	6.23	0.7287	23.20	170.59	0.8530
	8^4	25796	0.2048	8.19	0.7316	23.30	168.51	0.8426
	9^4	31297	0.2079	8.32	0.7453	23.74	167.94	0.8397
	10^4	55202	0.2148	8.59	0.7283	23.19	168.22	0.8411
MMR-CMAC	$2^4 + 8^4$	19234	0.4054	16.22	0.7322	23.32	160.64	0.8032
	$4^4 + 8^4$	22813	0.4451	17.80	0.7250	23.09	159.11	0.7956
	$3^4 + 10^4$	27328	0.2085	8.34	0.7674	24.44	167.22	0.8361
	$5^4 + 10^4$	26204	0.2056	8.22	0.7497	23.88	167.90	0.8395
	$2^4 + 4^4 + 8^4$	30718	0.4770	19.08	0.7071	22.52	158.40	0.7920
	$2^4 + 3^4 + 10^4$	36484	0.3253	13.01	0.7814	24.89	162.10	0.8105
	$2^4 + 5^4 + 10^4$	34687	0.3164	12.66	0.7689	24.49	162.85	0.8143
TMR-CMAC	$8^4 + (26 \times 2^4)$	24875	0.3552	14.21	0.7378	23.50	162.29	0.8115
	$8^4 + (10 \times 2^4) + (16 \times 4^4)$	36328	0.4093	16.37	0.6944	22.11	161.52	0.8076
	$8^4 + (6 \times 4^4) + (10 \times 8^4)$	52859	0.4093	16.37	0.6996	22.28	161.35	0.8068

* CMAC denotes the basic CMAC network, HCAQ-CMAC denotes the proposed HCAQ-CMAC network, MMR-CMAC denotes Moody's Multi-Resolution CMAC network, while TMR-CMAC is the Tree-based Multi-Resolution CMAC. ACD is the average deviation from the center of the lane, NACD is the normalized ACD, AOD denotes the average deviation of the orientation, NAOD is the normalized AOD, PI_1 denotes the Performance Index, and $\widehat{PI}_1$ is the normalized PI_1 .

cells per-dimension with a PI_1 value of 172. The hierarchical clustering technique of the HCAQ-CMAC network effectively allocates the available memory cells to the input regions with high utilization throughput. This ensures that more cells are allocated to important input regions that contain more information. Therefore, a small network size does not affect the accuracy and fine-tuning capability of the HCAQ-CMAC network. Moreover, with a smaller network size, fewer cells in HCAQ-CMAC are allocated to areas with little or no training data, thus improving the generalization ability of the network. The efficient memory allocation scheme of HCAQ-CMAC also reduces the required network training time. The best-performing HCAQ-CMAC network trains in the shortest time (i.e. 7562 ms) amongst all the benchmarked networks.

Subsequently, the effective cell utilization rates of the various networks were computed and the results (denoted as the Cell Occupancy Rate or COR) are tabulated in Table 3.4. The Cell Occupancy Rate (COR) is defined as the proportion of the trained network cells to the total network size. From Table 3.4, one can observe that the HCAQ-CMAC network achieves the highest COR value in comparison with the other three CMAC architectures. This clearly demonstrated the effectiveness of the proposed HCAQ-CMAC memory allocation scheme.

3.5.2 Case II: Modeling The Dynamics of The Human Glucose Metabolic Process

Diabetes is a chronic disease where the body is unable to properly and efficiently regulate the use and storage of glucose in the blood. This resulted in large perturbations of the plasma glucose level, leading to *hyperglycemia* (elevated glucose level) or *hypoglycemia* (depressed glucose level). Chronic hyperglycemia causes severe damage to the eyes, kidneys, nerves, heart and blood vessels of the patients while severe hypoglycemia can deprive the body of energy and causes the patient to lose consciousness, which can eventually become life threatening. Currently, the treatment of diabetes is based on a two-pronged approach: strict dietary control and insulin medication.

The key component to a successful management of diabetes is essentially to develop the ability to maintain a long-term near-*normoglycaemia* state of the patient. With respect to this objective, the therapeutic effect of discrete insulin injections is not ideal as the regulation of insulin is an open-looped process. Continuous insulin infusion through an

Table 3.4: Comparison of Cell Occupancy Rates (COR) for the various benchmarked networks*

Networks	Memory Size	Total Cells	Trained Cells	COR
CMAC	7^4	2401	92	3.83%
	8^4	4096	102	2.49%
	9^4	6561	collided	
	10^4	10000	152	1.52%
	11^4	14641	176	1.2%
	12^4	20736	186	0.9%
HCAQ-CMAC	5^4	625	73	11.68%
	6^4	1296	89	6.87%
	7^4	2401	127	5.29%
	8^4	4096	161	3.93%
	9^4	6561	172	2.62%
	10^4	10000	214	2.14%
MMR-CMAC	$2^4 + 8^4$	4112	106	2.58%
	$4^4 + 8^4$	4352	130	2.99%
	$3^4 + 10^4$	10081	166	1.65%
	$5^4 + 10^4$	10625	197	1.85%
	$2^4 + 4^4 + 8^4$	4368	134	3.07%
	$2^4 + 3^4 + 10^4$	10097	169	1.67%
	$2^4 + 5^4 + 10^4$	10641	201	1.88%
TMR-CMAC	$8^4 + (26 \times 2^4)$	4512	182	4.03%
	$8^4 + (10 \times 2^4) + (16 \times 4^4)$	8352	220	2.63%
	$8^4 + (6 \times 4^4) + (10 \times 8^4)$	46592	220	0.47%

*CMAC denotes the basic CMAC network, HCAQ-CMAC denotes the proposed HCAQ-CMAC network, MMR-CMAC denotes Moody's Multi-Resolution CMAC network, while TMR-CMAC is the Tree-based Multi-Resolution CMAC.

insulin pump, on the other hand, is a more viable approach due to its controllable infusion rate (Fletcher et al., 2001). Such insulin pumps are algorithm-driven, with an avalanche of techniques proposed, investigated and reported in the literature over the years (Schetky et al., 2003; Sorensen, 1985). Generally, all such proposed methods required some forms of *accurate* modeling of the glucose metabolic process of the diabetic patient before a suitable control regime can be devised.

In recent years, emerging evidences have suggested that glucose metabolism throughout the body is coordinated by the brain through the use of insulin (Schwartz and Porte Jr., 2005). This is reinforced by the fact that glucokinase, the established glucose sensor of the pancreatic β -cells, is observed to be also present in the central nervous system (CNS) (Porte Jr. et al., 2005). Precise experimentations have subsequently demonstrated that insulin,

Table 3.5: The profile of the simulated healthy person (Subject A)

Attribute Name	Attribute Value
Sex	Male
Age	40 years old
Race	Asian
Weight	67 kg (147.71 lbs)
Height	1.70 m (5 ft 7 in)
BMI	23 (Recommended for Asian)
Lifestyle	Typical office worker with moderate physical activities such as walking briskly, leisure cycling and swimming.

via acting on the hypothalamus (a subcortical brain structure central to the autonomic control of the human endocrine system), exerts a high level of supervisory control on glucose production by the liver (Pocai et al., 2005). This observation suggests that insulin can mediate the human glucose metabolic process through an unknown signaling pathway via the CNS (Fisher and Kahn, 2003; Obici et al., 2002). This notion subsequently motivates the use of the HCAQ-CMAC network, which is a brain-inspired computational model of the human cerebellum, for the dynamic modeling of the human blood glucose cycle.

The first step into constructing a model of the human glucose metabolic process is to determine the patient profile to be modeled. Due to the lack of real-life patient data and the logistical difficulties and ethical issues involving the collection of such data, a well-known web-based simulator known as *Glucosim* (Glucosim, Online) from the Illinois Institute of Technology (IIT) is employed to simulate a person subject to generate the blood glucose data that is needed for the construction of the glucose metabolism model. A person profile for the simulated healthy subject is created as in Table 3.5.

The simulated healthy person, code-named Subject A, is a typical middle-aged Asian male. His body mass index (BMI) is 23.0, which is within the recommended range for Asian. Based on the person profile of Subject A, his recommended daily allowance (RDA) of carbohydrate intake from meals is obtained from the website of the Health Promotion Board of Singapore (HPBSg, Online). According to his sex, age, weight and lifestyle, the recommended daily carbohydrate intake for Subject A is approximately 346.9g.

Figure 3.9 illustrates a sample output from Glucosim for Subject A. This output consists of six elements: blood glucose, blood insulin, intestinal glucose absorption rate, stomach glucose, total glucose uptake rate and liver glucose production rate of Subject A respectively

Figure 3.9: Sample glucose metabolism data output from the GlucoSim simulator

over a simulated time period of 24 hours. The peaks in the stomach glucose subplot of Figure 3.9 coincide with the timings of the assumed four daily meals (i.e. breakfast, lunch, afternoon snack and dinner) while those peaks in the intestinal glucose absorption rate subplot reflect a delay effect (response) of food intake on the blood glucose level of Subject A. The subplots of blood glucose and blood insulin illustrate the insulin-glucose regulatory mechanism in a healthy person such as Subject A and depict the dynamics of the metabolic process when subjected to disturbances such as food intakes.

Since the human glucose metabolic process depends on its own current (and internal) states as well as the exogenous food intakes, it is hypothesized that the blood glucose level at any given time is a non-linear function of prior food intakes and the historical traces of the insulin and blood glucose levels. To properly account for the effects of prior food ingestion to the fluctuation of the blood glucose level, a historical window of six hours is adopted to trace the carbohydrate content of the meals taken. A soft-windowing strategy is employed to temporally partition the six-hours historical window into three conceptual segments, namely: *Recent* (i.e. previous 1 hour), *Intermediate Past* (i.e. previous 1 to 3 hours) and *Long Ago* (i.e. previous 3 to 6 hours). Based on these windows, three normalized

Figure 3.10: Soft-windowing weighting functions to compute the carbohydrate content of meal(s) in the segmented windows of the 6-hours food history (In the figure, the current time is 1840 hrs and there are three instances of food intakes in the previous 6-hours; that is, lunch at 1245 hrs, afternoon snack at 1524 hrs and dinner at 1833 hrs respectively.)

weighting functions are introduced to compute the carbohydrate content of the meal(s) (with respect to current time) taken recently, in the intermediate past or long ago. Thus, inclusive of the measured blood glucose and insulin levels, there are a total of five inputs to the modeling task. Figure 3.10 depicts the weighting function for each of the respective segmented windows.

Based on the formulated hypothesis and the preprocessed glucose data generated from GlucoSim, a total of 100 days of glucose metabolic data for Subject A was collected. The carbohydrate content and the timings of the daily meals were varied on a daily basis during the data collection phase. This ensures that HCAQ-CMAC and the benchmarked networks are not trained on a cyclical data set, but are employed to model the inherent relationships between food intakes and the glucose metabolic process of a healthy person. The collected data set is partitioned into two non-overlapping groups: 20 days of data for training and the remaining 80 days for testing and evaluation of the networks.

Simulations to model the dynamics of the blood glucose level of Subject A using the HCAQ-CMAC network were performed and the results were benchmarked against those of the basic CMAC network, the MMR-CMAC network as well as the TMR-CMAC network. For this application, a neighborhood constant (N) of 0.1 and a learning rate (α) of 0.1 are empirically determined. Table 3.6 details the recall (training) and generalization (testing) performances of the various networks with different network sizes. Two performance indicators are employed to quantify the modeling quality of the networks: the *root mean-squared error* (RMSE) and the *Pearson correlation coefficient* between the actual and the computed blood glucose level. The RMSE and Pearson correlation measures were subsequently employed to

Table 3.6: Comparison of results for the various benchmarked networks on the modeling of the human glucose metabolism process

Networks	Memory size	Recall			Generalization		
		RMSE [mg/ml]	Pearcorr	PI ₂	RMSE [mg/ml]	Pearcorr	PI ₂
CMAC	3 ⁵	20.6009	0.8717	4.04	20.2713	0.8944	4.18
	4 ⁵	16.3755	0.9210	5.30	17.8086	0.9190	4.89
	6 ⁵	11.1891	0.9639	7.91	22.9020	0.8608	3.60
	8 ⁵	8.1866	0.9808	10.66	26.0175	0.8200	3.04
	10 ⁵	6.1730	0.9892	13.79	26.2614	0.8156	2.99
HCAQ-CMAC	6 ⁵	12.8754	0.9519	6.86	12.3468	0.9661	7.24
	7 ⁵	11.2529	0.9635	7.86	11.0548	0.9724	8.07
	8 ⁵	10.0242	0.9711	8.81	10.5101	0.9748	8.47
	9 ⁵	10.0217	0.9712	8.81	10.4405	0.9752	8.52
	10 ⁵	9.9606	0.9715	8.86	10.4312	0.9748	8.53
MMR-CMAC	2 ⁵ + 4 ⁵	16.3755	0.9210	5.30	14.7504	0.9489	6.02
	2 ⁵ + 6 ⁵	11.1891	0.9639	7.91	12.0470	0.9630	7.38
	2 ⁵ + 8 ⁵	8.1866	0.9808	10.68	11.3426	0.9668	7.83
	2 ⁵ + 4 ⁵ + 8 ⁵	8.1866	0.9808	10.68	10.3823	0.9724	8.54
TMR-CMAC	(3 × 2 ⁵) + (8 × 4 ⁵)	8.2073	0.9807	10.65	34.1890	0.7014	1.99
	3 ⁵ + (21 × 2 ⁵) + (3 × 4 ⁵)	10.2723	0.9697	8.60	30.1240	0.7767	2.50
	4 ⁵ + (49 × 2 ⁵)	8.2493	0.9806	10.60	30.7527	0.7631	2.40
	6 ⁵ + (45 × 2 ⁵)	7.7855	0.9827	11.19	32.2617	0.7398	2.22

*CMAC denotes the basic CMAC network, *HCAQ-CMAC* denotes the proposed HCAQ-CMAC network, *MMR-CMAC* denotes Moody's Multi-Resolution CMAC network, while *TMR-CMAC* is the Tree-based Multi-Resolution CMAC. *Recall* refers to in-sample testing, *Generalization* is out-of-sample testing, *PI*₂ denotes the Performance Index.

compute the *Performance Index* (PI₂) described by eq. (3.46).

$$PI_2 = \frac{Pearcorr}{(1 + RMSE)} \times 100 \quad (3.46)$$

Therefore, a higher PI₂ value reflects a better network modeling performance.

Due to the characteristics of the training data, the performances of the networks vary with their respective network sizes. From the PI₂ values in Table 3.6, one can observe that for the basic CMAC network, an optimal performance (for generalization) is achieved with a memory size of 4 cells per dimension. For the basic CMAC network, a memory size of 3 cells per-dimension is too small to extract the important details from the training data, while memory sizes with 6 to 10 cells per-dimension have too high a resolution to effectively capture the general trends. The optimal configuration for the MMR-CMAC in this experiment was found to be three-layers of overlapping CMACs with network sizes of 2,

4 and 8 cells per-dimension respectively. The experimental results in Table 3.6 indicated that the MMR-CMAC network required a large number of overlapping CMACs with different resolutions to obtain a good modeling performance. This can be observed by comparing the PI_2 value of the two-layered MMR-CMAC of size 2 and 8 cells per dimension to that of the three-layered MMR-CMAC of size 2, 4, and 8 cells per-dimension respectively. Although these two MMR-CMAC networks have the same base and top layer resolutions, a significant improvement in the PI_2 value was achieved by the three-layered MMR-CMAC via the addition of the middle-layer CMAC. On the other hand, even though the TMR-CMAC network also employs layers of CMAC networks of different resolutions, its performances were found to be relatively poor in comparison to the other benchmarked networks. This may be due to the overlay mechanism of the TMR-CMAC, where finer-resolution CMACs are selectively allocated to the input sub-regions with large output errors. The resolutions of these CMACs are increased as required and only the finest resolution layers are kept.

On the other hand, an optimal PI_2 value (generalization) was achieved by the HCAQ-CMAC network with a memory size of 10 cells per-dimension. This HCAQ-CMAC network has produced a rather good fit to the actual blood glucose profile as indicated by a high correlation value of 97.48% and a relatively low RMSE of 10.4312 mg/ml of blood glucose concentration. There are only slight improvements in the HCAQ-CMAC performances as the network size is increased from 6 to 10 cells per dimension. However, Table 3.6 clearly shows that the generalization capability of the proposed HCAQ-CMAC network surpasses that of the basic CMAC network for all the evaluated memory sizes. Therefore, depending on the memory consideration and the desired modeling accuracy, a HCAQ-CMAC network with memory size of 6 to 10 cells per-dimension can be chosen for the modeling task. From Table 3.6, one can observe that the three-layers MMR-CMAC achieves comparable performance to the HCAQ-CMAC network, while employing lesser memory cells. However, the overlay structure of the MMR-CMAC requires each layer of the network to be trained individually, thereby increasing the training time required. Furthermore, this overlay structure also makes output interpretation as well as hardware implementation awkward and more difficult as compared to the single-layer HCAQ-CMAC structure. In addition, the proposed HCAQ-CMAC network achieved comparable recall and generalization performances, which demonstrates that the network is able to efficiently extract the inherent relationships from the training data.

To further analyze the performance of the HCAQ-CMAC network, a three-day modeling

(a) Generalization performance of CMAC network (size = 10 cells per-dimension)

(b) Generalization performance of HCAQ-CMAC network (size = 10 cells per-dimension)

Figure 3.11: Modeling results of the CMAC and HCAQ-CMAC network for the glucose metabolic process of Subject A

result of the CMAC and HCAQ-CMAC networks (each of size 10 cells per-dimension) are depicted in Figure 3.11. Figure 3.11 clearly demonstrated the generalization and modeling accuracy of the HCAQ-CMAC network. Due to the nature of the training data, the static uniform quantization of the basic CMAC network resulted in an overwhelming number of untrained network cells as evidenced in Figure 3.11(a) (the effects of the untrained network cells are highlighted as **A**, **B** and **C** respectively). The HCAQ-CMAC network, on the other

hand, is able to selectively allocate the available memory space according to the information distribution of the training data, and thus significantly reduces the number of untrained network cells to facilitate a consistent modeling performance as evident in Figure 3.11(b).

3.6 Summary

This chapter presents a novel brain-inspired cerebellar architecture named the HCAQ-CMAC network. The proposed HCAQ-CMAC network employs the adaptive memory allocation approach of the self-organizing cerebellar memory framework (see Chapter 1) to non-uniformly quantize its computing (memory) cells. Inspired by the physiology of the human cerebellum, as well as the neuro-biological mechanisms of the neuronal selection process underlying human brain development, the HCAQ-CMAC network employs an information-driven memory allocation scheme. The proposed HCAQ-CMAC network extends from the basic CMAC model by employing a hierarchical clustering technique to selectively allocate more memory cells to the input regions that contain more training information as reflected by the variations in the output values. This translates to a finer output resolution in the critical regions of the input space. The mathematical description of the structural and the subsequent parameter learning process, together with the theoretical proof of HCAQ-CMAC learning stability, are presented in this chapter.

The performance of the proposed HCAQ-CMAC network was subsequently evaluated using two real-life case studies, namely: the automatic control of car maneuver and the modeling of human blood glucose dynamics. Simulation results have sufficiently demonstrated the effectiveness of the proposed network architecture in capturing the complex input-output relationships for both applications. In particular, significant improvements in the generalization as well as the accuracy of the network output were achieved by the HCAQ-CMAC network. Moreover, HCAQ-CMAC also resulted in more efficient memory utilization than the benchmarked systems of CMAC, MMR-CMAC and TMR-CMAC as reflected by the considerably higher cell occupancy rates observed in the car-driving experiment.

In the next chapter, a newly-developed adaptive memory allocation process inspired by the neurophysiological aspects of cerebellar learning is presented. The proposed memory allocation scheme is subsequently integrated in a novel cerebellar computational model named the PSECMAC network.

*"Genius is one percent inspiration,
ninety-nine percent perspiration."
—Thomas A. Edison*

Chapter 4

The PSECMAC Architecture

In the previous chapter, the HCAQ-CMAC network was presented to overcome the three major drawbacks associated with the highly regularized computing structure of the basic CMAC network. The HCAQ-CMAC network employs a hierarchical clustering technique for the non-uniform quantization of the input-output (I/O) mapping space to identify significant input segments and to subsequently allocate more memory cells to these regions. However, the hierarchical clustering technique employed in its memory allocation process results in a few limitations of the HCAQ-CMAC network. Firstly, the HCAQ-CMAC memory cells are distributed based on the variation of the target output. Due to this operational principle, the resultant HCAQ-CMAC architecture is a Multi-Input-Single-Output (MISO) system. Moreover, the performance of the HCAQ-CMAC network may be sensitive to the definition of the cost function employed in the quantization process. Finally, the use of the hierarchical clustering technique in the memory quantization process also renders the HCAQ-CMAC network less suitable for classification tasks.

Hence, a novel neurophysiologically-inspired multi-resolution associative memory network that is immune to the above-mentioned deficiencies is presented in this chapter. The proposed architecture, named *Pseudo Self-Evolving CMAC* (PSECMAC) network, is inspired by neuroscience research as well as human behavioral studies on the cerebellar learning process, where it has been shown that significantly higher densities of the cerebellar synaptic connections are located at the frequently-accessed regions of the cerebellum that are activated by repeated learning episodes (Federmeier et al., 2002). This cerebellar-based biological experience-driven synaptic plasticity phenomenon is emulated in the PSECMAC network by employing a data-driven adaptive memory quantization scheme for the deriva-

Figure 4.1: Comparison of CMAC and PSECMAC memory surface with neighborhood activation of the computing cells for a 2-dimensional input example

tion of its computing structure. That is, from a machine learning perspective, more memory cells are assigned to model regions of the data space that contain higher densities of the training exemplars.

Similar to the HCAQ-CMAC network, the proposed PSECMAC network consists of only a single-layer of computing cells. This attempts to enhance the computational comprehensibility of the PSECMAC network and to avoid the high memory requirement of the CMAC network in which extensive overlappings of the computing layers are required to achieve a smooth and accurate output. However, unlike the HCAQ-CMAC network that employs a winner-take-all computation, a *neighborhood activation* of computing cells is employed in the learning and computation of the PSECMAC output. The PSECMAC network retains the computational principles of the multi-layered CMAC counterpart via a neighborhood activation of its single layer of computing cells to facilitate: (1) smoothing of the computed output; (2) implementation of distributed learning paradigm; and (3) activation of highly correlated computing cells in the input space. Such a neighborhood activation is similar to the single layer model of the CMAC network presented in Appendix B. The neighborhood-based activation process is also depicted in Figure 4.1.

Figure 4.1 graphically illustrates the fundamental architectural difference in the organization of the memory (computing) cells to define the I/O mapping space between the proposed

PSECMAC network and the CMAC model. In CMAC, the memory cells are uniformly distributed over the entire associative (memory) space. The computing cells in the PSECMAC network, on the other hand, are intentionally assigned to create an efficient representation of the data distribution.

The rest of the chapter is organized as follows. In section 4.1, the neurophysiological motivations that inspire the development of the PSECMAC architecture is briefly discussed. Section 4.2 presents the details on the PSECMAC's structural learning and parameter tuning processes. In Section 4.3, the PSECMAC neural correlates are presented. The proof for learning convergence of the PSECMAC network is established in Section 4.4. Subsequently, the performance of the PSECMAC network is evaluated on two real-life case-studies, namely: (1) the pricing and arbitrage trading of the GBP vs. USD currency futures options (Section 4.5); and (2) the prediction and classification of US banking failures (Section 4.6). Section 4.7 summarizes this chapter.

4.1 PSECMAC's Neurophysiological Motivations

Findings from the neurobiological and neurophysiological research have provided an insight into the cerebellum's adaptive mechanism of learning and skill acquisition. As described in Chapter 2 of this Thesis, the cerebellar learning mechanism is facilitated by the synaptic plasticity and the structural plasticity of its neuronal connections. More importantly, the experience-driven cerebellar structural plasticity phenomenon (please refer to Section 2.1 of Chapter 2) suggests that the cerebellum organizes its learned knowledge in an adaptive and non-linear manner, where repeated training (exposures to a particular input-output mapping association tuple) yields an increase in the synaptic connections as well as finer calibrations in the neural circuitry of the Purkinje cells (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). Such a synaptic adaptation process results in the biological formation of a more precise knowledge representation scheme.

In addition, neuroscience research has also established that the human cerebellum adopts an error-correction-driven supervised learning paradigm (Kandel et al., 2000b). This implies that cerebellar learning requires extended trials with repeated exposures to similar sequence of movements in order to achieve a finely calibrated mapping between the intended and actual execution of motor movements. The existence of microzones (Doya, 1999; Voogd

and Glickstein, 1998), as well as the established role of the climbing fibers as the teaching signals to the Purkinje cells, suggests that the cerebellar circuitry performs neighborhood-based training of the synaptic weights. That is, the cerebellar input (motor commands and sensory signals) and output (corrective error signals) pairing of a learning episode alters the synaptic weights of a cluster of Purkinje cells that is topographically defined by the corresponding microzones. This neighborhood-based learning mechanism enables a faster convergence of the cerebellar learning process, and underlies the generalization of skill learning in everyday life.

These neurophysiological properties are functionally incorporated into the PSECMAC architecture. The structural formation of the proposed PSECMAC network is inspired by the experience-driven cerebellar adaptation mechanism, in which significantly higher densities of the cerebellar synaptic connections are located at the high-throughput regions of the cerebellum where frequent access facilitates the acquisition and execution of skilled behavioral responses in everyday life. This is achieved via a data-driven memory allocation process. The neighborhood-based activation of PSECMAC computing cells in the network learning and computation processes emulates the existence of microzones in the cerebellar circuitry. Finally, the PSECMAC network employs a modified Widrow-Hoff learning algorithm to implement the incremental supervised training of its network weights.

4.2 The PSECMAC Network

The PSECMAC network is a single-layered self-organizing multi-resolution computational model of the cerebellum that employs a data-driven adaptive memory quantization scheme. The experience-based synaptic adaptation process observed in the human cerebellum is emulated via the use of the Pseudo Self-Evolving Cerebellar (PSEC) (Ang and Quek, 2005b) clustering technique that determines the data density profile of the training exemplars along each input dimension. The memory quantization step sizes of the PSECMAC network (and hence the placements of its computing cells) are subsequently adapted based on the computed information distribution of the training data.

Figure 4.2 depicts a 2D example of the data density-based multi-resolution architecture of the PSECMAC network. In the PSECMAC network, memory efficiency is enhanced by allocating more memory (computing) cells to the densely data-populated regions of the I/O

Figure 4.2: An example of data-driven quantization of 2D PSECMAC memory cells

associative space. This subsequently lowers memory wastage as the number of untrained (unused) computing cells is reduced. Meanwhile, the accuracy of the PSECMAC network's output in the high-throughput I/O subspaces (which conceptually correspond to the often-accessed regions of the cerebellar cortex) is simultaneously enhanced with a higher modeling resolution. The structural formation and network operations of the proposed PSECMAC architecture are described in the following subsections.

4.2.1 The PSECMAC Structural Learning Process

The proposed PSECMAC network employs a two-phased learning process, namely: structural learning and parameter tuning. The objective of the structural learning phase is to create the PSECMAC network's associative structure by computing the quantization decision functions for each input dimension. Subsequently, the input to output associative information of the training data samples are learnt by adapting the memory contents of the PSECMAC network in the parameter tuning phase.

The initial step in the PSECMAC structural learning phase is to identify the regions of the I/O space with high data densities. Subsequently, more memory cells (i.e. a finer network output granularity) are assigned to these regions to emulate the experience-driven dendritic arborization phenomenon observed in the cerebellar learning process during skill acquisition (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). Analogical to the repeated exposures of the learning episodes during skill acquisition, these identified regions of the I/O space contain a large amount of training data points that coexisted in close proximity. The PSECMAC memory allocation and non-uniform quantization process is performed individually for each input dimension and consisted of several steps: (1) the identification of the data density clusters; (2) the allocation of the PSECMAC memory cells based on the computed density profile; and (3) the derivation of the respective PSECMAC quantization decision functions.

The computation of data density clusters

In the proposed PSECMAC network, significant data clusters supporting the inherent organization of the training dataset are first identified via the Pseudo Self-Evolving Cerebellar (PSEC) (Ang and Quek, 2005b) clustering algorithm through an analysis of the density distribution of the training data points along each input dimension. The PSEC algorithm is a density-based clustering algorithm which synergizes the merits of the incremental learning procedure of the *Learning Vector Quantization* (LVQ) (Kohonen, 1989) technique with the effectiveness of the density-based partitioning method of the DBSCAN algorithm (Ester et al., 1996). This clustering algorithm is inspired by the biological development of the human brain where neural cell death plays an integral part in the refinement process of the brain's neuronal organization (Ang and Quek, 2005b). Neurophysiological studies have established that there are two overlapping stages in the development of the human brain (Kandel et al., 2000b). The first stage of this development process encompasses the formation of the basic architecture of the brain system, in which coarse connection patterns emerge as a result of the genesis of the brain cells during prenatal development. Subsequently, in the second stage of the brain's development, the initial architecture is refined and extraneous synaptic connections are pruned throughout an individual's life-span via exposures to various activity-dependent experiences. These two stages of the human brain adaptation process are functionally emulated by the PSEC clustering algorithm.

The proposed PSECMAC network employs a modified PSEC (MPSEC) clustering algorithm to identify the centers of the density clusters along each input dimension of the training data space. A *density cluster* is defined as a cluster identified from the data density profile computed with the MPSEC algorithm. Each density cluster is associated with a *cluster center*, which denotes the point of highest data density in the cluster. The MPSEC algorithm commences with an initial set of regularly-spaced density clusters, with the mid points of these initial density clusters defined as the respective cluster centers. This initial set of clusters is incrementally *evolved* to capture the data density profile along each input dimension to derive a final set of density clusters. The LVQ iterative algorithm is subsequently employed to refine the positions of the cluster centers in this final set of density clusters.

Let J and L denote the total number of input and output dimensions for a given dataset respectively. Assume that a training dataset of $\mathbf{U} = \{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \dots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \dots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\}$ is used to train the PSECMAC network, where $\mathbf{X}_s = [x_{s,1} \ x_{s,2} \ \dots \ x_{s,J}]^T$ denote the s^{th} input training vector, and $\hat{\mathbf{Y}}_s = [\hat{y}_{s,1} \ \hat{y}_{s,2} \ \dots \ \hat{y}_{s,L}]^T$ denote the corresponding expected output target vector of the PSECMAC network. Let τ denote the clustering iteration in MPSEC and $\mathbf{C}_j^{(\tau)} = \{C_{j,1}^{(\tau)}, C_{j,2}^{(\tau)}, \dots, C_{j,n}^{(\tau)}, \dots, C_{j,n_{C,j}^{(\tau)}}^{(\tau)}\}$ denote the set of density clusters along the j^{th} input dimension at the τ^{th} iteration, where $n_{C,j}^{(\tau)}$ is the corresponding total number of density clusters. Let $\mathbf{C}_j^{(-1)}$ denote the initial set of density clusters for the MPSEC algorithm and $n_{C,j}^{(-1)}$ be the number of these regularly-spaced density clusters along the j^{th} input dimension. For each input dimension $j \in \{1 \dots J\}$, the MPSEC clustering algorithm is briefly outlined as follows:

Step 1 *Initialize the clustering parameters.*

An initial number of density clusters $n_{C,j}^{(-1)}$, together with a pseudo potential threshold β , a clustering termination criterion ε and the LVQ learning constant α_c are predefined prior to the start of MPSEC clustering iteration.

Step 2 *Construct the initial set of density clusters.*

The initial set of density clusters $\mathbf{C}_j^{(-1)}$ is subsequently constructed with $n_{C,j}^{(-1)}$ regularly spaced clusters such that $\mathbf{C}_j^{(-1)} = \{C_{j,1}^{(-1)}, C_{j,2}^{(-1)}, \dots, C_{j,n}^{(-1)}, \dots, C_{j,n_{C,j}^{(-1)}}^{(-1)}\}$. Each density cluster $C_{j,n}^{(\tau)}$ is associated with a cluster center $P_{j,n}^{(\tau)}$ and a density value $V_{j,n}^{(\tau)}$. In the initial set of density clusters $\mathbf{C}_j^{(-1)}$, the cluster center $P_{j,n}^{(-1)}$ is assigned to be the mid-point of the corresponding density cluster $C_{j,n}^{(-1)}$ and the

density value $V_{j,n}^{(-1)}$ is initialized to zero. This step emulates the formation of the initial brain system, in which extraneous connection patterns emerge as a result of the overproduction of neurons during the prenatal brain development phase.

Step 3 Compute the initial cluster density values.

MPSEC performs structural learning by executing a one-pass learning of the density values $V_{j,n}^{(-1)}$ to obtain a density distribution of the training data along the j^{th} input dimension.

Step 4 Evolve the initial set of density cluster.

For each input dimension, the initial set of density clusters $C_j^{(-1)}$ is evolved to capture the inherent data density profile by identifying all the local maxima in the set of computed density values $V_{j,n}^{(-1)}$. This step emulates the competitive neuronal selection process in human brain development, whereby neurons with high tropic factors are identified as the winning neurons and the remaining extraneous neurons are pruned to create a more refined structure of synaptic connections. The clusters in the initial set of density clusters $C_j^{(-1)}$ whose density values form prominent convex density peaks in the computed density distribution of Step 3 are included in the new set of density clusters $C_j^{(0)}$. The rest of the density clusters are removed (pruned). The clusters in the new set of density clusters $C_j^{(0)}$ are therefore analogous to the surviving neurons with high tropic factors in the brain neuronal selection process.

Step 5 Incremental learning of cluster centers.

The cluster centers $P_{j,n}^{(0)}$ of the new set of density clusters $C_j^{(0)}$ are subsequently refined to derive the accurate positioning of the density-induced cluster centers. The incremental learning of the cluster centers are performed iteratively using the LVQ algorithm (i.e. $\tau = \{1 \cdots \tau_{\max}\}$), resulting in the final set of density clusters $C_j^{(\tau_{\text{end}})}$, where τ_{end} denotes the last LVQ iteration performed and $1 \leq \tau_{\text{end}} \leq \tau_{\max}$.

Step 6 Compute the resultant density profile.

A one-pass learning of the density values $V_{j,n}^{(\tau_{\text{end}})}$ in the final set of density clusters $C_j^{(\tau_{\text{end}})}$ is performed to derive the density values at the final cluster centers $P_{j,n}^{(\tau_{\text{end}})}$. Finally, for each density cluster $C_{j,n}^{(\tau_{\text{end}})}$ in $C_j^{(\tau_{\text{end}})}$, the left and right cluster boundary $L_{j,n}^{(\tau_{\text{end}})}$ and $R_{j,n}^{(\tau_{\text{end}})}$ are defined as the mid-point between the cluster center $P_{j,n}^{(\tau_{\text{end}})}$ and the cluster centers of its corresponding left and right neighbors.

Figure 4.3: A sample output of the MPSEC clustering technique

Figure 4.3 illustrates the mechanism of the MPSEC clustering algorithm. Essentially, MPSEC computes a set of density-induced clusters, whose centers denote the highest density points in the respective clusters. The boundary between any two neighboring clusters is assumed to be at the mid-point of the two respective cluster centers. The detailed description and mathematical formulations of the MPSEC algorithm is listed as Appendix D.

The PSECMAC Memory Allocation Process

In the proposed PSECMAC network, the number of memory cells allocated to a density cluster is proportional to the normalized density value of the corresponding cluster center. Let $\hat{M}_j$ denote the total number of user predefined memory cells in the j^{th} input dimension. Then for each density cluster $C_{j,n}^{(\tau_{\text{end}})}$ in $C_j^{(\tau_{\text{end}})}$, the number of memory cells $M_{j,n}$ allocated to this cluster is computed using eq. (4.1)

$$M_{j,n} = \left\lfloor \frac{V_{j,n}^{(\tau_{\text{end}})}}{\sum_{n' \in \{1 \dots n_{C,j}^{(\tau_{\text{end}})}\}} V_{j,n'}^{(\tau_{\text{end}})}} \right\rfloor \times \hat{M}_j \quad (4.1)$$

where $M_{j,n}$ is the number of memory cells allocated to cluster $C_{j,n}^{(\tau_{\text{end}})}$ in the j^{th} input dimension, $V_{j,n}^{(\tau_{\text{end}})}$ is the density value of $C_{j,n}^{(\tau)}$, and $n_{C,j}^{(\tau_{\text{end}})}$ denotes the total number of computed density clusters in the j^{th} input dimension.

The PSECMAC Quantization Decision Functions

For the proposed PSECMAC network, a non-linear assignment scheme is introduced for

Figure 4.4: Adaptive memory cell distribution in cluster $C_{j,n}^{(\tau_{end})}$

the computation of the quantization decision functions to vary the quantization step sizes of the memory cells of the identified density clusters. In PSECMAC, the memory cells allocated to an arbitrary cluster $C_{j,n}^{(\tau_{end})}$ is equally distributed to the left and right side of the cluster center (i.e. the left and right subregions). In each of the two subregions, the quantization point of each memory cell is logarithmically assigned with respect to the cluster center. The quantization point of a memory cell is defined as the midpoint of the memory cell. The result of this computation is illustrated in Figure 4.4, which depicts the adaptively quantized memory cells inside a cluster. Computationally, the center of each density-induced cluster constitutes the finest data granularity. As a memory cell moves away from the cluster center, its quantization step size increases in response to a lower density of the observed training data.

In this work, a logarithmic quantization technique (commonly referred to as μ -law quantization (Orfanidis, 1995)) is employed to manage the distribution of the memory cells in a surviving cluster. The degree of non-linearity in the quantization step sizes of the memory cells is governed by a parameter μ . Subsequently, a quantization mapping function $\mathbf{Q}_j[\cdot] \rightarrow \{Q_{j,1}, Q_{j,2}, \dots, Q_{j,\hat{M}_j}\}$ is constructed to define the quantization of the memory cells in the j^{th} input dimension of the PSECMAC network, where $Q_{j,\bar{n}}$, $\bar{n} \in \{1 \dots \hat{M}_j\}$, denote the $\bar{n}^{th}$ quantization point. The derivation of the quantization function $\mathbf{Q}_j[\cdot]$ is described as follows:

- (a) Initialize $\bar{n} = 1$ (this is the first quantization point) and define the parameter $\mu \geq 0$ for the non-linear distribution of the memory cells, where $\mu = 0$ signifies a linear

quantization scenario. A larger μ value implies a higher degree of non-linearity in the distribution of the PSECMAC memory cells.

- (b) For $n = 1 \dots n_{C,j}^{(\tau_{\text{end}})}$: Let $M_{j,n}$ (computed from eq. (4.1)) denote the number of memory cells allocated to a density cluster $C_{j,n}^{(\tau_{\text{end}})}$ in the j^{th} input dimension and k be the index to the memory cells in the density cluster $C_{j,n}^{(\tau_{\text{end}})}$. For $k = 1 \dots M_{j,n}$, compute the quantization point for the k^{th} memory cell in $C_{j,n}^{(\tau_{\text{end}})}$ such that:

- IF the k^{th} memory cell is in the left subregion (i.e. $k \leq \lfloor \frac{M_{j,n}}{2} \rfloor$) THEN:

$$\text{step} = \frac{P_{j,n}^{(\tau_{\text{end}})} - L_{j,n}^{(\tau_{\text{end}})}}{\lfloor \frac{M_{j,n}}{2} \rfloor} \quad (4.2)$$

$$pt = L_{j,n}^{(\tau_{\text{end}})} + (k - 0.5) \cdot \text{step} \quad (4.3)$$

$$Q_{j,\bar{n}} = L_{j,n}^{(\tau_{\text{end}})} - \left[\frac{\left(P_{j,n}^{(\tau_{\text{end}})} - L_{j,n}^{(\tau_{\text{end}})} \right) \cdot \log \left(1 + \frac{\mu \cdot |L_{j,n}^{(\tau_{\text{end}})} - pt|}{(P_{j,n}^{(\tau_{\text{end}})} - L_{j,n}^{(\tau_{\text{end}})})} \right)}{\log(1 + \mu)} \right] \quad (4.4)$$

where $P_{j,n}^{(\tau_{\text{end}})}$ and $L_{j,n}^{(\tau_{\text{end}})}$ are the center and the left boundary of the density cluster $C_{j,n}^{(\tau_{\text{end}})}$ respectively. Update the index $\bar{n} = \bar{n} + 1$.

- ELSE IF $M_{j,n}$ is odd and the k^{th} memory cell is assigned to the cluster center (i.e. $\lfloor \frac{M_{j,n}}{2} \rfloor < k < \lfloor \frac{M_{j,n}}{2} \rfloor + 1$) THEN:

$$Q_{j,\bar{n}} = P_{j,n}^{(\tau_{\text{end}})} \quad (4.5)$$

Update the index of the current decision point $\bar{n} = \bar{n} + 1$.

- ELSE IF the k^{th} memory cell is in the right subregion (i.e. $k > \lfloor \frac{M_{j,n}}{2} \rfloor$) THEN:

$$\text{step} = \frac{R_{j,n}^{(\tau_{\text{end}})} - P_{j,n}^{(\tau_{\text{end}})}}{\lfloor \frac{M_{j,n}}{2} \rfloor} \quad (4.6)$$

$$pt = P_{j,n}^{(\tau_{\text{end}})} + \left(k - \lfloor \frac{M_{j,n}}{2} \rfloor - 0.5 \right) \cdot \text{step} \quad (4.7)$$

$$Q_{j,\bar{n}} = R_{j,n}^{(\tau_{\text{end}})} + \left[\frac{\left(R_{j,n}^{(\tau_{\text{end}})} - P_{j,n}^{(\tau_{\text{end}})} \right) \cdot \log \left(1 + \frac{\mu \cdot |pt - R_{j,n}^{(\tau_{\text{end}})}|}{(R_{j,n}^{(\tau_{\text{end}})} - P_{j,n}^{(\tau_{\text{end}})})} \right)}{\log(1 + \mu)} \right] \quad (4.8)$$

where $R_{j,n}^{(\tau_{\text{end}})}$ is the right boundary of the density cluster $C_{j,n}^{(\tau_{\text{end}})}$. Update the index $\bar{n} = \bar{n} + 1$.

Note that the second condition (eq. (4.5)) is met only when $M_{j,n}$ is odd. Otherwise, the number of allocated memory cells to the left and right subregions of cluster $C_{j,n}^{(\tau_{\text{end}})}$ is equal to $\left\lfloor \frac{M_{j,n}}{2} \right\rfloor$.

After the completion of this placement process, the quantization mapping function $\mathbf{Q}_j[\cdot]$ is defined for the j^{th} input dimension. The computed quantization decision points of each input dimension subsequently form the memory axes of the proposed PSECMAC network and are used to define its overall computing structure. The intersections of these memory axes denote the computing cells of the PSECMAC network and define the I/O associative space. A flowchart and computational analysis of the PSECMAC network formation is depicted in Figure 4.5. The three steps involved in the PSECMAC structural learning process, namely: (1) the computation of the data density clusters; (2) the PSECMAC memory allocation process; and (3) the PSECMAC quantization decision functions; are highlighted as the numbered boxes ❶–❸ in the figure. As shown in Figure 4.5, the worst-case complexity of the PSECMAC structural learning process is in the order of $O(Jn_{c,j}^{(-1)}) + O(JS\tau_{\text{max}}) + O(J\hat{M}n_{c,j}^{\text{end}}) + O(\hat{M}^J)$, where J denote the total number of PSECMAC input dimensions, $\hat{M}$ is the total number of PSECMAC memory cells per dimension, S is the total number of training samples used to construct the network, τ_{max} denote the maximum number of LVQ iteration, $n_{c,j}^{(-1)}$ is the initial number of density clusters, and $n_{c,j}^{\text{end}}$ is the final number of PSECMAC density clusters. A detailed flowchart and the analysis of computational complexity of the MPSEC density clustering is given in Appendix D. The training of this PSECMAC computing structure is described in the following two sections.

4.2.2 The PSECMAC Network Computational Process

The PSECMAC network employs a *Weighted Gaussian Neighborhood Output* (WGNO) computation process, where a set of neighborhood-bounded computing cells is activated to derive the network's output response to the given input stimulus. In this computation process, each of the neighborhood cells has a weighted degree of activation that is inversely proportional to the distance of the cell from the input stimulus point. The objective of the WGNO scheme is to minimize the influences of the input quantization errors on the com-

Figure 4.6: An example of a 2D PSECMAC neighborhood

puted network output. In addition, it introduces a "smoothing" effect on the PSECMAC output and enhances the generalization capability of the PSECMAC network.

Let $\mathbf{Y}_s$ denotes the computed PSECMAC network output for an input stimulus $\mathbf{X}_s = [x_{s,1}, x_{s,2}, \dots, x_{s,J}]^T$ to the PSECMAC network. The WGNO computation process is defined as follows:

Step 1 *Determine the region of activation.*

The size of the activated PSECMAC neighborhood with respect to input $\mathbf{X}_s$ is defined by $N \in [0 \dots 1]$. This is a user-specified parameter that governs the relative size of the neighborhood of activated PSECMAC cells to the overall memory space. The neighborhood activation boundaries are defined on per-dimension basis such that $N = 0.2$ denotes an activation boundary of 20% relative to the ranges of the respective input dimensions. A neighborhood constant of $N = 0.2$ therefore signifies a neighborhood activation of $(0.2)^J \times 100\%$ relative to the entire input space, where J denotes the total number of input dimensions. For the input stimulus $\mathbf{X}_s$, its activation neighborhood is defined by eqs. (4.9) and (4.10)

$$lb_{s,j} = x_{s,j} - 0.5 \cdot N \cdot range_j \quad (4.9)$$

$$rb_{s,j} = x_{s,j} + 0.5 \cdot N \cdot range_j \quad (4.10)$$

$$j \in \{1, 2, \dots, J\}$$

where $lb_{s,j}$ denotes the left activation boundary, $rb_{s,j}$ denotes the right activation boundary, and $range_j$ is the domain for the j^{th} input dimension. Subsequently, the memory cells encapsulated within the neighborhood defined by the computed boundaries are activated in response to the input stimulus $\mathbf{X}_s$. A PSECMAC activation neighborhood is illustrated as Figure 4.6. The size of the neighborhood affects the accuracy of the computed PSECMAC output. The larger the neighborhood size, the more generalized is the output of the PSECMAC network. Conversely, a smaller neighborhood size results in a more accurate output computation. Therefore, a larger neighborhood size is suitable for a dataset that is sparse in the input space as this increases the generalization ability of the PSECMAC network. A smaller neighborhood size, on the other hand, is suitable for a compact dataset so as to produce more accurate results.

Step 2 Compute the Gaussian weighting function.

A Gaussian weighting factor g_k is associated with each activated PSECMAC cell to determine its contribution towards the computation of the network output. The Gaussian weighting factor is defined by eq. (4.11)

$$g_k = (1 - d_k)e^{-d_k^2/2\gamma^2} \quad (4.11)$$

where γ is the Gaussian width constant and d_k denotes the normalized Euclidean distance from the k^{th} activated cell to the input stimulus $\mathbf{X}_s$ (see Figure 4.6). Let $\mathbf{K}_s$ be the set of activated PSECMAC cells in the computed neighborhood. Subsequently, d_k is defined by eq. (4.12)

$$d_k = \frac{\|\mathbf{Q}_k - \mathbf{X}_s\|}{\max_{k' \in \mathbf{K}_s} \|\mathbf{Q}_{k'} - \mathbf{X}_s\|} \quad (4.12)$$

where $\mathbf{Q}_k = [Q_{1,k}, Q_{2,k}, \dots, Q_{J,k}]$ denotes the quantization point of cell k in the memory space.

Step 3 Retrieve the PSECMAC Output.

The PSECMAC network output $\mathbf{Y}_s$ is computed as a weighted linear combination of the memory contents of the activated cells and is described by eq. (4.13)

$$\mathbf{Y}_s = \frac{\sum_{k \in \mathbf{K}_s} (g_k \cdot \mathbf{W}(k))}{\sum_{k \in \mathbf{K}_s} g_k} \quad (4.13)$$

where $\mathbf{K}_s$ denotes the set of neighborhood-activated PSECMAC cells, and $\mathbf{W}(k)$ is the stored weight value(s) of the activated PSECMAC cell with index k .

4.2.3 The PSECMAC Learning Process

This section describes the parameter tuning phase of the proposed PSECMAC network. Parameter tuning is performed for the PSECMAC network to learn the mapping of the input-output associative patterns from the training data tuples. To emulate the neighborhood learning phenomenon of the human cerebellum (Kandel et al., 2000b; Voogd and Glickstein, 1998), the PSECMAC network adopts a *Weighted Gaussian Neighborhood Update* (WGNU) process. WGNU combines the Widrow-Hoff training algorithm (Widrow and Stearns, 1985) with the Gaussian weighting function defined in eq. (4.11). The objective of this neighborhood update scheme is to distribute the effect of learning to increase the generalization capability of the PSECMAC network.

For an arbitrary input-output training data tuple $(\mathbf{X}_s, \hat{\mathbf{Y}}_s)$, the PSECMAC learning process is mathematically described as follows:

1. Compute the PSECMAC output $\mathbf{Y}_s^{(i)}$ at the i^{th} training iteration using eq. (4.14)

$$\mathbf{Y}_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} (g_k \cdot \mathbf{W}^{(i)}(k))}{\sum_{k \in \mathbf{K}_s} g_k} \quad (4.14)$$

where $\mathbf{K}_s$ is the set of activated computing cells corresponding to the input $\mathbf{X}_s$, g_k is the Gaussian weighting factor of the k^{th} activated memory (computing) cell, $\mathbf{W}^{(i)}(k)$ denotes the memory content of the k^{th} activated memory cell at the i^{th} training iteration, and $\mathbf{Y}_s^{(i)}$ is the PSECMAC output to the input $\mathbf{X}_s$ at the i^{th} iteration.

2. Compute the network output error at the i^{th} iteration using eq. (4.15)

$$\mathbf{Err}_s^{(i)} = \hat{\mathbf{Y}}_s - \mathbf{Y}_s^{(i)} \quad (4.15)$$

where $\mathbf{Err}_s^{(i)}$ denotes the output error of the PSECMAC network to the input $\mathbf{X}_s$ at the i^{th} iteration, and $\hat{\mathbf{Y}}_s$ is the desired (target) output of the PSECMAC network in response to the input $\mathbf{X}_s$.

Figure 4.7: The flowchart and computational complexity of the PSECMAC network computational process: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of PSECMAC memory cells per dimension; N – PSECMAC neighborhood size

3. Update the stored network weights using eqs. (4.16) and (4.17)

$$\mathbf{W}^{(i+1)}(k) = \mathbf{W}^{(i)}(k) + \Delta \mathbf{W}^{(i)}(k), \quad k \in \mathbf{K}_s \quad (4.16)$$

$$\Delta \mathbf{W}^{(i)}(k) = \alpha \underbrace{\frac{g_k}{\sum_{k' \in \mathbf{K}_s} g_{k'}}}_{\text{local error for cell } k} \text{Err}_s^{(i)}, \quad k \in \mathbf{K}_s \quad (4.17)$$

where α is the learning constant, and $\mathbf{W}^{(i)}(k)$ denotes the content (weight) of the k^{th} activated cell in the neighborhood $\mathbf{K}_s$ in PSECMAC in response to the input stimulus $\mathbf{X}_s$ during the i^{th} training iteration. Similar to the HCAQ-CMAC learning process, the learning constant α governs the learning speed of the PSECMAC network. Please refer to Section 3.2.2 for the effects of α on the network learning process.

The PSECMAC memory learning phase commences with the computation of the network output corresponding to the input stimulus $\mathbf{X}_s$. A learning error is computed based on the derived PSECMAC output and the target response. This error is subsequently distributed to all the activated computing (memory) cells based on the Gaussian weighting factors. The

local errors are then used to update the memory contents of the activated cells.

Figure 4.7 shows a flowchart and the analysis of computational complexity of the PSECMAC computational process. From Figure 4.7, it can be observed that the worst case complexity of both the learning and recall scenario of the PSECMAC network for a particular input $\mathbf{X}_s$ is in the order of $O(J\hat{M}) + O((N\hat{M})^J)$, where J denote the total number of input dimensions, $\hat{M}$ is the number of memory cells per dimension, and N is the neighborhood size parameter of the PSECMAC network. This is because the PSECMAC network employs a neighborhood-based computation for both its learning and recall computations.

4.3 The PSECMAC Neural Correlates

As a computational model of the human cerebellum, the PSECMAC network exhibits properties that are highly similar to the neurobiological and neurophysiological aspects of its biological counterpart. The correspondence between the characteristics of the human cerebellum and the proposed PSECMAC network are listed in Table 4.1.

4.4 The PSECMAC Learning Convergence

This section presents the mathematical proof of the learning convergence of the proposed PSECMAC network. Figure 4.8 depicts an example of the memory surface $\mathbf{Z}$ of a 2-input PSECMAC network. $\mathbf{Z}(\mathbf{q1}, \mathbf{q2})$ denotes the network cell with the address index $(\mathbf{q1}, \mathbf{q2})$. With respect to Figure 4.8, the quantization points along the X_1 dimension are $\{Q_{1,1}, Q_{1,2}, Q_{1,3}, \dots, Q_{1,\hat{M}_1}\}$ and along the X_2 dimension are $\{Q_{2,1}, Q_{2,2}, Q_{2,3}, \dots, Q_{2,\hat{M}_2}\}$ respectively. However, for simplicity, equal memory size per dimension is assumed in the proof of convergence, i.e. $\hat{M}_1 = \hat{M}_2 = \dots = \hat{M}$, where $\hat{M}$ is the memory size per dimension.

4.4.1 Mathematical Perspective of The PSECMAC Network

The PSECMAC network employs the WGNO and WGNU computations for the network retrieval and update operations respectively and the activated neighborhoods of the two processes are centered at the corresponding multi-dimensional input vector to the network

Table 4.1: Correspondence between the neurophysiological aspects of the human cerebellum and the functionalities of the proposed PSECMAC network

Characteristics	The Cerebellum	The PSECMAC Network
Physical connectivity	Purkinje cells are the main computational units of the human cerebellum and the parallel fiber inputs run perpendicularly to the flat fan-like dendritic arborization of the Purkinje cells.	Memory cells of the PSECMAC network are analogous to the Purkinje cells in the human cerebellum and the grid-like organization of these memory cells is inspired by the anatomy of the biological interconnections of the Purkinje cells and the parallel fibers.
Output computation	The output of the cerebellum originates from the deep cerebellar nuclei, which combine the outputs of the activated Purkinje cells.	The output of the PSECMAC network is a linear combination of the weighted contents of the activated memory cells.
Functionality	The cerebellum performs associative mapping from the input sensory afferent and cerebral efferent signals to the output of the brain construct.	The PSECMAC network performs associative mapping from the input vector of the network to the output response.
Learning principle	The human cerebellum adopts an error-correction-driven supervised learning paradigm.	The PSECMAC network adopts the modified Widrow-Hoff training algorithm, which is essentially an error-correction-based supervised learning scheme.
Activation and adaptation	The existence of <i>microzones</i> in the human cerebellum signifies the activation and adaptation of highly correlated Purkinje cells.	The neighborhood-based cell activations in the PSECMAC network support the adaptation of the computing cells that are highly correlated in the input space.
Experience-driven learning	Cerebellar plasticity studies have demonstrated that motor skill training leads to an increase in the number of synapses in the cerebellar cortex, signifying finer wiring of the circuitry in the region of the cerebellum related to training.	The PSECMAC network employs a density-based memory allocation scheme in which more memory cells are allocated to the densely populated data regions that signify extensive training.
Learning convergence	Learning stability is vital since the cerebellum is responsible for smooth and precise coordination of the motor movements.	Learning stability (convergence) in the PSECMAC network has been established (Refer to Section 4.4).
Performance accuracy	Motor skill learning and repeated training result in enhanced performance of the human subject.	The PSECMAC network is an experience-driven model of the cerebellum, whereby repeated exposures to similar training patterns result in finer granularities of the quantization step sizes, and thus superior accuracy of the computed output.

Figure 4.8: An example of the 2-input PSECMAC memory contents

Figure 4.9: 2D PSECMAC $Y \rightarrow W$ mapping

(See Figure 4.6). The conceptual memory surface $\mathbf{Z}$ of a multi-input PSECMAC network can be expressed as a one-dimensional weight array $\mathbf{W}$. Figure 4.9 illustrates the *linearization* of the conceptual memory surface $\mathbf{Z}$ to the physically implemented one-dimensional weight array $\mathbf{W}$ for a two-dimensional PSECMAC example. With respect to the PSECMAC network, the computed output for the s^{th} input sample at the i^{th} learning iteration is defined by eq. (4.18)

$$\mathbf{Y}_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} (g_k \cdot \mathbf{W}^{(i)}(k))}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \quad (4.18)$$

where $\mathbf{Y}_s^{(i)}$ is the computed output of the multi-dimensional PSECMAC to the s^{th} input training vector during the i^{th} iteration, $\mathbf{K}_s$ is the set of activated computing cells corresponding to the s^{th} input training vector, g_k is the Gaussian weighting factor of the k^{th} activated memory (computing) cell, and $\mathbf{W}^{(i)}(k)$ denotes the content of the k^{th} activated memory cell at the i^{th} iteration.

For simplicity, only scalar output is considered here. That is, the PSECMAC network presented here for the proof of learning convergence is a multi-input single-output (MISO) system. However, this proof can be readily extended to a multi-input multi-output (MIMO) PSECMAC system without any loss of generality. This is because a MIMO PSECMAC system is simply a parallel combination of multiple MISO PSECMAC systems. The computed output of the single output PSECMAC system is expressed as eq. (4.19).

$$y_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} (g_k \cdot \mathbf{W}^{(i)}(k))}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \quad (4.19)$$

where $y_s^{(i)}$ denotes the computed output of the single-output PSECMAC network with respect to the s^{th} input training sample at the i^{th} training iteration.

Assume a training dataset $\mathbf{U}$ of S tuples, i.e. $\mathbf{U} = \{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \dots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \dots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\}$. Let the total number of memory cells in the PSECMAC network be $\hat{M}^J$ where J is the total number of input dimensions and the column vector $\mathbf{A}_s$ denotes the activation mask of the PSECMAC memory cells with respect to the s^{th} input training sample.

That is,

$$\mathbf{A}_s^T = \underbrace{[a_{s,1} \ a_{s,2} \ \dots \ a_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \quad (4.20)$$

$$a_{s,j} = \frac{g_j}{\sum_{k \in \mathbf{K}_s} g_k}, \quad j \in \{1 \dots \hat{M}^J\} \quad (4.21)$$

$$g_j = 0, \quad \text{if } j \notin \mathbf{K}_s \quad (4.22)$$

where $\mathbf{K}_s$ denotes the set of activated memory cells in the neighborhood selected by the s^{th} input training sample. Hence, the mask $\mathbf{A}_s^T$ identifies the activated memory cells corresponding to the neighborhood selected by the s^{th} input training sample. Subsequently, the respective elements in the mask $\mathbf{A}_s^T$ weights these cells using the Gaussian neighborhood function defined in eq. (4.11). The scalar output of the PSECMAC network can thus be formulated as a vector multiplication described by eq. (4.23)

$$y_s = \underbrace{[a_{s,1} \ a_{s,2} \ \dots \ a_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \mathbf{W}_s = \mathbf{A}_s^T \mathbf{W}_s \quad (4.23)$$

where $\mathbf{W}_s$ is the memory content of the entire PSECMAC network structure at the time

when the s^{th} input training sample is presented.

