Cognitively inspired rule-plus-exemplar based pattern classification

Sit, Wing Yee

2014

Sit, W. Y. (2014). Cognitively inspired rule-plus-exemplar based pattern classification.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61749>

<https://doi.org/10.32657/10356/61749>

**Cognitively Inspired Rule-Plus-Exemplar based
Pattern Classification**

Sit Wing Yee

School of Electrical & Electronic Engineering

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

2014

Acknowledgments

By all accounts, the PhD experience should have been a painful one where I can only try to lessen my misery. Yet for reasons that I can only refer to as blessings, it has turned out to be a fulfilling and even enjoyable journey. There are too many reasons why, and I can only try to scrape the surface.

My utmost respect and gratitude for Dr Mao Kezhi, my full time supervisor and part time counselor, mentor, and teacher. None of this would have been possible without his guidance, whether on technical knowledge or personal development. His enthusiasm has kept me interested in the research; and his patience, encouragement and faith in me has kept me motivated throughout the process. This thesis would not have been possible without him.

My seniors Dr Tang Wenyin and Dr Yang Feng have graduated and moved on to increasingly better opportunities, but not before imparting knowledge and skills as well as materials and resources. The inspiring and thought invoking discussions have benefited me greatly, as do the experience that they still generously share.

The wonderful environment under which I completed this journey has been made possible by the University, not just in terms of the scholarship, but also the electronic and physical resources. The support and assistance from our lab technicians and administrative officers have also contributed to a positive overall experience, which was made even more fulfilling from my time with the Graduate Students' Club.

Indeed, the journey has not always been smooth sailing, but any time I encountered setbacks, it was a relief to be able to reach out to my colleagues and friends from EEE as well as other schools and offices. Misery loves company, but more so the understanding and peer support. It is amazing how they manage to brighten up the cloudy days and the everyday.

Much appreciation also goes to Dr Ng Gee Wah and Dr Mak Lee Onn of DSO, who started me on this journey and continued to encourage me in various ways.

Finally, my friends and family who sometimes claim to know nothing about what I go through, yet manage to give me all the support I ever need; I feel most blessed and grateful for them.

Table of Contents

Summary	iv
List of figures	vii
List of tables	viii
List of notations	ix
1 Introduction	1
1.1 Motivation	2
1.2 Objectives	4
1.3 Original contributions	5
1.4 Organization of thesis	6
2 Context and background	9
2.1 Pattern classifiers	10
2.1.1 Lazy learning methods	11
2.1.2 Eager learning methods	16
2.2 Classification issues arising from machine learning	26
2.2.1 Comprehensibility of classification	26
2.2.2 Learning and classifying in a dynamic environment	29
2.2.3 Bias from imbalanced classes	33
2.3 Evaluation of classifier performance	35
2.3.1 Data for evaluation	36
2.3.2 Evaluation metrics	37
2.4 Cognitive psychology	39
2.4.1 Categorization models	40
2.4.2 Structure of memory	41
2.4.3 Learning in cognitive psychology	42
2.5 Rule-plus-exemplar pattern classifiers	43
3 A rule-plus-exemplar framework for overcoming comprehensibility issues	45
3.1 Difficulty of improving comprehensibility of classifiers	46
3.2 Improving generalizability of comprehensible classifiers	49

3.3	Overall rule-plus-exemplar system	53
3.4	Experiments and results	55
3.4.1	Experimental data	56
3.4.2	Targeting poor performance regions with exemplars	56
3.4.3	Targeting low coverage rules with exemplars	58
3.5	Analysis	64
3.6	Chapter summary and concluding remarks	68
4	Rule-plus-exemplar framework for classification extrapolation	70
4.1	Motivation and problem description	71
4.2	Characterization of the training region.....	78
4.2.1	Choosing the spread parameter value	79
4.2.2	Determining the cut-off threshold value for identifying the training region.....	85
4.2.3	Identifying the training region	88
4.3	Reduced representation of the training region	89
4.4	Extrapolating beyond the training region.....	94
4.5	Experiments and results	98
4.5.1	Experimental dataset	98
4.5.2	Classification of unfamiliar samples outside the training region.....	100
4.5.3	Identification of training region	102
4.5.4	Improvement in extrapolation capability	104
4.6	Chapter summary and concluding remarks	105
5	Rule-plus-exemplar classification system for concept growth	107
5.1	The concept growth problem.....	108
5.1.1	Problem description and motivation	108
5.1.2	Problems with using concept drift techniques on the concept growth problem.....	111
5.1.3	Problems with using incremental learning techniques on the concept growth problem	113
5.2	Rule-plus-exemplar classification system.....	116
5.2.1	Difference between proposed system and existing methods	117
5.2.2	Architectural structure of the proposed system	119
5.2.3	Learning through rule module generation.....	121

5.2.4	Classification of a test sample.....	124
5.2.5	Implementing the rule-plus-exemplar classification system in concept growth environment.....	125
5.2.6	Rationale behind the rule-plus-exemplar classification system.	128
5.3	Experiments and results	130
5.3.1	Experimental data and simulating concept growth	130
5.3.2	Experimental setup and results	132
5.3.3	Analysis.....	135
5.4	Chapter summary and concluding remarks	141
6	Rule-plus-exemplar system for imbalanced classes undergoing concept growth	143
6.1	Imbalanced classes in concept growth	144
6.1.1	Problem description	144
6.1.2	Difficulty of applying techniques for imbalanced data in a concept growth environment	145
6.1.3	Difficulty to overcoming concept growth introduced by imbalanced classes	146
6.2	Using ideas from cognitive psychology	149
6.3	Rule-plus-exemplar classification system	152
6.3.1	Ambiguous region, R_A	154
6.3.2	Familiar region R_F	156
6.3.3	Unfamiliar region R_U	162
6.3.4	Rationale behind the rule-plus-exemplar system for imbalanced classes	162
6.3.5	Overall classification system and process.....	164
6.4	Experiments and results	166
6.4.1	Experimental data	166
6.4.2	Experimental setup and results	170
6.4.3	Analysis.....	173
6.5	Chapter summary and concluding remarks	175
7	Conclusion	178
	List of publications	183
	References.....	184

Summary

Despite the many strengths of machine learning pattern classification techniques, they have intrinsic weaknesses compared to human learning, which is far more comprehensible, adaptive, and flexible. These are issues that need to be addressed. In applications where the classifiers play supporting roles in decision making, comprehensibility is required, restricting the kinds of classifiers that can be used. In a dynamic environment, adaptation is important, particularly in view of evolving concepts. The training data that was originally provided may not be sufficient to represent the concept as it evolves. These qualities natural to humans are significantly more difficult to achieve efficiently in machines. To imitate these desired qualities, ideas are drawn from cognitive psychology, based on which machine learning classifiers are structured into systems that can better deal with such tasks. The relevant categorization models using rules and exemplars are employed as the basis of such systems.

First, we explore the comprehensibility issue. This trait is most directly achieved through the use of explicit IF-THEN rules that are directly interpretable to humans. However, comprehensibility often comes at reduced complexity of the classifier and thus poorer generalization performance. This tradeoff between interpretability and accuracy is well recognized. As humans use both the rule and exemplar models in cognitive psychology for categorization tasks, these models are intuitive to us and are largely sufficient in most tasks. The use of the exemplar model thus complements and aids the commonly used rule model in classification to improve generalization

capability. The resulting system shows marked improvement in classification accuracy compared to common interpretable classifiers.

The next problem considered is that of extrapolative generalizability. Classifiers learn from training data that is assumed to be representative of the concept to be learnt, but this condition is not always satisfied. Test samples sometimes do not arise from the represented concept, resulting in classification that is inconsistent and erratic. We propose a framework for dealing with such data, using the rule and exemplar models from cognitive psychology. The key to such a framework lies in the correct identification of the training region, which is the region in input feature space that has been covered by the training samples. The method provided is non-parametric and not dependent on any specific classification technique used. There are also no assumptions on the shape and density of the training samples in the input feature space. Experiments show that the system can extrapolate well as the samples outside the training region are correctly identified and appropriately handled.

Adaptability of classifiers is another desired quality but has received far more attention than the previous two problems. Changing concepts have been explored in the context of concept drift. Due to the change in concept, the classifier needs to adapt quickly so that it remains relevant and can properly classify the sample stream that is being presented to it. However, the problem of concept growth is also present in many applications. It is more difficult to solve because unlike in concept drift, the previous data cannot simply be forgotten. The rule-plus-exemplar structure is adopted in a new system that specifically deals with concept growth, and results show that the system

achieves a good balance of stability and plasticity to learn and correctly classify a changing sample stream.

Finally, as the concept growth problem often occurs in applications with imbalanced classes, a different learning and classification system is given. Even though it is based on the rule-plus-exemplar structure, the models are incorporated in a completely different way to deal with both issues simultaneously. There has been little existing work on this problem, and the available methods are compared with the proposed system. The performance of the rule-plus-exemplar system is significantly better and more consistent than other methods.

List of figures

Figure 2.1 Basic architecture for ANN	17
Figure 2.2 Outline for construction of decision tree	22
Figure 3.1 Simple vs complex boundaries	47
Figure 3.2 Algorithm for finding poor performing rules	52
Figure 3.3 Training of comprehensible rule-plus-exemplar system	53
Figure 3.4 Testing of comprehensible rule-plus-exemplar system	54
Figure 3.5 Distribution of rules with different coverage	60
Figure 4.1 Sample rules and features used in rule antecedents	72
Figure 4.2 Distribution of rules with different number of features	74
Figure 4.3 Erratic behavior of decision boundaries outside training region	76
Figure 4.4 Contour plot of under-regularized $T(x)$ values	81
Figure 4.5 Contour plots of $T(x)$ on various datasets	84
Figure 4.6 Identified training regions	87
Figure 4.7 Algorithm for identifying the training region	89
Figure 4.8 Algorithm for identifying training region with reduced representation	90
Figure 4.9 Identified training regions under reduced representation	93
Figure 4.10 Intuitive extension of decision boundary outside training region ..	96
Figure 4.11 Rule-plus-exemplar system for testing data that may fall outside training region	97
Figure 4.12 Algorithm for generating testing data outside training region	99
Figure 5.1 Unreliable extension of decision boundary under incomplete information in concept growth	115
Figure 5.2 Architecture of rule-plus-exemplar system for concept growth	119
Figure 5.3 Algorithm for compacting exemplars in episodic buffer	123
Figure 5.4 Algorithm for rule-plus-exemplar for concept growth	127
Figure 5.5 Algorithm for simulating concept growth	132
Figure 6.1 Intermediate snapshots of imbalanced data under concept growth	146
Figure 6.2 Inconsistency arising from SMOTE under concept growth conditions	148
Figure 6.3 Specialization of each identified partition	153
Figure 6.4 Relevant samples for classification of unfamiliar samples	160
Figure 6.5 Overall rule-plus-exemplar system and process for imbalanced classes in concept growth	164
Figure 6.6 Algorithm for generating concept growth for imbalanced classes .	168
Figure 6.7 Artificially generated 2-dimensional imbalanced data	169

List of tables

Table 2.1 Contingency table	37
Table 3.1 Samples and errors in poor performance rules	57
Table 3.2 Samples and errors in low coverage rules	61
Table 3.3 Reduced error rates using comprehensible rule-plus-exemplar system	63
Table 3.4 Comparison of error rates between black-box and comprehensible classifiers.....	65
Table 3.5 Average number of rules generated for rule-plus-exemplar classifiers	67
Table 4.1 Comparison of error rates on internal and external test sets	101
Table 4.2 Correct identification of samples inside and outside training region	103
Table 4.3 Improvements in error rates using exemplars outside training region	105
Table 5.1 Comparison of error rates for incremental validation under concept growth	134
Table 5.2 Comparison of error rates for incremental learning in concept growth using 10-fold cross validation	136
Table 5.3 Computational time and stored rules and exemplars for rule-plus- exemplar system for concept growth	138
Table 5.4 Error rates on actual vowel data	140
Table 6.1 Details of imbalanced data used	170
Table 6.2 Comparison of MCC values on imbalanced data under concept growth	172
Table 6.3 Comparison of classification of imbalanced classes under concept growth	173
Table 6.4 Analysis of regions identified	174

List of notations

Notation	Description
x	Sample
s_i, x_i	i -th training samples
$x(i)$	i -th feature of sample x
m	Number of samples in a dataset
$P_n(S)$	n -th percentile of set S
$T(x)$	Evaluation function for characterizing training region
$T_R(x)$	Evaluation function for reduced representation of training region
$T_i(x)$	Evaluation function for i -th rule module
$T_A(x)$	Evaluation function for characterizing ambiguous region
$T_F^{(i)}(x)$	Evaluation function for i -th rule to characterize familiar region
τ	Threshold for $T(x)$ to identify training region
τ_R	Threshold for $T_R(x)$ to identify training region
τ_i	Threshold for $T(x)$ of i -th rule module
τ_A	Threshold for $T_A(x)$ for determining ambiguous region
$\tau_F^{(i)}$	Threshold for $T_F^{(i)}$ for determining familiar region
$F_i(x)$	i -th Gaussian function
σ_i	Spread parameter for i -th Gaussian function
p_τ	Percentile for calculating threshold τ
p_c	Percentile for calculating σ from cluster
p_τ	Percentile for computing threshold τ_F
p_s	Portion of samples in familiar region to be stored

n_σ	Number of neighbors for calculating σ_i 's
r	Number of cluster centers for reduced representation
C_i	Cluster i for reduced representation
c_i	Cluster centers cluster C_i
R	Training region
R_A, R_F, R_U	Ambiguous, familiar and unfamiliar region
$a_i(x)$	Activation value for i -th rule module
$A(x)$	Set of activated rules for sample x
$A^*(x)$	Set of activated rule module indices with prediction ω^* for sample x
$csty(x)$	Consistency of rule subsystem prediction for sample x
ξ	Minimum value of $csty(x)$ for consistency
ρ	Target ratio of minority to majority class samples
ρ_{err}	Cut-off point for rule error
ρ_{cov}	Cut-off point for rule coverage
X_e, X_b	Set of samples in episodic long term memory; episodic buffer
$X_U, X_\neq$	Set of unfamiliar samples; misclassified samples
m_e	Maximum threshold allowed for total size of $X_U \cup X_\neq$
m_c	Minimum cluster size for generation of rule module
m_b	Maximum capacity of episodic buffer
d	Number of nearest ‘-’ samples for computing spread parameter

Chapter One

Introduction

Research in the field of machine learning pattern classification has yielded a large number of classification techniques. Some well known methods include the statistical methods like Naïve Bayes (Domingos and Pazzani 1997), simple instance similarity based methods such as k-nearest neighbors (KNN) (Cover and Hart 1967), intuitively comprehensible decision tree methods (Quinlan 1986) like classification and regression trees (CART) (Breiman 1984) and Quinlan's C4.5 (Quinlan 1993), brain inspired models like artificial neural networks (ANN) (Rumelhart, McClelland et al. 1986) and their variant using radial basis functions (RBF) (Broomhead and Lowe 1988), as well as the popular separation based support vector machines (SVM) (Cortes and Vapnik 1995). Each technique has its own strengths and limitations, so different techniques are suited to different problems. A wide range of variants for each

technique has been proposed in literature, aimed at addressing the weaknesses of each method and helping them to better perform under different situations.

1.1 Motivation

Pattern classification techniques offer a number of strengths and advantages over human abilities. While machines can process large amounts of data and higher number of attributes, humans have a more limited processing capacity. Although a person may encounter and remember a remarkable number of instances over a period of time and make meaningful inferences from such instances, he or she cannot simply take in the same volume of data in a short period of time and make the same generalizations on that data. In addition, complexity of the concept represented by the available data may be more than what humans can cope with, except perhaps extremely rare domain experts. Besides the difficulty in learning, the classification or utilization of knowledge may also face problems. Humans are easily influenced and affected by recent or exceptional events as well as the external environment, such that they could arrive at different decisions for the same case under different situations. There is thus a lack of reproducibility and objectiveness in classifications performed by humans. Finally, in tasks like surveillance or monitoring, it is far easier to allow machines to take over rather than assign an officer to scan through large amounts of data for long periods of time. This may lead to fatigue and risk possible oversights.

Despite the strengths and potential improvements offered by machines, pattern classification has not been applied as widely as it should have. Even though classifiers have been shown to perform well in tests, applying them in

1.1 Motivation

the real life problems and environments throws up many other issues that need to be resolved. The data that is actually provided to the classifiers for learning may not be of the assumed quality. For example, patient data that is collected may not always be complete. Some fields may not be available, or readings and measurements may have been wrong. Data available could also be limited, especially in one of the classes. This is the scenario faced in the detection of fraud cases, which are far and few compared to regular cases, but yet far more important to identify. Another issue is that the data provided may not even correctly represent the concept that the classifier has to learn. Training a speech recognition system with one or two speakers can barely represent the differences in all possible speakers that will be encountered by the system. Or the concept being represented can simply be changing and evolving over time. For example, a spam filtering system may be trained with abundant samples of spam mail, but as new disguises are created every day, the concept changes and the training data becomes outdated. All these issues violate some assumptions that the classifiers may have been trained under. Some classifiers are designed as decision support, requiring a certain amount of comprehensibility in the way they work. But many classifiers function as a black-box, taking the input and providing only the final output with no insight into their workings. This limits their use in applications such as medical diagnosis or safety critical applications.

Yet for all these limitations that are encountered by pattern classifiers in actual applications, humans are much better at handling them. We can work with incomplete information, such as recognizing partially occluded objects. We are able to pay attention to categories with limited samples, such as recognizing a rare animal or flower, even if we have very limited encounters

1.2 Objectives

with them compared to other categories. Knowledge extracted can be applied to widely different cases, such as identifying speech or written characters by people who were not part of the samples during learning. We are also adept at adjusting our knowledge to changing or evolving concepts, as shown in our ability to understand trends in weather forecasts, economic outlook or simply in fashion. We can explain why we make certain inferences or conclusions, allowing communication and transfer of such knowledge. Therefore despite the many strengths and the potential of pattern classifiers, there is much to learn from the human way of categorization, and thus they stand to gain better applicability and acceptance from being more human-like. In particular, we are interested in making classifiers more comprehensible to humans, as well as more adaptive to changes in the environment or limited data.

1.2 Objectives

In view of the shortcomings of pattern classification techniques in comparison to human cognitive abilities, this thesis seeks to tap on ideas from cognitive psychology to incorporate more human-like classification qualities. Although cognitive psychology is a large area still undergoing development, we can focus on the most relevant processes of categorization and learning, as well as the structure of memory which is crucial in supporting the categorization process. Of particular relevance are the rule and exemplar categorization models, which are thus the basis of our classification framework. We aim to use this framework to provide a novel way to present classification in a comprehensible way. In the extrapolation of the learnt classification boundary, we will also use the rule-plus-exemplar concept to provide a more consistent

and intuitive way of handling samples requiring extrapolation. The concept will also be extended to a framework for a classification system to deal with changing concepts. In the situation where data of one class are drastically limited, we seek to employ the rule-plus exemplar framework with necessary modifications to facilitate learning and classification of imbalanced data under a concept change.

1.3 Original contributions

This thesis provides four main original contributions that are based on ideas from cognitive psychology giving rise to the rule-plus-exemplar framework.

First, recognizing the problem of comprehensibility in many pattern classification techniques, we explore the combination of rule and exemplar models which are cognitively intuitive to humans, to improve on classification performance. The tradeoff between accuracy and comprehensibility has been reported on, but investigations have mainly been restricted to rule based classification. The integration of exemplar based classification is a new direction to improving classification accuracy while maintaining comprehensibility.

The next contribution is targeted at the violation of the assumption in pattern classification learning that the training samples are representative of the problem. As this assumption may not be satisfied in many real life applications, the classification boundary created will be extrapolated beyond the training region, leading to erratic and unreliable classifier behavior. We provide a framework which identifies where the classifier may no longer be classifying

1.4 Organization of thesis

consistently, and offers an intuitive as well as reliable alternative classifier in those regions.

In addition to providing a reliable way to classifying outside the training region, we extend the framework to learn these new samples. This framework can be used for different classifiers, even those which do not have incremental learning capability. The overall framework can continuously be updated with new information being presented in a concept growth environment.

The fourth contribution is a classification framework that can not only deal with classification boundary extrapolation and incremental learning, but do so in an environment with class imbalance. This is a difficult problem to overcome, and our framework copes well with these practical issues. The framework is applicable with various classifiers, even if they are not each tailored to handle class imbalances.

1.4 Organization of thesis

Chapter 2 of the thesis will begin with background knowledge and related information. It explains pattern classification and the types of classifiers available, in a way that will facilitate better understanding of how ideas from cognitive psychology are relevant. The desired traits for classifiers will help them to overcome certain limitations – lack of comprehensibility, poor response to changes in the environment, and overwhelming influence of the majority class in imbalanced classes. These limitations are also discussed in Chapter 2, along with the existing techniques that are currently available for each problem. As the proposed systems employ a rule-plus-exemplar structure that is

cognitively inspired, the relevant ideas from cognitive psychology are also presented.

Chapter 3 looks into the comprehensibility issue and its tradeoff with accuracy. In view of this tradeoff, we present a different way of achieving a more comprehensible classification. Instead of using only the traditional IF-THEN rules that are known to be interpretable to humans, exemplars are an equally important part of our intuitive cognitive categorization process. Using a combination of both rules and exemplars, we propose a system that can classify better but still remains comprehensible.

Chapter 4 considers the difficulty of extrapolating the learnt classifier for samples that have not been adequately represented by the training data. A rule-plus-exemplar framework is proposed, which can better classify such test samples. The heart of the system lies in identifying the training region in the input feature space where the training samples are covering. Through the correct identification of the training region and appropriate treatment of test samples outside the region, the classification boundaries obtained by the classification system extrapolate in a more intuitive way and achieves an improvement in classification performance.

Chapter 5 delves in to the problem of changing environments, specifically the concept growth problem. It is different from the well-studied concept drift scenario, and the nature of the problem makes it difficult to solve. The classifier not only has to adapt to the changes in concept, but also has to retain what it has learnt without discarding or interfering with them, thus highlighting the stability-plasticity dilemma. These are detailed in Chapter 5, which also

elaborates how our proposed classification system can effectively learn and classify even under concept growth.

As the concept growth problem often occurs in applications that also suffer from imbalanced classes in datasets, this issue is investigated in Chapter 6. While the cognitively inspired rule-plus-exemplar structure remains relevant, it has to be implemented in a completely different way. The method for identifying the training region proposed in Chapter 4 is used in a different way. In addition to identifying the training region, it is also used to identify the region of interest where the limited minority class samples are.

The thesis concludes in Chapter 7, which shows how the rule-plus-exemplar framework adopted from cognitive psychology gives way to classification systems that can behave in more human-like ways. There is increased comprehensibility and adaptability – qualities in classification which are more intrinsic to humans rather than machines. The individual systems are effective, but there are also open questions whose answers can further benefit the systems.

Chapter Two

Context and background

This chapter provides the context and background of the work that will be discussed. Relevant knowledge in pattern classification is first presented, followed by the problems and issues of classification that are in contrast to human ability in categorization. The available techniques for these issues are discussed, and it will be explained why they are not adequate. As ideas from cognitive psychology will be utilized to overcome these problems, a look into the related background knowledge on memory structure and categorization models will be given. The rule plus exemplar categorization models from cognitive psychology will be borrowed as inspiration for the proposed frameworks in this thesis, so pattern classifiers that are based on a similar concept will also be included for completeness.

2.1 Pattern classifiers

Pattern classification in machine learning is a task that is simple in principle. Given a set of training data, which have each been labeled with its corresponding output class label, a system is to learn the patterns involved. When an unlabeled set of data is provided, which may or may not be consisting of the ones used for learning, the system predicts the class label of this set based on the patterns it has learnt. In principle, that does not sound like a difficult task, as young children are able to do the same without being thought. They can differentiate between people, regardless of what the person is wearing; they can recognize a food being something they like or dislike; they can point at a pet and identify it as a cat or dog; they can pick out their own shoes from a rack of similar shoes. But what seems straightforward and intuitive, even trivial to humans, becomes a complicated task when implemented by machines.

While our senses gather different aspects of any object, and our processing can incorporate any other memory and experience we have, pattern classification deals with something that has to be simplified. Any object or sample is represented by its features, represented as a vector of values $x = [x(1), \dots, x(q)]^T$ denoting different attributes, and both learning and classification is carried out based on just these feature values. The two phases are mostly considered as distinct – *learning*, or training, where the system tries to extract knowledge or generalize from the given labeled data; and *classification*, or testing, when the system assigns class labels to unlabeled data. Since any data sample can be represented as a feature vector, it is often thought of as representing points in the input feature space. For example, a data sample

with a feature vector of length q represents a point in the q -dimensional feature space. Learning can then be interpreted as finding the separation or boundary between the training points of different classes within the feature space. Subsequently, classification of new unlabeled points can be done by considering their position in the feature space with respect to the boundary found.

There are different ways to learning and classification, as reflected by the vast choices of classification techniques available. However, the learning and classification phases are not always distinct. This marks the difference between lazy and eager learning techniques (Friedman, Yun et al. 1996).

2.1.1 Lazy learning methods

While it is conceptually straightforward to break down the pattern classification process into learning and classification phases, this is not always necessary. Lazy learning methods are classification techniques which delay the learning to the classification phase. When training data is provided, they are not processed and no induction is performed. The information is retained until an unlabeled sample is presented for classification, when the system refers to the training data to make a decision on the class label prediction. Examples of lazy learning methods include nearest neighbor algorithms and instance based learning. These are also commonly referred to as similarity based learning, as they are based on the logic that similar samples belong to the same class. Another classifier that can be considered a lazy learning algorithm is the Naïve Bayes classifier, which calculates the posterior probability that a sample belongs to either class, based on the features of the sample.

Nearest neighbor algorithms

Nearest neighbor algorithms classify any unlabeled sample by identifying the most similar instance, and assigning the same class label to the query sample. It functions on the philosophy that samples which are similar to each other should belong to the same class. This measure of similarity may be defined differently, but the most common way is to define it as the Euclidean distance of the samples in their input feature space. The features can also be weighted (Aha 1992; Cost and Salzberg 1993; Kohavi, Langley et al. 1997), which assigns different importance to different features, or reduce the number of features for the data so as to reduce the effect brought about by the curse of dimensionality (Bellman 1961).

A natural extension of the nearest neighbor algorithm is to consider not just the single nearest neighbor, but a number of them. The k -nearest neighbor (kNN) (Cover and Hart 1967) defines a neighborhood of k samples close to the unlabeled query sample, and classifies it according to the most frequently occurring class in that neighborhood. By considering more neighbors, the classification is more robust to noise. Variants include distance weighted kNN (Dudani 1976), which assigns greater importance to neighbors that are closer to the query sample.

As classification is carried out by comparing with the most similar samples, there is no separate learning phase required. Learning is delayed to the time when the query sample is presented, and no abstract representation or generalization is made beforehand. If new samples are made available, the subsequent classifications are automatically updated, and there is no need for a

retrain process. The shortcoming is that all the samples have to be stored in order to perform classification, and the classification process also becomes slow and tedious as it requires comparisons with all the available samples.

Instance based learning

Nearest neighbor algorithms can be considered a type of instance based learning (Aha, Kibler et al. 1991; Hastie, Tibshirani et al. 2009). The issues involved with instance based algorithms are well illustrated through IB1, IB2 and IB3 (Aha, Kibler et al. 1991). IB1 is similar to the 1-nearest neighbor algorithm, but each feature is given equal importance by normalizing the range of their values. With a bound on the similarity value contributions of each feature, missing feature values can be handled by assigning them with the lowest similarity. In order to reduce the storage requirements, which is a problem that affects both IB1 and nearest neighbor algorithms, IB2 focuses only on the concept boundary and stores samples that help to describe this boundary. These samples are identified through misclassifications by their neighbors, based on the logic that samples close to the boundary are closest to the other class, and this is where misclassifications occur. As a result, the memory requirements for instance storage are reduced. But misclassifications arise not only from the concept boundary; they also arise from noise. IB2 is thus more sensitive to noise, especially since a large number of such samples are saved. A further extension IB3 helps alleviate this by maintaining a classification record so that the poorly performing samples can be discarded. This is done using a significance test. The noise tolerant extension can be applied to kNN in principle, but will not work well with small datasets.

Like nearest neighbor algorithms, instance based learning faces the curse of dimensionality, as increased number of features make the concept description by samples highly inadequate.

Naïve Bayes classifier

The main idea driving the Naïve Bayes classifier is probability. Suppose a sample x is to be grouped into either of two classes, ω_1 or ω_2 . This decision can be made by comparing the posterior probabilities that x belongs to either class. In other words, we can compute

$$R_{NB} = \frac{P(\omega_1|x)}{P(\omega_2|x)} \quad (2.1)$$

and classify x as belonging to ω_1 if $R_{NB} > 1$, and ω_2 otherwise. The posterior probabilities can be computed using the Bayes formula for calculating probabilities,

$$P(\omega|x)P(x) = P(x|\omega)P(\omega). \quad (2.2)$$

R_{NB} can be written as

$$R_{NB} = \frac{P(x|\omega_1)P(\omega_1)}{P(x|\omega_2)P(\omega_2)} \quad (2.3)$$

where $P(\omega_1)$ and $P(\omega_2)$ are known as the prior probabilities of each class, which can be computed based on the number of samples in the data; and $P(x|\omega_1)$ and $P(x|\omega_2)$ are the likelihood probabilities of the sample x

belonging to each of the classes. These probabilities are computed based on the feature values of $x = [x(1), x(2), \dots, x(q)]^T$.

