uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Guichong Li

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Border Sampling Techniques in Machine Learning

TITRE DE LA THÈSE / TITLE OF THESIS

Nathalie Japkowicz

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Trevor Stocki

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Iluju Kiringa

Stan Matwin

Charles Ling (University of Western Ontario)

John Oommen

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

BORDER SAMPLING TECHNIQUES IN MACHINE LEARNING

By

Guichong Li

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Computer Science,
School of Information Technology and Engineering, at
University of Ottawa, Canada.

Your file Votre référence
ISBN: 978-0-494-66251-9
Our file Notre référence
ISBN: 978-0-494-66251-9

Abstract

Border identification (BI), which is regarded as a sample selection technique in Machine Learning, was previously proposed to help learning systems focus on the most relevant portion of the training set so as to improve learning accuracy. However, the traditional BI implementation suffers from a serious limitation: it is only able to identify partial borders. We first propose a new method called Border Identification in Two Stages, denoted as BI_2, to overcome this limitation by identifying a full border.

Based on BI_2, we develop a new sample selection method, called Border Sampling, for supervised learning tasks. This is achieved by adopting the Progressive Learning technique for augmenting borders, and by incorporating BI_2 with the Markov Chain Monte Carlo technique for scaling up Border Sampling on large datasets, and by assuming a novel geometric computation for improving the algorithmic convergence.

Further, we propose a novel Meta learning technique, called Cascading Customized Couple, for scaling up Bayes classifiers such as Naïve Bayes by assuming a domain separation strategy, which is regarded as a wrapped sample selection method while Border Sampling is shown as a filter sample selection method in the thesis.

Empirical results show that, first, Border Sampling (BS) is an efficient and effective sample selection technique for training common classifiers such as Naïve Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM), and Instance-based Learning (IBL), as compared with previously proposed sample selection techniques such as Condensed Nearest Neighbour rule and Edited Nearest Neighbour rule; second, the new Meta learning technique

Cascading Customized Couple (CCC) outperforms previously proposed Meta learning techniques such as Bagging, AdaBoost, and MultiBoostAB for boosting Naïve Bayes. We also apply our new techniques to a scientific application for explosion detection by building an optimal classification model.

Acknowledgements

I would like to express profound gratitude to my supervisor, Dr. Nathalie Japkowicz, who guides me to complete PhD study in University of Ottawa.

Especially, I am thankful for her invaluable support, supervision, and useful suggestions throughout this research. I do not think that I can achieve the same goal without her guidance.

I am also highly thankful to Dr. Trevor J. Stocki for his overall and incredible help throughout this study. It is unbelievable that many details in this study embody his intelligence and effort. I am grateful for the cooperation and encouragement of Radiation Protection Bureau of Health Canada leading by Dr R. Kurt Ungar and funding me in an application development. I really appreciate Ian Hoffman and Jing Yi for helping me to finish the experiments for judgement in the ICDM2008 competition such that I experienced a wonderful time.

I would like to acknowledge Dr. Howard J. Hamilton for supporting and guiding me for the completion of the Master study. This becomes a prerequisite for my further study, and starts a miracle of my life in Canada.

My full gratitude is given to my friends: Rongliang Li, Lian Yang, and Qingwen Miao for their firm supports. My sincere thanks go to my friend, Sheng Heng, who helps relieve my misfortune and hardship from the 5.12 Sichuan Earthquake in my hometown.

I am as ever, especially indebted to my parents, Qingshan and Jinzhong for their love and support throughout my life. Finally, I wish to express my appreciation to my wife and daughter, who came with me to go through the longstanding process of my study in Canada.