The memory update operation of the PSECMAC network for the s^{th} input training sample is defined by eq. (4.24)

$$\mathbf{W}_{s+1}^{(i)} = \mathbf{W}_s^{(i)} + \underbrace{\Delta \mathbf{W}_s^{(i)}}_{\alpha \times \text{local error}} = \mathbf{W}_s^{(i)} + \underbrace{\alpha \mathbf{A}_s \overbrace{\{\hat{y}_s - \mathbf{A}_s^T \mathbf{W}_s^{(i)}\}}^{\text{learning error}}}_{\text{local error}} \quad (4.24)$$

where $\mathbf{W}_{s+1}^{(i)}$ denotes the memory content of the entire PSECMAC network structure when the $(s+1)^{th}$ training sample is presented in the i^{th} training iteration, α is the learning constant, $\mathbf{A}_s$ is the activation mask of the PSECMAC memory cells, and $\hat{y}_s$ denotes the desired (expected) PSECMAC output for the s^{th} training sample.

The difference of the PSECMAC memory contents between two successive iterations for the s^{th} input training sample (denoted as $\mathbf{Dw}_s^{(i)}$) is therefore defined by eq. (4.25).

$$\begin{aligned} \mathbf{Dw}_s^{(i)} &= \mathbf{W}_s^{(i+1)} - \mathbf{W}_s^{(i)} \\ &= \underbrace{\mathbf{W}_{s-1}^{(i+1)} + \Delta \mathbf{W}_{s-1}^{(i+1)}}_{\mathbf{W}_s^{(i+1)}} - \underbrace{(\mathbf{W}_{s-1}^{(i)} + \Delta \mathbf{W}_{s-1}^{(i)})}_{\mathbf{W}_s^{(i)}} \\ &= \underbrace{\mathbf{Dw}_{s-1}^{(i)}}_{\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)}} + \underbrace{\alpha \mathbf{A}_{s-1} \{\hat{y}_{s-1} - \mathbf{A}_{s-1}^T \mathbf{W}_{s-1}^{(i+1)}\}}_{\Delta \mathbf{W}_{s-1}^{(i+1)}} - \underbrace{\alpha \mathbf{A}_{s-1} \{\hat{y}_{s-1} - \mathbf{A}_{s-1}^T \mathbf{W}_{s-1}^{(i)}\}}_{\Delta \mathbf{W}_{s-1}^{(i)}} \\ &= \mathbf{Dw}_{s-1}^{(i)} - \alpha \mathbf{A}_{s-1} \mathbf{A}_{s-1}^T \underbrace{(\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)})}_{\mathbf{Dw}_{s-1}^{(i)}} \\ &= (\mathbf{I} - \underbrace{\alpha \mathbf{A}_{s-1} \mathbf{A}_{s-1}^T}_{\text{outer product}}) \mathbf{Dw}_{s-1}^{(i)} \end{aligned} \quad (4.25)$$

Note that the activation mask $\mathbf{A}_s$ is constant for an arbitrary given input training sample s across different training iterations. This is because the PSECMAC network structure is fixed after the structural learning phase. Following eq. (4.25), let

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T) \quad \text{where } \mathbf{E}_s \text{ is } \hat{M}^J \times \hat{M}^J \text{ matrix; and} \quad (4.26)$$

$$\mathbf{Dw}^{(i)} \equiv \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \quad (4.27)$$

where S denotes the total number of input training samples.

The learning convergence of the PSECMAC network is established via the convergence of the network memory contents as training approaches infinity (Lin and Chiang, 1997; Ting, 2004). In this case, the sufficient and necessary condition for the PSECMAC learning process to convergence can be expressed by eqs. (4.28) and (4.29), where $[0]$ is the null matrix.

$$\lim_{i \rightarrow \infty} \mathbf{Dw}_s^{(i)} = [0], \quad \forall s \in \{1 \cdots S\}, \quad \text{or in matrix notation} \quad (4.28)$$

$$\lim_{i \rightarrow \infty} \mathbf{Dw}^{(i)} = [0]_{\hat{M}^J \times S} \quad (4.29)$$

Substituting eq. (4.26) into eq. (4.25), one obtains

$$\mathbf{Dw}_s^{(i)} = \mathbf{E}_{s-1} \mathbf{Dw}_{s-1}^{(i)} \quad (4.30)$$

The PSECMAC network is trained iteratively on a set of S training samples. When $s = 1$, from eq. (4.30)

$$\mathbf{Dw}_1^{(i)} = \mathbf{E}_0 \mathbf{Dw}_0^{(i)} \quad (4.31)$$

such that

$$\mathbf{Dw}_0^{(i)} = \mathbf{Dw}_S^{(i-1)} \quad (4.32)$$

(i.e. by recycling the training samples in their existing order for each new training iteration). Following the definition of eq. (4.32),

$$\begin{aligned} \mathbf{E}_0 &= \mathbf{E}_S \\ \Rightarrow \mathbf{A}_0 &= \mathbf{A}_S \text{ (From eq. (4.26))} \end{aligned} \quad (4.33)$$

From the results of eqs. (4.30 – 4.33), $\mathbf{Dw}^{(i)}$ (See eq. (4.27)) can be expanded and re-expressed as shown in eq. (4.34).

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\ &= \begin{bmatrix} \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \underbrace{\mathbf{E}_1 \mathbf{Dw}_1^{(i)}}_{\mathbf{Dw}_2^{(i)}} & \cdots & \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i)}}_{\mathbf{Dw}_S^{(i)}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}_S \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i-1)}}_{\mathbf{Dw}_S^{(i-1)}} & \mathbf{E}_1 \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \cdots & \mathbf{E}_{S-1} \underbrace{\mathbf{E}_{S-2} \mathbf{Dw}_{S-2}^{(i)}}_{\mathbf{Dw}_{S-1}^{(i)}} \end{bmatrix} \end{aligned} \quad (4.34)$$

Decompose the $\mathbf{Dw}$ terms on the right hand side repeatedly to obtain eq. (4.35).

$$\mathbf{Dw}^{(i)} = \begin{bmatrix} (\mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_1) \mathbf{Dw}_1^{(i-1)} & (\mathbf{E}_1 \mathbf{E}_S \cdots \mathbf{E}_2) \mathbf{Dw}_2^{(i-1)} & \cdots \\ & (\mathbf{E}_{S-1} \mathbf{E}_{S-2} \cdots \mathbf{E}_S) \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \quad (4.35)$$

Following eq. (4.35), let

$$\mathbf{G}_s \equiv \mathbf{E}_{s-1} \mathbf{E}_{s-2} \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_s, \quad s \in \{1 \cdots S\} \quad (4.36)$$

where $\mathbf{G}_s$ is a matrix multiplication of S terms. With the definition of $\mathbf{G}_s$, eq. (4.35) can be re-expressed as eq. (4.37).

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\ &= \begin{bmatrix} \underbrace{(\mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_1)}_{\mathbf{G}_1} \mathbf{Dw}_1^{(i-1)} & \underbrace{(\mathbf{E}_1 \mathbf{E}_S \cdots \mathbf{E}_2)}_{\mathbf{G}_2} \mathbf{Dw}_2^{(i-1)} & \cdots \\ & \underbrace{(\mathbf{E}_{S-1} \mathbf{E}_{S-2} \cdots \mathbf{E}_S)}_{\mathbf{G}_S} \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \cdots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \end{aligned} \quad (4.37)$$

It can be observed that

$$\mathbf{Dw}_s^{(i)} = \mathbf{G}_s \mathbf{Dw}_s^{(i-1)} \quad (4.38)$$

Consequently, it follows that

$$\begin{aligned} \mathbf{Dw}^{(i)} &= \begin{bmatrix} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \cdots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_1 \underbrace{\mathbf{G}_1 \mathbf{Dw}_1^{(i-2)}}_{\mathbf{Dw}_1^{(i-1)}} & \mathbf{G}_2 \underbrace{\mathbf{G}_2 \mathbf{Dw}_2^{(i-2)}}_{\mathbf{Dw}_2^{(i-1)}} & \cdots & \mathbf{G}_S \underbrace{\mathbf{G}_S \mathbf{Dw}_S^{(i-2)}}_{\mathbf{Dw}_S^{(i-1)}} \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{G}_1)^2 \mathbf{Dw}_1^{(i-2)} & (\mathbf{G}_2)^2 \mathbf{Dw}_2^{(i-2)} & \cdots & (\mathbf{G}_S)^2 \mathbf{Dw}_S^{(i-2)} \end{bmatrix} \end{aligned} \quad (4.39)$$

Decompose the $\mathbf{Dw}$ terms on the right hand side repeatedly to obtain eq. (4.40)

$$\mathbf{Dw}^{(i)} = \begin{bmatrix} (\mathbf{G}_1)^i \mathbf{Dw}_1^{(0)} & (\mathbf{G}_2)^i \mathbf{Dw}_2^{(0)} & \cdots & (\mathbf{G}_S)^i \mathbf{Dw}_S^{(0)} \end{bmatrix} \quad (4.40)$$

where $\mathbf{Dw}_s^{(0)}$ denotes the change in the memory contents of the PSECMAC network for the first training iteration when the s^{th} training sample is presented. With respect to eq. (4.40), the memory difference matrix $\mathbf{Dw}^{(i)}$ must approach a null matrix as training tends to infinity (i.e. $i \rightarrow \infty$) in order to establish the learning convergence of the proposed PSECMAC network. Hence, the PSECMAC learning process converges if and only if

$$(\mathbf{G}_s)^i \mathbf{Dw}_s^{(0)} = [0], \quad \forall s \in \{1 \cdots S\} \quad (4.41)$$

By definition, the difference vector $\mathbf{Dw}_s^{(0)}$ can be expressed as eq. (4.42).

$$\begin{aligned} \mathbf{Dw}_s^{(0)} &= \mathbf{W}_s^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_{s-1}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)}}_{\mathbf{W}_s^{(1)}} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-2}^{(1)}}_{\mathbf{W}_{s-1}^{(1)}} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \end{aligned} \quad (4.42)$$

Repeatedly decomposing the $\mathbf{W}_s^{(i)}$ terms in the right hand side to derive eq. (4.43).

$$\begin{aligned} \mathbf{Dw}_s^{(0)} &= \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_2^{(1)} + \cdots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_s^{(0)} + \Delta \mathbf{W}_s^{(0)}}_{\mathbf{W}_1^{(1)}} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_s^{(0)} + \Delta \mathbf{W}_s^{(0)}}_{\mathbf{W}_{s+1}^{(0)}} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} \end{aligned} \quad (4.43)$$

From eq. (4.24), the PSECMAC memory update due to the s^{th} input training sample at the i^{th} iteration ($\Delta \mathbf{W}_s^{(i)}$) is computed by eq. (4.44)

$$\Delta \mathbf{W}_s^{(i)} = \alpha \underbrace{\mathbf{A}_s \{ \hat{y}_s - \mathbf{A}_s^T \mathbf{W}_s^{(i)} \}}_{\text{local error}} \quad (4.44)$$

where $\{\hat{y}_s - \mathbf{A}_s^T \mathbf{W}_s^{(i)}\}$ is a scalar value and is the learning (training) error for the s^{th} input

training sample at the i^{th} iteration. If

$$u_s^{(i)} = (\hat{y}_s - \mathbf{A}_s^T \mathbf{W}_s^{(i)}) \quad (4.45)$$

then it follows from eq. (4.44) that

$$\Delta \mathbf{W}_s^{(i)} = \alpha \mathbf{A}_s u_s^{(i)} \quad (4.46)$$

From eqs. (4.40), (4.43) and (4.46),

$$\begin{aligned} (\mathbf{G}_s)^i \mathbf{D} \mathbf{w}_s^{(0)} &= (\mathbf{G}_s)^i \{ \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} \} \\ &= (\mathbf{G}_s)^i \{ \alpha \mathbf{A}_s u_s^{(0)} + \alpha \mathbf{A}_{s+1} u_{s+1}^{(0)} + \cdots + \alpha \mathbf{A}_{s-1} u_{s-1}^{(1)} \} \\ &= \alpha (\mathbf{G}_s)^i \{ \mathbf{A}_s u_s^{(0)} + \mathbf{A}_{s+1} u_{s+1}^{(0)} + \cdots + \mathbf{A}_{s-1} u_{s-1}^{(1)} \} \end{aligned} \quad (4.47)$$

Therefore, if $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{A}_v = [0]$ for all $v \in \{1 \cdots S\}$, $(\mathbf{G}_s)^i \mathbf{D} \mathbf{w}_s^{(0)}$ in eq. (4.47) evaluates as null. From eq. (4.40), the matrix $\mathbf{D} \mathbf{w}^{(i)} = [0]$ as $i \rightarrow \infty$ follows.

Consequently, the learning process of the proposed PSECMAC network converges.

4.4.2 Learning Convergence of The PSECMAC Network

Theorem 4.1 *The training process of the proposed PSECMAC network converges if the learning constant α is such that $0 < \alpha \leq 2$.*

Proof: It can be shown that for all $v \in \{1 \cdots S\}$, when $0 < \alpha \leq 2$, $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{A}_v = [0]$ (Refer to Appendix E for the detailed proof). Therefore, the training process of the PSECMAC network converges if the learning constant α satisfies the condition $0 < \alpha \leq 2$. ■

The following two sections present the experiments that have been conducted to evaluate the performance of the proposed PSECMAC network, namely: (1) Option pricing and arbitrage trading of the GBP vs. USD currency futures option; and (2) US banking failure classification. The performances of the PSECMAC network are dutifully evaluated against the basic CMAC and HCAQ-CMAC networks and two other representative CMAC variants, namely: (1) Moody's Multi-Resolution CMAC (MMR-CMAC) (Moody, 1989), and (2) the

Fuzzy CMAC with Yager Inference Scheme (FCMAC-Yager) (Sim et al., 2006). Other benchmarking architectures studied in this chapter include the Generic Self-Organizing Fuzzy Neural Network (GensoFNN) (Tung and Quek, 2002), the Rough Set-Based Pseudo-Outer-Product Fuzzy Neural Network (RSPOP) (Ang and Quek, 2005a), as well as the classical machine learning models such as the Radial Basis Function (RBF) network and a decision table model named Inducers of Decision Table Majority (IDTM) (Kohavi, 1995), both of which are implemented in the WEKA software package (WEKA, Online).

4.5 Case Study I: Option Pricing and Arbitrage Trading

Options are financial instruments, which provide a means to manage financial risks that arise from the uncertainty of factors such as volatile interest rates, exchange rates, stock prices and commodity prices in the course of running a business. They are playing an increasingly important role in modern financial markets (Chance, 2004). The buyer of an option enters into a contract with the right, but not the obligation, to purchase or sell an underlying physical or financial asset at a later date at a price agreed upon today. By using options, companies and individuals can transfer, for a price known as the *premium* (i.e. price) of the option, any undesirable financial risk to parties who either possessed the capabilities to offset such a risk (through hedging) or want to assume that risk as speculation for financial gains. Options belong to a class of financial products generally referred to as *derivative securities* whose returns are derived from those of other financial instruments, in this case, the underlying assets for which the options are issued (Boyle and Boyle, 2001).

In an *efficient market* (Fama, 1965), the prices of the options being traded reflect or approximate their true economic (intrinsic) values to the investors. Classical economic models such as the *rational expectations theory* (Muth, 1961) and the *efficient market hypothesis* (Fama, 1970) assumed a stationary world in which all options investors have access to the same market data, behave rationally towards the current and historical pricing information and subsequently adopting the same trading decisions. Hence, an efficient market is always at its equilibrium. Such a notion suggested that it is a futile effort to devise a trading system to compute the fair economic value of an option and to exploit any arbitrage opportunities arising from a misalignment of the expected theoretical fair value of the option and the bid-ask price spread offered by the market.

However, in a real world financial market, information efficiency is generally far from perfect. Investors require time to learn and comprehend the inflow of information to facilitate decision-making that drives their trading actions. In addition, traders may interpret the same information differently and thus their expectations become indeterminate, unstable and possibly self-fulfilling (Arthur, 1994). Such a notion is also supported by modern economic models and theories such as the noisy expectations model (Grossman and Stiglitz, 1980) and the rational beliefs theory (Kurz, 1994a,b). The noisy expectations model contemplated that the market *learns* while adapting to the incoming information, and allows for the possibility of profitable trading by using and analyzing certain information to take a position in anticipation of the price changes that will occur as the rest of the market learns about that piece of information (Cheng et al., 2005). The rational beliefs theory, on the other hand, argued that competing theories and beliefs about the future risks and the future movements of the market would result in different responses by the investors and market participants (Lakonishok et al., 2004). Hence, modern economic research has provided the necessary motivation and support for the need of an accurate and computationally efficient option pricing model and the subsequent construction of a mis-priced option trading system to exploit any arbitrage opportunity to maximize trading profits.

Option pricing is defined as the process to obtain the theoretical fair economic value of an option. The price of an option is determined by a set of pricing factors such as time to expiry and the intrinsic value of the option. The conventional approach to option pricing is to construct parametric models that are based on the assumptions of continuous-time finance theory (Nielsen, 1999). The pioneering models are the *Black-Scholes formula* (Black and Scholes, 1973) and the *Binomial Pricing Model* (Rendleman Jr. and Bartter, 1979). This line of research has focused on the idea of creating risk-free portfolios for trading of options through dynamic hedging strategies. However, these models presumed complex and rigid statistical and probabilistic formulations about the price processes of the underlying assets from which the options prices are deduced (Radzikowski, 2000). A misspecification of the stochastic processes for the price dynamics of the underlying assets will lead to systematic pricing errors for the options linked to the assets. In addition, the theory of continuous-time stochastic processes is an essential prerequisite for continuous-time finance (Nielsen, 1999). Such convoluted mathematical formalization, however, is not easily accessible to novice investors to develop a comprehensive understanding of the rationale behind the computed price valuations.

Nonparametric methods of option pricing based on neural networks (Amilon, 2001; Anders et al., 1998; Hutchinson et al., 1994; Qi and Maddala, 1995), genetic algorithms (Keber, 1999) and kernel regression (Ait-Sahalia and Lo, 1995), on the other hand, are model-free approaches that have attracted significant interest in recent years. The pricing model, which is usually represented as a nonlinear functional mapping between the input factors and the theoretical option price, is derived from vast quantities of historical data. However, these methods involve heuristics and therefore suffer from poor interpretability. More recently, neuro-fuzzy approaches (Tung and Quek, 2005) are introduced to overcome this problem. With these techniques, a set of comprehensible semantic rules can be extracted from historical trading data for rational pricing of the options.

Currently, nonparametric option pricing methods are generally based on a global learning paradigm, in which the system attempts to use a single formulated model to generalize or fit the behaviors/characteristics of the entire set of historical pricing data. Some researchers have argued that it is difficult, if not impossible, to obtain a general and accurate global learning model (Huang et al., 2005). Historical option pricing data may contain complex dynamics and pricing patterns that make it hard for a global learning model to accurately generalize the underlying pricing function. In contrast, a local learning paradigm focuses on capturing only useful local information from the observed data (Bottou and Vapnik, 1992). Instead of having a single formulated model, a local learning system can be considered as a collection of locally-active models, where each sub-model is learning to generalize a different subset of the training data.

In option trading, the prices of the options are determined by a set of pricing factors, such as time to expiry and the intrinsic values of the options. The complex relationship between the valuation of an option and its influencing factors may be modeled as combinatorial associations to be extracted from the historical pricing data. This motivates the use of a local associative model as a non-parametric computational method to option pricing. In this study, the use of PSECMAC as a novel non-parametric option pricing model is investigated. The PSECMAC network is used to construct a pricing model to predict the correct valuations for American call options on the British pound (GBP) and US dollar (USD) exchange rate futures contract. The proposed pricing model is subsequently employed in a mis-priced option arbitrage trading system. The performance of the PSECMAC option trading system is benchmarked against the Black-Scholes option trading system as well as two other systems based on the simple buy-and-hold trading strategy.

4.5.1 A PSECMAC based Option Pricing Model

This section presents an overview of the proposed PSECMAC-based option pricing model, starting with a brief discussion of the dataset used to construct the pricing model and the definition of the selected input factors considered to have an impact on the pricing of the option. The proposed pricing model is subsequently evaluated on the accuracy of its pricing decisions and the results are analyzed.

The Option Dataset

The data used in this study consists of the daily closing quotes of the GBP versus USD currency futures and the daily closing bid and ask prices of American-style call options on such futures in the Chicago Mercantile Exchange (CME) (WCME, Online) during the period of October 2002 to June 2003. The GBP versus USD currency futures options in the CME have at least three unique expiration dates: current month, following month, and two months ahead. On the fourth Friday of each month, some contracts expire (as some others may be exercised earlier since they are American style options), and new ones are introduced. For any new contracts, they often have eight strike prices around the current GBP versus USD currency futures value. If the index moves outside the current strike price range, another strike price is added for all expiration dates to bracket that index value. Thus, the strike prices reflect the path of the index during the time-to-maturity period.

Selected Inputs to the Option Pricing Model

A study to determine the most influential factors affecting the prices of European style Swedish OMX index options (Amilon, 2001) has identified five attributes to be very important. They are: (1) current underlying asset price (denoted as S_0); (2) time to maturity of the option (expressed in years and denoted as T); (3) the exercise price of the option, X ; (4) the risk-free interest rate (generally taken as the compounded return rate on one-year Treasury Bill and denoted as r); and (5) the historical price volatility of the last 30 days (denoted as σ_{30}).

In the proposed PSECMAC-based option-pricing model, all of the above factors except the risk-free interest rate r are selected as inputs to the nonparametric pricing model. This is

because the risk-free interest rate r can be assumed to have no effect on the price of an American option (Whaley, 1998). Thus, the American call option pricing formula can be represented as a function of the following inputs: S_0 , X , T , and σ_{30} ; where S_0 is the current GBP vs. USD exchange rate futures value; X is the strike price of the option on the GBP vs. USD exchange rate futures; T is time to maturity of the option in years; and σ_{30} is the historical price volatility for the last 30 trading days. In addition, we introduce the notion of *moneyness* (or intrinsic value) of the futures option, which is computed as the difference between the current futures value S_0 and the option strike price X (i.e. $S_0 - X$), as a combined input to the PSECMAC-based option pricing model. Thus, the pricing function f to be approximated by the PSECMAC network is:

$$C_0 = f(S_0 - X, T, \sigma_{30}) \quad (4.48)$$

where C_0 is current option price; and $(S_0 - X)$ reflects the moneyness of the option.

Option Pricing Results and Analysis

In total, 792 data samples are available in the selected futures option data set, which contains the historic pricing data for options with five different strike prices: \$158, \$160, \$162, \$166 and \$168, with 159, 158, 173, 137 and 165 data samples respectively. The 792 data samples are subsequently partitioned into three evenly distributed sub-groups denoted as A, B and C, each containing 264 data tuples. A total of six different cross-validation sets are constructed based on the permutations of the sub-groups, as outlined in Table 4.2. The six CV sets are organized into two different evaluation models, namely Model 1 and Model 2. In Model 1, the training set is constructed using data samples from only one sub-group while the data from the remaining two sub-groups constitute the testing set. The objective of this evaluation model is to assess the generalization ability of the trained pricing system. In contrast, Model 2 employs the data samples from two sub-groups for training and aims to investigate the performances of the pricing system as more training samples are provided.

A PSECMAC network with a memory size of 12 cells per dimension is constructed for the option pricing problem. A neighborhood size (N) of 0.2 and a Gaussian width constant (γ) of 0.5 have been empirically determined. Table 4.3 lists the *recall* (in-sample testing) and *generalization* (out-of-sample testing) performances of the PSECMAC option pricing

Table 4.2: Simulation set-ups based on permutations of the three sub-groups A, B and C to define the training and testing sets of the proposed PSECMAC option pricing model

Evaluation				
Model	Configuration	Simulation	Training set	Testing set
Model 1	1/3 training and 2/3 testing	I	Sub-group A	Sub-groups B and C
		II	Sub-group B	Sub-groups A and C
		III	Sub-group C	Sub-groups A and B
Model 2	2/3 training and 1/3 testing	IV	Sub-groups A and B	Sub-group C
		V	Sub-groups A and C	Sub-group B
		VI	Sub-groups B and C	Sub-group A

model for the various CV sets. *RMSE* denotes the root-mean-square-error between the set of computed and actual option prices; and *PearCorr* is the Pearson correlation coefficient, a statistical measure reflecting the goodness-of-fit between the approximated and actual implicit pricing functions. The performances of the proposed PSECMAC option pricing model are encouraging, with an average RMSE of approximately 0.13 and 0.26 for the recall and generalization assessments of Model 1 respectively. An average correlation of 0.98 is achieved by the PSECMAC model for the generalization evaluation (as compared to 0.99 for recall), indicating a less than 1% performance degradation as the evaluation emphasis shifts from the in-sample testing (recall) to the out-of-sample evaluation (generalization) capability of the PSECMAC pricing system.

From Table 4.3, one can also observe that a larger training dataset improves the generalization performance of the PSECMAC option pricing model. The experimental results of Model 2 showed a 18% improvement $((0.2584 - 0.2107)/0.2584)$ in the RMSE value over that of Model 1 for generalization (out-of-sample testing). This increase in the accuracy of the PSECMAC option pricing model can be attributed to the improvement in the network's ability to efficiently capture the price dynamics and the valuation principles of the futures options with respect to the underlying pricing factors as the number of training instances increases. A larger training dataset results in a more comprehensive training of the entire PSECMAC associative memory surface and this increases the generalization ability of the network towards the hold-out test samples.

The entire set of option pricing simulations is then repeated using the other benchmarked architectures. To ensure a fair comparison, the sizes of the CMAC and HCAQ-CMAC networks are defined as 12 cells in each input dimension, while the MMR-CMAC structure that is evaluated consists of two layers, each with 6 and 12 memory cells in each input

Table 4.3: Performances of the proposed PSECMAC option pricing model

Evaluation Model	Simulation	Recall		Generalization	
		RMSE	PearCorr	RMSE	Correlation
Model 1	I	0.1299	0.9956	0.2386	0.9858
	II	0.1376	0.9954	0.2727	0.9816
	III	0.1178	0.9964	0.2638	0.9847
	Average	0.1284	0.9958	0.2584	0.9840
Model 2	IV	0.1382	0.9952	0.2103	0.9889
	V	0.1404	0.9949	0.2210	0.9885
	VI	0.1353	0.9954	0.2007	0.9902
	Average	0.1380	0.9952	0.2107	0.9892

dimension respectively. The parameters for the FCMAC-Yager, RSPOP-CRI, GenSoFNN- TVR (Tung et al., 2004) and IDTM systems have been empirically optimized for best performances, while the RBF network is initialized to contain 100 hidden layer nodes. Based on the optimal setting, the FCMAC-Yager network on average employs 12 memory cells for each input dimension. Table 4.4 summarizes the average RMSE (ARMSE) and average Pearson correlation (APC) findings for the evaluation Model 1 and Model 2 across the different architectures. A *Performance Index* (PI) is used to combine the ARMSE and APC measures as described in eq. (4.49).

$$PI = \frac{APC}{(1 + ARMSE)} \times 100 \quad (4.49)$$

such that a higher PI value corresponds to a better pricing performance.

From Table 4.4, one can observe that the proposed PSECMAC network achieved the best generalization performance index (PI) as compared to the rest of the pricing systems for both evaluation models when sufficient data is provided. The PSECMAC network achieved a generalization PI of 78.19 for Model 1 and 81.70 for evaluation Model 2 respectively. The PSECMAC network has also comprehensively outperformed the basic CMAC network and the benchmarked CMAC variants based on the generalization results, thereby demonstrating clearly the effectiveness of the judicious memory cells allocation process of its self-organizing structure. Specifically, the multi-resolution structure of the PSECMAC network yields, on average, a 3.7% improvement in PI value over the uniformly-quantized CMAC of the same network size. However, the recall performances of the PSECMAC network were slightly lower than those of the CMAC and MMR-CMAC networks. This is because the static uniform memory quantization of the CMAC and MMR-CMAC networks results in

Table 4.4: Benchmarking results for various option pricing model

Evaluation Model	System	Recall			Generalization		
		ARMSE	APC	PI	ARMSE	APC	PI
Model 1	CMAC	0.0531	0.9992	94.88	0.2896	0.9792	75.93
	MMR-CMAC	0.0526	0.9993	94.93	0.3838	0.9639	69.66
	HCAQ-CMAC	0.1838	0.9914	83.74	0.3395	0.9717	72.54
	PSECMAC	0.1284	0.9958	88.24	0.2584	0.9840	78.19
	FCMAC-Yager	0.1924	0.9911	83.12	0.5221	0.9236	60.68
	GenSoFNN-TVR	0.1759	0.9943	84.56	0.2764	0.9839	77.08
	RSPOP-CRI	0.2562	0.9849	78.40	0.4204	0.9578	67.43
	RBF	0.1767	0.9920	84.30	0.3438	0.9701	72.19
	IDTM	0.2388	0.9853	79.54	0.3964	0.9596	68.72
Model 2	CMAC	0.0678	0.9988	93.54	0.2579	0.9834	78.18
	MMR-CMAC	0.0710	0.9987	93.25	0.3179	0.9756	74.03
	HCAQ-CMAC	0.2218	0.9869	80.77	0.3216	0.9716	73.52
	PSECMAC	0.1380	0.9952	87.45	0.2107	0.9892	81.70
	FCMAC-Yager	0.2365	0.9879	79.89	0.2829	0.9831	76.63
	GenSoFNN-TVR	0.1857	0.9948	83.90	0.2387	0.9908	79.99
	RSPOP-CRI	0.2938	0.9787	75.65	0.3461	0.9695	72.02
	RBF	0.2389	0.9854	79.54	0.3076	0.9758	74.63
	IDTM	0.2721	0.9812	77.13	0.3208	0.9740	73.74

structures that are highly optimized for the training set. The PSECMAC memory allocation procedure, on the other hand, is geared more towards obtaining an efficient characterization of the problem's input-output mappings. This is achieved by allocating the memory cells in a non-linear manner based on the distribution of the training data. The effective non-linear memory quantization of PSECMAC allows for a better description of the problem's characteristic surface to address a new/unseen testing data. Thus, the PSECMAC network is able to achieve an improved generalization performance despite a negligible degradation in the recall performance.

In addition, the performances of the PSECMAC network for both evaluation models are superior to those of the HCAQ-CMAC network. From the evaluation results in Table 4.4, one can also observe that although the performances of HCAQ-CMAC are comparable to the other benchmarked architectures, HCAQ-CMAC achieved lower PI values compared to the basic CMAC network. These observations, therefore, demonstrate the sensitivity of the definition of the HCAQ-CMAC's quantization cost function to the network performances. In particular, the straight-forward cost function employed in the HCAQ-CMAC quantization process is inadequate when applied for the option pricing problem that is characterized by highly non-linear and volatile option price outputs.

Table 4.4 also showed that the performances of the benchmarked CMAC variants were inferior to those of the CMAC as well as the PSECMAC networks. The MMR-CMAC network (Moody, 1989) is essentially a multi-layered multi-resolution CMAC system that employs a hierarchy of CMACs with increasing modeling resolutions to improve the generalization and accuracy of the system. Unlike the proposed PSECMAC network that allocates the memory cells selectively to enhance the network's generalization capability and the accuracy of the computed output, the MMR-CMAC network employs a coarsely partitioned CMAC at the base layer to generalize the characteristics of the training data. MMR-CMAC's output accuracy is then gradually refined with the increasingly finer modeling resolutions of the higher-layer CMACs. Such a hierarchical associative memory model, however, may not be suitable for applications with a high output volatility. The large performance degradation (by the MMR-CMAC network) shown from the recall to the generalization pricing assessment in Table 4.4 is a clear indication of the network's inability to adequately generalize the characteristics of the option dataset using the coarsely partitioned base layer and to subsequently learn the specificity of the pricing data using a second layer of computing cells. The FCMAC-Yager network, on the other hand, suffers from a poor output accuracy for both the recall and the generalization assessments due to the fuzzification of the inputs. FCMAC-Yager is a Mamdani fuzzy rule based system and employs trapezoidal-shaped membership functions that often lead to a low output accuracy due to the granularity of the membership functions. This is an inherent limitation associated with the use of the Mamdani rule system from knowledge representation.

The proposed PSECMAC network also achieved more accurate pricing decisions as compared to the benchmarked neuro-fuzzy systems (i.e. GenSoFNN, RSPOP) and classical machine learning techniques of RBF and IDTM. The evaluation results of Model 1 have shown that the PSECMAC network is able to efficiently generalize the characteristics of the training data despite a small training set. The performances of the proposed PSECMAC network compared favorably to those of the GenSoFNN-TVR, RSPOP-CRI, RBF, and IDTM pricing models for the generalization assessment of Model 1. The simulation results of Model 2 for out-of-sample testing further verified the effectiveness of the PSECMAC memory allocation procedure in addressing the generalization-accuracy dilemma. In summary, Table 4.4 have adequately demonstrated the performance of the proposed PSECMAC network as an accurate option pricing model.

Subsequently, the effective cell utilization rates of the PSECMAC network for all the six

Table 4.5: Comparison of the Cell Occupancy Rate of the PSECMAC and CMAC Option Pricing Model

Simulation	CMAC			PSECMAC			Improvement
	Total	Trained	COR	Total	Trained	COR	
I	1728	1064	61.57%	1728	1170	67.71%	6.14%
II	1728	1082	62.62%	1728	1178	68.17%	5.55%
III	1728	1088	62.96%	1728	1175	68.00%	5.04%
IV	1728	1126	65.16%	1728	1297	75.06%	9.91%
V	1728	1125	65.10%	1728	1227	71.01%	5.91%
VI	1728	1116	64.58%	1728	1276	73.84%	9.26%
Average			63.67%			70.63%	6.96%

CV sets were computed and compared against those of the CMAC network. The results (denoted as the Cell Occupancy Rate) are tabulated as Table 4.5. The Cell Occupancy Rate (COR) is defined as the proportion of the trained memory cells to the total network size. From Table 4.5, one can conclude that the PSECMAC network consistently achieves a higher COR value than the CMAC network. An average improvement of 6.9% was achieved for COR and this clearly demonstrates the effectiveness of the non-linear memory allocation scheme of the proposed PSECMAC network in reducing memory wastage. A higher COR value implies a more comprehensive training of the CMAC/PSECMAC input-output associative space and indirectly translates to an improved generalization ability of the network. Such a notion is reinforced by the higher generalization accuracy achieved by the PSECMAC network over the CMAC network. The associative structure of the PSECMAC model also facilitates the extraction of discrete pricing rules. For instance, "IF *time-to-maturity* is between 0 - 0.04 years, *volatility* is between 5.08 - 5.28 and *moneyness* is between \$5.03 - \$7.98 THEN the Option-Price (on average) is \$9.4" is a representative discrete rule extracted from the PSECMAC option pricing model that expresses the knowledge acquired from the training data. Such rules also enhance human comprehension of the pricing dynamics of the traded options.

Finally, the accuracy of the non-parametric PSECMAC option pricing model is also benchmarked against the pricing performance of the parametric Black-Scholes (BS) option pricing formula (Black and Scholes, 1973). Table 4.6 tabulates the pricing performances of the BS option pricing model based on the RMSE and the Pearson correlation coefficient values for the six CV groups. All of the parameters required by the BS pricing formula are derived from the spot market. Note that since there is no training for the BS model, the simulation results of Model 1 and Model 2 in Table 4.6 are mirrored of each other. From Table 4.6, one

Table 4.6: Performances of the Black-Scholes option pricing model (All parameters are derived directly from the spot market)

Evaluation Model	Simulation	Recall		Generalization	
		RMSE	PearCorr	RMSE	Correlation
Model 1	I	1.3670	0.8289	1.2777	0.8341
	II	1.3070	0.8327	1.3087	0.8314
	III	1.2477	0.8360	1.3374	0.8302
	Average	1.3072	0.8325	1.3079	0.8319
Model 2	IV	1.3374	0.8302	1.2477	0.8360
	V	1.2777	0.8341	1.3670	0.8289
	VI	1.3087	0.8314	1.3070	0.8327
	Average	1.3079	0.8319	1.3072	0.8325

can observe that the BS pricing performances are inferior to those of the non-parametric models outlined in Table 4.4. Parametric option pricing model such as the BS formula are generally derived via the continuous-time finance theory framework (Nielsen, 1999), and they often presumed complex and rigid statistical and probabilistic models about the price processes of the underlying assets to deduce the option prices. A misspecification of the price dynamics of the underlying assets will lead to systematic pricing errors that will be detrimental to the pricing performances of the parametric model. Table 4.6 has clearly demonstrated that the BS option pricing model suffers from *systematic pricing errors* for the option dataset used in this study.

4.5.2 A Cerebellar Associative Memory Approach to Arbitrage Trading

This section introduces a mis-priced option *arbitrage* trading system, where the PSECMAC option pricing model is employed to detect any misalignments between the market spot value and the theoretical valuation of an American call option on the GBP vs USD currency futures. When such mis-pricing occur, potential arbitrage trading opportunities on the options are created and investors can exploit these opportunities to derive risk-free profits.

Arbitrage

In finance, *arbitrage* refers to the simultaneous purchase and sale of the same assets or securities in different markets in order to profit from unequal prices. An option arbitrage opportunity arises when the *Law of One Price* (Chance, 2004) is violated, making it possible

for an investor to make a *risk-free* profit. That is, if the call option is mis-priced and violates the Law of One Price, then an investor can perform arbitrage trading by buying (or selling) the call options and at the same time selling (or buying) the equivalent worth of assets. This is simply buying low and selling high where the difference in portfolio values constitutes the risk-free profit.

In this chapter, an arbitrage trading strategy known as the *Delta Hedge Trading Strategy* (DHTS) (Chance, 2004) is employed in the proposed PSECMAC-based option trading system. In the DHTS, a delta hedge ratio h is computed to determine the quantity of the underlying asset (e.g. stock) required to cover the risk of taking a naked position on the futures call option. Hence, the selling of one call option is hedged by the buying of h quantity of the underlying asset and vice versa. The hedge ratio h is computed as in eq. (4.50)

$$h_t = \frac{\Delta C}{\Delta S} = \frac{(\hat{C}_{u,t+1} - \hat{C}_{d,t+1})}{(S_{u,t+1} - S_{d,t+1})} \in [0, 1] \quad (4.50)$$

where h_t is the hedge ratio at current time t (i.e. this trading opportunity) employed to build up a risk-free portfolio with proper ratio of call options and the underlying asset; $S_{u,t+1}$ is the price of the underlying asset at time $t + 1$ (i.e. the next trading opportunity) if the price goes up; $S_{d,t+1}$ is the price of the underlying asset at time $t + 1$ (i.e. the next trading opportunity) if the price goes down; ΔS is the change in value of the underlying asset due to the projected change in price S_t at time $t + 1$; ΔC is the change in value of the call options due to the projected change in price of the underlying asset at time $t + 1$; $\hat{C}_{u,t+1}$ is the predicted price of the call option if the value of the underlying asset is $S_{u,t+1}$ at time $t + 1$; and $\hat{C}_{d,t+1}$ is the predicted price of the call option if the value of the underlying asset is $S_{d,t+1}$ at time $t + 1$.

For simplicity, the price of the underlying asset (i.e. the currency futures) in this study is assumed to either go up by 0.5 unit price or go down by 0.5 unit price (i.e. $S_{u,t+1} = S_t + 0.5$ and $S_{d,t+1} = S_t - 0.5$) such that the variable ΔS in eq. (4.50) evaluates to unity. That is, there is only a unit change in the price of the underlying asset from time t to time $t + 1$. Hence, eq. (4.50) can be reduced to:

$$h_t = \frac{(\hat{C}_{u,t+1} - \hat{C}_{d,t+1})}{(S_{u,t+1} - S_{d,t+1})} = \frac{(\hat{C}_{u,t+1} - \hat{C}_{d,t+1})}{(S_t + 0.5 - (S_t - 0.5))} = (\hat{C}_{u,t+1} - \hat{C}_{d,t+1}) = \Delta C \quad (4.51)$$

Thus, the hedge ratio of the portfolio at current time t is computed as the difference in

the predicted prices of the call option at time $t + 1$. Note that eq. (4.51) also gives the expected change in the prices of the call option for every unit change in the spot price of the underlying currency futures. This is regarded as a measure of the sensitivity of the option price to the valuation of the underlying asset.

Trading Strategy

Based on the delta hedge trading strategy described in the last section, the PSECMAC-based option trading system is implemented. The general framework of the trading system proposed in this chapter is a modified version of the generic trading decision model found in (Gencay, 1998), and is illustrated in Figure 4.10. With respect to Figure 4.10, the PSECMAC option pricing model is first constructed via supervised learning using historic pricing information derived from the option and currency futures markets. That is, PSECMAC is employed to approximate the implicit pricing function (see eq. 4.48) from the observed market valuations. The inputs and output to the PSECMAC option pricing model during the training phase are moneyness ($S_0 - X$), time to maturity T (in calendar year), past 30 days price volatility of the GBP versus USD currency futures τ_{30} , and the known closing prices of the American call options C_{Closed} respectively. During the simulation period to evaluate the trading performance of the PSECMAC option trading system, only the required inputs (i.e. moneyness ($S_0 - X$), time T and volatility τ_{30}) are presented to the PSECMAC pricing model to compute the theoretical fair valuations (i.e. $C_{Theoretical}$) of the American call options. The computed $C_{Theoretical}$ and the spot bid-ask price spread provided by the option market are subsequently fed into the trading system to detect any occurrence of arbitrage opportunity and to generate the required trading decision F . After the trades have been executed, the profits or losses (i.e. returns R) are computed and reported at the end of the simulation period.

The format of the training set is as described in the previous section. Historic options data with strike prices of \$158, \$160, \$162, \$166 and \$168 respectively from October 2002 to February 2003 is used to train the PSECMAC-based option pricing model. The test set contains out-of-sample data consisting of the intra-day bid and ask prices of the American options with strike prices of \$158, \$159, \$160, \$164 and \$170 respectively from January 2003 to June 2003. The trading algorithm is summarized as follows:

Figure 4.10: General framework of the proposed mis-priced option arbitrage trading system

1. The proposed trading system takes in the theoretical option value $C_{Theoretical}$ computed by the PSECMAC-based option pricing model and subsequently compares it to the spot bid-ask prices of the option.
2. If the predicted theoretical option value $C_{Theoretical}$ falls out of the bid-ask spread range, the trading system assumes a mis-priced arbitrage opportunity is detected.
3. The trading system would take up trading positions according to the following trading strategy:

- (a) Evaluate if the call option is overpriced or under-priced using eq. (4.52).

$$\text{Call option} = \begin{cases} \text{Overpriced, if } C_{Theoretical} < \text{Option bid-price at time } t \\ \text{Underpriced, if } C_{Theoretical} > \text{Option ask-price at time } t \end{cases} \quad (4.52)$$

- (b) If the call option is overpriced, short sell the call option and hedge the risk by buying in h_t quantity of the underlying asset, i.e. the GBP vs. USD currency futures. The hedge ratio h_t is computed using eq. (4.51). Else, if the call option is under-priced, buy in the call option and short sell h_t quantity of the GBP vs. USD futures to hedge the risk.
4. If the trading system already possessed a portfolio (i.e. has either a long or short open position on the call option with the appropriate ratio of hedged futures), it would continuously check whether the mis-priced option has pulled back into the option bid-ask spread range. If it is the case, the trading system closes all the outstanding position immediately; else, it continues to hedge the portfolio by computing a new hedge ratio h_{t+1} and adjusting the portfolio composition.

Figure 4.11: Price plot of the GBP vs USD futures for the period of the study

Trading Results and Analysis

The proposed PSECMAC-based trading system is evaluated by observing its arbitrage performances using real-life GBP vs. USD currency futures options with various strike prices. To simplify the simulation setup, transaction costs are omitted here. The results are tabulated in Table 4.7. The "total capital outlay" column denotes the overall amount of investment made on the sales and purchases of the respective options and futures in the hedging exercises, while "return on investment" (ROI) denotes the profit earned from the trading endeavors. Table 4.7 is analyzed as follows. For the trading of options with a strike price of \$158, there are 26 under-priced (UO) and 19 over-priced (OO) arbitrage opportunities respectively detected by the PSECMAC-based arbitrage trading system. The total capital outlay is \$143,300 on performing the arbitrage trading activities (i.e. short-sell (buy-in) options and buy-in (short-sell) futures and the closing of the trading positions when the mis-aligned option price pulls back within the bid-ask price spread). Eventual absolute return on investment is \$7,964.80, thus giving an effective rate of return of 5.56%.

As shown in Table 4.7, the PSECMAC-based trading system has demonstrated fairly high returns for investment, with an average ROI of around 12.8% across all the five options. The simulation results have also demonstrated that the PSECMAC trading system performed well and returned a high effective rate of return of 23.15% and 21.14% for the traded options with strike prices of \$164 and \$170 respectively. Such an observation can be explained by a price plot of the GBP vs USD futures as shown in Figure 4.11.

Table 4.7: Arbitrage performances of the proposed PSECMAC-based option trading system. (Note: UO is option under-priced arbitrage opportunity; OO is option over-priced arbitrage opportunity; and ROI denotes the return on investment)

Option Strike Price X (\$)	Sim Period (days)	Num of UO transaction	Num of OO transaction	Total Capital Outlay (\$)	Absolute ROI (\$)	Percentage ROI (%)
158	156	26	19	143300	7964.80	5.56
159	61	7	15	50940	4228.60	8.3
160	65	0	17	30820	1809.30	5.87
164	97	17	10	20560	4759.60	23.15
170	94	10	12	5560	1175.20	21.14
				Average ROI (%)		12.80

Table 4.8: Comparison of arbitrage performances by the proposed PSECMAC-based arbitrage trading system (PSECMAC), the BS-based arbitrage trading system (Black-Scholes), simple Buy and Hold Strategy (BHS) and Buy and Hold Strategy with Hedging (BHSH)

Option Strike	PSECMAC				Black-Scholes				BHS				BHSH			
	Total Capital	Absolute	ROI (\$)	Outlay (\$)	Total Capital	Absolute	ROI (\$)	Outlay (\$)	Total Capital	Absolute	ROI (\$)	Outlay (\$)	Total Capital	Absolute	ROI (\$)	Outlay (\$)
Price (\$)																
158	143300	7964.80		5900		-4820		1740		6400		1740		4169.70		1740
159	50940	4228.60		8300		-3300		1840		5300		1840		974.26		1840
160	30820	1809.30		4100		1620		1760		3380		1760		-22.13		1760
164	20560	4759.60		7300		3100		1600		540		1600		-1286		1600
170	5560	1175.20		1720		-460		400		-400		400		-1074.30		400
Total	251180	19937.5		27320		-3860		7340		15220		7340		2761.53		7340
ROI (%)	7.94				-14.13				207.36				37.62			

Figure 4.11 depicts a plot of the trading prices of the GBP vs USD currency futures during the period of the simulation, together with two straight lines illustrating the options with strike prices of \$158 and \$164 respectively. As one may observe, during the testing (evaluation) phase of the PSECMAC trading system, the currency futures are trading at a price generally (for a majority of the evaluation period) above the support price of \$158 and below \$164. Hence, the options with an exercise price of \$158, \$159 and \$160 are having high moneyness value (i.e. the options are *in-the-money*). The high moneyness of these options, in turn, reduce the profitability of the arbitrage opportunities on these options. On the other hand, more profitable arbitrage opportunities exist for the options with an exercise price of \$164 and \$170. This is due to the fluctuations and the continued upward trend of the futures price, which would likely diversify the market opinions and the trading decisions of the option investors. This leads to the execution of many profitable arbitrage trading by the PSECMAC option trading system as shown by the results in Table 4.7.

The trading performance of the proposed PSECMAC-based arbitrage trading system was subsequently benchmarked against a baseline Black-Scholes-based option trading system as well as two trading strategies based on a simple buy-and-hold trading scenario. The Black-Scholes (BS) based option trading system is constructed by replacing the PSECMAC option pricing model in the trading framework of Figure 4.10. A buy-and-hold trading system, on the other hand, tries to emulate the behavior of a buy-and-hold option trader, who simply buys in the options at the start of the simulation and squares-off his position at the end of the simulation period. Two buy-and-hold trading strategies are examined in this study: (1) Buy and Hold Strategy (BHS); and (2) Buy and Hold Strategy with Hedging (BHSB). The only difference between these two strategies is the use of hedging in BHSB to offset the risks involved in buying and adopting an open position on the options. The benchmarking results are tabulated as Table 4.8. All the benchmarked trading systems square-off their position at the end of the simulation period.

From Table 4.8, one can observe that the BS arbitrage trading system reported the poorest trading performances as compared to the rest of the benchmarked systems. The inaccuracy of the computed fair valuations of the traded options by the BS arbitrage trading system resulted in a negative overall ROI of -14.13% . This further demonstrated the significant impact of the systematic pricing errors on the performance of the Black-Scholes parametric option pricing model. The BHS trading system, on the other hand, achieved the best trading performances with an overall ROI of 207.36% . However, BHS employs a risky

trading strategy due to the fact that the trader does not hedge his trading positions and therefore stands to lose all his capital if the market is not in his favour. In this study, the high return on investment of the BHS trading system may be attributed to the continued upward trend of the futures price during the trading period (see Figure 4.11). This leads to a major appreciation in the values of the options with lower strike prices (e.g. options with a strike price of \$158, \$159 and \$160), as reflected in the high ROI values of these traded options.

Conversely, options with higher strike prices result in a lower or even negative ROI, as in the case of the options with a strike price of \$170. This clearly demonstrated the speculative and hazardous nature of this trading strategy. The BSHS trading system, on the other hand, is explored in this study to provide a fairer comparison to the PSECMAC and BS-based trading systems that perform hedging. The BSHS trading system buys options at the start of the simulation and hedge its trading position by short-selling the currency futures, and finally squares-off its trading position at the end of the simulation period with no further trading performed in-between the two trades. Although BSHS hedges its trading position at the start of the simulation, this trading strategy is almost as risky as the BHS trading strategy. This is due to the fact that BSHS does not perform dynamic hedging of the traded options. Therefore, any big fluctuations of the futures price in between the trading period may lead to a substantial financial loss. Similar to the BHS system, the high ROI values achieved by the BSHS trading system ($\text{ROI} = 37.62\%$) in this study is because of the general up-trend of the futures price. However, in the real world trading market, this scenario cannot be assumed to be true as the general market is festered with random fluctuations.

Meanwhile, the PSECMAC-based option trading system outperformed its BS-based counterpart by generating an overall ROI of 7.94%. Although this ROI value is less than those of the BHS and BSHS systems, the PSECMAC-based option trading system produces this return via a relatively risk-free investment portfolio. This is demonstrated by the results in Table 4.8, where the PSECMAC-based option trading system achieved a positive return of investment on all the five traded options. This performance is in stark contrast to the two buy-and-hold trading strategies, which obtained negative returns on some of the traded options even in a favorable market condition such as in this study. In addition, the rate of return achieved by the PSECMAC option trading system is deemed to be highly encouraging given the risk-free nature of the investment portfolio constructed and when compared

against other risk-free investments available during the same time period. For example, according to the US Federal Reserve Board, the 3-months compounding interest rate of the US Treasury Bill is 0.93% on 30th September 2003, while the 3-month fixed deposit interest rate in Singapore is only 0.25% on 3rd October 2003 according to the financial data provided by the Development Bank of Singapore (DBS).

4.6 Case Study II: Banking Failure Classification

Bank failure prevention is an important issue for the regulators of the banking industries. The collapse and failure of a bank could trigger an adverse financial repercussion and generate negative impacts such as a massive bail out cost for the failing bank and loss of confidence from the investors and depositors. Technically, banks do not fail overnight, and very often, bank failures are due to prolonged periods of *financial distress*. Banking regulators can therefore establish the traits of financial distress that characterize bank failures to identify a potential failing bank. Some commonly used statistical methodologies are *multivariate discriminant analysis* (Sinkey Jr., 1975), *logit analysis* (Martin, 1977) and Cox's *proportional hazards model* (Cole and Gunther, 1995). However, the use of such classical approaches generally results in a decision model that is neither adaptive nor computationally interpretable to the banking regulators (Cheng, 2002). The PSECMAC network, on the other hand, is an associative memory network that employs a data-driven approach for its structural and network learning process. The single-layered PSECMAC network structure facilitates the automatic identification and subsequent human interpretation of the traits of financial distress characterizing a bank failure. This motivates the use of the proposed PSECMAC network to identify problem banks using the *financial covariates* extracted from the financial statements reported by the banks.

In this experiment, the monitored banks are classified as failed or survived (non-failing) banks based on their financial performances. Financial variables (covariates) used to characterize the banks' operational quality were extracted from the Call Reports available from the Federal Reserve Bank of Chicago (ChicagoBank, Online). There are a total of nine variables and they are listed as Appendix F of the Thesis. *Regulatory closure* is the defining event of the failure of a bank. The observation period of the survived (non-failing) banks spans from January 1980 to December 2000 inclusively. For consistency, the financial

reports for the failed and survived banks have the same balance sheet dates. The original data set is pre-processed to filter out the last available financial statement for each bank in the observation period. For the failed banks, this refers to the last records prior to failure, while the last records for the surviving banks are those submitted in year 2000 (last year of the observation period). After removing banks with missing data in their records, the final dataset consists of 548 failed banks (with failure dates spreading across the entire observation period) and 2555 banks that survived the observation period. Thus, the failed banks constituted approximately 17.7% of the dataset while the survived banks account for the remaining 82.3%.

Based on the nine selected covariates, bank failure classification using the data extracted from the last financial statements was performed. The entire banking dataset is partitioned into five mutually exclusive cross-validation (CV) groups denoted as CV1--5. Each group consists of one training and one test set that are randomly generated from the set of selected surviving and failed banks. The banking dataset is initially segregated into two pools: failed and survived (non-failing) banks. For each cross-validation group, 20% of both pools are randomly selected to form the training set while the remaining 80% of the data constitute the test set. Hence, the number of survived banks far exceeds the failed banks ("unbalanced" training scenario). A single output is used to differentiate between failed and survived banks. Failed banks are denoted with output "1" while survived banks are identified by output "0".

The proposed PSECMAC network and the benchmarked architectures which include CMAC, HCAQ-CMAC, MMR-CMAC, FCMAC-Yager and GenSoFNN-CRI (Tung et al., 2004) were used to perform the bank failure classification. The systems were constructed using the training set and the performances of the trained models are evaluated using the testing set. The simulation is performed for all the five CV groups. The classification threshold (to discern between failed and survived (non-failing) banks based on the nine input financial covariates) is varied to obtain the receiver-operating-characteristic (ROC) curves of each evaluated system. The *Equal Error Rate* (EER) extracted from the ROC curves are subsequently employed as the performance measure of the respective models. Type I error is defined as the error of mis-classifying a failed bank as a survived (non-failing) one whereas Type II error is the mis-classification of a non-failing bank as a failed entity. EER denotes the point where Type I equals Type II errors.

Figure 4.12: ROC curves of the PSECMAC banking failure classification

The best classification performance for the PSECMAC network was empirically determined with a network size of 3 cells per dimension, a neighborhood parameter (N) of 0.3 and a Gaussian width constant (γ) of 0.3. Similarly, the optimal size for the HCAQ-CMAC network was also defined with 3 cells per dimension. The CMAC network, on the other hand, obtained an optimal EER measure with 4 memory cells in each dimension, whereas the MMR-CMAC network achieves optimal classification rates using two layers of CMAC with 2 and 4 cells in each dimension respectively. The parameters for the remaining benchmarked systems (i.e. FCMAC-Yager and GenSoFNN-CRI) are all empirically optimized. The FCMAC-Yager system on average uses 4 cells in each dimension. Figure 4.12(a) depicts the ROC plots of the PSECMAC network banking failure classification system. The EER measures of the various benchmarked systems are listed as Table 4.9. It is evident that the classification performance of the proposed PSECMAC network surpasses all the benchmarked architectures. The PSECMAC network obtained a significantly lower EER value of approximately 5.4% as compared to the 7% EER achieved by the CMAC network and the 10.5% EER of GenSoFNN-CRI (Tung et al., 2004). The experimental results have clearly demonstrated the superior modeling accuracy of the proposed PSECMAC network that is due to its non-uniform memory allocation procedure. That is, when compared to CMAC and MMR-CMAC, PSECMAC has achieved a higher classification rate based on a smaller network size. The HCAQ-CMAC network, on the other hand, performed poorly in its classification attempts as hypothesized in the introduction of this chapter. The HCAQ-

Table 4.9: EER readings extracted from the ROC curves - Unbalanced training scenario

System	Equal Error Rate (EER) [%]					Mean EER [%]
	CV1	CV2	CV3	CV4	CV5	
CMAC	4.386	6.552	5.661	12.16	6.552	7.0622
MMR-CMAC	5.208	6.32	5.473	12.38	6.876	7.2514
HCAQ-CMAC	27.85	7.32	12.08	18.64	14.04	15.986
PSECMAC	5.097	4.453	5.328	8.445	3.91	5.4466
FCMAC-Yager	8.93	7.13	7.77	8.23	5.08	7.43
GenSoFNN-CRI	4.4	10.99	5.49	15.6	16.48	10.59

CMAC network adopts a hierarchical clustering techniques in its adaptive quantization process, where the memory cells is allocated based on the variations in the target output. Consequently, the HCAQ-CMAC network is expected to be less suitable for classification problems that have limited number of output levels.

In addition, from Table 4.9, one can observe that the classification performances of the CMAC-based systems outperformed those achieved by the benchmarked GenSoFNN-CRI network. A plausible explanation is the use of the "unbalanced" training scenario where the proportion of survived to failed banks in the training set is highly skewed. GenSoFNN-CRI is a connectionist neural fuzzy system whereby each presentation of the training data activates and trains the entire computing structure. Thus, the learning principle of the GenSoFNN-CRI to generalize or fit the behaviors/characteristics of such unbalanced training set would likely contribute to its poor performances that is observed when attempting to discern the two overlapping but contrasting concepts (i.e. failed and survived banks) as in the banking dataset. The PSECMAC, CMAC, MMR-CMAC and FCMAC-Yager, on the other hand, consist of an ensemble of locally active models constructed from different segments of the training data. In such a local learning model, a skewed dataset has a limited effect on the learned data characteristic mappings as opposed to the GenSoFNN network that learns globally. Hence, a local learning model such as PSECMAC can effectively segregate the information expressed by the two contrasting groups of banks even though the ratio of survived to failed banks is highly skewed. The results in Table 4.9 have demonstrated that the performance of PSECMAC, which employs localized learning, is less likely to be perturbed by a statistically skewed dataset.

