Protein secondary structure prediction using artificial neural networks and support vectors machines

Jin, Guosheng

2005

Jin, G. (2005). Protein secondary structure prediction using artificial neural networks and support vectors machines. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4454>

<https://doi.org/10.32657/10356/4454>

Nanyang Technological University

Downloaded on 26 Apr 2023 14:45:30 SGT

435/837
✓

Protein Secondary Structure Prediction Using Artificial Neural Networks And Support Vector Machines

Jin Guosheng

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

QA
76.87
J61
2005.

Acknowledgments

First of all, I would like to express my gratitude to my supervisor, Associate Professor Lipo Wang, for his invaluable and continued guidance throughout my Master program. With numerous discussions with him, I have not only gained a lot of professional knowledge and more importantly, learned the spirit of conducting research works. His constant suggestions, encouragement have greatly inspired me in facing and conquering difficult problems and finally reaching out satisfying solutions. Without his supervision, the thesis could not have been accomplished. I am also grateful for Dr. Huang Guang-bin, for his invaluable supervision especially in constructing and running simulations with modular support vector machines. He has provided a lot of valuable suggestions and encouragement for which I greatly appreciate.

I would like to sincerely thank Dr. Lin Chih-Jen and his co-workers for providing the LIBSVM library.

Thanks also go towards Nanyang Technological University for providing the financial help in the form of research scholarship.

Last but not least, I would like to thank all the staffs and students in Software Engineering Lab and Workstation Resource Lab, for their selfless help and suggestions.

Summary

In this thesis, a novel Hybrid Neural Networks Predictor (HNNP) system for the Protein Secondary Structure Prediction (PSSP) problem is described. By exploring such a new hybrid system, the intention is to investigate the feasibility of the HNNP in PSSP and even achieve improvements over existing methods. The proposed system is a cascaded network of the Radial Basis Function Neural Network (RBFNN) and the Multi-Layer Perceptron Neural Network (MLPNN). In the RBFNN, the overlapped clustering algorithm proposed by Fu and Wang is applied to set up the RBF kernels. In their study, they have shown the overlapped clustering technique assisted in enhancing the noise rejection capability of the RBF classifier. A novel distance scheme WINDist is also proposed for the distance measure of the clustering. The RBFNN predicted a protein's secondary structures based on its sequence. Based on the prediction of the RBFNN, an MLPNN using the traditional backpropagation learning algorithm performed the second stage prediction by using the correlation information hidden between secondary structures. Experimental results showed that the proposed HNNP was able to reach the prediction accuracy of 76.8%, which was higher than one of the best methods with prediction accuracy of 76.5%. These experiments confirm that HNNP is a viable system for the protein secondary structure prediction.

The Support Vector Machine (SVM) is investigated for the PSSP problem in this work. The system also includes two stages of prediction. Unlike the HNNP

system, the experimental results with the online SVM library LIBSVM showed that the LIBSVM could only reach a prediction accuracy of 74.2% from the sequence to structure prediction while no further improvements gained from the structure to structure prediction. The overall performance of the single C-Support Vector Classifier (C-SVC) performance was not as good as the HNNP in the protein secondary structure prediction.

Based on the LIBSVM, a Modular-SVMs architecture proposed by Huang is also applied for speeding up the learning speed. The modular network split the problem into subregions and conquered them one by one. Experimental results showed that the Modular-SVMs could speedup the learning by 20 times and meanwhile the prediction accuracy only dropped by 2%. In addition, the use of the Modular-SVMs has also successfully scaled down the large problem and reduced the high demand of computational resources. The application of the Modular-SVMs was successful in speeding up the learning of the PSSP system and meanwhile did not degrade the prediction accuracy significantly.

Contents

Acknowledgments	ii
Summary	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xi
List of Abbreviations	xiii
1 Introduction	1
1.1 Motivation.	1
1.2 Objectives	2
1.3 Major Contributions of the Thesis	3
1.4 Organization of the Thesis	4
2 The PSSP Basics	6
2.1 The Biological Background of the PSSP	6
2.1.1 Basic Protein Building Unit - amino acid	6
2.1.2 Types of the Protein Secondary Structure	7
2.1.3 The Task of the Prediction	9
2.2 Literature Review of the PSSP problem	11

3	Architectural Design of the HNPN	15
3.1	Framework Overview	15
3.1.1	Process Flow at the Training Phase	17
3.1.2	Process Flow at the Prediction Phase	18
3.2	First Stage: the Q2T Prediction	19
3.2.1	Architecture	19
3.2.2	Sequence Representation	22
3.2.3	Distance Measure Method for Data - WINDist	26
3.3	Second Stage: the T2T Prediction	30
3.3.1	Architecture	30
3.3.2	Sequence Representation	33
4	Training of the Hybrid Neural Networks Predictor (HNPN)	35
4.1	Training of the RBFNN	35
4.1.1	Hidden Layer Kernel Construction	35
4.1.2	Weights Estimation	38
4.2	Training of the MLPNN	41
4.2.1	The Delta Rule	41
4.2.2	Backpropagation	42
5	Experiments	47
5.1	Experimental Dataset	47
5.2	Accuracy Measure	48
5.2.1	Per-residue Accuracy	48
5.2.2	Matthew's Correlation	49
5.2.3	Segment Overlap Measure	50
5.3	Q2T Prediction- the raw prediction	51

5.3.1	Experiments with the Base and Alternative Distance Measure Schemes	53
5.3.2	Experiments with the Window Size and the Cluster Purity .	55
5.4	T2T Prediction - the final prediction	61
5.4.1	Feasibility of the Back Propagation	61
5.4.2	Strategy	63
5.4.3	Results	64
6	The PSSP Problem using Support Vector Machines	68
6.1	Introduction	68
6.2	SVM basics	69
6.2.1	SVM for Separable Patterns	69
6.2.2	SVM for Nonseparable Patterns	73
6.2.3	Construction of Support Vector Machines	76
6.2.4	Multi-class Problem with Binary Classifiers	78
6.3	Work with the LIBSVM	79
6.3.1	Q2T prediction	82
6.3.2	T2T prediction	87
6.3.3	Performance Comparison of the HNNP and the LIBSVM prediction	88
6.4	Modular-SVMs for the PSSP problem	89
6.4.1	The modular network for feedforward networks	89
6.4.2	The Modular Network with Support Vector Machines . . .	92
6.4.3	Speedup by the Modular-SVMs	93
6.4.4	Experimentwith tlic Modular-SVMs	93
7	Conclusion and Recommendation	98
7.1	Conclusion.	98

7.2 Recommendation	100
Bibliography	102

List of Figures

2.1	Alphabetical representation of the protein sequence and protein mutations	7
2.2	Three types of the protein secondary structure: α -helices are the dark ribbons on the boundary of the diagram, the gray ribbons in the centre are the β -strands that forms the β -sheet, coils are the springlike strings that binds the α -helix and the β -strand	8
2.3	The protein secondary structure prediction: two stages of implementation	11
3.1	Framework of the proposed hybrid neural networks predictor	16
3.2	Architecture of the sequence-structure prediction: the first stage . .	20
3.3	Bell-shaped Gaussian Profile: The kernel possesses the highest response at the center $x = 5$ and degrades to zero quickly	21
3.4	An example illustrating the fundamental sequence alignment theory	23
3.5	Input data representation: the raw profile (15×21) corresponds to the 10-th residue of the Hydrolase protein sequence.	25
3.6	Input data representation: the normalized profile	26
3.7	Relative displacement of the residues	28
3.8	Supervised weighting factor assignment: Scheme B	29
3.9	Architecture of the structure-structure prediction: the second stage	30
3.10	Response of the sigmoid function	31

3.11	Encoded form of the data: a) the full window, b) the partial window	34
4.1	Overlapped clustering: large overlaps for clusters of the same class and small overlaps for those of different classes	36
4.2	Architecture of the 3-layer multi-layer perceptron network.	43
6.1	A hyperplane for linearly separable patterns: The margin of separation between Class 1 and 2 is maximized	70
6.2	A hyperplane with the soft margin of separation: violations allowed either on the right or the wrong side of the decision surface for pattern A and B, respectively.	74
6.3	Architecture of the support vector machine as a binary classifier . .	78
6.4	Tertiary classifiers constructed for the protein secondary structure prediction problem based on a) One-Versus-One Scheme; b) One-Versus-All Scheme.	80
6.5	Architecture of the modular feedforward network with a group size of L	91
6.6	Architecture of the Modular-SVMs with a group size L	92

List of Tables

5.1	Performance comparison of two WINDist schemes: the base value b is varied within the range $[0.30.7]$ with 0.1 of change at each time .	54
5.2	Prediction accuracies of the sequence-structure prediction with window size $N = 11$ and various purity level ρ	57
5.3	Prediction accuracies of the sequence-structure prediction with various window size N and purity level ρ : $N \in \{13, 15, 17\}$	58
5.4	Prediction accuracies of the sequence-structure prediction with various window size N and purity level ρ : $N \in \{19, 21, 23\}$	59
5.5	Prediction accuracies of the sequence-structure prediction with various window size N and purity level ρ : $N \in \{25, 27\}$	60
5.6	Peak accuracies reached at each window size N : a summary	60
5.7	Prediction accuracies of the structure-structure prediction for window size $N = \{11, 13, 15, 17, 19, 21, 23, 25\}$	65
5.8	Contributions of the sequence-structure and the structure-structure prediction to the final prediction accuracy	65
5.9	More thorough performance evaluation of the proposed HNNP and its comparison with the PSIPRED method	67
6.1	Prediction accuracies of the C-SVC with different (C, γ) values: $N = 11$	85

6.2	Prediction accuracies of the C-SVC with different (C, γ) values: $N =$ 13	86
6.3	Prediction accuracies of the C-SVC with different (C, γ) values: $N =$ 15	86
6.4	Prediction accuracies of the C-SVC with different (C, γ) values: $N =$ 17	87
6.5	The structure-structure prediction accuracies for window size $N \in$ $\{15, 17, 19\}$	88
6.6	Performance comparison for C-SVC Vs HNNP: the HNNP is supe- rior to the C-SVC in terms of per-residue accuracy.	89
6.7	Prediction accuracies of the Modular-SVMs with different parame- ter set: $N = 15, L \in \{5, 10\}$ and various (C, γ) combinations	95
6.8	Accuracy and timing comparison for the Modular-SVMs and the single SVMs	95

List of Abbreviations

ABBREVIATIONS	FULL EXPRESSIONS
HNNP	Hybrid Neural Networks Predictor
MLP	Multi-Layer Perceptron
MLPNN	MLP Neural Network
PSSP	Protein Secondary Structure Prediction
Q2T	Sequence-Structure (prediction)
RBF	Radial Basis Function
RBFNN	RBF Neural Network
SVM	Support Vector Machine
T2T	Structure-Structure (prediction)
WTA	Winner-Take-All

Chapter 1

Introduction

1.1 Motivation

Proteins are the fundamental molecules for all living organisms. They build and repair body tissues, produce chemical catalysts (for example, enzymes), and protect the body from disease carrying bacteria and viruses. A protein has a three-dimensional structure and the functions it plays are fully specified by its protein sequence, e.g., the one-dimensional arraying of amino acids.

Nowadays, large-scale sequencing projects have already produced an exploding number of known protein sequences, currently numbered at around 145,000 [1]. On the other hand, the number of known protein structures, obtained from highly labor intensive experimental methods such as x-ray diffraction studies and nuclear magnetic resonance (NMR) analysis, is as low as one-sixth of that. According to the statistical report dated on 3, Mar 2004 from the single worldwide repository for protein 3D structure, Protein Data Bank (PDB), only 24,000 protein structures out of the 145,000 sequences have been solved [2].

Due to the high intensity of experimental methods and the large discrepancy between known sequences and structures, it would be a desirable attempt to predict protein's structure solely based on its primary amino acid sequence. However, the current theories of how amino acid sequences fold into three-dimensional struc-

1.2. Objectives

tures are incomplete and insufficient to explain the sequence-structure mappings. The process of deriving the tertiary structure from the primary sequence involves countless interactions between indeterminate factors [3] and this has made the transformation immensely complex. As the knowledge of such interactions is still unknown and intractable, this has been the major limitation of creating fully automated systems.

On the other hand, the protein structural backbone can be described by assigning the protein secondary structure to each amino acid residue [4]. The quality and resolution of protein structural features detected by statistical analysis of protein backbone chains is dependent on such description [5] [6]. Hence, the Protein Secondary Structure Prediction (PSSP) can provide and has been a popular first step toward predicting the protein's 3-dimensional structure [7] [8] [9] [10]. This has been an important step towards discovering the protein's three-dimensional structure as well as its functions.

The PSSP problem has been studied for decades and a lot of efforts have been dedicated to improve the prediction accuracy [11] [12] [13] [14] [15] [16]. By the time when this work was completed, the state-of-the-art protein secondary structure prediction has reached an accuracy level of 80% [17].

1.2 Objectives

The objective of this project is to design and develop a new classifier using artificial intelligent approaches. By exploring such new approaches, the intention is to achieve improvements in the protein secondary structure prediction accuracy over existing methods, or the extreme speedup in training without significant drop in accuracy. The proposed system should have the following capabilities:

1. Efficient in learning: Simulations on the high dimensional and extremely

1.3. Maior Contributions of the Thesis

large-scale protein data demand a lot of computational resources in terms of the CPU speed and the memory. It is desirable that a fast learning machine could be constructed such that it significantly improves the learning speed, reduces the memory requirement and meanwhile, does not reduce the accuracies significantly. Hence, the learning algorithm used for training should be efficient in terms of the computational complexity. Meanwhile, it is important that the learned system should generalize well.

2. Accurate in prediction: The proposed system should be able to achieve a satisfactory prediction accuracy and the accuracy shall be comparable or even better than the existing methods.

1.3 Major Contributions of the Thesis

This thesis describes the application of the artificial neural network and the support vector machine for solving the PSSP problem. The major contributions of the thesis are:

1. Proposed a novel Hybrid Neural Network Predictor (HNNP) which combines the Radial Basis Function Neural Network (RBFNN) and the Multi-Layer Perceptron Neural Network (MLPNN). This is the first application of the RBFNN in the PSSP problem. Experimental results showed that the HNNP could reach an accuracy that is 0.3% higher than the existing PSIPRED method.
2. Proposed two novel distance measure schemes WINDist A and B for computing the distance between two data point. Each data point represents one amino acid residue in the sliding window form. This form is built up by extending from the amino acid to the neighboring residues on the both side

1.4. Organization of the Thesis

of the sequence. The both schemes tried to evaluate the importance of each window position for representing the data characteristics. The experimental results showed that the proposed distance measure is efficient for clustering.

3. The application of an extremely fast Modular Support Vector Machines (Modular-SVMs) classifier for the PSSP problem. The classifier was able to split the input space into sub regions and learned them one by one. Experimental results showed that the Modular-SVMs could enhance the speed by 7 - 23 times with no more than 2% drop in accuracy, in comparison to the single Support Vector Machines (SVMs) case. Besides, the memory requirement has also been greatly reduced to at least 5 times less.

1.4 Organization of the Thesis

The thesis is organized as follow:

Chapter 2 introduces the background information related to the PSSP problem. This includes the constitution of a protein sequence and its secondary structure, the definition of the PSSP problem. In the later part of the chapter, a literature review is given regarding the history of the PSSP, the algorithms used for the PSSP problem nowadays.

Chapter 3 describes the architectural design of the proposed HNNP. The detailed elaboration of the process flow of the system, the data representation, the RBF network for the sequence-structure (Q2T) prediction and the MLP network for the structure-structure (T2T) prediction, are presented. Two distance measure schemes proposed for the clustering during the RBF kernel construction are also described.

Chapter 4 describes the theoretical as well as the mathematical background to algorithms applied by the RBFNN and the MLPNN are explained.

1.4. Organization of the Thesis

Chapter 5 demonstrates the experimental results obtained both in the Q2T and the T2T prediction. The strategies adopted for searching for optimal network parameter sets are explained and discussions are also presented for each set of simulation results. Comparisons are made with the existing algorithms.

Chapter 6 describes the application of the SVMs in the PSSP problem. A brief introduction of SVM basics is provided in the beginning part of the chapter. Application of the single SVMs and the fast Modular-SVMs on the PSSP problem are described. Experimental results are illustrated and relevant discussions are given.

In Chapter 7, conclusion are drawn for this work and the future research works are recommended.

Chapter 2

The PSSP Basics

2.1 The Biological Background of the PSSP

2.1.1 Basic Protein Building Unit - amino acid

A *protein sequence* is a linear arraying of amino acids. Each amino acid is translated from 3 consecutively ordered looking DNA bases (either can be A, or C, or T, or G). The amino acid is the basic unit of the protein sequence and is referred to as a *residue*. The triplet code implies that there are $4^3 = 64$ possible permutations. However, there are only 20 amino acid types and this results a redundancy in the genetic code. Thus, almost each of the amino acids is encoded by synonymous permutations which are interchangeable in the sense of producing the same amino acid. For the convenience of presentation, each amino acid type is represented by an alphabetic letter. For example, the amino acid named Alanine is represented by the letter 'A'. A protein sequence thus in the alphabetical representation is a long sequence of characters, as the example shown in Figure 2.1. A protein sequence may be subject to the evolutionary changes that may induce mutations, including insertions, deletions or substitutions, to the original protein sequence and thereafter produce different functions.

2.1. The Biological Background of the PSSP

Name : Complex Of Troponin C With A 47 Residue (1-47) Fragment Of Troponin I				
PDB ID : 1A2X:B				
Sequence :				
1) GDEEKRNR	TARRQHLKSV	MLQIAATELE	KEEGRREAEK	QNYLAEH
2) GDEEKRNR	TARRQHLK_	MLQIAATELE	KEEGRREAEK	QNYLAEH
3) GDEEKRNR	TARRQHLKSV	MLQIAATELEEEE	KEEGRREAEK	QNYLAEH
4) GDEGFRNR	TARRQHLKSV	MLQIAATELE	KEEGRREAEK	QNYLAEH
<u>Note</u> 1) Original Protein Sequence (47 Residues) 2) Deletion: several amino acids deleted from the chain 3) Insertion: amino acids FFE was inserted into the original sequence 4) Substitution: the replacement of amino acids segment by GF				

Figure 2.1: Alphabetical representation of the protein sequence and protein mutations

2.1.2 Types of the Protein Secondary Structure

Secondary structures are regular structural elements which are formed by hydrogen bonds between relatively small segments of the protein sequence. Often the driving force for the formation of secondary structure is the saturation of backbone hydrogen donors (NH) and acceptors (CO) with intra-molecular hydrogen bonds. This saturation allows the protein to bury hydrophobic side-chains in its interior (hydrophobic core) without conflicting with the polar backbone. There are three common secondary structures in proteins, namely α -helix, β -strand and coil.

α -helix

An α -helix is formed from a connected stretch of amino acids. The α helix is characterized by hydrogen bonds along the chain, almost co-axial. The α helix is the most abundant helical conformation found in globular proteins. The average length of an alpha helix is around 10 residues.

β -strand

β -strand is the principle component of a β -sheet. The β -sheet is characterized by hydrogen bonds crossing between chains. Each participating β -strand in a β -sheet

2.1. The Biological Background of the PSSP

is not continuous in term of the primary sequence and does not even have to be close to each other in the sequence. A β -strand has a sequence of 5-10 residues in a very extended conformation.

Coil

Approximately one-third of all residues in globular proteins are contained in coils. The coils in a protein serve to reverse the direction of the polypeptide chain. A coil is vary in length.

Figure 2.2 illustrates the visualized secondary structures of the protein. In the diagram shown, the dark ribbons represent helices. The gray ribbons are β -strands that form the β -sheet. The spring like strings in between these two secondary structures are the coils that bind them.