For each $\omega \in \{\omega_1, \omega_2\}$, $P(x|\omega)$ can be computed using

$$\begin{aligned} P(x|\omega) &= P(x(1), x(2), \dots, x(q)|\omega) \\ &= P(x(1)|\omega)P(x(2)|\omega) \dots P(x(q)|\omega) \\ &= \prod_{i=1}^q P(x(i)|\omega), \end{aligned} \tag{2.4}$$

where each $P(x(i)|\omega)$ can be estimated based on the relative frequencies from the training data. The assumption made here is that all the features are independent given the class. It is clear that such an assumption is rarely true in reality, hence the name of Naïve Bayes classifier. However, the estimation of actual conditional probabilities is difficult. The classifier also works only on discrete feature values. Continuous valued features can be discretized by binning. Another concern with the use of this method is that the multiplication of conditional probabilities in the computation of $P(x|\omega)$ will be affected by zero values. It is possible that certain feature values simply never occur with a particular class, resulting in a zero conditional probability which will wipe out the entire multiplication result. However, this can be overcome by small modifications such as using the Laplace estimator, which ensures that the probabilities can only be very small but never zero.

Although the naïve conditional independence assumption does not hold, the classifier can still perform well (Domingos and Pazzani 1997) and is often used as a benchmark for comparisons (Zhang 2004). Furthermore, it is easy to use,

does not require domain knowledge, and requires only a short computational time (Kotsiantis, Zaharakis et al. 2006).

The above formulation for the classifier is done using two classes, but it is easy to extend the classifier to a larger number of classes. Rather than computing R_{NB} , the posterior probabilities of the sample x belonging to each class can be computed separately beforehand and compared to determine which class it belongs to.

2.1.2 Eager learning methods

Unlike lazy learning methods, eager learning methods carry out learning or induction from the training data, to achieve an abstraction that can be applied to unlabeled query samples. As learning has been carried out, it is possible to discard the training data and store only the abstraction, such as trained networks or decision rules. Even though a dedicated learning phase is necessary, classification becomes faster as the generalization has already been performed offline. Most classifiers fall under this category, and we present some of the most commonly used classifiers.

Artificial neural network

The artificial neural network (ANN) is a simplified model imitating the structure and function of the biological neural network. The connectionist network (Rumelhart, McClelland et al. 1986) consists of basic processing units known as neurons. The structure usually consists of three layers – the input layer, output layer, and one in between known as the hidden layer. Each layer consists of a number of neurons which are connected to neurons from other

layers as shown in Figure 2.1. Training samples are supplied to the input layer and the result is obtained at the output layer after computation.

Figure 2.1 Basic architecture for ANN

Each neuron has input coming in and computes the output to the next layer. The input is multiplied by the corresponding weights and summed up, sometimes with an additional bias term as well. The weighted sum is then passed through some activation function to compute the output for use in the next layer. There are various activation functions that can be used, such as the linear or log-sigmoid functions.

Learning in the neural network is the adaptation of the weights connecting the neurons. As training samples are presented to the network, the weights are updated such that eventually the given inputs will give the desired outputs. In terms of classification, it means the output from the neural network given the training samples should be its correct class. A well known method for carrying

out the weights update is the back propagation algorithm (Rumelhart, Hinton et al. 1986), which is used in the feedforward network multilayer perceptron (MLP). It computes the error based on the current output and the desired output, and makes use of this error to compute the change in the relevant weights, subject to a factor called the learning rate. This learning rate controls how much of the error is being used for correction, and will determine how fast the weights get updated.

ANNs are a large class of networks and there are many variations. One example is the radial basis function (RBF) networks (Broomhead and Lowe 1988). These networks use radial activation functions at each neuron of the hidden layer. Another well-known variant is extreme learning machines (ELM) (Huang, Zhu et al. 2004), which removes the need for iterative computation of parameters and network training.

On the whole, ANNs can perform well since they are able to handle non-linear mappings. However, training time is lengthy and it is difficult to select the number of neurons, hidden layers (should one decide to use more than one hidden layer) as well as the activation functions. Another drawback is the lack of an interpretation for the way classification is carried out. There is no intuitive or human understandable way of explaining how the network arrived at its conclusion, thus making it difficult for humans to use as decision support.

Support Vector Machines

Support vector machines (Cortes and Vapnik 1995) learn by maximizing the separation between classes. Sample points belong to either of two classes $\omega_1 = +1$ or $\omega_2 = -1$ and they can be visualized as points in the feature space.

If the points of the two classes are linearly separable, we can find a line that separates these two classes such that the margin of separation between them is the largest. In higher dimensional space, this line corresponds to a hyperplane and can be represented as $w^T x - b = 0$ where w is the normal vector to the hyperplane. This hyperplane can be shifted in parallel while still separating the two classes, until the positions where it will intersect with samples from either class. The hyperplanes at these positions are the supporting hyperplanes, and the intersecting samples are the support vectors. The margin is the perpendicular distance between these two supporting hyperplanes and is given by $2/\|w\|$.

Maximization of this margin is an optimization problem, subject to the constraint that the support vectors from each class should be on different sides of the respective supporting hyperplanes. Points from the positive class or negative class should respectively satisfy the following inequalities:

$$w \cdot x_j - b \geq +1 \quad (2.5)$$

$$w \cdot x_j - b \leq -1. \quad (2.6)$$

These constraints can be combined as

$$\omega_j(w \cdot x_j - b) \geq 1. \quad (2.7)$$

Since the maximization of the margin can be rewritten as a minimization and the constraints are given, the goal is hence to solve a constrained optimization problem:

$$\begin{aligned}
& \text{Minimize } \|w\| \\
& \text{Subject to } \omega_j(w \cdot x_j - b) \geq 1.
\end{aligned} \tag{2.8}$$

This problem can be reformulated with Lagrange multipliers and Karush-Kuhn Tucker conditions. The optimization problem can then be solved numerically using methods such as quadratic programming.

The above example assumes that the points from the two classes are linearly separable, but this is usually not the case. This can be overcome by using the kernel trick (Boser, Guyon et al. 1992). The dot product operation can be replaced with a kernel function, which effectively projects all the points to a higher dimensional feature space, in which the points may be separable. In case of serious overlap between the two classes, a soft margin can be used, where a small number of points from one class are permitted to fall on the other side of the separating hyperplane.

SVM has reported overall high accuracy in performance, but is dependent on the cost parameter and appropriate kernel function. Commonly used kernel functions are the polynomial, radial basis and sigmoid functions:

$$K_{polynomial}(x_1, x_2) = (x_1^T x_2)^q \tag{2.9}$$

$$K_{radial\ basis}(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2), \gamma > 0 \tag{2.10}$$

$$K_{sigmoid}(x_1, x_2) = \tanh(kx_1^T x_2 + c), k > 0, c < 0. \tag{2.11}$$

The algorithm is also designed for binary classification, so data with more classes will have to be formulated as a series of two-class data.

Decision trees

Decision trees classification is based on the construction of decision trees which partitions the input space with rules. These are usually univariate rules, which means they split the data based on one feature at a time. The classification tree consists of nodes and leaves, where each node represents a split in the data according to a particular feature, and each leaf represents the class that a sample should be assigned to. Classification is performed by traversing the tree from the root node, according to the feature values of that sample, until it arrives at a leaf node which determines its classification. Learning involves building a classification tree from the training data.

The learning procedure is essentially the construction of a classification tree from the training data. An outline is provided in Figure 2.2. Given the training data, the feature with the best splitting measure is identified and the data is split into partitions depending on the feature values. This forms a branch with data partitions on each split. Partitions with only samples from one class will each form a leaf; otherwise the function is called recursively on the data partition. The key here lies in the splitting measure. Different heuristics can be used to determine a suitable feature to split the tree, and there is no single best way of determining the splitting feature.

The stopping criterion is also important in tree building. Some criteria include restrictions on the minimum node size, as well as a threshold on the impurity, which imposes a minimum quality on the splitting criterion value. This helps to control the growth of the tree such that it does not overfit the training data used to build it. Pruning methods can also be used to reduce the

size of the tree, by setting aside a validation or pruning set out of the training data.

Tree construction

- Find the feature with best splitting measure
 - For every (categorical) feature value or each partition of the (numerical) feature values
 - Add a branch
 - If all the samples belong to the same class, create leaf node
 - Else, call the function on this sub-tree

Figure 2.2 Outline for construction of decision tree

Two well-known algorithms include the classification and regression trees (CART) and C4.5, which are described below.

CART is a popular tree induction algorithm (Breiman 1984). The tree is constructed by splitting the data according to the most discriminative feature. For each of the subsets of split data, the process is carried out recursively, with each split corresponding to a node of the tree. The process stops when all the data in a split belongs to one class or some other stopping criteria (such as tree size restriction or growth in predictive accuracy) is fulfilled. The class of each of these final subsets is represented as leaves of the tree. The key to tree construction is how to determine the most discriminative feature, and the value at which the split should be conducted. This is done by considering the Gini impurity. For each feature, the Gini impurity can be computed for various splits, and the split resulting and subsets with the smallest Gini impurity is obtained.

This is done for all the features, and the feature with the lowest Gini impurity for its best split is the most discriminative feature used for the current node. The Gini impurity of any set S is given by

$$I_{Gini}(S) = 1 - \sum_{j=1,2} [P(\omega_j|S)]^2 \quad (2.12)$$

where ω_j are the possible classes. The split that gives the lowest Gini impurity of each partition is selected.

The C4.5 is another popular tree construction algorithm (Quinlan 1993) which operates like CART, but with a different method for finding the most discriminative feature and the corresponding split value. The entropy is calculated for all the sets resulting from various splits to compute the gain ratio which normalizes the information gain to prevent bias towards tests with many outcomes. The split maximizing the gain ratio is selected for the split and this test is carried out recursively on each of the subsets. The gain ratio is given by

$$Gain\ ratio = \frac{I(S) - \sum_{i=1,2} \frac{|S_i|}{|S|} I(S_i)}{- \sum_{i=1,2} \frac{|S_i|}{|S|} \log \left(\frac{|S_i|}{|S|} \right)}, \quad (2.13)$$

where

$$I(S) = - \sum_{j=1,2} P(\omega_j|S) \log[P(\omega_j|S)] \quad (2.14)$$

and $|S|$ is the size of set S .

Decision trees are easy to comprehend, and learning only involves one pass through the training data. They can handle categorical attributes and missing feature values, and also give some indications as to which features are likely to be more important in that data. However, the overall partition of the input space can only be in the form of hyper-rectangles, which are unable to fit a lot of complex datasets. Determining the best split is also computationally expensive as it needs to sort and compute for every feature. Most importantly, the decision trees constructed are highly sensitive to the data. A slight change in the training data set will lead to a very different decision tree.

The decision tree classifier can be rewritten as a list of IF-THEN rules, which can also be extracted through other rule-based classifiers. Rather than a top-down approach of splitting the input space, a sequential covering algorithms use a bottom-up approach. One well-known example is the PRISM algorithm (Cendrowska 1987). It generates rules by targeting one class at a time. Rules are generated through a depth-first search, finding attribute-value pairs that best describe that class. Subsequent attribute-value pairs are found which further refine the rule to describe that class, and these pairs are appended to the antecedent of the rule, until the rule can perfectly cover only samples of that class. The samples are then removed, and rules are constructed recursively until all the samples in the class have been removed. The process is then repeated for remaining classes.

Other well-known algorithms for generating rules include RIPPER (Cohen 1995), which is built on the IREP algorithm (Fürnkranz and Widmer 1994), the CN2 algorithm (Clark and Niblett 1989) and AQ algorithm (Michalski 1969).

Fuzzy classification systems

Fuzzy sets were introduced (Zadeh 1965) to model the notion of graded concepts. This is in contrast to the use of thresholds in crisp concepts, which was sometimes unsuitable. If a tall person was defined as someone over 175cm in height, under crisp concepts, someone of height 174cm would not be tall. This counter intuitive understanding of concept is addressed using fuzzy sets, which allowed different degrees of membership to a concept. Concepts are represented using membership functions, which could be triangular, Gaussian, trapezoidal, generalized bell shaped, polynomial, sigmoid, etc.

Classification is facilitated using fuzzy rules, such as “*If rain is heavy and wind is strong, stay indoors.*” The fuzzy rule base may be crafted by experts, or generated from the learning data (Kosko 1992; Wang and Mendel 1992). Each feature (i.e. *rain*, *wind*) is first evaluated based on its membership to a concept (i.e. *heavy*, *strong*). The process of computing the degree of membership to each fuzzy set is known as fuzzification.

Fuzzy rule antecedents could involve more than one feature, in which case they will be combined together using fuzzy operators. The fuzzy maximum, minimum and not operators correspond to the set operators intersection, union and complement. The fuzzy rule consequent also consists of a concept represented by a membership function. Implication is performed to obtain the truncated output function for each rule. Two well-known implication techniques are the minimum (Mamdani 1977) and the product (Larsen 1980) rules.

For any given input, more than one rule will usually be fired in parallel. The truncated output fuzzy sets for all the fired rules are aggregated to provide a

combined output fuzzy set. The result is then defuzzified to obtain a crisp value. Defuzzification techniques include centroid, maximum, mean of maxima, height, minimum, modified height methods.

Fuzzy systems can deal with imprecision and uncertainty, and have been successfully applied in many household electrical appliances. They are also able to capture human logic and represent relationships in linguistic terms. The fuzzy rule base can be crafted by humans to encode expert knowledge, and can also be validated and checked. However, it is non-trivial to construct the fuzzy rule base for a more complicated problem, especially when a large number of dimensions are involved.

2.2 Classification issues arising from machine learning

Pattern classifiers are essentially an attempt to imitate the human ability to learn and apply this knowledge to new cases. The described techniques represent only a small proportion of the vast variety of methods and variants available. But despite the potential of pattern classifiers to deal with bigger and more complex problems than humans can, there are also some issues which reflect certain limitations of machines.

2.2.1 Comprehensibility of classification

As previously shown, pattern classifiers can learn by optimizing certain measures, such as predictive error, class separation, similarity, or purity of sets. However, these trained classifiers are mostly black-boxes with little insight into how they arrive at classification output labels. Some methods are more interpretable than others, such as decision trees and rule-based classifiers that

can present their knowledge in intuitive IF-THEN rules. But many studies indicate that black-box models have higher accuracy than comprehensible models (Huysmans and Baesens 2006). Indeed, there is a tradeoff between comprehensibility and accuracy (Van de Merckt and Decaestecker 1995). Simple models are easier to understand and explain, making them more interpretable. But using simple models to describe complex concepts is like using a linear classifier to separate non-linear data, which is simply inadequate and thus results in poor classification accuracy.

Due to the lack of interpretability of many black-box classifiers, it becomes difficult to apply them widely outside research environments (Huysmans and Baesens 2006). Some applications do not allow the use of black-box classifiers by regulation, because there is no way to understand the logic behind the classification. One example is in medical diagnosis where the decision affects patient treatment (Fung, Sandilya et al. 2005). Other applications include safety critical areas such as airline or air traffic control, power stations, medical and health informatics and security related domains (Andrews, Diederich et al. 1995; Xing, Pei et al. 2011). In these areas, the classification should only support the decision making process by humans, so comprehensibility is important.

Comprehensibility of classifiers does not only improve user receptiveness. In reference (Allen and Brooks 1991), the importance of interpretability of classifiers is described through an analysis of the benefits of extracting rules from neural networks. An understanding of the classification process enables the user to determine whether the classifier is suitable for a task, and whether its results can be used and applied for certain applications. It also helps in verification of whether the classifier satisfies the requirements of the larger

system in which it will be integrated. Through an analysis of the classification process, generalization can be improved as the user can better identify weaknesses or poorly represented regions. Furthermore, an understanding of how the classifier makes its prediction contributes to knowledge acquisition and data exploration.

With the tradeoff between comprehensibility and accuracy, many works strive to find a balance between the two ends of the spectrum. One approach (Van de Merckt and Decaestecker 1995) is to make a distinction between knowing and explaining a concept. Knowing a concept involves being able to classify accurately, while explaining a concept involves providing comprehensible rules that provide insight into the workings of the classifier. This is implemented through a hybrid neural-symbolic system called GEM, consisting of a recognition function using a neural network, and a communication function using nearest neighbor rule on the prototypes. This approach can also be viewed as an oracle versus a mimetic model that mimics it (Blanco-Vega, Hernández-Orallo et al. 2004).

Another approach to dealing with the comprehensibility issue is through rule extraction. As simple interpretable classifiers are unable to classify accurately, black-box models are used to achieve the desired accuracy before rule extraction methods are applied to those models to improve their comprehensibility. Some of these methods are dependent on the classifier used, while others can be applied to any black-box classifier.

Rule extraction techniques for neural networks include the subset algorithm (Towell and Shavlik 1993), RuleNet (McMillan, Mozer et al. 1991), NofM

method (Towell and Shavlik 1993), validity interval analysis (Thrun 1993) and rule-extraction-as-learning (Craven and Shavlik 1994). These methods construct rules from the weights and biases of the links between neurons. There has also been increased interest in the rule extraction techniques for SVMs, such as RuleExSVM (Fu, Ong et al. 2004), eclectic rule-extraction (Barakat and Diederich 2006), ExtractRules-Vm and ExtractRules-PCM (Fung, Sandilya et al. 2005), FREx (Chaves, Vellasco et al. 2005), hyper-rectangle rule extraction (Zhang, Su et al. 2005), SVM with prototype (Núñez, Angulo et al. 2006) and SQReX-SVM (Barakat and Bradley 2007). A number of classifier-independent rule extraction techniques are also available, such as BIO-RE (Taha and Ghosh 1999), G-REX (Konig, Johansson et al. 2007) and TREPAN (Craven and Shavlik 1995). However, these rule extraction techniques do not always agree with the underlying models that they are trying to describe. Although they provide insight into the classifier, using only the extracted rules for classification generally do not yield the same level of accuracy.

2.2.2 Learning and classifying in a dynamic environment

A young child with little knowledge of his surroundings learns from his encounters. Over time, he builds up concepts, making guesses on completely new encounters and correcting himself. This process of knowledge building and refinement never stops as he grows and develops. On the other hand, learning in machines is not quite as natural and simple. Many classifiers have a separate learning and classification phase, rather than an ongoing learning process, and do not easily incorporate new information as they become available. A new retraining process is needed, and access to all the previously used training

samples is required due to catastrophic interference. New samples are also presented for classification, and they may reflect changes in the environment. This could be in the form of perturbed measurements due to surrounding external factors, or a change in the concept itself. When the testing samples reflect a change, the classifier is no longer adequately equipped, because the basic assumption in pattern classification is that the training data can adequately represent the concept. While this is a fair assumption for pattern classifiers, it is a stark contrast to the ability of humans to extend their knowledge from limited experience into very different domains and problems.

Incremental learning

Some classifiers do not have separate learning and classification processes. Lazy learning methods delay the learning to the time when the query is presented. However, they require access to all the training data. This is not feasible in actual applications, especially in a dynamic environment where samples are continuously made available, thus placing a strain on memory requirements. Incremental learning techniques are useful in such situations. These are classifiers that can be updated with new samples without revisiting the previous training data or undergoing retraining.

The most well-known classifier with incremental learning ability is fuzzy ARTMAP (Carpenter, Grossberg et al. 1992). This technique is based on generating hyperboxes to represent the training data, then mapping each hyperbox to its corresponding class. Hyperboxes are generated from training data, allowing them to grow and enclose as many as possible until they hit a maximum size. As new samples are presented, they may already be represented

by existing hyperboxes, otherwise new ones may be generated. This facilitates the incremental learning aspect. Classification is carried out according to the smallest hyperbox that encloses the query sample or the nearest one if it is uncovered. Fuzzy ARTMAP has numerous variants, but there are also other classifiers that are capable of incremental learning. The nearest generalized exemplar (Salzberg 1991) also uses hyperboxes, but takes into consideration the number of correct classifications made by each hyperbox. Growing neural gas (Fritzke 1995) uses a topology of nodes and connections to map the training data, updating the connections and allowing insertion or deletion of nodes. The generalized fuzzy min-max neural network (Gabrys and Bargiela 2000) also uses hyperboxes, where its end points are represented by the connections between the input and hidden layer. Incremental learning via function decomposition (Bouchachia 2006) employs a neural network, where the hidden layer represents prototypes that approximate each class. Training samples are mapped to prototypes via a clustering function, and the prototypes are mapped to the class via a labeling function.

Besides incremental learning classifiers, there are also incremental versions of other classifiers, such as incremental SVM (Syed, Liu et al. 1999a), incremental induction of decision trees (Utgoff 1989), incremental RBF network with long-term memory (Okamoto, Ozawa et al. 2003; Ozawa, Tabuchi et al. 2010) and Learn++ (Polikar, Upda et al. 2001) which generates an ensemble of neural network classifiers.

Concept drift

While incremental learning techniques allow the classifier to continuously learn new information that reflects the change in the environment, some changes may render the previous knowledge outdated. In concept drift, the concept changes with time so a classifier that is not updated will quickly deteriorate in performance. New data representing the concept in its current state must be incorporated into the classifier quickly to update it. It has been addressed in surveys in literature (Tsymbol 2004; Žliobaite 2009).

Concept drift can be categorized into three types. Gradual change is the most common type, and can be seen in fashion trends or customer preferences in business sales prediction models (Klinkenberg 2003). Such changes are usually handled using windowing techniques (Widmer and Kubat 1996; Salganicoff 1997; Kuncheva and Zliobaite 2009). The incoming stream of data is separated into chunks or windows, which facilitates the updating of the classifier with the current window of data. Old data is sometimes forgotten using outdating (Lughofer and Angelov 2011) or forgetting techniques (Pavlidis, Tasoulis et al. 2011). In ensemble methods (Blum 1997; Wang, Fan et al. 2003; Kolter and Maloof 2007), the base classifiers can be evaluated based on the current window of data to determine if they remain relevant to the concept, and how to combine their results in the classification. Instance selection or weighting can also be used (Klinkenberg 2004), and such methods have been applied in areas like information filtering (Allan 1996) and web page recommendations (Balabanovic 1997). Each sample can then be evaluated in terms of their relevance to the current concept. Unlike gradual change, abrupt changes occur more suddenly and are sometimes referred to as concept shift.

For example, an email topic is no longer of importance once the deadline it discusses has passed. Such changes can be identified through change detection techniques, such as cumulated sum CUSUM (Page 1954), geometric moving average (Roberts 1959), Hoeffding trees (Hulten, Spencer et al. 2001), drift detection method (Gama, Medas et al. 2004) and non-parametric test (Kifer, Ben-David et al. 2004), among others. These methods detect when a change has occurred and updates are necessary. The third commonly considered type of concept drift is that of recurring changes. This includes cyclical changes such as weather prediction (Tsymbol 2004), but also covers problems that simply consider possibility of old information becoming useful again at a later time. Methods like (Salganicoff 1997) retain previous samples, and ensemble methods like (Karnick, Muhlbaier et al. 2008) retain previous classifiers whose weights are later increased when deemed to be relevant to the concept again.

The goal of concept drift techniques is to update the classifier whenever the change in the environment renders it outdated, so that deterioration in performance can be quickly rectified. However, most methods do not consider the classification of samples representing the changed concept before the classifier can be updated. Humans, on the other hand, easily extend their knowledge to new situations, such as in drawing analogies, solving isomorphic problems and transfer of learning.

2.2.3 Bias from imbalanced classes

Concepts are difficult to capture and represented using classifiers not just because of change. Some concepts are simply poorly represented by the limited training data. While classifiers may not have been constructed if there were too

few training samples, data with adequate samples from one class but very few from another class is a very common scenario. Such data suffer from imbalanced classes. Concepts with such data are easily handled by humans, as we can easily remember exception cases. However, pattern classifiers are often affected by the bias resulting from a much larger majority class. This can be illustrated in error-driven methods like neural networks. Errors arising from the minority class are fewer due to their smaller number, leading to heavier influence from the errors arising from the majority class. This causes the classifier to be biased towards correctly classifying the majority class and neglecting the minority class. The same situation arises in most other classifiers, whether they are maximizing class separation, set purity or posterior probability. Yet the errors of the minority class are no less important, even if they are fewer in number.

There are techniques designed to overcome this bias caused by imbalanced classes (Japkowicz and Stephen 2002; Kotsiantis, Kanellopoulos et al. 2006). The data can be made more balanced through sampling techniques. Undersampling the majority class reduces its size, but risks discarding potentially useful information. A more discerning method can be used, such as Wilson's editing (Wilson 1972) or the Neighborhood Cleaning Rule (Laurikkala 2001), which removes noisy samples from the majority class. One-sided selection (Kubat and Matwin 1997) removes not only noisy samples, but also those that are redundant or unreliable. As kNN is used for identifying the noisy samples, high computational time is required and these techniques are not suitable on large datasets (Guo, Yin et al. 2008). Oversampling, on the other hand, increases the number of minority class samples. Random oversampling

by presenting duplicated data may result in overfitting, so different heuristics may be used. Synthetic minority oversampling technique (SMOTE) (Chawla, Bowyer et al. 2002) is a well known technique which creates new samples that fall approximately between a minority class sample and its nearest neighbors. One variant is Borderline-SMOTE (Han, Wang et al. 2005). The parameters determining the final proportion of minority to majority class samples are user-determined, but some works also look into the optimal proportion for learning (Weiss and Provost 2001). Another variant of SMOTE creates new samples according to the estimated probability distribution of the minority class (Gao, Hong et al. 2012), such that they populate the region of that class instead of increasing overlap between the two classes.

Rather than balancing the data distribution explicitly, algorithm level methods can also be used. Cost-sensitive learning (Domingos 1999; Elkan 2001) can rebalance the effect of the class imbalance according to the application. Adacost (Fan, Stolfo et al. 1999) carries out boosting on the misclassified samples with weights adjusted with a cost function, while SMOTEBoost (Chawla, Lazarevic et al. 2003) combines both SMOTE with boosting techniques. Ensembles can also be used, such as (Chan and Stolfo 2001), where the imbalanced data is redistributed into subsets that are balanced. Samples of the majority class are used in any one subset, while minority class samples are replicated across the sets. The subsets are then used for training base classifiers.

2.3 Evaluation of classifier performance

Evaluating the performance of classifiers allows comparisons between different techniques, and determining the suitability of a classifier to any given

problem. Evaluating classification inevitably involves a comparison of the actual and predicted class label, but also involves other considerations. A review is available in (Baldi, Brunak et al. 2000).

2.3.1 Data for evaluation

Data that is available can be used entirely for training a classifier, so that it can be provided with as much information as possible. However, evaluating the classification on the same data it learnt from does not give a fair judgment on how well the classifier would perform in actual applications, when presented with unseen data. Such evaluation obtained through resubstitution tends to inflate the performance, as the classifier was trained specifically for that dataset. Thus, datasets are often separated into two randomly ordered subsets – *training* and *testing* data. The training data is presented to the classifier for learning, while the trained classifier will be evaluated on the separate testing set. This holdout testing set provides a better picture of the generalizing capability of the classifier. One problem with such holdout sets is that part of the data is wasted and cannot be used for training. This can be overcome by using k-fold cross validation. The available dataset is randomly ordered and separated into k equal partitions. Training and testing is performed in k rounds. The k-th round sets aside the k-th partition for testing while the remaining partitions are used for training. Results are averaged over the k rounds, which better summarizes the classifier's performance given that dataset. Setting k to the number of samples in the dataset will be equivalent to the leave-one-out method. The k-fold cross validation method allows evaluation of the classifier, while training can eventually be performed using all the available data.

2.3.2 Evaluation metrics

The most straightforward metric is the accuracy, given by

$$accuracy = \frac{\text{no. of correct classifications}}{\text{total no. of classifications}}. \quad (2.15)$$

This is sometimes represented as the error instead, where

$$error = \frac{\text{no. of misclassifications}}{\text{total no. of classifications}}. \quad (2.16)$$

However, for datasets with imbalanced classes, this measure is no longer informative. For example, on a dataset with only 1% from the minority class, a classifier that predicts all the samples as the majority class will still give a 99% accuracy. This seemingly good performance is actually of no significance, because the classifier has failed to identify a single sample from the minority class. Instead, other measures will need to be considered.

Table 2.1 Contingency table

		Predicted class	
		Positive (minority)	Negative (majority)
Actual class	Positive (minority)	TP	FN
	Negative (majority)	FP	TN

Considering the data to contain samples from 2 classes (majority in negative class and minority in positive class), the following values can be computed according to Table 2.1: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN).

The contingency matrix can be formulated and additional measures are computed from this table. Sensitivity given by

$$sensitivity = \frac{TP}{TP + FN} \quad (2.17)$$

is also known as the true positive rate or recall, and measures the proportion of positive class samples that have been correctly identified. Specificity given by

$$specificity = \frac{TN}{TN + FP} \quad (2.18)$$

is also known as the true negative rate and measures the proportion of negative class samples that have been correctly identified. While the accuracy does not provide a single measure to fairly reflect the classifier performance on imbalanced datasets, the Matthews correlation coefficient (MCC) (Matthews 1975) provides a more balanced measure, without knowing the costs associated with misclassification of each class. MCC is given by

$$MCC = \frac{(TP)(TN) - (FP)(FN)}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \quad (2.19)$$

and its value lies between -1 and 1. An MCC value of 1 denotes perfect agreement between the prediction and actual class label, while -1 shows complete disagreement. Other common evaluations of imbalanced datasets include using the receiver operating characteristic (ROC) curve and finding the area under the ROC curve (AUC). However, these require reference to a decision function value v . Samples whose decision function value exceeds a threshold t_v will be classified as positive and those below will be classified as negative. The highest value of t_v will ensure all the negative samples are identified but all the positive samples misclassified, while the lowest value of t_v correctly identifies all positive samples and none of the negative samples. At some point in between the two extreme values, the balance between the two classes will be optimal. An ROC curve can be obtained by plotting the sensitivity against (1-specificity). The optimal balance corresponds to the point on the curve closest to the top left hand corner. AUC finds the area under this curve, and a curve that provides larger AUC can attain the optimal point closer to the top left hand corner, thus a classifier with larger AUC is considered to be of better performance.

2.4 Cognitive psychology

With respect to pattern classification, some concepts in cognitive psychology are particularly relevant. This section will elaborate on categorization models, the structure of memory and the learning process as inferred from cognitive experiments.

2.4.1 Categorization models

Pattern classification is modeled after the human ability to learn and apply the knowledge to new instances. Specifically, it is closely related to the categorization process in which information is retrieved and utilized. There are two widely accepted categorization models through which humans perform the process – rule and exemplar models.