The set of simulations on the benchmarked systems is subsequently repeated with a "balanced" training scenario. The training sets of the five CV groups are modified by randomly pruning away redundant survived banks until the number of survived banks equals the

Table 4.10: EER readings extracted from the ROC curves - Balanced training scenario

System	Equal Error Rate (EER) [%]					Mean EER [%]
	CV1	CV2	CV3	CV4	CV5	
CMAC	7.214	8.153	5.48	6.388	9.951	7.4372
MMR-CMAC	8.474	9.212	10.00	10.10	11.3	7.8172
HCAQ-CMAC	18.38	15.3	18.54	6.64	9.36	13.644
PSECMAC	4.995	5.574	5.898	6.932	6.90	6.0594
FCMAC-Yager	8.77	8.78	7.5	8.19	8.42	8.33
GenSoFNN-CRI	7.08	9.38	5.63	13.96	4.79	8.17

Table 4.11: Comparison of the Cell Occupancy Rate of the PSECMAC and CMAC Banking Failure Classification

Simulation	CMAC			PSECMAC			Improvement
	Total	Trained	COR	Total	Trained	COR	
CV1	262144	19368	7.39%	19683	11074	56.26%	48.87%
CV2	262144	4752	1.81%	19683	6142	31.20%	29.39%
CV3	262144	16125	6.15%	19683	18768	95.35%	89.20%
CV4	262144	9765	3.73%	19683	14512	73.73%	70.00%
CV5	262144	4478	1.71%	19683	6580	33.43%	31.72%
Average			4.16%			57.99%	53.83%

failed banks. The testing sets of the CV groups remain unchanged. The ROC curves of the proposed PSECMAC network for the "balanced" training scenario are depicted as Figure 4.12(b). The EER measures of the various benchmarked systems for the "balanced" training scenario are subsequently listed as Table 4.10. As observed, the mean EER value of the GenSoFNN-CRI network or the "balanced" training scenario improved by 2.4% in comparison with the "unbalanced" training scenario (note that the testing sets are the same for both scenarios). In addition, the simulation results in Table 4.9 and Table 4.10 also demonstrated that the PSECMAC and the CMAC-based systems yielded slightly higher EER values for the "balanced" than the "unbalanced" training scenarios. This minor degradation of classification performance (0.52% on average) could be attributed to the fact that the training sets in the "balanced" training scenario consists of a smaller number of training samples from which these localized learning networks can learn from the actual data distribution, and thus a slightly poorer classification performances resulted.

Lastly, the effective cell utilization rates (COR) of the proposed PSECMAC and CMAC banking failure classification systems across the five CV groups ("unbalanced") are analyzed. The COR values are tabulated as Table 4.11. One can observe that the PSECMAC based bank failure classifier achieved a remarkably high COR value of 58% (average) as compared

to the CMAC based classifier (4.2% average). This significant improvement of 53.8% in memory efficiency can be attributed to the smaller number of memory cells required by the PSECMAC network (3 cells per dimension) as compared to the CMAC network (4 cells per dimension) in achieving the respective optimal classification results. Thus, the COR results of this experiment has clearly demonstrated the effectiveness of the memory allocation scheme employed by the proposed PSECMAC network to resolve the high memory wastage encountered by the CMAC model.

4.7 Summary

In this chapter, a newly-developed neurophysiologically-inspired multi-resolution cerebellar associative memory model named PSECMAC is presented as part of the proposed cerebellar memory framework (see Figure 1.1 of Chapter 1). The proposed PSECMAC network addresses the three architectural deficiencies present in the basic CMAC network. At the same time, PSECMAC improves from the HCAQ-CMAC network by removing the computational limitations arising from the use of the hierarchical clustering technique for the adaptive memory allocation process. Motivated by the experience-driven synaptic plasticity phenomenon observed in the learning and adaptation process of the cerebellum, the PSECMAC network employs a data-driven memory quantization scheme for the derivation of its computing structure. In the proposed PSECMAC network, the biological cerebellar synaptic adaptation process is emulated by allocating more memory cells to the data-intensive regions of the input space that corresponds to the frequently-accessed areas of the cerebellum. This translates to a finer output resolution in the task-critical (important) regions of the PSECMAC input-output (I/O) associative space, which is analogous to the smooth and dexterous execution of well-trained motor skills observed in human behavioral studies. The structural formation and the subsequent learning processes of the proposed PSECMAC network are presented in the chapter together with the theoretical proof of the learning convergence of the system.

The performance of the PSECMAC network was subsequently evaluated using two real-life applications, namely: the pricing and trading of the GBP vs. USD currency futures options and the classification of US banking failures. PSECMAC was benchmarked against the CMAC and HCAQ-CMAC networks and two other representative CMAC variants (MMR-

CMAC and FCMAC-Yager) as well as the GenSoFNN and RSPOP neuro-fuzzy systems and classical machine learning techniques such as RBF and IDTM. The PSECMAC trading performances was also benchmarked against a Black-Scholes trading system as well as two buy-and-hold trading strategies. The simulation results have adequately demonstrated the effectiveness of the proposed PSECMAC network architecture in capturing the complex input-output relationships of the two financial applications while addressing the computational limitations of the CMAC network. The PSECMAC network also outperformed all the benchmarked systems and has achieved significant computational improvements over CMAC and its two well-established variants. It is evident that the judicious memory allocation scheme of the PSECMAC network results in more comprehensive training of its memory space as reflected by the significantly higher rate of memory utilization when compared to the CMAC network.

The evaluation results of the HCAQ-CMAC and PSECMAC networks have also demonstrated the effectiveness of the new data-driven adaptive memory quantization scheme of the PSECMAC network in addressing the limitations of the HCAQ-CMAC network. In the next chapter, a novel TSL-CMAC learning paradigm is proposed to address the learning deficiencies associated with the training of a CMAC network.

"Experience is not what happens to a man;
it is what a man does with what happens to him"
—Aldous Huxley

Chapter 5

TSL-CMAC: A Neurologically-Inspired CMAC Learning Framework

The principal strength of the CMAC associative memory network lies in its localized generalization capability, which allows for efficient network computations. However, there are also two significant issues associated with this property (please refer to Figure 1.1). Firstly, it is difficult to fully-train the entire CMAC network. As CMAC is a local-learning network (Miller et al., 1990), comprehensive planning is required to generate a training profile that ensures that all the network cells are trained. Furthermore, the construction of such a training profile is not always feasible in cases involving: (1) the modeling of ill-defined problems for which only limited amount of observations are available, or (2) the control of highly complex and dynamic processes for which the direct intervention in the training trajectory of the system is too tedious or simply not possible. This partially-trained CMAC problem is evidenced in the experimental results of the previous chapters (see Section 3.5 of Chapter 3 and Section 4.5 and 4.6 of Chapter 4), where the normal direct training of the CMAC network has resulted in the relatively low *cell occupancy rate* (COR) values. Secondly, the behavior of a CMAC network is undefined in the untrained regions of the network. Consequently, a partially-trained CMAC exhibits undesirable system behavior when presented with inputs that the network has not previously been trained on. In this Thesis, this phenomenon is referred to as the *empty cells phenomena*. Such a phenomenon was also observed in the experimental results of Chapter 3 (see Figure 3.11(a)).

On the other hand, research into the neurophysiology of the human brain has established that the human cerebellum plays a significant role in the learning and acquisition of motor skills (Tomprowski, 2003). The cerebellum mediates the *rapid* and *precise* executions of skilled motor responses and possesses the ability to *adapt* and *generalize* existing skill sets to novel but similar situations. In the psychological research on human and animal motor skill learning, the *transfer of learning* (Mazur, 2006) or *motor skill generalization* is a well-established property of skill acquisitions. It has been empirically demonstrated that humans, as well as animals, have the innate abilities to adapt and generalize skills acquired in well-trained motor tasks to novel but similar situations (Mazur, 2006; Tomporowski, 2003). Such adaptation enhances the execution as well as shortens the learning processes of new motor skills. Therefore, drawing inspirations from the psychological aspects of learning, generalization and adaptation in human motor skill acquisition, a computational approach to alleviate the problem of the partially-trained CMAC network is proposed in this chapter. This approach, referred to as "patching", constructs a plausible memory surface for the untrained memory cells in a CMAC network. The proposed CMAC "patching" technique (henceforth referred to as C-Patch) constitutes a computational approach to the generalization of the CMAC learning process.

In addition, it has been empirically demonstrated that the CMAC training time increases with the size of the network and that CMAC is known to suffer from the *generalization-accuracy dilemma*. That is, a small-sized network trains faster and has a better generalization of the data characteristics while a large-sized network produces more accurate outputs but at the expense of a longer training time and has a poorer abstraction of the training data. A long training time, however, may impede CMAC's performance in time-critical/real-time applications (e.g. control) while the alleged generalization-accuracy dilemma is akin to the bias-variance offset (Geman and Bienenstock, 1992) encountered in the machine learning paradigm. Therefore, in this chapter, the proposed C-Patch technique is subsequently incorporated into a three-stage computational CMAC learning framework. The proposed framework, named the *Three-Stage Learning CMAC framework* (TSL-CMAC), aims to resolve the three major deficiencies associated with the training of a CMAC network. That is: (1) the long training time correlated to a large network size, (2) the undesirable system performances due to partial or insufficient training; and (3) the generalization-accuracy dilemma encountered by the CMAC network. The TSL-CMAC framework proposed in this chapter functionally correlates to the three-stage psychological learning model suggested

for the acquisition of human motor (and cognitive) skills (Fitts and Posner, 1967; Tomporowski, 2003).

The rest of the chapter is organized as follows. Section 5.1 presents the control setup for a second order hypothetical plant as a case study to illustrate the performance sensitivity of the CMAC network to the various training trajectories. In this section, a full training profile that traverses all of the CMAC network cells is contrasted with several scenarios of partial training. Section 5.2 presents the C-Patch technique as a computational approach to alleviate the problem of a partially-trained CMAC network due to the incomplete training profiles. The performance of the C-Patch technique for the CMAC control of a hypothetical plant is carefully evaluated in Section 5.3. Subsequently, the TSL-CMAC learning framework that incorporates the C-Patch technique to address the deficiencies associated with the training of a CMAC network is presented in Section 5.4. Section 5.5 evaluates the performance of the proposed CMAC learning framework using the same control application and Section 5.6 summarizes this chapter.

5.1 The Control of A Second Order Hypothetical Plant : A Study of The CMAC Surface Traversal

The operation of a CMAC network is characterized by the table lookup access of its memory cells. However, due to the local learning behavior of the network, the training of the CMAC network has to be carefully planned so that all the cells in the CMAC memory are traversed. In this chapter, the control of a generic second order hypothetical plant is used as a case study to investigate the sequence of traversal of the CMAC memory cells during its learning process. CMAC was previously proposed for closed-loop control of complex dynamic systems in (Commuri et al., 1997; Ku et al., 1998; Larsen et al., 1995; Yamamoto and Kaneda, 1999). In this chapter, the CMAC closed-loop controller design as proposed in (Quek and Ng, 1996) is adopted to eliminate the need for a classical controller during its training process. This allows for a more direct control of the CMAC memory traversal sequence during the learning stage. The schematic diagram of the CMAC control loop is depicted in Figure 5.1.

The objective of the CMAC controller is to drive the plant output $P(t)$ to follow the desired

Figure 5.1: The Simulink diagram of the CMAC control loop

reference signal $R(t)$. The transfer function of the hypothetical plant is:

$$H(s) = \frac{P(t)}{O(t)} = \frac{1}{s^2 + 2s + 1} \quad (5.1)$$

There are two inputs to the CMAC controller: the current plant output $P(t)$ and the closed-loop error $E(t)$. Both inputs are limited to a value range of $[0, 1]$. The experiments in this chapter employed the single layer model of the CMAC network (please refer to Appendix B for the structural and computational details on the single layer model of the CMAC network). Hence, for a given reference signal $R(t)$, the output of the CMAC controller $O(t)$ is derived using the *Weighted Gaussian Neighborhood Output* (WGNO) computational process of the corresponding activated memory cells. $O(t)$ is subsequently applied to the hypothetical plant and the plant output $P(t)$ is computed using eq. (5.1).

To learn the memory surface (i.e. control schedule) of the CMAC controller, the update equations for the memory content $w_{i,j}$ of an arbitrary cell $c_{i,j}$ at time t in the activated neighborhood are:

$$w_{i,j}(t) = w_{i,j}(t-1) + \delta w_{i,j}(t) \quad (5.2)$$

$$\delta w_{i,j}(t) = \alpha \cdot g_{i,j} \cdot err(t) \quad (5.3)$$

$$err(t) = R(t-1) - P(t) \quad (5.4)$$

where (i, j) is the index to the CMAC memory cell $c_{i,j}$; $w_{i,j}(t)$ denotes the content of the cell $c_{i,j}$ at time t ; $g_{i,j}$ is the Gaussian weighting factor of the cell $c_{i,j}$; α is the CMAC learning constant; and $err(t)$ denotes the CMAC learning error for the controller at time (t) .

The plant control simulation was conducted on a CMAC with memory size of 32 cells per dimension (i.e. a 32×32 CMAC network). The memory contents of the CMAC network were initialized to zero. A neighborhood constant of 0.05 and a learning constant of 0.05 were adopted during the training process. A small learning constant is chosen in order to allow the CMAC learning process to converge at the optimal memory contents. The neighborhood size, on the other hand, is kept small to facilitate a full learning traversal of the CMAC memory surface. To study the training trajectory of the CMAC memory surface, the CMAC controller was initially trained to respond to step functions of different amplitudes (i.e. $R(t)$ is a step function). It was observed that the step function training on the CMAC controller results in memory characteristic surfaces resembling a straight

Figure 5.2: The CMAC characteristic surface after training with step functions

diagonal line. The diagonal contour extends from the lower right hand area of the CMAC memory surface to the upper left hand region, while its exact location is dependent on the amplitude of the training step function used. Figure 5.2 shows the characteristic surface plots of two step function-trained CMAC networks. Note that the dark regions constitute the areas of untrained cells. The reference signals are described by eqs. (5.5) and (5.6)

$$f_1(t) = u(t); \quad 0 \leq t \leq 100s \quad (5.5)$$

$$f_2(t) = 0.5u(t); \quad 0 \leq t \leq 100s \quad (5.6)$$

where $u(t)$ is the unit step function. From Figure 5.2, one can observe that the larger the amplitude of the training step function, the higher is the diagonal line (cells traversed) on the memory surface of the CMAC network. Therefore, full training of the CMAC memory cells for this control application can be achieved by generating a series of training signals to traverse multiple diagonal lines. Based on the above observations, a training profile that ensures the complete training traversal of all the CMAC memory cells in the two-input CMAC controller employed in this study was devised and formulated as shown in eq. (5.7)

$$f_{\text{full}}(t) = \begin{cases} \left\lceil \frac{t}{T_s} \right\rceil \times \frac{\eta}{2}, & \text{if } \left\lceil \frac{t}{T_s} \right\rceil \bmod 2 = 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.7)$$

$$0 \leq t \leq (2M + 1)T_s \text{ sec}$$

Figure 5.3: The devised training sequence for full training of the CMAC controller ($f_{full}(t)$)

where η is the resolution of the CMAC memory cells (i.e. in this example, $\eta = 1/M$), M is the CMAC memory size per dimension, and T_s is the waveform time interval, which is the minimum duration for which a particular reference signal level has to be maintained in order for the plant output to reach that desired reference value. T_s is empirically determined as 100 seconds. Notice that each reference value in the full training sequence $f_{full}(t)$ is targeted to specifically address a particular quantization level in the CMAC network. A zero-reference signal is therefore inserted in-between two reference values to ensure the full traversal of the memory cells for each quantization level. The time required to fully-train a CMAC network is thus computed as $(2M + 1)T_s$, and this value increases with the network size. The full training sequence is depicted in Figure 5.3. Figure 5.4 shows the two dimensional characteristic surface plot of a fully-trained CMAC network. From the figure, one can observe that the entire lower triangle of the CMAC memory surface has been trained.

To study the characteristic (memory) surface of an insufficiently trained CMAC controller, two new CMAC networks with the same configuration were created. These networks were respectively trained with two different random training sequences. These random sequences are depicted in Figure 5.5. The memory characteristic surfaces of the resultant CMAC networks are depicted in Figure 5.6. One can observe that the random training sequences generate regions ("holes") of untrained cells in the memory surfaces of the CMAC networks. These are denoted as **UR1**, **UR2** and **UR3**. The size and location of these "holes" are dependent on the training profile used to train the networks. Since there is no training in

Figure 5.4: The CMAC characteristic surface after training with $f_{full}(t)$

these regions, the behavior of the CMAC controller is undefined for test inputs that fall within these clusters of untrained cells. This *empty cells phenomena*, in turn, degrades the performance of the CMAC network.

5.2 C-Patch – To Generate A Plausible Memory Surface for the Untrained CMAC Cells

It is not always feasible to generate a training profile that trains all the memory cells in the CMAC network. In such cases, the *empty cells phenomena* occurs whenever the test input falls within the regions of untrained cells, resulting in an undesirable network output. However, this problem can be alleviated by constructing a plausible memory surface for the untrained cells of the CMAC network. Such a construction process is referred to as "patching" in this Thesis. The proposed C-Patch algorithm for "patching" a partially-trained CMAC network is inspired by the neurological studies of human motor skill learning. In human and animal behavioral research, the *transfer of learning* (Mazur, 2006) or *motor skill generalization* is a well-established phenomenon of skill acquisitions. Humans, as well as animals, have the innate abilities to accustom and generalize skills previously acquired in another context to novel but similar situations (Goodbody and Wolpert, 1998; Malfait et al., 2005; Mazur, 2006; Scheidt et al., 2001; Tomporowski, 2003).

Numerous physiological as well as psychological scientific evidences supporting the notion

Figure 5.5: The random training sequences used to train the CMAC network

of generalized learning in motor skill acquisition have been presented in the literature (Chen et al., 2005; Dick et al., 2000; Lam and Dietz, 2004; Malfait et al., 2002; Palmer and Meyer, 2000; Weigelt et al., 2000). The motor skill generalization ability observed in humans can be broadly categorized into: (1) *motor adaptation* (Scheidt et al., 2001) and (2) *contextual interference* (Tomprowski, 2003). Motor adaptation refers to the capacity to adapt the execution of a well-trained motor task to changes or variations in the external environment (Lam and Dietz, 2004). Such a skill generalization capability was prominently demonstrated in a study conducted by Palmer and Meyer (Palmer and Meyer, 2000). In that study, experienced pianists were first asked to learn a new piece of music. They were subsequently asked to play a variation of the melody which required different combinations of the hand and finger movements to the one that they have learnt. The study eventually concluded that motor skill learning is not simply a matter of acquiring specific muscle movements, because experienced learners are able to transfer their skills to new situations that require them to produce similar patterns of movements (Palmer and Meyer, 2000). Contextual interference, on the other hand, refers to the phenomenon whereby the training acquired on a specific motor task influences the learning process of another novel but similar task. Such skill generalization capability was demonstrated and studied in (Chen et al., 2005; Dick et al., 2000; Weigelt et al., 2000). Psychological studies (Tomprowski, 2003) have subsequently suggested that there is a high correlation between the amount of skill transfers and the similarity in the skills learnt. Generally, the more similar the two tasks

(a) Training with Random Sequence A

(b) Training with Random Sequence B

Figure 5.6: The corresponding CMAC characteristic surfaces after training with the randomly generated training profiles

are, the greater is the influence of one on the execution of the other (Mazur, 2006).

Similarly, the local generalization characteristic of the CMAC network is based on the principle that similar inputs will produce similar outputs. Based on this notion, as well as the motivation from psychological studies on motor skill generalization, the proposed C-Patch algorithm strives to construct a plausible memory surface for the untrained CMAC memory cells to alleviate the problem of partial training in a CMAC network.

The principle behind the C-Patch algorithm is the interpolation of memory surfaces from the trained memory cells to the regions of untrained memory cells. This process is briefly described as follows. Starting from the outer edge of an untrained region, the memory content of an untrained memory cell is computed as the weighted average of the memory contents of its trained direct neighbors. Assuming a two-input CMAC network, for an arbitrary cell $c_{i,j}$ at the edge of an untrained region (see Figure 5.7), the "patched" (henceforth referred to as C-Patched) value of the memory content $w_{i,j}$ is computed as in eq. (5.8)

$$w_{i,j} = \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \frac{1/d_{k,l}}{\left(\sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} \frac{1}{d_{m,n}} \right)} w_{k,l}, \quad (k,l) \neq (i,j) \quad (5.8)$$

where $d_{k,l}$ denotes the distance between the empty cell $c_{i,j}$ and its fully-trained neighboring cell $c_{k,l}$. The interpolated values are then propagated iteratively towards the center of the

Figure 5.7: The workings of the proposed C-Patch algorithm (2-input CMAC example)

Algorithm C-Patch

Define *PatchLinkedList* as a linked list structure to hold the CMAC patch data

Do while there is still an untrained/empty cell in the CMAC network

For all the cells $C_{\bar{\mathbf{X}}}$ in the CMAC network

S1: get cell index $\bar{\mathbf{X}}$ from the quantization functions

S2: if the training flag $F_{\bar{\mathbf{X}}}$ is not set

S2.1: if the neighboring cells are trained

S2.1.1: compute patch value $V_{\bar{\mathbf{X}}}$ via eq. 5.8

S2.1.2: create patch data consisting of the index $\bar{\mathbf{X}}$ and patch value $V_{\bar{\mathbf{X}}}$

S2.1.3: store patch data in *PatchLinkedList*

S2.2: else if no trained neighbors then patch at next iteration

S2.2.1: set the flag to indicate that there is still empty cell in the network

For all patch data in *PatchLinkedList*

S1: store the patch value $V_{\bar{\mathbf{X}}}$ as the weight of patched cell $C_{\bar{\mathbf{X}}}$

S2: delete current patch data from *PatchLinkedList*

end Algorithm C-Patch

Figure 5.8: The C-Patch Algorithm

untrained region. This is illustrated in Figure 5.7. This computational process results in a pseudo characteristic surface for the regions of untrained CMAC cells.

Let $C_{\bar{\mathbf{X}}}$ denote the memory cell of the CMAC network with the index $\bar{\mathbf{X}}$, where $\bar{\mathbf{X}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_J]$ and J is the total number of inputs to the CMAC network. $\bar{\mathbf{X}}$ is derived from the quantization functions of the CMAC network. Each of the memory cells $C_{\bar{\mathbf{X}}}$ of the CMAC network is associated with a weight value $W_{\bar{\mathbf{X}}}$ and a training flag $F_{\bar{\mathbf{X}}}$, such that the training flag $F_{\bar{\mathbf{X}}}$ is set if and only if the cell $C_{\bar{\mathbf{X}}}$ has been trained. The training flag $F_{\bar{\mathbf{X}}}$

Figure 5.9: The flowchart and computational complexity of the C-Patch Algorithm: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; $\hat{M}$ – number of CMAC memory cells per dimension

Figure 5.10: The characteristic surfaces of the two randomly-trained CMAC networks after the application of the proposed C-Patch technique

is therefore employed to discern the trained/untrained cells in the network. The C-Patch algorithm is summarized in an algorithmic form in Figure 5.8. Figure 5.9 depicts a flowchart and an analysis of worst-case computational complexity of the C-Patch algorithm presented in Figure 5.8. The estimated worst-case computational complexity of patching a CMAC network of the size of $\hat{M}$ memory cells per dimension is in the order of $O(\hat{M}^{2J})$ where J denotes the total number of CMAC input dimensions.

5.3 Performance of the C-Patch Technique

The proposed C-Patch technique is subsequently applied to the two partially-trained CMAC controllers of Section 5.1. Since only the lower triangular region of the memory surface is used to control the hypothetical plant, the memory contents of the partially-trained networks are first mirrored to populate the upper triangular memory surfaces of the networks. The results of the C-Patch technique on the two networks are shown in Figure 5.10.

To evaluate the effectiveness of the proposed C-Patch technique, the plant control performances of the patched networks are benchmarked against the performances of the partially-trained CMAC networks before the application of C-Patch as well as to the fully-trained CMAC network. Four test functions are used for the evaluation: a step waveform, a stair-

case waveform and two randomly generated waveforms, each of 100s duration. The results are tabulated as Table 5.1. Three performance measures of the closed-loop error $E(t)$ were employed to determine the effectiveness of the C-Patch technique: the root mean squared error (RMSE) value, the mean absolute error value, and the standard deviation of the error $E(t)$; as well as the average transient response time of the plant output controlled by the respective networks. Transient response time is defined as the rise and fall times of the plant output to the rising and falling edges of the test signals respectively.

Table 5.1 clearly demonstrates the effectiveness of the proposed C-Patch technique in addressing the empty cells phenomena. For example, the *Partial 1* network achieved performance gains of approximately 84% in the RMSE value and 81% in the average transient response time for the staircase test function after the application of the C-Patch algorithm. In general, it can be observed that there were significant improvements in the control performances of the C-Patched networks, and these performances are highly comparable to those achieved by the fully-trained CMAC network. Moreover, the "patched" networks required substantially less training time (approximately 20% to that of a fully-traversed CMAC network). This is because the C-Patched networks are derived from their partially-trained counterparts that have a much shorter training process, and the time required for the C-Patch process is negligible.

Figure 5.11 shows the respective plots of the controlled plant output in response to the various reference signals for the fully-trained CMAC network and the partially-trained network *Partial 1* before and after applying the C-Patch algorithm. The highlighted regions a_1 – c_2 in Figure 5.11(b) illustrate the undesirable system behaviors encountered when the untrained memory cells of the *Partial 1* CMAC network controller are accessed during the evaluation. Specifically, the poor system responses in a_1 , b_2 and c_2 are due to the untrained region **UR2** in the characteristic surface of the *Partial 1* CMAC network (see Figure 5.6(a)). On the other hand, the large control errors in a_2 , b_1 and c_1 are a consequent of the *Partial 1* CMAC controller's attempts to access the memory cells of the untrained region **UR1**. One can observe from Figure 5.11(a) and 5.11(c) that the proposed C-Patch technique eliminates the undesirable system behaviors exhibited by the empty cells phenomena in the *Partial 1* CMAC network. In some instances, the C-Patched CMAC network even outperforms its fully trained counterpart. This is illustrated in Figure 5.11(a) and 5.11(c), where the respective plant outputs are highlighted as region **P** and region **Q**.

Table 5.1: Comparison of the control performances of the various networks*

Test Function	Performance Measure	Training Profile					
		Complete	Partial 1	C-Patched 1	Partial 2	C-Patched 2 (Gain 2)	
					[%]	[%]	
Step	Root Mean Squared Error	0.0470	0.0499	0.0461	7.62	0.0516	0.0473
	Mean Absolute Error	0.0107	0.0126	0.0123	2.38	0.0134	0.0128
	Standard Deviation	0.0459	0.0484	0.0446	7.85	0.0499	0.0456
	Avg. Transient Response	0.169s	0.183s	0.138s	24.59	0.214s	0.155s
Stair	Root Mean Squared Error	0.0254	0.1832	0.0286	84.39	0.1815	0.0474
	Mean Absolute Error	0.0123	0.0854	0.0174	79.63	0.1232	0.0347
	Standard Deviation	0.0245	0.1630	0.0286	82.45	0.1342	0.0335
	Avg. Transient Response	0.506s	2.979s	0.544s	81.74	4.661s	2.412s
Random 1	Root Mean Squared Error	0.1624	0.2544	0.1513	40.53	0.2297	0.1517
	Mean Absolute Error	0.0693	0.1438	0.0645	55.15	0.1443	0.0684
	Standard Deviation	0.1619	0.2274	0.1511	33.55	0.2027	0.1509
	Avg. Transient Response	1.2s	3.098s	0.872s	71.85	3.818s	1.738s
Random 2	Root Mean Squared Error	0.1473	0.2440	0.1331	45.45	0.2348	0.1344
	Mean Absolute Error	0.0599	0.1336	0.0542	59.43	0.1574	0.0619
	Standard Deviation	0.1468	0.2159	0.1329	38.44	0.1905	0.1327
	Average Transient Response	1.153s	3.059s	0.819s	73.23	4.702s	2.538s
Network Training Time		6500s	1500s	1500s	1100s	1100s	1100s

*Complete refers to the CMAC network trained using the full training profile, *Partial 1* and *Partial 2* refer to the first and second partially-trained networks using the Random Sequence A and B respectively, and *C-Patched 1* and *C-Patched 2* refer to the resultant C-Patched CMAC networks of *Partial 1* and *Partial 2* respectively. *Gain 1* and *Gain 2* denote the performance improvements for network *Partial 1* and *Partial 2* respectively after applying the C-Patch technique.

Note that the training time for each CMAC network is the same for all the test functions because only *one* training episode is performed for the selected training profile.

(a) Fully-trained CMAC

(b) Partially-trained CMAC (Partial 1)

(c) C-Patched partially-trained CMAC (C-Patched 1)

Figure 5.11: The plots of control performances of a fully-trained CMAC network as compared to a partially-trained CMAC network and its C-Patched counterpart for various test sequences

5.4 TSL-CMAC: A CMAC Learning Framework

The previous section has demonstrated the effectiveness of the proposed C-Patch technique in eliminating the empty cells phenomena observed in a partially-trained CMAC network. In this section, a computational framework based on the C-Patch technique is presented to develop an improved training process for the CMAC network.

It has been observed that humans have the ability to generate fast and accurate motor responses as well as to adapt and generalize the executions of tasks through the display of skilled behaviors. A skilled performance is the manifestation of the precise coordinations of various motor and cognitive processes with high-speed execution of accurate motor movements. Psychological research on the acquisition of human motor (and cognitive) skills

have hypothesized that these meticulously calibrated set of actions are progressively acquired through three consecutive phases of learning: the *cognitive* phase, the *associative* phase and the *autonomous* phase (Fitts and Posner, 1967).

In this three-stage learning model, skill learning/acquisition begins with the cognitive phase, which involves the conscious attempts to form a general understanding of the task undertaken and to retain this information in the working memory. This phase is characterized by the presence of large learning errors due to the trial-and-error nature of the initial learning attempts. The processing of information is slow as it requires a lot of cognitive resources. In the associative phase, the human subject learns to respond more efficiently by identifying the effective actions and eliminating the ineffective ones through the feedbacks obtained from the repeated exposures to the learning episodes. As a more comprehensive understanding of the underlying task is acquired, the execution speed of the motor/cognitive responses increases and the learning error is minimized. Extensive training in this stage serves to further enhance the stimulus-action mapping and to increase the accuracy of the responses. Lastly, the autonomous phase is marked by the emergence of a relatively accurate mental representation of the skill knowledge and this facilitates the fast automatic execution of the required skilled responses.

Based on this three-stage skill learning model, a computational framework named Three-Stage Learning for CMAC (TSL-CMAC) is proposed to resolve the major deficiencies in the training of a CMAC network. That is: (1) the long training time correlated to a large network size, (2) the undesirable system performances due to insufficient training; and (3) the generalization-accuracy dilemma encountered by the CMAC network. The proposed learning framework corresponds to the three stages of the Fitts-Posner skill learning model and consists of three consecutive phases: (1) the initial learning stage, (2) the network generalization stage, and (3) the continuous fine-tuning stage. The flowchart of the TSL-CMAC learning framework is depicted as Figure 5.12. Each of these stages are described in the following subsections.

5.4.1 The Initial Learning Stage

The training process of the TSL-CMAC framework commences with the initial learning phase. In this stage, a small-sized CMAC network (denoted as $\text{CMAC}_{\text{small}}$) is constructed

Figure 5.12: The TSL-CMAC Learning Framework

and trained to capture the general characteristic of the training data. This stage models the cognitive phase of the human skill acquisition process, in which the basic knowledge of a given task is acquired and stored in the working memory. Note that for this phase, a small-sized CMAC trains faster and generalizes better than a large-sized CMAC but at the expense of reduced output accuracy. The size of the $\text{CMAC}_{\text{small}}$ network is also application-dependent.

5.4.2 The Network Generalization Stage

Although a small-sized CMAC trains faster, acquiring the full training of the CMAC network remains a difficult issue. As discussed in Section 5.2, comprehensive planning (if feasible) is required to construct a training profile that ensures the training traversal of the entire set of CMAC memory cells. Hence, to avoid the empty cells phenomena, the C-Patch algorithm presented in Section 5.2 is incorporated into the network generalization stage of the proposed TSL-CMAC framework to construct a plausible memory surface for the

Algorithm C-Upsize

Define *OldSize* as the original (small) size of the CMAC network
 Define *oldCMAC* as the original (small) CMAC network
 Define *ScaleFactor* as the scaling factor of the C-Upsize algorithm
 S1: Check that the *NewSize* of the resultant CMAC is an integer value
 S1.1: if $\text{NewSize} = \text{OldSize} \times \text{ScaleFactor}$ is not integer
 S.1.1.1: output error message
 S2: initialize the new CMAC structure
 S2.1: create a newCMAC structure with *NewSize* number of memory cells per dimension
 S2.2: zero-initialize the newCMAC memory content
 S2.3: define the quantization functions for the newCMAC structure
 S3: populate the weights of the new CMAC structure
 For all the cells $C_{\bar{\mathbf{x}}}$ in the newCMAC network
 S3.1: get cell index $\bar{\mathbf{x}}$ from the quantization functions
 S3.2: get the quantization point $P_{\bar{\mathbf{x}}}$ of the cell $C_{\bar{\mathbf{x}}}$ from the quantization functions
 S3.3: evaluate the *oldCMAC* output at the quantization point $P_{\bar{\mathbf{x}}}$
 S3.4: use the computed output of *oldCMAC* to populate the weight $W_{\bar{\mathbf{x}}}$ of the newCMAC
 end Algorithm C-Upsize

Figure 5.13: The C-Upsize Algorithm

partially-trained CMAC network. In this stage, the C-Patch algorithm is applied to the $\text{CMAC}_{\text{small}}^p$ network to construct the patched network $\text{CMAC}_{\text{small}}^p$.

However, the C-Patch algorithm cannot increase the fidelity of $\text{CMAC}_{\text{small}}^p$ since the accuracy of the computed outputs is dependent on the size of the network. Thus, a network scaling algorithm (denoted as C-Upsize) is devised to increase the memory size of $\text{CMAC}_{\text{small}}^p$ to obtain a finer output resolution. The memory contents of the enlarged CMAC network (denoted as $\text{CMAC}_{u(\text{small})}^p$) are computed by linearly interpolating the stored contents of $\text{CMAC}_{\text{small}}^p$. The C-Upsize algorithm is outlined in Figure 5.13. Figure 5.14 presents a flowchart and computational complexity analysis of the C-Upsize algorithm. Based on Figure 5.14, the worst case complexity of upsizing a J -dimensional CMAC network of *OldSize* cells per dimension into *NewSize* size per dimension is in the order of $O(\text{NewSize}^J) + O((N \cdot \text{OldSize} \cdot \text{NewSize})^J)$, where N is the neighborhood size.

5.4.3 The Continuous Fine-Tuning Stage

The memory cells of the enlarged network after the network generalization stage (i.e. $\text{CMAC}_{u(\text{small})}^p$) contain only the coarse characteristics of the training data. Thus, this

Figure 5.14: The flowchart and computational complexity of the C-Upsize Algorithm: $O(\cdot)$ – the Big-O notation; J – number of input dimensions; OldCMAC – number of the original CMAC memory cells per dimension; NewCMAC – number of the up-sized CMAC memory cells per dimension; N – CMAC neighborhood size

Table 5.2: The functional correlates of the proposed TSL-CMAC learning framework and the three-stage model of skill acquisition by Fitts and Posner

The Three-Stage-Model of Skill Acquisition	The TSL-CMAC Learning Framework
The Cognitive Stage The learner attempts to form a general understanding of the task. The learning is slow and produces large errors due to the trial-and-error nature of the initial learning attempts.	The Initial Learning Stage The goal is to capture the general characteristics of the training data using a small-sized CMAC. The learning process is slow and large training errors are observed since the CMAC network has no prior knowledge.
The Associative Stage Consolidation of knowledge occurs via the retention of effective actions and the elimination of ineffective ones. This provides credible evidences to suggest that the skill generalization phenomena is taking place.	The Network Generalization Stage The proposed C-Patch algorithm serves as a computational model for network consolidation, where the knowledge in the trained cells are propagated to their untrained neighbors. The C-Upsize technique increases the CMAC memory size to enhance the output resolution of the network. This is analogical to the retention and refinement of the effective knowledge.
The Autonomous Stage Fast and accurate responses are obtained. Repeated exposures to the training episodes serve to fine-tune the stimulus-action mapping of the responses.	The Continuous Fine-Tuning Stage The output of the enlarged CMAC network is continuously adapted to obtain a higher output accuracy. Learning is fast since the network has previously acquired the general characteristics of the training data.

enlarged CMAC lacks the required output precision. In the continuous fine-tuning stage, the memory contents of the $\text{CMAC}_{u(\text{small})}^p$ network are continuously adapted to the training data to enhance the accuracy of the computed outputs. This is analogical to the fine-tuning of the stimulus-response mapping observed in the autonomous phase of the three-stage skill learning model. Each of the three stages in the proposed TSL-CMAC framework emulates the corresponding learning phases of the three-stage psychological model for human skill acquisition. Table 5.2 outlines the functional correlates between the proposed learning framework and the three-stage skill acquisition model by Fitts and Posner.

5.5 Performance of the TSL-CMAC Framework

In this section, three sets of simulations were performed on the control of the hypothetical plant of Section 5.1 to investigate the effectiveness of each stage in the TSL-CMAC learning framework. In each simulation, two test sequences were employed in the evaluation of the

Table 5.3: Comparison of the control performances of the various networks*

Network Description	Training Time	Staircase Test			Random Test		
		RMSE	MAE	Avg TR	RMSE	MAE	Avg TR
8×8 fully-trained	1700s	0.0997	0.0968	3.402s	0.1313	0.1083	1.616s
8×8 partially-trained	900s	0.1671	0.1527	5.443s	0.2033	0.173	4.549s
8×8 partially-trained and patched	900s	0.1051	0.1026	6.25s	0.1398	0.1157	2.594s
32×32 fully-trained	6500s	0.0254	0.0123	0.506s	0.1076	0.0422	0.931s
32×32 partially-trained	900s	0.238	0.1674	5.947s	0.2593	0.1795	5.855s
32×32 partially-trained and patched	900s	0.0483	0.0357	2.381s	0.1122	0.0618	2.593s
8×8 partially-trained, patched and enlarged to 32×32	900s	0.0867	0.0826	6.30s	0.1262	0.0991	2.62s

*RMSE refers to the Root-Mean-Squared-Error, MAE is the Mean-Absolute-Error, and Avg TR is the average transient response time of a network.

CMAC control performances; they are: a staircase waveform and a randomly generated waveform. The results of the various simulations conducted are tabulated as Table 5.3. Three performance indicators are used to quantify the control efforts of the various networks. They are: the root-mean-squared error (RMSE), mean absolute error (MAE) between the desired and actual plant output, and the average transient response time (Avg TR) of the controlled plant output to the reference signal. Figure 5.15 depicts the plots of the plant outputs in response to the staircase waveform signal for each of the CMAC network configurations. The results are analyzed as follows.

The first set of simulations were conducted on four CMAC networks: two of size 8-cells per dimension and another two of size 32-cells per dimension. The objective of the simulations is to investigate the effect of partial (random) training on the CMAC with different network sizes. The memory contents of all the CMACs were initialized to zero at the start of the simulations. Thereafter, one 8×8 CMAC and one 32×32 CMAC were fully-trained using the specially designed training profile (see eq. (5.7)), while the remaining two networks were partially-trained using a random training sequence shown in Figure 5.16.

From the respective plots in Figure 5.15 and the results outlined in Table 5.3, one can observe that larger testing errors (in contrast to the fully-trained 32×32 CMAC) were recorded for the fully-trained small-sized (8×8) CMAC network due to the coarse granularity of its computed outputs. However, there is a significant reduction in the training time of this network because of the smaller network size. There are also significant degradations in the

Figure 5.15: Simulation results under different CMAC configurations for staircase reference

performances of the partially-trained networks in comparison to the control efforts of the respective fully-trained counterparts. This observation applies to both the small-sized 8×8 and large-sized 32×32 CMAC networks.

Several occurrences of the *empty cells phenomena* due to the untrained CMAC regions can be observed in the partially-trained networks. These are highlighted as P_1 , P_2 , S_1 and S_2 in

Figure 5.16: The random training sequence used to partially-train the CMAC networks

Figure 5.15. Furthermore, the effect of partial training is more pronounced in the large-sized 32×32 CMAC network (i.e. P_3 as compared to S_3) because the coarse memory granularity of the small-sized 8×8 CMAC facilitates better generalization of the characteristics of the training data. This is reflected by the numerical results in Table 5.3, where the RMSE values of the 32×32 and 8×8 partially-trained networks for both the staircase and random test waveforms differed significantly.

In the second set of simulations, the C-Patch algorithm is subsequently applied to the two partially-trained networks to resolve the empty cells phenomena. Evaluation results (as tabulated in Table 5.3 and depicted in Figure 5.15) showed that "patching" contributes to significant improvements in the control efforts of the partially-trained networks. In fact, the performances of the patched networks approximated those of their fully-trained counterparts. Note that such encouraging performances are achieved together with a substantial reduction in the network training time. However, similar to observations obtained from the fully-trained networks, the performances of the patched 8×8 CMAC network were inferior to its 32×32 counterpart. This is due to the fact that the coarse granularity of the 8×8 CMAC network resulted in a *generalization error* in the computed output due to a poorer (lower) output resolution as compared to its 32×32 counterpart. This generalization error is displayed in the plot of Figure 5.15, where all the plant outputs of the 8×8 CMAC networks (i.e fully-trained, partially-trained and patched) exhibited a larger steady-state error (offset/bias) to the reference waveform as compared to their large-sized 32×32 counterparts.

Subsequently, to investigate the effect of network scaling on the control performance of a

small-sized CMAC network, the C-Upsize algorithm is applied to the 8×8 partially-trained and patched CMAC network to produce a 32×32 CMAC. The control simulations on the enlarged CMAC were then performed using the two test sequences. From Table 5.3 and Figure 5.15, one can observe that the application of the C-Upsize algorithm resulted in the improvement of the plant control performance of the patched 8×8 CMAC network. In particular, the increased output granularity of the enlarged network yielded a 17.5% and a 9.73% improvement in the resultant RMSE for the staircase test and the random test respectively. The average transient responses of the enlarged network, however, were slightly poorer compared to its 8×8 counterpart. This may be caused by the interpolative nature of the C-Upsize technique that produces a smoothed-out characteristic surface in the enlarged network, which translates to an increase in the plant's transient response time. In addition, due to the fact that the control schedule of the enlarged network is derived from the small-sized 8×8 CMAC, the control performances of the enlarged network are also inferior to those achieved by the partially-trained and patched 32×32 CMAC network. This reinforces the need for the fine-tuning stage of the TSL-CMAC framework to enhance the output accuracy of the enlarged network.

Hence, several conclusions can be drawn from these results. Firstly, a small-sized 8×8 CMAC network trains much faster than a larger-sized 32×32 CMAC despite a lower accuracy in the computed outputs. This translates to a larger steady-state error in the controlled plant output for the 8×8 CMAC controllers. Secondly, a partially-trained CMAC network (this applies to both 8×8 and 32×32 CMACs) suffers from the empty cells phenomena but this can be resolved by applying the proposed C-Patch algorithm to propagate the learning process to the empty/untrained memory cells. A partially-trained CMAC also acquires its control schedule faster (i.e. 700% and 50% reductions in the network training times for the 32×32 and 8×8 CMACs respectively as shown in Table 5.3) than its fully-trained equal-size counterpart. Thirdly, the control performance of a partially-trained and patched CMAC network (i.e. the C-Patched network) approximates that of its fully-trained counterpart, and this is achieved with a significantly shorter training time (as the time required to run the C-Patch algorithm is negligible as compared to the time incurred to fully train a CMAC network). Lastly, an increase in the network granularity leads to significant performance improvements of the enlarged 32×32 CMAC network over its small-sized 8×8 predecessor. However, the simulation results have shown that a fine-tuning process is required to enhance the output accuracy of the enlarged network in order

Figure 5.17: Observed learning (closed-loop) error curves and controlled plant outputs for a 32×32 CMAC with TSL learning (denoted as CMAC_{TSL}) and a 32×32 CMAC with no TSL learning (denoted as CMAC_{Direct}) (CMAC_{TSL} refers to the 32×32 CMAC which was created from the partially-trained, patched and enlarged 8×8 CMAC network in the previous experiments. CMAC_{Direct} refers to the partially-trained 32×32 CMAC from Table 5.3.)

for it to achieve a control performance comparable to its 32×32 CMAC counterparts.

Hence, in the proposed TSL-CMAC learning framework, training starts with a small-sized CMAC network which is subsequently patched and enlarged to obtain a basic generalized knowledge of the training data before the fine-tuning process. Such a learning mechanism allows for faster training of the CMAC network as illustrated in Figure 5.17, which contrasts significantly against the continuous learning performances of two CMAC networks with and without TSL-CMAC learning respectively.

Figure 5.17 depicts the performances of two CMACs used to control the hypothetical plant of Section 5.1. The 32×32 CMAC with TSL learning (CMAC_{TSL}) denotes a 8×8 CMAC network that has been randomly trained, patched and enlarged into a 32×32 CMAC as reported in the previous experiments. The 32×32 CMAC with no TSL learning (CMAC_{Direct}), on the other hand, refers to the partially-trained 32×32 CMAC of Table 5.3. Note that both CMAC_{TSL} and CMAC_{Direct} have been trained with the same random profile of Figure 5.16. Figure 5.17(a) depicts the reference (desired) plant output and the respective control efforts of the CMAC_{TSL} and the CMAC_{Direct} controllers as recorded during the simulation. Both the CMAC controllers are in continuous learning mode. That is, the two networks

performed online adaptation to the changes in the reference signal. The continuous learning error plot in Figure 5.17(b) depicts the differences (closed-loop error) between the desired and the actual controlled plant outputs by the two CMAC controllers.

From Figures 5.16 and 5.17(a), one can observe that each test reference signal value (i.e. $R(t) = 0.9, 0.4, 0.65$ and 0.3) is novel to both CMACs as they have no prior training to respond to the observed reference signal. CMAC_{TSL}, however, is able to quickly adapt and minimize its learning error as compared to CMAC_{Direct}. This is illustrated by the continuous learning error curves of Figure 5.17(b). The differing performances is due to the fact that the CMAC_{Direct} network has no prior training of the test reference signals and its ability to adapt is subsequently hampered by large regions of untrained memory cells. In contrast, even though the CMAC_{TSL} network also has no prior training of the test waveforms, it has no untrained memory cells since a coarse base characteristic surface has been acquired through the first two stages of its TSL learning process. This coarse control schedule is subsequently computationally refined during the online learning phase and it allows the CMAC_{TSL} network to rapidly adapt to the novel changing reference signal as compared to the incremental learning paradigm adopted by the CMAC_{Direct} network. Note also that the CMAC_{Direct} controller (as compared to the CMAC_{TSL} network) takes increasingly longer time to drive the plant output to the desired response as the reference level increases. This is because a longer training time is required to train the untrained cells in the CMAC_{Direct} controller as the learning error increases. Hence, these simulation results have demonstrated the effectiveness of the proposed TSL-CMAC training framework.

5.6 Summary

This chapter presents a novel neurologically-inspired computational approach to overcome the problem of insufficient training in a CMAC network. The empty cells phenomena occur whenever the CMAC test inputs fall into the regions of untrained CMAC memory cells, resulting in undesirable behaviors of the system. The proposed C-Patch technique alleviates this deficiency by interpolating the memory surfaces around the regions of untrained cells to construct a plausible memory surface for these untrained memory cells. The proposed technique was evaluated through the control of a second order hypothetical plant by comparing the performances of patched networks to those of their original partially-trained

counterparts, as well as to the fully-trained CMAC network. It has been demonstrated that the C-Patch technique resulted in significant improvements in the performances of the patched networks as compared to the partially-trained networks.

Subsequently, the proposed C-Patch algorithm was incorporated into a computational framework for CMAC learning to address the three major drawbacks encountered during the CMAC training process. The proposed TSL-CMAC learning framework consists of three stages that functionally models the three phases of the human skill acquisition process. The TSL-CMAC framework commences with the training of a small-sized CMAC network to capture the basic characteristics of the training data. This trained network is subsequently patched and enlarged to obtain a higher output resolution. Lastly, this enlarged network is continuously fine-tuned to enhance the accuracy of the network output. The proposed TSL-CMAC learning framework was subsequently evaluated on the control of the hypothetical plant by evaluating the effects of each stage in the learning process. The experimentation results have sufficiently demonstrated the effectiveness of the proposed TSL-CMAC learning framework in addressing the highlighted problems observed in training a CMAC network.

The next two chapters investigate the use of the proposed HCAQ-CMAC and PSECMAC cerebellar models in real-life applications, namely: (1) the text-dependent speaker verification; and (2) a personalized approach to the treatment of diabetes mellitus.

"There is no index of character so sure as the voice"
—Benjamin Disraeli

Chapter 6

A Cerebellar-based Approach to Text-Dependent Speaker Verification

This chapter investigates the use of the proposed PSECMAC network of Chapter 4 in a real-life speaker verification application. The ability to accurately differentiate a person from another is the key requirement in applications that involve the control or authorization of access to secured areas or materials. Some of these include automated banking, computer network security, retrieval of confidential information and secured access to buildings or services. Traditionally, the automatic recognition of individuals can be categorized into: (1) the *knowledge-based* and (2) the *token-based* approaches (Aleksic and Katsaggelos, 2006). The knowledge-based person recognition approach is based on the private knowledge (e.g. passwords, PINs) that a person possesses. The token-based approach, on the other hand, requires an individual to present a certain token (e.g. smart cards, signatures, keys) as a proof of identity. However, due to the fact that passwords or tokens are indirect ways to establish the identity of a user and therefore are exposed to fraud and impersonation by someone other than the authorized person to gain authentication, both the knowledge-based and the token-based person recognition methods are vulnerable to impostor attacks.

Biometrics, or biometric recognition, refers to the kaleidoscope of technologies that employs biologically measurable and unique human physiological or behavioral characteristics for person recognition purposes (Jain et al., 2004, 2006). Examples of the biometric data

used for person recognition include fingerprints, palm prints, hand and finger geometry, hand veins, iris and retinal scans, DNA, facial characteristics, and voice. Biometric-based recognition has many advantages over the traditional person recognition methods. Unlike issued passwords or tokens, biometric characteristics cannot be forgotten or easily stolen and therefore a biometric-based person recognition system is more robust against impostor attacks (Phillips et al., 2000). The use of biometric-based authentication is also more secure and convenient as biometrics directly identifies an individual, and it is becoming socially acceptable since it is inexpensive and comfortable to use. The choice of biometric characteristics, however, depends on many factors. Some of the most important factors include the uniqueness, time-invariability and ease of acquisition of the biometric data, as well as the cost and size of the biometric sensor and its robustness to noise, all of which contribute to the achievable performance of the resultant biometric recognition system (Wildes, 1997).

The most commonly used biometric recognition is the fingerprint-based recognition system, which has been implemented widely in both commercial and forensics applications. Fingerprint is secure, convenient, time-invariant, and unique for each individual (Wahab et al., 2005). It is also relatively easy to control the quality of the measured biometric information (with reference to noise) during the fingerprint acquisition process. However, the use of fingerprint as a biometric characteristic is still subjected to several disadvantages. Firstly, fingerprint is a *passive* (i.e. proxy) biometric, and hence it is still vulnerable to identity theft. Secondly, the acquisition of fingerprints leaves fingerprint residues (marks) on the scanning devices, which may degrade the quality of subsequent prints. Thirdly, the placement of the finger on the scanning device also affects the performance accuracy of the recognition system. Moreover, most users are less comfortable if they need to dirty their fingers in the recognition process (e.g. the use of ink to increase the precision of the measurement).

Speaker recognition is an alternative method to person recognition using voice-induced biometrics information. Voice-based person recognition is highly economical and convenient as the users are only required to input a spoken phrase (through a microphone) to the system in order to have their identity verified. Since the human voice is an *active* biometric (i.e. the speaker has full control of the speech articulation process), it is more secure than fingerprint-based recognition. The individual-specific voice characteristics are derived from the uniqueness in the behavioral and physiological aspects of the speaker's speech production system. Phonetics and linguistics research has established that the main differentiating

Figure 6.1: Classification of speaker recognition system

physiological aspect of the human speech production system is the vocal tract (Tetschner, 1993). Since no two vocal tracts are exactly the same, each individual's voice has certain acoustic peculiarities that characterize his/her vocal tract. The objective of a speaker recognition system is therefore to capture and exploit these differentiating features to discern between speakers.

Speaker recognition can be classified into two problems (see Figure 6.1): (1) speaker identification and (2) speaker verification (authentication) (Furui, 1997). Speaker identification is the problem of determining the identity of a speaker from a closed set of candidates (i.e. registered speakers) (Campbell Jr., 1997). Speaker verification, on the other hand, refers to the problem of verifying the identity claim of the speaker (Bimbot et al., 2004). Both speaker identification and speaker verification systems can be further classified into the text-dependent and text-independent recognition methods. A text-dependent recognition approach requires the users to produce a specific sequence of words or voice input. On the contrary, no such restriction is present in a text-independent speaker recognition system. General overviews on speaker recognition systems have been presented in (Atal, 1976; Doddington, 1985).

Research on speaker recognition technologies has started way back in 1974 with the use of statistical pattern matching approaches (Atal, 1974; Markel and Davis, 1979) for text-dependent and text-independent speaker recognition. Currently, popular techniques for speaker recognition include the use of non-parametric probability distribution function (PDF) (Schwartz et al., 1982), Bayesian discriminant analysis (Attili et al., 1988), Nearest Neighbor classifier (NN) (Higgins et al., 1993), Vector Quantization (VQ) (Lund, 1996; Soong et al., 1985), Dynamic Time Warping (DTW) (Higgins and Wohlford, 1986), and Hidden Markov Model (HMM) (Savic and Gupta, 1990; Tishby, 1991). In addition, machine learning approaches to speaker recognition include the use of neural networks such

as the Multilayer Perceptron (MLP) (Oglesby and Mason, 1990), Time-delay Neural Network (TDNN) (Bennani and Gallinari, 1991), Radial Basis Function (RBF) (Oglesby and Mason, 1991), Learning Vector Quantization (LVQ) (Bennani and Gallinari, 1990), and other forms of connectionist systems (Farrell and Mammone, 1994; Rudasi and Zahorian, 1991). Although many technological advances and implementation successes in speaker recognition have been achieved recently, there are still major problems impeding the effective deployment of voice-based person recognition systems (Furui, 2005). Most of these problems can be attributed to speaker variability and noise interferences. Speaker-induced variability (e.g. speaking rate, acoustic variability due to colds or disguise) and the variability in recording conditions as well as channel distortions (e.g. due to channel or background noises) affect the quality of the measured voice biometrics data. To resolve these problems, a set of speaker-differentiating features that is robust against speaker-variability as well as a recognition system that is able to efficiently cope with such variability and distortions are necessary (Furui, 2001).

This chapter investigates the use of the newly developed cerebellar-based PSECMAC architecture presented in Chapter 4 of the Thesis to perform text-dependent speaker verification. The research is motivated by everyday observations that aptly demonstrate how a human effortlessly employs his/her natural innate ability that facilitates the accurate perception of different auditory stimulants to perform speaker recognition proficiently. The human ability to differentiate sounds stems from the human auditory system's capacity to distinguish the different frequency components of acoustic signals (Kandel et al., 2000b). In particular, the auditory regions of the human cerebral cortex play an active role of analyzing auditory information and the deconstruction of complex sound patterns such as human speech. Biological research has established that the human auditory nerves process sound *tonotopically* (Gold and Morgan, 2000), where different sets of auditory nerve fibers respond selectively to the different frequencies of the acoustic signal. The auditory areas in the human brain are organized in a distinctive neural *sound map* (Kandel et al., 2000b) formation where the nerve fibers are topologically arranged with respect to their respective frequency stimulus. This pattern of organization is highly similar to the one observed in the human cerebellum, where different regions of the cerebellar cortex process the information from different sensory inputs (Albus, 1989). This similarity subsequently motivates the use of the PSECMAC network, which is a cerebellar-inspired computational architecture, to perform automatic speaker verification.

The proposed PSECMAC-based speaker verification system employs the Mel-Frequency Cepstral Coefficients (MFCCs) to characterize the voiced speech segments of the speakers. The MFCCs are computed from the cosine transformation of the real logarithm of the short-time energy spectrum of the speech signal that is expressed using the Mel-frequency scale (Pols, 1966). As the Mel-frequency scale closely approximates the frequency responses of the human auditory system, MFCCs are the most commonly used features to describe the short-term spectral envelope of a speech signal (Campbell Jr., 1997).

The rest of this chapter is organized as follows. Section 6.1 presents a brief overview of the human speech production system. Section 6.2 presents the mechanisms of the proposed cerebellar-based speaker recognition system. The experimental results and analysis of the performances of the text-dependent PSECMAC-based speaker verification system are presented in Section 6.3. Section 6.4 summarizes this chapter.

6.1 The Human Speech Production System

The human speech production system consists of the lungs, vocal cords and vocal tract (Rahim, 1994). The vocal cords refer to a fold of mucous membrane that is projecting to the larynx. The vocal tract, on the other hand, consists of air passages from the vocal cords to the lips, including the nasal cavity (Lemmetty, 1999). Figure 6.2 depicts the various components of the human speech production system.

Speech is a sequence of sounds that is generated when a flow of air from the lungs is modulated by the vocal cords and perturbed by the vocal tract (Rabiner and Juang, 1993). In the process of human speech production, the air flow from the lungs through the vocal cords provides an excitation signal to the vocal tract. The vocal tract, on the other hand, acts as a resonance chamber that modifies the frequency components (*spectrum*) of the excitation signal to produce the speech signal (Honda, 2003). The shape and size of the vocal tract can be modified by moving the *articulatory organs* (i.e. the lips, teeth, tongue, velum, jaws, etc.) which results in changes in the resonant frequencies of the vocal tract. The resonant frequencies of the vocal tract are termed as *formants* (Campbell Jr., 1997).

Based on the types of the excitation signal, all speech sounds can be classified into one of the two classes (Silverman, 2003): (1) voiced; and (2) voiceless sound. Voiced speech

Figure 6.2: The human speech production system: (1) nasal cavity; (2) hard palate; (3) alveoral ridge; (4) velum; (5) apex; (6) dorsum; (7) uvula; (8) radix; (9) pharynx; (10) epiglottis; (11) false vocal cords; (12) vocal cords; (13) larynx; (14) esophagus, and (15) trachea (Adapted from (Lemmetty, 1999))

sound originates from the vibration of the vocal cords, which happens when air is forced through the gaps of the vocal cords and thereby modulating the air flow from the lungs. This modulation process results in a quasi-periodic excitation wave that consists of a fundamental frequency and a set of harmonic components. The voiceless sounds, on the other hand, are produced when air flow from the lungs are not modulated by the vocal cords. In this case, the excitation signal of the vocal tract is a white-noise-like pressure wave.

Research into human speech production has established that the characteristics (i.e. shape) of the vocal tract is the main distinguishing factor of human voices (Campbell Jr., 1997; Deng and O'Shaughnessy, 2003). The shape of the vocal tract determines its formant structure and this can be estimated from the spectral analysis of the resultant speech sound. Voiced speech sounds are commonly used for this purpose, since the noise-like excitation signal of voiceless sounds contains no fundamental frequency nor harmonics structure that renders the spectral analysis of voiceless sounds difficult. In this chapter, the MFCCs of the voiced speech signals are employed as the characteristic features to discern between the different speakers. The proposed cerebellar-based speaker verification system is described in the next section.

Figure 6.3: The proposed PSECMAC-based speaker verification system

6.2 The PSECMAC-based Speaker Verification System

In general, a speaker verification system consists of three main components (Campbell Jr., 1997): (1) the feature extractor; (2) the speaker reference model; and (3) the pattern matching module. The feature extraction module transforms the speech signal into a parametric representation that (in the ideal case) compactly and effectively characterizes the speaker-specific information. In the training phase of the speaker verification system, the extracted speech features are used by the speaker modeling module to construct a reference model for each of the individual speakers. In this phase, each of the registered speakers has to provide a speech sample to facilitate the training of the respective speaker reference models. This process is referred to as the speaker *enrollment* phase. Finally, during the operational (testing) phase of the speaker verification system, the extracted speech features are compared against the reference model of the purported speaker using the pattern matching module. The pattern matching module computes a match score that measures the similarity between the extracted input speech features and the reference model of the claimed speaker. The decision to accept/reject the identity claim of the speaker is subsequently derived from the computed match score.