Figure 2.2: Three types of the protein secondary structure: α -helices are the dark ribbons on the boundary of the diagram, the gray ribbons in the centre are the β -strands that forms the β -sheet, coils are the spring like strings that binds the α -helix and the β -strand

2.1. The Biological Background of the PSSP

2.1.3 The Task of the Prediction

The term “prediction” in the ProteinSecondary Structure Prediction (PSSP) domain carries the similar meaning as the data mining term “classification”: given a residue of the protein sequence, the predictor should classify it into one out of three secondary structure states based on its characteristics. Note that the outcome of the prediction is a state of the secondary structure rather than the secondary structure itself. One residue is only the constitutional element of a secondary structure. A protein secondary structure consists of several residues sharing the same secondary structure state. In other words, the secondary structure state is associated with one amino acid while the secondary structure is for an ensemble of amino acid residues. The prediction of the protein secondary structure can be conducted in two stages [18] [19] [20] [14]: the *sequence-structure (Q2T) prediction* and the *structure-structure (T2T) prediction*.

Sequence-Structure Prediction

The Q2T prediction predicts the protein secondary structure from the primary linear protein sequence. Given a protein sequence, the Q2T predictor maps each entry of the sequence to the relevant secondary structure state by using the data representation information of the input residue. The data representation attempts to capture information related to the type of the amino acid, the sequence context (that is, what are the neighboring residues of the input), the evolutionary information and etc.. The sequence-structure prediction gives the major contribution in the prediction in term of the amount of accuracies contributed.

Structure-Structure Prediction

As mentioned before, the secondary structure is an ensemble of consecutive amino acid residues sharing the same secondary structure state. The neighboring sequence

2.1. The Biological Background of the PSSP

positions usually present some correlation characteristics in terms of the secondary structure formation. For example, it is usually observed that an α -helix contains at least 3 consecutive amino acids who are all in the α -helix state. Suppose that there are alternative occurrences of the α -helix and the β -strand states (e.g., $\alpha\beta\alpha\beta\dots$) in the outcome of predicted secondary structures, it is obvious that the prediction is not right. The above mentioned is just one example of the correlations existed in the neighboring residues. There are also other correlations known or unknown. Therefore, the T2T prediction which is based on the outputs of the first stage is needed. This is the second stage of prediction. This stage of prediction attempts to correct unrealistic predictions from the previous stage and hence enhances the overall prediction accuracy. This stage of prediction acts as the complementary to the sequence-structure prediction.

Figure 2.3 illustrates the scenario of the secondary structure prediction with two stages of implementations.

It is important to note that the same type of amino acid needs not to be predicted to the same secondary structure state in the secondary structure prediction. For instance, in Figure 2.3, the 12-th and the 20-th amino residues counted from left hand side are both of type 'F' yet are assigned to two different secondary structure states. It is rather the distinct characteristics embedded with the residue such as the sequence context, the evolutionary information, biochemical properties, speed of translation, etc., play more significant role in the formation of the protein structure.

For the PSSP it is acknowledged that the neighboring residues have impact on the predictive capability of the secondary structure. Therefore, the prediction of the secondary structure at each sequence position should not solely rely on the residue at that position. Rather, a window expanding towards both directions of

2.2. Literature Review of the PSSP problem

the residue should be used to include the sequence context. This issue will be discussed in detail in the next chapter.

Figure 2.3: The protein secondary structure prediction: two stages of implementation

2.2 Literature Review of the PSSP problem

The problem of the protein secondary structure prediction was stimulated by the first publication of the protein 3D structure in 1960 [21] [22], which attempted to find the correlations between the content of protein sequence (amino acids) and that of the secondary structure (α -helix, β -strand). This was when the first generation of secondary structure predictors were born. In this generation, most methods developed prediction based on single residue statistics [23] [24] [25] [26] [27]. Preferences to particular amino acids from protein sequences were extracted and used in experiments. The accuracies of these methods were found over-estimated [19].

With the growth of known protein structure obtained from experiments, the second generation exploited the segment statistics to predict the secondary structure. A consecutive segment of residues was studied to find out how much likely the

2.2. Literature Review of the PSSP Problem

residue of the segment central belonged to one secondary structure state. Major algorithms applied in this generation includes the statistical information [26] [28]; sequence patterns [11] [as]; multi-layer networks [30] [12] [13] [31]; multivariate statistics [32]; nearest-neighbor algorithms [33].

Both the first and the second generations could reach an accuracy level no higher than 70%.

The earliest application of the artificial neural network on the protein secondary structure prediction was introduced by Qian and Sejnowski (1988) [34]. In their work, they used a three-layer back-propagation network. The input data was binary encoded by a code scheme called BIN21. Each input data was a sliding window of 13 residues obtained by extending 6 sequence positions from the central residue. The focus of each observation was only on the central residue, i.e., only the central residue was assigned to one out of three possible secondary structure states: the α -helix, the β -strand or the coil. Minor variations to the BIN21 scheme were used in two studies: Kneller [31] et al added one additional input unit to present the hydrophobicity scale of each amino acid residue and showed slightly increased accuracy; Sasagawa and Tajima [35] used the BIN24 scheme to encode three additional amino acid alphabets, B, X, and Z. As all above works applied local encoding scheme which utilized only the local information of the protein sequences, a performance ceiling of 65% accuracy was reached on their own datasets. In 1995, Vivarelli et al [36] used a hybrid system that combined a Local Genetic Algorithm (LGA) and neural networks for the protein structure prediction. Although LGA takes advantages in selecting network topologies efficiently, the result still showed that the ultimate performance of the network could not go beyond the limited accuracy, regardless what type of network architectures has been applied.

More research has been conducted focusing on analyzing and improving the

2.2. Literature Review of the PSSP problem

secondary structure prediction have been carried out [15][37][38][39][40].

A significant improvement of the 3-state secondary structure prediction accuracy reaching over 70% came from Rost and Sander's work (1993 & 1996) [18][41]. In their method (PHD), a similar multi-layer back-propagation network was used. In contrast to using the BIN21 coding scheme, a new key aspect—the evolutionary information in the form of *frequency statistics from the multiple sequence alignments*, was introduced to represent the input data. The inclusion of the protein family information in this form increased the prediction accuracy by around six percentages. Moreover, a second cascaded neural network architecture, which performed the additional structure-structure prediction, was used to introduce the correlations between the secondary structure of adjacent amino acid residues in the sliding window. With a dataset of 126 protein sequences (RS126), Rost and Sander broke the 70% magic accuracy barrier and achieved the overall network performance as high as 72% in accuracy. In 1996, further neural network architecture and machine learning refinements were employed by Riis and Krogh [42]. An adaptive encoding of the input amino acids by the neural network weight sharing technique was used to reduce the number of weights needed. The encoding is adaptive in that the encoding could learn throughout the training process. Specialized networks were designed for each secondary structure class and combined using another neural network. Despite that the architectural design was more complicated, the experiments on the RS126 dataset, which again made use of the frequency statistics from the multiple sequence alignment, reached the overall accuracy of 71.3%, which was still similar to that of the PHD method.

More recently, David T. Jones [20] used the *Position-Specific Scoring Matrix* (PSSM) [43][44], obtained from the online alignment searching tool PSI-Blast [45], to numerically represent the protein sequence. A PSSM was constructed automat-

2.2. Literature Review of the PSSP Problem

ically from a multiple alignment of the highest scoring hits in an initial BLAST search. The PSSM was generated by calculating position-specific scores for each position in the alignment. Highly conserved positions of protein sequence received high scores and weakly conserved positions received scores near zero. Due to its high accuracy in finding the biologically similar protein sequences, the evolutionary information carried by PSSM is more sensitive than those profiles obtained by other multiple sequence alignment approaches. With the neural network architectural similar to that of Rost and Sander's, Jones' PSIPRED method achieved an accuracy as high as 76.5% using a much larger dataset than RS126.

In 2001, Hua and Sun also proposed a Support Vector Machine (SVM) approach. This was an early application of the SVM on the PSSP problem. In their work, they first constructed 3 one-versus-one and 3 one-versus-all binary classifiers. Three tertiary classifiers were designed based on these binary classifiers through the use of the largest response, the decision tree and votes for the final decision. By making use of the Rost's data encoding scheme, they reached the accuracy of 71.6% and the segment overlap performance of 74.6% on the RS126 dataset. In 2003, Minh and Jagath constructed multi-class support vector machines using the Vapnik & Weston [46] [47] method. With the large dataset used by the PSIPRED method, the system reached a maximum accuracy of 79.5% [17]. This is the published best accuracy that has ever been reached by the time when this thesis was completed.

Chapter 3

Architectural Design of the HNNP

As discussed in the previous chapter, the secondary structure predictor needs two stages of the prediction. In this chapter, the architectural design of the proposed Hybrid Neural Network Predictor (HNNP) will be described. The predictor consists of two artificial neural networks: a Radial Basis Function Neural Network (RBFNN) at its first stage for the sequence-structure (Q2T) prediction and a Multi-Layer Perceptron Neural Network (MLPNN) at the second stage for the structure-structure (T2T) prediction. At the beginning of this chapter, a framework overview of the HNNP system would be described first. Detailed elements of the framework, e.g., the RBFNN and the MLPNN, would also be described in the subsequent sections.

3.1 Framework Overview

Figure 3.1 illustrates the overall flow of the HPPN. The “Sequence Content” storage contains a number of protein sequences used for the training or the prediction. At each flow through the predictor, only one residue instead of an entire protein sequence is fed as input.

3.1. Framework Overview

Figure 3.1: Framework of the proposed hybrid neural networks predictor

3.1. Framework Overview

3.1.1 Process Flow at the Training Phase

The flow of the prediction in the training phase is as follow:

1. Given a residue in the alphabetic form, it is numerically encoded into a 20-dimensional vector by the “Encoder A”. The 20-dimensional vector is the frequencies statistics obtained from the multiple sequence alignment for the residue as appearing in its belonging sequence.
2. The 20-dimensional vector is then passed to a “Normalizer”. The purpose of the normalization is to restrict values of the vector entries to within the range (0,1).
3. The sequence-structure prediction block is a three-layer Radial Basis Function Neural Network. In this network, the overlapped clustering technique proposed by Fu and Wang [48] was applied to construct the RBF kernels. In their study, Fu and Wang have shown that the overlapped clustering technique assisted in enhancing the noise rejection capability of classifier and could slightly improve the classification accuracy. In this work, the intention is to explore the noise rejection capability such a RBFNN in improving the PSSP accuracy. In the training phase of the sequence-structure prediction, the input was fed into this block to establish the network. This includes the discovering of a set of radial basis function kernels as well as the optimal weight set between the second and the third layer. After the network is established, the training samples are fed in again one by one. The output is stored as the training data for the next stage.
4. The output from the previous stage is further encoded by the “Encoder B” by real values between (0,1).

3.1. Framework Overview

5. The structure-structure prediction block is implemented by a three-layer MLP network. The block attempts to correct the predictions made from the sequence-structure prediction and improve the overall prediction accuracy. Originally, it is intended a same RBFNN to be used for the sequence-structure prediction. The early work revealed that the RBFNN for the structure-structure prediction actually degraded the overall prediction performance. This is not the case when the MLP was employed for the structure-structure prediction. Hence, the MLP instead of the RBFNN was employed for this stage of the prediction and this resulted a hybrid system combining different neural networks at each prediction stage. In the training phase of the structure-structure prediction, the output of first stage is fed into the MLP network one by one. The network learns from these samples and the network parameters are tuned accordingly to minimize the cost function. When the parameters are optimized, the training of the overall all predictor is done and testing could be carried on this trained system.

3.1.2 Process Flow at the Prediction Phase

The prediction phase performs the prediction on the testing dataset and the results are evaluated for accuracy. Given a protein sequence, the flow of the prediction is as follow:

1. The sequence is properly encoded by the “Encoder A”.
2. The set of 20-dimensional vectors representing the whole protein sequence is normalized.
3. Each protein residue is fed into the sequence-structure prediction block one by one. The RBFNN uses the established parameters to predict what the

3.2. First Stage: the Q2T Prediction

relevant secondary structure state is. This prediction generates the primary outcome of the prediction.

4. The output from the previous stage is again Encoded by “Encoder B” and fed into the MLP network. The MLP uses the learned parameters to generate the final secondary structure prediction.
5. The “Decision Maker” uses the Winner-Take-All (WTA) technique is employed to complete the secondary structure assignment. The secondary structure class with highest possibility is selected to be the predicted secondary structure state. The outcome of secondary structure patterns of a whole sequence is saved in the storage as the “Secondary Structure Content” of the sequence.

3.2 First Stage: the Q2T Prediction

3.2.1 Architecture

The architectural design of the Q2T prediction is shown in Figure 3.2. The RBFNN consists of three layers: the input layer, the hidden layer and the output layer.

The *input layer* is where the protein sequence as input is fed into the network on the residue basis. A protein sequence has to be encoded by certain data representation scheme to allow neural networks to manipulate on it. Section 3.2.2 would describe the data representation scheme used in details. As there is no weight associated with the input and the hidden neurons in RBF neural network, this layer is transparent. The input pattern is fed to hidden layer kernels directly without being weighted.

The *hidden layer* consists of a set of kernels corresponding to the prototypes in the input space. In this layer, the data is clustered into a finite number of

3.2. First Stage: the Q2T Prediction

Figure 3.2: Architecture of the sequence-structure prediction: the first stage

radial regions relevant to the input space. The response of each region can be represented by a kernel function. Typical transfer functions may be used for these kernels include:

- Gaussian function;
- Spline functions;
- Multi-quadratic functions;
- Inverse multi-quadratic functions.

In this project, the popular Gaussian transfer function of the form

$$g(x) = e^{-d(x)^2/2\sigma^2}, \quad (3.1)$$

is used, where x is the input pattern, $d(x)$ is the distance between the input pattern x and the center of the kernel, and σ defines the width of the kernel.

If the input pattern x is one-dimensional, the shape of the Gaussian function is like a bell (Figure 3.3), which possesses the highest response of 1 at the center and quickly degrades when x moves away from the center. The responses of kernel

3.2. First Stage: the Q2T Prediction

Figure 3.3: Bell-shaped Gaussian Profile: The kernel possesses the highest response at the center $x = 5$ and degrades to zero quickly

for data points whose distances to the center of the kernel exceeding two times of the kernel width (2σ) are rather weak and may be neglected.

The output layer linearly combines the outputs of hidden neurons. For any output neuron i , the output can be formulated as follow:

$$y_i^{(3)}(x) = \sum_{j=1}^m g_{ij}^{(2)}(x)w_{ij}, \quad (3.2)$$

where $g_{ij}^{(2)}(x)$ is the output of the j -th kernel for the input pattern x , and w_{ij} is the weight associated with the i -th output neuron and the j -th hidden kernel. m is the total number of Gaussian kernels (or hidden neurons) in the hidden layer. In this project, each output neuron corresponds to one of the 3 secondary structure states. The value associated with each neuron indicates the likelihood that the input pattern belongs to that secondary structure state. The decision of the final predicted secondary structure state is made by the Winner-Take-All (WTA) technique. Assume the secondary structure states assigned to output units 1, 2, 3 are helix, strand, and coil, respectively. Given an input, pattern x , the computed output values of three neurons are 0.1, 0.2, 0.7, respectively. According to the WTA technique, 0.7 implies the highest likelihood, and the 2nd output

3.2. First Stage: the Q2T Prediction

neuron wins. The input pattern is thus predicted to be in the secondary structure state associated with second output neuron - strand.

3.2.2 Sequence Representation

Local Coding Scheme

The sequence of the protein is a series of alphabetical characters. To use it as an input to the neural network, it is necessary to code them into numerical forms first before the network could perform manipulations on them. Early works of the PSSP problem used to apply the binary coding schemes to fulfill this task. Among them, one typical encoding scheme used is the orthogonal data representation [34] which can be dubbed as BIN21. Under this scheme, each amino acid is represented by a 21-dimensional vector. The first 20 entries (bits) of the vector correspond to the occurrences of 20 types of the amino acid at the specific sequence position. If the amino acid to be coded is of type i , then the i -th entry of the vector will be coded as 1, while the remaining 19 bits remain 0. The additional bit of the vector, 21st bit, is called a *spacer*. As mentioned early, each input data is a sliding window of N , there may be cases where the leading or the lagging part of the window may be missing, e.g., the starting residues or the terminating residues of a protein sequence. In this case, the spacer would be able to indicate the completeness of the window.

Position-Specific-Scoring-Matrix (PSSM)

The problem of the BIN21 encoding scheme is that it does not incorporate the evolutionary information which has been shown to provide input domain-specific knowledge in the protein secondary structure prediction (See Chapter 2 Section 2.2). To address this problem, a multiple sequence alignment method is applied for the coding of each residue. To give a simple concept on how multiple sequent

3.2. First Stage: the Q2T Prediction

alignment is, let us consider a pairwise alignment case, without even taking gap insertions into consideration: Given two sequences, for instance, “GDEEKRNRAI” and “GDSDKNSAAI”, they can be aligned as illustrated in Figure 3.4. Assume that a score of +3 is awarded to one column having residue match (e.g., Column 1) and a penalty score of -3, otherwise (e.g., Column 3). The aggregated score (6) of all the columns is an *alignment score* raised from this alignment. Note that these two sequences could have many other possible alignments when gaps are allowed to insert into sequences and this results more alignment scores. Among them, the alignment resulting the highest score is called the *optimal alignment* and this shall give a measurement of the similarity between two sequences. While the pairwise

Sequence Alignment Example										
Original Seq.:	G	D	E	E	K	R	N	R	A	I
Aligned Seq.:	G	D	S	D	K	N	S	R	A	I
<hr/>										
Score:	+3	+3	-3	-3	+3	-3	-3	+3	+3	+3
	= 6									

Figure 3.4: An example illustrating the fundamental sequence alignment theory

sequence alignment is used to determine the similarity between two strings, the multiple sequence alignment is used to determine the similarity among multiple strings, i.e., searching for the optimal multiple alignment.

A position-specific scoring matrix (PSSM) is a matrix based on the amino acid frequencies at every position of a multiple sequence alignment. For example, the following three sequences (or segments of protein sequences) have been aligned as shown:

NTEGEWI

NITRGEW

NIGGECC

3.2. First Stage: the Q2T Prediction

The frequencies at C o l u m 2 for amino acid type 'T' and 'I' are 0.33 and 0.67, respectively. The frequencies for the remaining 18 amino acid types are all 0. However, in order to model every possible sequence, amino acids that do not appear at this column of the multiple alignment shall also be considered and assigned a value called pseudo-count. This results a 20-dimensional frequency vector with 20 non-zero entries for Column 2 sequence position. Each vector entry corresponds to one of the 20 types of amino acid. Taking the logarithm of each vector entry over the background frequency (e.g., this may be “a priori” probability of the corresponding amino acid over the entire protein sequence database) gives the final score called *log-odds score*. The log-odds score show how conservative one amino acid is at its sequence position. Highly conserved positions receive higher scores, weakly conserved positions are scored lower. As each sequence position can be represented by one 20-dimensional vector, the entire protein sequence can thus be represented by a $N \times 20$ *position specific scoring matrix*, where N is the length of the sequence. Hence, in this example, each of the 3 protein sequences can be represented by a 7x20 PSSM. More details can be referenced from [49] [50] [43] [44]

As mentioned early, the neighboring residues have impact on the formation of protein secondary structures. It is important to incorporate this information. For this purpose, a sliding window spanning over N residues with the residue of observation located at the center may be used. Thus each data (residue) is actually represented by an $N \times 20$ matrix. In this sliding window, each window position shares different degree of importance in determining the secondary structure state. Figure 3.5 shown below illustrates the raw data format for the 10-th residue Aspartic acid (D) of the Hydrolase (O-Glycosyl) protein sequence (PDB ID: 119L) using the sliding window size $N = 15$. In this representation, the central residue extends $(N - 1)/2 = 7$ leading/lagging neighboring amino acids and forms a matrix of

3.2. First Stage: the Q2T Prediction

$N \times 20$ entries. Moreover, one more spacer is added to indicate the completeness of the window. Note that the spacer is only used as the window status indicator and does not contribute to any numerical manipulations regarding the PSSM. In the example given, the spacer 0 implies that both its 7 leading and 7 lagging residues of the window are available and the window is complete. However, if the focus of observation is on the first residue (position -7) of the window, with sequence alphabet 'I', the spacer value is -5. This means that for the window centered at that residue, 5 amino acid residues ($(N - 1)/2$ residues - residue 'M' residue 'N' $= 5|_{N=15}$) are missing and the minus sign indicates that the missing residues are located at the leading part of the window

Figure 3.5: Input data representation: the raw profile (15 x 21) corresponds to the 10-th residue of the Hydrolase protein sequence.