Categorization by rules is performed by comparing certain attributes of the instances with rules. These rules may be provided and communicated to the subject, or abstracted by the subject based on experience and encountered instances. Experiments have shown that human subjects make use of rules during categorization (Trabasso and Bower 1968; Shiffrin and Schneider 1977; Nosofsky, Clark et al. 1989; Rips 1989; Allen and Brooks 1991). This categorization model can explain the speed of categorization (Erickson and Kruschke 2002), as there is no need to remember all the encountered instances.

Categorization by exemplars is similarity based, comparing the query instance with those which have already been encountered before. Exemplars here refer to the individual instances themselves. They have been shown to be in use and are also able to explain a number of observations in cognitive experiments, such as typicality effect and high frequency exceptions (Medin and Schaffer 1978; Hintzman 1986; Nosofsky 1986; Kruschke 1992; Denton, Kruschke et al. 2008). The strengths and weaknesses of the exemplar categorization model complement that of the rule model. As all the instances need to be remembered, categorization by exemplars is slow and cannot match the speed at which people can perform the task, indicating that exemplars were

not always used alone in all categorization tasks (Nosofsky and Bergert 2007). On the other hand, the exemplar strategy enables ill-defined concepts to be better handled (McKinley and Nosofsky 1995), which would not have been described using only simple straightforward rules that humans use.

Both categorization models have been shown to be in use, and there is still ongoing debate over which strategy is more commonly used and under what circumstances they are called upon (Kruschke and Erickson 1994; Pothos 2005; Rouder and Ratcliff 2006; Wills, Milton et al. 2013). However, it is generally accepted that both strategies are utilized (Pothos 2005; Rouder and Ratcliff 2006) and there are hybrids combining the two that model the categorization process (Nosofsky, Palmeri et al. 1994; Vandierendonck 1995; Erickson and Kruschke 1998).

2.4.2 Structure of memory

The categorization models explain how the classification process is carried out, but learning involves encoding and storage of information from the environment before appropriate retrieval of the right information is carried out. This requires a look into the memory structure, at least as current understanding in cognitive psychology allows.

One major component of memory is long-term memory (LTM) (Atkinson and Shiffrin 1968). It has a large capacity that can be divided into implicit and declarative memory (Cohen and Squire 1980; Sternberg and Mio 2009). Implicit memory is that which cannot be verbalized or communicated with ease, including procedural memory such as knowing how to swim, cycle, or drive. Declarative memory can be communicated more easily, and consists of

semantic and episodic memory (Tulving and Donaldson 1972). Both of these are highly relevant to the categorization process. Semantic memory stores meanings, facts, concepts and knowledge. These can be acquired through direct communication or abstraction from experience. The explicit experiences themselves which have been remembered will make up episodic memory. Each episode contains contextual information that is related to the event.

In contrast to LTM, there is a short term memory store which has been termed working memory (WM) (Baddeley and Hitch 1974). It serves as a temporary storage for new information as well as a platform on which activated LTM information can be loaded and processed. The WM model originally consisted of three components – phonological loop, visuo-spatial sketchpad and central executive. It was later updated with a fourth component – the episodic buffer (Baddeley 2000). The phonological loop holds out auditory information while the visuo-spatial sketchpad holds out visual information from the sensory systems. This information fades quickly without rehearsal. The episodic buffer serves as a limited backup store that may bind the sensory information with LTM information to form episodes. The central executive (CE) is an attentional control system can switch between tasks and control information flow between the components and LTM (Baddeley 2012).

2.4.3 Learning in cognitive psychology

During learning, information from the sensory system binds with information from LTM to form episodes in the episodic buffer. Some of these episodes may be remembered and stored in episodic LTM, while meaning and knowledge can be extracted and stored in semantic LTM. There have been

experiments and investigations into the learning process of students (Conway, Gardiner et al. 1997; Herbert 1999; Herbert and Burt 2003; Herbert and Burt 2004), which found that high performance learning was associated with the transition from episodic representation to conceptual representation. This represents a shift from remembering to knowing, which means episodes of learning are schematized. Consciousness of the actual event fades but the knowledge and familiarity remains. In particular, learning of a new concept mostly requires remembering the episodes, until sufficient learning makes the concept familiar.

2.5 Rule-plus-exemplar pattern classifiers

In this thesis, the rule-plus-exemplar framework is central to the proposed architectures. While our use of this framework was inspired by cognitive categorization models, there are existing pattern classifiers which adopt similar ideas. For completeness, we will cover them in this section and explain the differences, which will be more apparent in later chapters.

Rule and exemplar based classification has been combined in various techniques that consider them as two separate classification means, which is aligned with our concept of the rule-plus-exemplar structure. Stefanowski (2001) uses direct rule induction as a set covering algorithm. The rules generated are in IF-THEN format, and may not cover some samples, which are then classified using exemplars. Surma and Vanhoof (1995) also applies the two classification means on different sets of samples that have been pre-split. The set of exceptions are classified using 1-nearest neighbor, while the set of standard cases are classified using a classifier from a rule induction algorithm. This

splitting is done based on the similarity of the samples, and all the samples need to be compared. These techniques can be considered to be similar to the rule-plus-exemplar framework.

Rules and exemplars have also been combined into a single classification means. Zhang and Michalski (1995) also uses set covering, but the sets are in the form of graded concepts that have been represented by both rules and exemplars. Each set is then mapped to one single class. Domingos (1996) treats exemplars as specific rules, and generalizes them to create rules. Also relevant is the SVM-KNN technique (Zhang, Berg et al. 2006), since we consider SVM to be a high level rule, and KNN is an exemplar based technique. However, the rule is being generated on the nearest exemplars to produce a single classification means. These techniques thus do not fall into the rule-plus-exemplar framework.

Although some of the described techniques are also of the rule-plus-exemplar framework, they are different from our proposed frameworks. They are not designed to work in dynamic environments. Even for Chapters 3 and 4, which can be considered to be static environments, these techniques differ from ours. We adopt a wider concept of rules, allowing not just interpretable IF-THEN rules, but also high level rules in the form of trained classifiers. Exemplars are used for classification not just outside the region that can be covered by rules, but also within the covered region. The differences will be more apparent with the details in the following chapters.

Chapter Three

A rule-plus-exemplar framework for overcoming comprehensibility issues

This chapter addresses the problem of comprehensibility of pattern classifiers through a rule-plus-exemplar framework based on cognitive psychology. We first explain the difficulty of this problem and the possible ways to overcome it – namely improving comprehensibility of black-box classifiers or improving generalizability of comprehensible classifiers. A rule-plus-exemplar framework is then used to improve the generalization accuracy while retaining comprehensibility of classification. Experimental results show that the system is better able to generalize and hence give better classification accuracy. Part of the work in this chapter has been published in (Sit and Mao 2012b).

3.1 Difficulty of improving comprehensibility of classifiers

Most trained pattern classifiers function as black-box models, taking in the samples as input and providing the predicted class label as output, without giving insight into how the classification was performed. This makes it difficult for users to understand the process, which limits the applications in which they can be used.

Some classifiers are more comprehensible and intuitive to humans. Decision trees are easily understood by humans as they can be expressed in the form of explicit IF-THEN rules. However, they underperform in terms of classification accuracy as compared to black-box models. Fuzzy classification systems are closer to human linguistics and allow natural graded concepts, but they do not necessarily imply comprehensibility. For the purpose of comprehending classification, the conclusion of any predicted class is not easily processed by humans. The fuzzification, implication, aggregation and defuzzification processes are more for engineering purposes than human interpretation, and the firing of multiple rules for any given input also compounds the understanding of the classification process. Furthermore, the comprehensibility of fuzzy systems is directly affected by the complexity of the system. Mencar and Fanelli (2008) explain how a number of interpretability constraints have to be imposed on a fuzzy system in order to achieve interpretability.

Unlike the simple rules used in human cognition, classification rules created through machine learning can be more complex. They can have more complicated antecedents and involve a larger number of dimensions. However, there is a limit to how complex the rules can be without compromising

comprehensibility. High level rules such as an SVM evaluation function does not provide an intuitive meaning that allows users to easily understand it. Given such constraints, it is difficult for simple rules to adequately describe complicated concepts. Black-box models can perform better for such concepts, but in return it requires higher complexity. This is where the tradeoff between comprehensibility and accuracy arises. Even comprehensible classifiers like decision trees and fuzzy rules are subjected to this tradeoff.

The tradeoff between comprehensibility and complexity can be illustrated with an example. Figure 3.1(a) shows a square-in-square problem while Figure 3.1(b) shows the circle-in-square problem. Both datasets consist of two classes. The circle-in-square problem has a more complex decision boundary between the classes, which in turn requires a more complex pattern classifier. For comparable classification accuracy, Figure 3.1(a) requires a classification tree with only 9 rules while Figure 3.1(b) requires up to 167 rules. This clearly demonstrates the difficulty of creating comprehensible classifiers for problems with complex decision boundaries.

Figure 3.1 Simple vs complex boundaries

A common approach to deal with this tradeoff is to improve the comprehensibility of black-box classifiers. Many rule extraction techniques are available, as shown in Chapter 2. A large number are dedicated to extraction from neural networks, and increasing numbers are specifically tailored to SVMs. But most of these methods are not easily extended to other black-box classifiers. Some classifier-independent techniques are also available. The extracted rules are sometimes presented as hyper-rectangles, which can also be expressed as IF-THEN rules that are more comprehensible. Despite the number of proposed methods, there is limited availability of the code for implementing them (Huysmans and Baesens 2006). This not only makes it difficult to adopt such techniques, but also hinders their objective evaluation. The performance of the extracted rules has been reported in some work. In addition to extracted rule accuracy on the test dataset, they are also evaluated on the fidelity, the coverage of the training samples by the extracted rules, the number of rules required, or their comprehensibility. Fidelity of the extracted rules refers to the consistency between the rules and the underlying classifiers, which do not necessarily produce the same classification result as the rules can only be an approximation. This is because rule extraction techniques have an additional level of separation, which is the underlying classifier, between themselves and the test samples. Even though the rules provide better comprehensibility, they can never fully describe the classifier, and their evaluation can only be made through these various measures, on the limited dataset that is available.

3.2 Improving generalizability of comprehensible classifiers

Instead of improving the comprehensibility of black-box classifiers, we can consider improving the generalizability of the comprehensible classifiers such as decision tree classifiers. Although techniques such as random forests, bagging and bootstrapping can improve the classification accuracy, they hurt the comprehensibility of decision trees. Thus the efforts in improving the generalization accuracy of the classifiers must not complicate the classifiers to the extent that they are no longer comprehensible.

While comprehensible classifiers are most frequently represented in IF-THEN rules, the categorization models in cognitive psychology have shown that humans can understand and also frequently use exemplars to perform classification tasks. As this means of categorization is natural and intuitive, it can be incorporated into the classification process while retaining comprehensibility. In addition, the exemplar model complements the rule model in many ways within cognitive psychology, as elaborated in Section 2.4.1, and such properties may be translated to improvements in the generalization capacity of rule type classifiers to achieve better accuracy.

In cognitive psychology, rule categorization is preferred for its accelerated speed of learning, and the reduced effort as there is no need for recollection of and comparisons against encountered instances. This is in contrast to exemplar categorization. However, exemplars are better at dealing with ill-defined categories which cannot be effectively described using rules. This complementary nature of the categorization models can be utilized when

improving the generalization capability of rule type classifiers. It should be noted that the goal here is not to imitate the cognitive processes, but to draw on relevant ideas. Rules that humans acquire or use are simple and limited in the attributes that are involved, while the rules that are obtained from decision trees can involve more attributes and are larger in number. This is necessary for dealing with the more complicated problems for which classifiers are needed.

We first identify situations where the rules have poor generalization accuracy, based on the comprehensible decision tree classifier. The most straightforward way to identify poor performance regions of rules is to consider their accuracies. After tree construction, resubstitution of the training data can provide the error of each rule. This gives an indication of whether the corresponding region of that rule has poor classification performance. This requires a cut-off point ρ_{err} in the rule error, above which a rule is considered to be of poor performance. The threshold differs between datasets. For ease of use, it can be set to $\rho_{err} = 0.2$. Thresholds that are too low will result in a large number of rules and covered samples to be directed to exemplar categorization.

However, due to the tendency of decision trees to overfit the training data, this resubstitution accuracy is often inflated. One common way to overcome this is to evaluate the rules on a separate validation dataset. Although this can provide a more objective evaluation of the rules, there are other problems that arise. Some dataset are not large enough to afford a separate validation dataset. Furthermore, some of the rules may not cover any of the samples in the validation dataset, so they may not be evaluated. We thus propose an alternative method to determining the region of poor rule performance.

Poor performing rule regions are not confined to high error rules. Due to the way rules are generated, some regions are partitioned into very fine rules covering only a few samples, with high resubstitution accuracy for each rule. Yet the region that has been partitioned is also poorly handled by rules because the rules have overfitted the training data, leading to poor generalizability. The finely partitioned regions can thus be merged together if they have little coverage and are subpartitions of the same region. In higher dimensions, fine partitions generally cover very few samples, thus subpartitions with total coverage of not more than 5 samples can be merged. These merged regions, together with the regions with poor classification resubstitution accuracy, will constitute the ill-defined regions that can be switched to exemplar categorization.

The algorithm for finding these poorly performing rules representing ill-defined regions is shown in Figure 3.2. When rules with total coverage of not more than 5 are merged, the rule error will exceed 0.2, which is in line with the threshold ρ_{err} . The parameter *totalCoverage* is set to 5 since overtraining in decision trees generally lead to partitions that cover not more than 2 or 3 samples. If two rules with total coverage of at most 5 are merged together, the minimum error rate of the resulting merged rule would be 0.2. Setting ρ_{err} to 0.2 will ensure such merged rules are also identified as poor performing.

It can be seen how poor performance of rules is associated with the low coverage rules. In cognitive psychology, one of the major limitations of rules is in dealing with ill-defined categories which cannot be well represented with rules. In pattern classification, even as the decision tree rules are more complex, they may still face difficulty in representing complicated concepts. The critical

region for classification thus lies on the boundary between two classes where the most ambiguity arises. This region cannot be easily characterized using the training region because of a lack of representative boundary points and the dimension parallel cuts produced. However, they are often characterized by a number of more finely partitioned rules with low coverage, which is a result of mapping out the class boundary.

Algorithm for finding poor performing rules**For** each decision tree node,**If** children of current node are leaves ANDtotal coverage of child nodes $\leq totalCoverage$,

Merge nodes and set class label as more frequently occurring class

End if**End for**

Translate leaf nodes to rules

For each rule,**If** error of rule $\geq \rho_{err}$,

Label rule as poor performing

End if**End for**

Figure 3.2 Algorithm for finding poor performing rules

Unlike rule accuracy, rule coverage does not require a separate validation dataset for an objective evaluation in this situation. The importance of the coverage here is to determine whether the rule has been generated from a large

number of samples, or created to cover only a small number of points which might be a case of overfitting the boundary region. It is not aimed at determining whether the rule is likely to cover many test samples. Exemplars can thus be used to support the rules in this region, which is reflected by the low coverage rules. Depending on the size of the dataset, the data distribution and the dimension of the data, there may be different number of samples that are covered by the generated rules. The threshold for low coverage of rules ρ_{cov} can be adjusted for each dataset, or set to the average number of samples covered by each rule.

Since the rules with low coverage may indicate the boundary region between classes, their covered region may contain a large proportion of misclassifications, even if each rule covers only a small portion of the samples. The rules with low coverage will thus be identified. Test samples that fall in these regions can thus be directed to exemplar categorization for support.

3.3 Overall rule-plus-exemplar system

Figure 3.3 Training of comprehensible rule-plus-exemplar system

Figure 3.4 Testing of comprehensible rule-plus-exemplar system

The incorporation of exemplar categorization into rules gives way to a classification system with rule and exemplar subsystems. The exemplar subsystem supplements the rule subsystem to improve the generalizability. As both types of categorization are used in cognitive psychology and are intuitive to humans, they keep the classification comprehensible. While rules that humans frequently infer and use are simple and limited to only a few features, the use of machine learning clearly requires more complex rules. But for the sake of comprehensibility, the rules should still be of the If-Then form that is intuitive to humans so that they can be easily processed. The rule subsystem can be implemented using decision trees such as CART or C4.5. The exemplar subsystem can be implemented through kNN classification or other instance

based lazy learning classifiers. As exemplar classification is used within the regions of low coverage rules, the combination of the two subsystems remains comprehensible. Classification of samples directed to the exemplar subsystem can be done according to an If-Then rule, e.g. *If $x(1) \geq a_1$ and $x(2) < a_2$, Then x is classified according to its nearest neighbors.*

The training and testing procedures are shown in Figure 3.3 and Figure 3.4. During learning, abstraction is carried out in the rule subsystem as decision tree classifiers learn from the training data and generate rules that are stored for later use. The samples are stored as exemplars in the exemplar subsystem, where no learning is carried out. During the testing phase, a test sample is compared against the rules that have been previously extracted. If the activated rule is a low coverage rule, the sample may be in the boundary region where the concept is ill-defined, so it is directed to the exemplar subsystem for classification. If the sample is covered by a rule that is not of low coverage, it will be classified by the rule subsystem according to that rule.

3.4 Experiments and results

In this section, we show a number of experiments and results that illustrate the problem, as well as how the rule-plus-exemplar system can improve the generalizability of a comprehensible classifier constructed from decision trees. We first present the datasets from which the results are obtained. We then investigate how the exemplar subsystem can be used alongside the rule subsystem in the proposed rule-plus-exemplar framework. We show that targeting the poor performance rule regions may not be an effective way to switch to the exemplar subsystem. Instead, the low coverage rules are better at

characterizing the boundary region where the rule subsystem may be weaker. Finally, the overall rule-plus-exemplar framework is tested on the datasets and show an improved generalizability while maintaining comprehensibility.

3.4.1 Experimental data

The datasets used in this section are from the UCI Machine Learning Repository. They are labeled as follows: D1: Australian (Statlog), D2: Cylinder Bands, D3: Credit Approval, D4: Glass (2 class), D5: Heart (Statlog), D6: Letters (classes ‘O’ and ‘Q’), D7: Parkinsons, D8: Pima Indians Diabetes, D9: Sonar, D10: Spect, D11: Spectf, D12: Transfusion, D13: Vertebral, D14: Wisconsin Diagnostic Breast Cancer, D15: Wisconsin Prognostic Breast Cancer. Datasets are separated into training (70%) and testing (30%) sets, repeated over 100 iterations with random orders of the dataset. All datasets are normalized to zero mean and unit variance.

3.4.2 Targeting poor performance regions with exemplars

To improve the generalization within the training region, the rules can be supported by the exemplar subsystem. It is thus important to correctly identify the region where the rule subsystem is not performing well. The straightforward way of identifying this region is to consider the rules with poor accuracy after merging the finely partitioned rules. In Table 3.1, the percentage of training and testing samples in the poor performance rule region is shown, along with the percentage of misclassified test samples that are inside the identified region. Classification is performed using CART and C4.5.

Table 3.1 Samples and errors in poor performance rules

	CART			C4.5		
Datasets	% of training samples in poor performance rules	% of testing samples in poor performance rules	% of test errors in poor performance rules	% of training samples in poor performance rules	% of testing samples in poor performance rules	% of test errors in poor performance rules
D1	11.31%	11.68%	32.48%	7.57%	7.93%	20.72%
D2	11.64%	12.26%	16.55%	3.44%	3.45%	3.75%
D3	8.95%	10.15%	26.35%	7.00%	7.30%	17.82%
D4	13.16%	13.88%	26.56%	9.56%	9.59%	13.24%
D5	13.23%	13.46%	24.63%	10.32%	10.12%	13.94%
D6	5.32%	5.57%	27.38%	4.80%	4.86%	24.87%
D7	10.96%	11.53%	27.78%	6.69%	8.14%	21.21%
D8	13.09%	13.33%	24.17%	7.88%	7.53%	11.44%
D9	10.14%	11.59%	18.28%	6.97%	6.35%	8.11%
D10	22.04%	22.96%	43.48%	22.74%	22.22%	40.21%
D11	9.19%	9.26%	18.60%	9.52%	9.51%	18.26%
D12	28.15%	28.00%	42.71%	29.56%	29.02%	41.87%
D13	12.95%	13.76%	35.14%	4.52%	4.30%	11.17%
D14	4.57%	5.03%	20.59%	5.00%	5.73%	25.55%
D15	9.56%	11.02%	18.38%	7.11%	6.78%	8.61%

Due to overfitting of the rules to the training data, there are few rules with error rates exceeding the threshold ρ_{err} . There are thus only a small proportion of testing samples that fall in the poor performance rule region. Even though the

percentage of misclassifications that fall within this region is higher compared to the samples covered, this method of identifying the poor performance rule region is not effective.

3.4.3 Targeting low coverage rules with exemplars

We have alternatively identified that the region corresponding to rules with low coverage may correspond to the boundary region between classes. These rules cover few samples each, but are finely partitioned and generated so as to map out the class boundary. In Figure 3.5, the distribution of rules with different coverage is shown for each dataset. The horizontal axis shows the number of samples covered by a rule, while the vertical axis shows the number of such rules. These rules are generated using CART on the training datasets, and collected over 100 iterations.

It can be seen from Figure 3.5 that majority of the rules generated indeed cover far fewer samples than the other rules. Although the maximum coverage for rules differs for each dataset, the number of rules with lower coverage is always significantly larger. However, these rules collectively cover only a small portion of the training samples. In Table 3.2, it can be seen that this is also true for the testing samples. For each dataset, the rules are generated using both CART and C4.5 on the training set. The percentage of training and testing samples covered by the low coverage rules are respectively shown, along with the percentage of misclassified test samples that are covered by these rules. Results are averaged over 100 iterations, and the threshold ρ_{cov} for defining low coverage rules is set to the average coverage of the rules for that dataset in that iteration.

Figure 3.5 Distribution of rules with different coverage

Table 3.2 Samples and errors in low coverage rules

Data sets	CART			C4.5		
	% of training samples in low coverage rules	% of testing samples in low coverage rules	% of test errors in low coverage rules	% of training samples in low coverage rules	% of testing samples in low coverage rules	% of test errors in low coverage rules
D1	25.83%	27.02%	63.71%	37.59%	28.08%	65.06%
D2	29.08%	30.83%	38.75%	27.79%	35.83%	44.76%
D3	24.40%	25.26%	60.91%	39.61%	27.70%	64.01%
D4	35.26%	36.33%	54.17%	32.11%	30.61%	48.51%
D5	28.73%	30.49%	54.92%	35.82%	34.94%	55.28%
D6	20.33%	21.15%	68.42%	44.85%	21.67%	51.87%
D7	25.00%	29.15%	55.56%	35.59%	27.63%	56.48%
D8	29.29%	30.39%	52.49%	27.90%	34.68%	55.93%
D9	28.21%	32.38%	50.00%	44.28%	34.60%	48.65%
D10	25.48%	26.79%	43.40%	3.76%	27.65%	47.64%
D11	25.54%	27.53%	54.59%	21.08%	31.23%	61.99%
D12	27.61%	28.58%	49.40%	14.03%	23.33%	45.22%
D13	28.71%	29.68%	64.04%	35.48%	30.43%	66.15%
D14	14.35%	16.08%	60.29%	34.27%	17.66%	68.79%
D15	26.81%	29.66%	46.20%	33.04%	30.85%	51.46%

Even though most of the rules generated have low coverage, these rules do not cover a large portion of the training set. The percentage of training samples in low coverage rules from CART ranges from 14.35% to 35.26%, while for C4.5 it is higher, ranging between 14.03% and 44.85%. The percentage of

testing samples in the low coverage rules is between 16.08% and 36.33% for CART, and between 17.66% and 35.83% for C4.5. In dataset D6, the low coverage rules by C4.5 cover 44.85% of the training samples but only 21.67% of the testing samples. This is because the low coverage rules represent overfitting and do not generalize well, so their coverage drops drastically for the testing data. Despite the low coverage of testing samples, a large portion of the misclassified samples fall within these low coverage rules. This shows that these rules indeed represent the region where rule categorization is underperforming, and can benefit from exemplar categorization. Table 3.3 shows the improvements to generalizability. The error rates on the test set are shown for each dataset, obtained from using rule categorization. These can be compared with the error rates obtained by the rule-plus-exemplar framework. All results are averaged over 100 iterations.

Based on training data, the low coverage rules are identified. The test samples that are covered by those rules are classified by kNN representing the exemplar categorization, and the resulting classification is indeed improved over the rules obtained through decision trees. To reduce the effect of outliers while retaining comprehensibility, kNN is implemented with $k=3$, without further scanning of k for better performance. This shows that the generalizability of rules can be improved using exemplars, which is also comprehensible to humans as it is a natural categorization process. The classification errors of the overall rule-plus-exemplar system are shown.

Table 3.3 compares the classification errors obtained from classification using only rule categorization against rule-plus-exemplar categorization. For

completeness, the classification errors using only exemplar categorization are also shown.

Table 3.3 Reduced error rates using comprehensible rule-plus-exemplar system

		CART		C4.5	
	kNN	R	R+E	R	R+E
D1	16.87%	16.55%	16.39%	18.75%	17.63%
D2	25.02%	34.58%	29.60%	34.79%	28.86%
D3	16.15%	16.74%	16.05%	18.66%	16.64%
D4	20.80%	24.47%	20.78%	27.94%	23.27%
D5	19.91%	25.15%	21.42%	30.65%	24.48%
D6	2.79%	5.74%	3.82%	7.95%	5.90%
D7	9.59%	16.05%	12.78%	17.98%	13.31%
D8	27.42%	29.31%	26.19%	30.10%	27.61%
D9	16.89%	28.92%	22.22%	35.92%	24.17%
D10	19.81%	20.05%	18.88%	23.77%	21.15%
D11	27.69%	26.79%	25.25%	26.98%	24.20%
D12	25.38%	26.16%	24.33%	23.77%	23.15%
D13	20.77%	19.91%	19.62%	21.20%	19.70%
D14	3.36%	7.39%	5.32%	8.04%	5.06%
D15	26.31%	31.66%	25.63%	34.51%	28.34%

The overall rule-plus-exemplar system improves the classification of rule categorization over the testing set for all the datasets. This improvement is statistically significant to a 5% significance level, for all the datasets except D1,

D3, D13 for CART, and D12 for C4.5. The regions where rule categorization faces difficulty have been correctly identified, allowing the use of exemplar categorization. The rule-plus-exemplar system retains comprehensibility by using only rules or exemplars, both of which are easily comprehensible since they are similar in nature to the cognitive process for categorization. The two subsystems complement each other to improve the overall generalization.

As exemplar categorization may be better at dealing with ill-defined categories, this brings about the question of whether it would be better to just use it throughout the testing dataset. Even though both categorization models are comprehensible, rules are preferred for their speed of classification and their ease of interpretation. Using exemplars throughout will hamper knowledge acquisition and slow down categorization, so they should only be used where necessary. Due to the ability of exemplar categorization to deal with ill-defined categories, the classification error by kNN is sometimes lower than that of both CART and C4.5. But the eventual classification error of the rule-plus-exemplar system is even lower for 9 out of 15 datasets. This not only shows that the regions where rules underperform have been correctly identified, but also that the remaining regions can indeed be better represented using rules than exemplars.

3.5 Analysis

Under the constraints of comprehensibility, the incorporation of exemplars to rule classifiers improves the classification accuracies of rule classifiers. This has been shown in the previous section. However, due to the tradeoff between comprehensibility and complexity, the performance achieved by

comprehensible classifiers will tend to be limited compared to that of black-box classifiers. Nevertheless, this gap can be reduced through the proposed system.

Table 3.4 Comparison of error rates between black-box and comprehensible classifiers

	Black-box classifiers	Comprehensible classifiers	
	SVM (linear or kernel)	R	R+E
D1	14.83%	16.55%	16.39%
D2	25.17%	34.58%	28.86%
D3	13.65%	16.74%	16.05%
D4	24.00%	24.47%	20.78%
D5	16.59%	25.15%	21.42%
D6	1.09%	5.74%	3.82%
D7	12.54%	16.05%	12.78%
D8	23.05%	29.31%	26.19%
D9	15.54%	28.92%	22.22%
D10	17.11%	20.05%	18.88%
D11	21.09%	26.79%	24.20%
D12	22.32%	23.77%	23.15%
D13	15.48%	19.91%	19.62%
D14	2.89%	7.39%	5.06%
D15	21.83%	31.66%	25.63%

Table 3.4 compares the error rates between the comprehensible rule classifiers and black-box SVM classifiers. The better error rate of CART and C4.5 is shown for rule classifier, while the better error rate of linear and kernel

SVM is shown for black-box classifiers. Kernel SVM is implemented using RBF kernel and the cost parameter c for SVM is scanned from 0.5 to 5 in step sizes of 0.5. It can be seen that black-box classifiers generally perform much better than the comprehensible rule classifiers. However, with the incorporation of exemplars as proposed in this chapter, the gap in performance between comprehensible and black-box classifiers can be reduced. The proposed method achieves comparable performance for datasets D7 and D10, and even outperforms SVM for D4.

The use of exemplars improves the classification performance of rule classifiers while retaining comprehensibility. The following shows two examples of rules that are generated from rule-plus-exemplar for D12:

- IF $x(1) \geq -0.001$, $x(2) \geq 1.881$, THEN class = -1
- IF $x(1) < -0.001$, $x(2) < 3.936$, $x(4) \geq 0.952$, $x(2) \geq 1.710$, THEN
classify according to the k most similar samples

Despite the tradeoff between comprehensibility and complexity, the accuracy of the proposed comprehensible classifier cannot be increased simply with a mere increase in the number of rules or rule antecedents for the extracted rules. This is because rules generated are restricted to hyper-rectangular mappings in feature space, which cannot effectively capture non-linear decision boundaries or regions, or even non-dimension parallel cuts. While the size of a rule set determines how comprehensible the trained system is, it is not solely determined by the complexity of the problem, therefore a large number of rules does not necessarily imply better ability at representing a complex problem.