Figure 6.3 depicts the block diagram of the proposed PSECMAC-based speaker verification system. The speaker verification system in Figure 6.3 consists of two main modules: a feature extraction block and the PSECMAC network. The feature extraction module computes the Mel-Frequency Cepstral Coefficients (MFCCs) to characterize the speech signals from the different speakers. During the training process, the PSECMAC network is used to learn and model the speaker-specific characteristics derived from the MFCCs values. In the testing phase, the PSECMAC-based speaker models are employed to perform the frame

by frame verification of the incoming speaker voice. The computational mechanisms of the PSECMAC-based speaker verification system are detailed in the following sub-sections.

6.2.1 Preprocessing of Speech Signal

As described in Section 6.1 of this chapter, human speech is the result of the vocal tract-induced perturbation of the speech excitation signal originating from the vocal cords. This acoustic perturbation modifies the frequency components of the excitation signal according to the formant characteristics of the vocal tract to produce the different speech sounds. Each vocal tract shape is characterized by a set of formant frequencies (Rabiner and Schafer, 1978). Therefore, as a speaker speaks, the spectral properties of his speech signal vary with time as the shape of the vocal tract changes. Due to the fact that the shape of the vocal tract and its mode of excitation change relatively slowly, the resultant speech signal is therefore a slow time-varying signal. The human speech thus can be considered as quasi-stationary over short periods of time (typically 20 to 30 ms). Hence, the short-time spectral analysis techniques are commonly used to characterize the human speech signal.

In this chapter, the front-end preprocessing techniques that are applied to the incoming speech signal consist of: (1) speech signal segmentation; (2) windowing; and (3) voicing detection. The voice samples from each speaker are segmented into frames, where each voice frame consists of $M = 256$ data samples (this is approximately 30 ms long for a sampling frequency f_s of 8192 samples/sec). A 20 ms overlap ($20\text{ms}/30\text{ms} \times 100\% = 66.67\%$ overlap) between successive frames is employed to avoid information loss due to the use of an improper starting point in the segmentation process. Subsequently, the Hamming windowing technique (Orfanidis, 1995) is applied to attenuate the effect of signal discontinuities at the beginning and the end of each frame. The windowing technique is used to minimize the spectral distortion introduced by the segmentation process. Let $w(m)$ denotes the Hamming windowing function and $x_i(m)$ be the i^{th} frame of the speech sample. The windowed speech output $\bar{x}_i(m)$ is computed as:

$$\bar{x}_i(m) = x_i(m) * w(m), \quad 0 \leq m \leq M - 1 \quad (6.1)$$

$$w(m) = 0.54 - 0.46 \cos\left(\frac{2\pi m}{M - 1}\right), \quad 0 \leq m \leq M - 1 \quad (6.2)$$

where $*$ denotes the convolutional operator and M is the total number of samples in a frame.

Figure 6.4: Examples of the voiced, unvoiced and silence speech frames

In the final step of the preprocessing stage, voicing detection is applied to the windowed speech segments. The purpose of the voicing detector is to identify and extract the voiced speech frames and to remove the unvoiced and silence frames. Figure 6.4 depicts the characteristics of the voiced, unvoiced, and silence speech frames. As described in Section 6.1, unvoiced speech is produced when the air flow from the lungs is not modulated by the vocal cords and this results in a white-noise-like excitation signal to the vocal tract. The unvoiced speech frames therefore do not possess the quasi-stationary property that can be exploited for speaker characterization. A silence frame, on the other hand, does not contain any speaker-specific information. The zero-crossing count is a commonly used indicator for the voicing detection process and it is an indicator of the frequency at which the energy of a signal is concentrated. As the excitation by the vocal cords results in a periodic air flow through the vocal tract, a voiced speech segment typically has a low zero crossing rate. On the contrary, due to the noise-like excitation of the unvoiced speech, an unvoiced frame

generally produces a high zero crossing count. The zero crossing count of a silence frame, on the other hand, is expected to be lower than that of an unvoiced frame and it may be comparable to that of a voiced frame.

In this chapter, a Modified Zero Crossing Rate (MZCR) algorithm (Wahab et al., 2005) is employed to effectively discern between the voiced, unvoiced and silence speech frames. The MZCR extends from the original ZCR algorithm by using two threshold values to track the rate of fluctuation of the speech signal. This improves the performance of the original ZCR technique since the MZCR algorithm removes the effect of noise or small vibrato around the zero-threshold value. From the output of the MZCR algorithm, a speech frame is considered as a voiced frame if the MZCR count is greater than 10. Otherwise, the frame will be considered as a silence or unvoiced frame. The MZCR technique also eliminates incomplete frames, that is, speech frames that are not entirely consisting of the voiced speech segments. The MZCR algorithm is presented as Appendix G of the Thesis.

6.2.2 Feature Extraction

As described in Section 6.1, the basic structures of the human speech production system consist of an input excitation signal from the vocal cords and an acoustic resonant tube facilitated by the vocal tract. Human speech can therefore be identified as the output of the convolution between the excitation signal and the impulse response of the vocal tract. Speech *deconvolution* (Gold and Morgan, 2000) refers to the attempt to analyze the human speech signal by performing the separation of the speech signal into the excitation input and the resonator. The cepstral analysis is one such techniques that separates the speech excitation signal containing limited speaker-specific characteristics from the resonator that provides the majority of the speaker-specific information.

The term *cepstrum* was first introduced by Bogert et al. (Bogert et al., 1963), where it was observed that the spectral density of a signal with an echo has the form of an envelope (the spectrum of the original signal) that modulates a periodic function of the frequency (the spectrum contribution of the echo) (Oppenheim and Schaffer, 2004). Let $x(m)$ denotes the observed speech samples, $e(m)$ denotes the excitation signal and $v(m)$ be the transfer function of the vocal tract.

The speech signal $x(m)$ can be formulated as in eq. (6.3):

$$x(m) = e(m) * v(m) \quad (6.3)$$

where m denotes the sample index of the speech signal. Thus, the spectral magnitude of the speech signal can be expressed as in eq. (6.4):

$$|X(\omega)| = |E(\omega)||V(\omega)| \quad (6.4)$$

where $|X(\omega)|$, $|E(\omega)|$ and $|V(\omega)|$ are the Fourier transform of $x(m)$, $e(m)$ and $v(m)$ respectively. Taking the logarithm of eq. (6.4) transforms the multiplicative operation of the spectral magnitudes into a linear relationship as shown in eq. (6.5):

$$\log |X(\omega)| = \log |E(\omega)| + \log |V(\omega)| \quad (6.5)$$

Such a linear relationship subsequently facilitate the filtering operation to separate the speech signal into its individual components.

The cepstral coefficients (i.e. cepstrum) are obtained by computing the inverse Fourier transform of the log magnitude spectrum of the speech signal:

$$c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |X(\omega)| e^{j\omega k} d\omega \quad (6.6)$$

where c_k is the k^{th} cepstral coefficient, and the resultant domain is referred to as the *quefreny* domain. In a voiced speech, the excitation $e(m)$ is a periodic signal that are extended in time and thus can be characterized by a rapidly varying function of the frequency (ω). On the other hand, the impulse response of the vocal tract is relatively short and therefore contributes to a component that varies slowly with ω . Hence, the inverse Fourier transform operation on the log magnitude spectrum of the speech signal effectively results in the separation of the excitation signal and the vocal tract resonances. The cepstral contribution of the excitation signal that varies rapidly with the frequency is concentrated at the high quefreny (k). In contrast, the vocal tract responses yield a cepstral contribution that is concentrated on the low quefreny. This is illustrated in Figure 6.5.

The Mel-Frequency Cepstral Coefficients (MFCCs) (Pols, 1966) refer to the set of cepstral coefficients that are computed using the Mel-frequency scale that closely approximates the

Figure 6.5: The cepstrum of the speech signal (Adapted from (Gold and Morgan, 2000))

Figure 6.6: The Mel-frequency bandpass filters

frequency responses of the human auditory system. Psychophysical studies on the human auditory system has established that the human perception of the frequency (pitch) of a sound does not follow a linear scale. Instead, the Mel-frequency scale (Volkman et al., 1937) is a perceptual scale of pitches derived from empirical studies on human listeners. The Mel-frequency scale follows a linear frequency spacing for the frequency range below 1000 Hz and a logarithmic spacing for the frequency range above 1000 Hz. The conversion from the normal frequency scale (Hz) to the Mel-frequency scale (Mel) can be approximated by eq. (6.7) (Hirsimaki, 2002):

$$f_{mel} = 2595 \log_{10}(1 + f/700) \quad (6.7)$$

where f_{mel} is the Mel-frequency value and f refers to the normal frequency measure.

In the computation of MFCCs, the spectrum of the speech signal is first passed through the Mel-frequency band-pass filters before its transformation into the quefrequency domain. This Mel-scale filter bank (MSFB) is essentially a set of overlapping triangular band-pass filters with equal bandwidth on the Mel-frequency scale. Figure 6.6 depicts the placement of the Mel-frequency band-pass filters on the normal frequency scale. The filters' bands are linearly-spaced for bandwidth below 1000 Hz and logarithmically-spaced above the 1000 Hz

point. In practice, the band-pass filtering process is often performed on the Mel-frequency scale by warping the normal frequency scale into the Mel-frequency scale via eq. (6.7). In this chapter, a total of 21 Mel-filters are employed for the extraction of the MFCC features to perform speaker verification. The filters occupy only the first half of the total spectrum bandwidth of the speech signal (i.e. $0 - f_s/2$ Hz, where f_s is the sampling frequency), since the other half of the spectrum is essentially a duplicate of the first half that is introduced by the sampling process of the speech signal.

The MFCCs feature extraction process of the proposed PSECMAC-based speaker verification system is described as follows:

1. Computation of the magnitude spectrum of the voiced speech segments

The Fast Fourier Transform (FFT) algorithm (Orfanidis, 1995) is applied to each of the voiced speech segments identified using the MZCR algorithm. The magnitude spectrum $|\bar{X}_i(\omega)|$ of the i^{th} windowed speech segment $\bar{x}_i(m)$ from the signal preprocessing phase is computed using eq. (6.8):

$$|\bar{X}_i(\omega)| = \bar{X}_{i,re}^2(\omega) + \bar{X}_{i,im}^2(\omega) \quad (6.8)$$

where $\bar{X}_{i,re}(\omega)$ and $\bar{X}_{i,im}(\omega)$ denote the real and imaginary components of the spectrum $\bar{X}_i(\omega)$ respectively.

2. Conversion from the normal frequency to the Mel-frequency scale

To accommodate the characteristics of the MSFB, the spectrum of the speech signal (in the normal frequency scale) is mapped onto the Mel-frequency scale via eq. (6.7) to obtain the corresponding Mel-spectrum of a voiced speech segment.

3. Mel-frequency band-pass filtering

The MSFB is subsequently applied to the Mel-spectrum of a speech segment. In the Mel-frequency scale, the MSFB consists of a set of overlapping and equally-spaced triangular filters with equal bandwidth. The spectral output of each Mel-frequency band-pass filter is derived from the multiplication of the filter's response with the average power spectrum of the speech segment in the corresponding frequency bandwidth. The resultant filter output, expressed in terms of the Mel power spectral coefficient,

is computed as eq. (6.9):

$$S_k = \sum_{m=0}^{(M/2)-1} (A_{k,m} P_m), \quad k = 1, 2, \dots, K \quad (6.9)$$

where S_k is the Mel power spectral coefficient of the k^{th} filter, M is the total number of samples in the voiced speech segment, $A_{k,m}$ is the amplitude response of the m^{th} frequency component of the magnitude spectrum of the voiced speech segment with reference to the k^{th} band-pass filter, and P_m is the average power spectrum of the speech signal at the corresponding m^{th} frequency component.

4. Computation of the MFCCs

The MFCCs of a voiced speech segment are computed via the Discrete Cosine Transform (DCT) (Orfanidis, 1995) of the log of the Mel power spectral coefficients:

$$\text{MFCC}_k = \sum_{k'=1}^K (\log S_{k'}) \cos \left(k(k' - 0.5) \frac{\pi}{K} \right), \quad k = 1, 2, \dots, K \quad (6.10)$$

where MFCC_k denotes the k^{th} MFCC, K is the total number of MFCCs and $S_{k'}$ is the k'^{th} Mel power spectral coefficient. In this study, the first MFCC component (i.e. MFCC_0) is excluded from the set of features employed for the speaker verification task. This is because MFCC_0 represents the mean value of the speech segment and thus contains little speaker-specific information (Wahab et al., 2005).

6.2.3 PSECMAC Modeling of Speaker-Specific Characteristics

The PSECMAC network (refer to Chapter 4 of the Thesis) is subsequently employed to model the speaker-specific characteristics extracted from the speech samples to implement the proposed cerebellar-based speaker verification system. In this study, each speaker is characterized by the first six MFCCs (i.e. $\text{MFCC}_1 - \text{MFCC}_6$) out of the 20 MFCCs computed from his/her digitized voiced speech samples. This is motivated by the fact that the characteristics of the vocal tract that discern between the individual speakers are concentrated in the low frequency domain of the voiced speech signal. Thus, the PSECMAC network performs the associative mapping between the speaker-specific characteristics of the vocal tract to the identity of the respective speaker.

6.3 Experiments and Results

This section presents the experiments conducted to evaluate the performances of the proposed PSECMAC-based speaker verification system described in Section 6.2. The data collection process is first described, followed by a brief discussion of the experimental setups. The proposed PSECMAC-based speaker verification system is subsequently evaluated on the accuracy of its frame-by-frame verification decisions and the results are analyzed.

6.3.1 Dataset

The voice samples used in this study were collected from 10 randomly selected adult speakers consisting of six males and four females. During the data collection process, each of the recruited speakers was told to read a pre-prepared passage in a normal reading pace. The speakers' voices were recorded using a digital tape recorder in a quiet laboratory environment. The collected speech samples are first converted to wave files (*.wav) with a sampling frequency (f_s) of 8192 samples/second to obtain a speech quality that is compatible to that of typical day-to-day telephony applications. The speech sample of each speaker is then partitioned into frames consisting of $M = 256$ samples per frame with a 66.67% (approximately 20ms) overlap between two successive frames. Subsequently, the MZCR algorithm is applied to the windowed (Hamming-weighted) speech frames to extract a total of 100 voiced speech frames per speaker. The first 20 valid MFCC values are then computed for each of the extracted voiced speech frames for all the speakers as described in Section 6.2.2.

Based on the computed MFCCs of the voiced speech segments, speaker verification is performed for the ten recruited human subjects. In the proposed PSECMAC-based speaker verification system, a speaker is characterized by the first six MFCCs of his/her voiced speech segment. The MFCCs of each speech segment formed a data tuple/sample. Therefore, there is a total of 100 such data tuples for each speaker in the experiments conducted. A three-fold cross validation (CV) approach was adopted throughout the evaluation process. The construction of the various cross-validation data groups used in the experiments is described as follows:

1. Randomize the presentation order of the data samples of each speaker:

The presentation order of the 100 MFCCs data tuples that are extracted from each

speaker is randomized to generate a comprehensive voice characterization of the speaker with respect to the prepared script. This is to ensure that speech segments from different parts of the passage appeared in both the training and testing sets of each CV group.

2. Form the CV groups for each speaker:

The randomized MFCCs dataset of each speaker is subsequently partitioned into three CV groups. Each CV group consists of a training and a testing set. In the experiments, the training set comprises of 40% of the entire MFCCs dataset of each speaker. The remaining 60% of the dataset constitutes the testing set such that the training and testing sets of each CV group are mutually exclusive. On the other hand, there is a 25% overlap (i.e. 10% of the total number of MFCCs data tuples of a speaker) between the training sets of successive CV groups. The CV partitioning scheme employed for the experiments is depicted in Figure 6.7.

3. Construct the final training and testing sets for the CV groups of each speaker:

For each speaker, the final training and testing sets of a CV group consist of the training and testing samples of the corresponding CV groups of all the registered speakers. For example, the final training and testing sets of the CV1 group of Person 1 are constructed using the training and testing sets of all the CV1 groups extracted from Persons 1 to 10. Hence, the training set of CV1 of Person 1 will consist of 40 MFCCs data tuples belonging to Person 1 that formed the positive (i.e. authentic) class and 360 MFCCs data tuples from Persons 2 to 10 that formed the negative (i.e. impostor) class. The same applies for the testing set where 60 MFCCs data tuples belong to the actual speaker and the remaining 540 MFCCs data tuples are provided by the impostors. A single output is subsequently used to differentiate between the MFCCs data samples belonging to the actual (authentic) speaker and those belonging to the impostors. The data samples that belong to the actual speaker are denoted with an output "1" while those that belong to the impostors are marked with an output "0". A total of three such CV group is constructed for each speaker for the experiments.

Since the number of impostor input samples far exceeds that of the actual speaker in the resultant training sets, the training of the PSECMAC-based speaker verification system using the CV groups described above is termed as the "unbalanced" training scenario.

Figure 6.7: The partitioning of the three CV groups for the MFCCs dataset of a speaker

6.3.2 Experimental Results and Analysis

The PSECMAC network is subsequently employed to model the voice characteristics of each individual speaker using the corresponding three CV groups. For each CV group, a PSECMAC speaker model is constructed for the speaker using the training set and the performance of the trained speaker model is evaluated with the corresponding testing set. The simulation is performed for all the three CV groups of the ten speakers; that is, a total of 30 experiments. The classification threshold (to discern between the actual speaker and the impostors based on the first six MFCC values of the incoming speech segments) is varied to derive the receiver-operating-characteristics (ROC) curves for each evaluated CV group. The *Equal Error Rate* (EER) readings extracted from the ROC curves are subsequently employed as the performance measure of the PSECMAC-based speaker verification system. Type I error is defined as the error of falsely rejecting the voice input of the actual speaker whereas Type II error is the error of accepting the impostor's voice input as that of the actual speaker. EER denotes the point where Type I error equals Type II error.

A PSECMAC network with a memory size of 6 cells per dimension is constructed to model the speaker-specific voice characteristics extracted from the voiced speech segments. A neighborhood size (N) of 0.1 and a Gaussian width constant (γ) of 0.5 have been empirically determined for the experiments. Table 6.1(a) lists the frame-by-frame EER verification performances of the proposed PSECMAC-based speaker verification system for all the three CV groups of Persons 1 to 10. From Table 6.1(a), one can observe that the performances of the proposed PSECMAC-based speaker verification system are fairly encouraging with an average EER of approximately 12% and a standard deviation of only 2.72% across the ten evaluated speakers. The ROC curves extracted from the verification performances of the PSECMAC-based speaker verification system for Persons 1 to 10 are presented as

Appendix H.1 of the Thesis.

In order to benchmark the performances of the proposed PSECMAC-based speaker verification system, the set of experiments is repeated by using various well-established computational architectures to construct the speaker model. The benchmarking systems evaluated in this study are: (1) the basic CMAC network (Albus, 1975a,b); (2) the HCAQ-CMAC network (presented in Chapter 3 of the Thesis); (3) the Multi-Layered Perceptron (MLP); (4) the Radial Basis Function network (RBF) (WEKA, Online); and (5) the Generic Self-Organizing Fuzzy Neural Network with the Compositional Rule of Inference scheme (GenSoFNN-CRI) (Tung and Quek, 2002). The network sizes of CMAC and HCAQ-CMAC have been defined as 6 cells per dimension to maintain a fair performance comparison with the PSECMAC network. For the HCAQ-CMAC network, the operational parameters such as the network's cost function weighting factors of $\beta_1 = \beta_2 = 0.5$, and a neighborhood size (N) of 0.5 were empirically determined. On the other hand, the MLP has a predefined structure that consists of six input, thirteen hidden and one output nodes respectively while the RBF network is initialized to contain 100 hidden layer nodes. The parameters of the GenSoFNN-CRI model has been empirically optimized via heuristic approaches.

Table 6.2(a) summarizes the average EER values achieved by the various speaker models in the speaker verification task of the ten speakers. As shown in Table 6.2(a), the PSECMAC network achieved the best verification performances among all the benchmarked architectures. The PSECMAC network reports the lowest average EER and EER standard deviation across all the evaluated speakers, thereby demonstrating the accuracy and consistency of its speaker models. The verification performances of the PSECMAC network outperformed those of the benchmarked basic CMAC as well as the HCAQ-CMAC networks. The degraded speaker verification performances of the basic CMAC network are largely due to the memory allocation scheme employed to partition the CMAC's input space. The basic CMAC network employs a uniform allocation of its memory cells. This rigid uniform partitioning of the input space limits the modeling accuracy of the CMAC network and thus leads to a suboptimal performance. The HCAQ-CMAC network, on the other hand, employs a hierarchical-clustering scheme for its memory allocation process. Unfortunately, this memory allocation scheme hinders the performance of HCAQ-CMAC for classification problems for which only a limited number of target output levels or variations existed.

Table 6.2(a) has also showed that the PSECMAC-based speaker verification system has

Subject	Gender	Equal Error Rate [%]			
		CV1	CV2	CV3	Average EER
Person 1	M	12.40	18.33	15.10	15.28
Person 2	M	11.64	15.87	11.64	13.05
Person 3	M	10.00	8.34	7.17	8.50
Person 4	F	5.04	9.73	10.97	8.58
Person 5	M	15.00	18.39	15.00	16.13
Person 6	F	10.08	6.52	20.26	12.29
Person 7	M	13.26	18.33	13.26	14.95
Person 8	F	9.98	9.02	12.60	10.53
Person 9	M	13.38	9.96	13.38	12.24
Person 10	F	11.74	10.02	5.90	9.22
Average EER					12.08
Std Deviation of EER					2.72

(a) Unbalanced training scenario

Subject	Gender	Equal Error Rate [%]			
		CV1	CV2	CV3	Average EER
Person 1	M	11.66	11.66	12.19	11.84
Person 2	M	11.68	13.34	12.41	12.48
Person 3	M	8.30	9.28	12.31	9.96
Person 4	F	8.27	8.34	6.81	7.81
Person 5	M	6.00	19.33	13.47	12.93
Person 6	F	6.69	5.06	16.67	9.47
Person 7	M	22.53	16.68	16.68	18.63
Person 8	F	10.00	10.00	8.33	9.44
Person 9	M	18.29	8.35	18.29	14.98
Person 10	F	7.82	8.32	12.48	9.54
Average EER					11.36
Std Deviation of EER					3.48

(b) Balanced training scenario

Table 6.1: The performances of the PSECMAC-based speaker verification system

comprehensively outperformed the benchmarked classical machine learning-based speaker models constructed from the MLP and RBF networks, as well as those derived using the GenSoFNN-CRI neural-fuzzy system. The PSECMAC network achieved a 56.9% $((28.08 - 12.08)/28.08)$ and 26.7% $((16.49 - 12.08)/16.49)$ improvement in the EER value over the MLP-based and GenSoFNN-CRI-based speaker verification system respectively. From the results tabulated in Table 6.2(a), one can also observe that the MLP network reports the poorest speaker verification performances with an average EER of approximately 28%. The inferior performances of the MLP-based speaker verification system as compared to the rest of the benchmarked architectures may be attributed to the use of the "unbalanced" training scenario, where the number of impostor speech samples in the training set far exceeds the number of authentic samples. Machine learning research has long established that the MLP is a connectionist network that employs global learning, whereby each presentation of a training sample adapts the weights of its entire network structure. Thus, training the MLP network with the "unbalanced" training scenario will result in constructing speaker models that are heavily biased towards the rejection of the impostors' speech samples. This phenomenon subsequently optimizes the Type II errors of the MLP speaker models at the expense of increased Type I errors. Consequently, the performances of the resultant MLP-based speaker verification system are severely impaired by the large Type I errors.

To investigate how the data composition of the training set affects the performances of the speaker verification systems, the set of simulations on the benchmarked architectures is subsequently repeated with a "balanced" training scenario. In the "balanced" training scenario, the training sets of the three CV groups of each speaker are modified by duplicating the positive training samples (i.e. samples belonging to the authentic class) until the number of authentic samples equals the number of impostor samples. Thus, the "balanced" training set of CV1 of Person 1 consists of 360 (i.e. 40×9 duplicate sets) MFCCs data tuples belonging to the subject (i.e. authentic class) and 360 (i.e. 40×9 persons) MFCCs data tuples from Persons 2 to 10 who formed the impostor class. The testing sets of the CV groups remain unchanged. The new ROC curves of the PSECMAC-based speaker verification system for the balanced training scenario are depicted as Appendix H.2 of the Thesis. Table 6.1(b) lists the EER readings for the evaluation phase of the PSECMAC-based speaker verification system that is constructed with the "balanced" training scenario for all the ten speakers.

Network	Average Equal Error Rate [%]										Average EER		EER Std Dev
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Total [%]		
CMAC	21.75	15.98	10.43	7.60	14.32	9.46	18.92	9.46	15.38	8.47	13.18		5.18
PSECMAC	15.28	13.05	8.50	8.58	16.13	12.29	14.95	10.53	12.24	9.22	12.08		2.72
HCAQ-CMAC	19.28	17.53	19.94	14.99	25.19	21.23	18.79	17.28	25.61	20.99	20.08		4.26
MLP (6-13-1)	36.93	39.59	24.92	21.38	10.15	24.54	27.60	40.17	36.09	19.44	28.08		17.17
RBF	22.30	17.95	16.44	6.42	20.10	11.70	22.51	10.77	17.13	12.37	15.77		5.29
GenSoFNN-CRI	22.20	16.02	18.76	11.69	20.26	17.67	13.60	11.60	19.18	13.92	16.49		3.94

P1, P2, ..., P10 refer to Person 1, Person 2, ..., Person 10 respectively.

(a) Unbalanced training scenario

Network	Average Equal Error Rate [%]										Average EER		EER Std Dev
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Total [%]		
CMAC	21.78	16.56	10.66	7.58	16.55	9.47	18.63	9.44	15.16	8.48	13.43		5.22
PSECMAC	11.84	12.48	9.96	7.81	12.93	10.53	14.03	9.51	14.98	9.54	11.36		3.48
HCAQ-CMAC	25.35	21.19	16.00	16.13	20.96	18.51	18.59	14.95	25.59	19.19	19.65		5.89
MLP (6-13-1)	28.72	33.62	9.86	5.63	13.68	12.74	23.98	15.19	8.77	17.27	16.95		9.16
RBF	19.19	17.20	14.01	8.85	18.10	16.68	16.79	11.72	13.47	11.15	14.72		2.76
GenSoFNN-CRI	22.38	15.66	16.68	12.53	21.28	19.48	15.50	12.68	17.95	16.04	17.02		3.78

P1, P2, ..., P10 refer to Person 1, Person 2, ..., Person 10 respectively.

(b) Balanced training scenario

Table 6.2: The performances of the benchmarked speaker verification systems

From the results tabulated in Tables 6.1(a) and 6.1(b), one can observe that the "balanced" training scenario improves the average EER performances of the PSECMAC-based speaker verification system by approximately 5.9% $((12.08 - 11.36)/12.08)$. The PSECMAC network employs a data-driven non-uniform memory allocation scheme that allocates more memory cells to the important (i.e. densely data-populated) regions of the input space. The overall improvement in the verification accuracy of the PSECMAC-based speaker verification system can therefore be attributed to the balanced data distribution of the positive (actual) and negative (impostor) samples in the training set, which allows for better memory cell allocations in the resultant trained PSECMAC speaker models. This generally results in the higher modeling accuracies of the speaker models that subsequently translate to improved speaker verification performances. However, a slight degradation is observed for the standard deviation of the average EERs (from 2.72% in the "unbalanced" training scenario to 3.48% for the "balanced" training scenario) across the ten evaluated speakers. This is due to the degradation in the ability to discern between the impostors and the authentic speaker for Persons 3, 7, 9 and 10. The drop in the verification performances for these four speakers are due to the inherent poorer quality of the collected speech samples as compared to the rest of the recruited speakers.

Table 6.2(b) tabulates the speaker verification performances of the benchmarked architectures for the "balanced" training scenario. From the results in Table 6.2(b), it is again evident that the PSECMAC-based speaker verification system comprehensively outperformed all the benchmarked systems based on the average EER values across all the evaluated speakers. Tables 6.2(a) and 6.2(b) also showed that the average EER values of the MLP network improved by 39.6% $((28.08 - 16.95)/28.08)$ as the data composition in the training sets is changed from the "unbalanced" to the "balanced" training scenario, thereby demonstrating the sensitivity of the MLP-based speaker models towards the structure of the training sets. Slight improvements were also noted for the HCAQ-CMAC and the RBF-based speaker verification systems and these can be attributed to the balanced data distribution of the two classes (authentic vs impostor) in the training sets. On the other hand, minor degradations in the performances of the CMAC and GenSoFNN-CRI-based speaker verification systems suggest that the duplicated positive samples introduced to the training sets have a slight detrimental effect on the performances of the corresponding speaker models. This may be due to the inherent learning and computational process of these two architectures.

6.4 Summary

Speaker verification via the use of sampled speech belongs to a class of biometric recognition problems that offered an alternative approach to the traditional knowledge and token-based techniques for automatic person authentication. The current research on speaker verification promote the use of statistical and probabilistic modeling of the speaker-specific characteristics as well as classical machine learning and pattern analysis-based speaker models. As the primary objective of a computerized speaker verification system is to be able to efficiently discern between an authentic speaker and an impostor, the accuracy of the speaker model employed to capture the speaker-specific characteristics extracted from the speech samples determines the performance level of the system.

In this chapter, a cerebellar-based approach to the text-dependent speaker verification problem is presented. The proposed speaker verification system employs the novel PSECMAC network, which is a neurologically-inspired computational model of the human cerebellum, to model the speaker-specific characteristics of the human voice via the MFCC values extracted from the sampled voice segments. The work in this chapter is motivated by the physiology of the human auditory system and the psychology of the human perception to acoustic sounds, which facilitate the human innate ability to accurately perform the speaker recognition process in everyday life.

The proposed PSECMAC-based speaker verification system is subsequently employed to verify the voice inputs of ten adult speakers. The verification performances of the PSECMAC speaker models are evaluated against those of the basic CMAC, HCAQ-CMAC, and GenSoFNN networks as well as the classical machine learning models of MLP and RBF networks. The experimental results have sufficiently demonstrated the superior accuracy of the PSECMAC-based speaker verification system to the benchmarked models.

The next chapter presents the application of the proposed HCAQ-CMAC and PSECMAC cerebellar models in the closed-loop regulation of exogenous insulin infusion for the personalized treatment of Type-1 diabetes.

"Every patient carries her or his own doctor inside"
–Albert Schweitzer

Chapter 7

Cerebellar-Based Insulin Pump Control for the Personalized Treatment of Diabetes

Diabetes is a chronic disease where the body is unable to properly and efficiently regulate the use and storage of glucose in the blood, leading to prolonged periods of high (hyperglycemia) or low (hypoglycemia) plasma glucose concentration. Chronic hyperglycemia causes severe damage to the eyes, kidneys, nerves, heart and blood vessels of the diabetic patients (Rubin et al., 1992) while severe hypoglycemia can deprive the human body of its primary energy source and cause a patient to lose consciousness, which may be life threatening (Cryer, 1992). The medical fraternity has classified the diabetes disease into two subtypes based on its pathogenesis:

1. Type-1 diabetes (juvenile onset or Insulin-Dependent-Diabetes-Mellitus).

The main cause is the death or the immuno-destruction of the insulin producing pancreatic beta cells, leading to an impaired capability of the pancreas to secrete the insulin hormone (Blasio et al., 1999).

2. Type-2 diabetes (adult onset or Non-Insulin-Dependent-Diabetes-Mellitus).

Obesity is the leading cause to a reduced insulin sensitivity in this group of patients, which disrupts the healthy glucose metabolic process as the tissue cells in the body are unable to absorb the energy providing glucose from the blood without sufficient insulin (DeFronzo, 1992).

The disease is reportedly the leading cause of adult blindness, end-stage renal failure, retinopathy, neuropathy, and lower-limb amputations. Diabetes patients are also 2-4 times more likely to develop medical conditions such as heart disease or stroke. As the number of diabetic patients increases worldwide (Wild et al., 2004), the proportion of national health care budgets allocated for diabetes treatment has multiplied. Diabetes exerts a heavy financial burden on the patients as well as the society due to the chronic nature and the severity of the complications related to the ailment. The American Diabetes Association has reported that diabetes is currently the fifth leading cause of death in the United States with an annual medical expenditure of approximately \$132 billion. This amount is projected to further increase to \$156 billion by 2010 and to \$192 billion by 2020 (American Diabetes Association, 2003). Driven by the concerns of escalating social and economic cost of the diabetic epidemic (World Health Organization, 2002), there has been an increased urgency in the research efforts to enhance the efficacy of diabetes treatment in the recent years. Many of the diabetes related medical complications, fortunately, can be prevented through the tight control of the diabetic blood glucose levels (Diabetes Control and Complication Trial, 1993, 1995).

The current standard treatment of diabetes primarily involves insulin medication coupled with strict dietary control. The insulin hormone can be administered subcutaneously, intravenously or peritoneally; and the insulin infusions are effected through discrete insulin injections or continuous insulin delivery via an insulin pump. There have been extensive studies of the advantages, disadvantages and peripheral issues with these insulin delivery approaches in the literature (Tyagi, 2002). A key component to the successful management of diabetes, however, is essentially to develop the ability to maintain a long-term near-normoglycaemia state of the patient (Rosenstock, 2001). Hence, discrete insulin injections are not therapeutically ideal for the treatment of diabetes as the regulation of the insulin hormone is an open-loop process. Continuous insulin infusion through a programmable insulin pump, on the other hand, offers an effective approach to normalize the diabetic blood glucose level due to the controllable insulin infusion rate (Fletcher et al., 2001).

Generally, the programmable insulin pumps are algorithmically driven and integrated with diabetes monitoring devices such as glucose sensors in a closed-loop paradigm to create a blood glucose regulatory system. An avalanche of regulatory techniques of these insulin pumps have been proposed, investigated and reported in the literature over the years (Fisher, 1991; Hovorka, 2005; Ollerton, 1989; Schetky et al., 2003; Sorensen, 1985).

The fundamental objective of these insulin pumps and their closed-loop systems is to artificially re-create (via variable insulin delivery) the healthy insulin profiles in a diabetic patient so as to regulate the diabetic blood glucose level within the homeostatic range of 60-110 mg/dl (Ashcroft and Ashcroft, 1992). Therefore, the performances of such pumps (and their regulatory systems) to manage diabetes correlate to their capability in replicating the insulin response of a healthy person. However, the majority of these insulin pumps currently employ static mathematical models of the human glucose metabolic process (obtained from data fitting of patient records, compartmentalized differential/difference equations, statistical or machine learning approaches) to compute the amount of insulin required by a diabetic patient. These models, however, often require manual tuning to cater for the metabolic biodiversity of the diabetic patients, as well as the intra and inter-day variability in the glucose metabolic rates of each specific patient (Bellazzi et al., 2001; Makroglou et al., 2006; Parker et al., 2001). Consequently, these closed-loop blood glucose regulatory systems are developed with fixed insulin regimes and required strict patient compliance to function properly. Even then, the therapeutic effects of these pseudo closed-loop systems remained poor and are too rigid and inflexible to be widely employed for diabetes treatment.

This chapter motivates the use of the PSECMAC network (presented in Chapter 4 of the Thesis) as a novel brain-inspired approach to manage diabetes mellitus. The human cerebellum is responsible for many sub-conscious but precise cognitive and behavioral manifestations (Kandel et al., 2000b). Therefore, the functional principles of the human cerebellum can be harnessed in a computational framework (i.e. PSECMAC) to model the biological autonomic decision processes of the pancreatic secretion of insulin to replicate the healthy insulin profile (via the control of an insulin pump) for the treatment of diabetes. In this chapter, the PSECMAC network is employed as a computational beta-cell to functionally model the biological decision process of insulin secretion in response to serum glucose fluctuations due to food ingestion, and to subsequently realize the physiological process of blood glucose regulation in a diabetic patient. The PSECMAC approach to the treatment of diabetes has a distinct advantage. One of the fundamental objectives of diabetes management research is to develop a treatment regime that is specifically tailored for an individual patient, i.e. personalized therapy to cater for the intra- and inter-day variability of the glucose metabolic process among the different individuals. The proposed PSECMAC insulin regulatory model can be easily adapted and customized to address the individual requirements of the patients in a diverse population. This specificity therefore ensures a close match

to the insulin needs of each patient to normalize the diabetic blood glucose levels and to prevent the over- or under-dosage of insulin bolus as seen in existing open-loop and static closed-loop insulin infusion systems.

The rest of this chapter is organized as follows. Section 7.1 reviews the current state of the art of diabetes therapeutic technologies reported in the literature. Section 7.2 describes the autonomous physiological process of pancreatic insulin secretion in the human glucose metabolism process that motivates the work in this chapter. The proposed cerebellar-based PSECMAC approach to diabetes treatment presented in this chapter consists of several stages. In Section 7.3, the patient profile and the dietary models employed in the study are first described. Section 7.4 presents the PSECMAC modeling of the healthy insulin response of the specified patient profile based on a normal diet. The resultant PSECMAC insulin model is subsequently evaluated on several different dietary habits in Section 7.5 to study the effect of the intra-day patient variability. Finally, in Section 7.6, the PSECMAC-based insulin model is employed in a proposed control system for the regulation of the plasma glucose level in a simulated diabetic patient. Section 7.7 summarizes this chapter.

7.1 Closed-Loop Glycaemic Controls for Diabetes Treatment

Currently, insulin is administered predominantly through discrete injections (3 or more times a day); and to a lesser extent, continuous infusions through the subcutaneous, intravenous or peritoneal pathways using a mechanical insulin pump. Other less established delivery routes that have been studied include oral and pulmonary administrations. Appendix I of the Thesis presents a brief review of the various types of insulin and the insulin delivery systems that are currently available. Despite much technological improvements, there remains numerous deficiencies associated with the use of these conventional insulin administration approaches that reduce their therapeutic effects in the treatment of diabetes. Firstly, the open-loop nature of these methodologies requires strict patient dietary adherence, as the amount of insulin (i.e. slow and fast-acting) to be administered is predetermined by a physician on the basis of a recommended diet. Secondly, these conventional approaches rely on direct patient intervention to adjust the insulin dosage used to counteract the blood glucose fluctuations arising from meal intakes. In practice, the diabetic plasma glucose level is controlled via three to five daily manual insulin injections where the

Figure 7.1: Taxonomy of Glucose-Insulin Regulatory Systems

insulin dose being administered is tuned according to three to seven daily blood glucose measurements as well as estimation of the food (carbohydrate) intake and the degree of physical exertion. For pediatric or an inexperience/negligent patient who fails to closely follow the stipulated dietary recommendations, it is difficult to achieve near-normal blood glucose levels. In addition, the treatment schedule does not account for the variability of the glucose metabolic cycle in the diabetic patient (i.e. sedentary vs. active lifestyle; nocturnal vs. daytime insulin requirements; and inter and intra-day metabolic variability) (Bremer and Gough, 1999; Simon et al., 1987). Consequently, due to the rigidity of the insulin medication regime, the patient experiences frequent and prolonged periods of elevated or depressed levels of blood sugar, which may eventually lead to medical complications.

Subsequently, there have been numerous attempts to construct a closed-loop blood glucose regulatory system for the treatment of diabetes via a programmable insulin pump (Bellazzi et al., 2001; Bequette, 2005; Parker et al., 2001). This is because continuous insulin infusion is found to be superior to multiple daily insulin injections in the treatment of diabetes (Boland et al., 1999; Hoogma et al., 2005; Wainstein et al., 2005), and thus pushing the research trend towards the development of an implantable insulin pump (Renard, 2004). The objective of a closed-loop glucose-insulin regulatory system is to control the release of insulin according to the variable needs of the diabetic patient. Such an approach eliminates the frequent occurrences of significant variations in the blood glucose concentration encountered by the open-loop methodologies. Current research work on the closed-loop regulation of blood glucose levels in diabetes treatment generally focusses on four different aspects (see Figure 7.1), namely: (1) the modeling approach of the glucose metabolic process, i.e. the patient model; (2) the meal announcement model; (3) the glucose monitoring and insulin delivery routes; and (4) the control paradigm adopted by the insulin pump to administer the required amount of insulin.

Based on the approach employed to model the glucose metabolic process of the diabetic patient, the various glucose–insulin regulatory systems in the literature can be broadly categorized into three types, namely: (1) *fundamental*; (2) *empirical*; and (3) *compartmental* modeling-based closed-loop systems. Fundamental models (De Gaetano and Arino, 2000; Keener, 2001; Puckett and Lightfoot, 1995; Sturis et al., 1991) are constructed by mathematically describing the known physiological glucose metabolic system behaviors such as the underlying glucose and insulin kinetics and transport dynamics in the human body. Various mathematical models based on Ordinary Differential Equations (ODE), Delay Differential Equations (DDE), Partial Differential Equations (PDE) etc. have been developed to model the mechanisms of the human biological glucose–insulin regulatory system (please refer to (Makroglou et al., 2006) for a review of these models). Due to the use of static mathematical equations, the fundamental modeling approach is inadequate when addressing the issue of patient specificity. The empirical approach, on the other hand, attempts to capture the human glucose metabolic behaviors from the observed input–output clinical data (Tresp et al., 1999). Based on a predetermined model structure, a set of empirical parameters is subsequently determined for the diabetic patient via data-fitting. The major drawback of the empirical approach stems from the data collection process, where the patients are often subjected to a clinical test environment that introduces artificial conditions such as fixed meal times, a carefully controlled dietary plan and limited exercise. Hence, the models developed to describe the glucose metabolic process of these patients may not accurately reflect the real-life conditions.

Therefore, the compartmental approach (Hovorka et al., 2004; Parker et al., 1999; Sorensen, 1985) is employed to combine the empirical and fundamental modeling techniques. Such models are derived by compartmentalizing the individual physiological component affecting the glucose metabolism process in the diabetic patients. Each of the components is described using fundamentally derived mathematical equations. However, patient specific parameters are obtained empirically via data fitting. Despite being the most popular approach used to construct the patient model, compartmental modeling has been associated with several technical issues impeding its effective usage. That is, some of the endocrine processes affecting the human glucose metabolism cycle are still not yet fully understood at present and therefore inhibits the effectiveness and accuracy of the compartmental modeling technique. Furthermore, the current modeling knowledge of the glycaemic effects of food, exercise, emotional stress, and other bodily functions are still very limited.

The second differentiating aspect to categorize current research efforts on closed-loop glucose-insulin regulatory systems is the meal announcement model. Meal announcement (Hovorka, 2006) refers to the way a glucose-insulin regulatory system handles the food intake disturbances. A system is a *blind* closed-loop controller when it performs glucose control without information about the time or size of the food intakes. The insulin infusion is purely controlled based on the (post-prandial) glucose fluctuations. An *informed* closed-loop regulatory system, on the other hand, is provided with the time and the size of meals, and this information is used to compute the required amount of insulin to be administered. Finally, a system is referred to as an *anticipated* closed-loop system when the controller is provided with the time of a meal, but not its size.

With respect to the issue of implementation, current closed-loop regulatory systems can be differentiated based on the glucose monitoring scheme and insulin delivery mechanism. In general, there are: (1) the subcutaneous-subcutaneous (SC-SC) route; (2) the intravenous-intraperitoneal (IV-IP) route; and (3) the intravenous-intravenous (IV-IV) route. Currently, the SC-SC route is the least invasive solution to glycemic control. That is, the glucose monitoring device and the point of external insulin infusion are located subcutaneously in a diabetic patient. However, the large delays associated with this route (in the range of ± 100 min) (Hovorka, 2006) presents problems with the effective regulation of large meal intake disturbances. The IV-IP route, on the other hand, allows for shorter response times and more accurate glucose measurements. The intraperitoneal delivery of insulin is also a more physiological insulin delivery mechanism (Tyagi, 2002). This approach, however, requires the surgical implantation of the insulin pump. Lastly, the IV-IV route is currently explored for critically ill patients for which direct IV access is necessary.

In addition, there are generally five types of control algorithms employed by the existing glucose-insulin regulatory systems. The *classical* control approach (Bellomo et al., 1982; Bergman et al., 1981; Fischer et al., 1990; Pfeiffer et al., 1974) refers to the collection of classical feedback control algorithms embodied in a traditional proportional-integral-derivative (PID) pump controller. Such control algorithm is usually derived from either the fundamental or compartmental models of the human glucose metabolic process. The *adaptive* control approach (Brunetti et al., 1993; Hovorka et al., 2004; Parker et al., 1999), on the other hand, refers to the glucose-insulin regulatory systems that adaptively update the control parameters at each time step to improve the performances of the controllers. In *optimal* control (Fisher and Teo, 1989; Swan, 1982), a family of convoluted control

algorithms is employed to compute the required amount of insulin to achieve an optimal control of the diabetic blood glucose level. The *predictive* control approach (Hovorka et al., 2004; Parker et al., 1999; Trajanoski and Wach, 1998), on the other hand, exploits the prediction capability of the integrated patient model to compute future glucose values based on the past glucose measurements and insulin inputs. The key benefit of using predictive control is the estimation of future glucose behavior, thus allowing for the adjustment in the rate of insulin delivery in response to the predicted hyper or hypoglycemia episodes. Lastly, in the *algorithmic* control approach (Albisser, 1989; Albisser et al., 1996), optimal control decisions for the insulin pump are selected according to formulated guidelines containing direct instructions of actions in different therapeutic situations. These guidelines consist of clinical therapeutic strategies that are encapsulated as rules or mathematical formulas to effectively bring the patient's blood glucose level into the normo-glycaemia range.

Appendix J of the Thesis summarizes some of the current efforts in developing a closed-loop blood glucose regulatory system for diabetes treatment. Several observations can be made from the review. Firstly, the performance of the insulin pump is directly linked to the accuracy of the patient model. This implies that empirically determined model-based methods may have an inherent advantage over fundamental model-based approaches in blood glucose control provided that accurate patient models are used in the synthesis of the pump controller. The empirical modeling approach attempts to capture the behavior of a system from input-output data. The model structure may be chosen a-priori or adapted with learning, and individualized empirical parameters are determined through a series of tests performed on each patient (patient specificity). Fundamental models, on the other hand, are constructed by mathematically describing known system behaviors, such as the underlying physiologic kinetics, material transport dynamics or mass balance equations. Secondly, patient specificity and metabolic variability are important considerations for the efficient regulation of the blood glucose concentration of a diabetic patient. A majority of the existing programmable insulin pumps is based on control regimes that are derived from a static nominal patient model and/or a static predefined control structure. The inherently static nature of the patient model limits the capability of the proposed insulin pump control regimes to dynamically adapt to 'drifts' in the system, i.e. the inter and intra-day variability of the patient's metabolic profile (Bremer and Gough, 1999; Simon et al., 1987). In addition, the formulated control algorithms are not personalized and cannot be easily tailored to suit a particular patient's needs. This is clinically not ideal as the

physiological conditions of the diabetes patients tend to vary due to biodiversity. Such a constraint results in less-than-satisfactory performances of the proposed regulatory systems when applied to different diabetic patients. These systems also failed to address the inter- and intra-day variability in the metabolic process of an individual patient. Hence, the ability to develop a personalized treatment regime that is specifically tailored to a patient is critical to the objective of maintaining long-term normoglycemia of the diabetic patient.

7.2 The Physiological Process of Pancreatic Insulin Secretion

Insulin is a metabolic hormone that ensure homeostasis of the human blood glucose level (Eple and Brinn, 1987). Much progress has been made in the last three decades to characterize the metabolic pathways that are involved in the physiological process of insulin secretion. Of all the pathways identified so far, the mechanisms of glucose metabolism in triggering insulin secretion from the pancreatic β -cells have been the most extensively studied (Doliba and Matschinsky, 2003). Blood glucose is the most effective physiological nutrient stimulus of insulin secretion (Matschinsky, 1996). The pancreatic β -cells are physiologically designed to sense the level of glucose in the blood on a moment-to-moment basis. This in turn triggers the secretion of insulin at rates that are effectively appropriate to ensure an optimal level of circulating nutrient in the blood (Doliba and Matschinsky, 2003). When the blood glucose level increases, insulin secretion is enhanced with a characteristic dependency.

Traditionally, the circulating blood glucose level has been viewed as the major stimulant for pancreatic insulin secretion; and insulin in turn is required by the peripheral tissues to extract the nutrient supplying glucose from the blood. The brain is speculated to be independent of this autonomous process (Woods et al., 1985). In recent years, emerging scientific evidences have suggested that glucose metabolism throughout the human body may in fact be coordinated by the brain through the use of insulin (Schwartz and Porte Jr., 2005). New evidences regarding the existence of neuronal circuitries controlling food intake has extended the scientific community's understanding of energy homeostasis, the process whereby energy intake is matched to energy expenditure over time (Schwartz, 2000). It is an important function of the human brain to ensure a steady supply of energy substrates to maintain bodily equilibrium. To accomplish this objective, afferent signals are integrated at the brain and transduced into homeostatic adjustments of food intakes, energy expenditures

Figure 7.2: The human model of energy and glucose homeostasis. Adapted from (Porte Jr. et al., 2005)

and to expediate nutrient metabolism (Porte Jr. et al., 2005).

The complex and intertwined relationship of energy and glucose homeostasis in the human body is depicted in Figure 7.2. As a central regulatory system for energy homeostasis, the brain has the control of numerous metabolic parameters that affects the feeding behavior as well as the ability to directly alter the rate of insulin secretion from the pancreas. There are several pathways through which the brain can influence insulin secretion (Woods et al., 1985). Firstly, there are direct neural inputs to the islet of Langerhans via the sympathetic and parasympathetic branches of the autonomic nervous system. Clinical experiments have demonstrated that increased activity in the parasympathetic nervous system increases pancreatic insulin secretion (Bergman and Miller, 1973; Helman et al., 1982; Ionescu et al., 1983), while electrical stimulation of the sympathetic nerves typically decreases insulin secretion (Miller, 1981; Woods and Porte Jr., 1974). Secondly, there are hormone secretions by the brain-pituitary system and other hormonal systems that either act directly on the pancreas or influence the glucose levels and/or insulin secretion indirectly. In turn, the circulating glucose and insulin levels are fed back to the brain to regulate the amount of food intake and body adiposity to complete the feedback loop. This is evidenced by the presence of glucose-sensors and insulin-sensitive neurons in the brain regions that are reckoned to be involved with the control of food intake (Levin et al., 2004; Woods et al., 1985).

It is well-established that the liver plays a central role in the human glucose metabolism

process by acting as a glucose buffer; that is, extracting glucose from the bloodstream in times of plenty and synthesizing glucose when needed by recognizing the different bodily energy states through the detection of changes in the blood insulin concentration (Gribble, 2005). Hitherto, insulin is known to target the liver directly. However, precise experimentations have subsequently demonstrated that insulin, via acting on the hypothalamus (a subcortical brain structure central to the autonomic control of the human endocrine system), exerts a higher level of supervisory control on glucose production by the liver (Pocai et al., 2005). This observation suggests that insulin can in fact modulate liver glucose production through a signaling pathway of the brain system (Fisher and Kahn, 2003; Obici et al., 2002). In addition, biophysiological studies have established the presence of an inhibitory physiological response to food intake when the insulin hormone is administered directly to the brain system, particularly the hypothalamus region (Woods et al., 1979, 1998). Hence, insulin appears to be required by the human brain to regulate food intake, body weight and homeostasis of numerous physiological processes (Schwartz, 2000).

Therefore, a complex system exists for direct neural alteration of insulin secretion from the pancreas. The various facts presented above have sufficiently demonstrated that the human insulin regulatory mechanism is a complex autonomous biological decision process, in which each input factor influences one another in a highly dynamic and nonlinear manner. This observation subsequently motivates the use of the PSECMAC network, which is a cerebellar-inspired adaptive learning memory model, to capture the dynamics of the metabolic insulin profile of a healthy person, and subsequently to synthesize the insulin response required to achieve homeostasis in the glucose metabolic process of a diabetic patient.

7.3 The Patient Profile – A Simulated Human Subject

The first step into constructing the personalized insulin regulation system is to determine the patient profile to be employed in the study. Due to the lack of real-life patient data and the logistical difficulties and ethical issues involving the collection of such data, a well-known web-based simulator known as *GlucoSim* (GlucoSim, Online) from the Illinois Institute of Technology is employed to simulate a person subject to generate the blood glucose and insulin data that is needed for the construction of the healthy and diabetic patient models. For this purpose, a human profile for the simulated subject (Subject A) is created and

Table 7.1: The profile of the simulated healthy Subject A

Attribute Name	Attribute Value
Sex	Male
Age	40 years old
Race	Asian
Weight	67 kg (147.71 lbs)
Height	1.70 m (5 ft 7 in)
BMI	23 (Recommended for Asian)
Lifestyle	Typical office worker with moderate physical activities such as walking briskly, leisure cycling and swimming.

described in Table 7.1. The simulated person, Subject A, is a typical middle-aged Asian male. His body mass index (BMI) is 23.0 and within the recommended range for Asian.

The GlucoSim simulator caters for two different types of modeling, namely the glucose metabolic process of a healthy person and that of a Type-1 diabetic patient respectively. The inputs to the GlucoSim simulator for the healthy person model are time and carbohydrate content of the meals taken by the person, the body weight as well as the duration for which the simulation is to be performed. An additional input of the amount of insulin injection is required for the diabetic patient model. The GlucoSim simulator employs a compartmental modeling technique of the human glucose metabolism process. The mathematical models used in GlucoSim to describe the glucose-insulin interactions in both the healthy and diabetic subjects are based on the work of (Puckett, 1992) and (Sorensen, 1985) respectively. The pharmacokinetic diagrams of the glucose and insulin models adopted by the GlucoSim simulator is presented as Appendix K of the Thesis.

Based on the profile of Subject A, his recommended daily allowance (RDA) of carbohydrate intake from meals is computed using an applet from the website of the Health Promotion Board of Singapore (HPBSg, Online). According to his sex, age, weight and lifestyle, the recommended daily carbohydrate intake for Subject A is approximately 346.9g per day. A sample menu is presented as Appendix L of the Thesis to illustrate the typical daily meals (i.e. breakfast, lunch, afternoon snack and dinner) and the carbohydrate contents of the various meals of Subject A based on his Asian diet. The nutritional composition is based on the recommended servings for each of the major food groups in the food pyramid. The carbohydrate contents of the food in the meals are derived using the publicly available Atkins Carb Gram Counter (Atkins, Online).

Figure 7.3: Sample glucose metabolism data output from the GlucoSim simulator

Figure 7.3 illustrates a sample output from GlucoSim for Subject A based on the carbohydrate intakes of Appendix L. This output consists of six elements: blood glucose, blood insulin, intestinal glucose absorption rate, stomach glucose, total glucose uptake rate and liver glucose production rate of Subject A respectively over a simulated time period of 24 hours. The peaks in the stomach glucose subplot of Figure 7.3 coincide with the timings of the assumed daily four meals (i.e. breakfast, lunch, afternoon snack and dinner) while those peaks in the intestinal glucose absorption rate subplot reflect a delay effect (response) of food intake on the blood glucose level of Subject A. The subplots of blood glucose and blood insulin illustrate the insulin-glucose regulatory mechanism in a healthy person such as Subject A and depict the dynamics of the metabolic process when subjected to disturbances such as food intakes.

Subsequently, *four* types of dietary profiles are defined to simulate the variations in the dietary habit of a human subject. These dietary profile are denoted as *normal*, *under*, *over*,

and *irregular* diet. They correspond to the normal, under-eating, overeating, and irregular profiles respectively. The normal diet refers to a dietary profile that conforms to the RDA of the carbohydrate intake of the subject. In this study, the carbohydrate intake of a normal diet for Subject A is defined by eq. (7.1).

$$\text{normal} \in [0.85 \times \text{RDA}, 1.15 \times \text{RDA}] \quad (7.1)$$

Similarly, the carbohydrate intakes of the under-eating and the overeating profiles are defined by eqs. (7.2) and (7.3) respectively.

$$\text{under} \in [0.425 \times \text{RDA}, 0.575 \times \text{RDA}] \quad (7.2)$$

$$\text{over} \in [1.275 \times \text{RDA}, 1.725 \times \text{RDA}] \quad (7.3)$$

The normal, under and over diet profiles are each characterized by *three* regular meals (i.e. breakfast, lunch and dinner) and *one* afternoon snack. The number of meals in the irregular profile, on the other hand, varies between *two* to *six* meals a day, with the carbohydrate intake defined by eq. (7.4).

$$\text{irregular} \in [0.5, 2.0] \times \text{normal} \quad (7.4)$$

Likewise, the meal timings of the irregular profile are randomized within the day.

In addition, the carbohydrate contents and the timings of the daily meals for each of the normal, under and over diets are not fixed and are varied from day-to-day during the data collection phase. To account for the inter and intra-day variability of the meal contents and the eating habits of Subject A, the computation listed in Table 7.2 were performed to generate different sets of inputs for each day of the simulated period. These inputs are subsequently used with the GlucoSim simulator to generate the glucose and insulin data.

7.4 The PSECMAC Insulin Model

The objective of the study presented in this section is to model the insulin profile of a healthy person. It has been established in Section 7.2 that blood glucose is the most effective physiological nutrient stimulus of pancreatic insulin secretion (Matschinsky, 1996).

Table 7.2: Computations for the generation of GlucoSim input parameters for the normal, under and over dietary profiles. (Note: 100 sets of input parameters are generated. $U(x,y)$: a uniformly distributed random number between x and y inclusively; and $N(\mu,\sigma)$: a normally distributed random number with mean μ and standard deviation σ .)

GlucoSim Input	Notation	Value	Remarks
Meal Timings			
Breakfast Time	BkTime	$U(0700hrs,0900hrs)$	Only four meals* per day and meal timings are typical of an office worker
Lunch Time	LunTime	$U(1130hrs,1200hrs)$	
Afternoon Snack Time	ASTime	$U(1500hrs,1600hrs)$	
Dinner Time	DinTime	$U(1830hrs,2030hrs)$	
			Meal timings are uniformly randomized within their specific ranges
Total Carbohydrate Intake Per Day			
Breakfast Carbohydrate	BkCarb	$U(12\%,18\%) \times \text{TotalCarb}$	TotalCarb = 350g + $N(\mu=0g,\sigma=50g)$ (RDA for Subject A is approx. 350g) Carbohydrate percentages are uniformly randomized within their specific ranges and normalized so that total sum is 100%
Lunch Carbohydrate	LunCarb	$U(25\%,35\%) \times \text{TotalCarb}$	
Afternoon Snack Carbohydrate	ASCarb	$U(13\%,19\%) \times \text{TotalCarb}$	
Dinner Carbohydrate	DinCarb	$U(35\%,45\%) \times \text{TotalCarb}$	

*It is assumed that Subject A does not take morning and evening snacks. Hence the morning and evening snack timings are kept constant at 1000 and 2200 hours during input to the simulator, and their respective carbohydrate contents are preset to 0g.

In addition, the existence of biological glucose sensors in the pancreas as well as at the various regions of the human brain further supports the hypothesis that the blood insulin concentration level at any given time is a nonlinear function of the blood glucose dynamics. Therefore, in this section, the PSECMAC network proposed in Chapter 4 of the Thesis is employed to capture the plasma insulin response of Subject A to food ingestion based on the current and past plasma glucose information.

The GlucoSim simulator is employed to generate a total of eight days of glucose and insulin data based on the profile of Subject A and his normal dietary habit. The collected metabolic data is subsequently partitioned into two groups: the first 4-days data is used for training the PSECMAC network, while the remaining 4-days data is used for the evaluation of the trained network. A sampling interval of 5 minutes is adopted to discretize the measurements of the blood glucose and insulin concentrations. Let $I_x^{H(P)}$ denotes the insulin profile of the healthy subject P under the dietary profile $x \in \{\text{normal, under, over, irregular}\}$.