Based upon the raw matrix obtained, the data is further normalized to the

3.2. First Stage: the Q2T Prediction

range (0,1) by using the standard logistic function:

$$x' = \frac{1}{1 + e^{-x}}. \quad (3.3)$$

After scaling, the ultimate input data changes to the form as in Figure 3.6

	A	R	N	D	C	Q	...	P	S	T	W	Y	V	
-7 ---	0.26894	0.04743	0.26894	0.11920	0.26894	0.04743	...	0.04743	0.26894	0.11920	0.04743	0.04743	0.50000	-5.00000
-6 ---	0.50000	0.04743	0.04743	0.01799	0.11920	0.04743	...	0.01799	0.50000	0.11920	0.11920	0.50000	0.50000	-4.00000
-5 ---	0.11920	0.73106	0.88080	0.88080	0.01799	0.73106	...	0.11920	0.50000	0.73106	0.01799	0.04743	0.04743	-3.00000
-4 ---	0.11920	0.73106	0.04743	0.01799	0.11920	0.26894	...	0.04743	0.11920	0.11920	0.11920	0.11920	0.50000	-2.00000
-3 ---	0.11920	0.04743	0.01799	0.01799	0.11920	0.04743	...	0.01799	0.04743	0.11920	0.04743	0.11920	0.73106	-1.00000
-2 ---	0.50000	0.98201	0.26894	0.11920	0.04743	0.50000	...	0.11920	0.26894	0.11920	0.01799	0.04743	0.50000	0.00000
-1 ---	0.11920	0.88080	0.98201	0.73106	0.26894	0.26894	...	0.04743	0.50000	0.26894	0.01799	0.04743	0.26894	0.00000
0 ---	0.50000	0.04743	0.26894	0.98201	0.04743	0.26894	...	0.11920	0.26894	0.11920	0.04743	0.26894	0.50000	0.00000
1 ---	0.11920	0.26894	0.26894	0.50000	0.00669	0.50000	...	0.11920	0.26894	0.11920	0.01799	0.04743	0.04743	0.00000
2 ---	0.26894	0.04743	0.26894	0.11920	0.01799	0.04743	...	0.04743	0.26894	0.04743	0.04743	0.01799	0.01799	0.00000
3 ---	0.26894	0.04743	0.04743	0.04743	0.99909	0.11920	...	0.01799	0.11920	0.11920	0.04743	0.11920	0.50000	0.00000
4 ---	0.26894	0.99753	0.26894	0.11920	0.04743	0.50000	...	0.04743	0.73106	0.26894	0.01799	0.04743	0.26894	0.00000
5 ---	0.11920	0.88080	0.04743	0.01799	0.11920	0.11920	...	0.50000	0.11920	0.50000	0.04743	0.73106	0.26894	0.00000
6 ---	0.26894	0.50000	0.73106	0.88080	0.04743	0.50000	...	0.11920	0.73106	0.50000	0.01799	0.04743	0.11920	0.00000
7 ---	0.88080	0.04743	0.04743	0.04743	0.11920	0.11920	...	0.99331	0.26894	0.26894	0.01799	0.04743	0.73106	0.00000

Figure 3.6: Input data representation: the normalized profile

3.2.3 Distance Measure Method for Data - WINDist

The distance for any two data points in the vector form is generally computed by the Euclidean Distance, assuming that each attribute of the vectors shares equal importance. In this clustering problem, however, each data point is represented in the form of matrix, and each row (i.e., window position) of the matrix shares different degree of importance. Therefore, a data specific distance measure should be used. Here, a unique distance measure WINDist, which is based on the Euclidean Distance and meanwhile takes the data characteristics into the consideration, is proposed. To compute the distance between two data points, the basic idea of WINDist is as follow: Given two data points (matrices) A and B, firstly match the two data row by row, e.g., first row of matrix A matches with that of matrix B while last row of matrix A matches with that of matrix B; secondly, compute

3.2. First Stage: the Q2T Prediction

the distance between two row vectors using the Euclidean Distance. Finally the weighted sum of the those row distances is used as the final distance measure.

What followed is a more detailed description of WINDisit.

Denote the i -th data point of a given sequence in the following form:

$$\mathbf{D}_i = [r_{-(N-1)/2}^i, r_{-(N-1)/2+1}^i, \dots, r_0^i, \dots, r_k^i, \dots, r_{(N-1)/2-1}^i, r_{(N-1)/2}^i], \quad (3.4)$$

where r_k^i is a 20-dimensional vector corresponds to k -th window position, k is either a positive or negative integer in the range $[-(N-1)/2, (N-1)/2]$. To compute the distance of D_i to another data point D_j , calculate the distance between any two matching rows using the Euclidean distance as follow:

$$d_k^{i,j} = \| r_k^i - r_k^j \|, \quad (3.5)$$

where $\| \bullet \|$ denotes the Euclidean norm. As the central position (represented by r_0^i) makes most important contribution to the prediction while the importance of the neighboring ones degrades with position, a weighting factor is assigned to each position. Again, let k denotes the window position. If $k = 0$, it means that this corresponds to the central position and the weight associated with it is most important, i.e., a weighting factor of 1. For the other position, it is assumed that *the significance of these residues degrades with their displacements to the central position*.

For the ease of explanation, let us first define a term $p(ab)$ addressing the relative displacement of two window positions with corresponding residue appearances a and b , respectively:

$$p(ab) = \text{displacement of } b \text{ to central residue} - \text{displacement of } a \text{ to central residue}.$$

From the definition, it is obvious that the p value is a relative difference of “closeness” to the position in centre. In Figure 3.2.3, the p value $p(E, L) = 4$ for residue

3.2. First Stage: the Q2T Prediction

Figure 3.7: Relative displacement of the residues

‘E’ and ‘L’ means that ‘E’ is 4 positions “closer” to the central residue than ‘L’. Now there shall be a method defined to determine the weighting factor associated with each row of the matrices. Assume that the degrading of the significance of each row vector is in a unique ratio b , or name it *base*. The base value should be within the range $[0,1]$. If the base value is 1, it is equivalent to the case where no weighting is used and all windows positions share the importance equally. The two alternative weighting schemes were proposed:

1. Scheme A: unsupervised weighting scheme. The weighting factor degrades in a geometric proportion with respect to the displacement. In other words, the weighting factor associated with residue r could be formulated as

$$w_r = b^{|p(r,c)|}, \quad (3.6)$$

where $p(r, c)$ is the displacement of residue r with respect to the central residue c .

2. Scheme B: supervised weighting scheme. For two residues r^1, r^2 with $p(r^1, r^2) = 1$, the weighting factor degrades if and only if the corresponding secondary structures are different. The reasoning of this weighting scheme is as follow: the secondary structure of a protein is usually formed by some consecutive amino acid residues. The consecutive residues sharing the same secondary structure states may contribute *equally* to the formation of the secondary

3.2. First Stage: the Q2T Prediction

structure and hence are of equal importance. Figure 3.8 illustrates how the weighting factors are assigned for the given the sequence and the secondary structure pair, “IFEMLRIDEGLRLKI” and “HHHHHEEEEEHHHHHC”, respectively.

Sequence:	I	F	E	M	L	R	I	D	E	G	L	R	L	K	I
Secondary Structure:	H	H	H	H	H	E	E	E	E	H	H	H	H	H	C
Weighting Factor:	b	b	b	b	b	1	1	1	1	b	b	b	b	b	b ²

Figure 3.8: Supervised weighting factor assignment: Scheme B

With the Euclidean distances computed on each row vectors, the aggregation of all these distances from the N residue positions gives the final estimation of the distance measure:

$$d(r^i, r^j) = \frac{\sum_{k=-(N-1)/2}^{(N-1)/2} (w_k \| r_k^i - r_k^j \|)}{\sum_{k=-(N-1)/2}^{(N-1)/2} w_k}, \quad (3.7)$$

where w_k is the weighting factor obtained by Scheme A or Scheme B.

Up to now, the distance measure deals with the situation that the both data points have full sliding window. However, there are situations where the sliding window might not be full, e.g., the starting or the terminating $(N-1)/2$ residues of the sequence. In this case, only the overlapped part of two windows is considered. Equation 3.7 is then generalized to the following form:

$$d(r^i, r^j) = \frac{\sum_{k=-l}^h (w_k \| r_k^i - r_k^j \|)}{\sum_{k=-l}^h w_k}, \quad (3.8)$$

where l is the number of overlapped positions in the leading half-window and h is the number of overlapped positions in the lagging half-window. For complete

3.3. Second Stage: the T2T Prediction

window, both the numbers of overlapped leading and lagging vectors are equal to $(N - 1)/2$ and hence Equation 3.7 is a special case of Equation 3.8

3.3 Second Stage: the T2T Prediction

3.3.1 Architecture

The second stage of the predictor is a three-layer MLPNN. Similar to the RBFNN, the network consists an input layer, a hidden layer and an output layer. Although the backpropagation technique allows any number of layers of the network to be trained, a three-layer structure is sufficient for approximating any function with finitely many discontinuities to the arbitrary precision, provided the activation functions of the hidden units are non-linear (the universal approximation theorem) [51] [52] [53] [54]. The architectural layout of the network is shown in Figure 3.9.

Figure 3.9: Architecture of the structure-structure prediction: the second stage

The input layer is where the windowed data is fed in. The number of neurons required depends on the number of numerical entries in the windowed data and is $N \times L$ in this problem, where N is the window size and L is the vector dimension.

The input to the *second layer* is the weighted sum of the input data:

3.3. Second Stage: the T2T Prediction

$$s_k^{(l)} = \sum_{i=1}^{m_{l-1}} (w_{ki}^{(l)} y_i^{(l-1)} + \theta_{l-1}), \quad (3.9)$$

where $w_{ki}^{(l)}$ is the weight connecting k -th neuron in layer $l = 2$ and i -th neuron in layer $l - 1 = 1$ and θ_{l-1} is the bias feedforwarded from layer $l - 1 = 1$. $y_i^{(l-1)}$ is the output of the i -th neuron at layer $l - 1$, i.e., the input layer. m_{l-1} is the number of the neurons in the layer $l - 1$.

With the input computed, the output or the activation of the neuron is obtained by a sigmoid function:

$$y_k^{(l)} = \frac{1}{1 + e^{-s_k^{(l)}}}, \quad (3.10)$$

where $y_k^{(l)}$ is the activation of $s_k^{(l)}$ in the k -th neuron of the l -th layer. The behavior of a sigmoid function with scalar input x is showed in Figure 3.10.

Figure 3.10: Response of the sigmoid function

Transformed by the sigmoid function, the inputs with large positive values are activated closely to 1 while large negative ones to 0. For the inputs close to 0, the

3.3. Second Stage: the T2T Prediction

responses are linearly related to the input values. The outputs of the second layer neurons are thus smoothly restricted to the range (0,1).

The *output layer* functions exactly the same as the hidden layer. The input to this layer is the weighted sum of outputs of the neurons from the proceeding layer. A sigmoid function is then used to smoothly restrict the output to the desirable range.

By comparing the RBFNN and the MLPNN, it can be seen that the networks are quite similar. The data is propagated layer by layer, from the input layer to the output layer. Each layer performs some transformation to the data before it is forwarded. The output layer constitutes three neurons each corresponding to one secondary structure class.

Despite the similarities mentioned above, the MLP network still possesses some variations in several aspects:

- The input neuron of the MLPNN as in Figure 3.9 is an assembly of $N \times L$ subneurons, with each subneuron connecting to all the neurons of the hidden layer. The capital letter L here represents the dimensionality of the vector of each residue.
- There are weights associated with the interconnections between the input layer and the hidden layer neurons. For the RBFNN, there is no weights between these two layers.
- The activation function used in the hidden layer is different from that of the RBFNN.
- In the RBFNN, the weighted sum of the activations from the hidden layer is directly used as the output. In the MLPNN, the additional smooth limiting is performed by the output neurons using the sigmoid function.

3.3. Second Stage: the T2T Prediction

Due to the architectural difference, the encoding of the data is slightly different from that of RBFNN.

3.3.2 Sequence Representation

Input preprocessing

The output of the first stage is a three-dimensional vector, with each entry representing how likely that the residue-in-observation belongs to the corresponding secondary structure class. In other words, they could be interpreted as the probabilities that the residue belongs to the three classes. Thus, it is expected that the values are to be restricted within (0,1) and the sum of them shall reach 1. Besides, it is also desirable that the data fed into the MLPNN to be in that range. Therefore, a preprocessing of the data is performed before the outputs of the first stage are directly used as the inputs of the MLP network. In this preprocessing, it is important that the relative likelihood of the three classes should be maintained and summed to 1. To satisfy these constraints, the normalized exponential transformation technique softmax [55] is used:

$$Q_j(x) = \frac{e^{V_j(x)}}{\sum_{k=1}^3 e^{V_k(x)}}, \quad (3.11)$$

where $V_j(x)$ is the j -th entry of output vector and $Q_j(x)$ is the transformed value.

Encoding Scheme

Considering the correlations existing between neighboring residues, a sliding window similar to that in the RBFNN also needs to be used to represent the data. As mentioned, there may be cases where the window is not complete. On the other hand, the input layer of the proposed MLPNN architecture consists of a fixed number: $N \times L$ neurons, which means that the network could only accept the data with full sliding window. To resolve the conflict, some modifications have to be

3.3. Second Stage: the T2T Prediction

done to the data before they could actually be used. Similar to the RBFNN case, a spacer is used as an additional entry to the three-dimensional vector. The spacer is an indicator of the window status. For each window, if the vector corresponds to the window position contains realistic values (e.g., have the occurrence of sequence residue at that position), the spacer is set to 0. Otherwise, the spacer is set to 1. In case where the window position is not realistic, the artificial values would have to be filled. Here, the scheme used in Rost and Sander's [18] work is adopted: all the entries are set to 0 except the spacer. Figure 3.11 shows the encoded form of the data for the full window case (a) and the partial window case (b). A window size of $N = 15$ is used in the illustration.

→	I	0.1158	0.3268	0.5574	0
	F	0.1047	0.1914	0.7038	0
	E	0.1067	0.1557	0.7377	0
	M	0.0685	0.1992	0.7324	0
	L	0.1096	0.2389	0.6515	0
	R	0.2047	0.2083	0.5870	0
	I	0.1150	0.2024	0.6827	0
	D	0.1300	0.1929	0.6771	0
	E	0.1404	0.2683	0.5913	0
	G	0.1678	0.4289	0.4034	0
	L	0.1030	0.6367	0.2603	0
	R	0.0585	0.7338	0.2077	0
	L	0.1169	0.5763	0.3069	0
	K	0.1617	0.2309	0.6074	0
	I	0.1064	0.2207	0.6729	0

(a)

		0	0	0	1
		0	0	0	1
→	I	0.1067	0.1557	0.7377	0
	F	0.0685	0.1992	0.7324	0
	E	0.1096	0.2389	0.6515	0
	M	0.2047	0.2083	0.5870	0
	L	0.1150	0.2024	0.6827	0
	R	0.1300	0.1929	0.6771	0
	I	0.1404	0.2683	0.5913	0
	D	0.1678	0.4289	0.4034	0
	E	0.1030	0.6367	0.2603	0
	G	0.0585	0.7338	0.2077	0
	L	0.1169	0.5763	0.3069	0
	R	0.1617	0.2309	0.6074	0
	L	0.1064	0.2207	0.6729	0

(b)

Figure 3.11: Encoded form of the data: a) the full window, b) the partial window

Similar to that in the RBF network, the data should also be expanded on its neighborhood before it can be used to set up the network or do the prediction.

Chapter 4

Training of the Hybrid Neural Networks Predictor (HNNP)

This chapter describes the setup of the HNNP system, which in essential, is the optimization of the neural network parameters. As the neural network architectures employed for the sequence-structure and the structure-structure prediction are different, the mathematical background of optimization for each network will be explained.

4.1 Training of the RBFNN

The training of the RBF network is to find the kernels of the hidden layer and the weights connecting the hidden and the output layer. The training of the two parameter sets, however, can be decoupled due to the local receptive field nature of the hidden units. This strategy has been shown to be very effective in terms of the training speed while compared to the MLP network.

4.1.1 Hidden Layer Kernel Construction

The construction of hidden layer kernels requires the setting up of two parameter sets: the center of the kernel, C and the kernel width, σ . Basically, the center

4.1. Training of the RBFNN

C is a $N \times 20$ matrix which carries the information similar to that of the input data. The kernel width σ defines the span of the Gaussian kernel. Typical method for finding these parameters is the *k-means clustering* technique. Yet k-means clustering has the problem in that the user has to blindly predefine the number of kernels without exactly knowledge of the input space. Moreover, k-means is weak in rejecting noise.

To overcome these problems, an overlapped RBF clustering algorithm [48] is applied. The basic idea of the overlapped clustering is to allow the large overlaps for the clusters of the same class and only small overlaps for the clusters of the different classes (see Figure 4.1). By making use of this algorithm, the number of kernels grows with the clustering process and the overlapped receptive nature between different clusters can improve the performance of the RBFNN classifier by rejecting noise [56]. Due to the large dataset used in this project, no limit is set for the minimum number of patterns of the clusters and the cluster purity is fixed for each epoch of the training. These modifications have helped to reduce the time spent on the clustering. Moreover, during the overlapped clustering process, there is always possibility that the early formed cluster is a subset of the later one. To remove the subclusters, all the clusters are scanned for cluster members after the clustering process is finished and any subclusters found are deleted.

Figure 4.1: Overlapped clustering: large overlaps for clusters of the same class and small overlaps for those of different classes

By accommodating these modifications, the clustering algorithm used for find-

4.1. Training of the RBFNN

ing kernels is as follow:

1. Generate the duplicated dataset V , the center pool set P , as copies of the original training data V . Set σ , the expected maximum distance of the temporary cluster, to σ_0 ;
2. Discard all the data points with only partial window from the center pool set P . This practice allows the randomly selected data point as the initial center to have the full window. And this in turn ensures that the formed clusters are in the form of full window;
3. Randomly select a data point from P as the initial center. Set the iteration number $I = 1$;
4. Search in V for all the patterns that are within the σ -neighborhood, where σ is the expected maximum distance of the temporary cluster. Any patterns with distances to the randomly,selected data less than σ are grouped into the temporary cluster C_t . Check the cluster purity ρ_t of the temporary cluster. If ρ_t is larger than some threshold ratio p , save the cluster, remove the cluster members from V_c and P , go to Step 5; Otherwise, $\sigma = \sigma - \Delta$, $I = I + 1$. Here Δ is the change of σ . If I is less than the maximally allowed iteration number, repeat Step 4; Otherwise go to Step 5;
5. Reset the expected maximum distance: $\sigma = \sigma_0$. Repeat Step 3 and 4 until V_c is empty;
6. Scan all clusters for subclusters. If the early formed cluster is a subset of the later one, delete the early one;
7. Calculate the center and the width of each cluster: the center is the mean of all data members and the width is the stand deviation.

4.1. Training of the RBFNN

4.1.2 Weights Estimation

Learning of the weights connecting the hidden layer and the output layer can be done by iteratively applying the delta rule. The iterative updating of weights can be formulated by the following equations:

$$\Delta w_{ij}(k) = -\eta(y_i^p(x_p) - t_i^p)h_j^p(x_p), \quad (4.1)$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k), \quad (4.2)$$

where $h_j^p(x_p)$, $y_i^p(x_p)$, t_i^p are the output of the j -th hidden unit, the output of the i -th output unit and the target, respectively, when the p -th input data x_p is present. $w_{ij}(k)$ is the weight connecting i -th output unit and j -th hidden unit and $\Delta w_{ij}(k)$ is the change of $w_{ij}(k)$. The constant η is the learning rate.