Table 3.5 Average number of rules generated for rule-plus-exemplar classifiers

	Average number of rules generated		Characteristic of dataset	
	CART + exemplar	C4.5 + exemplar	No. of features	No. of training samples
D1	26.17	57.01	14	483
D2	22.41	47.64	31	196
D3	25.82	54.04	15	458
D4	9.87	16.59	9	115
D5	14.35	36.13	13	189
D6	30.28	84.04	16	1076
D7	7.06	11.25	22	137
D8	49.15	93.12	8	538
D9	11.76	21.49	60	146
D10	12.68	25.49	22	187
D11	12.72	20	44	187
D12	43.72	35.9	4	524
D13	12.54	25.83	6	217
D14	8.99	14.14	30	399
D15	12.32	21.19	33	136

The average number of such rules created for each dataset is shown in Table 3.5. C4.5 tend to create larger trees than CART, resulting in a larger number of rules generated overall. This size is influenced by the complexity of the decision boundary between classes. However, in addition to the complexity, the size of the rule set is also influenced by the density of the training samples in

the feature space. Due to the hyper-rectangular segmentation of the feature space by decision trees, non-linear as well as non-dimension parallel decision boundaries tend to be poorly mapped by the rules. A sparse distribution of the training samples along the decision boundary provides more allowance for the larger hyper-rectangles, which translates to larger coverage rules and thus a smaller rule set size. It can be seen from Table 3.5 that the D6, D8, D12 with the largest number of rules have a large number of training samples for a lower dimensional feature space.

Although the rule set size affects how comprehensible the overall system is, the classification of each sample can be explained with only one single rule. This ensures the classification remains straightforward and comprehensible.

3.6 Chapter summary and concluding remarks

This chapter presented a rule-plus-exemplar classification system for comprehensible classification. Comprehensible classifiers usually suffer from poor generalization and accuracy, and there have been efforts to improve the comprehensibility of black-box classifiers that have better accuracy. However, the extracted rules do not fully match the trained black-box models, resulting in a gap in the classification using such rules. We explore an alternative to the comprehensibility problem, by raising the accuracy of comprehensible classifiers through improving their generalizability.

Highly comprehensible rules are extracted from the training data through decision trees. Generalizability is improved by identifying the boundary region where rules are not able to classify well. This region is characterized by low coverage rules that map out the class boundary but do not generalize well.

Results indicate that a large number of rules with low coverage are generated, and they cover only a small proportion of the test data with significantly higher proportion of misclassifications. Through the identification of this region, generalizability can be improved by directing samples in the region to exemplar categorization. The resulting rule-plus-exemplar system shows improved classification accuracy.

Comprehensibility of the rule-plus-exemplar system is retained through use of categorization models that are natural to humans and shown to be in use in cognitive experiments. The classification of any sample can be explained using the rule that applies to it, or the neighboring exemplars that are most similar to it. Although the representation of the training region involves some abstraction from the training dataset, it has a direct corresponding interpretation that is intuitive and easy to understand.

The strength of the proposed system lies in its intuitive classification and the explicit rules and exemplars that can be provided for any sample classification. This allows expert knowledge to be incorporated by representing the knowledge in similar form. Furthermore, these rules and exemplars exactly match the classification process, rather than provide an approximate understanding to the workings behind the process. The emphasis of the problem is on the interpretability and accuracy, thus the current implementation of the exemplar subsystem stores all the samples. This number can be reduced by storing only the samples near or covered by the low coverage rules.

Chapter Four

Rule-plus-exemplar framework for classification extrapolation

Pattern classifiers learn from training data assuming they are representative of the concept to be learnt. However, testing samples sometimes fall outside what has been represented by the training data and are thus difficult to classify. In this chapter, we propose a rule-plus-exemplar framework to deal with classification of data which may fall outside the training region, which is the input space covered by the training data. The key lies in identifying this training region, for which we propose a novel method as well as a more efficient variant. This will also aid in overcoming other problems, which will be detailed in later chapters where it is applied. Part of the contents of this chapter has been published in (Sit and Mao 2012a).

4.1 Motivation and problem description

While many pattern classification methods have been shown to perform well on a variety of scenarios, they are being trained and evaluated on a central assumption – that the training data is representative of the concept being learnt (Polikar 2006). Evaluation methods have been discussed in Chapter 2. Even though holdout sets or k-fold cross validation separate the available dataset into training and testing sets, they are merely randomly ordered subsets of the original dataset. The classifier learns from the training set, and is then evaluated on the testing set which falls within the same input feature space that has already been covered by the training data. This is generally a fair and justifiable assumption for pattern classifiers. The basic goal of a classifier is to learn from the training data available which hopefully sufficiently represents the concept it wants to learn. So testing data should arise from this same concept which has already been represented.

However, as the methods are being used in actual applications, this assumption gets progressively violated. Representative and complete training data may not be available at the time of classifier learning. The training data may also be part of a larger concept, of which the training data constitutes only a small portion. In addition, changes in a dynamic environment (Sayed-Mouchaweh and Lughofer 2012) may lead to testing data that falls in new regions in the input feature space (Lughofer 2012a).

We refer to the input feature space that has been represented by the training data as the training region. Even though the classifiers may not be adequately trained to classify outside the training region, most of them will still make a

prediction. This can be more clearly illustrated using decision tree classifiers that have been looked into in the preceding Chapter 3.

Decision tree classifiers compare the attributes of samples with certain values, creating dimension parallel cuts in the input feature space. These cuts partition the space into separate regions, each mapped to a specific class. While the partitions are often referred to as hyper-rectangles, they are not always bounded. This is most apparent beyond the training region. In addition, as the number of dimensions increases, not all the features will be used for most of the extracted rules, if not all. Figure 4.1 shows some sample rules extracted from a dataset with 14 features. The feature values of any sample x are given by $x(1)$ to $x(14)$. The rules extracted from the training data clearly cover more input space than represented by the training data. On any given feature, the rule may impose only one restriction that leaves it unbounded on one side. Furthermore, since not all the features are of relevance to the classification, many rules involve only a small subset of the features.

<p>IF $x(8) < -0.046$, $x(5) \geq 1.392$, THEN class = -1</p> <p>IF $x(8) \geq -0.046$, $x(14) \geq -0.154$, THEN class = 1</p> <p>IF $x(8) < -0.046$, $x(5) < 1.392$, $x(2) < 0.212$, THEN class = -1</p> <p>IF $x(8) < -0.046$, $x(5) < 1.392$, $x(2) \geq 0.212$, $x(2) < 0.230$, THEN class = 1</p> <p>IF $x(8) \geq -0.046$, $x(14) < -0.154$, $x(13) < -0.430$, $x(4) < -0.620$, THEN class = -1</p> <p>IF $x(8) \geq -0.046$, $x(14) < -0.154$, $x(13) < -0.430$, $x(4) \geq -0.620$, THEN class = 1</p>
--

Figure 4.1 Sample rules and features used in rule antecedents

Figure 4.2 Distribution of rules with different number of features

In fact, for most datasets, the rules involve only a small number of features out of the full feature set. Figure 4.2 shows the histograms depicting the distribution of rules extracted using CART from the datasets used in Chapter 3 Section 3.4. Results are accumulated over 100 iterations, and the dotted line shows the number of features for that dataset.

It can be seen that for most datasets, the maximum number of features used in rule creation are lower than the number of features for that dataset. Some datasets have rules that involve all the features, but even then they make up only a small portion of the total number of rules.

For the described decision trees, as well as many other classifiers, classification of test data outside the training region is a test of the extrapolative capabilities of the classifier. Training of classifiers can be seen as learning the boundary that separates samples from two classes in their input feature space. This boundary should already correctly separate most of the training samples. Test samples among the training samples can be classified according to the boundary found, which is a test of the classifier's interpolative abilities. Outside the region, it is not known how the boundary will be extrapolated.

The problem can be better illustrated in 2-dimensional space. Figure 4.3 shows the boundaries obtained by a number of pattern classifiers – SVM, CART, MLP and ELM. The points denoted by 'o' and '+' represent training data of different classes, and the shaded regions correspond to the decision space of each class as determined by each classifier. While the training data are mostly correctly classified by the learnt boundaries, these boundaries can

behave erratically outside the covered region. We will refer to the samples outside the covered region as unfamiliar samples hereafter.

Figure 4.3 Erratic behavior of decision boundaries outside training region

The erratic behavior of the classifiers outside the training region results in classification that is also counter-intuitive. From the diagrams, an unfamiliar sample is depicted by an empty diamond. Under classification by SVM and MLP, the unfamiliar sample is classified as the 'o' class even though it is much closer to the '+' class. Under classification by CART and ELM, the unfamiliar sample is classified as the '+' class even though it is much closer to the 'o' class. Intuitively, we are more likely to guess that the unfamiliar sample belongs to the same class as the samples which are closer to it. This behavior is in line with the exemplar categorization model in cognitive psychology.

Despite the inconsistent and unreliable classification of unfamiliar samples, most classifiers can and will assign class label outputs. This problem has been recognized by some, and variations can be made to specific algorithms to refrain from making decisions. For example, a sample that requires the nearest hyperbox in fuzzy ARTMAP to expand beyond a threshold can indicate that it is too far from the training data. Modifications can be made to multilayer perceptrons to train separate subnets for identifying whether a sample is unfamiliar based on low responses to all the subnets, such that the sample will not be classified (Chakraborty and Pal 2003). The issue of unfamiliar samples has also been considered under the concept of ignorance (Lughofer 2012b) and implemented (Lughofer and Buchtala 2012) in the context of evolving fuzzy classifiers.

Although the above changes prevent classification from being made on unfamiliar samples, the changes are dependent on the algorithms used and cannot be widely applied. The question persists for pattern classifiers in general, and it is important that they be equipped with a way to flag a sample when it is out of their scope of classification. We propose a way to identify the training region, such that any sample outside the training region can be determined as likely to be unreliably classified. Appropriate actions can then be taken on these identified samples to facilitate extrapolation. The training region has also been referred to as the sampling window, and estimated using a convex hull (Ripley and Rasson 1977; Smith and Jain 1984). However, this places restrictions on the distribution on the training data. The training region has also been estimated using a function (Schölkopf, Platt et al. 2001), obtained through applying one-class SVM on the training data. But this not only requires performing

optimization to identify the training region, it also requires adjusting the parameters to determine how much of the training data should be included in the training region. In contrast, our proposed method elaborated below does not require extensive parameter selection, and can represent arbitrary training data distributions.

4.2 Characterization of the training region

In 2-dimensional data, the training region of a given dataset can be easily identified through visual inspection. We are able to take into consideration factors such as outliers, density of points and whether the samples have a multi-modal distribution. However, the identification is not as straightforward for machines, and even more complicated in higher dimensions. Data distributions differ vastly between datasets, and parametric measures such as Mahalanobis distance place unnecessary assumptions on the distribution. The distance of a sample to the nearest training samples can give a clue to whether it lies within the training region, but then the question of how close the distance should be arises.

We propose an evaluation function to characterize the training region. This evaluation function has a large value for any sample that is close to the training samples, decreasing as it moves further away. It should also have lower values when it is close to fewer training samples. We define the evaluation function as

$$T(x) = \frac{1}{m} \sum_{i=1}^m F_i(x) \quad (4.1)$$

where m is the number of samples in the training data and $F_i(x)$ are Gaussian functions. Each $F_i(x)$ is centered at a training sample s_i and is defined as

$$F_i(x) = \exp \left[-\frac{\|x - s_i\|^2}{2\sigma^2} \right]. \quad (4.2)$$

$F_i(x)$ is largest when $x = s_i$ which is the center of the function. As x moves further away from the center, the function value will decrease. The spread parameter σ determines the spread of the Gaussian function, and is sometimes also referred to as the smoothing parameter. A large value corresponds to a wider spread of the function. This means that for a function $F_i(x)$ with larger parameter value of σ , the function value will decrease more slowly with distance of x from the center s_i .

This evaluation function $T(x)$ is similar to the kernel density estimation function (Silverman 1986) and Yager's mountain method for estimation of cluster centers in Fuzzy C-Means (Yager and Filev 1994). However, the function will not be evaluated at every grid point of the discretized input space, but only when required.

4.2.1 Choosing the spread parameter value

The significance of the spread parameter σ lies in the regularization of the overall evaluation function. A value too large results in over-regularization such that the evaluation function returns large values even for samples that are far from the training data. A value too small results in under-regularization, such that the evaluation function returns very small values even when the sample is

near the bulk of training data but not close to a particular training sample. The problem of over-regularization has been considered in other contexts (Meisel 1972). It is clear that the value of the spread parameter should differ according to the datasets, depending on the number of dimensions and the density of the training samples. Its value should thus be determined directly from the dataset.

The evaluation function $T(x)$ is the average value of the individual function values. In order for the sum of Gaussian functions to collectively give a larger value for any sample close to the bulk of training data, the spread parameter should be large enough such that the Gaussian functions can cover up to the nearby training samples. We can thus set the spread parameter to the average distance of the nearest neighbor from any sample,

$$\sigma = \frac{1}{m} \sum_{i=1}^m \|s_i - s_{i'}\| \quad (4.3)$$

where

$$i' = \underset{k}{\operatorname{argmin}} \{\|s_i - s_k\|\}, k \in \{1, \dots, m\} \setminus \{i\} \quad (4.4)$$

is the index of the nearest training sample.

(a)

(b)

Figure 4.4 Contour plot of under-regularized $T(x)$ values

Although the spread parameter is determined directly from the dataset by considering the average distance of the nearest neighbor, it does not take into consideration the different densities of distributions throughout the data. Consider the dataset shown in Figure 4.4(a). The training samples are distributed across two different clusters with different densities. Using the

average distance to nearest neighbor for the spread parameter results in Gaussian functions that are too narrow for the sparse cluster. As a result, the evaluation function $T(x)$ is under-regularized for that cluster, as shown from the contour plot in Figure 4.4(b). In the absence of color for the plot, the outermost contour lines represent the same value of $T(x)$.

It is thus necessary to provide different σ values to each Gaussian function

$$F_i(x) = \exp \left[-\frac{\|x - s_i\|^2}{2\sigma_i^2} \right] \quad (4.5)$$

by setting

$$\sigma_i = \min_{k=\{1,\dots,m\}\setminus\{i\}} \{\|s_i - s_k\|\}. \quad (4.6)$$

However, setting the spread parameter to the distance to the nearest neighbor may cause the evaluation function to be sensitive to outliers. Let

$$D_i = \{\|s_i - s_j\| : j \neq i, j = 1, \dots, m\} \quad (4.7)$$

be the set of distances between a sample s_i and all other samples. The distances are sorted in ascending order to obtain

$$D_i^{(sorted)} = \{d_1^{(i)}, d_2^{(i)}, \dots, d_{m-1}^{(i)}\} \quad (4.8)$$

(a)

(b)

(c)

(d)

Figure 4.5 Contour plots of $T(x)$ on various datasets

where $d_1^{(i)} \leq d_2^{(i)} \leq \dots \leq d_{m-1}^{(i)}$. The parameter σ_i can then be set as $d_{n_\sigma}^{(i)}$, which is the distance to its n_σ -th nearest neighbor. This allows the spread parameter values to be adjusted accordingly when the local data density changes, as well as to be adjusted according to the number of dimensions in the data. The purpose of the parameter n_σ is to focus on the local density, so it should be set to a small value. However, $n_\sigma = 1$ would cause the spread to be easily affected by outliers. Therefore, the parameter can be set as $n_\sigma = 3$ without necessitating further tuning. A number of tests have shown that this value performs adequately. Figure 4.5 shows the contour plots of the evaluation function $T(x)$ computed using Equations (4.1) to (4.8), with parameter $n_\sigma = 3$. Even though the evaluation function makes use of Gaussian functions, there is no assumption on the data distribution of the training set.

4.2.2 Determining the cut-off threshold value for identifying the training region

Given the evaluation function $T(x)$ which has higher values for samples close to the bulk of training data, the next step in identifying the training region is to select the appropriate cut-off threshold function value. The training region can then be identified as

$$R = \{x: T(x) \geq \tau\}. \quad (4.9)$$

This threshold τ can be computed from the available $T(x)$ values rather than set to a predetermined fixed value. Although it is possible to identify all the training samples by setting

(a)

(b)

(c)

(d)

Figure 4.6 Identified training regions

$$\tau = \min_i \{T(s_i) : i = 1, \dots, m\}, \quad (4.10)$$

this may make the training region susceptible to the influence of outliers.

Instead, we set

$$\tau = P_{p_\tau}(T(s_i)), i = 1, \dots, m \quad (4.11)$$

where $P_n(S)$ denotes the n -th percentile of a set of values S . This means only $p_\tau\%$ of the training samples would not be considered as part of the training region, so for small values of p_τ this may give better overall generalization. Figure 4.6 shows the training regions identified using $p_\tau = 5$ illustrated through the contour plots of the evaluation function values at the threshold value τ .

4.2.3 Identifying the training region

The algorithm for finding the training region can thus be summarized in Figure 4.7. The spread parameter for each Gaussian function is first computed from the distance to nearby samples, and the overall evaluation function $T(x)$ can be found. The threshold τ is then computed from the $T(x)$ values of the training samples. This gives the representation of the training region.

Algorithm for identifying the training region**Input:** Dataset $D = \{s_1, \dots, s_m\}$ **Initialize** n_σ and p_τ **For** $i = 1, \dots, m$ Find D_i according to Equation (4.7)Sort in ascending order to get $D_i^{(sorted)} = \{d_1^{(i)}, d_2^{(i)}, \dots, d_{m-1}^{(i)}\}$ Set spread parameter to $\sigma_i = d_{n_\sigma}^{(i)}$ **End for**

Obtain evaluation function

$$T(x) = \sum_{i=1}^m \exp \left[-\frac{\|x - s_i\|^2}{2\sigma_i^2} \right]$$

Calculate the threshold

$$\tau = P_{p_\tau}(T(s_i)), i = 1, \dots, m$$

Output: Function $T(x)$ and threshold τ

Figure 4.7 Algorithm for identifying the training region

4.3 Reduced representation of the training region

We have shown how the training region can be identified, providing a non-parametric explicit representation that is independent of the classification algorithm used. However, the representation requires reference to all the samples in the training dataset. This may not be feasible in some situations such as dynamic environments when the training samples cannot all be stored. To improve the efficiency, we propose a reduced representation which requires reference to far fewer training samples.

Algorithm for identifying training region with reduced representation**Input:** Dataset, D **Initialize** number of desired Gaussian functions, r Cluster dataset D to obtain cluster centers $\{c_1, \dots, c_r\}$ **For** $i = 1, \dots, r$

1. Find the samples in cluster C_i
2. Calculate the set of distances to cluster center, $DC_i = \{\|x - c_i\| : x \in C_i\}$
3. Set spread parameter $\sigma_i = P_{p_c}(DC_i)$ to the p_c -th percentile of DC_i

End for

Obtain reduced evaluation function

$$T_R(x) = \frac{1}{r} \sum_{i=1}^r \exp \left[-\frac{\|x - c_i\|^2}{2\sigma_i^2} \right] \quad (4.12)$$

Calculate threshold

$$\tau_R = P_{p_t}(T_R(x)), x \in D \quad (4.13)$$

Output: Function $T_R(x)$ and threshold τ_R

Figure 4.8 Algorithm for identifying training region with reduced representation

The goal of the reduced representation is to use fewer Gaussian functions to construct the evaluation function $T(x)$. Not only will there be fewer function centers to be stored, the number of Gaussian function computations will be fewer, thus improving computational efficiency. The training region is thus approximated with fewer Gaussian functions $F_i(x)$. Their function centers are found by clustering the training data into r clusters and setting them to the

cluster centers. The spread parameters σ_i must now be set in a different way. Each Gaussian function should stretch wide enough to cover samples in that cluster. The algorithm for implementation is given in Figure 4.8.

There are different ways that the clustering can be performed for reducing the number of Gaussian function centers. K-means is a commonly used clustering algorithm. The desired number of clusters k is specified, then k initial cluster centers are selected. The algorithm iteratively clusters the input samples to the nearest cluster center and updates them as the centroid of the respective cluster members. The iterations terminate when there are no further changes to the cluster centroids. Initialization of cluster centers can be randomly chosen from the input samples.

Using clustering, we reduce the number of Gaussian functions to be stored. Figure 4.9 shows the identification of the training region using this reduced representation. The number of Gaussian functions is set to $r = \lceil 0.1m \rceil$, which is only 10% of the total number of samples. For different compression rates of the reduced representation, the number of Gaussian functions r can be adjusted accordingly. The smaller r is, the more approximated and imprecise the representation will be. The spread parameter for each cluster is computed using $p_c = 90$, and the threshold τ_R is computed using $p_\tau = 5$ as before. The parameter p_c reflects the percentile of distance of points in a cluster to the cluster center. When it is set to 100, the spread of that Gaussian center is set to the distance of the furthest point in that cluster. But this renders the value susceptible to influence by outliers, thus it is set to 90 here.

(a)

(b)

(c)

(d)

Figure 4.9 Identified training regions under reduced representation

Clustering for reducing the number of Gaussian functions is performed using k-means clustering here although other methods can also be used. Some clustering techniques are less sensitive to the initialization of cluster centers

through adaptive learning (Chinrungrueng and Sequin 1995) or can reduce the effect of outliers that trap the algorithm in local minima (Chen 1995). The experiments in this chapter refer to the most basic k-means clustering that has not been optimized, as the purpose of clustering is to demonstrate the feasibility of a reduced representation. Parameter values are thus also not tuned individually. The resulting training regions are not as precisely identified as using the full representation, but there are substantial computational savings.

4.4 Extrapolating beyond the training region

In Section 4.1, we explained the importance of identifying the training region where a classifier has been provided with the data for learning. We then proposed two methods to explicitly describe the training region – one with full representation and one with reduced representation that is computationally more efficient. The next question that arises is what can be done when samples fall outside the training region. In a highly dynamic environment, frequent updating and continuous learning will be required. But even for static environments or other applications without drastic changes in the concept, the testing data may still fall outside the training region. For such unfamiliar samples, the classifier should take the appropriate measures to classify them.

Many existing methods focus on the identification of unfamiliar samples without classifying them. There are a large number of novelty detection techniques available, which are detailed in the reviews by Markou and Singh (Markou and Singh 2003a; Markou and Singh 2003b). They may be parametric approaches, making certain assumptions on the distribution of the data, or non-parametric approaches, which makes no such assumptions and can be used

generically. Classifier-specific approaches are also available, specifically tailored to neural networks or SVM classifiers. However, these novelty detection techniques are only focused on the identification of unfamiliar samples. A successful identification may result in the classifier refraining from making a classification, or may trigger off subsequent retraining.

In this section, we are focused on extrapolating the classification on unfamiliar samples, through ideas from cognitive psychology. Hooker (2004) considered the extrapolation problem and proposed to switch to a stable constant value model for extrapolation regions. This is done by adding data from a uniform distribution to the classifier, which reflects the assumption of an underlying uniform distribution. However, we make no such assumptions, and address unfamiliar samples using lazy learning methods as opposed to eager learning methods.

Eager learning methods make abstractions on the training data which can be applied on new unlabelled data. While these abstractions are not always explicit IF-THEN rules, they can be regarded as high level rules or sets of instructions on how to classify samples. In Figure 4.3, we have seen how the classifiers of eager learning type have inconsistent and counter-intuitive extension of their class boundaries outside the training region. This is because when the classifiers perform abstraction on the training data, the whole dataset has been considered, and the resulting abstraction is optimized to that dataset. When a test sample falls outside the training region, the classifier is tasked with processing it using information from all of the dataset, including much data that are irrelevant to that sample.

Both rule and exemplar categorization models have been established to be in use in cognitive psychology, and the two have complementary strengths and limitations. Here, as the eager learning classifiers representing higher order rules face difficulty in classifying unfamiliar samples, we explore the possibility of alternative techniques that are similar to exemplar categorization. Nearest neighbor and instance based algorithms are similarity based techniques that behave like exemplar categorization. Samples are classified by comparing with their nearest neighbors or most similar instances. For unfamiliar samples, this is more reliable than using eager learning classifiers because of the local nature of exemplar techniques. Only the most relevant samples from the available data will be used to classify an unfamiliar sample.

Figure 4.10 Intuitive extension of decision boundary outside training region

In the following, we will model the exemplar categorization using kNN algorithm. While it is also possible to use instance based techniques, it will be

clear in later chapters that they are not suitable for dynamic environments. Normalization of feature values is difficult because there is no prior knowledge to the possible range of values. Storing the concept boundary is also a problem because the knowledge of this boundary changes with time, depending on the development of the concept. Figure 4.10 shows how the class boundary obtained by kNN using $k = 3$ extends beyond the training region. Unlike the boundaries obtained in Figure 4.3, using exemplar categorization modeled by kNN offers a much more intuitive classification of unfamiliar samples, even if it may not necessarily be aligned with the actual class labels. We thus utilize the rule-plus-exemplar structure through two subsystems. Classification is directed between the two subsystems, operated by a switch that determines whether the sample is in the training region such that it can be classified by the trained rule subsystem, or outside the training region such that it should be handled by exemplar based classification. The process is depicted in the Figure 4.11.

Figure 4.11 Rule-plus-exemplar system for testing data that may fall outside training region

4.5 Experiments and results

This section demonstrates that the methods proposed can identify the training region represented by a given training dataset. While earlier parts of the chapter have illustrated this in two-dimensions which facilitate visualization, we now test the methods in higher dimensional data. This requires experimental data with testing samples outside the region covered by the training data. We will first present a method to generate such training and testing sets from any given dataset. Based on the generated datasets, we will illustrate the importance of identifying the training region by showing that the classifier accuracy outside the training region is indeed drastically poorer than inside. To test our proposed methods of identifying the training region using both the full and reduced representations, the locations of the samples are determined and their actual positions are compared to the training region. Finally, we demonstrate the benefit of correctly identifying the training region by showing the potential improvements that can be made to the overall classification when test samples fall outside the training region.

4.5.1 Experimental dataset

Given any dataset, the algorithm in Figure 4.12 generates four subsets – the training set TR , an internal testing set TI that comes from within the training region, a buffer set TB which lies just outside the training region, and an external testing set TE which is furthest from the training region. The buffer set represents the samples which may still be correctly classified by the classifier due to the extrapolative capability of most classifiers, even if it is limited. The external testing set will clearly illustrate the difficulty in classification of

samples outside the covered region. In extracting the external testing set and buffer set, care is taken to ensure no single class would be entirely removed from the training set and cause it to be completely unrepresented.

Algorithm for generating testing data outside training region

Input: Dataset $D = \{x_1, \dots, x_n\}$ with corresponding class labels $\{y_1, \dots, y_n\}$

Set $V = \{T(x) : x \in D\}$

Find set of distinct classes $W = \{\omega_1, \dots, \omega_c\}$ such that $y_i \in W$ for all i

For $i = 1, \dots, c$

1. Find $V_i = \{T(x_i) : x_i \in D, y_i = \omega_i\}$
2. Sort V_i in ascending order to get $V'_i = \{t_1, \dots, t_{n_i}\}$ where $t_1 \leq \dots \leq t_{n_i}$
3. Find $\tau_i^{ext} = t_{\lfloor 0.1n_i \rfloor}$ and $\tau_i^{buf} = t_{\lfloor 0.2n_i \rfloor}$

End for

Threshold for the external region $\tau^{ext} = \min_{i=1, \dots, c} \tau_i^{ext}$

Threshold for the buffer region $\tau^{buf} = \min_{i=1, \dots, c} \tau_i^{buf}$

External testing set $TE = \{x : T(x) \leq \tau^{ext}\}$

Buffer set $TB = \{x : \tau^{ext} < T(x) \leq \tau^{buf}\}$

Internal set $IS = \{x : T(x) > \tau^{buf}\}$

Randomly sample 20% from IS to form TI and the remaining to form TR

Output: TR, TI, TB, TE

Figure 4.12 Algorithm for generating testing data outside training region

The algorithm finds the $T(x)$ values for all the samples. The smaller values indicate the samples are further away from the bulk of the dataset. Thus the

samples with lowest values are placed in the external region, followed by a buffer region, while the remaining samples with higher values make up the internal region. The thresholds for $T(x)$ values defining each region are determined based on the values for each class so that no class will be unrepresented in the internal region where the training set will be constructed.

Datasets used in this section are from the UCI Machine Learning Repository. D1: Australian (Statlog), D2: Cylinder Bands, D3: Credit Approval, D4: Heart (Statlog), D5: Letters (classes ‘O’ and ‘Q’ only), D6: Parkinsons, D7: Pima Indian Diabetes, D8: Sonar, D9: Transfusion, D10: Wisconsin Diagnostic Breast Cancer, D11: Wisconsin Prognostic Breast Cancer. Some datasets used in Chapter 3 are omitted as they do not contain both classes in TR or TE . All datasets are normalized to zero mean and unit variance before applying the algorithm. Each data generation is performed on a random 70% of the dataset to ensure diversity in the generated datasets.

4.5.2 Classification of unfamiliar samples outside the training region

It is important to identify the training region for a classifier where it has been represented by the training data. This is because the classification of unfamiliar samples outside the training region is unreliable, especially for eager learning methods that perform abstraction. Table 4.1 shows the error rates of the SVM, MLP, ELM and CART classifiers for sets TI and TE , corresponding to testing samples inside and outside the training region. The following parameters for the classifiers are scanned for the optimal value using the training data. SVM cost parameter is scanned from the values 2^{-5} , 2^{-3} , 2^{-1} , 2^1 , 2^3 ,

2^5 , 2^7 ; and the sigma parameter is scanned from the values 2^{-2} , 2^0 , 2^2 , 2^4 , 2^6 . MLP and ELM hidden layer neurons are scanned from 10 to 30 in steps of 5. ELM activation function is scanned through the sigmoid, sine, hard limit, triangular basis and radial basis functions. Optimal parameter values are determined using 5-fold cross validation of the training data. The error rates are averaged over 100 iterations with a new data generation each time.