The insulin relationship to be modelled by the PSECMAC network is formalized as eq. (7.5)

$$\hat{I}_{\text{normal}}^{H(A)}(t+1) = \mathfrak{F}_{\text{normal}}^{H(A)}\left(\left\{Z_{\text{normal}}^{H(A)}(t)\right\}\right) \quad (7.5)$$

where $\hat{I}_{\text{normal}}^{H(A)}(t+1)$ is the predicted blood insulin concentration at time $t+1$; $\left\{Z_{\text{normal}}^{H(A)}(t)\right\}$ denotes the information set that characterizes the glucose metabolic process of the healthy Subject A due to a normal diet at time t ; and $\mathfrak{F}(\cdot)$ is a nonlinear function that implements the insulin model mapping from the input metabolic variables $\left\{Z_{\text{normal}}^{H(A)}(t)\right\}$ to the desired output, that is, the blood insulin concentration at the next sampling instance $I_{\text{normal}}^{H(A)}(t+1)$.

Based on the collected blood glucose data, a total of 18 variables (consisting of the current and past blood glucose measurements and its derivatives) are extracted as inputs to model the healthy insulin profile of Subject A. These variables are outlined as Appendix M of the Thesis. Due to the large number of input features available (18 variables), a novel feature selection algorithm named Monte Carlo Evaluative Selection (MCES) (Quah and Quek, 2007) is employed to identify the prominent features that best characterize the insulin response of a healthy person. That is, given by eq. (7.6), for the normal diet,

$$\hat{I}_{\text{normal}}^{H(A)}(t+1) = \mathfrak{F}_{\text{normal}}^{H(A)}\left(\mathfrak{R}\left(\left\{Z_{\text{normal}}^{H(A)}(t)\right\}\right)\right) \quad (7.6)$$

where $\mathfrak{R} = \text{MCES}$ denotes the MCES feature selection process. The reduced set of in-

puts/features for the normal diet is subsequently denoted by eq (7.7).

$$\left\{ z_{\text{MCES, normal}}^{H(A)}(t) \right\} = \Re \left(\left\{ Z_{\text{normal}}^{H(A)}(t) \right\} \right) \quad (7.7)$$

The MCES method has the advantages of (1) low computational cost; (2) the ability to identify both *correlated* and *irrelevant* features based on weight ranking; (3) being applicable to both classification and regression tasks; and (4) is independent of the underlying induction algorithm used to perform the feature selection process. The MCES algorithm is attached as Appendix N of the Thesis.

The MCES algorithm is executed independently for 50 times, where in each run, 50 iterations on the training set (first 4-days of metabolic data) is performed. The feature ranking results for 50 independent executions of the MCES algorithm are aggregated to determine the relevant features for the insulin modeling task. The salient/prominent features are identified based on their rankings and associated weights (evaluative feedback values). Appendix M lists the results of the MCES algorithm based on the 18 glucose variables for the insulin modeling task. The top four features, namely: the current glucose level ($G(t)$), the 4-point exponential moving average (EMA, refer to Appendix M) of the glucose level ($G_{\text{MA}_M}(t)$), the delta change in the glucose level over the last 5 minutes ($dG(t)$), and the 2-point EMA of the glucose level ($G_{\text{MA}_S}(t)$) are selected as the glucose indicators/inputs to the PSECMAC insulin model. The computational objective of the PSECMAC insulin model is subsequently formalized as eq. (7.8)

$$\begin{aligned} & \left\| I_{\text{normal}}^{H(A)}(t+1) - \hat{I}_{\text{normal}}^{H(A)}(t+1) \right\|_{\text{minimize}} \\ &= \left\| I_{\text{normal}}^{H(A)}(t+1) - \mathfrak{F}_{\text{normal}}^{H(A)} \left(\left\{ z_{\text{MCES, normal}}^{H(A)}(t) \right\} \right) \right\|_{\text{minimize}} \end{aligned} \quad (7.8)$$

where $I_{\text{normal}}^{H(A)}(t+1)$ is the actual measured blood insulin level at time $t+1$.

A PSECMAC network with a memory size of 8 cells per dimension is constructed to model the insulin profile of the healthy subject as shown in eq. (7.6). A neighborhood size (N) of 0.1 and a Gaussian width constant (γ) of 0.3 have been empirically determined to give the optimal modeling performance. As benchmarks, the insulin modeling task is also performed using various well-established empirical models. The benchmarking models studied in this work are the basic CMAC network (Albus, 1975a,b); two other cerebellar-based architectures, namely: (1) the HCAQ-CMAC network (refer to Chapter 3 of the Thesis) and

(2) the Fuzzy CMAC with Yager Inference Scheme (FCMAC-Yager) (Sim et al., 2006); a well-established neuro-fuzzy system termed the Generic Self-Organizing Fuzzy Neural Network with the Compositional Rule of Inference reasoning schema (GenSoFNN-CRI) (Tung et al., 2004); as well as the classical machine learning models of the Radial Basis Function (RBF) network (WEKA, Online) and the Multi-Layered Perceptron (MLP). The parameters for the FCMAC-Yager and the GenSoFNN-CRI systems have all been empirically optimized for best performances. There are two network structures of the MLP, each having one and two hidden layers respectively. These have also been empirically determined. The RBF network is initialized to contain 50 hidden layer nodes. In addition, the size of the CMAC and the HCAQ-CMAC networks has been defined as 8 cells per dimension for a fair comparison with the proposed PSECMAC network.

Table 7.3 lists the *recall* (in-sample testing) and the *generalization* (out-of-sample testing) performances of the various benchmarked insulin models. *RMSE* denotes the root-mean-squared-error between the set of computed and expected insulin levels; and *PC* is the Pearson correlation coefficient, a statistical measure reflecting the goodness-of-fit between the computed and expected insulin dynamics. A *performance index* (PI_1) measure is used to combine the RMSE and the PC values of the benchmarked networks as described in eq. (7.9).

$$PI_1 = \frac{PC}{1 + RMSE} \times 100, \quad PI_1 \in [-100, 100] \quad (7.9)$$

such that a higher PI_1 value corresponds to a better overall prediction performance of the insulin model. In addition, the generalization results are also reported in terms of the *mean-absolute-error* (MAE) and the *mean-squared-error* (MSE) values of the computed insulin response. The MSE measure magnifies the larger errors between the computed and the actual insulin concentrations; hence the impact of these errors is pronounced for this measure. Together with the MAE value, this would allow one to discern amongst the insulin models that give consistent but minute errors from the insulin models that provide highly accurate predictions at most of the sampled points but with occasional large errors. The MSE and MAE measures are subsequently combined as shown in eq. (7.10).

$$PI_2 = \frac{MSE}{1 + MAE}, \quad PI_2 \in [0, \infty] \quad (7.10)$$

such that a lower PI_2 value implies a more consistent prediction performance of the model.

Table 7.3: Simulation results for the various insulin models

Network	Recall			Generalization					
	RMSE	PC	PI ₁	RMSE	PC	PI ₁	MAE	MSE	PI ₂
PSECMAC	6.3011	0.9918	13.58	4.0737	0.9948	19.61	2.1187	16.595	5.32
CMAC	4.4990	0.9958	18.11	6.6692	0.9880	12.88	4.6351	44.478	7.89
HCAQ-CMAC	5.4990	0.9937	15.29	4.6476	0.9936	17.59	3.2322	30.239	7.15
FCMAC-Yager	6.7013	0.9929	12.89	6.8474	0.9899	12.61	6.0575	46.887	6.64
GenSoFNN-CRI	6.6710	0.9944	12.96	5.8942	0.9953	14.44	4.7014	34.742	6.09
MLP (4-120-1)	26.337	0.8861	3.24	24.291	0.8552	3.38	20.666	590.05	27.23
MLP (4-20-4-1)	23.450	0.8908	3.64	21.757	0.8607	3.78	18.637	473.37	24.11
RBF	6.4141	0.9915	13.37	5.3977	0.9906	15.48	3.4419	29.135	6.56

Figure 7.4: Measurement of prediction performance using the MSE and MAE indicators

Figure 7.4 further illustrates the use of the MSE and MAE values to quantify the quality of a set of predicted outputs. Let *Computed 1* and *Computed 2* denote the computed outputs of Model 1 and Model 2 respectively for the sine function. As shown in Figure 7.4, the average computational error of Model 1 (i.e. $MAE = 10$) is larger than the average computational error exhibited by Model 2 ($MAE = 3.84$). These MAE values, however, do not reflect the complete analysis as the goodness of the prediction performance of the respective models has not been examined. A small MAE value merely implies that the computations are accurate for most of the sampled outputs, while the effect of large glitches often get attenuated by the averaging process. In the computation of the MSE measure, however, large output errors are magnified, and thus their effects become pronounced as illustrated by the MSE values of Model 1 and Model 2 respectively. Therefore, by aggregating the MSE and MAE

measures, the PI_2 value (refer to eq. (7.10)) is able to provide a measure of consistency in the computed outputs of a particular insulin model. From Figure 7.4, one can observe that Model 1 (with a consistent error) achieves a lower PI_2 value in comparison with Model 2 that produces occasional large glitches in its predictions.

As shown in Table 7.3, the PSECMAC network achieved the best generalization performances among all the benchmarked models. The generalization evaluation of the PSECMAC network results in the highest PI_1 value and the lowest PI_2 value, which demonstrate the accuracy and consistency in its predicted insulin responses. The generalization results of the PSECMAC network outperformed those of the benchmarked cerebellar-based architectures (i.e. the CMAC, HCAQ-CMAC and FCMAC-Yager networks), thereby demonstrating the effectiveness of the PSECMAC network as a cerebellar-based insulin model. While the uniform quantization process of the CMAC network results in a lower accuracy of the computed output, the non-uniform quantization process of the HCAQ-CMAC network employs a hierarchical clustering technique that groups together neighboring regions of inputs having similar outputs. Although this approach is highly effective in optimizing the generalization ability of the HCAQ-CMAC network, it may result in a reduced output accuracy as in the case of the insulin prediction task. The FCMAC-Yager network, on the other hand, is a Mamdani fuzzy rule-based system that adopts trapezoidal-shaped fuzzy sets as membership functions. This often leads to a low output accuracy due to the coarse granularity of the membership functions.

The recall performances of the PSECMAC insulin model, however, are slightly inferior to those of its uniformly-quantized CMAC counterpart. This is because the static structure of the basic CMAC network results in a model that is optimized for the training set. The adaptive memory quantization mechanism of the PSECMAC network, on the other hand, is better equipped towards obtaining an efficient characterization of the complex and intertwined glucose–insulin relationships. This enables the PSECMAC network to achieve an improved generalization result despite poorer recall performances as demonstrated by the simulation results on the insulin prediction task. In contrast, a significant performance degradation is observed for the CMAC insulin model as the emphasis shifted from the recall to the generalization evaluation.

In addition, the PSECMAC insulin model achieved a 26.7% higher $((19.61 - 15.48)/15.48)$ PI_1 value and a 18.9% $((6.56 - 5.32)/5.32)$ lower PI_2 value over the best performing bench-

marked non-cerebellar-based model (i.e. RBF) for the generalization evaluation. The PSECMAC network has comprehensively outperformed the benchmarked GenSoFNN-CRI neuro-fuzzy system and the classical machine learning technique-based (i.e. MLP, RBF) insulin models. The simulation results outlined in Table 7.3 have also demonstrated the inability of the MLP network in capturing the underlying relationships between the selected glucose indicators and the desired insulin responses. Both the 3-layers and 4-layers MLPs reported the poorest recall and generalization performances amongst the benchmarked systems.

Figure 7.5 depicts a 3-days snapshot of the generalization performances of the CMAC and PSECMAC insulin models. Simulation results shown in Figure 7.5 and Table 7.3 have sufficiently demonstrated the highly encouraging accuracy of the PSECMAC insulin model in predicting the correct insulin response based on the selected glucose indicators.

7.5 Modeling the Insulin Response under Different Dietary Profiles

As discussed in the previous sections, it is highly desirable to have an accurate glucose–insulin response model that is able to address the intra- and inter-day variability observed in the human glucose metabolic process for the treatment of diabetes mellitus. In this section, the robustness of the proposed PSECMAC insulin model against the intra-day variability in the metabolic process of a healthy person is investigated.

The objective of the study is to evaluate how the insulin model $\mathfrak{F}_{\text{normal}}^{H(A)}$ (see eq. (7.6)) responds to the different dietary profiles of Subject A. For example, the computed insulin response when Subject A *overeats* based on the insulin model $\mathfrak{F}_{\text{normal}}^{H(A)}$ is defined as eq. (7.11)

$$\hat{I}_{\text{normal(over)}}^{H(A)}(t+1) = \mathfrak{F}_{\text{normal}}^{H(A)}\left(\left\{z_{\text{MCES,over}}^{H(A)}(t)\right\}\right) \quad (7.11)$$

where $\left\{z_{\text{MCES,over}}^{H(A)}(t)\right\}$ denotes the set of glucose indicators extracted from the dietary data of Subject A when he overeats. Note that the set of glucose indicators fed as inputs to the PSECMAC insulin model is the same for all the dietary profiles evaluated in this section. That is, the inputs are the salient glucose variables identified by the MCES feature selection process in Section 7.4. Hence, the purpose of the experiments performed in this section is essentially to ascertain whether the insulin response $\hat{I}_{\text{normal(x)}}^{H(A)}(t+1)$ is a good approximator

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.5: 3-days generalization performances of the CMAC and PSECMAC networks in modeling the insulin profile of a healthy person

Table 7.4: Simulation results of the insulin models for the various dietary profiles

Network	Dietary Profile	RMSE	PC	PI ₁	MAE	MSE	PI ₂
PSECMAC	Normal	4.0737	0.9948	19.61	2.1187	16.595	5.32
	Undereat	3.3064	0.9930	23.06	1.6454	10.932	4.13
	Overeat	17.668	0.9765	5.23	5.8056	312.16	45.87
	Irregular	34.769	0.9259	2.59	7.4270	1208.9	143.5
CMAC	Normal	6.6692	0.9880	12.88	4.6351	44.478	7.89
	Undereat	5.8997	0.9818	14.23	3.8684	34.806	7.15
	Overeat	24.680	0.9433	3.67	7.9923	609.10	67.74
	Irregular	40.319	0.8797	2.13	9.4144	1625.6	156.1

Table 7.5: Performance comparisons between the PSECMAC and CMAC insulin models for the various dietary profiles

Dietary Profile	PI ₁			PI ₂		
	PSECMAC	CMAC	% Gain	PSECMAC	CMAC	% Gain
Normal	19.61	12.88	52.25	5.32	7.89	32.57
Undereat	23.06	14.23	62.05	4.13	7.15	42.24
Overeat	5.23	3.67	42.51	45.87	67.74	32.29
Irregular	2.59	2.13	21.60	143.5	156.1	8.07
Average			44.60%			28.79%

of $I_x^{H(A)}(t+1)$, where $x \in \{\text{under, over, irregular}\}$ denotes the different dietary profiles, $\hat{I}_{\text{normal}(x)}^{H(A)}(t+1)$ is the insulin response computed with the insulin model $\mathfrak{F}_{\text{normal}}^{H(A)}$ that is constructed from the dietary data of Subject A according to his normal diet, and $I_x^{H(A)}(t+1)$ is the actual insulin response as observed from the metabolic data of Subject A.

For the experiments, a set of 4 days of glucose and insulin data is generated for each of the dietary profiles using GlucoSim. The PSECMAC insulin model trained with the data extracted from the normal dietary profile (as in Section 7.4) is then applied to respectively predict the healthy insulin responses for the different dietary profiles (i.e. under, over and irregular eating). Table 7.4 tabulates the performances of the PSECMAC network in predicting the insulin responses for the various dietary profiles. The RMSE, PC, PI₁, MAE, MSE, and PI₂ values (refer to Section 7.4 for their definitions) are employed as the performance measures of the PSECMAC insulin model. As a benchmark, the set of simulations is repeated using the similarly trained CMAC insulin model and the results are presented in Table 7.4. Table 7.5 subsequently lists the detailed performance comparisons between the PSECMAC and CMAC insulin models for all the dietary profiles.

From the results tabulated in Tables 7.4 and 7.5, one can observe that the PSECMAC

insulin model has comprehensively outperformed the benchmarked CMAC insulin model for all the evaluated dietary profiles. The PSECMAC insulin model yielded, on average, a 44% increment in PI_1 and a 29% decrement in PI_2 over the CMAC insulin model. The highest performance gains of the PSECMAC over the CMAC insulin model were noted for the undereat diet, with gains of 62% and 42.2% in the PI_1 and PI_2 values respectively. These gains are even higher than those for the normal diet (i.e. PI_1 gain of 52.3% and PI_2 gain of 32.6%), for which the PSECMAC and CMAC insulin models have been specifically trained. However, Table 7.5 also shows that the performance gains of the PSECMAC over the CMAC insulin model are lower for the overeat and irregular dietary profiles. This is due to the substantial decrease in the PI_1 and PI_2 values achieved by the PSECMAC insulin model for the overeat and irregular diets despite its superiority over the benchmarked CMAC insulin model. These observations are analyzed as follows.

The undereat dietary profile provides an evaluation on the sensitivity of the trained insulin models to the smaller fluctuations in the blood glucose concentration resulting from the reduced carbohydrate intakes. Figure 7.6 depicts the 3-days computed insulin responses of the CMAC and PSECMAC insulin models for the undereat diet. As shown in Figure 7.6, both the CMAC and PSECMAC insulin models were able to achieve a rather precise fit to the actual (desired) insulin response of Subject A. This is because the undereat diet is metabolically an attenuation of the normal dietary profile (refer to eq. (7.2)). Therefore, the training derived from the normal dietary profile is functionally adequate to *prime* the insulin models with the necessary characteristic mappings or domain knowledge to predict the insulin requirements for the undereat diet. However, the computed insulin responses of the CMAC insulin model were plagued by offset errors (refer to B_1 , B_2 and B_3 of Figure 7.6(a)) that contribute to its poorer performances. In addition, several overshoots were noted in the computed CMAC insulin responses as highlighted by A_1 and A_2 in Figure 7.6(a). These offsets and overshoots in the computed insulin responses may be attributed to the uniform partitioning of the input-output mapping space of the basic CMAC network. The uniform quantization of the CMAC memory space results in a static output resolution throughout the entire CMAC input space, and this often leads to an averaging effect and reduced output precision as manifested in the offsets and overshoots of Figure 7.6(a). In contrast, the PSECMAC network non-uniformly allocates its memory cells according to the characteristics of the training data, where more memory cells (and hence higher output resolutions) are allocated to the significant regions of the input space.

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.6: 3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the undereat diet

This enables PSECMAC to compute an accurate prediction of the insulin responses of Subject A for the undereat dietary profile (as shown by P_1 and P_2 in Figure 7.6(b)).

The overeat and irregular dietary profiles, on the other hand, stimulate the uncertainty associated with the meal consumption habits of Subject A and the inter-day variability of his metabolism process. This sought to investigate the robustness of the PSECMAC and CMAC-based insulin models under demanding conditions. Similar to the normal and undereat profiles, the overeat diet is characterized by four daily meals. The total daily carbohydrate intake when Subject A overeats, however, is greater than his recommended daily allowance (refer to eq. (7.3)). Hence, overeating causes the CMAC and PSECMAC insulin models to be subjected to meal (carbohydrate) intakes that exceed those of the normal diet (from which the models have previously been derived). The overeat diet effectively assesses the capacity of the insulin models in responding to the augmented fluctuations and increased levels of the glucose concentrations that are above the range encountered in the normal profile. On the other hand, the irregular dietary profile is defined based on the day-to-day variation in the amount of carbohydrate consumed together with the sporadic number of meals taken by Subject A in a day. As described in Section 7.3, the number of meals for the irregular dietary profile varies between 2 to 6 meals per day while the amount of daily carbohydrate intakes ranges between half to twice of the RDA for Subject A. Therefore, the irregular diet essentially evaluates the performances of the trained PSECMAC and CMAC insulin models for a challenging set of circumstances that is characterized by both large (rapid) and small (slow) fluctuations in the measured blood glucose concentrations.

Figures 7.7 and 7.8 depict the 3-days generalization results of the CMAC and PSECMAC insulin models for the overeat and irregular dietary profiles respectively. Based on the two figures, several observations can be made of the generalization performances of the two insulin models. Firstly, although the computed insulin responses of both the CMAC and the PSECMAC insulin models were able to closely match the actual (desired) insulin responses of the healthy subject within the range of low to moderate insulin values, both insulin models failed to track the occasional high peaks in the healthy insulin profile arising from the large amount of carbohydrate intakes associated with overeating. The poor generalization performances of the CMAC and PSECMAC insulin models for the prominent peaks of the overeat and irregular diets are thus primarily due to the large undershoot errors, as highlighted by A_1 , A_2 and A_3 in Figure 7.7(a), as well as P_1 , P_2 and P_3 in Figure 7.7(b).

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.7: 3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the overeat dietary profile

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.8: 3-days generalization performances of the CMAC and PSECMAC network in modeling the insulin profile of Subject A for the irregular dietary profile

These undershoot errors are due to the fact that no prior training has been performed to enable the models to derive the required insulin responses for large-sized meal disturbances, as a large amount of carbohydrate intake results in a surge of the blood glucose levels to a range that is beyond the information that has been extracted from the normal diet. Consequently, as the insulin response of a healthy person is dependent on the circulating blood glucose concentration, a high carbohydrate intake will result in a large undershoot error. This is especially evident in the generalization results for the irregular diet depicted in Figure 7.8 (see A_1 and A_2 of Figure 7.8(a) and P_1 and P_2 of Figure 7.8(b)).

Secondly, the lack of model training for the large-sized meal scenarios in the overeat and irregular diets also gives rise to the *empty cells phenomena* that degrades the consistency of the predicted insulin responses of the CMAC insulin model. The empty cells phenomena in a CMAC-based system occurs whenever the input vector accesses the regions of untrained CMAC memory cells and thus resulting in an undesirable system output. In this example, the effects of the untrained CMAC cells are highlighted as C_1 and C_2 in Figures 7.7(a) and 7.8(a). On the other hand, the non-uniform memory quantization scheme of the PSECMAC architecture allows the PSECMAC insulin model to better generalize the characteristics of the glucose–insulin relationships. This enables the PSECMAC insulin model to compute a reasonable prediction of the insulin response even for an input vector that has not been encountered before. Lastly, similar to the evaluation results observed of the undereat diet, the computed CMAC insulin responses suffer from offset errors that further degrade the performances of the CMAC insulin model (see B_1 and B_2 of Figures 7.7(a) and 7.8(a)).

From the results of the previous experiments, one can conclude that the poor performances of the CMAC and PSECMAC insulin models for the overeat and irregular dietary evaluations are predominantly caused by the lack of prior training on large-sized meal disturbances. Therefore, in order to enhance the generalization performances, the CMAC and PSECMAC insulin models are dynamically tuned to adapt their learnt schedules to the overeat and the irregular dietary profiles. For this purpose, a set of 4-days (new) glucose and insulin data is generated for each of the overeat and irregular diets respectively using the GlucoSim simulator for Subject A. The collected data is subsequently employed to tune the trained CMAC and PSECMAC insulin models. Tables 7.6 and 7.7 list the generalization performances of the tuned CMAC and PSECMAC insulin models for the overeat and irregular dietary profiles. The "% Gain" measure denotes the percentage gain in the performances of the PSECMAC insulin model over its CMAC counterpart. The "% Improve" measure,

Table 7.6: Simulation results of the tuned insulin models for the overeat and irregular dietary profiles

Network	Dietary Profile	RMSE	PC	PI ₁	MAE	MSE	PI ₂
PSECMAC	Overeat	9.9580	0.9922	9.05	4.8759	99.162	16.88
	Irregular	19.089	0.9763	4.86	6.9385	364.40	45.90
CMAC	Overeat	22.943	0.9512	3.97	6.8259	526.36	67.26
	Irregular	23.846	0.9621	3.87	11.884	568.63	44.13

Table 7.7: Performance comparisons between the tuned PSECMAC and CMAC insulin models for the various dietary profiles

Dietary Profile	Tuning Phase	PI ₁			PI ₂		
		PSECMAC	CMAC	% Gain	PSECMAC	CMAC	% Gain
Overeat	Before	5.23	3.67	42.51%	45.87	67.74	32.29%
	After	9.05	3.97	127.9%	16.88	67.26	74.90%
	% Improve	73.04%	8.17%		63.20%	0.71%	
Irregular	Before	2.59	2.13	21.60%	143.5	156.1	8.07%
	After	4.86	3.87	25.58%	45.90	44.13	-4.01%
	% Improve	87.64%	81.69%		68.01%	71.73%	

on the other hand, computes the percentage of performance improvement achieved by the tuning process of the respective insulin models. The 3-days predicted insulin responses of the tuned CMAC and PSECMAC insulin models are depicted in Figures 7.9 and 7.10 for the overeat and irregular diets respectively.

From the results tabulated in Tables 7.6 and 7.7, one can observe that the tuning process leads to substantial improvements in the performances of both the CMAC and PSECMAC insulin models. In accordance with the earlier hypothesis, the adaptive tuning mechanism enabled both the CMAC and PSECMAC insulin models to compute more accurate predictions of the insulin responses for the peaks associated with the large carbohydrate intakes present in the overeat and irregular diets. These are highlighted as A_1 , A_2 and P_1 , P_2 of Figures 7.9 and 7.10. In addition, the adaptive tuning has also increased the gain in the performance (% Gain) of the PSECMAC insulin model over its CMAC counterpart for the overeat diet. This is due to the fact that the non-uniform quantization scheme of the PSECMAC network allows for better characterization of the selected glucose variables to insulin responses relationships. As highlighted in B_1 , B_2 and C_1 of Figure 7.9(a), the suboptimal static quantization of the CMAC memory cells has resulted in the offset errors and the occurrence of the empty cell phenomena in the predicted CMAC insulin responses.

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.9: 3-days generalization performances of the CMAC and PSECMAC networks (after-tuning) in modeling the insulin profile of Subject A for the overeat dietary profile

(a) CMAC Insulin Model

(b) PSECMAC Insulin Model

Figure 7.10: 3-days generalization performances of the CMAC and the PSECMAC networks (after-tuning) in modeling the insulin profile of Subject A for the irregular dietary profile

However, with respect to Table 7.6, the generalization performances of the CMAC insulin model for the irregular dietary profile after the tuning process are comparable to those of the PSECMAC model. As depicted in Figure 7.10(b), the PSECMAC insulin model exhibited a considerable undershoot error (highlighted as P_1) despite being able to compute fairly accurate predictions of the insulin responses for most of the sampled points. This is due to the fact that the extra 4-days of new metabolic data might not be sufficient to capture the comprehensive characteristics of the irregular diet. In contrast to the overeat profile that adopts a constant 4-meals-a-day regime, the irregular dietary profile varies the number of meals taken together with an enlarged range for the daily total carbohydrate intakes. Such irregularities (uncertainties) translate to large permutations of eating patterns (i.e. insulin profile) that cannot be comprehensively characterized by only 4-days of metabolic data. Therefore, it is highly plausible that insufficient training (tuning) accounts for the uncharacteristically large undershoot errors observed in the computed PSECMAC insulin responses. On the other hand, the averaging principle in the output computation process of the CMAC network would likely have assisted in improving the overall performance of the CMAC insulin model. However, from Figure 7.10(a), it can be observed that the computed CMAC insulin responses suffer from many offset errors (see B_1 , B_2 , B_3 and B_4) as well as the empty cell phenomena (i.e. C_1). Therefore, by comparing the computed insulin responses of the CMAC (Figure 7.10(a)) and PSECMAC (Figure 7.10(b)) insulin models, it is evident that the PSECMAC insulin model produced more preferable insulin responses due to the general accuracy and consistency of its computed insulin profile. Lastly, the sets of results from the previous experiments have suggested that the cerebellar-based insulin models can be adapted to address the intra- and inter-day variability of the glucose metabolic process of a healthy subject.

7.6 PSECMAC-based Control of Insulin Infusion for Diabetes Treatment

The PSECMAC insulin model developed in Section 7.4 is subsequently applied as a pump controller in a closed-loop glucose-insulin regulatory system to regulate the insulin infusion rate to a simulated Type-1 diabetic patient. Unlike most of the existing closed-loop glucose-insulin regulatory systems reported in the literature, the closed-loop system proposed in this work is based on the Model Reference Adaptive Control (MRAC) paradigm (Kaufman

et al., 1994), where the control objective is to synthesize the healthy insulin responses in the diabetic subject. That is, the proposed PSECMAC-based glucose–insulin regulatory system aims to emulate the physiological process of pancreatic insulin release to drive the glucose profile of the controlled diabetic patient to follow that of a referenced healthy model.

In this study, the GlucoSim simulator for a Type-1 diabetic person is employed as the simulator for the diabetic patient. Similar to the healthy person model, the Type-1 diabetic model of GlucoSim is constructed based on the compartmental modeling of the various organs involved in the human glucose metabolic cycle and their respective interactions. In the Type-1 diabetic model of GlucoSim, however, an intra-peritoneal (IP) insulin injection (to the portal vein) sub-system is included in place of the pancreatic insulin production module of the healthy person model. The compartmental models employed in the GlucoSim simulator are based on the work of (Puckett, 1992) and (Sorensen, 1985). Appendix K of the Thesis presents a brief overview on the GlucoSim simulator. Please refer to (GlucoSim, Online) for the technical details and a web-based version of the GlucoSim simulator.

The work in this section is based on the hypothesis that the rate of insulin production by the pancreatic β -cells in a healthy subject is functionally proportional to the serum glucose concentration measured from time to time. The proposed insulin pump control module for the closed-loop glucose–insulin regulatory system therefore consists of two sub-systems. They are: (1) the PSECMAC insulin response model; and (2) the interface to the GlucoSim Type-1 diabetic simulator. In this setup, the PSECMAC insulin model predicts the required future blood insulin concentration based on the glucose indicator values extracted from the metabolic data of the diabetic patient. The simulator interface, on the other hand, serves to transform the predicted future blood insulin concentration (in microU/ml unit) into the equivalent insulin flow rate (in microU/min unit), and to subsequently scale the resultant flow rate to produce the required insulin injection rate. For this purpose, the calibration of a scaling factor is necessary. Therefore, as an initial stage to the development of the PSECMAC-based insulin regulatory system, a static closed-loop setup is proposed to establish the robustness of the PSECMAC insulin model and the feasibility of the closed-loop paradigm as well as the implemented interface to the GlucoSim simulator. Figure 7.11 depicts the static closed-loop control setup for the PSECMAC-based glucose–insulin regulation system.

In the static control structure of Figure 7.11, a fully diabetic model of Subject A has

Figure 7.11: Static closed-loop control setup for PSECMAC-based glucose-insulin regulation

been assumed (denoted as $FD(A)$). The diabetic patient model of the GlucoSim simulator accepts, as its inputs, the dietary data and the insulin injection rate as well as the previous metabolic state of the diabetic model. As outputs, the GlucoSim diabetic model computes the current insulin and glucose concentrations in the blood as well as the internal states of the various compartments in the model. All the internal variables are in turn used to compute the next state of the diabetic model.

To obtain the appropriate insulin injection rate, the required future insulin response of the diabetic subject $FD(A)$ under a particular dietary profile is first computed using the PSECMAC insulin model $\mathfrak{F}_{\text{normal}}^{H(A)}$ of Section 7.4. The computational output of the PSECMAC insulin model is formalized as eq. (7.12)

$$\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t+1) = \mathfrak{F}_{\text{normal}}^{H(A)} \left(\left\{ z_{\text{mces, normal}}^{FD(A)}(t) \right\} \right) \quad (7.12)$$

where $\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t+1)$ denotes the computed insulin requirement of the fully diabetic patient A (i.e. $FD(A)$) for the dietary profile $x_D \in \{\text{normal}, \text{under}, \text{over}, \text{irregular}\}$ based on the PSECMAC healthy insulin model constructed from the normal diet (i.e. $(H(A), \text{normal})$); $\{z_{\text{mces, normal}}^{FD(A)}(t)\}$ is the set of glucose indicators identified via the MCES feature selection process (see Section 7.4) after being extracted from the diabetic glucose metabolism data for the normal diet at time t ; and $\mathfrak{F}_{\text{normal}}^{H(A)}$ refers to the PSECMAC insulin model constructed for the healthy model of Subject A with a normal diet.

Subsequently, the computed insulin response $\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t+1)$ is applied to the simulator interface (H) to obtain the required insulin injection rate. This is formalized as in

eq. (7.13)

$$\begin{aligned} I_{\text{InjectRate}}(t) &= \mathfrak{H} \left(\hat{I}_{x_D, (H(A), \text{normal})}^{\text{FD}(A)}(t+1) \right) \\ &= h \times \hat{I}_{x_D, (H(A), \text{normal})}^{\text{FD}(A)}(t+1) \times CR \end{aligned} \quad (7.13)$$

where $I_{\text{InjectRate}}(t)$ denotes the required injection rate, $\mathfrak{H}$ denotes the transfer function of the simulator interface module (H), h is the scaling factor, and CR refers to the conversion rate from the predicted insulin concentration (microU/ml) to the insulin flow rate (microU/min). The conversion rate (CR) is computed as (Bischoff, 1986) described in eq. (7.14)

$$CR = (9.9314 \times \text{BodyWeight} + 0.6859) \quad (7.14)$$

where BodyWeight refers to the body weight of the diabetic patient.

A series of control simulations were performed on the normal, under, over and irregular dietary profiles to calibrate the scaling factor h of the proposed PSECMAC-based closed-loop glucose–insulin regulatory system. From the simulation results, a scaling factor of $h = 5$ was empirically determined to give the best control performances for all the dietary profiles. Table 7.8 lists the resultant 4-days control performances of the static closed-loop insulin control system with a scaling factor of $h = 5$ for the various dietary profiles. The control efforts of the static PSECMAC-based insulin regulatory system are benchmarked against those of the healthy model. RMSE denotes the root-mean-squared-error value between the blood glucose level of the controlled diabetic subject ($FD(A)$) and the (desired) blood glucose level of the healthy subject ($H(A)$). The PC value, on the other hand, measures the Pearson correlation coefficient between the blood glucose level of the controlled diabetic and that of the healthy subject. Based on the PC and RMSE values, a performance index PI_1 is computed as in eq. (7.9).

As a baseline comparison to the observed performances of the static PSECMAC-based insulin control system, the set of metabolic simulations for the different diets is repeated but with no insulin infusion. That is, it is assumed that there is no insulin administration to the diabetic patient model. Hence, the diabetic blood glucose fluctuations observed are due to the carbohydrate intakes and the result of the various insulin-independent glucose removal mechanisms. The simulation results for the uncontrolled (no insulin infusion) glucose metabolic process of $FD(A)$ are presented in Table 7.8.

Table 7.8: The control performances of the PSECMAC-based static closed-loop control system for various dietary profiles

Dietary Profile	PSECMAC-Controlled			Uncontrolled		
	RMSE	PC	PI ₁	RMSE	PC	PI ₁
Normal	38.915	0.6231	1.56	259.93	0.1778	0.07
Under	22.050	0.8287	3.60	203.30	0.0957	0.05
Over	64.766	0.4832	0.73	509.19	0.2699	0.05
Irregular	53.199	0.7727	1.43	561.48	0.2665	0.05

As highlighted previously, the objective of the glucose–insulin regulatory system proposed in this work is to synthesize the healthy glucose–insulin metabolic dynamics in a diabetic subject. From the simulation results tabulated in Table 7.8, one can observe that the static PSECMAC-based insulin regulatory system has achieved encouraging glucose control performances. This is shown by the good correlations between the controlled (diabetic) and healthy blood glucose levels. The best control performances were achieved for the under-eat diet with a Pearson correlation of approximately 82.9%, indicating a close fit between the controlled and healthy glucose–insulin dynamics. However, the performances of the static PSECMAC-based insulin regulation system are comparatively degraded for the irregular and over-eat diets. This may be due to the poor accuracy of the PSECMAC insulin model when applied to the irregular and over-eat profiles. The PSECMAC insulin model employed for the simulations was constructed based on the normal dietary data. As reported in Section 7.5, the output accuracy of the PSECMAC insulin model trained on the normal diet tends to degrade when applied to the over-eat and the irregular diets. Moreover, due to the highly non-linear characteristics of the glucose metabolic cycle, a small inaccuracy in the insulin infusion schedule may translate to a large disparity in the observed diabetic blood glucose concentrations (as compared to the healthy glucose response).

The simulation results in Table 7.8, however, have clearly demonstrated the significant impact of the proposed static PSECMAC-based insulin regulatory system in achieving an effective control of the diabetic blood glucose levels. The glucose metabolic profile of the diabetic Subject A, if left uncontrolled, has a RMSE of as high as 561 mg/dl (for the irregular diet) with a maximum correlation of only about 27% to the healthy glucose dynamics. Such a high RMSE value suggests that the patient is experiencing extreme and prolonged episodes of hyperglycemia as illustrated in Figure 7.12.

Figure 7.12 depicts the 3-days blood glucose levels of the diabetic Subject A (controlled and

(a) Normal Diet

(b) Undereat Diet

(c) Overeat Diet

(d) Irregular Diet

Figure 7.12: 3-days control performances of the PSECMAC-based static closed-loop glucose-insulin regulatory system for different dietary profiles

uncontrolled) for the different dietary profiles as benchmarked to those of a healthy person. Further detailed illustrations of the glucose control performances of the proposed static PSECMAC-based insulin regulatory system on the various dietary profiles are provided in Appendix O of the Thesis. Specifically, Figure 7.12 shows that the diabetic patient suffers from multiple severe hyperglycemia episodes for all the evaluated dietary profiles if there is no control of the blood glucose levels. The PSECMAC-based insulin regulatory system, on the other hand, has been shown to effect an adequate control of the diabetic blood glucose levels so that the patient achieved glucose levels that are highly comparable to those of a healthy person. In particular, with the help of the PSECMAC-based insulin regulation system, the diabetic patient managed to achieve healthy blood glucose levels for the undereat and normal diets. Although the controlled blood glucose responses of the diabetic Subject A for the overeat and irregular diets do not follow closely those of a healthy person, the effectiveness of the proposed closed-loop control structure in maintaining the normoglycemia state of the diabetic patient is clearly reflected in the fast control response time in bringing down the elevated blood sugar levels. The experimental results have therefore successfully verified the correctness of the implemented simulator interface and have also sufficiently demonstrated the effectiveness of the proposed PSECMAC-based closed-loop regulatory system.

However, from the results of the previous experiments, it is also apparent that the static control setup is inadequate to effectively manage the dietary variability of the diabetic subject. Therefore, to achieve better and more physiologic-like glucose control by the PSECMAC-based insulin regulatory system and to eventually personalize the PSECMAC insulin schedule to cater for the intra- and inter-day metabolic variability of a diabetic patient, an adaptive closed-loop control structure is proposed to tune the PSECMAC insulin model to dynamically adapt to the various dietary profiles. Such an adaptive control setup will enable an endocrinologist to prime the PSECMAC insulin schedule according to the *population-mean*, and to subsequently adapt the required insulin responses to cater for the specific needs of the individual diabetic patient.

The control setup of the adaptive PSECMAC-based glucose–insulin regulatory system is depicted in Figure 7.13. As shown in Figure 7.13, the adaptive control structure extends from its static counterpart by incorporating a feedback loop that compares the glucose responses of the controlled diabetic subject ($G(t+1)$) against that of a healthy reference model ($G_R(t+1)$), and subsequently uses the computed glucose error ($G_E(t+1)$) to tune

Figure 7.13: Adaptive closed-loop control setup for PSECMAC-based glucose-insulin regulation

the insulin prediction of the PSECMAC insulin model. The objective of the tuning process is essentially to adapt the characteristic surface of the PSECMAC insulin model to match the insulin requirements of the different dietary profiles. This is described in eq. (7.15)

$$\mathfrak{F}_{x_D}^{FD(A)} = \mathfrak{L} \left(\mathfrak{F}_{\text{normal}}^{H(A)} \right) \quad (7.15)$$

where $\mathfrak{F}_{x_D}^{FD(A)}$ denotes the (desired) insulin model of a fully diabetic subject A for the dietary profile x_D where $x_D \in \{\text{normal, under, over, irregular}\}$, $\mathfrak{F}_{\text{normal}}^{H(A)}$ refers to the PSECMAC insulin model constructed based on the metabolic data of the healthy Subject A with a normal diet, and $\mathfrak{L}$ is the learning algorithm used to evolve the PSECMAC insulin model $\mathfrak{F}_{\text{normal}}^{H(A)}$ to approximate $\mathfrak{F}_{x_D}^{FD(A)}$.

The computational process of the adaptive PSECMAC-based insulin regulatory system is summarized as follows:

1. The PSECMAC insulin model uses the glucose derivatives $\{z_{\text{mces}, x_D}^{FD(A)}(t)\}$ extracted from the measured glucose data $G(t)$ of the diabetic patient $FD(A)$ to compute the expected future insulin response $\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t+1)$.
2. The insulin response $\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t+1)$ is used by the simulator interface to compute the current injection rate $I_{\text{InjectRate}}(t)$ to be applied to the diabetic patient based on eq. (7.13).

3. The computed insulin injection rate is applied to the diabetic patient, and the resultant blood glucose level $G(t+1)$ is obtained based on the current metabolic state of the patient.
4. The measured blood glucose level $G(t+1)$ is compared against the referenced healthy blood glucose level $G_R(t+1)$ to obtain the glucose error $G_E(t+1)$ as described in eq. (7.16).

$$G_E(t+1) = G_R(t+1) - G(t+1) \quad (7.16)$$

5. If there is an error in the diabetic blood glucose level, i.e. $G_E(t+1) \neq 0$, the control system assumes that an insulin prediction error has occurred.
6. The control system would update the PSECMAC insulin model according to the following learning rules:
 - (a) Evaluate if the amount of insulin administered is too much or too little using eq. (7.17).

$$\text{InsulinAdministered} = \begin{cases} \text{Too much,} & \text{if } G_E(t+1) > 0 \\ \text{Too little,} & \text{if } G_E(t+1) < 0 \end{cases} \quad (7.17)$$

- (b) If the amount of insulin administered is too much, update the PSECMAC insulin model to reduce the expected insulin schedule of the diabetic patient for the glucose derivatives $\{z_{\text{mces}, x_D}^{\text{FD(A)}}(t)\}$ using eq. (7.18)

$$\hat{I}_{x_D, (H(A), \text{normal})}^{\text{FD(A), new}}(t+1) = (1 - \alpha) \times \hat{I}_{x_D, (H(A), \text{normal})}^{\text{FD(A)}}(t) \quad (7.18)$$

where α denotes the update factor so that the new rate of insulin injection $I_{\text{InjectRate}}^{\text{new}}(t)$ associated with the glucose indicators $\{z_{\text{mces}, x_D}^{\text{FD(A)}}(t)\}$ is reduced. Note that the update factor α is also the learning constant of the PSECMAC network. Hence, the update equation listed in eq. (7.18) is mathematically equivalent to the learning equation of PSECMAC (see eqs. (4.16) of Chapter 4). Following this, the selection of the update factor α is bounded by the theoretical range established for the learning convergence of the PSECMAC network in Chapter 4.

Table 7.9: The control performances of the PSECMAC-based adaptive closed-loop control system for the various dietary profiles

Dietary Profile	Adaptive			Static		
	RMSE	PC	PI ₁	RMSE	PC	PI ₁
Normal	33.012	0.7390	2.17	38.915	0.6231	1.56
Under	16.976	0.9074	5.05	22.050	0.8287	3.60
Over	64.005	0.5060	0.78	64.766	0.4832	0.73
Irregular	52.275	0.7721	1.45	53.199	0.7727	1.43

Table 7.10: Performance comparisons between the static and the adaptive PSECMAC insulin model for the various dietary profiles

Dietary Profile	PI ₁		% Improve
	Static	Adaptive	
Normal	1.56	2.17	39.10 %
Under	3.60	5.05	40.28 %
Over	0.73	0.78	6.85 %
Irregular	1.43	1.45	1.40 %

- (c) Else, if the amount of insulin administered is too little, update the PSECMAC insulin model to increase the expected insulin schedule of the diabetic patient for the glucose indicators $\{z_{mces, x_D}^{FD(A)}(t)\}$ using eq. (7.19):

$$\hat{I}_{x_D, (H(A), \text{normal})}^{FD(A), \text{new}}(t) = (1 + \alpha) \times \hat{I}_{x_D, (H(A), \text{normal})}^{FD(A)}(t + 1) \quad (7.19)$$

such that the new rate of insulin injection $I_{\text{InjectRate}}^{\text{new}}(t)$ is increased.

The control simulations for the different dietary profiles are repeated to assess the effects of the insulin schedule tuning process on the performance of the adaptive PSECMAC-based insulin regulatory system. Tables 7.9 and 7.10 tabulate the performance comparisons between the static and the adaptive PSECMAC-based glucose–insulin regulatory system for the various dietary profiles. As shown in Table 7.9, the use of an insulin schedule tuning algorithm enhances the glucose control performances of the adaptive PSECMAC-based insulin regulatory system in comparison to its static counterpart for all the evaluated dietary profiles. The adaptive PSECMAC-based glucose–insulin regulatory system produced a Pearson coefficient of approximately 90% between the controlled and the (desired) healthy blood glucose dynamics for the undereat diet. This subsequently leads to a significant increment of 40.28% in the PI₁ value as listed in Table 7.10. An improvement of about 39% was also achieved for the diabetic blood glucose control using the normal dietary profile, and this further demonstrated the effectiveness of the proposed adaptive tuning process.

However, from Table 7.10, one can also observe that the performance improvements for the overeat and the irregular dietary profiles were not as significant as those achieved for the undereat and normal diets. This is understandable given the substantial differences in the meal characteristics of the overeat and irregular dietary profiles as compared to those of the normal diet for which the PSECMAC insulin model has been previously trained for. Therefore, a longer tuning period may be required by the PSECMAC insulin model to adapt its characteristic surface to learn the insulin requirements of the overeat and the irregular diets. Nevertheless, the simulation results tabulated in Tables 7.9 and 7.10 have sufficiently demonstrated the effectiveness of the adaptive PSECMAC-based insulin regulatory system in managing the elevated blood glucose levels in a diabetic subject. This is illustrated by the plots of the 3-days diabetic blood glucose control performances by the adaptive PSECMAC-based closed-loop insulin regulatory system for the various dietary profiles as shown in Figure 7.14.

7.7 Summary

Diabetes is a disease in which the blood glucose levels in the body are no longer properly regulated. Patients with Type-I diabetes experience prolonged periods of hyperglycemia and hypoglycemia, which may lead to both chronic and acute health problems. Many of the diabetes-related complications, fortunately, can be prevented through the tight control of the blood/plasma glucose levels within the ideal homeostatic range of 60-110 mg/dl. For this purpose, however, constant medical monitoring and long-term pharmacological insulin treatment are necessary.

The key to a successful management of diabetes is to maintain long term near-normoglycemia of the diabetic patient. The current protocol and standard of diabetes treatment is not adequate chiefly due to the following reasons: (1) the de-facto therapy of daily discrete insulin injections for blood glucose control encountered suboptimal therapeutic outcomes as it is difficult to achieve near-normal blood glucose levels via an open-loop control; (2) a majority of the existing closed-loop insulin regulatory systems rely on static mathematical models of the human glucose metabolic process that are often ill-equipped to address the metabolic bio-diversity of the diabetic patients; and (3) the currently adopted diabetes treatment regimes do not account for the intra- and inter-day variability in the glucose metabolic

(a) Normal Diet

(b) Undereat Diet

(c) Overeat Diet

(d) Irregular Diet

Figure 7.14: 3-days diabetic blood glucose control performances by the PSECMAC-based adaptive closed-loop glucose–insulin regulatory system for the different dietary profiles

process of the individual patient.

In this chapter, a brain-inspired closed-loop approach to the personalized management of Type-1 diabetes is presented. In the experiments, the healthy and diabetic person model of the GlucoSim simulator are employed to generate the required healthy and diabetic metabolic data. Subsequently, four types of dietary profiles (i.e. normal, under, over and irregular) are created to simulate the intra- and inter-day variability in the meal intakes of the healthy and diabetic subjects. The objective of the proposed closed-loop glucose–insulin regulatory system is to emulate the physiological process of pancreatic insulin secretion in a healthy person. For this purpose, the PSECMAC network of Chapter 4 is employed to model the insulin profile of a healthy subject based on the normal diet. The resultant PSECMAC insulin model is subsequently evaluated for the different dietary profiles to investigate the modeling performances of the proposed insulin model. It has been demonstrated that the PSECMAC insulin model can be adapted to suit the intra- and inter-day variability in the glucose metabolic process of a healthy subject. Finally, the PSECMAC insulin model is incorporated in a closed-loop setup for insulin infusion regulation in a diabetic patient. The simulation results have shown that the PSECMAC-based insulin regulatory system holds a potential promise for the personalized treatment of diabetes.

The next chapter constitutes Part 6 of the Thesis that concludes the research presented in all the previous chapters and outlines the directions for future investigations.

*"What we call the beginning is often the end.
 And to make an end is to make a beginning.
 The end is where we start from"*
 –T.S. Eliot

Chapter 8

Conclusions and Future Research

The human cerebellum is a major brain construct and it is responsible for the regulation of bodily movements and procedural skill learning. As a movement calibrator and mediator of skill acquisition, the human cerebellum possesses the capability to model precisely highly complex and nonlinear physical dynamics (Imamizu et al., 2000; Kawato, 1999; Kawato and Gomi, 1992; Shidara et al., 1993). The *Cerebellar Model Articulation Controller* (CMAC) network is a computational model of the human cerebellum developed to emulate its nonlinear function learning ability. Despite its architectural simplicity, the rigidity of the CMAC uniform computing structure gives rise to three major operational deficiencies (see Chapter 1), namely: (1) suboptimal function modeling accuracy; (2) poor memory utilization; and (3) the generalization–accuracy dilemma. In addition, the performance of the CMAC network is also hampered by several drawbacks induced by its localized learning principle. These drawbacks are: (1) insufficient training; (2) exponential increase in the CMAC network training time with increased network size; and (3) the modeling generalization–accuracy dilemma. Despite a number of previous attempts reported in the literature that tried to resolve the computational problems associated with the uniform quantization scheme of the CMAC network, there has been limited success in effectively addressing the architectural shortcomings and training deficiencies in a CMAC-based system.

This Thesis focuses on the development of a novel neurophysiologically inspired self-organizing cerebellar associative memory framework to address the computational limitations observed in the CMAC network. Four primary approaches are presented in the proposed cerebellar framework (see Figure 1.1 of Chapter 1); namely: (1) *adaptive memory quantization*, (2) *neighborhood-based training*, (3) *memory patching*, and (4) *memory upsizing*. These efforts

culminated to the development of two new cerebellar computing architectures and a three-stage cerebellar learning paradigm. Specifically, the two proposed cerebellar-based computational architectures employ an adaptive memory allocation scheme to perform non-uniform quantization of the computing (memory) cells. This approach resolves the computing deficiencies due to the uniform partitioning of the memory cells in a CMAC network. The use of an adaptive memory allocation scheme improves the memory utilization rate, enhances the modeling accuracy and at the same time partially addresses the generalization–accuracy dilemma in the proposed cerebellar architectures. This is achieved by varying the quantization resolution across the input–output associative mapping space according to the information characteristics of the training data. Subsequently, a neighborhood-based training algorithm is proposed under the framework to implement a distributed learning paradigm in the new cerebellar memory models to further address the generalization–accuracy dilemma commonly observed in the CMAC network.

Meanwhile, the deficiencies related to the training of the CMAC network are resolved by the cerebellar memory patching and upsizing algorithms proposed in the Thesis. Memory patching refers to the construction of a plausible memory surface for the untrained CMAC memory cells to address the insufficient training problem of the CMAC network. The memory upsizing algorithm, on the other hand, reduces the time required to train a CMAC network by performing a two-stage training process; that is, a small-sized CMAC network is first trained to derive a generalized characterization of the training data before scaling up the trained network to achieve a finer modeling resolution. Both the memory patching and upsizing algorithms are subsequently incorporated into a computational technique named the Three-Stage-Learning CMAC (TSL-CMAC) paradigm that constitutes a neurologically-inspired approach to the effective training of a CMAC network.

The proposed self-organizing cerebellar memory framework and its two new computing architectures are subsequently applied to a diverse range of interesting applications that comprises of modeling, control, and pattern analysis/classification problems. The following section summarizes the contributions achieved by the research presented in the Thesis.

8.1 Major Contributions of The Thesis

Nine major contributions have been made with respect to the research objectives defined in

Chapter 1. These achievements can be categorized as theoretical and empirical research contributions and they are summarized as follows.

8.1.1 Theoretical Research Contributions

- A new cerebellar-based computational architecture named the HCAQ-CMAC network (Teddy and Lai, 2005; Teddy et al., 2007a) that employs a hierarchical clustering technique to non-uniformly allocate its computing (memory) cells throughout the input-output associative space for function approximation is proposed in Chapter 3. This memory/computing cell allocation process is inspired by the two-stage physiological development process observed in the human brain, where the excessive neurons and synaptic connections created during the formation of the infant brain are gradually pruned and refined to form the precise wirings of the developed adult brain (Jessell, 1996; Shaw et al., 2006). In the HCAQ-CMAC network, an oversized network structure is created initially. This excessive computing structure is gradually pruned to obtain a parsimonious architecture based on the output variations of the training data. This subsequently results in a finer quantization resolution at the input regions that contains more heterogeneous outputs and enhances the memory utilization of the HCAQ-CMAC network as well as improving the generalization and accuracy of the computed output. In addition, the HCAQ-CMAC learning process has been theoretically proven to be functionally stable and convergences on the proper setting of the learning rate (please refer to Chapter 3 for details).
- Chapter 4 presents the novel PSECMAC network (Teddy et al., 2006b, 2007c), which is a self-organizing multi-resolution cerebellar-based associative memory computing architecture. The proposed PSECMAC associative memory network is inspired by the neurophysiological studies on the cerebellar learning process, where it has been shown that significantly higher densities of the cerebellar synaptic connections are located at the frequently-accessed regions of the cerebellum that are activated by repeated learning episodes (i.e. experience-dependent adaptation) (Federmeier et al., 2002; Kleim et al., 1996, 1998a,b). This cerebellar-based experience-driven synaptic plasticity phenomenon is emulated in the PSECMAC network by employing a data-driven adaptive memory quantization scheme (i.e. MPSEC) for the derivation of its computing structure. In the PSECMAC network, the resolution of the computing

cells is adapted based on the computed information distribution of the training data, where more memory cells are allocated to the densely data-populated regions of the input–output associative space to enhance memory efficiency. In addition, the learning process of the PSECMAC network has also been theoretically proven to mathematically converge on the proper setting of the learning rate.

- A Three-Stage CMAC Learning (TSL-CMAC) (Teddy et al., 2006a, 2007d) paradigm is proposed (see Chapter 5) to address the computational issues associated with the training of a CMAC network, namely: (1) the long training time correlated to a large network size, (2) the undesirable system performances due to partial or insufficient training; and (3) the generalization-accuracy dilemma encountered by the CMAC network. In the proposed TSL-CMAC training paradigm, the memory patching technique is employed to construct a plausible memory surface for the untrained CMAC memory cells. Subsequently, the memory upsizing algorithm is deployed to reduce the CMAC training time as well as to address the generalization–accuracy dilemma encountered during the training of a CMAC network. The three training stages in the proposed TSL-CMAC process (i.e. *initial learning*, *network generalization* and *fine-tuning*) are based on a well-established three-stage psychological learning model for the acquisition of human motor and cognitive skills (Fitts and Posner, 1967; Tomporowski, 2003).

8.1.2 Empirical Research Contributions

- The proposed PSECMAC and HCAQ-CMAC networks have been applied to the modeling of the non-linear dynamics of the human glucose metabolic process (Teddy et al., 2005, 2006a, 2007a,f). In Chapter 3, the HCAQ-CMAC network is used to model the dynamics of the blood glucose cycle in a simulated healthy subject when perturbed by food intakes, where it is hypothesized that the blood glucose level of a healthy person at any given time is a non-linear function of prior food (carbohydrate) intakes and the historical traces of the measured insulin and blood glucose levels. Simulation with the HCAQ-CMAC network has achieved 97% correlation between the computed glucose output and the actual (measured) glucose level of the healthy person.

Meanwhile, the PSECMAC cerebellar network is employed in Chapter 7 to model the plasma insulin response of a healthy subject to food intakes. Unlike the study in

Chapter 3, the inputs to the PSECMAC insulin model consisted of only the current and past plasma glucose information. This approach eliminates the need to estimate the carbohydrate content of each meal consumed as it is difficult and often too cumbersome to be precisely measured. A correlation performance of 99% is achieved by the PSECMAC healthy insulin response model to the actual insulin levels observed in the simulated healthy subject. In addition, Chapter 7 also investigates the application of the proposed insulin modeling approach to different dietary profiles, i.e. the normal, undereat, overeat and irregular diets. Experimental results for the respective insulin models based on the different diets are encouraging as correlations of over 90% are achieved.

- A cerebellar-based approach to non-parametric option pricing and a novel mis-priced option arbitrage trading system are proposed in Chapter 4. In this application, the proposed PSECMAC network is used to predict the theoretical fair valuations of the call options on the GBP vs. USD currency futures (Teddy et al., 2006b, 2007b). The inputs to the PSECMAC option pricing model consisted of the moneyness and time to maturity of the options, and the historical price volatility of the underlying currency futures. The PSECMAC-based option pricing system thus constitutes a cerebellar-inspired local learning approach to the approximation of the associative characteristics between the call option price and its influencing factors. The associative structure of the PSECMAC network also enables discrete pricing rules to be extracted from the pricing system. Subsequently, the PSECMAC-based option pricing model is employed in a mis-priced option arbitrage trading system. Simulation results on the various options with different strike prices demonstrated that the PSECMAC-based mis-priced arbitrage trading system is able to construct risk-free investment portfolios with an encouraging rate of return on investment.
- The application of the HCAQ-CMAC for automatic control of car maneuver (Teddy and Lai, 2005; Teddy et al., 2007a) is investigated in Chapter 3. Vehicle-driving comprises of finely-tuned sets of sensory feedback to control action mappings that are accumulated through experiences and repeated practices. In this study, the HCAQ-CMAC network is employed to model the human decision-making process that relates the sensory (distance) inputs to the steering control output during a car driving episode. The performances of the HCAQ-CMAC driver model has demonstrated the applicability of the cerebellar-based architecture for autonomous car control while the

cell occupancy rate (COR) measure reinforced the importance of adaptive memory allocation in enhancing effective memory utilization in a cerebellar memory model.