The training stops at where the Least Mean Square (LMS) error is reached.

Alternatively, in this project, the faster and simpler weight estimation method which makes use of the generalized linear regression technique is adopted. The method takes advantages over the delta rule by simple matrix manipulation and faster estimation of unique weight that satisfies the LMS criterion.

To simplify the analysis, consider the situation where there is only one output unit. Given a set of input-output pair $\{(x_i, t_i)\}_{i=1}^P$, for any hidden layer unit j , the cost function to be minimized is stated as below [57]:

$$C = \frac{1}{2} \sum_{i=1}^P (t_i - y(x_i))^2 + \frac{1}{2} \sum_{j=1}^m \lambda_j w_j^2, \quad (4.3)$$

where

$$y(x_i) = \sum_{j=1}^m w_j h_j(x_i) \quad (4.4)$$

is the output of the third layer. $h_j(x_i)$ denotes the response of the hidden unit j when input data x_i is present and m is the total number of hidden layer neurons.

4.1. Training of the RBFNN

The regularization parameter $\lambda_j > 0$ controls the balance between fitting the data and the penalization of large weights [57].

Differentiate the cost function C with respect to weight w_j , set the left hand side to zero and rearrange the equation, it gives

$$\sum_{i=1}^P y(x_i) h_j(x_i) + \lambda_j w_j = \sum_{i=1}^P t_i h_j(x_i). \quad (4.5)$$

To find that unique solution, the language of matrices and vectors, linear algebra, may be employed. The above equation hence can be rewritten in the vector notation as:

$$\mathbf{h}_j^T \mathbf{t} = \mathbf{h}_j^T \mathbf{y} + \lambda_j w_j, \quad (4.6)$$

where

$$\mathbf{h}_j = [h_j(x_1), h_j(x_2), h_j(x_3), \dots, h_j(x_P)]^T,$$

$$\mathbf{t} = [t_1, t_2, t_3, \dots, t_P]^T,$$

$$\mathbf{y} = [y(x_1), y(x_2), y(x_3) \dots y(x_P)]^T$$

and T represents transpose. Combine each j from 1 to m results:

$$\begin{bmatrix} \mathbf{h}_1^T \mathbf{y} \\ \mathbf{h}_2^T \mathbf{y} \\ \vdots \\ \mathbf{h}_m^T \mathbf{y} \end{bmatrix} + \begin{bmatrix} \lambda_1 w_1 \\ \lambda_2 w_2 \\ \vdots \\ \lambda_m w_m \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T \mathbf{t} \\ \mathbf{h}_2^T \mathbf{t} \\ \vdots \\ \mathbf{h}_m^T \mathbf{t} \end{bmatrix}. \quad (4.7)$$

In matrix form, this is just equivalent to

$$\mathbf{H}^T \mathbf{y} + \mathbf{\Lambda} \mathbf{w} = \mathbf{H}^T \mathbf{t}, \quad (4.8)$$

where

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_m), \quad (4.9)$$

and

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_m] \quad (4.10)$$

4.1. Training of the RBFNN

is called the *design matrix*.

Moreover, it can be easily derived that

$$\mathbf{y} = \mathbf{H}\mathbf{w}. \quad (4.11)$$

Substitute equation (4.11) back into (4.8):

$$\begin{aligned} \mathbf{H}^T \mathbf{t} &= \mathbf{H}^T \mathbf{y} + \Lambda \mathbf{w} \\ &= \mathbf{H}^T \mathbf{H} \mathbf{w} + \Lambda \mathbf{w} \\ &= (\mathbf{H}^T \mathbf{H} + \Lambda) \mathbf{w}. \end{aligned} \quad (4.12)$$

The solution of the weight vector is thus

$$\mathbf{w} = (\mathbf{H}^T \mathbf{H} + \Lambda)^{-1} \mathbf{H}^T \mathbf{t}. \quad (4.13)$$

Further more, for ordinary least squares where there is no weight penalty, the diagonal components of the regularization matrix Λ are set to zeros and the weight vector equation is reduced to

$$\mathbf{w} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{t}. \quad (4.14)$$

In this project, there are three output units and thus the weights are embodied by an $m \times 3$ matrix $\mathbf{W}$ and the ultimate weight matrix is computed using the following equation:

$$\mathbf{W} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}, \quad (4.15)$$

where

$$\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3] \quad (4.16)$$

is the *target matrix*. Note the expression given in the equation (4.15) is actually the Moore-Penrose pseudoinverse solution to this least square problem.

4.2. Training of the MLPNN

4.2 Training of the MLPNN

The proposed MLPNN is a three-layer neural network. In this network, the two weight matrices inter-connecting the neighboring layers are unknown and need to be trained.

4.2.1 The Delta Rule

For the simplicity of discussion, let us first consider a single-layer network with one output unit with a linear activation function is considered. The output of the neuron is given by

$$y_k = \sum_j w_j x_k + \theta, \quad (4.17)$$

where y_k is the actual output obtained when the input pattern x_k is fed into the network. Suppose the expected output is t_k . The purpose of the training is that after tuning variables w_j into the proper value, the difference (error) between the actual and desired outputs is minimize. If a total of P patterns are to be used for training, the cost function needs to be minimized is the sum of the squared error - the Least Mean Square (LMS) problem:

$$E = \sum_{k=1}^P E_k = \frac{1}{2} \sum_{k=1}^P (t_k - y_k)^2. \quad (4.18)$$

The LMS procedure finds the values of all the weights that minimize the error function by a method called *gradient descent*. The idea is to change the weight proportional to the negative of the derivative of the error as measured on the current pattern with respect to that weight:

$$\Delta_k w_j = -\gamma \frac{\partial E_k}{\partial w_j}, \quad (4.19)$$

where γ is the learning rate. The derivative is:

$$\frac{\partial E_k}{\partial w_j} = \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial w_j}. \quad (4.20)$$

4.2. Training of the MLPNN

Because

$$\frac{\partial y_k}{\partial w_j} = x_k \quad (4.21)$$

for the linear units, and

$$\frac{\partial E_k}{\partial y_k} = -(t_k - y_k), \quad (4.22)$$

the update quantity of the weight w_j is

$$\Delta_k w_j = \gamma x_k (t_k - y_k). \quad (4.23)$$

The delta rule tunes the weight iteratively when patterns are passed through the network one by one.

4.2.2 Backpropagation

The delta rule derived from the single layer network could be generalized to train the MLPNN. The weights connecting the hidden and the output layer could be obtained by applying the delta rule directly, as the targets have already been defined. On the other hand, it is still not known what are the desired outputs of the hidden layer neurons and the delta rule can not be applied. To overcome this problem, a *back-propagation* technique has been proposed by [58]. The technique could learn more than two layers of network. The key idea of the back-propagation technique is that the error obtained from the output layer is propagated backward to the hidden layer and is used to guide the training of the weights between the hidden layer and the input layer.

Figure 4.2 shows a 3-layer MLPNN with the sigmoid function as the activation function for the hidden layer and the output layer. In the previous section, the delta, **rule** is used for training one-layer network with the linear response. Here, the rule has to be generalized to work with the non-linear activation.

The activation function of the neural network is a differential function of the

4.2. Training of the MLPNN

Figure 4.2: Architecture of the 3-layer multi-layer perceptron network.

4.2. Training of the MLPNN

total inputs, given by

$$y_i^k = \psi(s_i^k). \quad (4.24)$$

The input of the activation function $\psi(\bullet)$ is the weighted sum of the outputs of the proceeding layer when the input pattern x_k is present:

$$s_i^k = \sum_j w_{ij} y_j^k + \theta_i. \quad (4.25)$$

The total error E^k returned by the output units of the network is defined as

$$E^k = \frac{1}{2} \sum_{o=1}^{N_o} (t_o^k - y_o^k)^2, \quad (4.26)$$

where t_o^k, y_o^k are the desired and the actual output of the output unit o . N_o is the total number of output units. By chain rule,

$$\frac{\partial E^k}{\partial w_{ij}} = \frac{\partial E^k}{\partial s_i^k} \frac{\partial s_i^k}{\partial w_{ij}} \quad (4.27)$$

From Equation 4.25, it could be derived easily that

$$\frac{\partial s_i^k}{\partial w_{ij}} = y_j^k. \quad (4.28)$$

Define

$$\delta_i^k = -\frac{\partial E^k}{\partial s_i^k}, \quad (4.29)$$

substitute Equation 4.28 into Equation 4.27, a gradient descent can be obtained on the error surface if the change of the weight is

$$\Delta_k w_{ij} = \gamma \delta_i^k y_j^k, \quad (4.30)$$

where γ is the learning rate. A small learning rate for the training leads to a slow convergence while a large learning rate without inducing oscillations on the error surface speeds up the convergence.

The problem left with us now is to figure out what δ_i^k should be. Let us first, focus on the weights connecting the last two layers, i.e., the hidden layer and the

4.2. Training of the MLPNN

output layer. In the output layer, the error function is known. By chain rule, the derivative of the error with respect to the input of output unit o can be written as:

$$\delta_o^k = -\frac{\partial E^k}{\partial y_o^k} \frac{\partial y_o^k}{\partial s_o^k}. \quad (4.31)$$

Obviously, the second term in the above equation is simply the derivative of the activation function $\psi(\bullet)$. Also, with the error function defined in Equation 4.26, the derivation of error with respect to the output is easily obtained:

$$\frac{\partial E^k}{\partial y_o^k} = -(t_o^k - y_o^k). \quad (4.32)$$

Substitute Equation 4.32 into Equation 4.31, it comes

$$\delta_o^k = (t_o^k - y_o^k) \psi'(s_o^k). \quad (4.33)$$

Hence, the change of weight connecting the hidden unit h and the output unit o can be formulated as:

$$\Delta_k w_{oh} = \gamma (t_o^k - y_o^k) \psi'(s_o^k) y_h^k. \quad (4.34)$$

Having successfully formulated the weight updating for the last two layers, the concentration now is on the weight between the input layer and the hidden layer. As there is no explicit error function available for the hidden layer, the error at the output layer is propagated backward to guide the training. Again, by chain rule, it can be shown that:

$$\frac{\partial E^k}{\partial y_h^k} = \sum_{o=1}^{N_o} \frac{\partial E^k}{\partial s_o^k} \frac{\partial s_o^k}{\partial y_h^k} = \sum_{o=1}^{N_o} \frac{\partial E^k}{\partial s_o^k} \frac{\partial}{\partial y_h^k} \sum_{j=1}^{N_h} w_{jo} y_j^k = \sum_{o=1}^{N_o} \frac{\partial E^k}{\partial s_o^k} w_{ho} = - \sum_{o=1}^{N_o} \delta_o^k w_{ho}. \quad (4.35)$$

As

$$\delta_h^k = -\frac{\partial E^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial s_h^k} \quad (4.36)$$

for the hidden unit h , the equation then becomes

$$\delta_h^k = \psi'(s_h^k) \sum_{o=1}^{N_o} t_o^k w_{ho}. \quad (4.37)$$

4.2. Training of the MLPNN

Hence, the change of the weight connecting the input unit i and the output unit h can be formulated as:

$$\Delta_k w_{hi} = \gamma \psi'(s_h^k) \sum_{o=1}^{N_o} t_o^k w_{ho} x_i^k. \quad (4.38)$$

With Equation 4.34 and 4.38, the weights between each layer could be recursively updated.

At this point, the training of the MLPNN could almost be done. However, there is still a term which has not yet been formulated, i.e., the learning rate γ . As mentioned early, a small learning rate results a slow convergence while a large one may lead to oscillations at the error surfaces. Thus, it is important that a suitable learning rate be selected for the fast convergence and meanwhile oscillations on the error surface shall be avoided. For this purpose, a momentum is introduced by taking account the effect of the previous weight changes. With this consideration, the weight updating equation finally becomes

$$\Delta w_{ji}(t+1) = \gamma \delta_i^k y_j^k + \alpha \Delta w_{ji}(t). \quad (4.39)$$

Chapter 5

Experiments

5.1 Experimental Dataset

In the protein structure prediction, if an unknown protein sequence shares significant degree of sequence similarity with the known protein sequence or is a homology, the structure of the unknown protein can be reliably predicted by the simple comparison and the reference between the two sequences [59] [60] [61] [44] [62] [63]. On the other hand, it is not always true in practice that a new sequence has its homology whose structure has been identified. Hence, to evaluate the performance of the prediction fairly, the bias of the prediction arisen from the homology has to be removed. Moreover, according to Cuff and Barton [38], sequences with low sequence similarity yet significant structural similarity can remain by the pairwise sequence alignment methods. To avoid this, the structural similarity also needs to be investigated so as to further remove the bias towards producing favorable prediction accuracy.

The dataset used in this project was originally developed and used by Jones [20] for training and testing his PSIPRED method. The data can be downloaded from the website <http://bioinf.cs.ucl.ac.uk/psipred/>. The PSIPRED dataset, contains 2235 protein sequences in total for the training and 187 sequences for the test. The dataset was derived by considering constraints for the fair prediction evaluation, as

5.2. Accuracy Measure

discussed early. All the 187 test sequences have been scanned against the training sequences to ensure that they did not share the similar fold. Jones believe such a practice is at least equivalent to the 7-fold cross-validation test and the testing result should not be over-estimated [20]. With such specially designed training and test datasets, no further 7-fold cross validation [15] [18] is needed to avoid the overestimation.

As the secondary structures given in the dataset consists of eight states: H, I, G, E, B, S, T, -, the scheme outlined by Rost and Sander [18] was adopted to reduce them into the three secondary structure states (α -helix, β -strand and coil). H and G are reduced to the helix state. E and B are reduced to the strand states and all others are assigned to the coil state.

5.2 Accuracy Measure

5.2.1 Per-residue Accuracy

The per-residue based accuracy measurement emphasizes on the fraction of individual residues that are correctly classified into their secondary structure states. The measure could either return an aggregate accuracy over all sequences or an average of the accuracies of all individual protein sequences. What followed is a mathematical description of the way the accuracy is derived. Given a protein sequence k , define

$$a_{ij} = \begin{array}{l} \text{number of residues predicted to be in secondary structure state} \\ \text{j and observed to be in state i.} \end{array} \quad (5.1)$$

Where $i, j \in \{1 = \text{helix}, 2 = \text{strand}, 3 = \text{coil}\}$. The sum of a_{ij} for i from 1 to 3 gives rise the number of residues predicted to be the class j :

$$P_j^k = \sum_{i=1}^3 a_{ij}. \quad (5.2)$$

5.2. Accuracy Measure

The sum of a_{ji} for i from 1 to 3 gives rise the number of residues observed to be class j :

$$O_j^k = \sum_{i=1}^3 a_{ji}. \quad (5.3)$$

The sum of all residues in the dataset is

$$N^k = \sum_{i=1}^3 \sum_{j=1}^3 a_{ij}. \quad (5.4)$$

For class $r \in \{\alpha = helix, \beta = strand, c = coil\}$, the accuracy achieved is thus computed as:

$$Q_r^k = \frac{a_{jj}}{O_j^k}, \quad (5.5)$$

where r, j are the indices representing the same class. The overall accuracy of the predict on for protein sequence k , denoted as Q_3^k , is the aggregated sum of the correct predictions over the total number of residues in the sequence:

$$Q_3^k = \frac{\sum_{j=1}^3 a_{jj} N^k}{N} \quad (5.6)$$

Hence, if the dataset contains p protein sequences with N amino acid residues in total, the prediction accuracy Q_3 for all the p sequences is:

$$Q_3 = \frac{\sum_{k=1}^p Q_3^k N^k}{N} \quad (5.7)$$

The average of the accuracies for all individual sequences can be computed as

$$Q_{3s} = \frac{\sum_{k=1}^p Q_3^k}{p} \quad (5.8)$$

5.2.2 Matthew's Correlation

A more complicated accuracy measure is to use the Matthew's Correlation [64] which evaluates the success of predicting residues for each secondary structure state. The formula for computing the Matthew's Correlation is given by:

$$C_i = \frac{p_i n_i - u_i o_i}{\sqrt{(p_i + u_i)(p_i + o_i)(n_i + u_i)(n_i + o_i)}}, \quad (5.9)$$

5.2. Accuracy Measure

for $i \in \{\alpha, \beta, c\}$.

p_i is the number of correctly predicted residues in the secondary structure state i ; n_i is the number of those correctly not assigned to the state i ; u_i is the number of underestimated and o_i is the number of overestimated predictions. Mathematically, these terms can be expressed as

$$\begin{aligned} p_i &= a_{ii}, \\ n_i &= \sum_{j \neq i}^3 \sum_{k \neq i}^3 a_{jk}, \\ o_i &= \sum_{j \neq i}^3 a_{ji}, \\ u_i &= \sum_{j \neq i}^3 a_{ij}. \end{aligned} \tag{5.10}$$

for $i \in \{\alpha, \beta, c\}$.

5.2.3 Segment Overlap Measure

Segment Overlap Measure (SOV) was first proposed by Rost et al in 1994 [65] and later modified and improved by Adam Zemla et al in 1999 [66]. This scheme gives a measure on how well the predicted secondary structure segments are matched with the observed (obtained from experiments) ones. The accuracy is computed by counting the predicted and the observed segments and measuring their overlaps. Let (s_1, s_2) denote a pair of overlapping segments. A set $S(i)$ is defined as

$$\begin{aligned} S(i) &= \\ &\{(s_1, s_2) : s_1 \cap s_2 \neq \emptyset, s_1 \text{ and } s_2 \text{ are both in the secondary structure state } i\}. \end{aligned} \tag{5.11}$$

to represent the set of all overlapping pairs of segments (s_1, s_2) . Another set which defines the set of all segments (s_1, s_2) sharing no overlapping in their segment extents, is defined as

$$\begin{aligned} S'(i) &= \\ &\{s_1 : \forall s_2, s_1 \cap s_2 = \emptyset, s_1 \text{ and } s_2 \text{ are both in the secondary structure state } i\}. \end{aligned} \tag{5.12}$$

5.3. Q2T Prediction - the raw prediction

The segment overlap measure for state i is thus defined as

$$SOV(i) = 100 \times \frac{1}{N(i)} \sum_{S(i)} \left[\frac{\minov(s_1, s_2) + \delta(s_1, s_2)}{\maxov(s_1, s_2)} \times \text{len}(s_1) \right], \quad (5.13)$$

with the normalization value

$$N(i) = \sum_{S(i)} \text{len}(s_1) + \sum_{S'(i)} \text{len}(s_1). \quad (5.14)$$

The sum in Equation 5.13 and the first sum in Equation 5.14 are computed from all the segment pairs in state i which overlap by at least one residue, the second sum in Equation 5.14 is computed from the remaining segments in state i found in the observed (or target) assignment, $\text{len}(s_1)$ is the number of residues in segment s_1 , $\minov(s_1, s_2)$ is the length of the actual overlap of s_1 and s_2 , i.e., for which both segments have residues in state i , $\maxov(s_1, s_2)$ is the total extent for which either of the segments s_1 and s_2 has a residue in state i and $\delta(s_1, s_2)$ is defined as:

$$\delta(s_1, s_2) = \min \begin{cases} (\maxov(s_1, s_2) - \minov(s_1, s_2)); \\ \minov(s_1, s_2); \\ \text{int}(\text{len}(s_1)/2); \\ \text{int}(\text{len}(s_2)/2). \end{cases} \quad (5.15)$$

Generalization based on Equation 5.13 results a 3-state SOV measurement:

$$SOV(i) = 100 \times \left[\frac{1}{N} \sum_{i \in [\alpha, \beta, c]} \sum_{S(i)} \frac{\minov(s_1, s_2) + \delta(s_1, s_2)}{\maxov(s_1, s_2)} \times \text{len}(s_1) \right] \quad (5.16)$$

where the normalization value N is the sum of $N(i)$ overall all three secondary structure states.