Table 4.1 Comparison of error rates on internal and external test sets

	SVM		MLP		ELM		CART	
	TI	TE	TI	TE	TI	TE	TI	TE
D1	13.8%	46.0%	17.3%	30.9%	14.8%	24.5%	16.6%	31.1%
D2	28.9%	35.0%	36.2%	43.9%	37.2%	47.3%	33.9%	43.7%
D3	13.0%	40.9%	16.7%	27.3%	13.3%	24.5%	15.7%	27.4%
D4	20.2%	34.4%	28.2%	34.0%	24.1%	29.4%	26.4%	29.1%
D5	2.0%	22.7%	3.6%	21.1%	6.0%	19.9%	6.3%	28.4%
D6	16.2%	30.4%	22.9%	33.3%	24.3%	22.8%	22.3%	7.4%
D7	22.5%	44.4%	26.7%	39.1%	25.4%	39.6%	29.0%	35.3%
D8	18.9%	71.1%	27.6%	45.1%	31.7%	40.8%	31.6%	46.4%
D9	23.9%	30.0%	23.3%	39.5%	22.1%	39.2%	26.7%	30.1%
D10	4.0%	19.7%	5.8%	18.4%	4.8%	11.7%	7.9%	15.7%
D11	23.1%	21.4%	31.0%	41.2%	29.3%	34.1%	32.3%	44.1%

The error rates for *TI* and *TE* show that the classification outside the training region is drastically poorer than inside the region. They are also highly inconsistent among classifiers, such as for dataset D6 where the error rate for

TE ranges from 7.4% to 33.3%. Within the same classifier, the extent of classification deterioration in TE also differs for each dataset, even though the extension of boundary may sometimes coincide with the actual class boundary. The error rates reflect the difficulty of reliably classifying unfamiliar samples outside the training region.

4.5.3 Identification of training region

Although TB and TE are generated such that they have the lowest $T(x)$ values, these values are computed using Gaussian functions centered at every sample of the dataset. In identifying the region covered by TR , only the samples within TR can be used as centers for the Gaussian functions in $T(x)$. The identification process is carried out with only the set TR known, and no knowledge of whether other samples were generated from within or outside the training region. From the contour plots in Figure 4.5, we have seen that the samples with highest $T(x)$ values may not necessarily converge at the same region in the input feature space, depending on the data distribution.

Using the samples in TR , the training region is identified. This representation is checked by finding the percentage of samples in TI that have been correctly identified as inside, and the percentage of samples in TB and TE that have been correctly identified as outside the training region. The results are shown in Table 4.2. Both the full representation using parameters $p_\tau = 5$, $n_\sigma = 3$ and reduced representation using k-means clustering with parameters $r = [0.1m]$, $p_c = 90$, $p_\tau = 5$, $n_\sigma = 3$ are shown.

Table 4.2 Correct identification of samples inside and outside training region

	Samples in TI identified as inside training region		Samples in TB and TE identified as outside training region	
	Full representation	Reduced representation	Full representation	Reduced representation
D1	93.5%	97.1%	99.9%	99.1%
D2	92.7%	96.0%	100.0%	96.5%
D3	93.9%	96.9%	99.4%	97.1%
D4	92.7%	96.1%	100.0%	97.8%
D5	93.8%	97.7%	98.2%	82.9%
D6	92.6%	95.7%	100.0%	97.9%
D7	93.1%	96.9%	100.0%	98.4%
D8	93.6%	95.9%	100.0%	99.4%
D9	66.5%	95.2%	94.2%	90.1%
D10	94.1%	97.0%	100.0%	99.9%
D11	90.7%	95.2%	100.0%	97.2%

The full representation of the training region describes it more precisely, such that most samples in TB and TE can be correctly identified. Samples in TI are not all identified due to the selection of threshold τ using $p_\tau = 5$. For dataset D9, the sparseness of the training samples leads to a very precise tracking of TR samples, such that many samples in TI are deemed to be outside the training region. With a far reduced representation, the training region is more generalized and more TI samples are correctly identified to be inside the

region. Samples in TB and TE are less accurately identified to be outside compared to using the full representation, but the method still performs well.

4.5.4 Improvement in extrapolation capability

Eager learning classifiers that represent high level rule categorization have been shown to face difficulty in classification of samples outside the training region. Due to the local nature of exemplar based categorization using kNN, the extension of classification boundary outside the training region may be more intuitive and less erratic. In this section, we identify the training region from the available training samples TR using the reduced representation. The testing set comprises of TI , TB and TE , where samples may fall inside or outside the training set. Based on the representation of the training region, samples outside are identified and classified using kNN instead of the original classifier. The improvements are shown in Table 4.3. Columns labeled ‘R’ contain the error rates on the testing set using the stated rule type classifier, while the columns labeled ‘R+E’ contain error rates using the rule-plus-exemplar system described in Figure 4.11. Parameters are set to $r = [0.1m]$, $p_c = 90$, $p_t = 5$, $n_\sigma = 3$, and kNN parameter $k = 3$.

Through identification of the training region, unfamiliar samples are directed to kNN for classification. This almost always improves the error rates over the base classifier. A few exceptions occur where the classification boundary extends in a way that coincides with the actual class boundaries. The exemplar subsystem is used only for samples that have been identified to be outside the training region, where kNN would be more suitable than the rule subsystems. The last column of Table 4.3 shows the error rate if all the samples

were to be classified using kNN. The error rate would not be better than the proposed rule-plus-exemplar framework, except for dataset D6.

Table 4.3 Improvements in error rates using exemplars outside training region

	SVM %err		MLP %err		ELM %err		CART %err		kNN
	R	R+E	R	R+E	R	R+E	R	R+E	%err
D1	17.3	15.8	21.3	18.9	18.6	17.1	20.9	18.2	17.5
D2	35.4	33.3	38.6	39.0	40.7	38.3	37.4	37.7	33.9
D3	18.4	15.5	21.2	18.2	17.8	16.8	19.3	17.4	17.1
D4	18.6	18.5	30.2	25.8	26.2	23.8	27.9	23.8	18.8
D5	6.2	5.3	8.3	6.5	10.1	9.0	12.6	9.2	6.5
D6	14.0	9.8	24.1	14.5	25.1	15.3	16.7	14.0	9.1
D7	27.4	27.9	31.5	30.6	30.7	29.4	31.8	31.2	30.5
D8	30.2	24.5	32.0	31.6	35.1	33.9	37.2	32.3	27.1
D9	24.3	24.3	28.2	24.7	26.8	24.2	26.9	26.1	25.8
D10	6.0	4.1	9.4	5.2	6.0	4.8	8.6	6.4	4.7
D11	24.0	24.3	35.9	28.2	30.8	27.0	36.9	27.3	26.7

4.6 Chapter summary and concluding remarks

Identification of the training region is important as the generalization capability of classifiers is based on the central assumption that the training data is adequately representative of the concept to be learnt, such that the testing data occur from the same concept. This chapter presented two methods for identifying the training region by providing an explicit representation for the region, against which samples can be checked to determine whether a sample

falls inside. The first method uses all the training data in the representation, so for efficiency reasons, a second method with reduced representation is proposed. Based on 2-dimensional data that can be easily visualized, we see that both methods are able to correctly identify the arbitrarily shaped training region even when the data is multi modal or has different densities within the distribution. Parameters used in the representation do not require scanning for optimal values, and no assumption is made on the data distribution. Higher dimensional data are also used to validate the identified training region, which is identified entirely from the training data.

There are possible improvements that can be made to the proposed methods. The threshold value for the evaluation function $T(x)$ is computed based on a certain percentile value of the training sample function values. This leaves out part of the training data from the training region. Setting the threshold to the smallest $T(x)$ value may put it at risk of influence from outliers, but it is also possible to first check for outliers by observing a graphical plot of decreasing values of $T(x)$. Another possible improvement is on the use of k-means in the reduced representation for identifying the training region. Although k-means is a straightforward clustering technique that can be used, it is sensitive to outliers. These can be removed first before k-means is applied. These improvements only need to be applied once when the training region is being determined, and does not add further computational load during the classification stage.

The identification of training region from any training dataset will be used in the following chapters, where they can determine when the rule or exemplar type of classifier can be used.

Chapter Five

Rule-plus-exemplar classification system for concept growth

Humans are able to adapt naturally in a changing environment by updating concepts and extending knowledge to apply in new instances. This adaptability is a desirable quality for pattern classifiers, especially when applying them in changing environments. In this chapter, we are specifically interested in the problem of concept growth, in which the concept to be learnt and classified is enriched with more information over time. While humans can intuitively deal with such concepts, pattern classifiers face difficulty due to the dynamic nature of the problem. A novel classification system is proposed, based on the rule and exemplar categorization models as well as other ideas in cognitive psychology. Parts of the content in this chapter have been published in (Sit and Mao 2013c) and some under review in (Sit and Mao 2013a).

5.1 The concept growth problem

This section discusses the concept growth problem and explains why it is difficult to solve. Existing methods that deal with concept drift or perform incremental learning only solve a single aspect of the overall problem without addressing other aspects.

5.1.1 Problem description and motivation

Dynamic environments bring with them different kinds of challenges, among which the concept drift problem has gained immense interest. A description of the problem and techniques available has been provided in Chapter 2. Training and testing are no longer clearly separated into distinct phases, but are interspersed as new data is constantly made available. As the concept evolves, the classifier keeps up with the change by updating according to the most recent data, and appropriately phasing out the old knowledge to accommodate the new one.

Concept drift assumes that the change renders previous data less relevant than the current data, even in recurring change where previous classifier knowledge is merely retained in case it is deemed to be relevant again based on future data. However, this is not always the case. In many applications, the data distribution changes but previously learnt knowledge remains relevant. For example, in the detection of spam mail, new forms are generated everyday with much creativity to mask their intention. A classifier has to adapt to these changes continuously, but cannot discard any of the previous knowledge. Similarly, signatures of attacks in network intrusion detection represent

acquired knowledge that cannot be discarded by the classifier, even as attacks evolve and new ones need to be quickly learnt and identified. Or the effect on speech may vary with different variability sources, but the speech samples should still be recognized under different conditions.

This presents a different problem from that of concept drift, and there are three aspects to the problem. Firstly, like concept drift, there are changes in the concept that require adaptation of the classifier. Secondly, over the course of use in a dynamic environment, the classifier needs to not only learn the concept but also classify samples that have undergone a change in distribution. Finally, as the changes to the concept do not render previous information outdated, this knowledge should be retained. We thus refer to this problem as “concept growth” to better reflect these aspects of the problem.

The described changes in data distribution have appeared in concept drift literature under the names “virtual concept drift” (Widmer and Kubat 1993; Tsymbal 2004; Hoens, Polikar et al. 2012), “sampling shift” (Salganicoff 1997) and “population drift” (Kelly, Hand et al. 1999). Although there is no change in the definition of the classes, a drift arises from the order of data presentation or from a limited training set. This leads to testing data that have different data distributions from that of the training data. This aspect is similar to the problem of dataset shift (Quiñonero-Candela 2009), specifically covariate shift (Storkey 2009; Moreno-Torres, Raeder et al. 2012), and sample selection bias (Zadrozny 2004). Corrections based on training and testing distributions are available, as well as weighted evaluations, but the methods are not designed for continuous learning and updating.

The virtual concept drift problem has recently generated interest (Yamauchi 2009; Bartocha and Podolak 2011). Although the proposed methods adapt their learning and classification according to the concept changes, due to the relevance of previously learnt information, all the samples need to be stored for those methods. This poses a feasibility issue for use in a dynamic environment. Yamauchi later proposed regeneration of pseudo-samples to overcome this issue (Yamauchi 2010). However, even though the samples are not stored explicitly, key information and weights for all the samples are retained to facilitate approximate regeneration, after which all the generated samples are used for retraining. In a dynamic environment, we cannot afford to store and repeatedly access all the samples, or even regenerate the full set of all encountered samples.

Even though incremental learning techniques can allow the new samples to be learnt while retaining old knowledge, the adaptive aspect of the concept growth problem is neglected. Unlike standard evaluation of incremental learning techniques which only classify after the learning process is completed, dynamic environments cannot afford the same wait, which means this classification is carried out under incomplete representation of the concept. The next two sections further elaborate why the techniques for concept drift and incremental learning are inadequate for overcoming the concept growth problem.

5.1.2 Problems with using concept drift techniques on the concept growth problem

A system experiences concept growth when the data presented covers progressively new feature space. This corresponds to new knowledge which cannot be adequately represented using the previously seen training data. With more samples, the concept becomes increasingly enriched, and the goal of the system is to learn this whole concept without forgetting any of the previously learnt knowledge. This is different from the problem of concept drift, thus many methods proposed for concept drift cannot be used, as seen in the following.

Windowing techniques (Widmer and Kubat 1996; Salganicoff 1997) separate the incoming stream of data into chunks or windows such that the latest window represents the most updated information. The size of windows can be adjusted to ensure the window contains sufficient relevant information to reflect the most updated concept. While this allows focus on the most recent information, the classifiers are retrained and lose the previously learnt knowledge. For concept growth, previously presented information remains valid, so retraining results in loss of information. Some methods retain previous samples (Salganicoff 1997) in case they are relevant again later, but they are activated or deactivated according to their consistency with the current window.

Ensemble methods (Blum 1997; Kolter and Maloof 2007) evaluate base classifiers based on the current data to provide feedback on the relevance of each classifier, and this information is then used for fusion of the classifier results. Previously constructed base classifiers can also be retained (Karnick, Muhlbaier et al. 2008) in case they are evaluated to be relevant again at a later

time. This assumes that the current data is most relevant, and that test samples also occur in line with the current data. However, the concept is still growing and the previously presented information remains relevant, such that the current test samples may be in line with that previous information instead. Weighting the base classifiers according to current data makes an invalid assumption on the way test samples are presented. Instance weighting (Allan 1996) also makes similar assumptions that current data is most relevant, which is not necessarily so for concept growth.

There are a number of change detection techniques for abrupt changes (Page 1954; Roberts 1959; Hulten, Spencer et al. 2001; Gama, Medas et al. 2004; Kifer, Ben-David et al. 2004). But in concept growth, the data distribution does not necessarily change in a particular direction or manner, as samples that are presented may or may not represent new knowledge. Since there is no assumption on the direction of growth, there is no basis for comparison to detect a change.

In concept growth, the samples presented progressively cover new regions in the input feature space. As a result, data presented in the same period of time may not necessarily be similar, neither are they representative of the whole concept. If assumptions can be made about the persistence of the change, this knowledge can be incorporated into the prediction (Freund and Mansour 1997). However, samples may not necessarily be presented in a particular way. The temporal relationship often utilized in concept drift methods thus cannot be used, and previously learnt knowledge should not be discarded. This characteristic has been considered under virtual concept drift. Though existing methods consider the previously seen data during subsequent learning, storage

of these data is a difficult problem. The most straightforward way is to store all the training data for further selection or weighting during learning (Yamauchi 2009; Bartocha and Podolak 2011), or regenerate them during every round of retraining (Yamauchi 2010). Incremental learning ability would be required to retain the previously learnt knowledge without storing all the samples. Some of these ideas have been considered in the context of neuro-fuzzy systems (Angelov, Filev et al. 2010) but are not easily applied to other classifiers.

5.1.3 Problems with using incremental learning techniques on the concept growth problem

While incremental learning techniques fulfill the requirement of learning the growing concept without discarding previous information, they do not cater to the dynamic aspect of the problem. The common assumption of these methods is that while training data need not be provided all at one go, they are only tested on data which occur in feature space that has already been represented by the training data. This can be seen from the typical evaluation measures such as cross validation (Syed, Liu et al. 1999b; Ruping 2001; He, Chen et al. 2011), or accuracy on a separate partition of testing data from a randomly ordered dataset (Polikar, Upda et al. 2001; Bouchachia, Gabrys et al. 2007; Kapp, Sabourin et al. 2010). Such measures do not take into account how the data has changed, but is instead concerned with the final predictive ability of the methods. This is in contrast to concept growth where samples can progressively cover new feature space.

Under concept growth situations, incremental learning methods can still make predictions on test samples that fall outside the covered feature space. But

it will be illustrated in Figure 5.1 that this prediction is unreliable. Figure 5.1(a) shows the data distribution of the full concept, and the location of the query sample denoted by the triangle. Samples from each class are denoted by 'o' and '+' respectively. The query sample belongs to the '+' class here. Under concept

(a)

(b)

(c)

(d)

Figure 5.1 Unreliable extension of decision boundary under incomplete information in concept growth

growth conditions, an intermediate stage of sample presentation may only provide training data as shown in Figure 5.1(b), with the query sample falling outside the covered space. The possible class boundaries denoted by dotted lines indicate that the boundary determined by a classifier trained on such limited data can extend in different ways. The prediction of the query sample thus becomes unreliable. Under usual evaluations of incremental learning methods, where the training data can be as shown in Figure 5.1(c), the representation of the concept is reasonably adequate already, so there is little ambiguity in the query sample. The shaded area shows the region classified as ‘+’ using fuzzy ARTMAP. However, under concept growth, the classification by fuzzy ARTMAP can be seen in Figure 5.1(d) which instead classifies the query sample as ‘o’.

Thus in order to deal with concept growth, it is necessary to equip the classifier with not only an incremental learning ability such that previous information is not discarded or forgotten; but also the adaptive ability to provide the best possible prediction without waiting for the accumulation of representative training data.

5.2 Rule-plus-exemplar classification system

This section details the architecture and functional parts of the proposed system, and how it differs from existing methods. The steps for learning and classification are elaborated, and it is also explained why such a system can deal with the concept growth problem.

5.2.1 Difference between proposed system and existing methods

The proposed classification system for dealing with the concept growth problem operates by adopting the two main categorization strategies – rules and exemplars. Functionally, it consists of the rule subsystem made up of rule modules and the exemplar subsystem made up of training samples. Each rule module contains a trained classifier and an activation field in which the classifier is used. The system learns in a concept growth environment by progressively covering the feature space that has been populated by the training data. It classifies test samples by activating the rule modules or using exemplars for unfamiliar samples that have not been covered by the rule modules.

The learning strategy of the proposed system is different from incrementally growing strategies like fuzzy ARTMAP or sequential covering algorithms (Cendrowska 1987; Clark and Niblett 1989; An and Cercone 1998; Parpinelli, Lopes et al. 2001). Regions in the feature space are not described using dimension parallel cuts, but are instead arbitrarily shaped. Each region is not directly assigned to only one class, but rather a trained classifier valid for that region, such that the class is determined by that classifier. It is also different from (Nishida, Yamauchi et al. 2005; Nishida and Yamauchi 2007), which generate a number of classifiers but combine the results in an ensemble geared towards handling concept drift.

The proposed system directs a test sample to either the rule or exemplar subsystem for classification. Selection type ensemble systems operate in a similar way, but through identifying regions of competence (Kuncheva 2004) to determine which base classifier to use. For example, (Dasarathy and Sheela

1979) identifies a conflict domain to switch from the linear classifier to kNN. For a general collection of ensembles, the region of competence can be determined during the training phase by evaluating the classifier performances on the training or validation datasets. Alternatively, they can be determined during the testing phase according to the location of the test sample. One well known method of dynamic classifier selection is the local accuracy measure (Woods, Bowyer et al. 1996), which determines the competent classifier based on the accuracy of the classifier among the neighbors of the test sample.

However, in a dynamic environment, the competence of the classifiers cannot be pre-computed in the training phase due to the incremental presentation of new samples. These samples cannot accurately reflect the true concept which has not been fully represented yet. Determining the competence dynamically during testing also requires finding the nearest neighbors and their classification performances, which is computationally consuming and demanding on storage.

Our classification system does not identify the activation field based on evaluated performance, but on the training data used. As we make no assumptions on the direction of concept growth, training data used for different rule modules may fall in the same feature space and lead to overlaps in activation fields. This means that more than one classifier may be activated for each test sample, but inconsistencies in class predictions will be handled by the exemplar subsystem, which will be detailed in the next section. It should be noted that while each classifier has its area of activation, this does not necessarily imply its superiority over other classifiers.

5.2.2 Architectural structure of the proposed system

Figure 5.2 Architecture of rule-plus-exemplar system for concept growth

The architecture of the rule-plus-exemplar classification system is based on the memory model in humans, consisting of the LTM along with the WM. The overall structure is shown in Figure 5.2. It is based on the concepts of memory and categorization in cognition which has been explained in Section 2.4, but will not adhere strictly to any particular model.

The LTM used here consists of only the semantic and episodic memory. Episodic LTM stores the remembered exemplars, which is a small portion of all the samples that have been encountered in the training data. This set of samples will be denoted by X_e . Semantic LTM consists of the extracted rules and knowledge about the concept. While human cognition uses verbalizable and simple explicit rules, we can make use of higher dimensional and more

complex rules. These are implemented as trained pattern classifiers, which represent the knowledge of defining a concept and distinguishing it from another concept. The contents of the LTM originate from the WM, where learning takes place and where activated LTM will be loaded into for classification.

Newly encountered or unprocessed labeled samples are stored in the episodic buffer, which has a limited capacity. The set of labeled samples stored in the episodic buffer will be denoted as X_b . As the capacity approaches its limit, learning will need to be carried out, emulating the schematization that shifts information from episodic representation to conceptual representation. This conceptual representation is realized through trained classifiers stored in semantic LTM. The transfer of information between memory components is depicted by black solid arrows in the diagram.

During schematization, part of the episodic representation is stored. In human cognitive psychology, specific episodes and their context can be recalled given the appropriate cues. However, the implementation of a classification system in a dynamic environment with large amounts of data does not allow the remembering of all samples, so only a selected portion is retained. The selection of these samples is governed by their representativeness of the concept, and will be further elaborated later.

During classification, information from the LTM is activated and loaded into WM. A test sample activates the relevant and applicable classifiers in the semantic LTM, which will be loaded into WM for processing. Stored samples in the episodic LTM can also be recalled with the presentation of the test

sample. This process is similar to mental imagery (Kosslyn 1988; Kosslyn, Behrmann et al. 1995) in cognitive psychology, where a chunk of memory is recalled for processing in order to solve a particular problem. Along with other new samples, they can provide a basis for exemplar categorization. This loading of information is controlled by the central executive and depicted with dotted connector lines in the diagram. The specific mechanisms for each process will be detailed later.

Classification can be carried out by the rule or exemplar subsystem, as shown by the solid connector lines. The proposed system is thus referred to as the rule-plus-exemplar classification system.

5.2.3 Learning through rule module generation

The rule module generation process represents knowledge acquisition, transiting from remembering to knowing. It is triggered when the errors arising from rule categorization or number of unfamiliar samples exceeds a maximum threshold m_e . The errors indicate that an update to the rule subsystem is required due to inadequacy in the current state, and the set of rule-misclassified errors is denoted by $X_{\neq}$. Unfamiliar samples are defined as those which have not been well represented by the seen training data, and the set of unfamiliar samples is denoted by X_U . This is determined based on activation fields which are determined from the training data during rule module generation. The computation of activation fields will be elaborated later. Unfamiliar samples indicate that new knowledge has been presented and schematization should be carried out.

After the process is triggered, whether it is successfully followed through depends on the next step – the available samples are clustered and classifiers are generated only for clusters that satisfy a minimum size m_c . In the context of pattern classification, this is a check to ensure that there are sufficient training data that are representative of their region in feature space. Cognitively, this phenomenon of category clustering has also been observed (Bousfield 1953) during recall. Such a natural sequence of recall would provide the basis for conscious rule extraction.

When the conditions for rule module generation are fulfilled, a classifier is trained on the qualifying cluster, plus some other samples in episodic LTM. These samples activated are those which are similar to the current cluster samples, in line with the semantic proximity effect (Kahana, Howard et al. 2008). This is implemented through the definition of activation fields. The activation field of a rule module specifies the region in which that trained classifier is valid. It can be determined from the training cluster in the episodic buffer. To limit the overlap between activation fields of different rule modules, samples which can be correctly classified by the current state of rule subsystem are omitted when determining the activation fields. Given the set of data, the activation field can be found using the reduced representation presented in Figure 4.8 of Chapter 4.

The rule module thus consists of the trained classifier and the activation field defining the region in which this classifier is active. Classifiers are trained on all samples in LTM and WM that fall within this field. After the classifier is trained, the samples of the qualifying cluster are discarded, except for some

Algorithm for compacting exemplars in episodic buffer**Input:** Samples in order of presentation**Initialize** compacting parameter p_e and set $X_e^{(curr)} = \emptyset$ **For** each class i with samples $\{x_1, \dots, x_{n_i}\}$

1. Compute final number of samples to store, $u = \lfloor p_e n_i \rfloor$
2. **For** $j = 1, \dots, \lfloor n_i/u \rfloor$
 - a. Set $X_e^{(curr)} = X_e^{(curr)} \cup x_1$ and $v = \min\{\lfloor n_i/u \rfloor - 1, n_i - (j - 1)\lfloor n_i/u \rfloor\}$
 - b. Discard nearest v neighbors
 - c. Re-label remaining samples in order of presentation $\{x_1, x_2, \dots\}$

End for**End for****Output:** $X_e^{(curr)}$

Figure 5.3 Algorithm for compacting exemplars in episodic buffer

which are remembered and stored in episodic LTM according to the algorithm in Figure 5.3. This loss of episodic information is in line with the schematization process, in which conscious awareness of the episodes are replaced by familiarity and knowing. The algorithm is a heuristic that aims to maintain an approximation of the data distribution. It selects the earliest seen sample and discards nearby neighbors, doing this repeatedly until the desired number of samples remain.

In addition to rule module generation, forgetting of some information in the episodic buffer occurs, and is implemented by compacting the number of

samples. In many concept drift methods, forgetting is implemented based on time. This is not valid in the concept growth situation. Instead, the same procedure in Figure 5.3 for selecting samples to store in LTM is used here. This is carried out whenever the episodic buffer reaches its maximum capacity m_b .

5.2.4 Classification of a test sample

A test sample presented is classified according to the available classifiers and samples stored. Any activated classifier is loaded into WM, and several scenarios are possible. If no classifier is activated, the sample is simply classified by exemplars, as it represents new knowledge that is handled through remembering rather than knowing. If only one classifier has been activated, a prediction can be made according to that classifier. If more than one classifier is activated, there may be inconsistencies that arise in the predictions. Inconsistencies reflect the difficulty of classifying the sample using rules, so the prediction can be complemented with exemplars.

Consistency of predictions does not need to be enforced as a strict consensus in class prediction, but rather an agreement among reasonable majority. A consistency value can be computed for any prediction.

Given the rule modules with covering functions $T_i(x)$ and thresholds τ_i , find the activation of each rule module to a test sample x , given by

$$a_i(x) = \begin{cases} 1, & \text{if } T_i(x) \geq \tau_i \\ 0, & \text{otherwise} \end{cases} \quad (5.1)$$

and subsequently the activated rule set

$$A(x) = \{i: a_i(x) = 1\}. \quad (5.2)$$

Let the class prediction by exemplar subsystem be denoted by $predE(x)$ and the class prediction by a rule module j be denoted by $predR_j(x)$. The rule subsystem prediction is then ω^* which is the most commonly predicted class among the activated rule modules. The set of indices of rule modules predicting this class can be determined as

$$A^*(x) = \{j \in A(x): predR_j(x) = \omega^*\}. \quad (5.3)$$

The consistency of a rule subsystem prediction is then computed using

$$csty(x) = \begin{cases} \frac{|A^*(x)|}{|A(x)|}, & \text{if } |A(x)| \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.4)$$

and the prediction is said to be consistent if the consistency has a minimum value of ξ .

5.2.5 Implementing the rule-plus-exemplar classification system in concept growth environment

The overall algorithm for classification and learning of the rule-plus-exemplar classification system is given in Figure 5.4. The classifiers used for the rule modules are implemented using an SVM classifier in this paper. Exemplar classification is similarity based and intuitively matches the classification performed by kNN.

Algorithm for rule-plus-exemplar for concept growth

Input: Initial training set, subsequent training samples $\{x_1, x_2, \dots\}$ and labels $\{y_1, y_2, \dots\}$

Initialize parameters m_e, m_c, m_b and set $X_b, X_\neq, X_U = \emptyset$

Generate rule module for initial training set and find X_e

For every (x, y) presented,

1. CLASSIFY

If $csty(x) = 0$,

Classify x as $predE(x)$ using samples from $X_b \cup X_e$

Set $X_U = X_U \cup \{(x, y)\}$

Else if $csty(x) < \xi$,

Classify x as $predE(x)$ using samples from $X_b \cup X_e$

Else,

Classify x as ω^*

If $\omega^* \neq y$, $X_\neq = X_\neq \cup \{(x, y)\}$ End if

End if

2. $X_b = X_b \cup \{(x, y)\}$

3. LEARN

If $|X_U| > 0$ and $|X_\neq \cup X_U| \geq m_e$,

Set $G = \emptyset$

Cluster X_b into clusters $\{\Psi_1, \Psi_2, \dots\}$

For $|\Psi_i| \geq m_c$,

Find activation field F from $\Psi_i \cap \{X_\neq \cup X_u\}$

Figure 5.4 Algorithm for rule-plus-exemplar for concept growth

A number of issues are associated with the implementation and are addressed here. The first rule module generated is based on the initial training set, without any clustering performed. This is because the initial training set is assumed to be sufficient and coherent, otherwise learning would not have commenced. Subsequent learning requires clustering of X_b first as there is no assumption on the concept growth pattern.

Step 3 in Figure 5.4 performs clustering on the buffer samples. In the current implementation, the agglomerative clustering function provided in

MATLAB is used, using a cutoff inconsistency value of 1.5. Different clustering algorithms such as (Hamerly and Elkan 2002) may be used, but as the goal of this work is to validate the rule-plus-exemplar structure for concept growth, the issue of clustering is set aside first.

While the overlaps between activation fields introduce some amount of redundancy to classification to improve robustness, steps must be taken to limit this overlap and the subsequent complications that may arise. When generating rule modules from a single cluster of X_b , the activation field is computed based on only the samples that cannot be correctly classified under the current state of the rule subsystem. This focuses on samples that are not covered by existing rules, as well as those which were covered but not correctly classified, hence requiring generation of another rule.

Even when there are overlaps between activation fields, the classifier is trained such that they reduce potential inconsistencies. The training data does not only cover those samples in that cluster of the X_b samples, but also any other samples from other clusters in X_b of WM and remembered samples in X_e of LTM which fall within the activation field. This improves consistency between different rule modules.