- Chapter 7 described a newly-developed cerebellar-based approach to the personalized treatment of diabetes (Teddy et al., 2007g). Specifically, the PSECMAC network is used to implement a computational β -cell for the automated closed-loop regulation of blood glucose level in a simulated Type-1 diabetic patient. The objective of the proposed closed-loop glucose–insulin regulatory system is to emulate the physiological process of pancreatic insulin secretion in a healthy person. Unlike the works previously reported in the literature, the proposed cerebellar-based computational β -cell model does not assume prior knowledge of the timings and carbohydrate contents of the various meals consumed by the diabetic patient. This potentially enables the proposed PSECMAC-based glucose regulatory system to be applied to general populations of diabetic patients with diverse insulin needs.
- The proposed HCAQ-CMAC and PSECMAC networks are also employed to implement a bank failure classifier (Teddy et al., 2007c) to identify the characteristics of financial distress that led to the failures of US banks (see Chapter 4). The financial health of the banks is characterized by nine financial covariates (listed in Appendix F) extracted from the financial statements of the banks. Experimental results have shown that the PSECMAC banking failure classifier achieved a significantly lower equal error rate (EER) value of approximately 6% (as compared to the other benchmarked systems) in discerning between the surviving and failed banks based on the reported financial covariates.
- Finally, a text-dependent speaker verification system (Teddy et al., 2007e) is proposed in Chapter 6, where the novel PSECMAC network is employed as the speaker reference model used to capture the speaker-specific voice characteristics extracted from sampled speeches to perform person verification. The speaker-specific characteristics of the human voice used in the application consisted of the first six *mel-frequency cepstrum coefficient* (MFCC) values of a sampled speech segment. The experimental results obtained from the verification of ten recruited adult speakers have sufficiently demonstrated the superior accuracy of the proposed PSECMAC based speaker verification system to the various benchmarked models.

8.2 Constraints of the Reported Work

As with all other research endeavors, there are certain limitations to the proposed self-organizing cerebellar memory framework as well as the novel HCAQ-CMAC and PSECMAC networks and the three-stage TSL-CMAC learning paradigm. Below is a brief discussion on such limitations.

- Although the novel HCAQ-CMAC and PSECMAC networks have successfully addressed the architectural deficiencies of the basic CMAC network (as evidenced by the significant improvements in their cell occupancy rates), the proposed adaptive memory allocation schemes of the two cerebellar models are at present targeted towards achieving optimal performances with the predefined number of available memory cells. Currently, the network (memory) sizes employed in the respective applications have all been empirically determined. The proposed adaptive quantization techniques do not possess the ability to automatically determine the optimal number of memory cells for the two cerebellar models. This is computationally not ideal because depending on the data characteristics, different applications require different network (memory) sizes to achieve an optimal performance. Consequently, the adjustable parameters of the two proposed memory allocation processes have to be empirically determined. In addition, due to the computing nature of the memory allocation schemes employed in the proposed HCAQ-CMAC and PSECMAC networks, these cerebellar models are currently non-evolving models that are more suitable for offline training. To address the above-mentioned limitations, a new evolving adaptive memory allocation scheme that supports online training is required.
- Currently, the adaptive memory allocation processes in the proposed HCAQ-CMAC and PSECMAC networks allocate the available cerebellar computing (memory) cells separately for each input dimension. This is designed to reduce the computational complexity arising from the introduction of adaptive memory quantization, and it has been observed to work reasonably well for low dimensional problems but may result in undesirable memory wastage for higher dimensional problems.
- HCAQ-CMAC employs a hierarchical clustering technique to allocate its memory cells based on the variation of the target output. Due to this operational principle, the resultant HCAQ-CMAC architecture is a Multi-Input-Single-Output (MISO) system.

The use of the hierarchical clustering technique in the memory quantization process also renders the HCAQ-CMAC network less suitable for classification tasks.

- The proposed TSL-CMAC framework adopts a straightforward definition/identification of the trained and untrained CMAC cells. A memory cell is considered as a trained cell as long as it has been activated by one or more training sample. Otherwise, it is an untrained cell. Currently, the TSL-CMAC framework does not take into account the extent of training (i.e. for partially-trained cells) that a memory cell has received.
- Brief outlines of the estimated worst-case complexity of the various proposed algorithms have been reported in the Thesis and are summarized in Table 8.1. However, the detailed analysis on the computational complexity and scalability of the proposed models remained to be investigated. This effort, however, requires an in-depth study and forms part of the future research endeavors.

8.3 Future Research Directions

This section presents several possible directions for future research classified under two main categories: (a) the extensions to the proposed cerebellar memory framework, and (b) the application domains for the HCAQ-CMAC and PSECMAC networks.

8.3.1 Extensions to the Cerebellar Memory Framework

The proposed architectural extensions involve the development of algorithms and techniques to address the limitations described in the previous section. One promising research direction is to extend the proposed cerebellar models (i.e HCAQ-CMAC and PSECMAC) into a dynamically evolving cerebellar memory model that supports online training. Such an evolving cerebellar architecture will eliminate the need to empirically determine the required number of memory cells and has a great potential for real-time and dynamic (time-varying) applications. For this purpose, an online adaptive memory allocation technique to dynamically quantize the cerebellar memory cells according to the moment-to-moment characteristics of the training data is required. In addition, a novel algorithm for the structural reorganization of the cerebellar architecture during its training process need to be devised.

Table 8.1: Estimated worst-case computational complexity of the various algorithms presented in the Thesis

Algorithm	Complexity	Legends	Page #
HCAQ-CMAC Structural Learning Process	$O(JS^2) + O(\hat{M}^J)$	J is the total number of HCAQ-CMAC input dimensions, $\hat{M}$ is the memory size per dimension and S is the number of training samples.	44
HCAQ-CMAC Computational Process	$O(J\hat{M}) + O((N\hat{M})^J)$	J is the total number of HCAQ-CMAC input dimensions, $\hat{M}$ is the memory size per dimension and N is the neighborhood size.	48
PSECMAC Structural Learning Process	$O(Jn_{c,j}^{(-1)}) + O(JS\tau_{\max}) + O(J(\hat{M})n_{c,j}^{\text{end}}) + O(\hat{M}^J)$	J is the total number of PSECMAC input dimensions, $\hat{M}$ is the memory size per dimension, S is the number of training samples, $\tau_{\max}$ is the maximum number of LVQ iteration, $n_{c,j}^{(-1)}$ is the initial number of density clusters, and $n_{c,j}^{\text{end}}$ is the final number of PSECMAC density clusters.	82
PSECMAC Computational Process	$O(J\hat{M}) + O((N\hat{M})^J)$	J is the total number of PSECMAC input dimensions, $\hat{M}$ is the memory size per dimension and N is the neighborhood size.	86
C-Patch	$O(\hat{M}^{2J})$	J is the total number of CMAC input dimensions and $\hat{M}$ is the memory size per dimension.	132
C-Upsize	$O(\text{NewSize}^J) + O((N \cdot \text{OldSize} \cdot \text{NewSize})^J)$	J is the total number of CMAC input dimensions, OldSize is the old CMAC size, NewSize is the new CMAC size and N is the neighborhood size.	141
MPSEC	$O(n_{c,j}^{(-1)}) + O(S\tau_{\max})$	S is the total number of training samples, $\tau_{\max}$ is the maximum number of LVQ iteration and $n_{c,j}^{(-1)}$ is the initial number of MPSEC density clusters.	275

Another possible future extension is the fuzzification of the proposed cerebellar models. This process involves the fuzzification of the receptive fields of the cerebellar computing cells and the subsequent mapping of a fuzzy inference scheme to define the computation of the model output. A novel fuzzy clustering technique to effectively and efficiently define the fuzzy receptive fields is therefore of paramount importance in the construction of such a fuzzified cerebellar model. The fuzzification of the two proposed cerebellar models offers an improved interpretability of the computation process of the networks and allows for the extraction of a set of human-comprehensible fuzzy rules to explain the computed outputs.

Finally, two possible enhancements to the proposed TSL-CMAC paradigm are currently being investigated. The first enhancement is to identify the partially-trained cerebellar computing cells and to subsequently design a patching technique to address (i.e. populate)

these partially-trained computing (memory) cells. Currently, the proposed TSL-CMAC framework simply assumes a trained status for all the memory cells that have been updated at least once by the training samples, and the C-Patch algorithm employed in the TSL-CMAC learning framework focuses only on the construction of the memory contents of the untrained CMAC cells. Thus, memory patching the partially-trained cells may potentially improve the performance of the resultant TSL-CMAC-trained network. Secondly, the proposed C-Patch and C-Upsize algorithms adopt a simple linear interpolation during the memory construction of the patched and upsized network cells. The implementation of more sophisticated interpolation functions are presently being investigated to enhance the modeling accuracy of the constructed memory cells.

8.3.2 Application Areas of the Cerebellar Memory Models

Some of the future application domains and extensions to the existing application studies of the proposed cerebellar models are briefly described as follows.

Computational Finance

In the option pricing and arbitrage trading application presented in Chapter 4, the effects of market frictions such as transaction costs, taxes, short-selling and capital borrowing costs and the time value of money were not considered. Future enhancements to the development of the proposed option pricing and arbitrage trading system will therefore include the incorporation of transaction costs to the computation of the trading decisions and profits. Furthermore, in order to facilitate a more realistic analysis of the option market and the different trading scenarios, the daily closing price data employed in the current study will be substituted with real-time (i.e. per second or per minute) option price data. Other interesting financial applications that the proposed cerebellar memory framework can be applied to include credit risk assessments, automated stock trading and portfolio management.

Signal Processing

As future work, the proposed speaker verification system described in Chapter 6 can be extended to address the more complex problem of text independent speaker verification

and identification. In addition, the current cerebellar-based speech analysis and modeling technique developed for the application can also be employed for emotion modeling and classification via the study of prosody of the human speech. The research work in this direction can subsequently be combined and packaged together for various interesting applications such as an intelligent tutoring system, which monitors the attention and interest levels of a student (via speech and emotion recognitions) to appropriately generate the required teaching materials to maximize the learning capacity of the student. The proposed cerebellar memory framework can also be applied to the characterization and classification of brain signals (obtained from EEG) to assist in the investigation of the various brain functions.

Medical/Clinical Research

In Chapter 7, a closed-loop regulation of insulin infusion based on the proposed cerebellar architectures for diabetes treatment is presented. Specifically, the PSECMAC network is employed to model the healthy human insulin response to food intakes and to subsequently replicate this insulin response in a Type-1 diabetic patient. However, during the construction of the PSECMAC insulin model, an exhaustive set of glucose-related features that include the different glucose derivatives and their historic traces are defined as the proxies to the actual insulin dynamics. Subsequently, a feature selection process is performed to extract a reduced set of features that presumably best characterizes the insulin responses of the healthy reference subject. An alternative approach to better characterize the insulin dynamics via the glucose proxies is to perform an embedded dimension analysis in addition to feature selection to determine the optimal historical inputs and delays to the PSECMAC insulin model. In addition, the proposed approach for the personalized treatment of diabetes is based on the experimental studies of computationally simulated healthy and diabetic subjects. The research team is currently in close collaboration with a local hospital to undertake a human pilot study to collect actual human metabolic data. The team also intends to extend the research to the management of Type-2 diabetes. Finally, the proposed cerebellar models may also be employed as an intelligent medical diagnostic system to facilitate the analysis of large clinical datasets.

– End of Thesis –

Bibliography

- Y. S. Abu-Mostafa and J-M. St. Jaques. Information capacity of the hopfield model. *IEEE Transactions on Information Theory*, IT-31(4):461–464, 1985.
- D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzman machines. *Cognitive Sciences*, 9:147–169, 1985.
- Y. Ait-Sahalia and A. W. Lo. Nonparametric estimation of state-price densities implicit in financial asset price. *LFE-1024-95, MIT-Sloan School of Management*, 1995.
- A. M. Albisser. Intelligent instrumentation in diabetic management. *CRC Critical Reviews in Biomedical Engineering*, 17:1–24, 1989.
- A. M. Albisser, R. H. Harris, S. Sakkal, I. D. Parson, and S. C. E. Chao. Diabetes intervention in the information age. *Medical Informatics*, 21:297–316, 1996.
- J. S. Albus. A theory of cerebellar function. *Mathematical Biosciences*, 10(1):25–61, 1971.
- J. S. Albus. *Theoretical and experimental aspects of a cerebellar model*. PhD thesis, University of Maryland, College Park, 1972.
- J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control. Transactions ASME*, pages 220–227, 1975a.
- J. S. Albus. Data storage in cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control. Transactions ASME*, pages 228–233, 1975b.
- J. S. Albus. *Brains, Behavior and Robotics*. BYTE Books, McGraw-Hill, 1981.
- J. S. Albus. Marr and Albus theories of the cerebellum: Two early models of associative memory. *Proceedings of IEEE COMPCON*, pages 577–582, 1989.
- P. S. Aleksic and A. K. Katsaggelos. Audio-visual biometrics. *Proceedings of the IEEE*, 94(11):2025–2044, 2006.
- S. Amari. Part 1: Tutorial series on brain-inspired computing. *New Generation Computing*, 23(4):357–359, 2005.
- S. Amari and N. Kasabov. *Brain-like Computing and Intelligent Information Systems*. Singapore; New York: Springer, 1998.

- American Diabetes Association. Economic costs of diabetes in the U.S. in 2002. *Diabetes Care*, 269(3):917–932, 2003.
- H. Amilon. A neural network versus black-scholes: A comparison of pricing and hedging performances. *Scandinavian Working Papers in Economics, Lund University series, Department of economics, Lund, Sweden*, 2001.
- U. Anders, O. Korn, and C. Schmitt. Improving the pricing of options - a neural network approach. *Journal of Forecasting*, 17(5–6):369–388, 1998.
- J. A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14:197–220, 1972.
- K. K. Ang and C. Quek. RSPOP: Rough set-based pseudo outer-product fuzzy rule identification algorithm. *Neural Computation*, 17(1):205–243, 2005a.
- K. K. Ang, C. Quek, and A. Wahab. MCMAC-CVT: a novel on-line associative memory based CVT transmission control system. *Neural Networks*, 15(2):219–236, 2002.
- K.K. Ang and C. Quek. Stock trading using PSEC and RSPOP: A novel evolving rough set-based neuro-fuzzy approach. *Proceedings of IEEE Congress on Evolutionary Computation*, 2:1032–1039, 2005b.
- W. B. Arthur. Inductive reasoning and bounded rationality. *American Economic Association Papers and Proceedings*, 84:406–411, 1994.
- F. M. Ashcroft and S. J. H. Ashcroft. *Insulin: Molecular Biology to Pathology*. Oxford University Press, New York, 1992.
- B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.
- B. S. Atal. Automatic recognition of speakers from their voices. *Proceedings of the IEEE*, 64:460–475, 1976.
- H. E. Atallah, M. J. Frank, and R. C. O'Reilly. Hippocampus, cortex and basal ganglia: Insights from computational models of complementary learning systems. *Neurobiology of Learning and Memory*, 82(3):253–267, 2004.
- Atkins. Atkins carb gram counter. <http://atkins.com>, Online.
- J. B. Attilli, M. Savic, and J. P. Campbell Jr. A tms32020-based real time, text-independent automatic speaker verification system. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 599–602, 1988.
- P. Bajcsy and N. Ahuja. Location- and density-based hierarchical clustering using similarity analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):1011–1015, 1998.
- A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition. Tr-98-038, International Computer Science Institute, Berkeley, California, USA, 1998.

- A. G. Barto, J. T. Buckingham, and J. C. Houk. A predictive switching model of cerebellar movement control. *Neural Information Processing Systems*, 8:138–144, 1995.
- A. G. Barto, A. H. Fagg, N. Sitkoff, and J. C. Houk. A cerebellar model of timing and prediction in the control of reaching. *Neural Computation*, 11:565–594, 1999.
- J. Basak and R. Krishnapuram. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):121–132, 2005.
- R. Bellazzi, G. Nucci, and C. Cobelli. The subcutaneous route to insulin-dependent diabetes therapy. *IEEE Engineering in Medicine and Biology*, 20(1):54–64, 2001.
- G. Bellomo, P. Brunetti, G. Calabrese, D. Mazzotti, E. Sarti, and A. Vincenzi. Optimal feedback glycaemia regulation in diabetics. *Medical & Biological Engineering and Computing*, 20:329–335, 1982.
- Y. Bennani and P. Gallinari. A connectionist approach for speaker identification. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 265–268, 1990.
- Y. Bennani and P. Gallinari. On the use of tdnn-extracted features information in talker identification. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 385–388, 1991.
- B.W. Bequette. A critical assessment of algorithms and challenges in the development of a closed-loop artificial pancreas. *Diabetes Technology and Therapeutics*, 7:28–47, 2005.
- M. Berger, P. Berchtold, H. J. Cuppers, H. Drost, H. K. Kley, W. A. Muller, W. Wiegmann, H. Zimmermann-Telschow, F. A. Gries, H. L. Kruskemper, and H. Zimmermann. Metabolic and hormonal effects of muscular exercise in juvenile type diabetics. *Diabetologia*, 13:355–365, 1977.
- R. N. Bergman and R. E. Miller. Direct enhancement of insulin secretion by vagal stimulation of the isolated pancreas. *American Journal of Physiology*, 225:481–486, 1973.
- R. N. Bergman, L. S. Phillips, and C. Cobelli. Physiologic evaluation of factors controlling glucose tolerance in man. *Journal of Clinical Investigation*, 68:1456–1467, 1981.
- M. Bezzi, T. Nieuw, O. J. M. D. Coenen, and E. D. Angelo. An integrate-and-fire model of a cerebellar granule cell. *Neurocomputing*, 58-60:575–580, 2004.
- E. Biermann and H. Mehnert. DIABLOG: A simulation program of insulin–glucose dynamics for education of patients. *Computational Methods and Program in Biomedicine*, 32:311–318, 1990.
- F. Bimbot, J. F. Bonastre, C. Fredouille, C. Gravier, I. Magrin-Chagnolleau, S. Meignier, J. Ortega-Garcia, D. Petrovska-Delacretaz, and D. A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP Journal on Applied Signal Processing*, 4:430–451, 2004.
- K. B. Bischoff. Physiological pharmacokinetics. *Bulletin of Mathematical Biology*, 48(3–4):309–322, 1986.

- F. Black and N. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–659, 1973.
- B. F. De Blasio, P. Bak, Pociot F., A. E. Karlsen, and J. Nerup. Onset of type 1 diabetes – a dynamical instability. *Diabetes*, 48:1677–1685, 1999.
- B. Bogert, M. Healy, and J. Tukey. The quefrency analysis of time series for echos. In *M. Rosenblatt (Ed.), Proceedings of Symposium on Time Series Analysis, Chapter 15*, pages 209–243. Wiley, New York, 1963.
- E. Boland, M. Grey, A. Oesterle, L. Fredrickson, and W. V. Tamborlane. Continuous subcutaneous insulin infusion: A new way to lower risk of severe hypoglycemia, improve metabolic control, and enhance coping in adolescent with type 1 diabetes. *Diabetes Care*, 22:1779–1784, 1999.
- L. Botelho and H. Coelho. A schema-associative model of memory. *Proceedings of the 4th Golden West International Conference on Intelligent Systems (GWICS'95)*, pages 81–85, 1995.
- L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.
- P. Boyle and F. Boyle. *Derivatives: The Tools That Changed Finance*. London: Risk Books, 2001.
- T. Bremer and D. A. Gough. Is blood glucose predictable from previous values? *Diabetes*, 48(3):445–451, 1999.
- J. Brod. *The Kidney*. Butterworth & Co. Ltd., London, 1973.
- P. Brunetti, C. Cobelli, P. Cruciani, P. G. Fabietti, F. Filipucci, and F. Santeusano. A simulation study on a self-tuning portable controller of blood glucose. *International Journal of Artificial Organ*, 16:51–57, 1993.
- C2iWeb. Centre for computational intelligence, school of computer engineering, nanyang technological university, singapore. <http://www.c2i.ntu.edu.sg>, Online.
- J. P. Campbell Jr. Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 1997.
- F. Camu and E. Rasio. Peripheral glucose uptake in relation to physiological levels of plasma and lymph insulin. *European Journal of Clinical Investigations*, 2:188–194, 1972.
- B. Candas and J. Randziuk. An adaptive plasma glucose controller based on a nonlinear insulin/glucose model. *IEEE Transactions on Biomedical Engineering*, 41(2):116–124, 1994.
- G. A. Carpenter and S. Grossberg. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, 1991.
- E. R. Carson and D. G. Cramp. A systems model of blood glucose control. *International Journal of Bio-Medical Computing*, 7:21–34, 1976.

- R. M. Castro, M. J. Coates, and R. D. Nowak. Likelihood based hierarchical clustering. *IEEE Transactions on Signal Processing*, 52(8):2308–2321, 2004.
- S. Cetinkunt and A. Donmez. CMAC learning controller for servo control of high precision machine tools. In *Proceedings of American Control Conference*, pages 1976–1980, San Fransico, CA, 1993.
- D. M. Chance. *An Introduction to Derivatives & Risk Management*. Thomson, 6 edition, 2004.
- T. Chen and S. Amari. Stability of asymmetric hopfield network. *IEEE Transactions on Neural Networks*, 12(1):159–163, 2001.
- Y. Chen, X. Y. Chen, L. B. Jakeman, G. Schalk, B. T. Stokes, and J. R. Wolpaw. The interaction of a new motor skill and an old one: H-reflex conditioning and locomotion in rats. *J. Neurosci.*, 25(29):6898–6906, 2005.
- P. Y. K. Cheng. *Predicting Bank Failures: A Comparison of the Cox Proportional Hazards Model and the Time Varying Covariates Model*. PhD thesis, Nanyang Business School, Nanyang Technological University, Singapore, 2002.
- P. Y. K. Cheng, M. I. Mah, and C. Quek. Predicting the impact anticipatory action on us stock market event study using anfis (a neural fuzzy model). *Proceedings of 2005 IEEE Congress on Evolutionary Computation (CEC2005)*, 3:2669–2676, 2005.
- ChicagoBank. Repository for bank data. *Federal Reserve Bank of Chicago*, <http://www.chicagofed.org>, Online.
- T. A. Chung and F. Scarselli. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998.
- P. S. Churchland and T. J. Sejnowski. *The Computational Brain*. Cambridge, MA: MIT Press, 1994.
- O. J. M. D. Coenen, C. Boucheny, D. Marchal, M. P. Arnold, B. Barbaour, E. Ortigosa, R. Agis, and E. Ros. Spiking cerebellar model for adaptive control. *Forum of European Neuroscience Abstract*, 2004.
- R. Cole and J. Gunther. Separating the likelihood and timing of bank failure. *Journal of Banking and Finance*, 19(6):1073–1089, 1995.
- S. Commuri, S. Jagannathan, and F. L. Lewis. CMAC neural network control of robot manipulators. *J. Robot Syst.*, 14(6):465–482, 1997.
- P. E. Cryer. Iatrogenic hypoglycemia as a cause of hypoglycemia associated autonomic failure in IDDM: a vicious cycle. *Diabetes*, 41:255–260, 1992.
- DARPA. Biologically-inspired cognitive architectures (BICA) - proposer information pamphlet. http://www.darpa.mil/ipto/solicitations/open/05-18_PIP.htm, Online.
- A. De Gaetano and O. Arino. Mathematical modeling of the intravenous glucose tolerance test. *Journal of Mathematical Biology*, 40:136–168, 2000.

- R. A. DeFronzo. Pathogenesis of type 2 (non-insulin dependent) diabetes mellitus: a balanced overview. *Diabetologia*, 35(4):389–397, 1992.
- R. A. DeFronzo, E. Jacot, E. Jequier, E. Maeder, J. Wahren, and J. P. Felber. The effect of insulin on the disposal of intravenous glucose. *Diabetes*, 30:1000–1007, 1981.
- L. Deng and D. O’Shaughnessy. *Speech Processing: A Dynamic and Optimization-Oriented Approach*. Marcel Dekker, Inc., 2003.
- M. Deppe, W. Schwindt, J. Kramer, H. Kugel, H. Plassmann, P. Kenning, and E. B. Ringelstein. Evidence for a neural correlate of a framing effect bias-specific activity in the ventromedial prefrontal cortex during credibility judgments. *Brain Research Bulletin*, 67:413–421, 2005.
- J. E. Desmond and J. A. Fiez. Neuroimaging studies of the cerebellum: language, learning and memory. *Trends in Cognitive Sciences*, 2(9):355–362, 1998.
- Diabetes Control and Complication Trial. Diabetes control and complication trial (DCCT): The effect of intensive treatment of diabetes in the development and progression of long-term complications in insulin-dependent diabetes mellitus. *The New England Journal of Medicine*, 329:977–986, 1993.
- Diabetes Control and Complication Trial. The effect of intensive treatment of diabetes in the development and progression of long-term complications in insulin-dependent diabetes mellitus. *Diabetes Care*, 18:1468–1478, 1995.
- M. B. Dick, R. Andel, S. Hsieh, J. Bricker, D. S. Davis, and C. Dick-Muehlke. Contextual interference and motor skill learning in alzheimer’s disease. *Aging, Neuropsychology, and Cognition*, 7(4):273–287, 2000.
- G. R. Doddington. Speaker recognition – identifying people by their voices. *Proceedings of the IEEE*, 73:1651–1664, 1985.
- N. M. Doliba and F. M. Matschinsky. *The Metabolic Basis of Insulin Secretion*. In M. A. Sperling (Ed.), *Type I Diabetes: Etiology and Treatment*. Humana Press, 2003.
- C. V. Doran, J. G. Chase, G. M. Shaw, K. T. Moorhead, and N. H. Hudson. Derivative weighted active insulin control algorithms and intensive care unit trials. *Control Engineering Practice*, 13:1129–1137, 2005.
- K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, 12:961–974, 1999.
- W. Duch. Brain-inspired conscious computing architecture. *Journal of Mind and Behavior*, 26(1–2):1–22, 2005.
- W. C. Duckworth, C. D. Saudek, and R. R. Henry. Why intraperitoneal delivery of insulin with implantable pumps in niddm? *Diabetes*, 41:657–661, 1992.
- H. Eichenbaum. *The Cognitive Neuroscience of Memory: An Introduction*. Oxford University Press, 2002.
- W. Enders. *Applied Econometric Time Series (2nd Edition)*. Wiley, 2003.

- A. Epple and J. E. Brinn. *The Comparative Physiology of the Pancreatic Islets*. Berlin: Springer-Verlag, 1987.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
- A. H. Fagg, N. Sitkoff, A. G. Barto, and J. C. Houk. A computational model of cerebellar learning for limb control. *Proceedings of the Spring 1997 Meeting of the Neural Control of Movement*, 1997.
- E. F. Fama. The behavior of stock-market prices. *Journal of Business*, 38:34–105, 1965.
- E. F. Fama. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25:383–417, 1970.
- K. R. Farrell and R. J. Mammone. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing*, 2(1):194–205, 1994.
- D. Fasulo. An analysis of recent work on clustering algorithms. Technical report 01-03-02, Department of Computer Science & Engineering, University of Washington, Seattle, USA, 1999.
- K. D. Federmeier, J. A. Kleim, and W. T. Greenough. Learning-induces multiple synapse formation in rat cerebellar cortex. *Neuroscience Letters*, 332:180–184, 2002.
- U. Fischer, E. Salzsieder, E. -J. Freyse, and G. Albrecht. Experimental validation of a glucose-insulin control model to simulate patterns in glucose turnover. *Computer Methods and Programs in Biomedicine*, 32:249–258, 1990.
- M. E. Fisher. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE Transaction on Biomedical Engineering*, 38(1):57–61, 1991.
- M. E. Fisher and K. L. Teo. Optimal insulin infusion resulting from a mathematical model of blood glucose dynamics. *IEEE Transaction on Biomedical Engineering*, 36:479–486, 1989.
- S. J. Fisher and C. R. Kahn. Insulin signaling is required for insulin's direct and indirect action on hepatic glucose production. *Journal of Clinical Investigation*, 111(4):463–468, 2003.
- P. M. Fitts and M. I. Posner. *Human Performance*. Belmont, CA: Brooks/Cole, 1967.
- L. Fletcher, G. Hraban, P. Huang, B. Srinivasan, and R. Venook. Feasibility of an implanted, closed-loop, blood-glucose control device. *Immunology*, 230, 2001.
- E. W. Forgy. Cluster analysis of multivariate data efficiency versus interpretability of classification. *Biometric Society Meetings, Riverside, California. Abstract in Biometrics*, 213: 768, 1965.

- S. Furui. Toward flexible speech recognition - recent progress at tokyo institute of technology. *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 1:631–636, 2001.
- S. Furui. 50 years of progress in speech and speaker recognition. *Proceedings of the International Conference on Speech and Computer*, pages 1–9, 2005.
- S. Furui. Recent advances in speaker recognition. *Patter Recognition Letters*, 18(9):859–872, 1997.
- B. Gabrys and A. Bargiela. General fuzzy min-max neural network for clustering and classification. *IEEE Transactions on Neural Networks*, 11(3):769–783, 2000.
- X. Gao, C. Wang, X. M. Gao, and S. J. Ovaska. A new CMAC neural network model with adaptive quantization input layer. *3rd. Intl. Conf. Singal Processing*, 2:1417–1420, 1996.
- S. Geman and E. Bienenstock. Neural networks and the bias–variance dilemma. *Neural Computation*, 4:1–58, 1992.
- R. Gencay. The predictability of security returns with simple trading rules. *Journal of Empirical Finance*, 5(4):347–359, 1998.
- Z. J. Geng and W. Shen. Fingerprint classification using fuzzy cerebellar model arithmetic computer neural networks. *Journal of Electronic Imaging*, 6(3):311–318, 1997.
- A. B. Geva. Hierarchical unsupervised fuzzy clustering. *IEEE Transactions on Fuzzy System*, 7(6):723–733, 1999.
- F. H. Glanz and W. T. Miller. Deconvolution using a CMAC neural network. *Neural Networks*, 1:440, 1988.
- M. A. Gluck, M. Meeter, and C. E. Myers. Computational models of the hippocampal region: linking incremental learning and episodic memory. *Trends in Cognitive Sciences*, 7(6):269–276, 2003.
- M. A. Gluck, C. E. Myers, and M. Meeter. Cortico-hippocampal interaction and adaptive stimulus representation: A neurocomputational theory of associative learning and memory. *Neural Networks*, 18(9):1265–1279, 2005.
- GlucoSim. GlucoSim: A web-based educational simulation package for glucose-insulin levels in the human body. *Illinois Institute of Technology*, <http://216.47.139.198/glucosim/gsimul.html>, Online.
- B. Gold and N. Morgan. *Speech and Audio Signal Processing*. John Wiley & Sons, Inc., 2000.
- S. J. Goodbody and D. M. Wolpert. Temporal and amplitude generalization in motor learning. *Journal of Neurophysiology*, 79:1825–1838, 1998.
- B. Graham and D. Willshaw. Capacity and information efficiency of the associative net. *Network*, 8:35–54, 1997.
- F. M. Gribble. A higher power for insulin. *Nature*, 434:965–966, 2005.

- S. Grossberg. Linking mind to brain: The mathematics of biological intelligence. *Notices of the American Mathematical Society*, 47(11):1361–1372, 2000.
- S. Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, 1(1):17–61, 1988.
- S. J. Grossman and J. E. Stiglitz. On the impossibility of informationally efficient markets. *American Economic Review*, 70(393–408), 1980.
- C. Harris, X. Hong, and Q. Gan. *Adaptive Modelling, Estimation and Fusion from Data A Neurofuzzy Approach*. Springer: Berlin Heidelberg, 2002.
- M. H. Hassoun. *Associative Neural Memories: Theory and Implementation*. Oxford University Press, 1993.
- S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall: Upper Saddle River, NJ, USA, 1994.
- C. He, L. Xu, and Y. Zhang. Learning convergence of CMAC algorithm. *Neural Processing Letters*, 14(1):61–74, 2001.
- D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. New York: John Wiley and Sons, 1949.
- A. Helman, M. Marre, E. Bobbioni, P. H. Poussier, G. Reach, and R. Assan. The brain-islet axis: the nervous control of the endocrine pancreas. *Diabetes and Metabolism*, 8:53–64, 1982.
- J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–136, 2001.
- A. L. Higgins and R. E. Wohlford. A new method of text-independent speaker recognition. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 869–872, 1986.
- A. L. Higgins, L. G. Bahler, and J. E. Porter. Voice identification using nearest-neighbor distance measure. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 375–378, 1993.
- G. E. Hinton and J. A. Anderson. *Parallel Models of Associative Memory*. Lawrence Erlbaum Assoc., 1989.
- T. Hirsimäki. A decoder for large-vocabulary continuous speech recognition. Master's thesis, Helsinki University of Technology, 2002.
- K. J. Holyoak and B. A. Spellman. Thinking. *Annual Review of Psychology*, 44:265–315, 1993.
- M. Honda. Human speech production mechanisms. *NTT Technical Review*, 1(2):24–29, 2003.

- R. P. L. M. Hoogma, P. J. Hammond, R. Gomis, D. Kerr, D. Bruttomesso, K. P. Bouter, K. J. Wiefels, H. de la Calle, D. H. Schweitzer, M. Pfohl, E. Torlone, L. G. Krinelke, and G. B. Bolli. Comparison of the effects of continuous subcutaneous insulin infusion (CSII) and NPH-based multiple daily insulin injections (MDI) on glycaemic control and quality of life: results of the 5-nations trial. *Diabetic Medicine*, 23:141–147, 2005.
- J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Sciences*, 79:2554–2558, 1982.
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- J. C. Houk, J. T. Buckingham, and A. G. Barto. Models of the cerebellum and motor learning. *Behavioral and Brain Sciences*, 19(3):368–383, 1996.
- R. Hovorka. Management of diabetes using adaptive control. *International Journal on Adaptive Control and Signal Processing*, 19:309–325, 2005.
- R. Hovorka. Continuous glucose monitoring and closed-loop systems. *Diabetic Medicine*, 23(1):1–12, 2006.
- R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, and M. E. Wilinska. Nonlinear model predictive control of glucose concentration in subjects with type-1 diabetes. *Physiological Measurement*, 25:905–920, 2004.
- HPBSg. Health promotion board singapore. <http://www.hpb.gov.sg>, Online.
- J. Hu and G. Pratt. Self-organizing cmac neural networks and adaptive dynamic control. *Proceedings of IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pages 259–265, 1999.
- J. Hu, J. Pratt, and G. Pratt. Stable adaptive control of a bipedal walking robot with CMAC neural networks. *Proceedings of IEEE International Conference on Robotics and Automation*, 2:1050–1056, 1999.
- X. Hu and J. Wang. Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network. *IEEE Transactions on Neural Networks*, 17(6):1487–1499, 2006.
- K. Huang, H. Yang, I. King, and M.R. Lyu. Local learning vs. global learning: An introduction to maxi-min margin machine. In *Wang L. (Eds.), Support Vector Machines: Theory and Applications*, volume 177, pages 113–132. Springer, 2005.
- K. L. Huang, S. C. Hsieh, and H. C. Fu. Cascade-CMAC neural network applications on the color scanner to printer calibration. *Intl. Conf. Neural Networks*, 1:10–15, Jun, 9–12 1997.
- Z. Huang and Q. Shen. Fuzzy interpolative reasoning via scale and move transformation. *IEEE Transactions on Fuzzy Systems*, 14(2):340–359, 2006.
- D. H. Hubel and T. N. Weisel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

- J. Hutchinson, A. Lo, and T. Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49:851–889, 1994.
- Y. Iiguni. Hierarchical image coding via cerebellar model arithmetic computers. *IEEE Transactions on Image Processing*, 5(10):1393–1401, 1996.
- W. T. Miller III. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE J. Robot. Autom.*, RA-3:157–165, 1987.
- H. Imamizu, S. Miyauchi, T. Tamada, Y. Sasaki, R. Takino, B. Puetz, T. Yoshioka, and M. Kawato. Human cerebellar activity reflecting an acquired internal model of a novel tool. *Nature*, 403:192–195, 2000.
- E. Ionescu, F. Rohner-Jeanreanud, H. R. Berthoud, and B. Jeanreanud. Increases in plasma insulin levels in response to electrical stimulation of the dorsal motor nucleus of the vagus nerve. *Endocrinology*, 112:904–910, 1983.
- M. Ito. Mechanisms of motor learning in the cerebellum. *Brain Research*, 886:237–245, 2000.
- M. Ito. *The Cerebellum and Neural Control*. New York: Raven Press, 1984.
- A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.
- A. K. Jain, A. Ross, and S. Pankanti. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security*, 1(2):125–143, 2006.
- T. Jessell. Development of the nervous system. In *E. R. Kandel and J. H. Schwartz and T. M. Jessell (Eds.), Essentials of Neural Science and Behavior*, pages 89–110. McGraw-Hill, 1996.
- H. Jiang, C. Quek, and G. S. Ng. FCMAC-EWS: A bank failure early warning system based on novel localized pattern learning, semantically associative fuzzy neural network. *Expert Systems with Applications*, In press, 2007.
- S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 323:241–254, 1967.
- C. C. Jou. A fuzzy cerebellar model articulation controller. *IEEE International conference on Fuzzy Systems*, pages 1171–1178, 1992.
- J. H. Kaas, R. J. Nelson, M. Sur, and C. S. Lin. Multiple representations of the body within the primary somatosensory cortex of primates. *Science*, 204:521–523, 1979.
- J. H. Kaas, R. J. Nelson, M. Sur, and M. M. Merzenich. Organization of somatosensory cortex in primates. In *F. O. Schmitt, F. G. Worden, G. Adelman, and S. G. Dennis (Eds.), The Organization of the Cerebral Cortex: Proceedings of a Neuroscience Research Program Colloquium*, pages 237–261. MIT Press, 1981.

- E. R. Kandel, I. Kupfermann, and S. Iversen. *E. R. Kandel, J. H. Schwartz and T. M. Jessell (Eds.), Principles of Neural Science, 4th Edition*, chapter Learning and Memory, pages 1227–1246. Series on Studies in Computational Intelligence: Springer-Verlag, 2000a.
- E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science, 4th Edition*. McGraw-Hill, Health Professions Division., 2000b.
- P. Kanerva. *M. H. Hassoun (Ed.), Associative Neural Memories: Theory and Implementation*, chapter Sparse Distributed Memory and Related Models. Oxford University Press, 1993.
- H. Kano and K. Takayama. Learning control of robotic manipulators based on neurological model CMAC. In *Proceedings of 11th Triennial World Congr. Int. Federation of Automat. Contr.*, pages 249–254, Tallinn, USSR, 1990.
- N. Kasabov. *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. Springer-Verlag: London, 2003.
- H. Kaufman, I. Bar-Kana, and K. Sobel. *Direct Adaptive Control Algorithm: Theory and Applications*. Springer Verlag, USA, 1994.
- M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9:718–727, 1999.
- M. Kawato and H. Gomi. The cerebellum and vor/okr learning models. *Trends in Neurosciences*, 15:445–453, 1992.
- M. Kawato and K. Samejima. Efficient reinforcement learning: Computational theories, neuroscience and robotics. *Current Opinion in Neurobiology*, 17(2):205–212, 2007.
- M. Kawato, K. Furawaka, and R. Suzuki. A hierarchical neural network model for the control and learning of voluntary movements. *Biological Cybernetics*, 56:1–17, 1987.
- C. Keber. Option pricing with the genetic programming approach. *Journal of Computational Intelligence in Finance*, 7(6):26–36, 1999.
- J. P. Keener. Diffusion induced oscillatory insulin secretion. *Bulletin of Mathematical Biology*, 63:625–641, 2001.
- J. Ker, C. Hsu, Y. Kuo, and B. Liu. A fuzzy CMAC model for color reproduction. *Fuzzy Sets and Systems*, 91(1):53–68, 1997.
- K. H. Kienitz and T. Yoneyama. A robust controller for insulin pumps based on h-infinity theory. *IEEE Transactions on Biomedical Engineering*, 40:1133–1137, 1993.
- D. Kim. A design of CMAC-based fuzzy logic controller with fast learning and accurate approximation. *Fuzzy Sets and Systems*, 125(1):93–104, 2002.
- H. Kim and C. S. Lin. Use of adaptive resolution for better CMAC learning. *Intl. Joint Conf. Neural Networks*, 117:517–522, 1992.
- Y. H. Kim and F. L. Lewis. Optimal design of CMAC neural-network controller for robot manipulators. *IEEE Transactions on Systems, Man and Cybernetics, Part C, Applications and Reviews*, 30(1):22–31, 2000.

- B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.
- G. L. King and S. M. Jahnsen. Receptor-mediated transport of insulin across endothelial cells. *Science*, 227:1583–1586, 1985.
- S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
- T. Kitamura. *What should be Computed to Understand and Model Brain Function?: From Robotics, Soft Computing, Biology and Neuroscience to Cognitive Philosophy*. Singapore; River Edge, NJ: World Scientific Publishing, 2001.
- J. A. Kleim, E. Lussnig, E. R. Schwars, T. A. Comery, and W. T. Greenough. Synaptogenesis and FOS expression in the motor cortex of the adult rat after motor skill learning. *The Journal of Neuroscience*, 16(14):4529–4535, 1996.
- J. A. Kleim, M. A. Pipitone, C. Czerlanis, and W. T. Greenough. Structural stability within the lateral cerebellar nucleus of the rat following complex motor learning. *Neurobiology of Learning and Memory*, 69:290–306, 1998a.
- J. A. Kleim, R. A. Swain, K. A. Armstrong, R. M. A. Napper, T. A. Jones, and W. T. Greenough. Selective synaptic plasticity within the cerebellar cortex following complex motor skill learning. *Neurobiology of Learning and Memory*, 69:274–289, 1998b.
- Ron Kohavi. The power of decision tables. In Nada Lavrac and Stefan Wrobel, editors, *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 914, pages 174–189, Berlin, Heidelberg, New York, 1995. Springer Verlag.
- T. Kohonen. *Self-Organization and Associative Memory*, 3rd Edition. Berlin, New York: Springer-Verlag, 1989.
- T. K. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 21: 353–359, 1972.
- T. K. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
- B. Kosko. Bidirectional associative memory. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):49–60, 1988.
- B. Kosko. Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11):1329–1333, 1994.
- A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen. Neural networks for vector quantization of speech and images. *IEEE Journal on Selected Areas in Communications*, 8(8):1449–1457, 1990.
- S. Ku, G. A. Larsen, and S. Cetinkunt. Fast servo control for ultra-precision machining at extremely low feed rates. *Mechatronics*, pages 381–393, 1998.
- M. Kurz. Asset prices with rational beliefs. *CEPR Publication No. 375*, 1994a.

- M. Kurz. On rational belief equilibria. *Economic Theory*, 4:859–876, 1994b.
- J. Lakonishok, I. Lee, and A. M. Potesman. *Investor Behavior in the Option Market*. National Bureau of Economic Research, Inc., 2004.
- T. Lam and V. Dietz. Transfer of motor performance in an obstacle avoidance task to different walking conditions. *Journal of Neurophysiology*, 92:2010–2016, 2004.
- H. D. Landahl and G. M. Grodsky. Comparisons of models of insulin release. *Bulletin of Mathematical Biology*, 44:399–409, 1982.
- S. H. Lane, D. A. Handelman, and J. J. Gelfand. Theory and development of higher-order CMAC neural networks. *IEEE Control System Magazine*, 12(2):23–30, 1992.
- G. A. Larsen, S. Cetinkunt, and A. Donmez. CMAC neural network control for high precision motion control in the presence of large friction. *Journal of Dynamic Systems, Measurement, and Control*, 117:415–420, 1995.
- C. C. Lee. Fuzzy logic in control systems: Fuzzy logic controller, part ii. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):419–435, 1990.
- H. M. Lee, C. M. Chen, and Y. F. Lu. A self-organizing HCMAC neural network classifier. *IEEE Transactions on Neural Networks*, pages 15–27, 2003.
- H. S. Lee. On system identification via fuzzy clustering for fuzzy modeling. *Proceedings of IEEE International Conference on Systems, Man & Cybernetics*, 2:1956–1961, 1998.
- S. Lemmetty. Review of speech synthesis technology. Master's thesis, Department of Electrical and Communications Engineering, Helsinki University of Technology, 1999.
- B. E. Levin, V. H. Routh, L. Kang, N. M. Sanders, and A. A. Dunn-Meynell. Neuronal glucosensing: What do we know after 50 years? *Diabetes*, 53:2521–2528, 2004.
- X. Liang and J. Wang. A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints. *IEEE Transactions on Neural Networks*, 11(6):1251–1262, 2000.
- C. S. Lin and C. T. Chiang. Learning convergence of CMAC technique. *IEEE Trans. Neural Networks*, 8(6):1281–1292, 1997.
- C. S. Lin and K. Hyongsuk. CMAC-based adaptive critic self-learning control. *IEEE Trans. Neural Networks*, 2:530–533, 1991.
- C. T. Lin and C. S. G. Lee. *Neural Fuzzy Systems – A Neuro-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice Hall, 1996.
- R. P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, 27(11):47–50, 59–64, 1989.
- M. Lund. A robust sequential test for text-independent speaker verification. *Journal of the Acoustical Society of America*, 99(1):609–621, 1996.

- S. M. Lynch and B. W. Bequette. Estimation-based model predictive control of blood glucose in type-1 diabetics: A simulation study. *Proceedings of IEEE 27th Annual Northeast Bioengineering Conference*, pages 79–80, 2001.
- J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of Symp. Math. Statist. & Probability*, pages 281–297, 1967.
- A. Makroglou, J. Li, and Y. Kuang. Mathematical models and software tools for the glucose-insulin regulatory system and diabetes: An overview. *Applied Numerical Mathematics*, 56:559–573, 2006.
- N. Malfait, D. M. Shiller, and D. J. Ostry. Transfer of motor learning across arm configurations. *Journal of Neuroscience*, 22(22):9656–9660, 2002.
- N. Malfait, P. L. Gribble, and D. J. Ostry. Generalization of motor learning based on multiple field exposures and local adaptation. *Journal of Neurophysiology*, 93:3327–3338, 2005.
- E. H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic systems. *IEEE Transactions on Computers*, C-26:1182–1191, 1977.
- J. G. Marin-Blazquez and Q. Shen. From approximative to descriptive fuzzy classifiers. *IEEE Transactions on Fuzzy Systems*, 10(4):484–497, 2002.
- J. D. Markel and S. B. Davis. Text-independent speaker recognition from a large linguistically unconstrained time-spaced database. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-27(1):74–82, 1979.
- D. Marr. A theory of cerebellar cortex. *Journal of Physiology of London*, 202:437–470, 1969.
- A. Marschner, T. Mell, A. Villringer, F. M. Reischies, and H. R. Heekeren. Reward-based decision-making and aging. *Brain Research Bulletin*, 67:382–390, 2005.
- D. Martin. Early warning of bank failure: A logit regression approach. *Journal of Banking and Finance*, 1(3):249–276, 1977.
- T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. ‘neural-gas’ network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.
- F. M. Matschinsky. A lesson in metabolic regulation inspired by the glucokinase glucose sensor paradigm. *Diabetes*, 45:223–241, 1996.
- J. E. Mazur. *Learning and Behavior*. Pearson/Prentice Hall, 2006.
- J. F. Medina and M. D. Mauk. Simulations of cerebellar motor learning: Computational analysis of plasticity at the mossy fiber to deep nucleus synapse. *Journal of Neuroscience*, 19(16):7140–7151, 1999.
- A. Menozzi and M. Chow. On the training of a multi-resolution CMAC neural network. *23rd. Intl. Conf. Ind. Electron. Contr. Instrum.*, 3:1130–1135, 1997.

- F. A. Middleton and P. L. Strick. The cerebellum: An overview. *Trends in Cognitive Sciences*, 27(9):305–306, 1998a.
- F. A. Middleton and P. L. Strick. Cerebellar output: Motor and cognitive channels. *Trends in Cognitive Sciences*, 27(9):348–354, 1998b.
- R. E. Miller. Pancreatic neuroendocrinology: peripheral neural mechanisms in the regulation of the islets of langerhans. *Endocrine Reviews*, 2:471–494, 1981.
- W. T. Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):825–831, 1989.
- W. T. Miller, F. H. Glanz, and L. G. Kraft. CMAC: An associative neural network alternative to backpropagation. *Proceedings of the IEEE*, 78(10):1561–1657, 1990.
- S. Mitra and Y. Hayashi. Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks*, 11(3):748–768, 2000.
- J. Moody. Fast-learning in multi-resolution hierarchies. In *Adv. Neural Infor. Processing Syst.*, volume 14, pages 29–38. Morgan Kaufman Publishers, 1989.
- J. Moren. *Emotion and Learning: A Computational Model of the Amygdala*. PhD thesis, Lund University, Sweden, 2002.
- J. A. Muth. Rational expectations and the theory of price movements. *Econometrica*, 29(6):315–335, 1961.
- D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems*. Chichester, England; New York: John Wiley, 1997.
- M. C. Nechyba and Y. Xu. Human control strategy: Abstraction, verification, and replication. *IEEE Control Systems Magazine*, 17(5):48–61, 1997.
- A. Newell and H. A. Simon. *Human Problem Solving*. Englewood Cliffs: NJ: Prentice-Hall, 1972.
- J. G. Nicholls, A. R. Martin, B. G. Wallace, and P. A. Fuchs. *From Neuron to Brain*, 4th Edition. Sinauer Associates Inc., 2001.
- J. H. Nie and D. A. Linkens. A fuzzified CMAC self-learning controller. *Second IEEE International Conference on Fuzzy Systems*, 1:500–505, 1993.
- L. T. Nielsen. *Pricing and Hedging of Derivative Securities – Textbook in Continuous-Time Finance Theory*. Oxford University Press, 1999.
- M. Nomura, M. Shichiri, R. Kawamori, Y. Yamasaki, N. Iwama, and H. Abe. A mathematical insulin-secretion model and its validation in isolated rat pancreatic islets perfusion. *Computers and Biomedical Research*, 17:570–579, 1984.
- S. Obici, B. B. Zhang, G. Karkanias, and L. Rossetti. Hypothalamic insulin signaling is required for inhibition of glucose production. *Nature Medicine*, 8(12):1376–1382, 2002.

- R. J. Oentaryo and M. Pasquier. GenSoFNN-Yager: A novel hippocampus-like learning memory system realizing Yager inference. *Proceedings of 2006 IEEE International Joint Conference on Neural Networks*, pages 705–712, 2006.
- J. Oglesby and J. S. Mason. Optimization of neural models for speaker identification. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 261–264, 1990.
- J. Oglesby and J. S. Mason. Radial basis function networks for speaker recognition. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 393–396, 1991.
- M. Oja, A. Kaski, and T. Kohonen. Bibliography of self-organizing map (som) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156, 2002.
- H. Okano, T. Hirano, and E. Balaban. Learning and memory. *Proceedings of the National Academy of Sciences*, 97(23):12403–12404, 2000.
- R. L. Ollerton. Application of optimal control theory to diabetes mellitus. *International Journal of Control*, 50:2503–22, 1989.
- A. V. Oppenheim and R. W. Schaffer. From frequency to quefrency: A history of the cepstrum. *IEEE Signal Processing Magazine*, 21(5):95–106, 2004.
- R. C. O'Reilly. Y. Munakata and M. H. Johnson (Eds.) *Processes of Change in Brain and Cognitive Development: Attention and Performance XXI*, chapter Modeling Integration and Dissociation in Brain and Cognitive Development, pages 375–401. Oxford: Oxford University Press, 2006.
- S. J. Orfanidis. *Introduction to Signal Processing*. Prentice Hall, 1995.
- J. Ozawa, I. Hayashi, and N. Wakami. Formulation of CMAC-fuzzy system. *IEEE International Conference on Fuzzy Systems*, pages 1179–1186, 1992.
- C. Palmer and R. K Meyer. Conceptual and motor learning in music performance. *Psychological Science*, 11(1):63–68, 2000.
- J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.
- R. S. Parker, F. J. Doyle III, and N. A. Peppas. A model-based algorithm for blood glucose control in type-1 diabetic patient. *IEEE Transactions on Biomedical Engineering*, 46: 148–157, 1999.
- R. S. Parker, F. J. Doyle III, and N. A. Peppas. The intravenous route to insulin-dependent diabetes therapy. *IEEE Engineering in Medicine and Biology*, 20(1):65–73, 2001.
- M. Pasquier, C. Quek, and M. Toh. Fuzzylot: A self-organising fuzzy-neural rule-based pilot system for automated vehicles. *Neural Networks*, 14(8):1099–1112, 2001.
- G. Pehlig, P. Tessari, J. E. Gerich, M. W. Haymond, F. J. Service, and R. A. Rizza. Abnormal meal carbohydrate disposition in insulin-dependent diabetes. *Journal of Clinical Investigations*, 74:985–991, 1984.

- E. F. Pfeiffer, C. Thum, and A. H. Clemens. The artificial beta cell – a continuous control of blood sugar by external regulation of insulin infusion. *Hormone and Metabolic Research*, 487:339–342, 1974.
- P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki. An introduction to evaluating biometric systems. *Computer*, 33(2):56–63, 2000.
- A. Pocai, T. K. T. Lam, R. Gutierrez-Juarez, S. Obici, G. J. Schwartz, J. Bryan, L. Aguilar-Bryan, and L. Rossetti. Hypothalamic K_{ATP} channels control hepatic glucose production. *Nature*, 434:1026–1031, 2005.
- L. C. W. Pols. *Spectral analysis and identification of Dutch vowels in monosyllabic words*. PhD thesis, Free University, Amsterdam, The Netherlands, 1966.
- D. Porte Jr., D. G. Baskin, and M. W. Schwartz. Insulin signaling in the central nervous system. *Diabetes*, 54(5):1264–1276, 2005.
- W. R. Puckett. *Dynamic modeling of diabetes mellitus*. PhD thesis, Departement of Chemical Engineering, Wisconsin-Madison, 1992.
- W. R. Puckett and E. N. Lightfoot. A model for multiple subcutaneous insulin injections developed from individual diabetic patient data. *American Journal of Physiology*, 269: E1115–E1124, 1995.
- M. Qi and G. S. Maddala. Option-pricing using artificial neural networks: the case of S&P500 index call options. *Neural Networks in Financial Engineering*, pages 78–92, 1995.
- K.H. Quah and C. Quek. Mces: A novel monte carlo evaluative selection approach for objective feature selections. *IEEE Transactions on Neural Networks*, 18(2):431–448, 2007.
- C. Quek and Z. Guo. FCM-AARS: A fuzzy cerebellar model network based on the approximate analogical reasoning schema. *Under preparation*, 2007.
- C. Quek and P. W. Ng. Realisation of neural network controllers in integrated process supervision. *Artificial Intelligence in Engineering*, 10:135–142, 1996.
- C. Quek, D. Shi, and A. Harkisanka. FCMAC-TVR: A novel semantic association memory with truth value restriction inference. *Submitted to AI Communications*, 2006.
- L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, New Jersey, 1993.
- L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall Signal Processing Series, 1978.
- P. Radzikowski. Non-parametric methods of option pricing. *Proceedings of Informs-Korms (Seoul 2000 conference)*, pages 474–480, 2000.
- M. G. Rahim. *Artificial Neural Networks for Speech Analysis/Synthesis*. Chapman & Hall, 1994.

- M. Rapoport, R. van Reekum, and H. Mayberg. The role of the cerebellum in cognition and behavior. *Journal of Neuropsychiatry and Clinical Neurosciences*, 12:193–198, 2000.
- E. Renard. Implantable insulin delivery pumps. *Minimally Invasive Therapy and Allied Technologies*, 13:328–335, 2004.
- R. J. Rendleman Jr. and B. J. Bartter. Two-state option pricing. *Journal of Finance*, 34:1093–1110, 1979.
- L. J. Rips. Reasoning. *Annual Review of Psychology*, 41:321–353, 1990.
- F. Rosenblatt. The perceptron, a probabilistic model for information storage and organization in the brain. *Psychological Review*, 62:589, 1958.
- J. Rosenstock. Insulin therapy: Optimizing control in the type-1 and type-2 diabetes. *Clinical Cornerstone*, 4:50–64, 2001.
- R. J. Rubin, W. M. Altman, and D. N. Mendelson. Health care expenditures for people with diabetes mellitus. *Journal of Clinical Endocrinology and Metabolism*, 78:809A–809F, 1992.
- L. Rudasi and S. A. Zahorian. Text-independent talker identification with neural networks. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 389–392, 1991.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *D. E. Rumelhart, J. L. McClelland et al. (Eds.), Parallel Distributed Processing, Vol. 1, Chapter. 8*, chapter Learning internal representations by error propagation, pages 375–401. Cambridge, MA: MIT Press, 1986.
- M. S. Salman. The cerebellum: it's about time! but timing is not everything—new insights into the role of the cerebellum in timing motor and cognitive tasks. *Journal of Child Neurology*, 17(1):1–9, 2002.
- Y. Salu. *Understanding Brain and Mind: A Connectionist Perspective*. Singapore: World Scientific Publishing, 2001.
- M. Savic and S. K. Gupta. Variable parameter speaker verification system based on hidden markov modeling. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 281–284, 1990.
- R. A. Scheidt, J. B. Dingwell, and F. A. Mussa-Ivaldi. Learning to move amid uncertainty. *J. Neurophysiol.*, 86:971–985, 2001.
- L. M. Schetky, P. Jardine, and F. Moussy. A closed loop implantable artificial pancreas using thin film nitinol mems pumps. *Proceedings of International Conference on Shape Memory and Superelastic Technologies (SMST-2003)*, 2003.
- E. D. Schutter. A new functional role for cerebellar long term depression. *Progress in Brain Research*, 114:529–542, 1997.
- M. W. Schwartz. Central nervous system control of food intake. *Nature*, 404:661–671, 2000.

- M. W. Schwartz and D. Porte Jr. Diabetes, obesity and the brain. *Science*, 307:375–379, 2005.
- R. Schwartz, S. Roucos, and M. Berouti. The application of probability density estimation to text-independent speaker identification. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 1649–1652, 1982.
- P. Shaw, D. Greenstein, J. Lerch, L. Clasen, R. Lenroot, N. Gogtay, A. Evans, J. Rapoport, and J. Giedd. Intellectual ability and cortical development in children and adolescents. *Nature*, 440(3):676–679, 2006.
- Z. Shi and J. Shi. Perspectives on cognitive informatics. *Proceedings of The Second IEEE International Conference on Cognitive Informatics*, pages 129–133, 2003.
- M. Shidara, K. Kawano, H. Gomi, and M. Kawato. Inverse-dynamics model eye movement control by purkinje cells in the cerebellum. *Nature*, 365:50–52, 1993.
- S. Shimoda, K. Nishida, M. Sakakida, Y. Konno, K. Ichinose, M. Urkhara, T. Nowak, and M. Shichiri. Closed-loop subcutaneous insulin infusion algorithm with a short acting insulin analog for long-term clinical application of a wearable artificial endocrine pancreas. *Frontiers of Medical and Biological Engineering*, 8:197–211, 1997.
- F. H. Silverman. *Essentials of Speech, Language, and Hearing Disorders*. Thinking Publications, 2003.
- J. Sim, W. L. Tung, and C. Quek. FCMAC-Yager: A novel Yager inference scheme based fuzzy CMAC. *IEEE Transactions on Neural Networks*, 17(6):1394–1410, 2006.
- D. Simmons, C. F. Thompson, C. Conroy, and D. J. Scott. Use of insulin pumps in pregnancies complicated by type-2 diabetes and gestational diabetes in a multi-ethnic community. *Diabetes Care*, 24:2078–2082, 2001.
- G. Simon, G. Branderberger, and M. Follenius. Ultradian oscillations of plasma glucose, insulin and c-peptide in man during continuous enteral nutrition. *Journal of Clinical Endocrinology & Metabolism*, 64:669–674, 1987.
- J. Sinkey Jr. A multivariate statistical analysis of the characteristics of problem banks. *Journal of Finance*, 1:21–36, 1975.
- P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman: London, UK, 1973.
- F. K. Soong, A. Rosenberg, L. Rabiner, and B. Juang. A vector quantization approach to speaker recognition. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 387–390, 1985.
- J. T. Sorensen. *A Physiologic Model of Glucose Metabolism in Man and its Use to Design and Assess Improved Insulin Therapies for Diabetes*. PhD thesis, Departement of Chemical Engineering, MIT, 1985.
- D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.
- L. R. Squire, B. Knowlton, and G. Musen. The structure and organization of memory. *Annual Review of Psychology*, 44:453–495, 1993.