5.3 Q2T Prediction - the raw prediction

As discussed early, the prediction of the secondary structure is implemented in two stages: the sequence-structure prediction and the structure-structure prediction.

5.3. Q2T Prediction - the raw prediction

For the sake of easiness in implementation, the trainings of the two stages are performed independently. The first stage of the predictor is trained first and the predictions are made to the training data. Based on the predictions, the second stage uses the data to train the second stage of the predictor. Upon the accomplishment of the training, the test data may then be passed through the stages of the system. The output from the second stage determines the final secondary structure state assignment.

The Q2T prediction is implemented by a RBF neural network. To obtain the optimal performance of the prediction, several parameters related to the network topology, the kernel formation, etc., have to be selected through experiments. More precisely, these parameters include

1. The base ratio b . This term determines the degrading of the weighting factors in the proposed distance measure WINDist as described in Chapter 3.
2. The alternative WINDist Scheme A or B. The both Schemes are proposed on the basis of some assumptions. Experimental results have to be used to find the better Scheme.
3. The window size N . A small window size may lose the useful information associated with farther-neighboring residues. On the other hand, bringing residues too distant away into observation may introduce noises, as the information from those residues is usually irrelevant. A suitable window size which exploits the sufficient amount of relevant information and meanwhile is kept from noises is desirable.
4. Purity level p of the clusters (kernels). A higher p value results more accurate response from the kernel yet also reduces the generalization performance of the RBF network. Besides, the purity level also controls the performance

5.3. Q2T Prediction - the raw prediction

of the overlapped clustering algorithm. A high purity ensures that ONLY clusters of the same class could have large overlap. This practice helps to reject noises and improves the performance of the RBFNN.

Ideally, these parameters should be tuned together such that a combination which leads to the optimal performance of the Q2T prediction could be found. However, doing so would demand huge amount of computational burden and time, especially when the data used in this project is large in scale and high in dimensionality. According to our simulations, each trial of training takes 1-2 days to finish by the computational resource available. For this reason, a relative suboptimal yet time saving tuning scheme was adopted: change some set of parameter, fix the rest.

Now, It shall be determined which parameters are to be tuned in prior. Basically, the training of the RBF network starts by applying the overlapped clustering algorithm. The parameters affecting this early step are the base ratio b and the WINDist Scheme. The both variable would affect the distance measure between data points and hence influence the clustering. Thus, these two variables are selected in prior.

5.3.1 Experiments with the Base and Alternative Distance Measure Schemes

Strategy

To determine the suitable base ratio b and the best WINDist scheme, simulations against two WINDist Schemes (A, B) were carried out independently. For each WINDist Scheme, the base value b is varied from 0.3 to 0.7 with a 0.1 of change at each time. The purpose of doing so is first to find the best base value for each scheme and then select the scheme which outperforms the other.

5.3. Q2T Prediction - the raw prediction

Results

Table 5.1 shows the experimental results.

base (b)	Scheme A				Scheme B			
	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
0.3	74.0	74.9	56.6	82.7	74.6	75.6	57.7	82.8
0.4	74.5	75.5	56.5	83.3	74.9	75.8	58.0	83.4
0.5	74.7	75.6	56.9	83.5	75.2	76.2	57.9	83.7
0.6	74.4	75.3	56.4	83.3	75.1	76.0	57.9	83.6
0.7	74.2	75.4	56.1	82.8	74.6	76.1	56.8	82.5

Table 5.1: Performance comparison of two WINDist schemes: the base value b is varied within the range [0.3 0.7] with 0.1 of change at each time

It can be seen by observing the table that the general performance of Scheme B is superior to that of Scheme A. The highest Q_3 accuracy Scheme B could reach is 75.2% when the base value is set to 0.5. It happens that the highest Q_3 of Scheme A is also reached at the same base yet the accuracy is only 74.7%. This accuracy is lower than that of Scheme B. Comparison of Q_3 values at other base values also shows that Scheme B is superior to Scheme A. Thus, the supervised WINDist Scheme is more appropriate for the proposed system and the all further simulations would use this Scheme.

Besides, by observing the " Q_3 " column of Scheme B, It could be seen that base values either larger or smaller than 0.5 degraded the prediction accuracies. Expectedly, this base value has more relevantly reflected the degrading of the significance of the neighboring residues and is more relevant to the proposed WINDist.

5.3. Q2T Prediction - the raw prediction

5.3.2 Experiments with the Window Size and the Cluster Purity

Strategy

By deciding the optimal value for the base and the appropriate scheme for the distance measure in prior, The computational demand has been reduced effectively such that it is affordable for the available computational resources. Now there are two major parameters left to be determined:

- The window size which determines the span of the sliding window.
- The cluster purity which decides the minimum purity that a cluster (kernel) has to satisfy.

To get the best performance of the first stage, different combinations of these two parameters have to be investigated. To carry out the simulations orderly, the following strategy is adopted:

1. Define some possible starting values of the both parameters. For the window size, a value of 15, which is within the range of the popular window sizes [7, 17] [67], is selected as the starting point. The recent predictors like PHD, PSIPRED, etc., have already reached an accuracy level higher than 70%; Considering that the contribution of a kernel should also be reliable enough to the predictor, a purity level of $\rho = 0.6$ was set as the starting point.
2. Define a step size for each parameter. For window size N , a minimum step size of 2 is used; This is equivalent to expand 1 position further towards the both sides of the neighborhood. For the purity level, the step size is set to be $\Delta\rho = 0.05$ (except the change of the purity level from 0.6 to 0.7, as the accuracies obtained from the initial purity level are relatively low.).
3. Run simulations on the initial parameter set,.

5.3. Q2T Prediction - the raw prediction

4. Set the window size intact and increase the purity level step by step until it reaches 1.0. Conduct experiments on each step.
5. Increase window size N by 2. Repeat Step 4.
6. Repeat Step 5 until the significant performance degrading is observed.
7. Reset window size to the starting point. Decrease the window size step by -step and repeat experiments with respect to ρ at each step. Stop further trials if the significant decay in performance is observed.
8. Record the accuracies returned from all trials of simulation.

By conducting experiments in the above strategy, It could be seen how the generalization performance of the RBFNN prediction is impacted by the purity value ρ . Meanwhile, It would also can be told which window size is optimal to the RBFNN prediction.

Results

Table 5.2 shows the experimental results for window size $N = 11$ with the purity level ρ changing from 0.6 to 1.0. The "Q3" column of the table shows the overall accuracies obtained with different purity levels. From the column, It can be seen that the change of accuracy against the purity includes two parts. In the first part, when ρ keeps increasing from 0.6 through 0.85, the accuracy is either increasing or maintained at the top accuracy value. This is where the higher cluster purity plays "active" role: The higher purity level makes each cluster more accurate in response and hence improves the prediction accuracy. Also, higher cluster purity results the better noise rejection. When the purity goes beyond 0.85 and increases further to 1.0, this is the second part of the change. In this change, the further

5.3. Q2T Prediction - the raw prediction

Window Size(N)	Purity Level(ρ)	Accuracies			
		$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
11	0.6	72.9	73.9	55.1	81.6
	0.7	74.7	75.5	57.1	83.4
	0.75	74.8	75.6	57.2	83.5
	0.8	74.8	75.6	57.3	83.5
	0.85	74.8	75.6	57.3	83.4
	0.9	74.7	75.3	57.3	83.6
	0.95	74.6	75.1	57.2	83.5
	1.0	74.3	74.9	56.7	83.1

Table 5.2: Prediction accuracies of the sequence-structure prediction with window size $N = 11$ and various purity level ρ

increase in ρ reduces the generalization capability of the RBF network and this results the drop of the prediction accuracy.

Table 5.3, 5.4 and 5.5 shows the experimental results for window size $N \in \{13, 15, 17\}$ and $\{19, 21, 23\}$ and $\{25, 27\}$, respectively. In these results, the similar scenario of the accuracy change against the purity could be observed. On the other hand, the ρ value for peak accuracies are not exactly the same for different window sizes. This variation is reasonable: the window size determines the span of window; for various window size, the information carried by the data may be different; consequently, the distance computed for any two data points would also change, and this in turn, influences the value of optimal purity level ρ for clustering. Despite of the difference in optimal purity value ρ what is important is that there is always a ρ balances the accurate responses of the kernels and the good generalization performance of the RBFNN.

Table 5G summarizes the best accuracies obtained for each window size. From this table, the relationship of the window size N and the RBFNN prediction ac-

5.3. Q2T Prediction - the raw prediction

Window Size(N)	Purity Level(ρ)	Accuracies			
		$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
13	0.6	73.4	74.3	56.0	81.9
	0.7	74.9	76.2	57.6	83.2
	0.75	75.1	75.9	57.7	83.7
	0.8	75.0	75.6	57.4	83.8
	0.85	75.1	76.0	57.6	83.8
	0.9	75.0	75.7	57.5	83.8
	0.95	74.9	75.9	57.2	83.7
	1.0	74.8	75.8	57.2	83.5
15	0.6	73.4	74.4	56.2	81.8
	0.7	75.2	76.2	57.7	83.6
	0.75	75.2	76.2	57.9	83.7
	0.8	75.2	76.2	57.8	83.8
	0.85	75.2	76.1	57.7	83.8
	0.9	75.3	76.2	58.2	83.7
	0.95	75.2	75.8	58.2	84.0
	1.0	75.0	75.5	57.7	83.8
17	0.6	73.7	74.8	56.8	82.0
	0.7	75.2	76.3	58.0	83.5
	0.75	75.3	76.4	58.0	83.6
	0.8	75.3	76.3	58.1	83.8
	0.85	75.3	76.5	57.9	83.6
	0.9	75.2	76.2	57.9	83.7
	0.95	75.2	75.9	57.8	83.9
	1.0	74.9	75.3	57.9	83.8

Table 5.3: Prediction accuracies of the sequence-structure prediction with various window size N and purity level ρ : $N \in (13, 15, 17)$

5.3. Q2T Prediction - the raw medication

Window Size(N)	Purity Level(ρ)	Accuracies			
		$Q_3(\%)$	$Q_c(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
19	0.6	73.8	74.8	57.0	82.0
	0.7	75.2	76.1	57.9	83.8
	0.75	75.4	76.5	58.2	83.7
	0.8	75.3	76.4	58.6	83.5
	0.85	75.4	76.5	58.1	83.6
	0.9	75.3	76.5	57.9	83.6
	0.95	75.3	75.8	58.4	83.9
	1.0	75.0	76.1	57.7	83.4
21	0.6	73.7	74.7	56.7	81.8
	0.7	75.4	76.4	58.6	83.5
	0.75	75.3	76.3	58.1	83.7
	0.8	75.5	76.7	58.5	83.5
	0.85	75.6	76.6	58.3	83.9
	0.9	75.3	76.3	57.4	84.0
	0.95	75.1	75.8	58.5	83.5
	1.0	74.9	75.8	58.2	83.1
23	0.6	73.8	74.6	56.7	82.3
	0.7	75.3	76.1	58.1	83.9
	0.75	75.3	76.6	57.8	83.6
	0.8	75.4	76.8	58.4	83.4
	0.85	75.5	76.6	58.3	83.8
	0.9	75.4	76.5	57.9	83.7
	0.95	75.2	75.9	58.3	83.7
	1.0	74.9	76.0	57.9	83.2

Table 5.4: Prediction accuracies of the sequence-structure prediction with various window size N and purity level $\rho : N \in \{19, 21, 23\}$

5.3. Q2T Prediction - the raw medication

Window Size(N)	Purity Level(ρ)	Accuracies			
		$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
25	0.6	73.6	74.6	56.3	82.1
	0.7	75.2	76.0	58.2	83.6
	0.75	75.3	76.6	58.0	83.5
	0.8	75.2	76.2	57.9	83.5
	0.85	75.4	76.5	58.6	83.4
	0.9	75.4	76.3	58.3	83.6
	0.95	75.1	75.8	58.1	83.6
	1.0	74.7	75.9	57.3	83.0
27	0.6	73.4	73.8	56.3	82.1
	0.7	75.2	76.4	58.1	83.3
	0.75	75.2	76.2	58.0	83.6
	0.8	75.2	76.3	58.0	83.5
	0.85	75.2	76.3	57.8	83.5
	0.9	75.1	76.2	57.7	83.6
	0.95	75.1	76.0	57.9	83.5
	1.0	74.8	75.8	57.4	83.2

Table 5.5: Prediction accuracies of the sequence-structure prediction with various window size N and purity level ρ $N \in \{25, 27\}$

Window Size (N)	11	13	15	17	19	21	23	25	27
Q_3	74.8	75.1	75.3	75.3	75.4	75.6	75.4	75.4	75.2

Table 5.6: Peak accuracies reached at each window size N : a summary

5.4. T2T Prediction - the final prediction

curacy has been quite obvious. The best Q_3 accuracy is 75.6% when the window size is $N = 21$. Either a window size larger or less than 21 leads to the degraded prediction performance. In other words, the window size 21 has most relevantly captured the useful information hidden in the neighborhood and serves best for the learning of the RBFNN.

5.4 T2T Prediction - the final prediction

5.4.1 Feasibility of the Back Propagation

With the optimal prediction performance reached at the first stage, the second stage shall also be optimized such that the overall prediction capability of the HNNP can be maximized. As introduced in the previous chapter, this stage was implemented using an MLP network. The network was trained with the fundamental backpropagation technique-gradient descent. This training algorithm is relatively simple and straightforward and is the most traditional way for the training of an MLP network. On the other hand, there are other training algorithms which are more superior and efficient like the Conjugate Gradient (CG) method [68] [69], the Levenberge-Marquadt(LM) method [70] [71], etc.. It might seem unreasonable to select a “naive” rather than a “smart” algorithm for our training. The rational should have been explained at the time when the training algorithms were described, in Chapter 4. However, as some parts of rational are related to the experimental trials, the discussions are actually left here.

The LM method is a non-linear optimization algorithm which combines the advantages of the steepest-descent and the Gauss-Newton method: fast convergence from steepest-descent method at far minimum point and fast convergence from Gauss-Newton method at near minimum point. The operation of the algorithm is

5.4. T2T Prediction - the final prediction

based on

$$H = J^T J, \quad (5.17)$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases. In other words, this matrix has a size: number-of-weights-bias $\times$ number-of-patterns. In our problem, the number of training patterns is around 2×10^5 . If we need 1000 weights (which is quite moderate to our problem) to train these patterns, the memory required to store the Jacobian matrix would be $2 \times 10^5 \times 10^3 \times 8$ bytes (double precision) ≈ 1.6 GB. Such a demand for memory space is beyond the computation resources available, not to mention the even space-demanding inner product between the Jacobian matrix and its transpose. Hence, it is not feasible to use the LM method despite of its faster convergence.

The backpropagation algorithm searches the minimum of the error surface in the negative direction of the gradient. The CG method, however, applies a line search technique for which the search is guided by the conjugate directions. The algorithm is especially suitable to the error minimization problem with the quadratic error surface. In comparison to the backpropagation method, the CG method is generally faster in convergence and may help to escape from some local minima like the saddle points. On the other hand, the CG method has to adopt the batch learning technique such that the direction found is suitable for all training data: the newly found direction have to be tested on all training data against the sum of the squared error; the method tries several times of search until an appropriate direction has been found; otherwise, a randomization of the search would be made and the newly found direction is again evaluated for correctness, and the similar scenario repeats until the minimum of the error surface is reached. Each trial of search requires several GOs (to check whether the sum of the squared error is re-

5.4. T2T Prediction - the final prediction

duced) through all training samples. Hence, the time consumed in the direction search may become significant when the dataset used is extremely large in scale and high in dimensionality. As a result, the overall convergence speed might be lower despite that the CG method requires less steps to reach the minimum error. And our experiments using CG method seems to match the above rational. From the experiment, it is observed that the method spent most of its time searching for a new direction that minimized the error. As an example for the comparison of the convergence, the backpropagation algorithm reached its minimum with a sum of the squared error of 0.23 using around one night of time; While for CG, it could only reach a sum of the squared error 0.3 within the same amount of time. Hence, the backpropagation algorithm is more suitable for our problem.

5.4.2 Strategy

The second stage predictor is a simple yet large MLP network. The network is simple as it is implemented with a standard three-layer feedforward network and is trained using the fundamental backpropagation technique; The network is also large in that several thousands of neurons are used in the hidden layer and a large amount of data is used for the training.

To setup the network, the online learning method was adopted. That is, the weights of the MLPNN were updated using backpropagation technique at each time when an input pattern is present. When all of the input patterns have been used, this was one *epoch* of the training. At each epoch, the training efficiency and the generalization performance of the network are examined. The training continued if both the training efficiency and generalization performance are increasing. The training stops at the optimal state where there was no further improvement of generalization with the increase of training efficiency. To obtain the optimal performance, the use of different windows size N shall be investigated for the secondary

5.4. T2T Prediction - the final prediction

structure prediction. The strategy adopted for the experiment is as follow:

1. Randomly start with a window size of 17. Set up the MLP network with the output of the training data from the first stage.
2. Feed the testing dataset into the network to do the prediction. Record the prediction accuracy.
3. Increase the window size by 2. Train the network and do the prediction. Compute the prediction accuracy.
4. Repeat Step 3 until the significant decrease in prediction accuracy is found.
5. Set the window size to 15. Train the network and do the prediction. Compute the prediction accuracy.
6. Decrease the window size by 2. Train the network and do the prediction. Compute the prediction accuracy.
7. Repeat Step 6 until the significant decrease in the prediction accuracy is found.
8. Finalize the ultimate secondary structure assignment on testing dataset by selecting the best prediction.

5.4.3 Results

Table 5.7 shows the experimental results for window size N varying from 11 to 25. From the table, the best accuracy Q_3 of the T2T prediction is 76.8% when the window size N is 17. This is also the best final prediction accuracy that the propped HNNP could reach. For the best prediction accuracy obtained, HNNP reached the corresponding class accuracy $Q_\alpha=78.9\%$ for α -helix, $Q_\beta=65.6\%$ for β -strand and $Q_c=81.2\%$ for coil.

5.4. T2T Prediction - the final prediction

Window Size(N)	Accuracies			
	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
11	76.4	79.6	61.9	81.6
13	76.7	79.1	61.9	82.7
15	76.7	79.4	61.7	82.5
17	76.8	78.9	65.6	81.2
19	76.6	79.8	61.8	81.9
21	76.5	80.0	61.0	82.1
23	76.5	79.8	60.9	82.1
25	76.4	80.4	61.2	81.3

Table 5.7: Prediction accuracies of the structure-structure prediction for window size $N = \{11, 13, 15, 17, 19, 21, 23, 25\}$

Table 5.8 shows the peak accuracies reached in the Q2T and the T2T predictions. From this table, it can be seen clearly the contributions of the sequence-

Prediction Stage	Accuracies			
	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
Q2T	75.6	76.6	58.3	83.9
T2T	76.8	78.9	65.6	81.2

Table 5.8: Contributions of the sequence-structure and the structure-structure prediction to the final prediction accuracy

structure prediction and the structure-structure prediction to the final prediction accuracy. The Q2T prediction, i.e., the RBFNN, is the crucial part of the PSSP problem and gives rise a Q_3 accuracy of 75.6%. The secondary structure prediction almost entirely relied on this stage of prediction. Based on this prediction, the T2T prediction, i.e., the MLPNN, improves the raw prediction by 1.2%. Although this contribution is minor in compare to that of the RBFNN, it is not negligible.

5.4. T2T Prediction - the final medication

The improvement is a “reward” received from studying the correlations hidden in the protein secondary structures and the improvement is worth of treasuring.

The table also shows the change of prediction accuracies for each secondary structure class. The use of a structure-structure prediction improved the prediction of α -helix by 2.3% and β -strand by 7.3%. Obviously, the improvements were gained from correcting wrong predictions to the coil state. On the other hand, the correction has also incorrectly predicted some coil states to helix or strand states and this resulted the drop of Q_c . Despite of that, the correction by T2T is still successful as the overall accuracy Q_3 has been improved.