5.2.6 Rationale behind the rule-plus-exemplar classification system

In a concept growth environment, there are two capabilities that the classification requires to learn and classify well – the ability to learn incrementally and the adaptation to novel samples which have not been represented by the training data.

Incremental learning in the rule-plus-exemplar system is enabled through the use of rule modules. Like the hyperboxes in fuzzy ARTMAP, rule modules can be created whenever required, imparting plasticity to the system to cope with growing needs. However, unlike the risk of category proliferation in fuzzy ARTMAP, measures are in place to prevent the pervasive creation of rule modules so that there will not be an explosion in the size of the system.

Adaptation to novel samples is achieved through the exemplar subsystem. Few methods consider this problem of classifying samples not represented by the training data because of the common assumption on the representativeness of the training set. Learn++ (Polikar, Krause et al. 2003) with dynamic voting weight update deals with this through assigning more importance to classifiers whose training data are nearer to the novel sample. In the rule-plus-exemplar system, we bring this distance even closer by using exemplars.

Exemplar classification for the unfamiliar sample outside the activation fields of the classifiers focuses on the nearest samples. These samples may come from the training data of any of the existing classifiers, as well as new samples which have not been schematized yet. And the reason why such samples provide better classification than trained classifiers is because of the local nature of the classification.

In cognitive investigations, there is a transition from episodic to conceptual representation over the process of learning a new domain. This is the motivation for using exemplar classification for unfamiliar samples which have not been adequately represented by the training data. Within machine learning, using similarity based classification such as kNN focuses on local information

while classifiers such as SVM and other eager learning methods aim to optimize globally over available training data. For a test sample that falls outside the covered region, classifying by global methods will involve all the other training data which may not be generalized to this sample. On the other hand, local methods can focus the attention on the most similar samples among the training data to make the best possible prediction on the test sample. This has been described in Section 4.1, and will be further illustrated in the next section.

5.3 Experiments and results

In this section, we first describe the experimental data and setup required for testing in a concept growth environment. The testing results of the proposed rule-plus-exemplar classification system are compared with other methods and further analysis is provided.

5.3.1 Experimental data and simulating concept growth

In order to test the ability of the classification system to deal with data undergoing concept growth, such data should be generated which can reflect the scenario and the difficulties involved. This can be done by using the function $T(x)$ to identify samples furthest from the bulk of the data. Figure 5.5 shows the algorithm for generating concept growth from a given set of data. The main idea is that the innermost samples of the dataset should be used as training data, while those lying progressively further out should be presented for testing and subsequent learning. However, the covering function cannot be used directly by simply setting the highest values as training and the others as testing according

to decreasing values of $T(x)$. The testing data should include samples from the covered region as well.

The procedure of simulating concept growth in a given dataset is described as follows. The samples furthest out in a dataset can be identified by finding the smallest $T(x)$ values. Along with a random selection of the remaining samples, these can be placed in the testing set which will reflect concept growth. While this represents limited growth in concept, the process can be repeated on the remaining samples repeatedly, with the earliest extracted samples presented last as testing data. The number of repetitions and the percentage of lowest $T(x)$ values will be denoted as n_f and p_{out} respectively. The process is described in the algorithm in Figure 5.5. Due to the random nature introduced in the process, each simulation yields a different training and testing dataset.

The parameter n_f controls how smoothly the concept grows, with a larger value corresponding to smoother growth. The parameter p_{out} determines the rate of growth as a larger value indicates a larger portion of the samples fall outside the current covered region. The data used for simulating concept growth are obtained from the UCI Machine Learning Repository. They are D1: Australian (Statlog) Credit Approval, D2: Cylinder Bands, D3: Credit Approval, D4: Heart (Statlog), D5: Letters (classes 'O' and 'Q' only), D6: Parkinsons, D7: Pima Indian Diabetes, D8: Sonar, D9: Transfusion, D10: Wisconsin Diagnostic Breast Cancer, and D11: Wisconsin Prognostic Breast Cancer.

Algorithm for simulating concept growth**Input:** Dataset $D = \{x_1, \dots, x_n\}$ **Initialize** $n_f = 8$ and $p_{out} = 0.9$ Set $n_{out} = \left\lfloor \frac{n}{n_f+1} p_{out} \right\rfloor$ and $n_{in} = \left\lfloor \frac{n}{n_f+1} \right\rfloor - n_{out}$ **For** $i = n_f, n_f - 1, \dots, 1$

1. Set n_i to the number of samples in D
2. Find function $T(x)$ for D according to Section 4.2
3. Sort the set $\{T(x): x \in D\}$ in ascending order to get $\{x'_1, \dots, x'_{n_i}\}$
4. R = randomly selected n_{in} samples from $\{x'_{n_{out}+1}, \dots, x'_{n_i}\}$
5. $S_i = \{x'_1, \dots, x'_{n_{out}}\} \cup R$
6. Remove S_i from D

End for**Output:** Training set $TR = D$ and testing set $TE = S_1 \cup \dots \cup S_{n_f}$ presented in order

Figure 5.5 Algorithm for simulating concept growth

5.3.2 Experimental setup and results

To demonstrate the effectiveness of the rule-plus-exemplar classification system in a concept growth scenario, comparisons with other methods are required. Fuzzy ARTMAP (FAM) is a highly plastic incremental learning technique that can be used in a dynamic environment by setting to single epoch training. As the proposed classification system is implemented using SVM as the rule module classifiers, it is appropriate to include an incremental form of

SVM (Syed, Liu et al. 1999a) (incrSVM) for comparison. In addition, a method designed to cope with concept drift is used, which additionally considers classification of unfamiliar samples. This is the Learn++ with dynamic voting weight update (Polikar, Krause et al. 2003), which generates ensembles of classifiers from each window of data, and weights classifiers more heavily if they are nearer to the novel samples. These three methods provide a fairer basis for comparison for the proposed system. All the datasets are also normalized to zero mean and unit variance before presenting to each classifier. In addition, FAM further requires the data to be scaled to $[0,1]$.

Parameter values for each classifier are set to the same values for each dataset. This is because the parameters cannot be scanned for an optimal value in a dynamic environment, nor will the optimal value hold as changes are introduced. FAM parameter values are set as $\rho = 0.75$, $\alpha = \varepsilon = 0.001$. IncrSVM parameter values are set as $c = 1$ and $\sigma = \sqrt{\text{no. of features}/2}$, which are also used for the SVM classifiers in the rule-plus-exemplar system. Learn++ generates 3 classifiers per ensemble and the base classifiers are single hidden layer multilayer perceptrons as in the original literature. The number of hidden neurons used is 30. Rule-plus-exemplar (R+E) parameters are set to $p_c = p_e = 0.1$, $p_\sigma = 90$, $p_\tau = 2$, $\xi = 0.7$, $m_b = 120$, $m_e = 5$, $m_c = 50$. The exemplar subsystem is implemented with kNN using $k = 3$.

The implementation of incrSVM in a dynamic environment gives rise to certain issues. It is computationally infeasible to perform retraining for every sample encountered, so retraining must be performed in batches. Similarly, Learn++ requires a batch of data on which an ensemble is created. The batch

size in both cases will be set to the same value as the episodic buffer capacity parameter m_b in rule-plus-exemplar system for better comparison.

To evaluate the performances of various classifiers in the concept growth environment, the incremental validation (Delany, Cunningham et al. 2006) setup is adopted. Every test sample is classified and evaluated before it is learnt. The error rates on the test data are shown in Table 5.1. Each error rate is averaged over 50 iterations with a new concept growth simulation.

Table 5.1 Comparison of error rates for incremental validation under concept growth

Datasets	FAM	IncrSVM	Learn++	R+E
D1	29.24%	27.93%	31.41%	17.18%
D2	37.22%	35.42%	46.43%	31.08%
D3	26.00%	34.07%	23.32%	16.60%
D4	35.81%	44.23%	38.08%	21.13%
D5	12.54%	18.81%	7.88%	5.03%
D6	26.51%	20.39%	43.83%	12.68%
D7	39.29%	35.91%	37.03%	30.66%
D8	35.90%	46.80%	43.17%	24.05%
D9	34.86%	29.11%	33.99%	25.36%
D10	15.67%	32.41%	15.87%	5.77%
D11	42.62%	25.32%	41.73%	30.83%

It can be seen from Table 5.1 that our proposed rule-plus-exemplar system provides the best performance for most of the datasets. Test samples that are novel and unfamiliar can be better handled by the system through correct

identification and classification by the exemplar subsystem. Even though Learn++ also considers the problem of classifying novel samples, it deals with them by increasing the weights of the more suitable existing base classifier. Along with incrSVM and FAM, the results of these classifiers show that even though they may sometimes provide accurate classification for the unfamiliar test samples, the predictions are not always reliable and consistent in performance.

5.3.3 Analysis

The stability-plasticity dilemma has been a persistent issue to consider in methods with incremental learning abilities. While the rule-plus-exemplar system has demonstrated excellent plasticity in adapting its representation to learn new information, it is natural to consider whether stability has to be sacrificed significantly in this trade-off relationship. Although the testing data in our setup also includes samples that have been represented in the training data, the effect on stability can be more clearly seen with additional tests.

Following common evaluation methods for incremental learning techniques, 10-fold cross validation is used for testing the classifiers. The training data are learnt incrementally, and the testing data are presented at the end of learning. As the datasets are randomly shuffled, the testing data can be better represented by the training data. This will allow clearer evaluation of the stability aspect.

Table 5.2 shows the classification error rates in the described setup. Firstly, the comparison between the two tables shows that the error rates in the concept growth setup were indeed higher than the cross-validation setup. The error rates were also more erratic when there is concept growth, due to the lack of concept

representation by the available training data. Secondly, it can be seen that while the rule-plus-exemplar classification system does not always outperform all the other classifiers in all datasets, its performance is more robust.

Table 5.2 Comparison of error rates for incremental learning in concept growth using 10-fold cross validation

Datasets	FAM	IncrSVM	Learn++	R+E
D1	31.31%	14.92%	30.66%	14.35%
D2	32.25%	28.62%	47.72%	32.36%
D3	32.09%	14.17%	14.68%	14.97%
D4	34.08%	20.97%	43.21%	17.90%
D5	14.86%	5.48%	2.18%	3.09%
D6	26.60%	23.40%	14.79%	15.43%
D7	35.01%	32.89%	25.19%	27.25%
D8	35.12%	47.32%	27.34%	21.07%
D9	39.78%	23.41%	21.82%	23.73%
D10	7.84%	7.84%	14.58%	3.57%
D11	39.89%	24.47%	30.90%	22.35%

Due to the large differences in classifier performances, statistical tests such as the Friedman test (Demsar 2006) will not reflect the robustness of any method. This is because only the ranking in performance is considered but not the difference in performances. Observing directly on the error rates, rule-plus-exemplar system has the best performance for 5 of the datasets, and for most of the remaining datasets, its error rates are not too far from the lowest rates

achieved. This shows that the system maintains a good balance of plasticity and stability despite its ability to learn incrementally.

The stability of the rule-plus-exemplar system can be attributed to the measures taken to limit the overlap in rule module activation fields and maintaining consistency across the overlapping classifiers. This helps to prevent forgetting and catastrophic interference, which is common to incremental learning techniques that are highly plastic. The growing structure of such techniques leads to a tendency of previous knowledge being overwritten with newly acquired ones. By limiting the overlap and keeping the learnt knowledge consistent, this problem is controlled.

Even though Learn++ outperforms rule-plus-exemplar under the cross validation setting for 4 of the datasets, this performance comes at the cost of higher computational effort. This is because Learn++ generates an ensemble for every window of data samples, while rule-plus-exemplar generates classifiers from the buffer only when required. Classification only involves the activated classifiers in rule-plus-exemplar, whereas all the classifiers in Learn++ need to be considered before their output is weighted.

Computational complexity in rule-plus-exemplar depends on the nature of the dataset and how much concept growth there is, but the time taken is directly related to the number of classifiers generated. It determines the number of activation field checks and subsequent classifications that need to be performed. The storage required also arises from the number of classifiers generated, in addition to the samples stored for exemplar classification.

Table 5.3 Computational time and stored rules and exemplars for rule-plus-exemplar system for concept growth

Dataset	Data size	No. of rule modules generated	Percentage of exemplars stored	CPU time for R+E (seconds)	CPU time for Learn++ (seconds)
D1	690	10.18	11.02%	1.16	8.79
D2	279	3.66	9.72%	0.30	7.10
D3	653	9.14	10.89%	1.00	7.31
D4	270	2.90	8.83%	0.19	2.79
D5	1536	15.40	9.49%	2.90	15.80
D6	195	2.02	8.89%	0.10	4.37
D7	768	12.00	10.96%	1.99	7.25
D8	208	2.04	8.06%	0.12	16.40
D9	748	12.00	11.16%	1.79	7.71
D10	569	4.00	8.43%	0.43	10.63
D11	194	2.00	9.29%	0.11	6.48

Table 5.3 shows the number of classifiers generated by the rule-plus-exemplar system under the incremental validation setting as used in Table 5.1. The percentage of samples stored for exemplar classification is also shown, and this depends on the parameter p_e which determines the number of samples from the episodic buffer that will be remembered in episodic LTM. The size of each dataset is provided as a basis for comparison. The average CPU time (obtained

using MATLAB profiler) for each iteration of incremental validation is shown for both rule-plus-exemplar and Learn++.

It can be seen that despite the different number of samples in each dataset, the number of rule modules generated are within less than 2% of the datasize, while the number of exemplars are maintained around 11%. These numbers can be further reduced by adjusting the parameters of the system.

Compared to incremental SVM which maintains only one classifier throughout learning, the rule-plus exemplar system stores a number of classifiers. However, not all of them need to be used for classification. It is simply infeasible to update and retrain the classifier in incremental SVM for every sample presented in a dynamic environment. Yet there are no measures in place for the classifier to take care of unfamiliar samples between updates. This problem is not obvious in the cross validation setup because the training data are fully presented and covers the concept well. However, it becomes apparent in the concept growth setting as shown in Table 5.1 where the performance of incremental SVM is significantly poorer than the rule-plus-exemplar system.

Application on vowel recognition data

The previous experiments and results show that the rule-plus-exemplar classification system is able to deal with concept growth by maintaining a balance between plasticity and stability. This section applies the system on real data to demonstrate the relevance and practicality of the proposed method. The dataset used here is from the Vowel Recognition data in the UCI Machine Learning Repository. There are 15 speakers providing vowel utterances and the system distinguishes between “hid” and “hId”. It is selected because it reflects

the concept growth problem. Utterances from different speakers represent change which needs to be learnt and adapted. At the same time, previous speakers should still be recognized rather than forgotten.

Table 5.4 Error rates on actual vowel data

	FAM	IncrSVM	Learn++	R+E
Incremental validation error	12.38%	20.44%	38.91%	15.84%
Hold-out dataset error	50.10%	44.00%	30.47%	8.33%

The parameter values used are the same as before except $m_b = 20$ and $m_c = 15$, as each speaker only provides 12 utterances. The setup consists of both incremental validation and testing on a hold-out dataset. Two random utterances from each speaker are set aside in the hold-out dataset. Among the remaining samples, 3 random speakers are selected as the initial training set, and the others are presented for incremental validation one speaker at a time. Finally, the resulting system is tested on the hold-out dataset to determine if the system can still classify the utterances from previous speakers. Results are averaged over 100 iterations.

Table 5.4 shows that the rule-plus-exemplar system is able to classify the new vowel utterances from different speakers well, and at the same time retain this knowledge so that it can still classify correctly when the speakers are encountered again. Even though the incremental validation error of FAM is lower than rule-plus-exemplar, the adaptation provided by the plasticity

outweighs its stability and classification on previously seen speakers is drastically affected.

5.4 Chapter summary and concluding remarks

This chapter explained the significance of the concept growth problem and its difference from the well-known concept drift problem. As it cannot be handled by concept drift measures or incremental learning techniques, we proposed a cognitively inspired rule-plus-exemplar classification system. Through a growing number of rule modules, the system can learn incrementally while retaining previously learnt knowledge through the use of activation fields. Its exemplar subsystem enables reliable classification of novel test samples that result from a growth in concept. There is no assumption or restriction on the direction of growth or the speed of growth.

While the overlap in activation fields has been limited and there are measures in place to ensure better consistency across rule modules, possible improvements can be considered. The activation field of a rule module can be shrunk when the classifier makes a wrong prediction. This allows the system to better cope with underperforming rule modules without creating another module. Pruning of rule modules may also be possible with the inclusion of misclassification history of each module. A growing concept inevitably requires a more complex system, but additional measures can be placed to ensure that rules are not generated when new samples no longer reflect a growth in concept. Rule module updates or improvements can be incorporated instead to refine the classifiers with additional information. Although the number of samples stored in LTM is governed by the selected parameter, further reductions are possible

through redundancy checks. Fewer samples which are already within the activation fields of other rule modules need to be stored, as compared to those that are lying in new unrepresented feature space.

The rule-plus-exemplar classification system proposed in this chapter for concept growth is a general architecture that offers flexibility. The rule modules have been implemented using SVM classifiers here, but any other classifier that is not instance based can also be used. Multi-class data can be handled when implementing the rule subsystem with a classifier supporting such classification. The modular nature of the architecture also allows other techniques such as feature selection to be used in conjunction with the rule or exemplar subsystems.

Depending on the classification method used for each rule module, further efficiency improvements are possible. For example, the use of SVM classifiers here can be taken advantage of by considering the support vectors as part of the episodic LTM since the samples must be stored anyway. Incremental classifiers can also be implemented for each rule module to allow for improvements to the existing rule modules. However, these modifications are not yet included in order to allow the validation of the overall system architecture. The current implementation already shows much improved accuracy over other methods, and much better reliability. With further adjustments to the parameters and other aspects of the system, it can be applied in a wide range of problems.

Chapter Six

Rule-plus-exemplar system for imbalanced classes undergoing concept growth

In the previous chapter, we have presented a system based on the rule-plus-exemplar architecture which can deal with concept growth. Learning is carried out continually through the generation of rule modules when necessary, and maintaining a limited number of new samples in between rule generations. Classification is performed using rule or exemplar categorization, which may be implemented using different classifiers. However, when there is a class imbalance in the dataset, a bias may arise in the classification. This is because the problem of imbalanced classes that plagues the underlying base classifier for the rule modules manifests itself in the overall system. Yet applications that face concept growth are also prone to imbalance in classes, such as detection of

network intrusions, fraud cases, oil spill detection or product defects. Samples from the general class are more widely available than the interesting cases that need to be flagged. Even though the previously presented system may face difficulty under conditions of class imbalance, the idea of a rule-plus-exemplar structure remains valid. This chapter presents a completely different way of implementing the rule-plus-exemplar architecture in a system that can deal with imbalanced classes in concept growth. Part of the work in this chapter has been published in (Sit and Mao 2013b).

6.1 Imbalanced classes in concept growth

This section discusses the problem of imbalanced classes in the presence of concept growth. While there are techniques available for classifying imbalanced data, they cannot be effectively applied in a concept growth environment. On the other hand, the presence of imbalanced classes complicates the problem of concept growth, subjecting the previous system to potential difficulties in handling the minority class.

6.1.1 Problem description

The problem of concept growth can be summarized into two aspects at this stage. First is the need to incrementally learn the concept as it grows and new data are made available. The previously learnt knowledge remains relevant, unlike in concept drift, and thus cannot be discarded. This places large demands on storage requirements. Second is the classification of new samples in a dynamic environment. Given any sample which may represent new growth in the concept, the classification must be carried out with the best of current

knowledge. This means that the newest training samples should be taken into consideration even if abstraction has not been performed yet. This also means that the classifier should extrapolate its knowledge as accurately as possible using the most relevant knowledge.

With the introduction of the imbalance in classes, the standard problems affecting static applications arise. Classification tends to be biased towards the well-represented majority class, and evaluation of classification performance should take into consideration the fewer number but potentially more important minority class samples. However, other implications also arise because of the concept growth setting, which makes this problem difficult to solve. In the following, we will refer to the majority class samples as ‘-’ and the minority class samples as ‘+’.

6.1.2 Difficulty of applying techniques for imbalanced data in a concept growth environment

In a dynamic environment, learning is conducted based on limited training data at any time, be it a window of samples or a single new sample. This is because the dynamic nature of the application does not allow the storage of all the previous samples, and the number of samples that can be revisited is very limited. Figure 6.1 illustrates concept growth with imbalanced class distributions. The first three plots show the data distribution of minority class samples ‘+’ and majority class samples ‘•’ at intermediate time steps. Samples given at the current time step are depicted by larger symbols in darker shade, while those already encountered in previous time steps are shown in smaller

symbols in lighter shade. The concept can grow in any direction, so in time step t_2 and t_3 , the input spaces covered by the samples are not compact.

Figure 6.1 Intermediate snapshots of imbalanced data under concept growth

6.1.3 Difficulty to overcoming concept growth introduced by imbalanced classes

At any time, the training set from which abstraction can be performed is always limited. Due to the low incidence of minority samples, the training set may not always contain samples from that class. In the generation of new classifiers for ensembles, the minority class is sometimes completely unrepresented. Even if training of the classifier is delayed until a '+' sample

appears or when the minority class is more adequately represented, there is a limit to how many ‘-’ samples can be stored in the meantime. Some will have to be discarded, without the knowledge of where the next ‘+’ sample may be. But the removed ‘-’ sample may turn out to be important with respect to a future ‘+’ sample, due to its proximity or its significance in determining the correct decision boundary. This is in addition to the intrinsic bias in the generated classifier caused by the imbalance in classes.

The class imbalance makes it difficult to learn in dynamic environments, while concept growth also compounds the class imbalance problem. Even though sampling or cost-sensitive learning techniques can be employed in static environments to compensate the bias towards the majority class, they are not easily applicable under concept growth conditions. Consider the SMOTE technique which generates synthetic ‘+’ samples. Figure 6.2(a) shows the initial training data that is available for generating a classifier. The ‘+’ samples are denoted by a cross and ‘-’ samples denoted by circles. By applying SMOTE, some synthetic ‘+’ samples are generated, depicted by the triangles in the diagram. Figure 6.2(b) shows the full data distribution when all the samples have been presented. Due to the inadequate state of information in Figure 6.2(a), the synthetically generated ‘+’ samples are incorrect, and contradict the actual class distribution. The overlay of the synthetic samples in Figure 6.2(b) show that the resulting classifier trained on inadequate data and incorrect synthetic samples will misclassify the testing data. This understanding extends beyond SMOTE to other data generation techniques. It also extends to cost-sensitive learning as well since the classifier also adjusts its decision boundary to reduce

the bias, but this adjustment may be inaccurate when the concept has not yet been fully presented.

(a)

(b)

Figure 6.2 Inconsistency arising from SMOTE under concept growth conditions

There are a number of techniques that are designed to handle imbalanced classes in concept drift. One technique is to create an ensemble of classifiers from current as well as previous training data (Gao, Fan et al. 2007). The ‘-’ samples are sampled from the current available data, while all the ‘+’ samples are accumulated and used during training. The ensemble of classifiers is built from the current data and will be replaced by a new ensemble when a new set of

data is available. Alternatively, all the '+' samples can be stored but only the most relevant ones with the current '-' samples are used for training (Chen and He 2009; Chen and He 2011). A single classifier or an ensemble can be created, and replaced whenever new data becomes available. As these methods are designed for concept drift, there is no mechanism for determining which '-' samples to retain and store. There is also no consideration on how to classify samples before an update to the classification is provided.

The Learn++ family of algorithms consists of many methods designed for use in a concept drift environment, some of which retain previous knowledge and also consider the problem of imbalanced classes, such as (Muhlbaier, Topalis et al. 2004). It creates ensembles for the current data set, but also retains all previously created ensembles. Their results are combined by weights that are adjusted according to their class conditional probabilities as well as their performance on the current data set. Learn++.SMOTE (Ditzler, Polikar et al. 2010) considers the class imbalance problem by using SMOTE to generate synthetic '+' samples for each classifier created. Beyond the Learn++ family, measures in (Lichtenwalter and Chawla 2010) also consider imbalanced classes in drifting data streams. While previous information is retained for these methods, they are still evaluated based on the current data set, which is assumed to be most relevant to the concept.

6.2 Using ideas from cognitive psychology

In contrast to the difficulty of classifying imbalanced data in a concept growth scenario, humans are able to deal with such situations with far more ease. Given limited experience, we are able generalize the concept to make

reasonable judgment and informative guesses on new instances. The concept is continually updated with new information, and adjusted or refined when it is wrong. In distinguishing between two imbalanced categories, we are able to form a concept of the general case corresponding to the majority class. When there are exception cases to the general concept, these cases corresponding to the minority class can still be remembered and distinguished from the general case despite the overwhelmingly large difference in numbers. These characteristics of the categorization process are relevant to the problem and are summarized below.

- Instances from the general case can be readily recognized if they are sufficiently dissimilar to exception cases
- Although not all the encountered instances can be remembered, they invoke a sense of familiarity
- Some instances can be recollected explicitly, especially exceptions that are special cases of the concept
- When an instance matches the template of an exception case, that case will be activated
- Instances that are similar to exception cases and other samples give rise to ambiguity which require further contemplation before decision making
- Unfamiliar instances that are novel and dissimilar to any encountered instances also require further contemplation

The mechanisms by which humans categorize objects under such situations give insight into how the classifier can adapt to concept growth in the presence

of imbalanced classes. We consider the various difficulties that arise from the problem.

The first difficulty caused by concept growth is that of data storage. Not all the samples can be stored for classification or abstraction. During categorization, humans recognize the general cases through familiarity rather than recollection of all the instances. They are compacted or summarized for efficient representation, such as using a rule or prototype that can describe them. Exception cases are better remembered because they are different and are special cases. Likewise, the pattern classification system does not need to store all the ‘-’ samples. These majority class samples can be described using a higher level rule, and the familiarity can be realized through identification of a familiar region, or the represented region. On the other hand, ‘+’ samples representing the exception cases can be stored.

The second difficulty arising from concept growth is the classification of test samples which reflect a growth in concept. Unfamiliar instances that are unlike any exception or general cases need to be contemplated by considering both possibilities. In classification, when the test sample falls outside the training region, it is no longer familiar. As shown in Chapter 4, these samples are better classified by considering whether they are more similar to the majority or minority class.

The third difficulty of the problem arises from the imbalance in classes. Exceptions are not neglected in human cognition; rather, they are better remembered and can be activated for comparisons with the general case when a query instance is sufficiently similar to it. To learn the imbalanced classes,

minority samples can be stored along with nearby similar majority class samples, since this is when further contemplation is needed.

Therefore the rule-plus-exemplar structure is again applicable in this problem. The general case can be represented using rules, while the exception cases can be represented using exemplars, with special consideration for distinguishing between the two classes. Section 6.3 elaborates on how this structure is implemented.

6.3 Rule-plus-exemplar classification system

The rule-plus-exemplar classification system is implemented based on partitions of the input feature space. A brief process is depicted in Figure 6.3 which describes the specialization of each region. The first region is the ambiguous region, where a query sample is similar to both minority and majority class samples, leading to difficulty in classification. The second region is the familiar region, which has been represented by training samples encountered to date. This region includes samples from both classes. As the region has been represented, there is knowledge on how to classify a query sample, depending on whether it falls in the ambiguous region. Finally, the remaining space belongs to the unfamiliar region, representing the space which has not yet been represented by training samples, and also not close enough to the minority class samples to fall in the ambiguous region. The regions are represented as rules.

Classification of samples depends on the region which they lie in. A sample in the ambiguous region is classified according to a trained classifier. Otherwise, if it lies in the familiar region but not the ambiguous region, there is little

ambiguity and the sample is simply classified as the majority class. Any other sample that does not lie in either region is falling in the unfamiliar region, which means it is outside the training region. As explained in Chapter 4, they can be classified by exemplars. The rule-plus-exemplar system thus consists of regions defined as rules, as well as the classifier for the ambiguous region represented as a high level rule. The exemplars are used for classification of unfamiliar samples. The following sections will elaborate on the function of each region and how they are identified.

Figure 6.3 Specialization of each identified partition

6.3.1 Ambiguous region, R_A

The ambiguous region defines where classification is most likely to be difficult because samples are close to both classes. Due to the complexity of the concept here, a classifier is trained specifically for this region. As there is an imbalance in classes, this region is defined in such a way that limits the number of majority class samples. The description of the region is similar to that of the training region in Chapter 4, but is modified to focus on the minority class samples.

Identification of the ambiguous region

Given a training dataset $\{s_1, \dots, s_m\}$, partition into the majority and minority class samples given by $\{s_1^-, \dots, s_n^-\}$ and $\{s_1^+, \dots, s_p^+\}$ respectively. Gaussian functions will be centered on each minority class sample s_i^+ with different spread parameters σ_i for each function. The evaluation function for defining the ambiguous region can be represented by

$$T_A(x) = \frac{1}{p} \sum_{i=1}^p \exp\left(\frac{-\|x - s_i^+\|^2}{2\sigma_i^2}\right). \quad (6.1)$$

The spread parameters are determined from the ‘-’ samples nearest to the function centers. For any ‘+’ sample s_i^+ , find the nearest d samples $\{s_{i_1}^-, \dots, s_{i_d}^-\}$ at that point in time, where

$$\max_{j=1, \dots, d} \{\|s_{i_j}^- - s_i^+\|\} \leq \min_{\substack{j=1, \dots, n \\ j \neq i_1, \dots, i_d}} \{\|s_j^- - s_i^+\|\}. \quad (6.2)$$

The corresponding spread parameter σ_i is chosen such that the Gaussian function will cover the nearby majority class samples. It can then be computed as

$$\sigma_i = \frac{1}{d} \sum_{j=1}^d \|s_{ij}^- - s_i^+\|. \quad (6.3)$$

The function value of $T_A(x)$ is then computed for all the majority ‘−’ samples to obtain $\{T_A(s_1^-), \dots, T_A(s_n^-)\}$. These values are sorted in descending order to obtain $\{t_1, \dots, t_n\}$ such that $t_1 \geq \dots \geq t_n$. The threshold τ_A for defining the ambiguous region can be selected according to how many majority class samples are allowed. We define a target ratio ρ to specify the desired ratio of ‘+’ samples to ‘−’ samples. The threshold for $T_A(x)$ can then be calculated by

$$\tau_A = t_{\min\{\frac{p}{\rho}n\}}. \quad (6.4)$$

The ambiguous region R_A can then be explicitly defined as

$$R_A = \{x | T_A(x) \geq \tau_A\}. \quad (6.5)$$

Storage of exemplars in the ambiguous region

Recall that in cognitive psychology, the exception cases are easily remembered. Even though the general cases are too many and not all remembered, there is a sense of familiarity. Given the appropriate cues, some of these samples may be retrieved. Due to the importance and the limited availability of minority class samples which correspond to the exception cases,

they are all retained for reference. While it is impossible to store all the majority class samples, the ones which are sufficiently close to the minority class samples can be stored. This corresponds to future retrieval of the samples where the minority class samples serve as cues. Thus, the samples of both classes that are covered by the ambiguous region will be stored.