- P. M. Steele and M. D. Mauk. Inhibitory control of LTP and LTD: Stability of synapse strength. *Journal of Neurophysiology*, 81(4):1559–1566, 1999.
- P. Strata and F. Rossi. Plasticity of the olivocerebellar pathway. *Trends in Neuroscience*, 21:407–413, 1998.
- J. Sturis, K. S. Polonsky, E. Mosekilde, and E. van Cauter. Computer-model for mechanisms underlying ultradian oscillations of insulin and glucose. *American Journal of Physiology Endocrinology and Metabolism*, 260:E801–E809, 1991.
- M. Sugeno and K. Murakami. Fuzzy parking control of model car. *Proceedings of 23rd IEEE Conference on Decision and Control*, 1984.
- M. Sugeno and K. Murakami. An experimental study on fuzzy parking control using a model car. *Proceedings of Industrial Applications of Fuzzy Control*, pages 125–138, 1985.
- M. Sugeno and M. Nishida. Fuzzy control of model car. *Fuzzy Sets & Systems*, 16:103–113, 1985.
- G. W. Swan. An optimal control model of diabetes mellitus. *Bulletin of Mathematical Biology*, 44:793–808, 1982.
- J. D. Sweatt. *Mechanisms of Memory*. Elsevier, 2003.
- T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15:116–132, 1985.
- T. Z. Tan, G. S. Ng, and C. Quek. Ovarian cancer diagnosis by hippocampus and neocortex-inspired learning memory structures. *Neural Networks*, 18(5–6):818–825, 2005.
- J. G. Taylor. The brain as a complex system. [Online] http://complexsystems.lri.fr/Portal/tiki-index.php?page=The+Brain+as+a+Complex+System&bl#The_Brain_as_a_Complex_system, 2005.
- S. D. Teddy and E. M-K. Lai. Hierarchical clustering for efficient memory allocation in CMAC neural networks. *Proceedings of International Conference on Artificial Neural Networks: Biological Inspiration. Lecture Notes in Computer Science*, 3697:473–478, 2005.
- S. D. Teddy, E. M-K. Lai, and C. Quek. A neurologically inspired cerebellar model of the human blood glucose metabolic process. *Proceedings of 12th International Conference on BioMedical Engineering*, 2005.
- S. D. Teddy, E. M-K. Lai, and C. Quek. A neuropsychologically-inspired computational approach to the generalization of cerebellar learning. *Proceedings of 13th International Conference on Neural Information Processing, Part I. Irwin King et al. (Eds), Lecture Notes in Computer Science*, 4232:155–164, 2006a.
- S. D. Teddy, E. M-K. Lai, and C. Quek. A brain-inspired cerebellar associative memory approach to option pricing and arbitrage trading. *Proceedings of 13th International Conference on Neural Information Processing, Part III. Irwin King et al. (Eds), Lecture Notes in Computer Science*, 4234:370–379, 2006b.

- S. D. Teddy, E. M-K. Lai, and C. Quek. Hierarchically clustered adaptive quantization CMAC and its learning convergence. *IEEE Transactions on Neural Networks*, 18(6): 1658–1682, 2007a.
- S. D. Teddy, E. M-K. Lai, and C. Quek. A cerebellar associative memory approach to option pricing and arbitrage trading. *Neurocomputing: under review*, 2007b.
- S. D. Teddy, C. Quek, and E. M-K. Lai. PSECMAC: A novel self-organizing multi resolution associative memory architecture. *IEEE Transactions on Neural Networks: under 2nd review*, 2007c.
- S. D. Teddy, C. Quek, and E. M-K. Lai. TSL-CMAC: A neurologically-inspired three-stage cmac learning paradigm. *IEEE Transactions on Neural Networks: under preparation*, 2007d.
- S. D. Teddy, C. Quek, and E. M-K. Lai. A cerebellar-based approach to intelligent speaker verification. *Expert Systems with Applications: under preparation*, 2007e.
- S. D. Teddy, C. Quek, and E. M-K. Lai. Modeling of intra and inter-days glucose metabolic profiles of a healthy subject for diabetes treatment using a cerebellar-based computational model. *Artificial Intelligence in Medicine: under preparation*, 2007f.
- S. D. Teddy, C. Quek, and E. M-K. Lai. A cerebellar-based computational β -cell for the personalized treatment of diabetes. *Neural Computation: under preparation*, 2007g.
- W. Tetschner. *Voice Processing*. Artech House, Boston, 1993.
- W. T. Thach. On the specific role of the cerebellum in motor learning and cognition: Clues from PET activation and lesion studies in man. *Behavioral and Brain Sciences*, 19(3): 411–431, 1996.
- W. T. Thach. What is the role of the cerebellum in motor learning and cognition? *Trends in Cognitive Sciences*, 27(9):331–337, 1998.
- R. F. Thompson and S. A. Madigan. *Memory: The key to Consciousness*. Washington, D.C.: Joseph Henry Press, 2005.
- C. Ting and C. Quek. TSK-FCMAC: A novel fuzzy CMAC based on the zero-ordered TSK fuzzy inference scheme. *Submitted to Artificial Intelligence Journal*, 2006.
- C. W. Ting. Learning convergence of $\text{tsk}^0\text{-fcmac}$: A novel fuzzy cmac based on the zero-ordered tsk fuzzy inference scheme. Technical Report C2i-TR-04/003, Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, 2004.
- N. Z. Tishby. On the application of mixture ar hidden markov models to text-independent speaker recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-39:563–569, 1991.
- P. D. Tomporowski. *The Psychology of Skill: A life-Span Approach*. Praeger, 2003.
- Z. Trajanoski and P. Wach. Neural predictive controller for insulin delivery using the subcutaneous route. *IEEE Transaction on Biomedical Engineering*, 45:1122–1134, 1998.

- V. Tresp, T. Briegel, and J. Moody. Neural network models for the blood glucose metabolism of a diabetic. *IEEE Transactions on Neural Networks*, 10(5):1204–1213, 1999.
- Y. Tsukamoto. *M. M. Gupta, R. K. Ragade and R. R. Yager (Eds.), Advances in Fuzzy Set Theory and Applications*, chapter An approach to fuzzy reasoning method. North-Holland, Amsterdam, 1979.
- W. L. Tung. *A Generalized Framework for Fuzzy Neural Architecture*. PhD thesis, Nanyang Technological University, 2003.
- W. L. Tung and C. Quek. GenSoFNN: A generic self-organizing fuzzy neural network. *IEEE Transactions on Neural Networks*, 13(5):1075–1086, 2002.
- W. L. Tung and C. Quek. GenSo-OPATS: A brain-inspired dynamically evolving option pricing model and arbitrage trading system. *Proceedings of IEEE CEC 2005, Edinburgh, Scotland*, 3:2429–2436, 2005.
- W. L. Tung and C. Quek. *Chance Discoveries in Real World Decision Making*, chapter A NeuroCognitive Approach to Decision-Making in Chance Discovery, pages 223–242. Series on Studies in Computational Intelligence: Springer-Verlag, 2006a.
- W. L. Tung and C. Quek. A hippocampal-inspired self-organising learning memory model with analogical reasoning for decision support. *Proceedings of 2006 IEEE World Congress on Computational Intelligence*, 2006b.
- W. L. Tung and C. Quek. A brain-inspired fuzzy semantic memory model for learning and reasoning with uncertainty. *Neural Computing & Applications: to appear*, 2007.
- W. L. Tung, C. Quek, and P. Y. K. Cheng. GenSo-EWS: A novel neural-fuzzy based early warning system for predicting bank failures. *Neural Networks*, 17(4):567–587, 2004.
- K. Turnheim and W. K. Waldhausl. Essentials of insulin pharmacokinetics. *Wiener Klin Wochenschrift*, 100(3):65–72, 1988.
- P. Tyagi. Insulin delivery systems: Present trends and the future direction. *Indian Journal of Pharmacy*, 34:379–389, 2002.
- T. Tyrrell and D. Willshaw. Cerebellar cortex: Its simulation and the relevance of Marr's theory. *Philosophical Transactions: Biological Sciences*, 336(1277):239–257, 1992.
- J. Volkman, S. S. Stevens, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.
- J. Voogd and M. Glickstein. The anatomy of the cerebellum. *Trends in Cognitive Sciences*, 2(9):307–313, 1998.
- A. Wahab, E. C. Tan, and H. Abut. HCMAC amplitude spectral subtraction for noise cancellation. *Intl. Conf. Neural Inform. Processing*, 2001.
- A. Wahab, G. S. Ng, and R. Dickiyanto. Speaker authentication system using soft computing approaches. *Neurocomputing*, 68:13–37, 2005.

- J Wainstein, M. Metzger, M. Boaz, O. Minuchin, Y. Cohen, A. Yaffe, Y. I. R. Yerushalmy, and I. Harman-Boehm. Insulin pump therapy vs. multiple daily injections on obese type 2 diabetic patients. *Diabetic Medicine*, 22:1037–1046, 2005.
- J. Wang. A deterministic annealing neural network for convex programming. *Neural Networks*, 7(4):629–641, 1994.
- Y. Wang. On cognitive informatics. *Brain and Mind*, 4(2):151–167, 2003.
- Y. Wang. Cognitive informatics: Exploring the theoretical foundations for natural intelligence, neural informatics, autonomic computing, and agent systems. *International Journal of Cognitive Informatics and Natural Intelligence*, 1(1):1–10, 2007.
- Y. Wang and W. Kinsner. Recent advances in cognitive informatics. *IEEE Trans on Systems, Man and Cybernetics, Part C*, 36(2):121–123, 2006.
- Y. Wang and Y. Wang. Cognitive informatics models of the brain. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 36(2):203–207, 2006.
- J. H. Ward. Hierarchical grouping to optimise objective function. *Journal of American Statistical Association*, 58(301):236–244, 1963.
- WCME. Chicago mercantile exchange, us. <http://www.cme.com>, Online.
- C. Weigelt, A. M. Williams, T. Wingrove, and M. A. Scott. Transfer of motor skill learning in association football. *Ergonomics*, 43(10):1698–1707, 2000.
- WEKA. WEKA 3: Data mining software in java. *The University of Waikato*, <http://www.cs.waikato.ac.nz/ml/weka/>, Online.
- R. W. Whaley. On valuing american futures options. In *D. F. DeRosa (Ed.), Currency Derivatives: Pricing Theory, Exotic Options, and Hedging Applications*, pages 147–161, 1998.
- M. F. White. Insulin signaling in health and disease. *Science*, 302:1710–1711, 2003.
- B. Widrow and M. E. Hoff. Adaptive switching circuits. *Proc. 1960 IRE West. Elect. Show Conv. Rec., Part 4*, pages 96–104, 1960.
- B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. New Jersey: Prentice-Hall inc., 1985.
- S. Wild, G. Roglic, A. Green, R. Sicree, and H. King. Global prevalence of diabetes. *Diabetes Care*, 27(5):1047–1053, 2004.
- R. P. Wildes. Iris recognition: An emerging biometric technology. *Proceedings of the IEEE*, 85(9):1348–1363, 1997.
- D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969.
- Y. Wong and A. Sideris. Learning convergence in the cerebellar model articulation controller. *IEEE Trans. Neural Networks*, 3(1):115–121, 1992.

- S. Woods, E. Lotter, L. McKay, and D. Porte Jr. Chronic intracerebroventricular infusion of insulin reduces food intake and body weight of baboons. *Nature*, 282:503–505, 1979.
- S. C. Woods and D. Porte Jr. Autonomic control of the endocrine pancreas. *Physiological Reviews*, 54:956–619, 1974.
- S. C. Woods, D. Porte Jr., E. Bobbioni, E. Ionescu, J. F. Sauter, F. Rohner-Jeanrenaud, and B. Jeanrenaud. Insulin: its relationship to the central nervous system and to the control of food intake and body weight. *American Journal of Clinical Nutrition*, 42(5): 1063–1071, 1985.
- S. C. Woods, R. J. Seeley, D. Porte Jr., and M. W. Schwartz. Signals that regulate food intake and energy homeostasis. *Science*, 280:1378–1383, 1998.
- World Health Organization. Diabetes: The cost of diabetes. *Fact Sheet No. 236*, 2002.
- Y. Wu and D. A. Pados. A feedforward bidirectional associative memory. *IEEE Transactions on Neural Networks*, 11(4):859–866, 2000.
- Y. Xia and J. Wang. A recurrent neural network for solving linear projection equations. *Neural Networks*, 13(3):337–350, 2000.
- W. Xu, S. Xia, and H. Xie. Application of CMAC-based networks on medical image classification. *Proceedings of International Symposium on Neural Networks: Advances in Neural Network. Lecture Notes in Computer Science*, 3173:953–958, 2004.
- T. Yamamoto and M. Kaneda. Intelligent controller using CMACs with self-organized structure and its application for a process system. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(5):856–860, 1999.
- M. F. Yeh and H. C. Lu. On-line adaptive quantization input space in CMAC neural network. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 4:6–11, 2002.
- L. A. Zadeh. Fuzzy sets. *Inform. Contr.*, 8:338–353, 1965.
- N. Zahid, O. Abouelala, M. Limouri, and A. Essaid. Unsupervised fuzzy clustering. *Pattern Recognition Letters*, 20:123–129, 1999.
- K. Zhang and F. Qian. Fuzzy CMAC and its application. *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, pages 944–947, 2000.

Appendices

Appendix A

Major Development Milestones in ANN Research

Table A.1: Important milestones in ANN research presented in chronological order

Hebbian Rule (Hebb, 1949)	Hebb developed the first learning rule used to update the weights neural network. The synaptic connection between two neurons intensified or strengthened if it repeatedly fires the receptive neuron.
Perceptron (Rosenblatt, 1958)	In 1958, the <i>perceptron</i> was developed to make use of the Hebbian principle to learn its weights. It is an artificial neural network consisting of only a single neuron with one or more inputs and a single output. The perceptron is able to solve <i>linearly separable</i> problems.
ADALINE (Widrow and Hoff, 1960)	Widrow and Hoff developed the ADALINE (Adaptive Linear Neuron) model in 1960. The learning rule used in the training of the ADALINE is the <i>delta-learning rule</i> or the <i>least mean square</i> (LMS) rule. This is an improved learning process over that of the perceptron as it uses the concept of negative gradient descent.
Linear Associative Memory (Anderson, 1972; Kohonen, 1972)	During the 1970s, Kohonen and Anderson independently developed the <i>linear associated memory</i> using neural networks.
CMAC (Albus, 1975a,b)	The CMAC neural network was developed in 1975 by Albus as a neural network model of the human cerebellum. The grid-like structure of the CMAC neural network is inspired by the synaptic connections between the parallel fibers and the dendrites of the Purkinje cells. The CMAC network adopts the supervised error-correction learning (i.e. modified Widrow-Hoff learning algorithm) to update its network weights.

continued on next page

continued from previous page

Kohonen network (Kohonen, 1982)	In 1982, Kohonen published his work on self-organizing networks that make use of the unsupervised learning algorithm known as <i>competitive learning</i> or the <i>winner-takes-all</i> algorithm. Later, Kohonen also introduced the concept of <i>self-organizing feature maps</i> (SOFM). These neural networks create a feature space in which other than the actual winning node, neighboring output nodes are also activated in response to the inputs.
Hopfield network (Hopfield, 1982)	In 1982, Hopfield published a paper on a model that is known as <i>Hopfield network</i> . Hopfield has proven in his paper that the stable states of the network have minimum energy levels and learning always converges. The Hopfield network has been used as memory associator and to optimize Non-polynomial (NP) hard problems such as the <i>traveling salesman problem</i> (TSP).
Boltzman machine (Ackley et al., 1985)	The <i>Boltzman machine</i> developed by Ackley <i>et al.</i> uses the Hopfield network as a backbone structure. The problem of false wells is overcome by changing the activation function of the neurons to be of stochastic nature. Training of the Boltzman machine uses a scheme known as <i>simulated annealing</i> (Kirkpatrick et al., 1983) to gradually reduce the randomness of the outputs of the neurons through the adjustment of a parameter called temperature (denoted as T).
Back-propagation (Rumelhart et al., 1986)	The discovery of the <i>back-propagation</i> learning algorithm leads to the prosperous age of neural network. The back-propagation learning algorithm extends the negative-gradient-descent concept to update the weights of multi-layered networks. The back-propagation learning algorithm removes the limitation of the perceptron and ADALINE models so that more complex multi-layered networks can be developed to solve real-world problems. Networks trained with the back-propagation learning algorithm are termed <i>back-propagation networks</i> .

Appendix B

Single Layer Model of The CMAC Network

In the original implementation of the CMAC network proposed by Albus (Albus, 1981), the network computing (memory) cells are divided into layers of uniformly quantized multi-dimensional memory array. (Please refer to Chapter 2 for the details on the layered structure of the basic CMAC network). However, the multi-layered structure of the CMAC network often renders the network operations difficult to comprehend. Moreover, in such an implementation, extensive layers of overlapping computing cells are required to produce a smooth output. The optimization of the memory allocation process (non-uniform quantization) of a multi-layered, multi-input CMAC neural network is not only tedious, but can also be computationally expensive, especially for high-dimensional input problems that require extensive layers of computing cells to achieve the desired output resolution or accuracy. This is because it is difficult to manipulate the distribution of the memory cells in the individual layers as the computation of all the overlapping layers are intertwined and tightly-coupled to produce the CMAC output. Therefore, in this Thesis, a generic single layer CMAC structure with neighborhood computations is proposed to retain the modeling principles of the original multi-layered CMAC network.

Figure B.1 illustrates such a single-layered model of a 2-input CMAC network consisting of 64 memory cells. The two dimensional computing grid corresponds to the memory space of the CMAC network. In Figure B.1, each input dimension is *quantized* into eight discrete

Figure B.1: An example of a single layer implementation of the 2D CMAC network example shown in Figure 2.5 of Chapter 2

quantization steps (or levels). Similar to the multi-layered CMAC, the input vector to this network is quantized to the corresponding level for each input dimension to obtain the index address of the winner memory cell. Smoothing of the computed output is achieved by a *neighborhood activation* of the computing cells. The activated neighborhood for the input vector (6,6) to the single-layered CMAC implementation is depicted in Figure B.1. That is, to ensure the continuity in the output surface, each input vector selects a cluster of memory cells or neurons that is centered at the winner neuron (see shaded square of Figure B.1).

The conceptual similarities between the proposed single-layered model and the original multi-layered implementation of the CMAC network can be examined from their respective modeling principles. The layered cell activations in the original CMAC network contributed to three significant computational objectives: (1) smoothing of the computed output; (2) facilitating a distributed learning paradigm; and (3) activating similar or highly correlated computing cells in the input-output (I/O) associative space. These three modeling principles are similarly conserved in the single-layered model of the CMAC network via the introduction of a neighborhood-based computational process. The activation of the neighboring cells in the input space of the single-layered CMAC corresponds to the simultaneous activation of the highly correlated cells in its multi-layered counterpart. This contributes

to the smoothing of the computed output since the neighborhood-based activation process results in continuity of the network output. In addition, the distributed learning paradigm for each input-output training pair is achieved by a neighborhood update process of the single layer model.

To retain the computational principles of the multi-layered CMAC network, a *Weighted Gaussian Neighborhood Output* (WGNO) computational process is introduced in this Thesis. The single layer CMAC model employs the WGNO computation where a set of neighborhood-bounded computing cells is activated to derive an output response to the input stimulus. This computation process is described as follows.

Let J and L denotes the total number of input and output dimensions of the CMAC network. Assume that a training dataset of $\mathbf{U} = \{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \dots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \dots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\}$ is used to train the network, where $\mathbf{X}_s = [x_{s,1} \ x_{s,2} \ \dots \ x_{s,J}]^T$ denotes the s^{th} input training vector, and $\hat{\mathbf{Y}}_s = [\hat{y}_{s,1} \ \hat{y}_{s,2} \ \dots \ \hat{y}_{s,L}]^T$ denotes the corresponding expected output target vector of the CMAC network. For each input stimulus $\mathbf{X}_s$, the computed output is derived as follows:

Step 1 Determine the region of activation

Each input stimulus $\mathbf{X}_s$ activates a neighborhood of CMAC computing cells. The size of the activated CMAC neighborhood with respect to input $\mathbf{X}_s$ is defined by $N \in [0 \dots 1]$, a user-specified parameter that governs the relative size of the neighborhood of activated CMAC cells to the overall memory space. For the input stimulus $\mathbf{X}_s$, its activation neighborhood is defined as:

$$lb_{s,j} = x_{s,j} - 0.5 \cdot N \cdot range_j \quad (\text{B.1})$$

$$rb_{s,j} = x_{s,j} + 0.5 \cdot N \cdot range_j \quad (\text{B.2})$$

$$j \in \{1, 2, \dots, J\}$$

where $lb_{s,j}$ denotes the left activation boundary, $rb_{s,j}$ denotes the right activation boundary, and $range_j$ is the domain for the j^{th} input dimension. Consequently, the memory cells encapsulated within the neighborhood defined by the computed boundaries are activated in response to the input stimulus.

Step 2 Compute the Gaussian weighting factors

A Gaussian weighting factor g_k is associated with each activated CMAC cell to

determine its contribution towards the computation of the network output. The Gaussian weighting factor is defined as:

$$g_k = (1 - d_k)e^{-d_k^2/2\gamma^2} \quad (\text{B.3})$$

where γ is the Gaussian width constant and d_k denotes the normalized Euclidean distance from the k^{th} activated cell to the input stimulus $\mathbf{X}_s$. Let $\mathbf{K}_s$ be the set of activated CMAC cells in the computed neighborhood. Subsequently, d_k is defined as

$$d_k = \frac{\|\mathbf{Q}_k - \mathbf{X}_s\|}{\max_{k' \in \mathbf{K}_s} \|\mathbf{Q}_{k'} - \mathbf{X}_s\|} \quad (\text{B.4})$$

where $\mathbf{Q}_k = [Q_{1,k}, Q_{2,k}, \dots, Q_{J,k}]$ denotes the quantization point of cell k in the memory space.

Step 3 Retrieve the CMAC output

The single layer CMAC network output $\mathbf{Y}_s$ is computed as a weighted linear combination of the memory contents of the activated cells such that:

$$\mathbf{Y}_s = \frac{\sum_{k \in \mathbf{K}_s} (g_k \cdot \mathbf{W}(k))}{\sum_{k \in \mathbf{K}_s} g_k} \quad (\text{B.5})$$

where $\mathbf{K}_s$ denotes the set of neighborhood-activated CMAC cells, and $\mathbf{W}(k)$ is the stored weight value(s) of the activated CMAC cell with index k .

Following this, the single layer CMAC model adopts a modified *Widrow-Hoff learning rule* (Widrow and Stearns, 1985) to implement a *Weighted Gaussian Neighborhood Update* (WGNU) learning process. The network update process is as follows:

Step 1 Computation of the network output

The output $\mathbf{Y}_s^{(i)}$ of the network corresponding to the input stimulus $\mathbf{X}_s$ at the i^{th} training iteration is computed based on the WGNU process described above.

Step 2 Computation of learning error

The learning error is defined as the difference between the expected output and the current output of the network. The network output error at the i^{th} training

iteration is computed as follows:

$$\mathbf{Err}_s^{(i)} = \hat{\mathbf{Y}}_s - \mathbf{Y}_s^{(i)} \quad (\text{B.6})$$

where $\mathbf{Err}_s^{(i)}$ denotes the output error of the CMAC network to the input $\mathbf{X}_s$ at the i^{th} iteration, and $\hat{\mathbf{Y}}_s$ is the desired (target) output of the CMAC network in response to the input $\mathbf{X}_s$.

Step 3 *Update of active cells*

The learning error is subsequently distributed to all of the activated cells based on their respective weighting factors.

$$\mathbf{W}^{(i+1)}(k) = \mathbf{W}^{(i)}(k) + \Delta \mathbf{W}^{(i)}(k), \quad k \in \mathbf{K}_s \quad (\text{B.7})$$

$$\Delta \mathbf{W}^{(i)}(k) = \alpha \underbrace{\frac{g_k}{\sum_{k' \in \mathbf{K}_s} g_{k'}}}_{\text{local error for cell } k} \mathbf{Err}_s^{(i)}, \quad k \in \mathbf{K}_s \quad (\text{B.8})$$

where α is the learning constant, and $\mathbf{W}^{(i)}(k)$ denotes the content (weight) of the k^{th} activated cell in the neighborhood $\mathbf{K}_s$ in CMAC in response to the input stimulus $\mathbf{X}_s$ at the i^{th} training iteration.

Appendix C

The HCAQ-CMAC Proof of Learning Convergence

This appendix provides the mathematical proof of the expression: $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{C}_a = 0$ for all $a \in \{1 \cdots S\}$, if and only if the learning constant α satisfies the condition $0 < \alpha < 2$.

Lemma C.1 *Given the definition of the matrix $\mathbf{E}_s$ as*

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T), \quad s \in \{1 \cdots S\}$$

and the activation mask $\mathbf{C}_s$ of the HCAQ-CMAC network for the s^{th} input training vector as

$$\begin{aligned} \mathbf{C}_s^T &= \underbrace{[c_{s,1} \quad c_{s,2} \quad \cdots \quad c_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \quad s \in \{1 \cdots S\} \\ c_{s,j} &= \begin{cases} 1, & \text{if the } j^{\text{th}} \text{ memory cell is activated} \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

The $\hat{M}^J \times \hat{M}^J$ matrix $\mathbf{E}_s$ has the following properties:

Property C.1.1 $\mathbf{E}_s$ is a symmetric matrix.

Proof:

$$\begin{aligned}
 \mathbf{E}_s &= \mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T \\
 \mathbf{E}_s^T &= (\mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T)^T \\
 &= \mathbf{I} - \alpha (\mathbf{C}_s \mathbf{C}_s^T)^T \\
 &= \mathbf{I} - \alpha (\mathbf{C}_s^T)^T \mathbf{C}_s^T \\
 &= \mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T \\
 &= \mathbf{E}_s
 \end{aligned}$$

Thus, $\mathbf{E}_s$ is a symmetric matrix. ■

Property C.1.2 Let matrix $\mathbf{E}_s$ be denoted as $[e_{p,q}]_{\hat{M}^J \times \hat{M}^J}$. The diagonal elements of matrix $\mathbf{E}_s$ can be expressed as

$$e_{p,p} = \begin{cases} 1, & \text{if } p^{\text{th}} \text{ element of } \mathbf{C}_s \text{ is } 0 \\ 1 - \alpha, & \text{if } p^{\text{th}} \text{ element of } \mathbf{C}_s \text{ is } 1 \end{cases} \quad p \in \{1 \cdots \hat{M}^J\}$$

and the non-diagonal elements of matrix $\mathbf{E}_s$ are always zero, i.e.

$$e_{p,q} = 0, \quad \forall p \neq q, \quad p, q \in \{1 \cdots \hat{M}^J\}$$

Proof:

According to the definition of the activation mask $\mathbf{C}_s$, as well as from the principle of the winner-take-all learning algorithm of HCAQ-CMAC, there will only be one non-zero element in $\mathbf{C}_s$ for the s^{th} input training vector. Consequently, as the matrix $\mathbf{E}_s$ is defined as:

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{C}_s \mathbf{C}_s^T), \quad s \in \{1 \cdots S\}$$

it follows that

$$e_{p,p} = \begin{cases} 1, & \text{if } p^{\text{th}} \text{ element of } \mathbf{C}_s \text{ is } 0 \\ 1 - \alpha, & \text{if } p^{\text{th}} \text{ element of } \mathbf{C}_s \text{ is } 1 \end{cases}$$

and $e_{p,q}$ is 0 for all $p \neq q$. ■

Lemma C.2 Let the matrix $\mathbf{H}$ be the multiplication result for any arbitrary $\hat{M}^J \times \hat{M}^J$ matrix $\mathbf{B}$ and the matrix $\mathbf{E}_s$ such that

$$\mathbf{H} = \mathbf{B} \mathbf{E}_s, \quad s \in \{1 \cdots S\}$$

where $\mathbf{H}$ is a $\hat{M}^J \times \hat{M}^J$ matrix. If the learning rate α satisfies the condition $0 < \alpha \leq 2$, then the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H}$ will be bounded by the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$. That is,

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 \leq \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{if } 0 < \alpha \leq 2$$

Proof:

Let the matrix $\mathbf{H}$ be denoted as $[h_{r,q}]_{\hat{M}^J \times \hat{M}^J}$ and matrix $\mathbf{B}$ be denoted as $[b_{r,q}]_{\hat{M}^J \times \hat{M}^J}$ respectively. The L_2 -norm of the r^{th} row vector of $\mathbf{B}$ is evaluated as

$$\begin{aligned} \| \text{row } r \text{ of } \mathbf{B} \|_2^2 &= \sum_{q=1}^{\hat{M}^J} (b_{r,q})^2 \\ &= (b_{r,1})^2 + (b_{r,2})^2 + \cdots + (b_{r,\hat{M}^J})^2 \end{aligned}$$

On the other hand, the L_2 -norm of the r^{th} row vector of $\mathbf{H}$ can be derived as

$$\begin{aligned} \| \text{row } r \text{ of } \mathbf{H} \|_2^2 &= \| \text{row } r \text{ of } \mathbf{B} \mathbf{E}_s \|_2^2 \\ &= \sum_{q=1}^{\hat{M}^J} (h_{r,q})^2 \\ &= \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,1} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,2} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,3} \right]^2 + \cdots + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,\hat{M}^J} \right]^2 \end{aligned}$$

From Property C.1.2, it is established that all of the non-diagonal elements $e_{p,q}$ of the matrix $\mathbf{E}_s$ evaluate as zero. Furthermore, of all the diagonal elements of the matrix $\mathbf{E}_s$, exactly *one* element is equal to $(1 - \alpha)$. Let this element be at the k^{th} position in the $\mathbf{C}_s$ activation mask, i.e. $e_{k,k} = (1 - \alpha)$. Substituting the value of $e_{k,k}$ into the L_2 -norm of the r^{th} row vector of $\mathbf{H}$ yields

$$\begin{aligned} \| \text{row } r \text{ of } \mathbf{H} \|_2^2 &= \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,1} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,2} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,3} \right]^2 + \cdots + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,\hat{M}^J} \right]^2 \\ &= (b_{r,1})^2 + (b_{r,2})^2 + \cdots + ((1 - \alpha)b_{r,k})^2 + \cdots + (b_{r,\hat{M}^J})^2 \\ &= (b_{r,1})^2 + (b_{r,2})^2 + \cdots + \underbrace{(b_{r,k})^2 + \alpha(\alpha - 2)(b_{r,k})^2}_{((1-\alpha)b_{r,k})^2} + \cdots + (b_{r,\hat{M}^J})^2 \\ &= \underbrace{(b_{r,1})^2 + (b_{r,2})^2 + \cdots + (b_{r,\hat{M}^J})^2}_{\| \text{row } r \text{ of } \mathbf{B} \|_2^2} + \alpha(\alpha - 2)(b_{r,k})^2 \\ &= \| \text{row } r \text{ of } \mathbf{B} \|_2^2 + \alpha(\alpha - 2)(b_{r,k})^2 \end{aligned}$$

For $\| \text{row } r \text{ of } \mathbf{H} \|_2^2 \leq \| \text{row } r \text{ of } \mathbf{B} \|_2^2$, the term $\alpha(\alpha - 2)(b_{r,k})^2$ has to be less than or equal to 0, i.e.

$$\begin{aligned} \alpha(\alpha - 2) \underbrace{(b_{r,k})^2}_{\text{always +ve when } b_{r,k} \neq 0} &\leq 0 \\ \alpha(\alpha - 2) &\leq 0 \\ 0 &\leq \alpha \leq 2 \end{aligned}$$

However, the condition $\alpha = 0$ signifies no learning and therefore does not apply. Hence $\| \text{row } r \text{ of } \mathbf{H} \|_2^2 \leq \| \text{row } r \text{ of } \mathbf{B} \|_2^2$ when the learning constant α is $0 < \alpha \leq 2$. ■

Lemma C.3 *Following Lemma C.2, if the learning constant α satisfies the condition $0 < \alpha < 2$ and $b_{r,k} \neq 0$, then the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H}$ will always be smaller than the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$. That is,*

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 < \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{if } 0 < \alpha < 2 \quad \text{and} \quad b_{r,k} \neq 0$$

Furthermore, given that the learning constant α satisfies the condition $0 < \alpha < 2$, the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H}$ will be equal to the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$ if and only if $b_{r,k} = 0$, where k denotes the position of the winning neuron in the activation mask $\mathbf{C}_s$, i.e.

$$\text{if } 0 < \alpha < 2, \quad \| \text{row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{iff } b_{r,k} = 0 \quad \text{and} \quad e_{k,k} = 1 - \alpha$$

Proof:

From Lemma C.2, it has been established that if the learning constant α satisfies the condition $0 < \alpha \leq 2$, then $\| \text{row } r \text{ of } \mathbf{H} \|_2^2$ will not be greater than $\| \text{row } r \text{ of } \mathbf{B} \|_2^2$. It follows that $\alpha(\alpha - 2)(b_{r,k})^2$ of Lemma C.2 is always less than zero if $0 < \alpha < 2$ and $b_{r,k} \neq 0$, where k denotes the position of the winner neuron in the activation mask $\mathbf{C}_s$. Hence,

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 < \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{if } 0 < \alpha < 2 \quad \text{and} \quad b_{r,k} \neq 0$$

Consequently, the condition of

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{row } r \text{ of } \mathbf{B} \|_2^2$$

will hold if and only if $b_{r,k} = 0$ for $e_{k,k} = 1 - \alpha$. ■

Lemma C.4 *If the learning constant α satisfies the condition $0 < \alpha < 2$, then the L_2 -norm of any arbitrary r^{th} row vector in $(\mathbf{G}_s)^i$ is bounded by the L_2 -norm of the corresponding r^{th}*

row vector in $(\mathbf{G}_s)^{i-1}$. That is,

$$\| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 \leq \| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2, \quad s \in \{1 \cdots S\}$$

Furthermore, as the training iteration i approaches infinity,

$$\lim_{i \rightarrow \infty} \| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0, \quad r \in \mathbf{K}_U$$

where $\mathbf{U}$ denotes the entire set of the training data and $\mathbf{K}_U$ is the set of indexes of the trained cells in HCAQ-CMAC due to $\mathbf{U}$.

Proof:

From the definition of the matrix $\mathbf{G}_s$ (see eq. 3.32),

$$\mathbf{G}_s = \underbrace{\mathbf{E}_{s-1} \mathbf{E}_{s-2} \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_s}_{S \text{ terms}}$$

where S is the total number of input training samples and $\mathbf{G}_s$ is a $\hat{M}^J \times \hat{M}^J$ matrix. From Property C.1.2 (Lemma C.1), $\mathbf{E}_s$ is a diagonal matrix such that

$$\mathbf{E}_s = \{e_{p,p}\} = \begin{cases} 1, & \text{if } p \neq k \\ 1 - \alpha, & \text{if } p = k \end{cases} \quad s \in \{1 \cdots S\}$$

where k is the index of the activated cell in $\mathbf{C}_s$. Hence, $\mathbf{G}_s$ is also a diagonal matrix.

From Lemma C.3, if $0 < \alpha < 2$,

$$\begin{aligned} \| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2 &\geq \| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1} \|_2^2 \\ &\geq \| \text{row } r \text{ of } ((\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1}) \mathbf{E}_{s-2} \|_2^2 \\ &\geq \cdots \\ &\geq \| \text{row } r \text{ of } ((\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1} \mathbf{E}_{s-2} \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_{s+1}) \mathbf{E}_s \|_2^2 \\ &\geq \| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \mathbf{G}_s \|_2^2 \\ &\geq \| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 \end{aligned}$$

Note that $\| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 1$ if $r \in \{1 \cdots \hat{M}^J\}$ is an untrained cell. On the other hand, if $r \in \mathbf{K}_U$, it follows from Lemma C.3 that $\| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2 > \| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2$ as $0 < \alpha < 2$ and $g_{r,k} \neq 0$. Following from above, as the training iteration i tends to infinity,

$$\lim_{i \rightarrow \infty} \| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0, \quad r \in \mathbf{K}_U$$

■

Lemma C.5 *If the learning constant α satisfies the condition $0 < \alpha < 2$, then as the training iteration i tends to infinity, the term $(\mathbf{G}_s)^i \mathbf{C}_a$ converges to a null matrix for all $a \in \{1 \cdots S\}$. That is,*

$$\text{if } 0 < \alpha < 2, \quad \lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [\mathbf{0}], \quad \forall a \in \{1 \cdots S\} \quad s \in \{1 \cdots S\}$$

Proof:

From Lemma C.4,

$$\lim_{i \rightarrow \infty} \|\text{row } r \text{ of } (\mathbf{G}_s)^i\|_2^2 = 0, \quad r \in \mathbf{K}_U$$

Hence, it follows that

$$\text{if } 0 < \alpha < 2, \quad \lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [\mathbf{0}], \quad \forall a \in \{1 \cdots S\}$$

■

Appendix D

Modified PSEC Density Clustering Algorithm

Let:

- J the total number of input dimensions,
- L the total number of output dimensions,
- $\mathbf{U}$ training dataset to the PSECMAC network where
 $\mathbf{U} = \{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \dots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \dots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\}$ and
 $\mathbf{X}_s = \begin{bmatrix} x_{s,1} & x_{s,2} & \dots & x_{s,J} \end{bmatrix}^T$ denotes the s^{th} input training vector, and
 $\hat{\mathbf{Y}}_s = \begin{bmatrix} \hat{y}_{s,1} & \hat{y}_{s,2} & \dots & \hat{y}_{s,L} \end{bmatrix}^T$ denotes the corresponding expected output target vector,
- τ the index of the MPSEC clustering iteration,
- $\mathbf{C}_j^{(\tau)}$ the set of density clusters along the j^{th} input dimension at the τ^{th} iteration where $\mathbf{C}_j^{(\tau)} = \{C_{j,1}^{(\tau)}, C_{j,2}^{(\tau)}, \dots, C_{j,n}^{(\tau)}, \dots, C_{j,n_{C,j}^{(\tau)}}^{(\tau)}\}$ and $n_{C,j}^{(\tau)}$ is the corresponding total number of density clusters,
- $\mathbf{C}_j^{(-1)}$ the initial set of density clusters for the MPSEC algorithm,
- $n_{C,j}^{(-1)}$ the initial number of (regularly-spaced) density clusters along the j^{th} input dimension.

For each input dimension $j \in \{1 \dots J\}$, the MPSEC clustering algorithm is performed as follows:

Step 1: *Initialize the clustering parameters.*

Select an initial number of density clusters $n_{C,j}^{(-1)}$, a pseudo potential threshold β , a clustering termination criterion ε and a maximum number of clustering iterations τ_{max} . Initialize the LVQ learning constant $\alpha_c = 1/S$, where S is the total number of training samples.

Step 2: *Construct the initial set of density clusters.*

Construct the initial set of density clusters $\mathbf{C}_j^{(-1)}$ with $n_{C,j}^{(-1)}$ regularly spaced clusters. Let $C_{j,n}^{(\tau)}$ denotes the n^{th} density cluster in the j^{th} input dimension at the τ^{th} clustering iteration. Each density cluster is associated with a cluster center $P_{j,n}^{(\tau)}$ and a density value $V_{j,n}^{(\tau)}$. In the initial set of density clusters $\mathbf{C}_j^{(-1)} = \{C_{j,1}^{(-1)}, C_{j,2}^{(-1)}, \dots, C_{j,n}^{(-1)}, \dots, C_{j,n_{C,j}^{(-1)}}^{(-1)}\}$, the clusters are initialized such that:

$$\begin{aligned} P_{j,n}^{(-1)} &= \min_j + (n - 0.5)(gap_j) \\ V_{j,n}^{(-1)} &= 0 \end{aligned}$$

where $n \in \{1 \dots n_{C,j}^{(-1)}\}$ denotes the cluster index, $\min_j$ and $\max_j$ are the minimum and maximum values of the training samples along the j^{th} input dimension respectively, while gap_j denotes the equal-width partitioning of the density clusters computed as:

$$\begin{aligned} \min_j &= \min_s(x_{s,j}), & s \in \{1 \dots S\} \\ \max_j &= \max_s(x_{s,j}), & s \in \{1 \dots S\} \\ gap_j &= \frac{\max_j - \min_j}{n_{C,j}^{(-1)}} \end{aligned}$$

This step emulates the formation of the initial brain system, in which extraneous connection patterns emerge as a result of the overproduction of neurons during the prenatal brain development phase.

Step 3: *Compute the initial cluster density values.*

Perform a one-pass learning of the density values of the initial set of density clusters. For each training data point $x_{s,j}$, where $s \in \{1 \dots S\}$:

(a) Find the index to the winner cluster $n_{w,j}$ where:

$$n_{w,j} = \left\lfloor \frac{x_{s,j} - \min_j}{gap_j} \right\rfloor + 1$$

(b) Compute the distance $d_{w,j}$ between the data point $x_{s,j}$ and the winner cluster

$C_{j,n_{w,j}}^{(-1)}$ such that:

$$d_{w,j} = x_{s,j} - P_{j,n_{w,j}}^{(-1)}$$

(c) Determine the index of the runner-up cluster $n_{r,j}$ such that:

$$n_{r,j} = \begin{cases} n_{w,j} - 1 & \text{if } d_{w,j} < 0 \text{ and } (n_{w,j} - 1) \geq 1 \\ \emptyset & \text{if } d_{w,j} = 0 \text{ or } (n_{w,j} - 1) < 1 \text{ or } (n_{w,j} + 1) > n_{C,j}^{(-1)} \\ n_{w,j} + 1 & \text{if } d_{w,j} > 0 \text{ and } (n_{w,j} + 1) \leq n_{C,j}^{(-1)} \end{cases}$$

where $n_{r,j} = \emptyset$ represents a condition where there is no runner-up cluster. This occurs when the data point falls exactly at the cluster center of the winner cluster, or when the index of the runner-up cluster $n_{r,j}$ falls outside the range of $\{1 \dots n_{C,j}^{(-1)}\}$.

(d) Update the density value $V_{j,n}^{(-1)}$ of the winner and runner-up clusters as follows:

$$\begin{aligned} V_{j,n_{w,j}}^{(-1)} &= \begin{cases} V_{j,n_{w,j}}^{(-1)} + \left(1 - \frac{|d_{w,j}|}{gap_j}\right) \alpha_c & \text{if } n_{r,j} \neq \emptyset \\ V_{j,n_{w,j}}^{(-1)} + \alpha_c & \text{if } n_{r,j} = \emptyset \end{cases} \\ V_{j,n_{r,j}}^{(-1)} &= V_{j,n_{r,j}}^{(-1)} + \left(\frac{|d_{w,j}|}{gap_j}\right) \alpha_c \quad \text{if } n_{r,j} \neq \emptyset \end{aligned}$$

Step 4: *Evolve the initial set of density clusters.*

For each input dimension, identify all the local maxima in the set of computed density values to capture the inherent data density profile. This step emulates the competitive neuronal selection process in human brain development, whereby neurons with high tropic factors are identified as the winner neurons and the remaining extraneous neurons are pruned to create a more refined structure of synaptic connections. Let $C_j^{(0)}$ denotes the resultant set of density clusters, and idx and δ be two temporary variables. Initialize $C_j^{(0)} = \emptyset$, $n_{C,j}^{(0)} = 0$, $idx = 0$ and $\delta = 0$. Based on the initial set of density clusters $C_j^{(-1)}$, the cluster selection process is defined as follows:

For $n = 1 \dots n_{C,j}^{(-1)}$:

- IF $idx = 0$ and $V_{j,n}^{(-1)} > (\delta + \beta)$ THEN update the values of the temporary variables idx and δ such that:

$$idx = n, \quad \delta = V_{j,n}^{(-1)}$$

In this case, idx corresponds to the cluster index associated with the current local maxima δ in the set of computed density values, and β is the pre-defined threshold parameter that governs if a density cluster $C_{j,n}^{(-1)}$ is identified as a new

local maxima.

- ELSE IF $idx \neq 0$ and $V_{j,n}^{(-1)} < (\delta - \beta)$ THEN include the current local maxima into the new set of density clusters $\mathbf{C}_j^{(0)}$ and re-initialize idx and δ . That is,

$$C_{j,idx}^{(-1)} \in \mathbf{C}_j^{(0)}, \quad n_{C,j}^{(0)} = n_{C,j}^{(0)} + 1, \quad idx = 0, \quad \delta = V_{j,n}^{(-1)}$$

This step identifies the characteristic drop in the density values of the density clusters which marks a local minima in the density profile.

- ELSE IF $idx = 0$ and $V_{j,n}^{(-1)} < \delta$ THEN update the lowest density value detected after the previous local maxima such that

$$\delta = V_{j,n}^{(-1)}$$

End.

If by the end of the iteration (i.e. $n = n_{C,j}^{(-1)}$) and $idx \neq 0$ then the last identified local maxima $C_{j,idx}^{(-1)}$ is included into the new set of density clusters. That is,

$$C_{j,idx}^{(-1)} \in \mathbf{C}_j^{(0)}, \quad n_{C,j}^{(0)} = n_{C,j}^{(0)} + 1$$

Step 5: *Incremental learning of the cluster centers via LVQ.*

Iteratively refine the cluster centers $P_{j,n}^{(0)}$ of the new set of density clusters $\mathbf{C}_j^{(0)}$ using the LVQ algorithm. Initialize the LVQ learning error $e^{(0)} = 0$ and the incremental learning of the cluster centers $P_{j,n}^{(0)}$ is depicted as follows:

For $\tau = 1 \cdots \tau_{max}$:

- Initialize the learning error for the current LVQ iteration $e^{(\tau)} = 0$ and the cluster set $\mathbf{C}_j^{(\tau)} = \mathbf{C}_j^{(\tau-1)}$.
- For $s = 1 \cdots S$:
 - Find the index of the winner cluster $n_{w,j}$ such that:

$$\|x_{s,j} - P_{j,n_{w,j}}^{(\tau)}\| = \min_n \left(\|x_{s,j} - P_{j,n}^{(\tau)}\| \right)$$

where $n \in \{1 \cdots n_{C,j}^{(\tau)}\}$ and $n_{C,j}^{(\tau)}$ is the total number of density clusters in $\mathbf{C}_j^{(\tau)}$.

- Update the winner cluster such that:

$$P_{j,n_{w,j}}^{(\tau)} = P_{j,n_{w,j}}^{(\tau)} + \alpha_c (x_{s,j} - P_{j,n_{w,j}}^{(\tau)})$$

(iii) Update LVQ learning error such that:

$$e^{(\tau)} = e^{(\tau)} + \|x_{s,j} - P_{j,n_{w,j}}^{(\tau)}\|$$

(c) Compute the delta change in the LVQ learning error:

$$\Delta e^{(\tau)} = e^{(\tau)} - e^{(\tau-1)}$$

(d) Compare $\Delta e^{(\tau)}$ to the terminating condition. If $\Delta e^{(\tau)} \leq \varepsilon$, terminate the LVQ learning iteration (i.e. $\tau_{end} = \tau$).

Step 6: *Compute the resultant density profile.*

Perform a one-pass learning of the density values of each cluster in the final set of density clusters $\mathbf{C}_j^{(\tau_{end})}$. Initialize $V_{j,n}^{(\tau_{end})} = 0$, $\forall n \in \{1 \dots n_{C,j}^{(\tau_{end})}\}$. For each training data point $x_{s,j}$ where $s \in \{1 \dots S\}$:

(a) Find the index to the winner cluster $n_{w,j}$ where:

$$n_{w,j} = \arg \min_n \|x_{s,j} - P_{j,n}^{(\tau_{end})}\|$$

(b) Compute the distance $d_{w,j}$ between the data point $x_{s,j}$ and the winner cluster $C_{j,n_{w,j}}^{(\tau_{end})}$ using:

$$d_{w,j} = x_{s,j} - P_{j,n_{w,j}}^{(\tau_{end})}$$

(c) Determine the index of the runner-up cluster $n_{r,j}$ as follows:

$$n_{r,j} = \begin{cases} n_{w,j} - 1 & \text{if } d_{w,j} < 0 \text{ and } (n_{w,j} - 1) \geq 1 \\ \emptyset & \text{if } d_{w,j} = 0 \text{ or } (n_{w,j} - 1) < 1 \text{ or } (n_{w,j} + 1) > n_{C,j}^{(\tau_{end})} \\ n_{w,j} + 1 & \text{if } d_{w,j} > 0 \text{ and } (n_{w,j} + 1) \leq n_{C,j}^{(\tau_{end})} \end{cases}$$

where $n_{r,j} = \emptyset$ represents a condition where there is no runner-up cluster.

(d) Update the density value $V_{j,n}^{(\tau_{end})}$ of the winner and runner-up clusters as follows:

$$V_{j,n_{w,j}}^{(\tau_{end})} = \begin{cases} V_{j,n_{w,j}}^{(\tau_{end})} + \left(1 - \frac{|d_{w,j}|}{\|P_{j,n_{w,j}}^{(\tau_{end})} - P_{j,n_{r,j}}^{(\tau_{end})}\|}\right) \alpha_c & \text{if } n_{r,j} \neq \emptyset \\ V_{j,n_{w,j}}^{(\tau_{end})} + \alpha_c & \text{if } n_{r,j} = \emptyset \end{cases}$$

$$V_{j,n_{r,j}}^{(\tau_{end})} = V_{j,n_{r,j}}^{(\tau_{end})} + \left(\frac{|d_{w,j}|}{\|P_{j,n_{w,j}}^{(\tau_{end})} - P_{j,n_{r,j}}^{(\tau_{end})}\|}\right) \alpha_c \quad \text{if } n_{r,j} \neq \emptyset$$

where $n_{C,j}^{(\tau_{end})}$ denotes the total number of density clusters in $\mathbf{C}_j^{(\tau_{end})}$. Subsequently, for each density cluster $C_{j,n}^{(\tau_{end})}$ in $\mathbf{C}_j^{(\tau_{end})}$, define the respective left and right boundary

$L_{j,n}^{(\tau_{end})}$ and $R_{j,n}^{(\tau_{end})}$ as:

$$L_{j,n}^{(\tau_{end})} = \begin{cases} \min_j & \text{if } n = 1 \\ \frac{P_{j,n}^{(\tau_{end})} + P_{j,n-1}^{(\tau_{end})}}{2} & \text{otherwise} \end{cases}$$

$$R_{j,n}^{(\tau_{end})} = \begin{cases} \max_j & \text{if } n = n_{C,j}^{(\tau_{end})} \\ \frac{P_{j,n+1}^{(\tau_{end})} + P_{j,n}^{(\tau_{end})}}{2} & \text{otherwise} \end{cases}$$

Figure D.1 depicts a flowchart summarizing the MPSEC algorithm described above. The figure also includes an outline of the computational complexity analysis of the density clustering algorithm. As shown in Figure D.1, the worst-case complexity of MPSEC clustering is in the order of $O(n_{c,j}^{(-1)}) + O(S\tau_{\max})$ where S denotes the total number of training samples to be clustered; $\tau_{\max}$ is the maximum number of LVQ iteration; and $n_{c,j}^{(-1)}$ is the initial number of density clusters of the MPSEC clustering algorithm.

Figure D.1: The flowchart and computational complexity of the MPSEC algorithm: $O(\cdot)$ – the Big-O notation; S – number of training samples; τ_{max} – maximum number of LVQ iteration; $n_{c,j}^{(-1)}$ – initial number of density clusters

Appendix E

The PSECMAC Proof of Learning Convergence

This section provides the mathematical proof of the expression: $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{A}_v = [0]_{\hat{M}^J \times 1}$ for all $v \in \{1 \cdots S\}$, when the learning constant α satisfies the condition $0 < \alpha \leq 2$. The proof presented in this Appendix follows closely that of the CMAC learning convergence presented in (Lin and Chiang, 1997). A similar proof for the fuzzy CMAC model has also been reported in (Ting, 2004).

Lemma E.1 *Given the definition of the matrix $\mathbf{E}_s$ as*

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T), \quad s \in \{1 \cdots S\}$$

and the activation mask $\mathbf{A}_s$ of the PSECMAC network for the s^{th} input training sample as

$$\mathbf{A}_s^T = \underbrace{[a_{s,1} \quad a_{s,2} \quad \cdots \quad a_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}}, \quad s \in \{1 \cdots S\}$$

such that

$$\begin{aligned} a_{s,j} &= \frac{g_j}{\sum_{k \in \mathbf{K}_s} g_k}, \quad j \in \{1 \cdots \hat{M}^J\} \quad \text{and} \\ g_j &= 0, \quad \text{if } j \notin \mathbf{K}_s \end{aligned}$$

where g_k the Gaussian weighting factor of the k^{th} activated memory (computing) cells, and $\mathbf{K}_s$ the set of activated computing cells corresponding to the s^{th} input training vector.

The $\hat{M}^J \times \hat{M}^J$ matrix $\mathbf{E}_s$ has the following properties:

Property E.1.1 $\mathbf{E}_s$ is a symmetric matrix.

Proof:

$$\begin{aligned}
 \mathbf{E}_s &= \mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T \\
 \mathbf{E}_s^T &= (\mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T)^T \\
 &= \mathbf{I} - \alpha (\mathbf{A}_s \mathbf{A}_s^T)^T \\
 &= \mathbf{I} - \alpha (\mathbf{A}_s^T)^T \mathbf{A}_s^T \\
 &= \mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T \\
 &= \mathbf{E}_s
 \end{aligned}$$

Thus, $\mathbf{E}_s$ is a symmetric matrix. ■

Property E.1.2 Let matrix $\mathbf{E}_s$ be denoted as $[e_{p,q}]_{\hat{M}^J \times \hat{M}^J}$. The diagonal elements of matrix $\mathbf{E}_s$ can be expressed as

$$e_{p,p} = 1 - \left[\alpha \frac{(g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right], \quad p \in \{1 \dots \hat{M}^J\}$$

and the non-diagonal elements of matrix $\mathbf{E}_s$ can be expressed as

$$e_{p,q} = -\alpha \frac{g_p g_q}{(\sum_{k \in \mathbf{K}_s} g_k)^2}, \quad p, q \in \{1 \dots \hat{M}^J\} \quad \text{where } p \neq q.$$

Proof:

Let $a_{s,j}$ be the j^{th} element of $\mathbf{A}_s$ and the matrix $\mathbf{F}_s = \mathbf{A}_s \mathbf{A}_s^T$.