By comparing the prediction accuracies for different classes, it can be seen that the prediction accuracies for helix, strand and coil are quite different. This difference, however, is a reflection of the class distribution of the training data. And this is relevant to the statistical analysis on the class distributions of the training data. The analysis shows that the population ratio for helix, strand and coil class in the training is roughly 3:2:4. The most populated coil class achieves the highest prediction accuracy and the least populated strand class achieves the lowest. The reason of such discrimination is due to the discrimination in training. The coil class tends to “teach” the learning machine more frequently than the strand and helix classes. Hence, the machine becomes more sensitive to the coil class and less sensitive to the other two classes. As a result, such discrimination is reflected in the prediction accuracies of the three classes.

Table 5.9 gives a more thorough evaluation of the final prediction with additional information like Segment Overlap Measure (SOV) and the Matthew’s correlations (ρ_α , ρ_β , ρ_c). The result is also compared with one of the existing leading method PSIPRED [20].

The above table shows that the proposed HNNP outperforms the PSIPRED

5.4. T2T Prediction - the final prediction

Method	Per-Residue				Matthew			Segment
	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$	ρ_α	ρ_β	ρ_c	$SOV(\%)$
HNNP	76.8	78.9	65.6	81.2	0.72	0.61	0.59	73.6
PSIPRED	76.5	-	-	-	-	-	-	73.5

Table 5.9: More thorough performance evaluation of the proposed HNNP and its comparison with the PSIPRED method

method in both the per-residue accuracy and the segment overlap measure, i.e., 76.8% Vs 76.5% for Q_3 and 73.6% Vs 73.5% for SOV . Although the improvement is not so significant, it at least shows that the proposed HNNP system is feasible for the PSSP problem. For the proposed architecture, it is the first time that the RBFNN was applied for PSSP problem. To train the RBFNN, a distance measure scheme WINDist which tried to address the importance of neighboring residues was proposed. The overlapped clustering technique was applied to enhance the noise rejection capability of the network. All the efforts made to pursue the accurate secondary structure prediction have been rewarding as the hybrid networking of the RBFNN with the MLPNN has reached the comparable performance with the existing leading method. The proposed HNNP architecture is viable for the PSSP problem.

Chapter 6

The PSSP Problem using Support Vector Machines

6.1 Introduction

The Support Vector Machine (SVM) is a relative new category of feedforward networks pioneered by Vapnik [72] [73][74] [47]. It was derived from the statistical learning theory, aiming at solving the problem of interest ONLY without solving a more difficult problem as an intermediate step. The support vector machine is developed based on the structural risk minimization principle. The principle incorporates the capacity control to prevent the over-fitting and thus is a partial solution to the bias-variance trade-off dilemma [75]. Ever since its invention, support vector machines have been widely applied in pattern classification problems [76] [77] [16] [78][79] and nonlinear regressions [80] [81] [82].

As this project works on the data classification problem, all the discussions followed are specific to the SVM as a classifier.

Basically, a support vector machine is used to distinguish patterns of one class from the other. In essence, it solves a binary classification problem. For the ease of explanation on how the SVM works, let us consider a set of patterns that belong to two classes and are separable. The main idea of the support vector machine

6.2. SVM basics

is to construct a hyperplane which maximizes *the margin of separation* between two classes. The hyperplane performs a function as the decision maker: patterns distributed on the same side of the plane are assigned to the same class and the rest to the other class. According to the structural risk minimization principle, the generalization performance of a SVM is bounded by 1). the sum of the training error and 2). the Vapnik-Chervonenkis (VC) dimension [83]. For the classes of patterns that are successfully separated by the hyperplane, the first term is always at its minimal value of 0; The task of the SVM is thus only to minimize the VC dimension. Accordingly, the support vector machine can reach good generalization for classification problems even though it does not incorporate problem-domain knowledge. This is an unique property of the support vector machine.

In the subsequent section, a more detailed description of the theoretical basics of support vector machines is presented.

6.2 SVM basics

6.2.1 SVM for Separable Patterns

Consider a two-class (e.g., Class 1 and 2) problem, suppose we have a set of P input-output pairs $\{(x_i, t_i)\}_{i=1}^P$. x_i represents i -th input pattern and t_i is the desired output or class. A hyperplane that possesses decision capability should satisfy the following equation

$$w^T x + b = 0, \quad (6.1)$$

where w is the weight vector and b is the bias. T represents the *transpose* operation. With the definition of the hyperplane, the classification of the patterns could be expressed mathematically as

$$\begin{aligned} w^T x + b &\geq 0, & \text{for Class 1;} \\ w^T x + b &< 0, & \text{for Class 2.} \end{aligned} \quad (6.2)$$

6.2. SVM basics

To distinguish the two classes easier, Equation 6.2 is modified to the form as

$$\begin{aligned} w^T x + b &\geq 1, & \text{for Class 1;} \\ w^T x + b &\leq -1, & \text{for Class 2.} \end{aligned} \quad (6.3)$$

Note that the change in the modified expression is actually a result of rescaling of adjustable variables w and b and thus would not affect the decision of classification.

In the pattern classification context, a hyperplane is said to be optimal when the margin of separation between Class 1 and 2 is maximized. Figure 6.1 illustrates how an optimal hyperplane is used to separate a set of two-dimensional patterns.

From the diagram, we see two classes of patterns represented by solid squares

Figure 6.1: A hyperplane for linearly separable patterns: The margin of separation between Class 1 and 2 is maximized

and small circles, respectively. The shaded rectangle is the region of separation for the two classes. The dashed line crossing the symmetry of the rectangle represents the optimal hyperplane that maximizes the margin of separation. On the upper and lower boundaries of the rectangle, we also see 5 patterns (i.e., A, B, C, D, E) lying exactly on the line. These patterns are the vectors that bound the region of

6.2. SVM basics

separation and are called the support vectors. Feed them into Equation 6.3 would make the expressions get the exact equality. In the figure, we also see a lower case character d which represents the distance between the optimal hyperplane and the separation boundary of Class 1. Mathematically, the algebraic distance could be computed by

$$d = \frac{g(x_s)}{\|w\|}, \quad (6.4)$$

where

$$g(x_s) = w^T x_s + b, \quad (6.5)$$

and x_s is a support vector with

$$g(x_s) = \begin{cases} \frac{1}{\|w\|} & , \text{ if } t_s \text{ is in Class 1;} \\ -\frac{1}{\|w\|} & , \text{ if } t_s \text{ is in Class 2.} \end{cases} \quad (6.6)$$

The margin of separation between the two classes thus is

$$\begin{aligned} \rho &= 2d \\ &= \frac{2}{\|w\|}. \end{aligned} \quad (6.7)$$

In Equation 6.7, the denominator is actually the Euclidean norm of the weight vector w . In order to find an optimal hyperplane, the Euclidean norm has to be minimized such that a maximum margin of separation could be reached.

The Optimal Hyperplane

The construction of the SVM classifier requires to find an optimal free parameter set, i.e., the weight vector w and the bias term b , such that the error of classification based on rules defined in Equation 6.3 is minimized. Let us assign numerical values to the two classes:

$$t_i = \begin{cases} +1 & , \text{ if } t_i \text{ belongs to Class 1;} \\ -1 & , \text{ if } t_i \text{ belongs to Class 2.} \end{cases} \quad (6.8)$$

6.2. SVM basics

Combine this with Equation 6.3, it follows that the optimal hyperplane may be constructed subject to the following constraint:

$$t_i(w^T x_i + b) \geq 1 \quad , \quad \text{for } i = 1, 2, \dots, P. \quad (6.9)$$

The optimization constraint formulated in the above equation could thus be stated as follow:

Given the set of input-output pairs $\{(x_i, t_i)\}_{i=1}^P$, find the optimal weight vector w and bias b such that they satisfy the constraints

$$t_i(w^T x_i + b) \geq 1 \quad , \quad \text{for } i = 1, 2, \dots, P,$$

and the weight vector w minimizes the cost function:

$$\Phi(w) = \frac{1}{2} w^T w. \quad (6.10)$$

This constraint optimization problem is called the *primal problem*. The constraint has the following features:

1. The cost function $\Phi(w)$ is a convex function of w .
2. The constraints are linear in w .

Hence, the method of Lagrange multipliers [84] may be applied. And by the duality theorem [84], we obtain the *dual problem* which is a equivalent statement to the primal one:

Given the set of input-output pairs $\{(x_i, t_i)\}_{i=1}^P$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^P$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^P \alpha_i - \frac{1}{2} \sum_{i=1}^P \sum_{j=1}^P \alpha_i \alpha_j t_i t_j x_i^T x_j,$$

subject to the constraints

$$\begin{aligned} (1) & \sum_{i=1}^P \alpha_i t_i = 0; \\ (2) & \alpha_i \geq 0, \quad \text{for } i = 1, 2, \dots, P. \end{aligned}$$

6.2. SVM basics

The detailed derivation could be referred in [85].

By comparing the two statements, we see that the dual problem is related to the training data only. The minimization of the objective function $Q(\alpha)$ depends only on the input patterns.

With the Lagrange multipliers determined, the optimal weight vector w_o can be computed using the following equation:

$$w_o = \sum_{i=1}^P \alpha_{o,i} t_i x_i. \quad (6.11)$$

And the optimal bias b_o could be computed using Equation 6.3 pertaining to Class 1 support vector x_s :

$$b_o = 1 - w_o^T x_s, \quad \text{for } t_s \text{ in Class 1.} \quad (6.12)$$

6.2.2 SVM for Nonseparable Patterns

In the previous section, we introduced how to find the optimal hyperplane for the set of patterns $\{(x_i, t_i)\}_{i=1}^P$ that are linearly separable. In practice, however, it is quite common that the patterns may not be separable. The construction of a hyperplane that successfully classifies the nonseparable pattern set without error thus is no longer viable. To address this problem, a *soft margin* of separation is defined which allows some small amount of patterns violate the condition as in Equation 6.9. The violation can be in either of the following ways:

1. The pattern (x_i, t_i) falls inside the region of separation but on the right side of the decision surface. Pattern *A* in Figure 6.2 represents this violation.
2. The pattern (x_i, t_i) falls on the wrong side of the decision surface. Pattern *B* in Figure 6.2 is in such a violation.

6.2. SVM basics

Figure 6.2: A hyperplane with the soft margin of separation: violations allowed either on the right or the wrong side of the decision surface for pattern A and B, respectively.

To treat the nonseparable patterns in a formal way, a slack variable set $\{\xi_i\}_{i=1}^P$ is introduced. The slack variable holds a positive value and represents the deviation of a pattern from the ideal condition of pattern separability. Put the slack variable into decision Equation 6.9, we have:

$$t_i(w^T x_i + b) \geq 1 - \xi_i, \quad \text{for } i = 1, 2, \dots, P. \quad (6.13)$$

According to the above expression, a ξ_i value large than 1 would make the right hand side to be less than zero; this implies the decision made by the hyperplane is wrong. For $0 \leq \xi_i \leq 1$, the decision is right yet the pattern falls inside the region of separation. An optimal hyperplane again, shall minimize the classification error on the training set. The minimization can be realized by minimizing the function

$$\Phi(\xi) = \sum_{i=1}^P \xi_i \quad (6.14)$$

with respect to the weight w , subject to the constraint in Equation 6.13 and the minimization of $\|w\|$. In all, the function needs to be minimized could be simply

6.2. SVM basics

written as:

$$\Phi(W, \xi) = \frac{1}{2}w^T w + C \sum_{i=1}^P \xi_i, \quad (6.15)$$

where the first term of the right hand side is related to the VC dimension while the second term bounds the testing error. The parameter C here is a regularization factor that controls the trade off between the complexity of the learning machine and the number of nonseparable points. The parameter may be experimentally selected by the user.

Given Equation 6.15, we state the primal problem for the nonseparable pattern set as:

Given the set of input-output pairs $\{(x_i, t_i)\}_{i=1}^P$, find the optimal weight vector w and bias b such that they satisfy the constraint

$$t_i(w^T w + b) \geq 1 - \xi_i \quad , \quad \text{for } i = 1, 2, \dots, P,$$

$$\xi_i \geq 0 \quad , \quad \text{for all } i,$$

and such that the weight vector W and the slack variables ξ_i minimize the cost function

$$\Phi(w, \xi) = \frac{1}{2}w^T w + C \sum_{i=1}^P \xi_i.$$

Again, with the method of Lagrange multipliers and the duality theorem, the dual problem could be derived and stated as:

Given the set of input-output pairs $\{(x_i, t_i)\}_{i=1}^P$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^P$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^P \alpha_i - \frac{1}{2} \sum_{i=1}^P \sum_{j=1}^P \alpha_i \alpha_j t_i t_j x_i^T x_j,$$

subject to the constraints

- (1) $\sum_{i=1}^P \alpha_i t_i = 0;$
- (2) $0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \dots, P$

6.2. SVM basics

6.2.3 Construction of Support Vector Machines

The basic ideas of a support vector machine fulfilling a classification problem could be summarized as follow:

1. Map the input space into a high-dimensional feature space using nonlinear transformations. The feature space is hidden from both the input and output space.
2. Construct an optimal hyperplane such that the patterns can be separated in the high-dimensional feature space.

To construct a SVM, we first should have a set of functions which perform the nonlinear transformations for any input patterns x . Let the set of functions to be $\{\psi_j(x_i)\}_{j=1}^{m_1}$, m_1 is the number of the features in the hidden space. A feature vector $\Psi^{(m_1)}(x)$ is obtained by passing the input pattern x to the transformation functions one by one. In vector form, $\Psi^{(m_1)}(x)$ could be written as

$$\Psi^{(m_1)}(x) = [\psi_1(x), \psi_2(x), \dots, \psi_{m_1}(x)]^T. \quad (6.16)$$

With the feature vector shown above, we could start constructing the optimal hyperplane. From Equation 6.2, the optimal hyperplane is defined as:

$$w^T \Psi^{(m_1)}(x) + b = 0, \quad (6.17)$$

where ω is a m_1 -dimensional vector connecting the feature space to the output space. More generally, we define a feature $\psi_0(X) = 1$ and the bias term b can be represented by weight w_0 . Thus, Equation 6.17 becomes

$$w^T \Psi(x) = 0, \quad (6.18)$$

with

$$\Psi(x) = [\psi_0(x), \psi_1(x), \psi_2(x), \dots, \psi_{m_1}(x)]. \quad (6.19)$$

6.2. SVM basics

From Equation 6.11, we have the similar equation for the feature space:

$$w_o = \sum_{i=1}^P \alpha_{o,i} t_i \Psi(x_i). \quad (6.20)$$

Take the transpose of both sides and multiply them by $\Psi(x)$, we have

$$\sum_{i=1}^P \alpha_i t_i \Psi^T(x_i) \Psi(x) = 0. \quad (6.21)$$

The term $\Psi^T(x_i) \Psi(x)$ represents the inner product of two vectors induced in the feature space by the input vector x and input pattern x_i pertaining to the i -th example. We call it the *inner-product kernel*. The common kernel types which are easy to compute, well-defined and efficient in mapping includes:

- Gaussian-Radial Basis Function
- two-layer Multi-Layer Perceptron
- polynomial classifier

With the inner-product kernel defined, the optimal hyperplane could thus be constructed in the feature space without having to consider the feature space itself in explicit forms. Figure 6.3 shows the architecture of a support vector machine. The input layer contains m_0 input neurons which correspond to the m_0 features of the input vector. The hidden layer maps the input pattern x into the high-dimensional space by the inner product kernels $\{K(x) \in \mathbf{R}^{m_1} : K_i(x) = \Psi^T(x_i) \Psi(x)\}_{i=1}^{m_1}$. With the bias b and the weights w connecting the hidden layer and the output layer, the output neuron makes the decision on the image of the input pattern using Equation 6.3. A positive output with a value larger than 1 is assigned to Class 1 while a negative output with a value less than -1 is assigned to Class 2.

6.2. SVM basics

Figure 6.3: Architecture of the support vector machine as a binary classifier

6.2.4 Multi-class Problem with Binary Classifiers

In general pattern classification problems, it always happens that there are more than 2 classes of patterns, i.e., the multi-class classification. The support vector machine, on the other hand, is basically a binary classifier. To deal with the multi-class classification problem, a set of binary SVM classifiers may be constructed and the final decision is made by combining the outputs of these binary SVM classifiers. Depending on the combination scheme adopted, various multi-class classifiers may be constructed using single binary classifiers. In the protein secondary structure problem, there are three classes of patterns, namely: H for α -helix, E for β -strand and C for coils. The typical tertiary classifiers may be constructed using:

1. One-Versus-One Scheme

Under this scheme, a binary classifier is constructed for each possible class-pair combination. This results a total of $C_2^{n_c}$ classifiers to be constructed, where n_c is the number of classes in the output space. For this project's case $n_c = 3$, $C_2^3 = 3$ classifiers need to be constructed: the H/E classifier, the

6.3. Work with the LIBSVM

H/C classifier and the E/C classifier. In the prediction, each binary classifier contributes a *vote* to the secondary structure assignment. The final decision of prediction is the secondary structure class that wins the vote.

2. One- Versus-All Scheme

In comparison to the One-Versus-One scheme, this scheme is low in complexity. It only requires to construct one binary classifier for each class which distinguishes the class from the others. As a result, the number of binary classifiers needed is equal to the number of the classes in the output space. Therefore, only 3 classifiers need to be constructed for the PSSP problem, if necessary: the $H/\tilde{H}$ classifier, the E/E classifier and the $C/\tilde{C}$ classifier. The sign $\sim$ represents the logical “NOT”. To decide the class assignment of the pattern, the decimal outputs from three classifiers are compared. The pattern is assigned to Class i if the i -th classifier has the largest positive output. In essence, this is the Winner-Take-All (WTA) technique which has already been used in the proposed HNNP.

Figure 6.4 illustrates the scenario of how the above two tertiary classifiers are constructed for fulfilling the classification task.

6.3 Work with the LIBSVM

The LIBSVM is a library of support vector machines that is developed by Chang and Lin [86]. It includes an assembly of support vector machines for classification: the C-Support Vector Classifier(C-SVC) proposed by Cortes and Vapnik [73], the ν -Support Vector Classifier(ν -SVC) proposed by Scholkopf et al [87]; for distribution estimation: the One-Class SVM proposed by [88]; and for regression: the ϵ -Support Vector Regression proposed by Vapnik [47], the ν -Support Vector Regression proposed by Scholkopf et al.. The library is available online at

6.3. Work with the LIBSVM

Figure 6.4: Tertiary classifiers constructed for the protein secondary structure prediction problem based on a) One-Versus-One Scheme; b) One-Versus-All Scheme.

<http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

As our problem of concern is a classification problem, we would briefly introduce the implementation of the LIBSVM for classification.

Given a set of input-output pairs $\{(x_i, t_i)\}_{i=1}^P$, the following primal problem has to be optimized for the C-SVC:

$$\begin{aligned}
 \min_{w,b,\xi} \quad & \frac{1}{2}w^T w + C \sum_{i=1}^P \xi_i; \\
 & t_i(w^T \Psi(x_i) + b) \geq 1 - \xi_i; \\
 & \xi_i \geq 0, \quad \text{for } i = 1, 2, \dots, P.
 \end{aligned} \tag{6.22}$$

And the dual problem is

$$\begin{aligned}
 \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q \alpha - e^T \alpha; \\
 & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \dots, P; \\
 & t^T \alpha = 0.
 \end{aligned} \tag{6.23}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an $P \times P$ positive semidefinite matrix and $\Psi(x_i)$ is the image of the input pattern x_i in the high dimensional feature space. The difficulty of solving Equation 6.23 is the density of

6.3. Work with the LIBSVM

Q as the entries of this matrix are usually nonzero. To conquer the problem, the LIBSVM applied the decomposition method [89] [90] [91] [92]. The algorithm is as follow

1. Define the working set size $q \leq P$. Find the initial solution of α_1 . Let $k=1$.
2. Stop if α^k is an optimal solution to the dual problem. Otherwise, find a working set B with size q , $B \subset \{1, 2, \dots, P\}$. Let $N = \{1, 2, \dots, P\} \setminus B$ and α_B^k and α_N^k to be the sub-vectors of α^k corresponding to B and N , respectively.
3. Solve the following sub-problem with the variable α_B :

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (p_B + Q_{BN} \alpha_N^k)^T \alpha_B; \\ & 0 \leq (\alpha_B)_t \leq C, \quad \text{for } t = 1, 2, \dots, q; \\ & D_B^T \alpha_B = \Delta - t_N^T \alpha_N^k, \end{aligned}$$

where

$$\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$$

is a permutation of the matrix Q .