Update of the ambiguous region

New samples presented to the system will be stored in the buffer. When abstraction is due, the ambiguous region must be updated regardless of whether there are new minority class samples. In addition, it should also be updated upon presentation of a new minority class sample because this has significant influence on the region and the classification. The ambiguous region is determined from the samples in the new dataset, as well as the previous samples stored for this region. The new ambiguous region is identified using Equations (6.1) to (6.5), with Gaussian functions centering on all the minority class samples that have been presented so far. The exemplars to be stored will also be updated based on this new definition.

6.3.2 Familiar region R_F

Intuitively, the familiar region refers to the input feature space which has been covered by the training data already. It represents samples which are similar enough to the previous experience. Functionally, identifying the familiar region allows us to determine how to classify a query sample. If the sample falls in the familiar region but is not in the ambiguous region, it does not match any known exception case. The sample will thus be classified as the majority class. This familiar region covers all the training samples, which cannot all be

stored due to the high volume of data that will be presented over time. The description of this region must be represented in a simplified way.

Identification of the familiar region

Given a set of training data $\{s_1, \dots, s_m\}$, the reduced representation for describing the training region in Chapter 4 can be used to represent the familiar region. For the given training set, the familiar region is given by

$$R_F = \{x | T_F(x) \geq \tau_F\} \quad (6.6)$$

where the covering function is

$$T_F(x) = \frac{1}{r} \sum_{i=1}^r \exp\left(\frac{-\|x - c_i\|^2}{2\sigma_i^2}\right) \quad (6.7)$$

and the threshold for the covering function is set to the p_τ -th percentile of the T_F values

$$\tau_F(x) = P_{p_\tau}(\{T_F(s_i), i = 1, \dots, m\}). \quad (6.8)$$

The Gaussian centers c_i and corresponding spread parameters σ_i are computed according to the algorithm in Figure 4.8 in Chapter 4. The number of clusters r determines the number of Gaussian function centers in the covering function.

As the familiar region represents the training data that has been encountered to date, the familiar region will be generated from all the samples in a given training set, including the samples for determining the ambiguous region.

However, it may not necessarily include the entire ambiguous region because of the representation of the regions. They are described using different Gaussian function centers and spread parameters, so even if the familiar and ambiguous regions overlap, there may be parts of the ambiguous region that fall outside the familiar region.

Storage of exemplars in the familiar region

Any sample outside both the ambiguous region and familiar region will be deemed to be unfamiliar, and is classified according to exemplar categorization. This requires the storage of a portion of samples to facilitate classification of the unfamiliar samples. The Gaussian function centers c_i which are cluster centers from the training data only serve as prototypes determining whether a sample is familiar. For classification, specific training samples need to be stored. All the minority class samples have already been stored since they occur in the ambiguous region. The exemplar based classification will depend on the majority class samples. Those lying in the ambiguous region have also been stored. The rest in the familiar region should be selectively chosen to facilitate the classification. Two scenarios are considered for this.

Figure 6.4 provides a simplified illustration of the two scenarios with a small number of points shown. In Figure 6.4(a), if the ambiguous region is completely covered within the familiar region, classification of any sample in the unfamiliar region, as denoted by the triangle, depends only on the samples in the familiar region but not the ambiguous region. Samples in this region are denoted by the empty circles and squares. The former represents samples that are lying deeper within the region, far from the unfamiliar samples; the latter

represents samples that are closest to the outer boundary of the familiar region. The exemplar classification of the sample x is more dependent on the outer square samples than the inner circle samples. In Figure 6.4(b), the ambiguous region lies near the outer edge of the familiar region, or part of it may lie outside. For an unfamiliar sample x_1 that is near the familiar region but far from the ambiguous region, exemplar classification depends on samples in the familiar region, particularly those lying on the outer edge as denoted by the squares. For a sample x_2 that lies close to the ambiguous region but not the familiar region, exemplar classification involves samples that are in the ambiguous region, which are already all stored. An unfamiliar sample x_3 that lies close to both regions may be classified according to samples from both regions. Again, the outer samples in the familiar region represented by the squares are more relevant than those inside, while the samples in the ambiguous region which may be from both classes will also be considered.

Therefore, in determining the samples to store for the familiar region, it is not necessary to consider the samples that lie deep within the region, as they have little role in the exemplar based classification of the unfamiliar regions. The outer samples to be stored can be identified by selecting a portion p_s of the samples with the lowest $T_F(x)$ values. The function $T_F(x)$ is computed according to the algorithm in Figure 4.8 for which the contour plot of the region boundary function value has been shown in Figure 4.9.

Discarding the inner samples of the familiar region leads to concerns only when a future minority class sample occurs near to the discarded samples. This undermines the quality of the classifier generated because of the loss of nearby

majority samples. However, a number of majority samples have already been better represented. Consider the situation where the ambiguous region has been updated to cover a larger space but some ‘-’ samples have been discarded. If a ‘-’ sample is misclassified, this can only occur within the ambiguous region, but the misclassified sample will then be used to update the classifier.

(a)

(b)

Figure 6.4 Relevant samples for classification of unfamiliar samples

Update of the familiar region

As new samples are presented to the system, they are stored in a temporary buffer. When the samples accumulate beyond the buffer limit, the familiar region will be extracted from the buffer data. The initial familiar region is given by

$$R_F = \{x | T_F^{(1)}(x) \geq \tau_F^{(1)}\}, \quad (6.9)$$

while the addition of a new rule extracted will be appended to update the familiar region as

$$R_F = \{x | T_F^{(1)}(x) \geq \tau_F^{(1)}, T_F^{(2)}(x) \geq \tau_F^{(2)}\}. \quad (6.10)$$

The new rule $T_F^{(2)}(x) \geq \tau_F^{(2)}$ for the region is computed from the buffer data using Equations (6.7) and (6.8). The buffer data will be cleared hereafter for subsequent samples. New rules that are extracted are appended accordingly.

Before the update of the familiar region and selection of exemplars for long term storage, the samples in the buffer are accessible for exemplar based classification of unfamiliar samples. During this interval, the addition of new majority class samples does not affect the classification of samples in the familiar regions. Even though it may affect the classification in the ambiguous region, the majority class is already better represented, which diminishes this effect.

6.3.3 Unfamiliar region R_U

Rather than an explicit description of the unfamiliar region, it is simply the input feature space beyond the ambiguous or familiar regions. Any sample here is classified according to exemplar based classification, using samples in the temporary buffer as well as all stored samples. This can be implemented using kNN classification as in Chapter 4, which has been shown to provide more reliable classification outside the training region as represented by the familiar region. The influence of class imbalance is also reduced because kNN is a local classifier that is less prone to bias than a global classifier.

6.3.4 Rationale behind the rule-plus-exemplar system for imbalanced classes

There are three aspects to the problem of imbalanced classes under concept growth. The classification system proposed can handle these aspects of the problem through the use of rules and exemplars, as well as the separate regions.

Firstly, learning under concept growth conditions is enabled through retraining of the classifier within the ambiguous region and expansion of the familiar region. Retraining is triggered when there is a new minority class sample, or when there is an update to the ambiguous region from new training samples in the buffer. The familiar region expands with the new samples that are progressively made available, represented with rules that enrich the description of the region. Retraining is made possible within the ambiguous region through the samples that are stored only for this region.

Secondly, classification of new samples that have undergone concept growth requires updated information. While the frequency of updates is limited, it can be customized according to the region. As new minority class samples in the ambiguous region represent a significant enrichment in the concept, updates are immediate to facilitate better classification of samples in this region. Otherwise, the familiar region is less sensitive to addition of new majority class samples, so updates can be buffered and delayed slightly. The unfamiliar region where new samples may lie after a growth in concept requires the most updated and relevant information, so classification is carried out based on exemplars which include the newest samples.

Finally, the imbalance in classes is handled through the specialized ambiguous region. It focuses on the exception cases of interest, which are the minority class samples, and takes into consideration the most relevant majority class samples that will shape the learnt concept. The ambiguous region is selected in a way that maintains a better balance of classes.

The mechanisms through which the system deals with the problem are summarized in the next section.

6.3.5 Overall classification system and process

Figure 6.5 Overall rule-plus-exemplar system and process for imbalanced classes in concept growth

Figure 6.5 shows the overall system for dealing with imbalanced classes under concept growth conditions. The system is made up of rules and exemplars. The descriptions for the ambiguous and familiar regions are in the form of rules, while the classifier in the ambiguous region represents a high level rule. Any eager learning classifier can be used here, although experiments in this chapter will implement the classifier using SVM. Exemplars in the buffer are stored for short durations while they accumulate, while a selected

portion of exemplars from the familiar region will be stored long term. Minority class samples are retained and frequently used for training and updates in the ambiguous region, while the majority class samples may change and be replaced over time.

The initial training process is as follows:

1. Identify ambiguous region R_A by finding T_A and τ_A
2. Find samples in R_A to store and generate classifier
3. Identify familiar region R_F by finding $T_F^{(1)}$ and $\tau_F^{(1)}$
4. Store a portion p_s of the R_F samples with lowest $T_F^{(1)}$ values and discard the rest

The classification of a sample x and its evaluation process is as follows:

1. If $T_A(x) \geq \tau_A$, sample is in R_A and will be classified by the classifier
2. Otherwise if $T_F^{(i)}(x) \geq \tau_F^{(i)}$ for some i , sample is in R_F and is predicted as majority class
3. Otherwise sample is in R_U and will be classified according to kNN using buffer samples, R_A samples and stored samples from R_F
4. Check the prediction against the actual class y when available for evaluation
5. Store (x, y) pair in the buffer

Updates are carried out under two conditions:

1. If the buffer is not exceeding maximum size but $y = \text{minority class}$
 - a. Redefine R_A and update samples in R_A
 - b. Retrain classifier for R_A

2. If the buffer exceeds maximum size
 - a. Redefine R_A and update samples in R_A
 - b. Retrain classifier for R_A
 - c. Find new $T_F^{(i)}$ and $\tau_F^{(i)}$
 - d. Append new rules to description of R_F
 - e. Store a portion p_s of current buffer samples
 - f. Discard the remaining buffer samples to clear the buffer

6.4 Experiments and results

This section presents a number of experiments on imbalanced data in a concept growth environment. Data satisfying such conditions are required, which will be detailed. The overall classification system will be compared with an existing method that is best able to deal with the described problem. An analysis then provides verification of the ambiguous, familiar and unfamiliar regions.

6.4.1 Experimental data

The data on which the system should be tested should reflect the problem of imbalanced classes under concept growth. Such data should have the following characteristics:

1. Data stream – The data for classification and subsequent learning should be presented in a data stream to demonstrate the dynamic aspect of the application environment. Incremental validation is carried out, by presenting one sample at a time for classification, after which its actual class will be made available for evaluation and learning. The order of

presentation of the data samples is only for reflecting the growth in concept, but no assumption should be made on the direction of growth. Time information should not be used.

2. Imbalanced classes – The availability of one class should be significantly lower than the other, such that the minority class may not be encountered in the data stream for long intervals. The initial set may not necessarily consist of the minority class, which is consistent with learning in humans.
3. Concept growth – The presentation of samples should be ordered such that the testing samples may fall inside or outside the training region thus far. Samples inside test the system's stability in retaining the learnt knowledge, while those outside test the system's ability to extend its knowledge to new cases.

While datasets with class imbalance are available, those that also reflect concept growth are not as publicly accessible despite their presence in many applications. Concept growth is thus simulated on a number of imbalanced datasets. Unlike the algorithm in Figure 5.5, special care needs to be taken to ensure that the minority class is also participating in the concept growth. If all the minority class samples are presented at the same time, it undermines the difficulty of the problem.

Figure 6.6 shows the algorithm for generating concept growth in an imbalanced dataset. The training set consists of a core set of samples, where testing samples will fall progressively further outside this set. As the samples appended to Dat_{new} are randomly selected and ordered, which also influences

subsequent iterations in Step 4, each run of the algorithm produces a differently ordered testing set.

Algorithm for generating concept growth for imbalanced classes

Input: Dataset $Dat = \{s_1, \dots, s_m\}$

Initialize: empty dataset $Dat_{new} = \emptyset$, segment size $s = \lfloor 0.2m \rfloor$

1. Calculate $T(x)$ values for all samples using full representation from Chapter 4
2. Separate dataset into minority and majority class $Dat = Dat_+ \cup Dat_-$
3. Separate set of $T(x)$ values into minority and majority classes Tx_+ and Tx_-
4. **For** $i = 20, 40, \dots, 100$
 - a. $threshold = \min\{P_i(Tx_+), P_i(Tx_-)\}$
 - b. Find at most s randomly selected samples from Dat such that $T(x) \geq threshold$
 - c. Append samples to Dat_{new}
 - d. Remove samples from Dat

End

5. New dataset $Dat_{new} = \{s_1^{(new)}, \dots, s_m^{(new)}\}$
6. Training set $trDat = \{s_1^{(new)}, \dots, s_s^{(new)}\}$
7. Testing set $teDat = \{s_{s+1}^{(new)}, \dots, s_m^{(new)}\}$

Output: Training set $trDat$ and ordered testing set $teDat$

Figure 6.6 Algorithm for generating concept growth for imbalanced classes

Figure 6.7 Artificially generated 2-dimensional imbalanced data

Table 6.1 Details of imbalanced data used

	Number of features	No. of minority class samples	No. of majority class samples
D1	2	60	600
D2	2	60	600
D3	2	30	600
D4	2	60	600
D5	22	48	147
D6	44	55	212
D7	4	178	570
D8	33	46	148

The algorithm is operated on two groups of datasets. The first group D1 to D4 consists of artificially generated 2-dimensional data as shown in Figure 6.7. The second group D5 to D8 consists of datasets from the UCI Machine Learning Repository – Parkinsons, Spectf, Transfusion and WPBC datasets respectively. Details of the datasets are given in Table 6.1.

6.4.2 Experimental setup and results

Although most incremental learning techniques or methods for concept drift are unable to deal with the problem of imbalanced classes in concept growth, some are better positioned to handle this problem. Of relevance and interest is Learn++.SMOTE, which considers the imbalance classes under concept change conditions. It generates an ensemble of classifiers, retaining previously generated classifiers whose results are combined through adjusted weights.

Both Learn++.SMOTE and the proposed rule-plus-exemplar system (R+E) for imbalanced classes will be trained on an initial training set $trDat$. Subsequent samples from $teDat$ will be presented one at a time for incremental validation. Each sample is classified before the actual class label is provided for evaluation. This can provide the number of misclassifications, but the accuracy measure is not a meaningful evaluation for imbalanced classes. Therefore, the number of true and false positives and negatives will be provided instead. In addition, the Matthews Correlation Coefficient (MCC) will be shown. As shown in Chapter 2, the values lie between -1 and 1, with 1 being the best possible value. Due to the absence of a threshold value that can directly produce a tradeoff in classification of positive or negative class samples, results cannot be presented using ROC curves or AUC.

The parameters for both methods are set as follows. The window size in Learn++.SMOTE is set to 300, which is also the maximum size for the buffer in R+E. Learn++.SMOTE parameters are chosen as $a = 0.5$, $b = 15$, its SMOTE implementation $k = 3$, $N = 1500$, base classifiers consisting of MLP with 25 hidden layer nodes. This is in accordance with the implementation in the original literature. R+E target ratio is set to $\rho = 0.6$, parameter $d = 3$ for calculating spread of R_A , and parameters $r = \lfloor 0.1M \rfloor$, $p_\tau = 5$, $n_\sigma = 3$, $p_c = 90$, $p_s = 0.1$ for identification and exemplar storage in R_F . The classifier for R_A in the rule subsystem is implemented using RBF kernel SVM with cost parameter $c = 5$, and the kNN for exemplar based classification with $k = 3$. Due to the dynamic nature of the experiment, where there is no knowledge on how the testing data would differ from the training data, it is not practical to perform scanning or optimization of the parameter values. The same values are thus

applied across the various experimental datasets, which tests the robustness of the methods. Incremental validation results shown in Table 6.2 and Table 6.3 are averaged over 100 iterations, each with a new generation of data using the algorithm in Figure 6.6.

Table 6.2 Comparison of MCC values on imbalanced data under concept growth

Datasets	MCC	
	Learn++.SMOTE	R+E
D1	0.4657	0.6788
D2	0.3744	0.5459
D3	0.503	0.7138
D4	0.5284	0.7424
D5	0.074	0.6935
D6	0.2153	0.2068
D7	-0.0219	0.1867
D8	0.0933	0.1329

Classification of samples with imbalanced classes is prone to a tradeoff between the number of TP and TN correctly identified. The ideal balance between the two depends on the application so it is difficult to pass judgment in many cases. However, the MCC gives a reasonably balanced measure, while the breakdown of TP, FN, TN and FP is not application-specific. The MCC achieved for R+E is better than that of Learn++.SMOTE for all the datasets except D6. Furthermore, for datasets D1, D3 and D5, the number of TP and TN identified by R+E are more than Learn++.SMOTE. Even though R+E identifies

fewer TN for D7, it correctly identifies a significantly larger number of TP. Datasets D2, D4 and D8 show comparable number of TP identified by R+E but more TN.

Table 6.3 Comparison of classification of imbalanced classes under concept growth

	Learn++.SMOTE				R+E			
Datasets	TP	FN	TN	FP	TP	FN	TN	FP
D1	42.04	12.73	400.3	72.93	46.87	9.71	442.22	29.2
D2	32.12	23.47	413.94	58.47	30.72	25.48	454.01	17.79
D3	18.77	9.79	449.01	26.43	20.96	7.97	467.43	7.64
D4	41.67	3.18	407.55	75.6	40.01	5.73	463.28	18.98
D5	16.16	25.91	78.76	35.17	31.18	10.95	106.32	7.55
D6	21.37	15.38	121.11	56.14	15.22	21.51	144.26	33.01
D7	23.24	119.03	372.67	84.06	71.48	70.07	321.88	135.57
D8	18.32	20.59	73.82	43.27	17.41	21.4	81.51	35.68

6.4.3 Analysis

The MCC values show that R+E provides better overall classification of imbalanced data under concept growth conditions. This is dependent on the identification of the regions R_A , R_F and R_U . In Table 6.4, the percentage of test samples in each region is shown. Depending on the rate of concept growth, the proportion in R_U varies, and the difficulty of classifying samples outside the training region depends on the way the concept grows. However, for the regions R_A and R_F which are within the training region, the ambiguous region does

contain a relatively larger proportion of errors for its coverage, compared to the familiar but unambiguous region $R_F \setminus R_A$. It is also more difficult to classify samples in R_A , as the error rates are higher than that in $R_F \setminus R_A$ even though all samples are classified as the majority class there.

Table 6.4 Analysis of regions identified

Data sets	Percentage of test samples in region (%)			Percentage of errors in region (%)			Error rate of region (%)		
	R_A	$R_F \setminus R_A$	R_U	R_A	$R_F \setminus R_A$	R_U	R_A	$R_F \setminus R_A$	R_U
D1	27.77	39.23	32.99	64.28	1.41	34.31	17.06	0.27	7.66
D2	20.49	57.51	22	62.58	7.23	30.18	25.03	1.03	11.24
D3	13.81	58.45	27.73	85.14	2.5	12.36	19.09	0.13	1.38
D4	35.07	21.89	43.04	92.68	0.04	7.28	12.37	0.01	0.79
D5	43.87	9.03	47.1	52.54	8.97	38.49	14.2	11.78	9.69
D6	37.88	2.8	59.31	54.37	0.94	44.7	36.56	8.5	19.2
D7	41.36	6.33	52.31	38.05	3.04	58.91	31.59	16.47	38.66
D8	31.75	6.76	61.49	30.38	4.87	64.75	35.01	26.38	38.53

The difficulty of classifying imbalanced classes is mostly contained within the ambiguous region. Datasets D6, D7 and D8 have the lowest MCC scores, and correspondingly the number of samples in R_A is much larger than $R_F \setminus R_A$ for these datasets. A large R_A indicates that the minority class samples are largely scattered across the familiar region. In higher dimensions, this leads to an underrepresentation of the minority class, thus leading to difficulty in

classification. In addition, these 3 datasets have a high proportion of samples in R_U compared to R_A and R_F . This indicates a faster rate of concept growth which makes classification even more difficult.

6.5 Chapter summary and concluding remarks

In this chapter, the rule-plus-exemplar classification system has been implemented in a new way to cater to classification of data with imbalanced classes, under conditions of concept growth. The system proposed in Chapter 5 cannot be used because the underlying classifiers may be affected by the class imbalance. Furthermore, the long intervals between presentations of minority class samples will result in the generation of many classifiers that only covers one class. On the other hand, techniques for handling imbalanced classes cannot be easily applied because of the growth in concept. There is no knowledge to how the incoming samples may behave, so the bias adjustments and generation of additional samples may contradict the actual class distribution. Furthermore, the stream of data that is presented cannot all be stored, but it is difficult to determine which samples should be retained for later reference.

The proposed rule-plus-exemplar system is relevant because it tailors the rule or exemplar classification according to where a sample lies. By partitioning the input feature space into ambiguous, familiar but unambiguous, and unfamiliar regions, samples can be better handled for both learning and classification. The difficulty of classifying imbalanced data is contained in the ambiguous region, where a high level rule in the form of a classifier learns from a balanced dataset and classifies accordingly. Samples in the familiar region excluding the ambiguous region can be classified with a simple rule that

predicts the majority class. Outside in the unfamiliar region, based on ideas in Chapter 4, the classification of new samples that fall outside the training region can be done using the local classifier kNN for more reliable classification. The results were compared with an ensemble method that is designed for use with imbalanced classes in dynamic environments with concept change. Our proposed rule-plus-exemplar system was able to provide a good balance in the classification of both classes, and further outperformed the existing method by correctly identifying more samples for both classes in a number of datasets.

There are possible improvements that can be made to the system. The system currently employs only a single classifier that will be retrained for the ambiguous region. Using an ensemble of classifiers created from sampling an expanded ambiguous region can possibly improve the classification of the imbalanced data. While this imposes additional computational efficiency, the number of classifiers remains fixed throughout learning because they can be replaced. Another improvement can be to further optimize the representation of the familiar region. The rules created and successively appended to the representation can be simplified. The dataset from which these rules are created can be minimized by removing those which are already covered by previous rules. However, additional investigations are required because excessively removing such samples may result in too few samples available for approximating the region.

The rule-plus-exemplar framework has been successfully applied to concept growth for both situations with and without imbalance in classes. Depending on the application, the right rule-plus-exemplar classification system can be used.

The framework remains applicable and relevant. It is also generic and allows use with various classification techniques.

Chapter Seven

Conclusion

Although machine learning classifiers are able to handle far more complex and larger datasets than humans can, they fall behind in terms of qualities like comprehensibility and adaptability. Humans are able to communicate the process behind which a decision is made, but many classifiers are black-box models which may provide good generalization accuracy but give little or no insight to the process behind the prediction. This limits their use in safety critical applications where they play decision support roles. In addition, humans are able to learn and classify well in a dynamic environment, adapting the model and the categorization as the concept changes and undergoes growth. Classifiers often employ separate training and testing processes, which obstructs quick adaptation to the change in concept. This makes it difficult for the classifier to update fast enough in a dynamic application, or to classify samples that reflect a concept change which renders training data inadequate.

These comprehensible and adaptive qualities are both desirable characteristics that are important in different applications.

To imitate these qualities intrinsic to humans, ideas from cognitive psychology are drawn and incorporated into pattern classifiers. Based on the rule and exemplar categorization models in cognitive psychology, a rule-plus-exemplar structure is proposed. This taps on the complementary nature of the two subsystems, which supplement each other in different shortfalls. Through different architectures, the rule and exemplar categorization solve different problems of interest.

Comprehensible classifiers have long been limited to rule representations, which limit the complexity of the concept that can be modeled. Recognizing that humans use exemplar categorization which is naturally intuitive, instance based lazy learning techniques are incorporated to support the straightforward rules. This improves the generalizability of the classification while retaining its interpretability. Although the tradeoff between comprehensibility and accuracy remains, it is now a more favorable scenario.

Extrapolation of a classifier is difficult because of the inadequate representation of the concept outside the training region. By identifying this region, appropriate measures can be taken for test samples. Eager learning classifiers which perform abstraction like the rule categorization model are utilized within the training region where it has been trained. Outside this region, instance based lazy learning classifiers which behave like exemplar categorization models can be used for intuitive extrapolation. Together, the

system switches between the rule and exemplar subsystems to provide more reliable and consistent classification.

Under concept growth conditions, new information must be learnt while retaining previously abstracted knowledge. Test samples that reflect this growth must be reliably classified to the best of current available knowledge. The rule subsystem generates new rule modules to capture new knowledge in a plastic way, but retains the previous modules which will be activated when required, thus providing stability to the system. Before knowledge can be abstracted, new information is stored as exemplars which provide immediate access for most updated classification of new samples. Where there is imbalance in the data classes, the rule-plus-exemplar system is implemented in a different way. Data regions of different characteristics are identified, to which different classification methods can be applied. The complementary nature of the rule and exemplar subsystems provides specialization in the different regions to classify the imbalanced data.

The rule-plus-exemplar classification systems proposed are able to perform well under the different scenarios, but there are a number of open questions whose answers can provide improvements. Some additional effort may also benefit the systems further.

Identification of the training region is not heavily dependent on the data distribution density or shape, but for a reduced representation, the question of how many clusters to use persists. This question can be traced back to the problem of clustering where the optimal number of clusters is not known beforehand. In addition, outliers that influence the underlying clustering

technique also affect the resulting training region representation. Techniques that are not sensitive to outliers can be used, or outlier removal can be performed first before identifying the training region.

Given the tradeoff between comprehensibility and accuracy, it is inevitable that black box models still outperform the interpretable methods. But the use of exemplars brings them to a far more advantageous position. Nevertheless, the comprehensibility of the classification system is still influenced by the number of rules generated, the number of rule antecedents involved, and number of exemplars used. There is no agreed number that is optimal in terms of interpretability and meaningfulness, and this likely depends on the dataset and application. While the exemplars can much improve the generalizability, they are more easily influenced by the number of features of the dataset compared to rules that have selected the most relevant ones. Feature selection techniques may be applied before exemplar classification, and the techniques used may improve the accuracy as well as facilitate understanding.

Dealing with concept growth in dynamic environments puts the classifiers in the stability-plasticity dilemma. To retain the learnt knowledge while learning new information, the size of the system is prone to growth. This is required to a certain extent in order to adequately represent the enriched concept, but care must be taken to prevent uncontrolled growth. There must be a way to determine whether the new sample represents new information that should be learnt, so that existing rule modules can be better utilized. Where the new knowledge constitutes contradiction to existing abstractions, learning without creating new abstractions can limit the growth in system size. These questions have been answered to a certain extent, but correcting existing

abstractions without generating a new rule module remains a difficult problem because a certain amount of redundancy is required in order to improve the robustness of the classification. Another question that remains open is how to determine when a part of the concept has been well represented enough for abstraction. In the uncertainty of dynamic environments, this can only be estimated based on the limited available information.

Concept growth further compounds the problem of imbalanced classes, which also afflicts many real life applications. The identification of the ambiguous region focuses learning efforts on the more important minority class, by storing their samples and retraining when required. Results have been encouraging but can potentially benefit from the storage of additional majority class samples. Within certain limit on the ratio of majority to minority class samples, a larger representation of the majority class can provide useful information for distinguishing between the classes in the ambiguous region. This is particularly important in higher dimensional space where the minority class becomes even less adequately represented. The number of samples that can be practically maintained for the ambiguous region is dependent on the application and implementation, so appropriate adjustments can further improve the classification. While the experiments did not incorporate any cost information for the classes, if this information is made available, the classifier and the identification of the ambiguous region can further benefit to provide more suitable classification for that application. Nevertheless, the classification framework based on rules and exemplars remain valid and can be used with different classification techniques.

List of publications

- Sit, W. Y. and Mao, K. Z. (2012a). Cognitively inspired classification for adapting to data distribution changes. *IEEE Conf. Evolving and Adaptive Intelligent Systems*, Madrid, 17-18 May, pp. 41-46.
- Sit, W. Y. and Mao, K. Z. (2012b). A cognitively inspired rule-plus-exemplar framework for interpretable pattern classification. *15th International Conference on Information Fusion (FUSION)*, Singapore, 9-12 July, pp. 1188-1195.
- Sit, W. Y. and Mao, K. Z. (2013a). "Cognitively inspired rule-plus-exemplar classification framework for concept growth." *Journal of Machine Learning Research*, under review.
- Sit, W. Y. and Mao, K. Z. (2013b). Learning imbalanced classes in the presence of concept growth. *2013 IEEE Symposium on Evolving and Adaptive Intelligent Systems*, Singapore, 16-19 April, pp. 62-69.
- Sit, W. Y. and Mao, K. Z. (2013c). A rule-plus-exemplar classification system for adapting to concept growth. *2013 IEEE Int. Conf. Cybernetics (CYBCONF)*, Lausanne, 13-15 June, pp. 13-15.