From the definition of vector $\mathbf{A}_s$,

$$a_{s,j} = \frac{g_j}{\sum_{k \in \mathbf{K}_s} g_k}$$

The elements $f_{p,q}$ of the matrix $\mathbf{F}_s$ can be expressed as

$$\begin{aligned}
 f_{p,q} &= a_{s,p} \times a_{s,q} \\
 &= \left(\frac{g_p}{\sum_{k \in \mathbf{K}_s} g_k} \right) \left(\frac{g_q}{\sum_{k \in \mathbf{K}_s} g_k} \right) \\
 &= \frac{g_p g_q}{\left(\sum_{k \in \mathbf{K}_s} g_k \right)^2}
 \end{aligned}$$

The elements of the matrix $\mathbf{E}_s$ can thus be derived as

$$\begin{aligned}
 e_{p,p} &= 1 - \alpha f_{p,p} \\
 &= 1 - \left[\alpha \frac{(g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right]
 \end{aligned}$$

$$\begin{aligned}
e_{p,q} &= 0 - \alpha f_{p,q} \\
&= -\alpha \frac{g_p g_q}{(\sum_{k \in \mathbf{K}_s} g_k)^2}
\end{aligned}$$

■

Property E.1.3 Let E_{p_1} and E_{p_2} be any two arbitrarily chosen row vectors of $\mathbf{E}_s$, where $p_1 \neq p_2$. The dot product of these two vectors can be expressed as

$$E_{p_1} \cdot E_{p_2}^T = \sum_{q=1}^{\hat{M}^J} e_{p_1,q} e_{p_2,q} = \left[\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] (g_{p_1} g_{p_2})$$

Proof:

$$\sum_{q=1}^{\hat{M}^J} e_{p_1,q} e_{p_2,q} = \sum_{q=1; q \neq p_1; q \neq p_2}^{\hat{M}^J} (e_{p_1,q} e_{p_2,q}) + \underbrace{e_{p_1,p_1} e_{p_2,p_1}}_{q=p_1} + \underbrace{e_{p_1,p_2} e_{p_2,p_2}}_{q=p_2}$$

By Property E.1.2:

$$\begin{aligned}
\sum_{q=1}^{\hat{M}^J} e_{p_1,q} e_{p_2,q} &= \left[\sum_{q=1; q \neq p_1; q \neq p_2}^{\hat{M}^J} \left(\frac{-\alpha g_q g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \left(\frac{-\alpha g_q g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \right] \\
&\quad + \underbrace{\left[1 - \frac{\alpha g_{p_1} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] \left[-\alpha \frac{g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right]}_{q=p_1} \\
&\quad + \underbrace{\left[1 - \frac{\alpha g_{p_2} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] \left[-\alpha \frac{g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right]}_{q=p_2} \\
&= \left[\sum_{q=1; q \neq p_1; q \neq p_2}^{\hat{M}^J} \frac{\alpha^2 (g_q)^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right] + \left[\frac{-\alpha g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] + \left[\frac{\alpha^2 (g_{p_1})^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right] \\
&\quad + \left[\frac{-\alpha g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] + \left[\frac{\alpha^2 (g_{p_2})^2 g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right] \\
&= \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1; q \neq p_1; q \neq p_2}^{\hat{M}^J} (g_q)^2 \right] + \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} (g_{p_1})^2 \right] \\
&\quad + \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} (g_{p_2})^2 \right] - \left[2 \frac{\alpha g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] \\
&= \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1}^{\hat{M}^J} (g_q)^2 \right] - \left[2 \frac{\alpha g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right]
\end{aligned}$$

For each input training sample s , the Gaussian weighting factor $g_q = 0$ if $q \notin \mathbf{K}_s$ for all $q \in \{1 \cdots \hat{M}^J\}$. Therefore, following from above, the dot product for E_{p_1} and E_{p_2} can be

re-expressed as

$$\begin{aligned}
 E_{p_1} \cdot E_{p_2}^T &= \sum_{q=1}^{\hat{M}^J} e_{p_1,q} e_{p_2,q} = \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1}^{\hat{M}^J} (g_q)^2 \right] - \left[2 \frac{\alpha g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] \\
 &= \left[\frac{\alpha^2 g_{p_1} g_{p_2}}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=k \in \mathbf{K}_s} (g_q)^2 \right] - \left[2 \frac{\alpha g_{p_2} g_{p_1}}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] \\
 &= \left[\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right] (g_{p_1} g_{p_2})
 \end{aligned}$$

■

Property E.1.4 The L_2 -norm of any row vector E_p of the matrix $\mathbf{E}_s$ can be expressed as

$$\begin{aligned}
 \|\text{row } p \text{ of } \mathbf{E}_s\|_2^2 &= \|E_p\|_2^2 \\
 &= \sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 \\
 &= 1 + \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) (g_p)^2
 \end{aligned}$$

Proof:

$$\sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 = \sum_{q=1; q \neq p}^{\hat{M}^J} e_{p,q} e_{p,q} + \underbrace{e_{p,p} e_{p,p}}_{q=p}$$

By Property E.1.2:

$$\begin{aligned}
 \sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 &= \left[\sum_{q=1; q \neq p}^{\hat{M}^J} \left(\frac{-\alpha g_p g_q}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \left(\frac{-\alpha g_p g_q}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \right] \\
 &\quad + \underbrace{\left[\left(1 - \frac{\alpha g_p g_p}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \left(1 - \frac{\alpha g_p g_p}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \right]}_{q=p} \\
 &= \left[\sum_{q=1; q \neq p}^{\hat{M}^J} \frac{\alpha^2 (g_p)^2 (g_q)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right] + \left[1 - \frac{2\alpha (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} + \frac{\alpha^2 (g_p)^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right] \\
 &= 1 + \left[\frac{\alpha^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1; q \neq p}^{\hat{M}^J} (g_q)^2 \right] + \left[\frac{\alpha^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} (g_p)^2 \right] - \frac{2\alpha (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \\
 &= 1 + \left[\frac{\alpha^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1}^{\hat{M}^J} (g_q)^2 \right] - \frac{2\alpha (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2}
 \end{aligned}$$

For each input training sample s , the Gaussian weighting factor $g_q = 0$ if $q \notin \mathbf{K}_s$ for all $q \in \{1 \cdots \hat{M}^J\}$. Therefore, following from above, the L_2 -norm for E_p can be re-expressed as

$$\begin{aligned} \|E_p\|_2^2 &= \sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 = 1 + \left[\frac{\alpha^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{q=1}^{\hat{M}^J} (g_q)^2 \right] - \frac{2\alpha (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \\ &= 1 + \left[\frac{\alpha^2 (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \sum_{k \in \mathbf{K}_s} (g_k)^2 \right] - \frac{2\alpha (g_p)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \\ &= 1 + \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) (g_p)^2 \end{aligned}$$

■

Lemma E.2 Let the matrix $\mathbf{H}$ be the multiplication result for any arbitrary $\hat{M}^J \times \hat{M}^J$ matrix $\mathbf{B}$ and the matrix $\mathbf{E}_s$ such that

$$\mathbf{H} = \mathbf{B}\mathbf{E}_s, \quad s \in \{1 \cdots S\}$$

where $\mathbf{H}$ is a $\hat{M}^J \times \hat{M}^J$ matrix. If the learning rate α satisfies the condition $0 < \alpha \leq 2$, then the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H}$ will be bounded by the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$. That is,

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 \leq \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{if } 0 < \alpha \leq 2$$

Proof:

Let the matrix $\mathbf{H}$ be denoted as $[h_{r,q}]_{\hat{M}^J \times \hat{M}^J}$ and matrix $\mathbf{B}$ be denoted as $[b_{r,q}]_{\hat{M}^J \times \hat{M}^J}$ respectively. The L_2 -norm of the r^{th} row vector of $\mathbf{H}$ can be derived as

$$\begin{aligned} \| \text{row } r \text{ of } \mathbf{H} \|_2^2 &= \| \text{row } r \text{ of } \mathbf{B}\mathbf{E}_s \|_2^2 \\ &= \sum_{q=1}^{\hat{M}^J} (h_{r,q})^2 \\ &= \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,1} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,2} \right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,3} \right]^2 \\ &\quad + \cdots + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q} e_{q,\hat{M}^J} \right]^2 \end{aligned}$$

$$\begin{aligned}
&= (b_{r,1})^2(e_{1,1})^2 + (b_{r,2})^2(e_{2,1})^2 + (b_{r,3})^2(e_{3,1})^2 + \cdots + (b_{r,\hat{M}^J})^2(e_{\hat{M}^J,1})^2 \\
&\quad + 2 \left\{ \begin{aligned} &b_{r,1}e_{1,1}b_{r,2}e_{2,1} + b_{r,1}e_{1,1}b_{r,3}e_{3,1} + \cdots + b_{r,1}e_{1,1}b_{r,\hat{M}^J}e_{\hat{M}^J,1} \\ &\quad + b_{r,2}e_{2,1}b_{r,3}e_{3,1} + \cdots + b_{r,2}e_{2,1}b_{r,\hat{M}^J}e_{\hat{M}^J,1} \\ &\quad \quad \quad \vdots \\ &\quad \quad \quad + b_{r,(\hat{M}^J-1)}e_{(\hat{M}^J-1),1}b_{r,\hat{M}^J}e_{\hat{M}^J,1} \end{aligned} \right\} \\
&+ (b_{r,1})^2(e_{1,2})^2 + (b_{r,2})^2(e_{2,2})^2 + (b_{r,3})^2(e_{3,2})^2 + \cdots + (b_{r,\hat{M}^J})^2(e_{\hat{M}^J,2})^2 \\
&\quad + 2 \left\{ \begin{aligned} &b_{r,1}e_{1,2}b_{r,2}e_{2,2} + b_{r,1}e_{1,2}b_{r,3}e_{3,2} + \cdots + b_{r,1}e_{1,2}b_{r,\hat{M}^J}e_{\hat{M}^J,2} \\ &\quad + b_{r,2}e_{2,2}b_{r,3}e_{3,2} + \cdots + b_{r,2}e_{2,2}b_{r,\hat{M}^J}e_{\hat{M}^J,2} \\ &\quad \quad \quad \vdots \\ &\quad \quad \quad + b_{r,(\hat{M}^J-1)}e_{(\hat{M}^J-1),2}b_{r,\hat{M}^J}e_{\hat{M}^J,2} \end{aligned} \right\} \\
&+ \cdots \\
&+ (b_{r,1})^2(e_{1,\hat{M}^J})^2 + (b_{r,2})^2(e_{2,\hat{M}^J})^2 + (b_{r,3})^2(e_{3,\hat{M}^J})^2 + \cdots + (b_{r,\hat{M}^J})^2(e_{\hat{M}^J,\hat{M}^J})^2 \\
&\quad + 2 \left\{ \begin{aligned} &b_{r,1}e_{1,\hat{M}^J}b_{r,2}e_{2,\hat{M}^J} + b_{r,1}e_{1,\hat{M}^J}b_{r,3}e_{3,\hat{M}^J} + \cdots + b_{r,1}e_{1,\hat{M}^J}b_{r,\hat{M}^J}e_{\hat{M}^J,\hat{M}^J} \\ &\quad + b_{r,2}e_{2,\hat{M}^J}b_{r,3}e_{3,\hat{M}^J} + \cdots + b_{r,2}e_{2,\hat{M}^J}b_{r,\hat{M}^J}e_{\hat{M}^J,\hat{M}^J} \\ &\quad \quad \quad \vdots \\ &\quad \quad \quad + b_{r,(\hat{M}^J-1)}e_{(\hat{M}^J-1),\hat{M}^J}b_{r,\hat{M}^J}e_{\hat{M}^J,\hat{M}^J} \end{aligned} \right\} \\
&= (b_{r,1})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{1,q})^2 \right] + (b_{r,2})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{2,q})^2 \right] + (b_{r,3})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{3,q})^2 \right] + \cdots \\
&\quad + (b_{r,\hat{M}^J})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{\hat{M}^J,q})^2 \right] \\
&+ 2 \left\{ \begin{aligned} &b_{r,1}b_{r,2}(e_{1,1}e_{2,1} + \cdots + e_{1,\hat{M}^J}e_{2,\hat{M}^J}) + \cdots + b_{r,1}b_{r,\hat{M}^J}(e_{1,1}e_{\hat{M}^J,1} + \cdots + e_{1,\hat{M}^J}e_{\hat{M}^J,\hat{M}^J}) \\ &\quad + b_{r,2}b_{r,3}(e_{2,1}e_{3,1} + \cdots + e_{2,\hat{M}^J}e_{3,\hat{M}^J}) + \cdots + b_{r,2}b_{r,\hat{M}^J}(e_{2,1}e_{\hat{M}^J,1} + \cdots + e_{2,\hat{M}^J}e_{\hat{M}^J,\hat{M}^J}) \\ &\quad \quad \quad \vdots \\ &\quad \quad \quad + b_{r,(\hat{M}^J-1)}b_{r,\hat{M}^J}(e_{(\hat{M}^J-1),1}e_{\hat{M}^J,1} + \cdots + e_{(\hat{M}^J-1),\hat{M}^J}e_{\hat{M}^J,\hat{M}^J}) \end{aligned} \right\}
\end{aligned}$$

Thus:

$$\begin{aligned}
\| \text{row } r \text{ of } \mathbf{H} \|_2^2 &= \underbrace{\sum_{p=1}^{\hat{M}^J} \left\{ (b_{r,p})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 \right] \right\}}_{\text{(I)}} \\
&\quad + 2 \underbrace{\sum_{p_1=1}^{\hat{M}^J-1} \left\{ b_{r,p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} \left[b_{r,p_2} \sum_{q=1}^{\hat{M}^J} (e_{p_1,q}e_{p_2,q}) \right] \right\}}_{\text{(II)}}
\end{aligned}$$

Solving for (I) and (II) separately yields:

$$\begin{aligned}
 \underbrace{\sum_{p=1}^{\hat{M}^J} \left\{ (b_{r,p})^2 \left[\sum_{q=1}^{\hat{M}^J} (e_{p,q})^2 \right] \right\}}_{\text{(I)}} &= \overbrace{\sum_{p=1}^{\hat{M}^J} \left\{ (b_{r,p})^2 \left[1 + \underbrace{\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) (g_p)^2 \right]}_{\sum_{q=1}^{\hat{M}^J} (e_{p,q})^2} \right\}}^{\text{(Substituting Property E.1.4 of Lemma E.1)}} \\
 &= \underbrace{\sum_{p=1}^{\hat{M}^J} (b_{r,p})^2 + \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \sum_{p=1}^{\hat{M}^J} \{ (b_{r,p})^2 (g_p)^2 \}}_{\text{(III)}}
 \end{aligned}$$

$$\underbrace{2 \sum_{p_1=1}^{\hat{M}^J-1} \left\{ b_{r,p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} \left[b_{r,p_2} \sum_{q=1}^{\hat{M}^J} (e_{p_1,q} e_{p_2,q}) \right] \right\}}_{\text{(II)}}$$

$$\begin{aligned}
 &\overbrace{\left(\text{Substituting Property E.1.3 of Lemma E.1} \right)} \\
 &= 2 \sum_{p_1=1}^{\hat{M}^J-1} \left\{ b_{r,p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} \left[b_{r,p_2} \underbrace{\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) (g_{p_1} g_{p_2})}_{\sum_{q=1}^{\hat{M}^J} (e_{p_1,q} e_{p_2,q})} \right] \right\} \\
 &= \underbrace{2 \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \sum_{p_1=1}^{\hat{M}^J-1} b_{r,p_1} g_{p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} b_{r,p_2} g_{p_2}}_{\text{(IV)}}
 \end{aligned}$$

By substituting (I) with (III) and (II) with (IV), the L_2 -norm of the r^{th} row vector of $\mathbf{H}$ can be re-expressed as

$$\begin{aligned}
 \|\text{row } r \text{ of } \mathbf{H}\|_2^2 &= \underbrace{\sum_{p=1}^{\hat{M}^J} (b_{r,p})^2}_{\|\text{row } r \text{ of } \mathbf{B}\|_2^2} + \underbrace{\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \sum_{p=1}^{\hat{M}^J} \{ (b_{r,p})^2 (g_p)^2 \}}_{\text{(III)}} \\
 &\quad + \underbrace{2 \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \sum_{p_1=1}^{\hat{M}^J-1} b_{r,p_1} g_{p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} b_{r,p_2} g_{p_2}}_{\text{(IV)}}
 \end{aligned}$$

$$\begin{aligned}
&= \|\text{row } r \text{ of } \mathbf{B}\|_2^2 + \left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \times \\
&\quad \underbrace{\left(\sum_{p=1}^{\hat{M}^J} \{ (b_{r,p})^2 (g_p)^2 \} + 2 \sum_{p_1=1}^{\hat{M}^J-1} b_{r,p_1} g_{p_1} \sum_{p_2=p_1+1}^{\hat{M}^J} b_{r,p_2} g_{p_2} \right)}_{\left(\sum_{p=1}^{\hat{M}^J} (b_{r,p} g_p) \right)^2} \\
&= \|\text{row } r \text{ of } \mathbf{B}\|_2^2 + \underbrace{\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \left(\sum_{p=1}^{\hat{M}^J} (b_{r,p} g_p) \right)^2}_{(V)}
\end{aligned}$$

For $\|\text{row } r \text{ of } \mathbf{H}\|_2^2 \leq \|\text{row } r \text{ of } \mathbf{B}\|_2^2$, term (V) ≤ 0 . That is,

$$\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \underbrace{\left(\sum_{p=1}^{\hat{M}^J} (b_{r,p} g_p) \right)^2}_{(IV) \geq 0} \leq 0$$

If (IV) = 0, the learning constant α is unbounded. However, since we want to find a bounded condition for α to guarantee learning convergence, we assume that (IV) > 0 . (IV) > 0 if and only if $\exists p$ s.t. $b_{r,p} \neq 0$ when $g_p \neq 0$, where $p \in \mathbf{K}_s$. Hence, the following expression results.

$$\begin{aligned}
\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} &\leq 0 \\
\alpha^2 \frac{\sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} - 2\alpha &\leq 0
\end{aligned}$$

Due to the neighborhood activation of computing cells in the proposed PSECMAC network, the cardinality of $\mathbf{K}_s$ is always greater than one while the Gaussian weighting function $g_k \in (0, 1], \forall k \in \mathbf{K}_s$. That is,

$$\begin{aligned}
|\mathbf{K}_s| &> 1 \quad \text{and} \\
g_k &\in (0, 1] \quad \text{for } \forall k \in \mathbf{K}_s
\end{aligned}$$

Numerically, when two real numbers $\mathcal{A}$ and $\mathcal{B}$ are bounded between 0 and 1, i.e. $0 \leq \mathcal{A}, \mathcal{B} \leq 1$, the triangular inequality $(\mathcal{A} + \mathcal{B})^2 \geq \mathcal{A}^2 + \mathcal{B}^2$ implies that

$$\frac{\mathcal{A}^2 + \mathcal{B}^2}{(\mathcal{A} + \mathcal{B})^2} \leq 1$$

Therefore,

$$\frac{\sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \in (0, 1)$$

Hence, by taking the upper limit of unity, the bounded condition of the learning constant α becomes

$$\begin{aligned}\alpha^2 - 2\alpha &\leq 0 \\ \Rightarrow \alpha(\alpha - 2) &\leq 0 \\ \Rightarrow 0 &\leq \alpha \leq 2\end{aligned}$$

However, the condition $\alpha = 0$ signifies no learning in PSECMAC and therefore it does not apply in the proof of learning convergence. Hence, the bounded condition of the learning constant α is $0 < \alpha \leq 2$ such that $\|\text{row } r \text{ of } \mathbf{H}\|_2^2 \leq \|\text{row } r \text{ of } \mathbf{B}\|_2^2$. ■

Lemma E.3 *If the learning constant α satisfies the condition $0 < \alpha \leq 2$, then the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H} = \mathbf{B}\mathbf{E}_s$ will always be smaller than the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$, provided that this vector in $\mathbf{B}$ is not null at the elemental position corresponding to the activated computing cells in $\mathbf{E}_s$. That is,*

$$\|\text{row } r \text{ of } \mathbf{H}\|_2^2 < \|\text{row } r \text{ of } \mathbf{B}\|_2^2 \quad \text{if } 0 < \alpha \leq 2 \quad \text{and } \exists p \in \mathbf{K}_s \text{ s.t. } b_{r,p} \neq 0$$

Furthermore, given that the learning constant α satisfies the condition $0 < \alpha \leq 2$, the L_2 -norm of any arbitrary r^{th} row vector in $\mathbf{H}$ will be equal to the L_2 -norm of the corresponding r^{th} row vector in $\mathbf{B}$ if and only if $b_{r,p} = 0, \forall p \in \mathbf{K}_s$. That is,

$$\text{if } 0 < \alpha \leq 2, \quad \|\text{row } r \text{ of } \mathbf{H}\|_2^2 = \|\text{row } r \text{ of } \mathbf{B}\|_2^2 \quad \text{iff } b_{r,p} = 0, \forall p \in \mathbf{K}_s$$

Proof:

From Lemma E.2 it has been established that if the learning constant α satisfies the condition $0 < \alpha \leq 2$, then $\|\text{row } r \text{ of } \mathbf{H}\|_2^2$ will not be greater than $\|\text{row } r \text{ of } \mathbf{B}\|_2^2$. The boundary condition of the learning constant α was computed with the upper limit of

$$\frac{\sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} = 1 \quad \text{such that } 0 < \alpha \leq 2$$

However, due to the neighborhood activation of computing cells in the PSECMAC network, the cardinality of $\mathbf{K}_s$ is always greater than one while the Gaussian weighting function $g_k \in (0, 1], \forall k \in \mathbf{K}_s$. Thus

$$\frac{\sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \in (0, 1)$$

By adopting a more restricted boundary condition of the learning constant α such that $0 < \alpha \leq 2$,

$$\underbrace{\alpha^2 \frac{\sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^2}}_{\in (0,1)} - 2\alpha < 0 \quad \text{if } 0 < \alpha \leq 2$$

It follows that term (V) of Lemma E.2 is always less than zero if $0 < \alpha \leq 2$ and $\exists p \in \mathbf{K}_s$ such that $b_{r,p} \neq 0$. That is,

$$\left(\left\{ \frac{\alpha^2 \sum_{k \in \mathbf{K}_s} (g_k)^2}{(\sum_{k \in \mathbf{K}_s} g_k)^4} \right\} - \frac{2\alpha}{(\sum_{k \in \mathbf{K}_s} g_k)^2} \right) \left(\sum_{p=1}^{\hat{M}^J} (b_{r,p} g_p) \right)^2 < 0, \\ \text{if } 0 < \alpha \leq 2 \quad \text{and} \quad \exists p \in \mathbf{K}_s \text{ s.t. } b_{r,p} \neq 0$$

Hence,

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 < \| \text{row } r \text{ of } \mathbf{B} \|_2^2 \quad \text{if } 0 < \alpha \leq 2 \quad \text{and} \quad \exists p \in \mathbf{K}_s \text{ s.t. } b_{r,p} \neq 0$$

Consequently, the condition of

$$\| \text{row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{row } r \text{ of } \mathbf{B} \|_2^2$$

will hold *if and only if* $b_{r,p} = 0$ for all $p \in \mathbf{K}_s$ since $g_p = 0$ for all $p \notin \mathbf{K}_s$. That is, from term (V) of Lemma E.2,

$$\text{if } 0 < \alpha \leq 2, \quad \left(\sum_{p=1}^{\hat{M}^J} (b_{r,p} g_p) \right)^2 = 0 \quad \text{iff } b_{r,p} = 0, \forall p \in \mathbf{K}_s$$

Therefore, $\| \text{row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{row } r \text{ of } \mathbf{B} \|_2^2$ follows. ■

Lemma E.4 *If the learning constant α satisfies the condition $0 < \alpha \leq 2$, then the L_2 -norm of any arbitrary r^{th} row vector in $(\mathbf{G}_s)^i$ is asymptotically bounded by the L_2 -norm of the corresponding r^{th} row vector in $(\mathbf{G}_s)^{i-1}$. That is,*

$$\| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 < \| \text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2, \quad s \in \{1 \cdots S\}$$

Furthermore, as the training iteration i approaches infinity,

$$\lim_{i \rightarrow \infty} \| \text{row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0$$

Proof:

From the definition of the matrix $\mathbf{G}_s$ (See Equation 4.36),

$$\mathbf{G}_s = \underbrace{\mathbf{E}_{s-1} \mathbf{E}_{s-2} \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_s}_{S \text{ terms}}$$

where S is the total number of input training samples and $\mathbf{G}_s$ is a $\hat{M}^J \times \hat{M}^J$ matrix. From Lemma E.3,

$$\begin{aligned}
 \text{if } 0 < \alpha \leq 2 \quad \|\text{row } r \text{ of } (\mathbf{G}_s)^{i-1}\|_2^2 &\geq \|\text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1}\|_2^2 \\
 &\geq \|\text{row } r \text{ of } ((\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1}) \mathbf{E}_{s-2}\|_2^2 \\
 &\geq \dots \\
 &\geq \|\text{row } r \text{ of } ((\mathbf{G}_s)^{i-1} \mathbf{E}_{s-1} \mathbf{E}_{s-2} \dots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \dots \mathbf{E}_{s+1}) \mathbf{E}_s\|_2^2 \\
 &\geq \|\text{row } r \text{ of } (\mathbf{G}_s)^{i-1} \mathbf{G}_s\|_2^2 \\
 &\geq \|\text{row } r \text{ of } (\mathbf{G}_s)^i\|_2^2
 \end{aligned}$$

However, since the matrix $\mathbf{G}_s = \mathbf{E}_{s-1} \mathbf{E}_{s-2} \dots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \dots \mathbf{E}_s$ and $(\mathbf{G}_s)^{i-1}$ is a multiplication of $(i-1)$'s number of $\mathbf{G}_s$, therefore $\|\text{row } r \text{ of } (\mathbf{G}_s)^{i-1}\|_2^2 > \|\text{row } r \text{ of } (\mathbf{G}_s)^i\|_2^2$ (i.e. Condition 2 of Lemma E.3 does not apply when we consider the entire set of training sequences). Following from above,

$$\begin{aligned}
 \|\text{row } r \text{ of } (\mathbf{G}_s)^{i-1}\|_2^2 &> \|\text{row } r \text{ of } (\mathbf{G}_s)^i\|_2^2 \\
 &> \|\text{row } r \text{ of } (\mathbf{G}_s)^{i+1}\|_2^2 \\
 &> \|\text{row } r \text{ of } (\mathbf{G}_s)^{i+2}\|_2^2 \\
 &> \dots
 \end{aligned}$$

Thus, as the training iteration i tends to infinity, $\lim_{i \rightarrow \infty} \|\text{row } r \text{ of } (\mathbf{G}_s)^i\|_2^2 = 0$.

This implies $(\mathbf{G}_s)^i = [0]$ as the training iteration i tends to infinity. ■

Lemma E.5 *If the learning constant α satisfies the condition $0 < \alpha \leq 2$, then as the training iteration i tends to infinity, the term $(\mathbf{G}_s)^i \mathbf{A}_v$ converges to a null matrix for all $v \in \{1 \dots S\}$. That is,*

$$\text{if } 0 < \alpha \leq 2, \quad \lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{A}_v = [0], \quad \forall v \in \{1 \dots S\} \quad s \in \{1 \dots S\}$$

Proof:

From Lemma E.4, $\lim_{i \rightarrow \infty} (\mathbf{G}_s)^i = [0]$, if $0 < \alpha \leq 2$.

Hence,

$$\text{if } 0 < \alpha \leq 2, \quad \lim_{i \rightarrow \infty} (\mathbf{G}_s)^i \mathbf{A}_v = [0], \quad \forall v \in \{1 \dots S\}$$
■

Appendix F

Financial Covariates for Banking Failure Classification

Table F.1: Definition of covariates and their expected impact on failure (Numbers in brackets are the identification of the data elements from the Call Reports)

CAMEL Category	Covariates	Expected Impact on Failure
Capital Adequacy	CAPADE average total equity capital (3210) / average total assets (2170) - (higher is the ratio, greater is the capacity to absorb losses, smaller is the probability of failure)	-ve
Asset (loan) Quality	OLAQLY average (accumulated) loan loss allowance (3123) / average total loans & leases, gross (1400) (smaller is the ratio, better is the loan quality, smaller is the probability of failure)	-ve
	PROBLO average (accumulated) loans 90 + days late (1407) / average total loans & leases, gross (1400) (higher is the ratio, poorer is the loan quality, higher is the probability of failure)	+ve
	PLAQLY (annual) loan loss provisions (4230) / average total loans & leases, gross (1400) (higher is the ratio, poorer is the loan quality expected to be, higher is the probability of failure)	+ve

continued on next page

continued from previous page

CAMEL Category	Covariates	Expected Impact on Failure
<u>Management</u>	NIEOIN non interest expense (4093) / operating income (4000) (higher is the ratio, less operationally efficient and profitable is the bank, higher is the probability of failure)	+ve
<u>Earnings</u>	NINMAR total interest income (4107) - interest expense (4073) / average total assets (2170) (higher is the net interest margin, more profitable is the bank, smaller is the probability of failure)	-ve
	ROE net income (after tax) (4340) + applicable income taxes (4302) / average total equity capital (3210) (higher is return on equity before tax, smaller is the probability of failure)	-ve
<u>Liquidity</u>	LIQUID average cash (0010) + average federal funds sold (1350) / average total deposits (2200) + average fed funds purchased (2800) + average banks liability on acceptances (2920) + average other liabilities (2930) (higher liquidity indicates inefficient utilization of resources; it can also reflect an expectation of unfavourable events (runs on deposits for example). Overall, higher liquidity suggests a higher probability of failure)	+ve
<u>Miscellaneous</u>	GROWLA total loans & leases, gross (1400) _t - total loans & leases, gross (1400) _{t-1} / total loans & leases, gross (1400) _{t-1} (with appropriate credit control and adequate loan loss provisions, a bank with higher loan growth rate would have better profitability and smaller probability of failure)	-ve

Appendix G

The MZCR Algorithm

The Modified Zero Crossing Rate (MZCR) algorithm is as listed below.

Algorithm MZCR

- a. Define a noise threshold level Th^{noise} .
- b. If the maximum signal amplitude in the current speech segment/frame is smaller than Th^{noise} , then the frame is considered as a silence frame.
- c. Else check for frame completeness.

Check the first 80 and the last 80 samples of the speech segment.

If the maximum signal amplitude in this first or last speech fragment is smaller than the noise threshold Th^{noise} , then the frame is considered as an incomplete frame.

- d. Compute the MZCR count for the identified voiced/unvoiced frame.

Let $\bar{x}_i(m) = (\bar{x}_i(1), \bar{x}_i(2), \dots, \bar{x}_i(m), \dots, \bar{x}_i(256))$ denotes windowed speech segment.

Define the MZCR threshold values Th^{MZCR} as:

$$Th^{MZCR} = mean(\bar{x}_i) \pm 0.05$$

$$\text{where } mean(\bar{x}_i) = \frac{\max_m(\bar{x}_i(m)) + \min_m(\bar{x}_i(m))}{2 \times \max(\max_m(\bar{x}_i(m)), abs(\min_m(\bar{x}_i(m))))}$$

denotes the normalized mean amplitude value of the current speech frame.

Compute the MZCR value for the current speech frame as the number of times the speech signal crosses over the two MZCR thresholds, i.e. from upper to lower thresholds or from lower to upper thresholds.

- e. If the MZCR count is larger than 10, the frame is considered as a voiced frame.
- f. Else, the frame is considered as an unvoiced frame.

End MZCR

Appendix H

ROC Curves of the PSECMAC-based Speaker Verification System

H.1 Unbalanced Training Scenario

The following figures depict the ROC curves extracted from the verification performances of the PSECMAC-based speaker verification system of the ten recruited speakers for the "unbalanced" training scenario.

(a) Person 1

(b) Person 2

(c) Person 3

(d) Person 4

(e) Person 5

(f) Person 6

(g) Person 7

(h) Person 8

(i) Person 9

(j) Person 10

Figure H.1: The ROC curves of the PSECMAC-based speaker verification system across the ten speakers for the "unbalanced" training scenario

H.2 Balanced Training Scenario

The following figures depict the ROC curves extracted from the verification performances of the PSECMAC-based speaker verification system of the ten recruited speakers for the "balanced" training scenario.

(a) Person 1

(b) Person 2

(c) Person 3

(d) Person 4

(e) Person 5

(f) Person 6

(g) Person 7

(h) Person 8

(i) Person 9

(j) Person 10

Figure H.2: The ROC curves of the PSECMAC-based speaker verification system across the ten speakers for the "balanced" training scenario

Appendix I

Insulin and Its Delivery Systems – A Review

Diabetes mellitus is a complex disorder that arises from various causes, including dysregulated glucose sensing or insulin secretion, autoimmune β -cell destruction, or insufficient compensation for peripheral insulin resistance (White, 2003). It is a chronic disease that is costly due to the required long-term treatment as well as the possible complications that may arise from the ailment. Literature abounds with evidences to show that glycaemic control in diabetes is efficient in preventing micro-vascular and neurological complications (Tyagi, 2002). Results from the Diabetes Control and Complications Trial (DCCT) have confirmed that intensive glucose control delays the onset and retards the progression of micro-angiopathic complications of type-1 diabetes (Diabetes Control and Complication Trial, 1993, 1995). Currently, the treatment of diabetes is mainly achieved through insulin medications. This appendix provides a brief review on insulin and its delivery systems.

Table I.1 lists the commonly used types of human insulin analogue and their response times. Regular or clear insulin was the fastest acting insulin until the introduction of Lispro. Most of the longer acting insulin analogues are cloudy due to the contained insulin suspensions and retarding agents. The time of onset, peak activity and duration of action of the insulin can only be approximated due to the variation in the degree of insulin absorption both between and within individuals from day to day. In some cases, the variation may reach as high as 25-50%, thus often leading to unexplained fluctuations in glucose control. The

Table I.1: Various types of insulin and their actions

Insulin Type	Actions			Trade Names
	Onset	Peak	Duration	
Rapid-acting analogue (clear)	10–15 min	60–90 min	4–5 hr	Humalog [®] (Lispro) NovoRapid [®] (Aspart)
Fast-acting (clear)	0.5–1 hr	2–4 hr	5–8 hr	Humulin [®] -R Novolin [®] ge Toronto
Intermediate-acting (cloudy)	1–3 hr	5–8 hr	up to 18 hr	Humulin [®] -N Humulin [®] -L Novolin [®] ge NPH
Long-acting (cloudy)	3–4 hr	8–15 hr	22–26 hr	Humulin [®] -U
Extended Long-acting analogue	90 min	–	24 hr	Lantus [®] (Glargine)
Premixed (cloudy)	contains fixed % ratio of fast- and intermediate-acting			Humalog [®] Mix25 [™] Humulin [®] (20/80, 30/70) Novolin [®] ge (10/90, 20/80, 30/70, 40/60, 50/50) NovoRapid Mix

speed of the insulin action also depends on the injection site. Insulin is absorbed fastest from the abdominal wall, slowest from the leg and buttock and at an intermediate rate from the arm. Insulin absorption also varies inversely with subcutaneous fat thickness, and is also determined by the rate of subcutaneous blood flow. Figure I.1 depicts the speed of insulin actions for various injection sites.

Figure I.1: Various sites of insulin injection

Insulin can be administered subcutaneously, intravenously or peritoneally. In general, there are two types of insulin delivery route:

1. Peripheral insulin delivery – administered via subcutaneous or intravenous route.

Subcutaneous insulin administration is often unreliable due to a variable absorption rate and hyperinsulinemia (Tyagi, 2002). The intravenous route, on the other hand, has the significant advantages of rapid delivery, a higher percentage of the drug reaching the bloodstream in comparison with the subcutaneous or intraperitoneal administrations, and potential for improved close-loop control (Parker et al., 2001). However, the peripheral delivery route administers the insulin to the body in a non-physiological manner as the insulin enters the *systemic* circulation first instead of the *portal* circulation. Moreover, there is a risk of complications of chronic vascular access associated with the use of the intravenous route.

2. Peritoneal insulin delivery.

The potential advantages of the peritoneal administration of insulin include the direct absorption of the insulin hormone into the portal system and liver which is a more physiological process (Duckworth et al., 1992), and a more reliable and predictable insulin delivery rate since the absorption variability and observed local degradation is minimal. Blood insulin levels peak within 15 minutes of intra-peritoneal delivery, as compared to the 60-90 minutes delay experienced for the intramuscular and subcutaneous routes. Other advantages include less fluctuations in the blood glucose levels, less alteration of lipid metabolism and less prominent immune response. Therefore, insulin pumps that deliver insulin intraperitoneally hold considerable promise in diabetes treatment because of the more physiologic delivery of insulin and the ability to inhibit hepatic glucose production selectively with less peripheral *insulinemia* than subcutaneous insulin injections (Simmons et al., 2001).

Appendix J

Current Glucose-Insulin Regulatory Systems

Table J.1: Existing Closed-Loop Glucose-Insulin Regulatory Systems

Glucose-Insulin Regulatory System	Patient Model	Meal Model	Route	Control Paradigm	Evaluation Method
(Fisher and Teo, 1989)	mathematical	blind	N.A.	optimal	The required amount of insulin is computed via optimal control theory. Insulin infusion by means of single injection, continuous infusion and closed-loop infusion are compared.
(Fischer et al., 1990)	mathematical	blind	IV-IV	classical	Validation is performed on diabetics Alsatian dogs.
(Kienitz and Yoneyama, 1993)	mathematical	blind	N.A.	classical	Validation is performed on simulated patient model. Objective is to maintain normoglycemia.
(Candas and Randziuk, 1994)	mathematical	N.A.	EX ¹ -IV,SC	classical/adaptive	A glucose clamping experiment is performed in-vivo on 3 pigs. Objective is to control the glucose infusion to maintain normoglycemia in response to insulin infusion.

continued on next page

continued from previous page

Glucose-Insulin Regulatory System	Patient Model	Meal Model	Route	Control Paradigm	Evaluation Method
(Shimoda et al., 1997)	mathematical	blind	SC-SC	classical	Trial on 10 patients with type-1 diabetes is performed. Objective is to follow insulin response of simulated healthy model.
(Trajanoski and Wach, 1998)	compartmental	blind	SC-SC	predictive	Validation is performed for simulated OGTT ² . Objective is to maintain normoglycemia.
(Parker et al., 1999)	compartmental	blind	N.A.	predictive/adaptive	Validation is performed for simulated postprandial and OGTT. Objective is to maintain normoglycemia.
(Lynch and Bequette, 2001)	mathematical	blind	N.A.	predictive	Validation is performed on simulated patient model. Objective is to maintain normoglycemia.
(Hovorka et al., 2004)	compartmental	blind	IV-SC	predictive/adaptive	Clinical experiments on 10 subjects with type-1 diabetes are performed. Objective is to maintain normoglycemia.
(Doran et al., 2005)	compartmental	blind	IP-IP	classical	ICU clinical trial is performed for OGTT. The predicted and actual insulin requirements are contrasted at the end of the trial to determine the control effectiveness.

¹ EX refers to external glucose measurement (by drawing blood).² OGTT stands for Oral Glucose Tolerance Test.

Appendix K

The GlucoSim Simulator

The mathematical models used in the GlucoSim simulator to describe the glucose-insulin interactions in the healthy and Type-1 diabetic subjects are based on two PhD dissertations (Puckett, 1992; Sorensen, 1985). The models developed in the two dissertations consider the pharmacokinetic dynamics of glucose and insulin, where the human body is partitioned into several compartments. Physiologically, each compartment represents a primary body organ involved in the glucose-insulin regulations. In GlucoSim, a material balance equation is formulated for each compartment, resulting in a set of differential equations that are solved simultaneously to derive the metabolic state of the simulated subject. This appendix presents a brief review on the compartmental models of (Puckett, 1992) and (Sorensen, 1985) as well as the compartmental model of the human metabolic process adopted by the GlucoSim simulator.

K.1 Puckett's Model of the Human Metabolic Process

The model developed by Puckett (Puckett, 1992) describes the glucose-insulin dynamics in a typical Type-1 diabetic subject. Except for the interstitial insulin transport sub-model, Puckett's diabetic model was developed from real patient data. The effects of exercise, body hormones (i.e. glucagon, epinephrine, cortisol, and growth hormone) and other metabolites (e.g. free fatty acids and amino acids) on blood glucose regulation were not considered in the model. The data used for the model development was obtained from human subjects

with Type-1 diabetes taking both short and long-acting insulin. It is assumed that the absorption rate of the long-acting insulin could be approximated as a zero-order process with constant rate. The time taken to inject the short-acting insulin, on the other hand, is assumed to be short and therefore the insulin effect is considered as instantaneous. The absorption of regular insulin from the subcutaneous tissues was subsequently modeled based on the works of (Turnheim and Waldhausl, 1988) and (King and Jahnson, 1985), resulting in a two compartmental representation. The interstitial insulin transport sub-model was identified using the data reported in (Camu and Rasio, 1972).

On the other hand, the rate of insulin independent glucose uptake (i.e. glucose uptake by the nervous system) is assumed to be constant and was evaluated with the data of (Pehlig et al., 1984). Glucose dependent glucose utilization is assumed to occur only in the kidneys, where glucose is excreted if the blood glucose concentration exceeds the limit of 176 mg/dl. The sub-model for glucose absorption from the small intestine is adopted from (Biermann and Mehnert, 1990), but the model parameters were obtained by fitting the model to the data for the rate of glucose absorption reported in (Pehlig et al., 1984). Lastly, the sub-model for liver glucose production was also identified from the data of (Pehlig et al., 1984).

K.2 Sorensen's Model of the Human Metabolic Process

In contrast to the model developed in (Puckett, 1992), the compartmental model of (Sorensen, 1985) was developed by mathematically representing the glucose-insulin-glucagon dynamics in healthy subjects. By removing the sub-model for pancreatic insulin release, a model for blood glucose regulation in subjects with Type-1 diabetes is then obtained. Physiological parameters of the Sorensen's model were selected to represent a typical 70 kg adult male. The same set of limitations (with respect to the Puckett's metabolic model) also applies to the Sorensen's model: no effects of exercise, body hormones and metabolites were considered.

The sub-model of glucose excretion by the kidneys was developed from the data presented in (Brod, 1973). This includes the rates of glucose filtration, reabsorption, and excretion as a function of the plasma glucose concentration. The red blood cell glucose uptake and brain glucose uptake are considered to be constants. In addition, it is assumed that the adipose tissue does not contribute significantly to the disposal of glucose in non-obese subjects. The

liver glucose production rate of the model was determined from experimental data using isotope dilution methods and the basal liver glucose production rate was determined based on the study of (DeFronzo et al., 1981). The effects of insulin, glucose and glucagon on the liver glucose production rate was identified based on existing literature studies (Sorensen, 1985). Similarly, the model parameters for liver glucose uptake were estimated based on published literature data. The liver glucose uptake rate is assumed to be mediated by glucose and insulin only. In contrast, the rate of the gut glucose uptake is assumed to be insensitive to changes in the glucose and insulin levels and was assigned a constant value.

To develop a model of pancreatic insulin secretion, Sorensen has adopted and modified the model presented in (Landahl and Grodsky, 1982). It has been established that both glucose and glucagon affect the insulin clearance process by the liver. In Sorensen's model, it is assumed that the liver clears 40% of the insulin presented to it, independent of the blood insulin concentration. The kidneys, on the other hand, is assumed to clear approximately 30% of the insulin flowing to it. In addition, Sorensen assumed that 15% of the insulin presented to the peripheral tissue is cleared. Lastly, the model assumed that glucagon is cleared primarily by the liver and kidneys at a linear rate of removal.

K.3 The GlucoSim Model of the Human Metabolic Process

The GlucoSim simulator adopts Puckett's Type-1 diabetic model for both the healthy and diabetic glucose metabolic processes. In the healthy person model, however, the subcutaneous insulin injection sub-model is replaced by a pancreatic insulin secretion sub-model. There are two such insulin secretion sub-models provided in the GlucoSim simulator. They are, namely: (1) Nomura's insulin secretion model (Nomura et al., 1984) that is based on the proportional-plus-derivative control theory; and (2) Carson's insulin secretion model (Carson and Cramp, 1976). The GlucoSim glucose metabolic process model also extends from Puckett's work to include the liver glucose production and intestinal glucose absorption functions of Sorensen's model (GlucoSim, Online).

Figure K.1 depicts the pharmacokinetics diagrams of the glucose and insulin models employed by the GlucoSim Simulator. The definitions of the notations used in the diagrams are listed as Table K.1. The glucose model (Figure K.1, left) contains the compartments such as the heart, brain, liver, kidneys and muscle, where the circulating glucose is used for energy.

Figure K.1: Pharmacokinetics diagrams of the glucose and insulin models of GlucoSim. Adapted from (GlucoSim, Online)

Glucose excretion by the kidneys and the functions of the gastrointestinal tract where the exogenous glucose enters the blood are also included. In the insulin model (Figure K.1, right), a compartment for the subcutaneous tissues is included as a source of insulin (via injection) for the Type-1 diabetic model, where it is assumed that the pancreas no longer produces any insulin at all.

In addition, GlucoSim also provides the avenue to simulate the metabolic effect of performing moderate exercise using the Type-1 diabetic patient model. Experimental data from (Berger et al., 1977) has been used to estimate the parameters for the total glucose uptake and the subcutaneous insulin absorption sub-models in response to exercise.

Table K.1: The definitions of the parameters used in the pharmacokinetics diagrams of GlucoSim

Variables	
G	Glucose concentration (mg/dl)
I	Insulin concentration (microU/dl)
r	Metabolic source/sink rate (mg/min)
Subscripts	Physiological Compartments
B	Blood
K	Kidney
NS	Nervous System
PR	Periphery
GT	Gastrointestinal Track
H	Heart
L	Liver
SC	Subcutaneous Tissue
Subscripts	Metabolic Rates
KGU	Kidney Glucose Utilization
NSGU	Nervous System Glucose Utilization
HGU	Heart Glucose Utilization
LGU	Liver Glucose Utilization
PRGU	Periphery Glucose Utilization
LGP	Liver Glucose Production
GA	Glucose Absorption
GE	Glucose Excretion
KIR	Kidney Insulin Removal
PRIR	Periphery Insulin Removal
LIR	Liver Insulin Removal
IA	Insulin Absorption

Appendix L

Asian Diet – Sample Menu

Table L.1: Typical daily meals (and carbohydrate contents) of Subject A based on the Asian diet

Meal	Food	Amount	Carb (g)	Total Carb (g)
Breakfast	Fresh Milk	1 cup	11.4	46.6
	White Bread	2 slices	29.8	
	Tuna	150 g	0	
	Cabbage, Green	0.5 cup	1.9	
	Tomato, Chopped	0.5 cup	3.5	
Lunch	Chicken Breast, Boneless	300 g	0	103.8
	Rice, White, Cooked	2 cups	89.2	
	Cucumber	0.25 pc	1	
	Coffee, Brewed	1 cup	0	
	Papaya	1 cup	13.6	
Afternoon Snack	Tea, Brewed	1 cup	0	54.8
	Blueberry Muffin, 2 oz	2 pc	54.8	
Dinner	Noodle, Egg, Cooked	2 cups	79.6	132.1
	Soup, Chicken Noodle	2 cups	18.8	
	Leeks	0.5 pc	6.3	
	Chicken Shreds	100 g	0	
	Egg, Whole	1 pc	0.6	
	Orange Juice	1 cup	26.8	
				337.3

Appendix M

The Extracted Glucose Variables for Modeling of Insulin Response

This appendix describes the glucose variables and the derivatives that are extracted from the current and past blood glucose measurements and subsequently used by the MCES feature selection algorithm to identify the salient features for the modeling of the insulin response in Chapter 7.

Let $G(t)$ denotes the measured blood glucose concentration G at time t , where the sampling interval between successive glucose measurements is 5 minutes. The glucose variables extracted from the training dataset include the current and past glucose measurements (i.e. $G(t)$ and $G(t - \tau)$, where $\tau \in \{1, 2, 3\}$), the current and past changes in the blood glucose level (i.e. $dG(t)$ and $dG(t - \tau)$, where $\tau \in \{1, 2\}$), and the current and past rate of changes in the blood glucose level (i.e. $ddG(t)$ and $ddG(t - \tau)$, where $\tau = 1$). In addition, *exponential moving averages* (EMA) (Enders, 2003) are used to capture the trend in the blood glucose fluctuations. In this work, the EMA of the last 2 (past 5 minutes period), 4 (past 15 minutes period) and 7 (past 30 minutes period) measurements of the blood glucose level $G(t)$ are tracked. That is,

$$G_{MA_S}(t) = MA_{N=2}(G(t)) \quad (M.1)$$

$$G_{MA_M}(t) = MA_{N=4}(G(t)) \quad (M.2)$$

$$G_{MA_L}(t) = MA_{N=7}(G(t)) \quad (M.3)$$

Table M.1: The glucose variables extracted to model the healthy insulin response

Index	Feature	Definition
1	$G(t)$	the current blood glucose level (at t)
2	$G(t-1)$	the blood glucose level 5 minutes ago (at $t-1$)
3	$G(t-2)$	the blood glucose level 10 minutes ago (at $t-2$)
4	$G(t-3)$	the blood glucose level 15 minutes ago (at $t-3$)
5	$dG(t)$	the change in the blood glucose concentration in the past 5 minutes interval, i.e. $dG(t) = G(t) - G(t-1)$
6	$dG(t-1)$	$dG(t-1) = G(t-1) - G(t-2)$
7	$dG(t-2)$	$dG(t-2) = G(t-2) - G(t-3)$
8	$ddG(t)$	the rate of change in the blood glucose concentration for the past 5 minutes, i.e. $ddG(t) = dG(t) - dG(t-1)$
9	$ddG(t-1)$	$ddG(t-1) = dG(t-1) - dG(t-2)$
10	$G_{MA_S}(t)$	the 2-point exponential moving average (EMA) of the blood glucose level $G(t)$, i.e. $G_{MA_S}(t) = MA_{N=2}(G(t))$
11	$G_{MA_M}(t)$	the 4-point EMA of the blood glucose level $G(t)$, i.e. $G_{MA_M}(t) = MA_{N=4}(G(t))$
12	$G_{MA_L}(t)$	the 7-point EMA of the blood glucose level $G(t)$, i.e. $G_{MA_L}(t) = MA_{N=7}(G(t))$
13	$dG_{MA_1}(t)$	the velocity of the blood glucose fluctuation, i.e. $dG_{MA_1}(t) = G_{MA_S}(t) - G_{MA_L}(t)$
14	$dG_{MA_2}(t)$	$dG_{MA_2}(t) = G_{MA_S}(t) - G_{MA_M}(t)$
15	$dG_{MA_3}(t)$	$dG_{MA_3}(t) = G_{MA_M}(t) - G_{MA_L}(t)$
16	$ddG_{MA_1}(t)$	the acceleration of the blood glucose fluctuation, i.e. $ddG_{MA_1}(t) = dG_{MA_1}(t) - MA_{N=3}(dG_{MA_1}(t))$
17	$ddG_{MA_2}(t)$	$ddG_{MA_2}(t) = dG_{MA_2}(t) - MA_{N=3}(dG_{MA_2}(t))$
18	$ddG_{MA_3}(t)$	$ddG_{MA_3}(t) = dG_{MA_3}(t) - MA_{N=3}(dG_{MA_3}(t))$

where S, M, L refers to *short* (2), *medium* (4) and *long* (7) respectively; and $MA_{N=n}(\cdot)$ denotes the EMA operator such that:

$$\begin{aligned}
 G_{MA_n}(t) &= MA_{N=n}(G(t)) \\
 &= (G(t) - G_{MA_n}(t-1)) \times \frac{2}{n+1} + G_{MA_n}(t-1)
 \end{aligned} \tag{M.4}$$

where $G_{MA_n}(t)$ is the EMA tracking the past n readings of G up to $G(t)$; and $G_{MA_n}(t-1)$ is the EMA tracking the past n readings of G up to $G(t-1)$. The parameter N determines the tracking horizon of the computed EMA.

The computed EMA values provide indicators to the trend or movement of the glucose concentration. By using the principle of *moving averages convergence-divergence* (MACD) (En-

ders, 2003), the trend of the blood glucose concentration $G(t)$ can be tracked as follows:

$$dG_{\text{MA}}(t) = G_{\text{MA}_{n1}}(t) - G_{\text{MA}_{n2}}(t), \quad n1 < n2 \quad (\text{M.5})$$

where

$$G(t) \text{ is } \begin{cases} \text{increasing} & \text{if } dG_{\text{MA}}(t) > 0 \\ \text{decreasing} & \text{if } dG_{\text{MA}}(t) < 0 \\ \text{no change} & \text{if } dG_{\text{MA}}(t) = 0 \end{cases} \quad (\text{M.6})$$

where $dG_{\text{MA}}(t)$ is defined as the convergence-divergence indicator for $G(t)$. The larger the magnitude of $dG_{\text{MA}}(t)$, the greater is the change of $G(t)$ from the last time instance. Thus, $dG_{\text{MA}}(t)$ may be interpreted as the *velocity* of $G(t)$.

Besides tracking the velocity of the glucose level $G(t)$, it is also useful to track the acceleration of the glucose fluctuation. The acceleration of $G(t)$ is defined as:

$$ddG_{\text{MA}}(t) = dG_{\text{MA}}(t) - \text{MA}_{N=3}(dG_{\text{MA}}(t)) \quad (\text{M.7})$$

where $\text{MA}_{N=3}(dG_{\text{MA}}(t))$ is the 3-point EMA operation on $dG_{\text{MA}}(t)$. Based on the combination of the 2-point, 4-point, and 7-point EMA of the blood glucose level, three measures of velocity and acceleration respectively are computed and considered as potential input to the MCES feature selection algorithm.

The glucose variables that are considered for the feature selection process are listed in Table M.1. Subsequently, Table M.2 tabulates the results of the MCES feature selection algorithm on these 18 glucose variables.

Table M.2: The results of the MCES algorithm

Rank	Index	Feature	MCES Value
1	1	$G(t)$	28.3393
2	11	$G_{MA_M}(t)$	7.6631
3	5	$dG(t)$	4.3246
4	10	$G_{MA_S}(t)$	4.0106
5	8	$ddG(t)$	0.6800
6	7	$dG(t-2)$	0.5975
7	15	$dG_{MA_3}(t)$	0.1781
8	16	$ddG_{MA_1}(t)$	0.1613
9	18	$ddG_{MA_3}(t)$	0.1028
10	14	$dG_{MA_2}(t)$	0.0727
11	17	$ddG_{MA_2}(t)$	0.0240
12	13	$dG_{MA_1}(t)$	-0.0219
13	6	$dG(t-1)$	-0.0365
14	9	$ddG(t-1)$	-0.0997
15	2	$G(t-1)$	-1.3326
16	4	$G(t-3)$	-5.1261
17	12	$G_{MA_L}(t)$	-9.7638
18	3	$G(t-2)$	-15.1147

Appendix N

The MCES Algorithm

The Monte Carlo Evaluative Selection (MCES) feature selection algorithm is as listed below.

Algorithm MCES

- a. Generate a random mask of the feature set, denoted as mask 1 (i.e. xxx1xxxxx...).
- b. Select a random position P in the feature set.
- c. Invert position P of the generated mask 1 to form mask 2 (i.e. xxx0xxxxx...).
- d. Obtain an instance of the training examples.
- e. Define Example 1 = instance with mask 1.
- f. Define Example 2 = instance with mask 2.
 - For mask with an enabled (1) position, preserve the value of the input feature.
 - For mask with a disabled (0) position, use the average value for the input feature.
- g. Compute output y_1 with Example 1 using the induction algorithm.
- h. Compute output y_2 with Example 2 using the induction algorithm.
- i. If computed output y_1 is closer than computed output y_2 to the target output y
 - Assign a positive feedback r_{jP} to mask 1 over mask 2 for that selected position P .
- Else
 - Assign a negative feedback r_{jP} to mask 1 over mask 2 for that selected position P .
- j. Repeat step (a) to (i) for all training examples for a predefined number of iterations.
- k. Compute the average feedback value for each bit position (input feature).
- l. Rank the features in the feature set according to the computed weights.

End MCES

To illustrate the workings of the MCES algorithm, assume a training data set where there are I input features and J training exemplars. In addition, let

- $\mathbf{X}$ denotes a matrix of size $J \times I$ consisting of J training exemplars and I input variables;
- $\mathbf{x}_i$ is a column vector that denotes the values of feature i for all the J training exemplars, such that $\mathbf{x}_i = [x_{1i}, x_{2i}, \dots, x_{ji}, \dots, x_{JI}]^T$;
- $\bar{\mathbf{x}}_i$ denotes the average value of all elements in $\mathbf{x}_i$ such that $\bar{\mathbf{x}}_i = \frac{1}{J} \sum_{j=1}^J x_{ji}$;
- $\mathbf{x}_j$ is a row vector that denotes the values of all the input features for exemplar j such that $\mathbf{x}_j = [x_{j1}, x_{j2}, \dots, x_{ji}, \dots, x_{jI}]$;
- x_{ji} denotes the value of feature i in exemplar j , where $x_{ji} \in \mathbf{X}$;
- $\mathbf{m}_j$ denotes the mask vector for exemplar j , where $\mathbf{m}_j = [m_{j1}, m_{j2}, \dots, m_{ji}, \dots, m_{jI}]$;
- m_{ji} denotes the masking bit for feature i in exemplar j , where $m_{ji} \in \{0, 1\}$;
- $\bar{r}_i$ denotes the average feedback value for feature i ;
- r_{jP} denotes the positive/negative feedback of feature P for exemplar j due to the switching of feature P from "on" to "off" or vice-versa based on bit P in *mask 1* and *mask 2* (Note that *mask 1* and *mask 2* differs only by the single bit P);
- W denotes the parameter set of the underlying induction algorithm; and
- $E(\cdot)$ denotes the error function based on the distance between the predicted and target outputs.

According to steps (a)-(i) of the MCES algorithm, the feedback r_{jP} is computed as:

$$r_{jP} = \begin{cases} |E(\bar{x}_P, \text{Mask}_{-P}(\mathbf{x}_j, \mathbf{m}_j), W) - E(x_{jP}, \text{Mask}_{-P}(\mathbf{x}_j, \mathbf{m}_j), W)|, & \text{if } |E(\bar{x}_P, \dots)| \geq |E(x_{jP}, \dots)| \\ -|E(\bar{x}_P, \text{Mask}_{-P}(\mathbf{x}_j, \mathbf{m}_j), W) - E(x_{jP}, \text{Mask}_{-P}(\mathbf{x}_j, \mathbf{m}_j), W)|, & \text{Otherwise} \end{cases} \quad (\text{N.1})$$

where $\text{Mask}_{-P}(\cdot)$ denotes the masking operation based on the similar parts in *mask 1* and *mask 2* without the differing bit P . The masking operation is defined by eq. (N.2) as follows.

$$\text{Mask}(\mathbf{x}_j, \mathbf{m}_j) = \underset{\forall i \in \{1, \dots, I\} \wedge i \neq P}{\text{Mask}}([x_{ji}], [m_{ji}]) \text{ s.t. } \text{Mask}(x_{ji}, m_{ji}) = \begin{cases} x_{ji}, & \text{if } m_{ji} = 1 \\ \bar{x}_i, & \text{if } m_{ji} = 0 \end{cases} \quad (\text{N.2})$$

Subsequently, the average feedback value $\bar{r}_i$ for feature i is computed as $\bar{r}_i = \frac{1}{J} \sum_{j=1}^J r_{ji}$.

The computed average feedback value or weight for each feature i reflects the relative importance of that feature over the rest of the input features. Negative or near-zero weight

indicates that the corresponding feature is irrelevant to the output(s). The degree of relevance increases with a larger value of the weight. In the current implementation, the error function E is defined as the difference between the actual and the predicted outputs. That is, $E = (y_m - y)$, $m \in 1, 2$. Thus eq. (N.1) can be simplified as:

$$\begin{aligned} r_{jP} &= \begin{cases} |(y_1 - y) - (y_2 - y)|, & \text{if } |E(\bar{x}_P, \dots)| \geq |E(x_{jP}, \dots)| \\ -|(y_1 - y) - (y_2 - y)|, & \text{Otherwise} \end{cases} \\ &= \begin{cases} |y_1 - y_2|, & \text{if } |E(\bar{x}_P, \dots)| \geq |E(x_{jP}, \dots)| \\ -|y_1 - y_2|, & \text{Otherwise} \end{cases} \end{aligned} \quad (\text{N.3})$$

where y denotes the actual (target) output; y_1 denotes the predicted output by the underlying induction algorithm with *mask 1* being applied to the inputs, and y_2 denotes the predicted output with the *mask 2* applied to the inputs.

The working of the novel MCES algorithm is graphically summarized as Figure N.1.

Figure N.1: Feature selection process of the MCES algorithm

Appendix O

Results of the PSECMAC Static Closed-Loop Insulin Control

The following Figures depict the detailed glucose control performances of the PSECMAC static closed-loop insulin control.

(a) Normal Diet

(b) Undereat Diet

(c) Overeat Diet

(d) Irregular Diet

Figure O.1: 3-days control performances of the PSECMAC-based static closed-loop glucose-insulin regulatory system under different dietary profiles

Appendix P

Publication List

Journal Papers

Papers Published:

1. **S.D. Teddy**, E. M-K. Lai and C. Quek. "Hierarchically Clustered Adaptive Quantization CMAC and Its Learning Convergence", *IEEE Transactions on Neural Networks*, 18(6):1658–1682, 2007.

Papers Accepted:

1. **S.D. Teddy**, C. Quek and E. M-K. Lai. "PSECMAC: A Novel Self-Organizing Multi Resolution Associative Memory Architecture", accepted for publication in *IEEE Transactions on Neural Networks*.
2. **S.D. Teddy**, E. M-K. Lai and C. Quek. "A Cerebellar Associative Memory Approach to Option Pricing and Arbitrage Trading", accepted for publication in *Neurocomputing*.

Conference Papers

Papers Published:

1. **S.D. Teddy**, E. M-K. Lai and C. Quek. "Cerebellar-based Text-Dependent Speaker Verification", *Fourth International Conference on Computational Intelligence, Robotics and Autonomous Systems – CIRAS 2007*, November 2007, Palmerston North, New Zealand.
2. **S.D. Teddy**, E. M-K. Lai and C. Quek. "A Cerebellar-based Approach to the Modeling of The Healthy Human Insulin Response to Food Disturbances", to be presented

- at 2007 International Congress on Modeling and Simulation – MODSIM 2007, December 2007, Christchurch, New Zealand.
3. **S.D. Teddy**, E. M-K. Lai and C. Quek. "A Brain-Inspired Cerebellar Associative Memory Approach to Option Pricing and Arbitrage Trading", in *Neural Information Processing: 13th International Conference – ICONIP 2006 Hong Kong, China, October 2006, Proceedings Part III*, Irwin King et al. (Eds), Lecture Notes in Computer Science, Volume 4234, Springer-Verlag, 2006, pp.370–379. **Best Student Paper Finalist.**
 4. **S.D. Teddy**, E. M-K. Lai and C. Quek. "A Neuropsychologically-Inspired Computational Approach to The Generalization of Cerebellar Learning", in *Neural Information Processing: 13th International Conference – ICONIP 2006 Hong Kong, China, October 2006, Proceedings Part I*, Irwin King et al. (Eds), Lecture Notes in Computer Science, Volume 4232, Springer-Verlag, 2006, pp.155–164.
 5. **S.D. Teddy**, E. M-K. Lai and C. Quek. "A Neurologically Inspired Cerebellar Model of The Human Blood Glucose Metabolic Process", *12th International Conference on BioMedical Engineering – ICBME 2005*, December 2005, Singapore.
 6. **S.D. Teddy**, E. M-K. Lai and C. Quek. "FQCMAC: A Multi-Resolution Cerebellar Learning Memory Model", *Third International Conference on Computational Intelligence, Robotics and Autonomous Systems – CIRAS 2005*, December 2005, Singapore.
 7. **S.D. Teddy** and E. M-K. Lai. "Hierarchical Clustering for Efficient Memory Allocation in CMAC Neural Networks", in *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005: 15th International Conference, Warsaw, Poland, September 2005. Proceedings, Part II*, W. Duch, J. Kacprzyk, E. Oja and S. Zadrozny (Eds.), Lecture Notes in Computer Science, Vol. 3697, Springer-Verlag, 2005, pp.473–478.
 8. **S.D. Teddy** and E. M-K. Lai. "Model-based Approach to Separating Instrumental Music from Single Channel Recordings", *Eighth International Conference on Automation, Robotics, Control and Vision – ICARCV 2004*, December 2004, Kunming, China, pp.1808-1813.