4. Set α_B^{k+1} to be the optimal solution of Step 3 and $\alpha_N^{k+1} = \alpha_N^k$. Increase k by 1 and go to Step 2.

Basically, the decomposition method is an iterative method. At each iteration, the training set is separated into two disjoint sets B and N . By fixing the variable α_N , the problem is decomposed in part to solve the problem for α_B . The working set B is updated at each iteration and the optimization process repeats until the convergence is reached.

The same decomposition method was also applied by LIBSVM to solve the quadratic optimization problem of ν -SVC.

6.3. Work with the LIBSVM

Now it is time to study the capability of the LIBSVM, or more specially, the C-SVC, for solving the protein secondary structure prediction problem. The datasets used for the training and the testing were the same as those used by the proposed HNNP. To see the Performance of the C-SVC, a C-SVC model shall be set up for which all the free parameters are optimal. These parameters include

1. Types of the kernel $K(x_i, x_j)$ used to map the input pattern into the feature space;
2. Size of the sliding window N which controls the span of the neighborhood;
3. The parameter γ which controls the span of the kernel: $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$;
4. The parameter C which controls the complexity of the classifier.

To reduce the search space of optimal parameters, only the RBF kernels is considered as transformation kernels. It is then left with three parameters. Again, it shall be decided how the parameters are related to each other and the relevant experimental strategy should be chosen accordingly.

Similar to the HNNP prediction, two stages of the secondary structure prediction are performed, i.e., the sequence-structure(Q2T) prediction and the structure-structure(T2T) prediction.

6.3.1 Q2T prediction

Strategy

As mentioned, the window size N determines the span of the sliding window. In other words, it implicitly determines how many informative and how many “noisy” neighbors to be included in the window. Hence, this parameter impacts the prediction performance more on the data itself rather than on the classifier.

6.3. Work with the LIBSVM

Consequently, this parameter could be considered independently. γ is the parameter related to the span of a RBF kernel: The smaller the value is, the wider the kernel spans. As a result, it controls the tradeoff between the training accuracy and the generalization performance of the machine. The last parameter, C , controls the tradeoff between the complexity of the machine and the degree of separability; it also impacts the accuracy of the machine. As both γ and C contribute to the prediction capability of the C-SVC, they are joined as a pair (C, γ) . Various combinations of them have to be tried, ideally, to find the optimal combination. On the other hand, the general range of variations for γ and C are $[0 \ 1]$ and $[1 \ 1000]$, respectively. Searching the optimal (C, γ) pair from such wide ranges becomes very difficult especially when the dataset used in this project is extremely large in scale and requires a lot of computations. It is preferred that a (C, γ) pair shall be selected such that it provides the basic concept of the scopes of the two parameters. Luckily, such information can be referenced from Lin and Lin's work [93]. In their work, they found an optimal $(C, \gamma) = (2, 0.125)$ for the PSSP problem with a smaller (about 10 times smaller) dataset. Despite of the difference in the size of the dataset used, the parameter pair should still serve well as a guide for scopes. To search for an optimal parameter combination of the window size N , γ and C , a random (C, γ) pair within the scope can be selected and the changes (increase/decrease) of parameters shall be performed again one of them only at each time. If the change results the better prediction accuracy, the change (increase/decrease) continues; Otherwise, the change is reversed (decrease/increase). With the optimal value found for this parameter, same procedure is repeated for the other parameters. And to see the impact of further variations of the parameter pair, a starting (C, γ) different from that of the previously selected is used and the similar scenario repeats.

6.3. Work with the LIBSVM

Results

Table 6.1, 6.2, 6.3 and 6.4 shows the experimental results for various (C, γ) pair with the window size $N \in \{11, 13, 15, 17\}$, respectively. Note that the results are sorted according to the C value for the sake of ordered presentation. However, by doing so does not mean that the performance of the C-SVC is dependant on γ only. The impact of the parameter C maybe observed from the rows sharing the same γ value.

From the tables, it can be seen that optimal (C, γ) values for window size $N = \{11, 13, 15, 17\}$ are $(1.5, 0.03)$, $(2, 0.045)$, $(2, 0.04)$ and $(2, 0.03)$, respectively. A closer look at the tables also reveals how γ and C influence the C-SVC's prediction accuracy.

To see the impacts of γ , for example, entries of the third column (under the table attribute ' γ ') in Table 6.3 reveal the relationship, when C is fixed to 2. From the table, it can be seen that the accuracy Q_3 first increases with γ , reaches maximum, and then drops with increase of γ . What happens during the change of γ is that: when γ is in small value, i.e., 0.06, the RBF kernel has a wide span and the training accuracy has not yet reached higher enough; Consequently, the testing accuracy could not reach the highest value; On the other hand, when γ reaches the large value, i.e., 0.15, the training accuracy has already been higher than sufficient; This leads to the overfitting of the machine and the generalization capability of the machine is reduced; As a result, only a 69.0% of Q_3 accuracy is reached. The optimal γ is found at the median with a γ value of 0.04. And this corresponds to the best $Q_3 = 74.2\%$ for $N = 15$. Thus, an optimal γ value should either make the training sufficient and meanwhile, would not reduce the generalization capability of the machine.

To see the impacts of C , prediction accuracies shall be compared against fixed

6.3. Work with the LIBSVM

$\gamma = 0.04$. From Table 6.1 and 6.2, we get two (C, Q_3) pair sets $\{(1, 73.8), (2, 73.7), (4, 73.3)\}$ and $\{(1.5, 74.1), (2, 74.1), (4, 73.2)\}$. From the first set, C changes in $1 \rightarrow 2 \rightarrow 4$ which is single increasing. The corresponding Q_3 decreases with the increase of C in $73.8 \rightarrow 73.7 \rightarrow 73.3$. From this observation, it implies that the higher machine complexity actually has reduced the prediction accuracy of the C-SVC. The same scenario could also be observed in the second (C, Q_3) set. The reason causing such changes might be that the overtraining of the machine has occurred when the machine complexity is too high.

Summarize the 4 Tables, the best “ Q_3 ”s the C-SVC could reach for $N = \{11, 13, 15, 17\}$ are 73.9%, 74.2%, 74.2%, 74.1%, respectively. A window size of 13 or 15 seems to be the optimal window size that could most efficiently capture the information hidden in the neighboring residues. The best accuracy reached is 74.2%.

Window Size(N)	C	γ	Accuracies			
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
11	1	0.02	73.8	71.7	54.0	85.5
	1	0.04	73.8	72.4	53.9	85.1
	1.5	0.03	73.9	72.6	54.2	84.9
	2	0.04	73.7	73.1	54.4	84.0
	2	0.045	73.7	73.3	54.5	83.8
	2.5	0.04	73.6	73.3	54.8	83.4
	2.5	0.045	73.7	73.3	55.2	83.4
	4	0.04	73.3	73.4	55.9	82.0

Table 6.1: Prediction accuracies of the C-SVC with different (C, γ) values: $N = 11$

6.3. Work with the LIBSVM

Window Size(N)	C	γ	Accuracies			
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
13	1	0.02	73.9	72.3	54.8	84.9
	1.5	0.008	73.6	71.4	54.3	85.0
	1.5	0.02	73.9	72.6	54.7	84.8
	1.7	0.04	74.1	73.6	54.8	83.4
	2	0.025	74.0	73.0	55.1	84.3
	2	0.04	74.1	73.9	55.0	83.9
	2	0.045	74.2	74.1	55.9	83.5
	4	0.04	73.2	73.9	55.5	81.7

Table 6.2: Prediction accuracies of the C-SVC with different (C, γ) values: $N = 13$

Window Size(N)	C	γ	Accuracies			
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
15	2	0.006	73.4	70.8	54.2	85.2
	2	0.03	74.1	73.6	55.6	84.0
	2	0.04	74.2	73.9	55.7	83.7
	2	0.045	74.0	73.7	55.4	83.7
	2	0.05	74.0	73.7	55.4	83.6
	2	0.15	69.0	63.3	32.7	91.9
	2.5	0.2	74.0	73.0	55.6	84.0
	2.5	0.03	74.1	74.0	55.9	83.5
	4	0.025	74.0	73.8	55.8	83.4

Table 6.3: Prediction accuracies of the C-SVC with different (C, γ) values: $N = 15$

6.3. Work with the LIBSVM

Window Size(N)	C	γ	Accuracies			
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
17	1	0.125	70.0	63.6	36.0	91.3
	2	0.03	74.1	73.5	56.2	83.7
	2.5	0.001	71.3	68.1	52.4	83.5
	2.5	0.02	74.0	68.1	52.4	83.5
	2.5	0.04	74.0	75.0	55.8	83.1

Table 6.4: Prediction accuracies of the C-SVC with different (C, γ) values: $N = 17$

6.3.2 T2T prediction

The T2T prediction uses the output of the Q2T prediction as its input. This practice is similar to the Q2T prediction in the HNNP prediction. Again, to incorporate the long range correlations associated with the protein secondary structures, each residue is expanded in its neighborhood to form the window.

Strategy

As the same C-SVC classifier is used for the secondary structure prediction, the model selection strategy used for the Q2T prediction can be adopted.

Results

Table 6.5 shows the best accuracies reached for window size $N \in \{15, 17, 19\}$ with the corresponding C and γ values. From the table, it is unexpectedly observed that the structure-structure prediction has actually degraded the prediction performance. Take a closer look at the accuracies for each secondary structure class reveals that the prediction for the coils has been much less accurate. In comparison to the early results (Table 6.1, 6.2, 6.3 and 6.4) in the first stage, the $\&C$ accuracy has dropped from a level of 84% to 75%. By sacrificing the accuracy for coils, the

6.3. Work with the LIBSVM

Window Size (N)	C	γ	Accuracy			
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
15	1	2^{-5}	72.6	77.9	60.8	74.3
17	1	2^{-4}	72.6	78.0	60.4	74.5
19	1	2^{-6}	72.8	78.2	60.1	74.9

Table 6.5: The structure-structure prediction accuracies for window size $N \in \{15, 17, 19\}$

predictions for the other two secondary structures have improved. However, the improvements are not as significant as that of the more populated coil class and this leads to the reduction of the overall 3-state accuracy Q_3 .

6.3.3 Performance Comparison of the HNNP and the LIBSVM prediction

Table 6.6 shows and compares the performance of the C-SVC against the proposed HNNP. As the structure-structure prediction by the LIBSVM did not improve the prediction accuracy, the best results obtained from the first stage prediction are referenced for the comparisons. From the table, the Q_3 accuracy for the HNNP is 76.8% and 74.2% for the C-SVC. The C-SVC does not perform as well as the proposed HNNP, despite the fact that SVM is usually more accurate than neural networks. Further observations on the accuracies of each class reveal that the C-SVC is better at predicting the coil state and the accuracy Q_c is even higher than that of the HNNP. On the other hand, the accuracies of the C-SVC for the two other classes are much worse than that of the HNNP and this pulls down the overall prediction accuracy Q_3 . Considering that the coil is the most populated class in the training data, it seems that the C-SVC is more discriminating to different classes during the training when the dataset is not balanced.

6.4. Modular-SVMs for the PSSP problem

Method	Accuracy			
	$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$
C-SVC	74.2	73.9	55.7	83.7
HNNP	76.8	78.9	65.6	81.2

Table 6.6: Performance comparison for C-SVC Vs HNNP: the HNNP is superior to the C-SVC in terms of per-residue accuracy.

6.4 Modular-SVMs for the PSSP problem

The PSSP problems usually need large amount of protein data to train the prediction tools before the testing protein sequence could be predicted accurately. This in turn introduces a problem of timing. For the PSIPRED dataset used in this project, for example, it includes around 2×10^5 data points and each data point is a matrix with the dimension $N \times 21$: N is the size of the sliding window that the data spans in its neighborhood. Consequently, it requires a large amount of computations and it takes very long time to learn. The problem becomes more serious when a support vector machine is used as the training algorithm used by it is computation intensive. For experiments conducted in the previous section, the longest time that is taken for the training is about 6 days, and even the fastest training could only be done in one day. To address this problem, a Modular-SVMs network architecture is used.

The modular-SVMs network is originally proposed by Huang et al [94]. They inherited the concept of the modular network architecture for feedforward neural networks [95] and generalized it for support vector machines.

6.4.1 The modular network for feedforward networks

The basic idea of the modular network is to divide the problem of concern into L subregions such that each subproblem is solved independently. Suppose we have

6.4. Modular-SVMs for the PSSP problem

a set of patterns $S = \{x_i \in R^n; i = 1, 2, \dots, P\}$: n is the dimension of the pattern and P is the total number of patterns. To divide S into L groups or subregions, a random weight vector $w = [w_1, w_2, \dots, w_n]^T$ of dimension n is generated. Suppose x_i is a column vector, the set S is mapped to a space different from that of input by the product $b = w^T x$. Sort b in the ascending order and re-index the set such that $w^T x_1 \dots \leq w^T x_i \dots \leq w^T x_P$. Define a *bounding vector* $b = [b_0, b_1, \dots, b_k, \dots, b_L]$ for which

$$b_k = \begin{cases} w^T x_1, & \text{for } k = 0; \\ \frac{1}{2}(w^T x_{k[P/L]} + w^T x_{k[P/L]+1}), & \text{for } 1 \leq k < L; \\ w^T x_P, & \text{for } k = L; \end{cases} \quad (6.24)$$

where $[\bullet]$ represents the integer obtained by discarding the floating part. Subsequently, the dataset S is successfully divided into L groups with $[P/L]$ patterns in each subregion. The bounding pair (b_{k-1}, b_k) defines the lower and upper limits for the k -th group. A modular network architecture then could be constructed to work on these independent subregions.

Figure 6.5 illustrates the architecture of a modular network for feedforward networks. It is a four-layer network with two hidden layers between the input layer and the output layer. The input layer consists of n neurons for the n -dimensional input vector. The first hidden layer consists of two parts. The left part is p hidden neurons shared by L subproblem groups; the weights connecting to or from the neurons of this layer are trained independently using the data of the subregion. The right part contains L pairs of neurons (A, B) which are called the *quantizers*. Each pair of quantizer is only connected to one neuron of the second hidden layer and this gives a second hidden layer of size L . Note that the both hidden layers use the sigmoid activation function for their neurons. The output layer consists of one linear neuron for which the output is simply the linear combination of its input. The basic concept of the modular network is built up on the neural

6.4. Modular-SVMs for the PSSP problem

Figure 6.5: Architecture of the modular feedforward network with a group size of L

quantizers. When a pattern $x_i \in S_k$ (i.e., group k) is fed into the network, the network parameters, i.e., T , U and bias terms of the neural quantizers, are set such that only the k -th quantizer pair $(A^{(k)}, B^{(k)})$ contributes inputs of 0 to the successive layer. The rest quantizers' responses are almost 1. Hence, in the second hidden layer, the extremely large (negative) weight $-U$ make the inputs to all the neurons to be extremely large negative values except only the k -th neuron. As a result, only the k -th neuron contributes to the final output while the rest are “turned off” by the large negative inputs. Essentially, the L neural quantizer pairs serve as a group indicator for the pattern x_i and tell the network which set of weights should be used or learned for the left part shared hidden neurons. In this way, the network could fulfill the training and the testing on individual subregions successfully. The detailed mathematical derivations, formulations, etc., can be referred from [95]

6.4. Modular-SVMs for the PSSP problem

6.4.2 The Modular Network with Support Vector Machines

Based on the modular implementation for feedforward neural networks, the Modular Support Vector Machines (Modular-SVMs) is easily derived by generalization. Figure 6.6 shows the architectural layout of the Modular-SVMs. In this figure, each

Figure 6.6: Architecture of the Modular-SVMs with a group size L

shaded box represents a SVMs for solving a subregion problem and the GI unit is used to indicate the group that the input pattern x_i belongs to. Given a set of training data $TR = \{(x_i, t_i)\}_{i=1}^{P_{tr}}$, a random weight vector w is first generated. The training set TR is divided into L groups using bounding vector b which satisfies Equation 6.24. Each individual group is then treated as an independent subregion problem and trained using one SVMs. Note that “one SVMs” here may either be a single SVM for the binary classification or an assembly of SVMs for the multi-class

6.4. Modular-SVMs for the PSSP problem

classification. Once each individual SVMs has been trained, the machine modules are saved with indices of group number. In the testing, when the input pattern is fed into the Modular-SVMs, the *GI* unit would decide which subregion the data is in using the bounding vector b . The corresponding SVMs module is turned on and the machine parameters are loaded. In this way, the testing dataset is also implicitly split into L subregions and is tested on these subregions.

6.4.3 Speedup by the Modular-SVMs

Given a set of P training data, assume that the training complexity for the single SVMs is $O(P^\alpha)$. Now the training set is divided into L groups. The number of the training data for each SVMs becomes $P_k = P/L$. To train the single SVMs at each subregion, the time complexity now is of the order $O_k = O((P/L)^\alpha) = O(p^\alpha)/L^\alpha$. If all the SVMs modules are trained sequentially, the total time complexity for learning the whole problem is $L \times O_k = L^{1-\alpha}O(P^\alpha)$. Thus, theoretically speaking, the modular network can improve the speed of the training by at least $L^{\alpha-1}$ times in compare to the single SVMs case. As the time complexity for the single SVMs is at least quadratic [91] [89] [96], i.e., $\alpha \geq 2$, the Modular-SVMs is able to gain a speedup at least L . The larger the value L is, the more the speedup can be obtained.

6.4.4 Experiment with the Modular-SVMs

Strategy

In this experiment, the C-SVC is used as one SVMs module. As the experimental results for the single C-SVC case have shown that the second stage could not improve the prediction accuracy, the experiments for the Modular-SVMs would focus on the sequence-structure prediction. In these experiments, there are several parameters need to be tuned. These include the two parameters inherited from

6.4. Modular-SVMs for the PSSP problem

the single SVMs: γ , C ; the window size N and the additional group number L inherited from the Modular-SVMs. The strategy for searching optimal (γ, C) pair is the same as that used in the single SVMs case. For the group number L , usually a larger L leads to the higher speedup. Of course, the larger L on the other hand may also result the drop of the prediction accuracy. One extreme case is that the group number is equal to the number of the patterns, i.e., $L = P$. In this case, each data point serves as one training subset and supposedly, there would not be a good generalized performance. So, it is important that the tradeoff between the high speedup and the high prediction accuracy is balanced. A moderate L value may be more desirable than a larger one. Besides, an appropriate window size N also needs to be used.

Results

Table 6.7 shows the experimental results for window size $N = 15$, group size $L \in (5, 10\}$ and with various (C, γ) combinations. In early experiments, results on the single SVMs case have already shown that a window size of $N = 15$ would be optimal for the C-SVC. Thus, no further experiments are conducted with other window sizes. Table 6.8 compares the performance of the Modular-SVMs with the single SVMs in terms of the prediction accuracy and the training time.