References

- Aha, D., Kibler, D., et al. (1991). "Instance-Based Learning Algorithms." *Machine Learning*(6): 37-66.
- Aha, D. W. (1992). "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms." *International Journal of Man-Machine Studies* 36(2): 267-287.
- Allan, J. (1996). Incremental Relevance Feedback for Information Filtering. *Proc. 19th Annu. Int. ACM SIGIR Conf. Research and development in information retrieval*.
- Allen, S. W. and Brooks, L. R. (1991). "Specializing the Operation of an Explicit Rule." *Journal of Experimental Psychology: General* 120(1): 3-19.
- An, A. and Cercone, N. (1998). ELEM2: A learning system for more accurate classifications. *Advances in Artificial Intelligence*. R. E. Mercer and E. Neufeld, Springer Berlin Heidelberg. 1418: 426-441.
- Andrews, R., Diederich, J., et al. (1995). "Survey and critique of techniques for extracting rules from trained artificial neural networks." *Knowledge-Based Systems* 8(6): 373-389.
- Angelov, P., Filev, D. P., et al. (2010). *Evolving intelligent systems : methodology and applications*. New York John Wiley.
- Atkinson, R. C. and Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes *The psychology of learning and motivation: Advance in research and theory* K. W. Spence and J. T. Spence. New York Academic Press 2: 29-195.
- Baddeley, A. (2000). "The episodic buffer: a new component of working memory?" *Trends in cognitive sciences* 4(11): 417-423.
- Baddeley, A. (2012). "Working Memory: Theories, Models, and Controversies." *Annual Review of Psychology* 63: 1-29.
- Baddeley, A. D. and Hitch, G. (1974). Working Memory. *Psychology of Learning and Motivation*. H. B. Gordon, Academic Press. 8: 47-89.
- Balabanovic, M. (1997). An adaptive Web page recommendation service. *Proc. 1st Int. Conf. Autonomous agents*, Marina del Rey, California, USA, ACM.
- Baldi, P., Brunak, S., et al. (2000). "Assessing the accuracy of prediction algorithms for classification: an overview." *Bioinformatics* 16(5): 412-424.

- Barakat, N. and Diederich, J. (2006). "Eclectic Rule-Extraction from Support Vector Machines." *World Academy of Science, Engineering and Technology* 17.
- Barakat, N. H. and Bradley, A. P. (2007). "Rule Extraction from Support Vector Machines: A Sequential Covering Approach." *IEEE Transactions on Knowledge and Data Engineering* 19(6): 729-741.
- Bartocha, K. and Podolak, I. (2011). Classifier Ensembles for Virtual Concept Drift – The DEnBoost Algorithm. *Hybrid Artificial Intelligent Systems*. E. Corchado, M. Kurzyński and M. Woźniak, Springer Berlin Heidelberg. 6679: 164-171.
- Bellman, R. (1961). Adaptive control processes - A guided tour, Princeton University Press.
- Blanco-Vega, R., Hernández-Orallo, J., et al. (2004). Analysing the trade-off between comprehensibility and accuracy in mimetic models. *Discovery Science*. E. Suzuki and S. Arikawa, Springer Berlin Heidelberg. 3245: 338-346.
- Blum, A. (1997). "Empirical support for Winnow and weighted-majority algorithms: Results on a calendar scheduling domain." *Machine Learning* 26(1): 5-23.
- Boser, B. E., Guyon, I. M., et al. (1992). A training algorithm for optimal margin classifiers. *Proc. 5th Annu. Workshop on Computational Learning Theory*, Pittsburgh, Pennsylvania, USA, ACM.
- Bouchachia, A. (2006). Incremental learning by decomposition. *5th Int. Conf. Machine Learning and Applications*, IEEE Computer Society.
- Bouchachia, A., Gabrys, B., et al. (2007). Overview of Some Incremental Learning Algorithms. *IEEE Int. Conf. Fuzzy Systems*: 1-6.
- Bousfield, W. A. (1953). "The occurrence of clustering in the recall of randomly arranged associates " *The Journal of General Psychology* 49: 229-240.
- Breiman, L. (1984). Classification and regression trees. Belmont, California, Wadsworth International Group.
- Broomhead, D. S. and Lowe, D. (1988). "Multivariable functional interpolation and adaptive networks." *Complex Systems* 2: 321-355.
- Carpenter, G. A., Grossberg, S., et al. (1992). "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps." *IEEE Transactions on Neural Networks* 3(5): 698-713.
- Cendrowska, J. (1987). "PRISM: An algorithm for inducing modular rules." *International Journal of Man-Machine Studies* 27(4): 349-370.

- Chakraborty, D. and Pal, N. R. (2003). "A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning." *IEEE Transactions on Neural Networks* 14(1): 1-14.
- Chan, P. K. and Stolfo, S. J. (2001). Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*.
- Chaves, A. C. F., Vellasco, M. M. B. R., et al. (2005). Fuzzy rule extraction from support vector machines. *5th Int. Conf. Hybrid Intelligent Systems*.
- Chawla, N., Lazarevic, A., et al. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD 2003*. N. Lavrač, D. Gamberger, L. Todorovski and H. Blockeel, Springer Berlin Heidelberg. 2838: 107-119.
- Chawla, N. V., Bowyer, K. W., et al. (2002). "Smote: synthetic minority oversampling technique." *Journal of Artificial Intelligence Research*(16): 321-357.
- Chen, S. (1995). "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning." *Electronics Letters* 31(2): 117-118.
- Chen, S. and He, H. (2009). SERA: selectively recursive approach towards nonstationary imbalanced stream data mining. *Int. Joint Conf. on Neural Networks*, Atlanta, Georgia, USA, IEEE Press.
- Chen, S. and He, H. (2011). "Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach." *Evolving Systems* 2(1): 35-50.
- Chinrungrueng, C. and Sequin, C. H. (1995). "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate." *IEEE Transactions on Neural Networks* 6(1): 157-169.
- Clark, P. and Niblett, T. (1989). "The CN2 induction algorithm." *Machine Learning* 3(4): 261-283.
- Cohen, N. J. and Squire, L. R. (1980). "Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that." *Science* 210(4466): 207-210.
- Cohen, W. W. (1995). Fast effective rule induction. *Proc. 12th Int. Conf. Machine Learning*, Morgan Kaufmann.
- Conway, M. A., Gardiner, J. M., et al. (1997). "Changes in memory awareness during learning: The acquisition of knowledge by psychology undergraduates." *Journal of Experimental Psychology: General* 126(4): 393-413.

- Cortes, C. and Vapnik, V. (1995). "Support-vector networks." *Machine Learning* 20(3): 273-297.
- Cost, S. and Salzberg, S. (1993). "A weighted nearest neighbor algorithm for learning with symbolic features." *Machine Learning* 10(1): 57-78.
- Cover, T. and Hart, P. (1967). "Nearest neighbor pattern classification." *IEEE Transactions on Information Theory* 13(1): 21-27.
- Craven, M. and Shavlik, J. (1994). Using sampling and queries to extract rules from trained neural networks. *11th Int. Conf. Machine Learning*.
- Craven, M. and Shavlik, J. W. (1995). Extracting tree-structured representations of trained networks. *Neural Information Processing Systems*.
- Dasarathy, B. V. and Sheela, B. V. (1979). "A composite classifier system design: Concepts and methodology." *Proceedings of the IEEE* 67(5): 708-713.
- Delany, S. J., Cunningham, P., et al. (2006). A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering. 19th Int. Conf. Artificial Intelligence G. S. a. R.Goebel, AAAI Press: 340-345.
- Demsar, J. (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets." *Journal of Machine Learning Research* 7: 1-30.
- Denton, S., Kruschke, J., et al. (2008). "Rule-based extrapolation: A continuing challenge for exemplar models." *Psychonomic Bulletin & Review* 15(4): 780-786.
- Ditzler, G., Polikar, R., et al. (2010). An incremental learning algorithm for non-stationary environments and class imbalance. *20th Int. Conf. Pattern Recognition*
- Domingos, P. (1996). "Unifying instance-based and rule-based induction." *Machine Learning* 24(2): 141-168.
- Domingos, P. (1999). MetaCost: a general method for making classifiers cost-sensitive. *5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, ACM Press.
- Domingos, P. and Pazzani, M. (1997). "On the optimality of the simple Bayesian classifier under zero-one loss." *Machine Learning* 29: 103-130.
- Dudani, S. A. (1976). "The distance-weighted k-nearest-neighbor rule." *IEEE Transactions on Systems, Man and Cybernetics* 6(4): 325-327.
- Elkan, C. (2001). The foundations of cost-sensitive learning. *Proc. 17th Int. Joint Conf. Artificial Intelligence*.

- Erickson, M. and Kruschke, J. (2002). "Rule-based extrapolation in perceptual categorization." *Psychonomic Bulletin & Review* 9(1): 160-168.
- Erickson, M. A. and Kruschke, J. K. (1998). "Rules and exemplars in category learning." *Journal of Experimental Psychology: General*(127): 107-140.
- Fan, W., Stolfo, S., et al. (1999). AdaCost: misclassification cost-sensitive boosting. *Proc. 16th Int. Conf. Machine Learning*, Morgan Kaufmann.
- Freund, Y. and Mansour, Y. (1997). Learning under persistent drift. *Proc. 3rd European Conf. Computational Learning Theory*
- Friedman, J., Yun, Y., et al. (1996). Lazy Decision Trees. *Proc. 13th Nat'l Conf. Artificial Intelligence*.
- Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*. G. Tesauro, D. S. Touretzky and T. K. Leen, MIT Press: 625-632.
- Fu, X., Ong, C., et al. (2004). Extracting the knowledge embedded in support vector machines. *IEEE Int. Joint Conf. on Neural Networks*.
- Fung, G., Sandilya, S., et al. (2005). Rule Extraction from Linear Support Vector Machines *ACM SIGKDD*.
- Fürnkranz, J. and Widmer, G. (1994). Incremental reduced error pruning. *11th Int. Conf. Machine Learning*, Morgan Kaufmann.
- Gabrys, B. and Bargiela, A. (2000). "General fuzzy min-max neural network for clustering and classification " *IEEE Transactions on Neural Networks* 11: 769-783.
- Gama, J., Medas, P., et al. (2004). Learning with drift detection. *SBIA Brazilian Symp. Artificial Intelligence*.
- Gao, J., Fan, W., et al. (2007). A general framework for mining concept-drifting data streams with skewed distributions. *7th SIAM Int. Conf. Data Mining* Minneapolis, Minnesota, USA
- Gao, M., Hong, X., et al. (2012). Probability density function estimation based over-sampling for imbalanced two-class problems. *Int. Joint Conf. Neural Networks* Brisbane, Australia
- Guo, X., Yin, Y., et al. (2008). On the Class Imbalance Problem. *4th Int. Conf. Natural Computation*
- Hamerly, G. and Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. *Proc. 11th Int. Conf. Information and Knowledge Management. McLean, Virginia, USA*, ACM: 600-607.

- Han, H., Wang, W.-Y., et al. (2005). Borderline-SMOTE: A new over-sampling method in Imbalanced data sets learning. *Proc. Int. Conf. Intelligent Computing*, China.
- Hastie, T., Tibshirani, R., et al. (2009). Prototype methods and nearest-neighbors *The Elements of Statistical Learning: Data Mining Inference and Prediction*. New York Springer-Verlag New York: 459-484.
- He, H., Chen, S., et al. (2011). "Incremental Learning From Stream Data." *IEEE Transactions on Neural Networks* 22(12): 1901-1914.
- Herbert, D. (1999). Remembering and knowing: the student learning experience *HERDSA Annu. Int. Conf.*, Melbourne
- Herbert, D. M. B. and Burt, J. S. (2003). "The effects of different review opportunities on schematisation of knowledge." *Learning and Instruction* 13(1): 73-92.
- Herbert, D. M. B. and Burt, J. S. (2004). "What do students remember? Episodic memory and the development of schematization " *Applied Cognitive Psychology* 18(1): 77-88.
- Hintzman, D. L. (1986). "'Schema abstraction" in a multiple-trace memory model." *Psychological Review* 93(4): 411-428.
- Hoens, R., Polikar, R., et al. (2012). "Learning from streaming data with concept drift and imbalance: an overview." *Progress in Artificial Intelligence* 1(1): 89-101.
- Hooker, G. (2004). Diagnostics and extrapolation in machine learning. Department of Statistics *California* Stanford University. Doctor of Philosophy: 127.
- Huang, G.-B., Zhu, Q.-Y., et al. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. *Proc. 2004 IEEE Int. Joint Conf. Neural Networks*.
- Hulten, G., Spencer, L., et al. (2001). Mining time-changing data streams. *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. San Francisco, California*, ACM: 97-106.
- Huysmans, J. and Baesens, B. (2006). Using rule extraction to improve the comprehensibility of predictive models. FETEW Research Report. *Leuven, Belgium Katholieke Universiteit Leuven* 1-55.
- Japkowicz, N. and Stephen, S. (2002). "The class imbalance problem: A systematic study." *Intelligent Data Analysis*(6): 429-449.
- Kahana, M. J., Howard, M. W., et al. (2008). "Associative retrieval processes in episodic memory " *Psychology* 3.

- Kapp, M. N., Sabourin, R., et al. (2010). Adaptive incremental learning with an ensemble of support vector machines. *20th Int. Conf. Pattern Recognition*
- Karnick, M., Muhlbaier, M. D., et al. (2008). Incremental learning in non-stationary environments with concept drift using a multiple classifier based approach. *19th Int. Conf. Pattern Recognition*
- Kelly, M., Hand, D., et al. (1999). The impact of changing populations on classifier performance. *Knowledge Discovery and Data Mining*, San Diego, California, USA, ACM.
- Kifer, D., Ben-David, S., et al. (2004). Detecting change in data streams. *30th Int. Conf. Very Large Data Bases*, Toronto, Canada, VLDB Endowment.
- Klinkenberg, R. (2003). Predicting phases in business cycles under concept drift *Proc. LLWA*, Karlsruhe, Germany
- Klinkenberg, R. (2004). "Learning drifting concepts: Example selection vs. example weighting." *Intelligent Data Analysis* 8(3): 281-300.
- Kohavi, R., Langley, P., et al. (1997). The Utility of Feature Weighting in Nearest-Neighbor Algorithms. *Proc. ECML*
- Kolter, J. Z. and Maloof, M. A. (2007). "Dynamic weighted majority: An ensemble method for drifting concepts." *The Journal of Machine Learning Research* 8: 2755-2790.
- Konig, R., Johansson, U., et al. (2007). Genetic programming - a tool for flexible rule extraction. *IEEE Congr. Evolutionary Computation*
- Kosko, B. (1992). Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence, Prentice-Hall, Inc.
- Kosslyn, S. M. (1988). "Aspects of a cognitive neuroscience of mental imagery." *Science* 240(4859): 1621-1626.
- Kosslyn, S. M., Behrmann, M., et al. (1995). "The cognitive neuroscience of mental imagery." *Neuropsychologia* 33(11): 1335-1344.
- Kotsiantis, S., Kanellopoulos, D., et al. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*: 25-36.
- Kotsiantis, S., Zaharakis, I., et al. (2006). "Machine learning: a review of classification and combining techniques." *Artificial Intelligence Review* 26(3): 159-190.
- Kruschke, J. K. (1992). "ALCOVE: An exemplar-based connectionist model of category learning." *Psychological Review* 99(1): 22-44.

- Kruschke, J. K. and Erickson, M. A. (1994). Learning of rules that have high-frequency exceptions: New empirical data and a hybrid connectionist model *16th Annu. Conf. Cognitive Science Society* Hillsdale, N.J.: Erlbaum.
- Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. *1997 Int. Conf. Machine Learning*
- Kuncheva, L. I. (2004). Combining pattern classifiers : methods and algorithms. New York, J. Wiley: 189-202.
- Kuncheva, L. I. and Zliobaite, I. (2009). "On the window size for classification in changing environments." *Intelligent Data Analysis* 13(6): 861-872.
- Larsen, P. M. (1980). "Industrial applications of fuzzy logic control." *International Journal of Man-Machine Studies* 12(1): 3-10.
- Laurikkala, J. (2001). Improving Identification of Difficult Small Classes by Balancing Class Distribution. *Proc. 8th Conf. AI in Medicine in Europe: Artificial Intelligence Medicine*, Springer-Verlag.
- Lichtenwalter, R. N. and Chawla, N. V. (2010). Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. *New Frontiers in Applied Data Mining*. T. Theeramunkong, C. Nattee, P. L. Adeodato et al, Springer Berlin Heidelberg. 5669: 53-75.
- Lughofer, E. (2012a). "A dynamic split-and-merge approach for evolving cluster models." *Evolving Systems* 3(3): 135-151.
- Lughofer, E. (2012b). "Single-pass active learning with conflict and ignorance." *Evolving Systems* 3(4): 251-271.
- Lughofer, E. and Angelov, P. (2011). "Handling drifts and shifts in on-line data streams with evolving fuzzy systems." *Applied Soft Computing* 11(2): 2057-2068.
- Lughofer, E. and Buchtala, O. (2012). "Reliable All-Pairs Evolving Fuzzy Classifiers." *IEEE Transactions on Fuzzy Systems* PP(99): 1-1.
- Mamdani, E. H. (1977). Applications of fuzzy set theory to control systems: A survey. *Fuzzy Automata and Decision Processes*. M. M. Gupta, G. N. Saridis and B. R. Gaines. New York, North-Holland: 1-13.
- Markou, M. and Singh, S. (2003a). "Novelty detection: a review—part 1: statistical approaches." *Signal Processing* 83(12): 2481-2497.
- Markou, M. and Singh, S. (2003b). "Novelty detection: a review—part 2: neural network based approaches." *Signal Processing* 83(12): 2499-2521.
- Matthews, B. W. (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme." *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405(2): 442-451.

- McKinley, S. C. and Nosofsky, R. M. (1995). "Investigations of exemplar and decision bound models in large, ill-defined category structures " *Journal of Experimental Psychology: Human Perception and Performance* 21(1): 128-148.
- McMillan, C., Mozer, M., et al. (1991). The connectionist scientist game: Rule extraction and refinement in a neural network. *13th Annu. Conf. Cognitive Science Society*.
- Medin, D. L. and Schaffer, M. M. (1978). "Context theory of classification learning." *Psychological Review* 85(3): 207-238.
- Meisel, W. S. (1972). Computer-oriented approaches to pattern recognition Orlando, Academic Press 100-101.
- Mencar, C. and Fanelli, A. M. (2008). "Interpretability constraints for fuzzy information granulation." *Information Sciences* 178(24): 4585-4618.
- Michalski, R. S. (1969). On the Quasi-Minimal Solution of the General Covering Problem. *Proc. 5th Int. Symp. Information Processing (Switching Circuits)*, Yugoslavia.
- Moreno-Torres, J. G., Raeder, T., et al. (2012). "A unifying view on dataset shift in classification." *Pattern Recognition* 45(1): 521-530.
- Muhlbaier, M., Topalis, A., et al. (2004). Incremental learning from unbalanced data. *IEEE Int. Joint Conf. Neural Networks*.
- Nishida, K. and Yamauchi, K. (2007). Adaptive classifiers-ensemble system for tracking concept drift. *Int. Conf. Machine Learning and Cybernetics*.
- Nishida, K., Yamauchi, K., et al. (2005). ACE: Adaptive classifiers-ensemble system for concept-drifting environments. *Multiple Classifier Systems*. N. Oza, R. Polikar, J. Kittler and F. Roli, Springer Berlin Heidelberg. 3541: 176-185.
- Nosofsky, R. M. (1986). "Attention, similarity, and the identification-categorization relationship." *Journal of Experimental Psychology: General*(115): 39-57.
- Nosofsky, R. M. and Bergert, F. B. (2007). "Limitations of exemplar models of multiattribute probabilistic inference." *Journal of Experimental Psychology: Learning, Memory, and Cognition*(33): 999-1019.
- Nosofsky, R. M., Clark, S. E., et al. (1989). "Rules and exemplars in categorization, identification, and recognition." *Journal of Experimental Psychology: Learning, Memory, and Cognition*(15): 282-304.
- Nosofsky, R. M., Palmeri, T. J., et al. (1994). "Rule-plus-exception model of classification learning." *Psychological Review* 101(1): 53-79.

- Núñez, H., Angulo, C., et al. (2006). "Rule based learning systems for support vector machines." *Neural Processing Letters* 24: 1-18.
- Okamoto, K., Ozawa, S., et al. (2003). A fast incremental learning algorithm of RBF networks with long-term memory. *Proc. Int. Joint Conf. Neural Networks*.
- Ozawa, S., Tabuchi, T., et al. (2010). "An autonomous incremental learning algorithm for radial basis function networks." *Journal of Intelligent Learning Systems and Applications* 2(4): 179-189.
- Page, E. S. (1954). "Continuous Inspection Schemes." *Biometrika* 41(1/2): 100-115.
- Parpinelli, R. S., Lopes, H. S., et al. (2001). An ant colony based system for data mining: applications to medical data. *Proc. 2001 Genetic and Evolutionary Computation Conference* Morgan Kaufmann.
- Pavlidis, N. G., Tasoulis, D. K., et al. (2011). "Lamba-Perceptron: An adaptive classifier for data streams." *Pattern Recognition* 44(1): 78-96.
- Polikar, R. (2006). *Pattern Recognition Wiley Encyclopedia of Biomedical Engineering* M. Akay. New York, NY: Wiley Interscience 4: 2695-2716.
- Polikar, R., Krause, S., et al. (2003). Ensemble of classifiers based incremental learning with dynamic voting weight update. *Int. Joint Conf. Neural Networks*.
- Polikar, R., Upda, L., et al. (2001). "Learn++: an incremental learning algorithm for supervised neural networks." *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.* 31(4): 497-508.
- Pothos, E. M. (2005). "The rules versus similarity distinction." *Behavioral and Brain Sciences* 28(01): 1-14.
- Quinlan, J. R. (1986). "Induction of Decision Trees." *Machine Learning* 1(1): 81-106.
- Quinlan, J. R. (1993). *C4.5 : programs for machine learning*. San Mateo, Calif., Morgan Kaufmann.
- Quinonero-Candela, J. (2009). *Dataset shift in machine learning* Cambridge, Mass. : MIT Press, c2009.
- Ripley, B. D. and Rasson, J. P. (1977). "Finding the edge of a Poisson forest " *Journal of Applied Probability* 14: 483-491.
- Rips, L. J. (1989). Similarity, typicality, and categorization. *Similarity and analogical reasoning*, Cambridge University Press: 21-59.
- Roberts, S. W. (1959). "Control Chart Tests Based on Geometric Moving Averages." *Technometrics* 1(3): 239-250.

- Rouder, J. N. and Ratcliff, R. (2006). "Comparing exemplar- and rule-based theories of categorization." *Current Directions in Psychological Science* 15: 9-13.
- Rumelhart, D. E., Hinton, G. E., et al. (1986). Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition*. E. R. David, L. M. James and C. P. R. Group, MIT Press. 1: 318-362.
- Rumelhart, D. E., McClelland, J. L., et al., Eds. (1986). *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1: foundations, MIT Press.
- Ruping, S. (2001). Incremental learning with support vector machines. *IEEE Int. Conf. Data Mining*
- Salganicoff, M. (1997). "Tolerating concept and sampling shift in lazy learning using prediction error context switching." *Artificial Intelligence Review* 11(1): 133-155.
- Salzberg, S. (1991). "A nearest hyperrectangle learning method." *Machine Learning* 6(3): 251-276.
- Sayed-Mouchaweh, M. and Lughofer, E. (2012). *Learning in non-stationary environments methods and applications* New York Springer
- Schölkopf, B., Platt, J. C., et al. (2001). "Estimating the Support of a High-Dimensional Distribution." *Neural Computation* 13(7): 1443-1471.
- Shiffrin, R. M. and Schneider, W. (1977). "Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory." *Psychological Review*(84): 127-190.
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. New York Chapman and Hall.
- Sit, W. Y. and Mao, K. Z. (2012a). Cognitively inspired classification for adapting to data distribution changes. *IEEE Conf. Evolving and Adaptive Intelligent Systems*, Madrid, 17-18 May, pp. 41-46.
- Sit, W. Y. and Mao, K. Z. (2012b). A cognitively inspired rule-plus-exemplar framework for interpretable pattern classification. *15th International Conference on Information Fusion (FUSION)*, Singapore, 9-12 July, pp. 1188-1195.
- Sit, W. Y. and Mao, K. Z. (2013a). "Cognitively inspired rule-plus-exemplar classification framework for concept growth." *Journal of Machine Learning Research*, under review.
- Sit, W. Y. and Mao, K. Z. (2013b). Learning imbalanced classes in the presence of concept growth. *2013 IEEE Symposium on Evolving and Adaptive Intelligent Systems*, Singapore, 16-19 April, pp. 62-69.

- Sit, W. Y. and Mao, K. Z. (2013c). A rule-plus-exemplar classification system for adapting to concept growth. *2013 IEEE Int. Conf. Cybernetics (CYBCONF)*, Lausanne, 13-15 June, pp. 13-15.
- Smith, S. P. and Jain, A. K. (1984). "Testing for uniformity in multidimensional data." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6(1): 73-81.
- Stefanowski, J. (2001). Multiple and hybrid classifiers *Post-proc. 2nd Int. Conf. Formal methods and intelligent techniques in control, decision making, multimedia and robotics*, Warszawa
- Sternberg, R. J. and Mio, J. S. (2009). Cognitive psychology. Belmont, California, Wadsworth Cengage Learning.
- Storkey, A. (2009). When training and test sets are different: characterizing learning transfer *Dataset Shift in Machine Learning*. Cambridge, Massachusetts MIT Press: 1-28.
- Surma, J. and Vanhoof, K. (1995). Integrating rules and cases for the classification task. *Case-Based Reasoning Research and Development*. M. Veloso and A. Aamodt, Springer Berlin Heidelberg. 1010: 325-334.
- Syed, N., Liu, H., et al. (1999a). Incremental learning with support vector machines. Workshop on Support Vector Machines at Int. Joint Conf. Artificial Intelligence.
- Syed, N. A., Liu, H., et al. (1999b). Handling concept drifts in incremental learning with support vector machines. *5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Diego, California, United States, ACM.
- Taha, I. A. and Ghosh, J. (1999). "Symbolic interpretation of artificial neural networks." *IEEE Transactions on Knowledge and Data Engineering* 11(3): 448-463.
- Thrun, S. B. (1993). Extracting provably correct rules from artificial neural networks. *Bonn, Germany* University of Bonn.
- Towell, G. G. and Shavlik, J. W. (1993). "The extraction of refined rules from knowledge-based neural networks " *Machine Learning*: 71-101.
- Trabasso, T. and Bower, G. H. (1968). Attention in learning: theory and research. New York Wiley.
- Tsymbol, A. (2004). The problem of concept drift: Definitions and related work. Technical Report
- Tulving, E. and Donaldson, W. (1972). Episodic and semantic memory *Organization of memory* New York Academic Press: 382-402.
- Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*.

- Van de Merckt, T. and Decaestecker, C. (1995). About breaking the trade off between accuracy and comprehensibility in concept learning *Workshop on Comprehensibility in Machine Learning, IJCAI-95*, Montreal
- Vandierendonck, A. (1995). "A parallel rule activation and rule synthesis model for generalization in category learning." *Psychonomic Bulletin & Review* 2(4): 442-459.
- Wang, H., Fan, W., et al. (2003). Mining concept-drifting data streams using ensemble classifiers. *9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Washington, D.C., ACM.
- Wang, L. X. and Mendel, J. M. (1992). "Generating fuzzy rules by learning from examples." *IEEE Transactions on Systems, Man and Cybernetics* 22(6): 1414-1427.
- Weiss, G. and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study, Rutgers University
- Widmer, G. and Kubat, M. (1993). Effective learning in dynamic environments by explicit context tracking. *Proc. ECML, LNCS 667*.
- Widmer, G. and Kubat, M. (1996). "Learning in the presence of concept drift and hidden contexts." *Machine Learning* 23(1): 69-101.
- Wills, A., Milton, F., et al. (2013). "Is overall similarity classification less effortful than single-dimension classification?" *Quarterly Journal Of Experimental Psychology* 66(2): 299-318.
- Wilson, D. L. (1972). "Asymptotic properties of nearest neighbor rules using edited data." *IEEE Transactions on Systems, Man and Cybernetics* SMC-2(3): 408-421.
- Woods, K., Bowyer, K., et al. (1996). Combination of multiple classifiers using local accuracy estimates. *IEEE Computer Society Conf. Computer Vision and Pattern Recognition*
- Xing, Z., Pei, J., et al. (2011). Extracting interpretable features for early classification on time series *Proc. 11th SIAM Int. Conf. Data Mining* Phoenix, Arizona, USA
- Yager, R. R. and Filev, D. P. (1994). "Approximate clustering via the mountain method." *IEEE Transactions on Systems, Man and Cybernetics* 24(8): 1279-1284.
- Yamauchi, K. (2009). "Optimal incremental learning under covariate shift." *Memetic Computing* 1(4): 271-279.
- Yamauchi, K. (2010). Incremental learning and model selection under virtual concept drifting environments. *2010 Int. Joint Conf. Neural Networks*
- Zadeh, L. (1965). "Fuzzy Sets." *Information and Control* 8: 338-353.

- Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. *Proc. Int. Conf. Machine Learning*
- Zhang, H. (2004). The optimality of naive Bayes. *FLAIRS Conf.*, AAAI Press.
- Zhang, H., Berg, A. C., et al. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *IEEE Computer Society Conf. Computer Vision and Pattern Recognition*.
- Zhang, J. and Michalski, R. S. (1995). "An integration of rule induction and exemplar-based learning for graded concepts." *Machine Learning* 21(3): 235-267.
- Zhang, Y., Su, H., et al. (2005). Rule extraction from trained support vector machines. *Advances in Knowledge Discovery and Data Mining*. T. Ho, D. Cheung and H. Liu, Springer Berlin / Heidelberg. 3518: 92-95.
- Žliobaitė, I. (2009). Learning under concept drift: An overview. *Vilnius, Lithuania* Faculty of Mathematics and Informatics, Vilnius University.