From the table, it can be seen that the prediction accuracies Q_3 of the Modular-SVM are slightly lower than that of the single SVMs, with a 2% and 1.3% differences for the group size $L = 10$ and $L = 5$, respectively. The major decrease occurs in the prediction accuracy Q_α for the α -helix and Q_β for the β -strand. The secondary structure prediction for the coils is almost not affected. The reason of such change might due to the population distribution of three secondary structure classes. The population ratio for the α -helix, the β -strand and the coil class in the training data is around 3:2:4. The data points for the coil class are more sufficient

6.4. Modular-SVMs for the PSSP Problem

Group Size(L)	C	γ	Accuracies				Time (mins.)
			$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$	
10	1.0	2^{-7}	71.0	67.0	50.9	84.3	111.0
	1.0	2^{-6}	71.8	68.7	51.7	83.5	137.2
	1.0	2^{-5}	72.2	69.4	52.0	84.7	213.8
	2.0	2^{-7}	72.0	69.7	52.2	84.0	185.5
	2.0	2^{-6}	71.6	68.4	52.3	84.0	118.7
	4.0	2^{-7}	71.8	69.4	52.4	83.4	165.3
	4.0	2^{-6}	71.7	69.9	53.3	82.5	352.8
	8.0	2^{-6}	70.8	70.2	53.4	80.0	729.7
5	1.0	0.01	72.2	68.7	53.0	84.7	238.3
	1.0	0.03	72.8	70.4	53.1	84.61	431.8
	1.0	0.05	72.9	70.6	52.6	85.1	676.7
	1.0	0.1	71.2	66.2	44.6	88.8	729.7
	3.0	0.01	72.5	70.3	53.7	83.7	1362.5
	6.0	0.01	72.5	70.8	53.8	83.3	351.7
	10.0	0.01	72.1	71.4	54.5	81.6	703.9
	10.0	0.05	72.5	70.3	53.9	83.6	1165.4

Table 6.7: Prediction accuracies of the Modular-SVMs with different parameter set: $N = 15$, $L \in \{5, 10\}$ and various (C, γ) combinations

Method	Group L	Accuracies				Time (mins.)
		$Q_3(\%)$	$Q_\alpha(\%)$	$Q_\beta(\%)$	$Q_c(\%)$	
Modular SVMs	10	72.2	69.4	52.0	84.7	213.8
	5	72.9	70.6	52.6	85.1	676.7
Single SVMs	1	74.2	73.9	55.7	83.7	4919.0

Table 6.8: Accuracy and timing comparison for the Modular-SVMs and the single SVMs

6.4. Modular-SVMs for the PSSP Drobblem

than that for the other two classes. By dividing them into subregions, the number of the training data for coils is still sufficient and representative in its input space. For the rest two classes, data points in L subregions become less representative after dividing and this makes the learning machine to be less accurate in these two classes. As a result, the accuracies for these two classes are reduced.

Other than the prediction accuracy, the most important concern of applying the Modular-SVMs is to achieve the speedup in training. The last column of Table 6.8 shows the training time used by the Modular-SVMs and the single SVMs. By observing the column, it can be seen that the speedups for the Modular-SVMs are quite significant, reaching 23 and 7 times for $L = 10$ and $L = 5$ cases, respectively. Expressed in more straightforward way, the PSSP problem now could be solved using the Modular-SVMs in 1 afternoon instead of 3 days plus more than 1 afternoon. Without the significant drop in the prediction accuracy, these speedups have been impressive. Besides, by dividing the large PSSP problem into L subregions, the demand for advanced computational resources like CPUs and the memory storage becomes less critical. This has effectively conquered the common difficulties associated with the large-scale PSSP problem and supposedly, even larger protein dataset may be used to train the artificial machines.

Another issue that shall be addressed here is the overall Q_3 accuracy of the Modular-SVMs in comparison to the PSIPRED method. Although the Modular-SVMs is a bit low in accuracy than that of the PSIPRED method, it shall not be concluded that the difference is solely due to the modular networking. Rather, it is mainly affected by the learning capability of the single SVMs classifier. This impact has been discussed in detail in the single SVMs case. By comparing the learning time and the prediction accuracies with the same SVMs, it is more important that the Modular-SVMs could speedup the learning significantly with only slight drop

6.4. Modular-SVMs for the PSSP problem

in the prediction accuracy. The application of the Modular-SVMs on the PSSP problem is valuable especially when more and more new protein sequences are discovered and larger datasets may be used to train the machine.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In this thesis, a novel Hybrid Neural Networks Predictor (HNNP) was proposed for the Protein Secondary Structure Prediction (PSSP) problem. The proposed predictor consists of a Radial Basis Function Neural Network (RBFNN) for its sequence-structure prediction and a Multi-Layer Perceptron Neural Network (MLPNN) for the structure-structure prediction.

The RBFNN applied the overlapped clustering algorithm to set up the RBF kernels. The practice of allowing large overlaps for clusters of the same class and small overlaps for clusters of different classes helps to reject the noise disturbance to the RBFNN. Also, in order to formulate the distance measure for the windowed protein data, the distance measure scheme WINDist was proposed. The scheme tried to address the importance of neighboring residues and incorporate this information for the distance computation. To estimate the weights associated with the hidden layer and the output layer, the local-minima-prune Least Mean Square problem was solved.

The MLPNN was a simple 3-layer network trained using the traditional back-propagation algorithm. Due to the large scale and the high dimensionality of the training data, the backpropagation algorithm was found to be more appropriate

7.1. Conclusion

for the MLPNN in compare to other advanced algorithms like the Levenberg-Marquardt method and the Conjugate Gradient.

Experimental results showed that the proposed HNNP was able to reach the per-residue accuracy and the segment overlap at least comparable to the existing leading method. Among them, the first application of the RBFNN in the PSSP problem has been successful and this delivered the major Q_3 accuracy of 75.6%. It is believed that both the WINDist and the overlapped clustering technique has contributed actively for this high Q_3 accuracy. Based on the prediction of the RBFNN, the MLPNN has also successfully fulfilled the structure-structure prediction task. Some error predictions by the RBFNN were corrected by the MLPNN and this gave rise a 1.2% improvement. In all, the proposed HNNP is successful in learning and prediction for the PSSP problem and is able to compete with the leading PSSP algorithms such as PSIPRED.

In this thesis, the application of Support Vector Machines (SVM) for the PSSP problem was also studied. The online SVM library LIBSVM was used for both the sequence-structure (Q2T) prediction and the structure-structure (T2T) prediction. Like the proposed HNNP prediction, the LIBSVM was able to do the Q2T prediction with an accuracy higher than 74%. On the other hand, the T2T prediction by the LIBSVM did not seem to work well, due to the significant drop of the prediction accuracy for the coil class. By comparing the best results obtained by the LIBSVM and the HNNP, it was found that the single C-SVC's performance was not as good as that of the proposed HNNP.

Based on the LIBSVM, the Modular Support Vector Machines (Modular-SVMs) architecture was also applied to the PSSP problem. The major concern was to speed up the learning, especially when the dataset used in our experiments is large and high dimensional and requires a lot of computations. The results on our dataset

7.2. Recommendation

showed that the Modular-SVMs has successfully cut down the learning time by 7 to 23 times and meanwhile still maintained the relative high prediction accuracy while taking reference to the single SVMs case. Besides, the modular networking has also reduced the hardware requirements significantly. By the speedup and reduced requirements for the computational resource, more and more protein data may be used to “teach” machines and make the learned machines more accurate in the prediction. This is even more valuable when more and more new proteins are discovered in nowadays.

7.2 Recommendation

Due to the time constraint, there are still further work that is valuable yet has not been carried out. These include:

1. Feature weighting algorithm may be exploited to evaluate the importance of the different window positions. The discovered information shall more faithfully reflect the correlations associated with neighboring residues.
2. Different implementations of Support Vector Machines may be applied to the PSSP problem. The performance of different output encoding schemes, e.g., One-Versus-One, One-Versus-All, may also be studied.
3. More accurate SVMs may be used for the Modular-SVMs architecture. Remained high speedups and higher prediction accuracies are desirable. Moreover, the HNNP may also be used to replace the SVMs module for the PSSP problem.
4. The supersecondary structure building block identification problem, which is an important step towards building a tertiary structure from secondary structures, can be conducted.

7.2. Recommendation

5. The more difficult tertiary structure prediction may also be explored.

Bibliography

- [B] <http://www.ebi.ac.uk/swissprot/>.
- [2] <http://www.rcsb.org/pdb/>.
- [3] B. Sander. Protein structure prediction in 1d, 2d, and 3d. *Encyclopedia of Computational Chemistry, N.L.A. P. von Rague-Schleyer, T. Clark, J. Gasteiger, P. A. Kollman, H.F., Editor*, pages 2242–2255, 1998.
- [4] C. Sander W. Kabsch. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577C637, 1983.
- [5] S. Rackovsky A. D. Solis. Optimized representations and maximal information in proteins. *Proteins: Struct. Funct. Genet.*, 38:149–64, 2000.
- [6] S. Rackovsky A. D. Solis. Optimally informative backbone structural propensities in proteins. *Proteins: Struct. Funct. Genet.*, 48:463–86, 2002.
- [7] P. Rotkiewicz B. Ilkowski A. R. Ortiz, A. Kolinski and J. Skolnick. Ab initio folding of proteins using restraints derived from evolutionary information. *Proteins: Suppl.*, 3:177–185, 1999.
- [8] A. K. Felts V. A. Eyrich, D. M. Standley and R. A. Friesner. Protein tertiary structure prediction using a branch and bound algorithm. *Proteins*, 35:41–57, 1999.

Bibliography

- [9] I. D. Pogozheva A. L. Lomize and H. I. Mosberg. Prediction of protein structure: The problem of fold multiplicity. *Proteins Suppl.*, pages 199–203, 1999.
- [10] P. Koehl R. Samudrala, E. S. Huang and M. Levitt. Constructing side chains on near-native main chains for ab initio protein structure prediction. *Proteins Eng.*, 13:453–457, 2000.
- [11] W. R. Taylor and J. M. Thornton. Prediction of super-secondary structure in proteins. *Nature*, 301:540–542, 1983.
- [12] H. L. Holley and M. Karplus. Protein secondary structure prediction with a neural network. *Proc. Natl. Acad. Sci., U.S.A.*, 86:152–156, 1989.
- [13] P. Stolorz; A. Lapedes and Y. Xia. Predicting protein secondary structure using neural net and statistical methods. *J. Mol. Biol.*, 225:363–377, 1992.
- [14] Riis S.K. Combining neural networks for protein secondary structure prediction. neural networks. *Proceedings, IEEE International Conference on*, 4:1744–1748, 1995.
- [15] P. Baldi; S. Brunak; P. Frasconi; G. Soda and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.
- [16] Sujun Hua and Zhirong Sun. A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *J. Mol. Biol.*, 308:397–407, 2001.
- [17] Jagath C. Rajapakse Minh N. Nguyen. Multi-class support vector machines for protein secondary structure prediction. *Genome Informatics*, 14:218–227, 2003.

Bibliography

- [18] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, 232:584–99, 1993.
- [19] B. Rost and C. Sander. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins*, 19:55–72, 1994.
- [20] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, 292:195–202, 1999.
- [21] J. C. Kendrew; R. E. Dickerson; B. E. Strandberg; R. J. Hart; D. R. Davies et al. Structure of myoglobin: A three-dimensional fourier synthesis at 2 åresolution. *Nature*, 185:422–427, 1960.
- [22] M. F. Perutz; M.G. Rossmann; A. F. Cullis; G. Muirhead; G. Will et al. Structure of haemoglobin: A three-dimensional fourier synthesis at 5.5 åresolution. *Nature*, 185:416–422, 1960.
- [23] H. A. Scheraga. Structural studies of ribonuclease iii. a model for the secondary and tertiary structure. *J. Am. Chem. Soc.*, 82:3847–3852, 1960.
- [24] D. R. Davids. A correlation between amino acid composition and protein structure. *J. Mol. Biol.*, 9:605–609, 1964.
- [25] B. Robson and R. H. Pain. Analysis of the code relating sequence to conformation in proteins: Possible implications for the mechanism of formation of helical regions. *J. Mol. Biol.*, 58:237–259, 1971.
- [26] P. Y. Chou and U. D. Fasma. Prediction of protein conformation. *Biochem.*, 13:211–215, 1974.

Bibliography

- [27] V. I. Lim. Structural principles of the globular organization of protein chains. a stereochemical theory of globular protein secondary structure. *J. Mol. Biol.*, 88:857–872, 1974.
- [28] K. Nagano. Triplet information in helix prediction applied to the analysis of super-secondary structures. *J. Mol. Biol.*, 109:251–274, 1977.
- [29] M. J. Rooman; J. P. Kocher and S. J. Wodak. Prediction of protein backbone conformation based on seven structure assignments: Influence of local interactions. *J. Mol. Biol.*, 221:961–979, 1991.
- [30] H. Bohr; J. Bohr; S. Brunak; R. M. J. Cotterill; B. Lautrup et al. Protein secondary structure and homology by neural networks. *FEBS Lett.*, 241:223–228, 1988.
- [31] D. G. Kneller; F. E. Cohen and R. Langridge. Improvements in protein secondary structure prediction by an enhanced neural network. *J. Mol. Biol.*, 214:171–82, 1990.
- [32] S. Muggleton; R. D. King and M. J. E. Sternberg. Protein secondary structure predictions using logic-based machine learning. *Prot. Engin.*, 5:647–657, 1992.
- [33] A. A. Salamov and V. V. Solovyev. Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignment. *J. Mol. Biol.*, 247:11–15, 1995.
- [34] N. Qian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202:865–84, 1988.

Bibliography

- [36] F. Vivarelli; G. Giusti; M. Villani; R. Campanini; P. Fraisselli; M. Compiani and R. Casadio. Lgann: a parallel system combining a local genetic algorithm and neural networks for the prediction of secondary structure of proteins. *Computer Applications in the Biosciences*, 11:763–9, 1995.
- [37] C. Ruggiero; R. Sacile; G. Rauch. A hybrid algorithm for determining protein structure. *Biomedical Engineering, IEEE Transactions on*, 40(11):1114 – 1121, 1993.
- [38] J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Struct. Funct. Genet*, 34:508–519, 1999.
- [39] Owen Lamont; Hiew Hong Liang and Matthew Bellgard. Data representation influences protein secondary structure prediction using artificial neural networks. *Intelligent Information Systems Conference, The Seventh Australian and New Zealand 2001*, 18-21:411 – 416, 2001.
- [40] Xiru Zhang. A hybrid algorithm for determining protein structure. *Expert, IEEE*, 9(4):66–74, 1994.
- [41] B. Rost. Phd: Predicting one-dimensional protein secondary structure by profile-based neural network. *Methods in Enzymology*, 266:525–39, 1996.
- [42] S. K. Riis and A. Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3:163–183, 1995.
- [43] F. A. Stephen; G. Warren; M. Webb; W. M. Engene and J. L. David. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

Bibliography

- [44] Stephen F. Altschul; Alejandro A. Schaffer; Jinghui Zhang; Zheng Zhang; Webb Miller and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
- [45] <http://www.ncbi.nlm.nih.gov/BLAST/>.
- [46] C. Watkins J. Weston. *Multi-class Support Vector Machines*. In Verleysen, M., editor, *Proceedings of ESANN99*, Brussels, D. Facto Press, 1999.
- [47] V. N. Vapnik. *Statistical Learning Theory*. New York: Wiley, 1998.
- [48] Xiuju Fu and Lipo Wang. Data dimensionality reduction with application to simplifying rbf network structure and improving classification performance. *IEEE Transactions on Systems, Man, And Cybernetics-part B: Cybernetics*, 33(3):399–409, 2002.
- [49] D. Eisenberg M. Gribskov, A. D. McLachlan. Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci U S A*, 84(13):4355–8, 1987.
- [50] Martin Tompa. *Biological Sequence Analysis*. 2000.
- [51] K. Hornik; M. Stinchcombe and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [52] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):193–192, 1989.
- [53] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

Bibliography

- [54] E. J. Hartman; J. D. Keele and J. M. Kowalshi. Layered neural networks with gaussian hidden units as universal approximations. *Neural Computation*, 2(2):210–215, 1990.
- [55] John S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parametersin. *Advances in Neural Information Processing Systems*, 2:211–217, 1990.
- [56] P. Maffezoni and P. Gubian. Approximate radial basis function neural network to learn smooth relationships from noisy data. *Proceedings of the 37-th Midwest Symposium on Circuits and Systems*, 1:553–556, 1994.
- [57] Mark J. L. Orr. *Introduction to Radial Basis Function Networks*. 1996.
- [58] D. E. Rumelhart; G. E. Hinton and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [59] R. Schneider C. Sander. Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.*, 9:56–68, 1991.
- [60] R. Schneider C. Sander. Twilight zone of protein sequence alignments. *Protein Eng.*, 12:85–94, 1999.
- [61] C. Chothia S. E. Brenner and T. J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl Acad. Sci. USA*, 95:6073C6078, 1998.
- [62] T. Koike R. Lopez T. J. Gibson D. G. Higgins R. Chenna, H. Sugawara and J. D. Thompson. Multiple sequence alignment with the clustal series of programs. *Nucl. Acids Res.*, 31:3497C3500, 2003.

Bibliography

- [63] M. Cline M. Diekhans L. Grate K. Karplus, C. Barrett and R. Hughey. Predicting protein structure using only sequence information. *Proteins: Struct. Funct. Genet.*, S3:121C125, 1999.
- [64] B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochim. Biophys. Acta*, 405:442–451, 1975.
- [65] B. Rost; C. Sander and R. Schneider. Redefining the goals of protein secondary structure prediction. *J. Mol. Biol.*, 235:13–26, 1994.
- [66] A. Zemla; C. Venclovas; K. Fidelis and B. Rost. A modified definition of sov, a segment-based measure for protein secondary structure prediction assessment. *PROTEINS: Structure, Function, and Genetics*, 34:220–223, 1999.
- [67] C. H. Wu and J. W. McLarty. *Neural Networks and Genome Informatics, 1st Edition*. Elsevier Health Sciences, 2000.
- [68] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Comput. J.*, 7:149–154, 1964.
- [69] E. Polak. *Computational Method in Optimization: a Unified Approach*. Academic Press, New York, 1971.
- [70] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.
- [71] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.
- [72] B. Boser; I. Guyon and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

Bibliography

- [82] G. T. C. Van; J. A. K. Suykens; D. E. Baestaens; A. Lambrechts; G. Lanckriet ; B. Vandaele ; B. De Moor and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on*, 12(4):809–821, 2001.
- [83] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probability and Its Applications*, 17:264–280, 1971.
- [84] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I and II*, Belmont. MA:Athenas Scientific; 1995.
- [85] Simon Haykin. *Neural Networks: A Comprehensive Foundation, second edition*. Prentice Hall International, 1999.
- [86] Chi-Chung Chang and Chih-Jen Lin. *LIBSVM: a Library for Support Vector Machines*. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [87] B. Scholkopf; A. Smolar; R. C. Williamson and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.
- [88] B. Schölkopf; J. C. Platt; J. Shawe-Taylor; A. J. Smola and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.
- [89] T. Joachims. *Making large-scale SVM learning practical*. In B. Scholkopf, C.J.C. Burges and A.J.Smola(Eds.), *Advances in Kernel Methods - Support Vector Learning*, Cambridge, Ma. MIT Press, 1998.
- [90] E. Osuna; Freund **and** F. Girosi. Training support vector machines: An application to face detection. *Proceedings of CVPR'97, New York, NY*, pages 130 136, 1997.

Bibliography

- [91] J. C. Platt. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. In B. Scholkopf, C.J.C. Burges and A.J.Smola(Eds.), *Advances in Kernel Methods - Support Vector Learning*, Cambridge, Ma. MIT Press, 1998.
- [92] L. Bottou; B. Scholkopf C. Saunders; M. O. Stitson; J. Weston and A. Smola. *Support Vector Machine Reference Manual*. Technical Report CSD-TR-98-03, Royal Holloway, University of London, Egham, UK, 1998.
- [93] Kuan-Ming Lin and Chih-Jen Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1449–1559, 2003.
- [94] Guang-Bin Huang; Ke Zhi Mao and Chee-Kheong Siew. *Fast Modular Network Implementation for Support Vector Machines*. School of Electrical and Electronic Engineering, Nanyang Technological University, Technical Report: ICIS/47/2003, 2003.
- [95] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *Neural Networks, IEEE Transactions on*, 14(2):274–281, 2003.
- [96] R. Collobert; S. Bengio and Y. Bengio. A parallel mixtures of svms for very large scale problems. *Neural Computation*, 14:1105–1114, 2002.