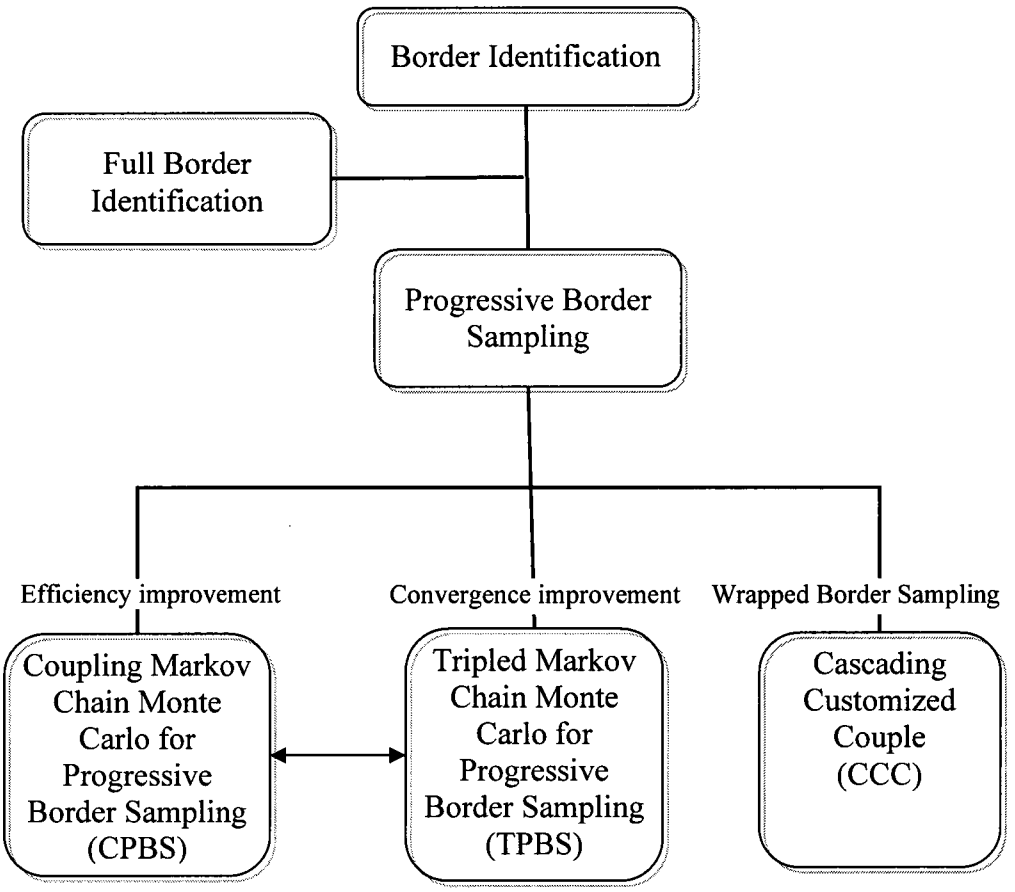

Figure 1.1. Flow of the dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AB	AdaBoost
AL	Active Learning
AODE	Aggregating One-Dependence Estimators
AODEsr	AODE with Subsumption Resolution
AUC	Area Under the Receiver Operating Characteristic (ROC) Curve
BI	Border Identification
BI_2	Border Identification In Two Stages
BS	Border Sampling
CC	Customized Classifier
CCC	Cascading Customized Couple
CFTP	Coupling From The Past
CIP	Class Imbalance Problem
CL	Cascade Learning
CNN	Condensed Nearest Neighbour rule
Cosine	Cosine measure
CPBS	Coupling Markov Chain Monte Carlo for Scaling Up Progressive Border Sampling
DROP3.1	A variant for combining Decremental Reduction Optimization Procedure 3 (DROP3) with Iterative Case Filtering (ICF)
ENN	Edited Nearest Neighbour rule
GNB	Gaussian Estimator for Naïve Bayes
HNB	Hidden Naïve Bayes
HVDM	Heterogeneous Value Difference Metric
LRLS	Linear Regression with Local Sampling
MCMC	Markov Chain Monte Carlo

MNB	Maximum Likelihood Estimator for Naïve Bayes
NB	Naïve Bayes
OA	One-against-All
OO	One-against-One
OSS	One-Side Selection
PBS	Progressive Border Sampling
PL	Progressive Learning
PS	Progressive Sampling
RBC	Recursive Bayesian Classifiers
RBF	Radial-Based Function
RENN	Repeated Edited Nearest Neighbour
SBC	Selective Bayesian Classifier
SMOTE	Synthetic Minority Over-Sampling Technique
SSL	Semi-Supervised Learning
SVM	Support Vector Machine
TPBS	Tripled Monte Chain Monte Carlo for Scaling Up Progressive Border Sampling
UCIKDD	University of California, Irvine, Knowledge Discovery in Databases
VDM	Value Difference Metric
WAODE	Weightily Averaged One-Dependence Estimators
Weka	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

Traditional instance selection [63][96], which is regarded as a preprocessing technique, has been studied for several decades in pattern recognition, machine learning, and data mining areas. There is a demand for this research area in many practical applications. Indeed, learning on massive amounts of data can exhaust computational resources and, thus, hinder and distract learners from building a successful classifier [49][72][93] when the massive amount of data contains much redundancy.

However, previously proposed instance selection methods still suffer from failure to achieve their goals. For example, earlier instance selection techniques, also called Prototype Reduction Schema (PRS), such as Condensed Nearest Neighbour rule (CNN) [96], which is similar to IB2 [2], are mainly designed for reducing sample complexity and improving prediction accuracy in Instance-Base Learning (IBL), which is one type of induction algorithms for classification. The resultant sample is subject to ineffectiveness for building other classifiers. This shows that approaches suffer from loss of information while they are used for reducing the sample complexity.

In addition to their ineffectiveness, early approaches suffer from the limitation of scalability on large datasets and the drawback of model-dependence. The recent research has proposed Progressive Sampling (PS) techniques [72] such as dynamic sampling and geometric sampling, which are quite scalable on large datasets by defining a random sampling schedule. However, the PS techniques are regarded as a wrapped sampling method

used in the base learner, and the resultant sample might not be effective for other classifiers. The PS and early other approaches are also regarded as model-dependent, and their application is restricted.

Recent research on instance selection techniques has emphasized the importance of scalability and model-independence [63]. Our purpose was to develop an advanced instance selection approach bearing these goals such as effectiveness, scalability, and model-independence in mind as well as the following ones.

In particular, we wondered whether the approach we designed could be used to address the Class Imbalance Problem (CIP) [12][13] [46][48][55], which many classifiers are subject to, because the CIP serviously degrades the performance of classifiers, especially for cost-sensitive learning [18][102].

We also realized that we cannot separate instance selection from classification issues although instance selection is regared as a preprocessing technique. An advanced instance selection technique can therefore also help learn successful classifiers and overcome the limitation of previously proposed learning algorithms. To achieve all these goals, we thus developed a new technique called Border Sampling for machine learning and data mining tasks. In addition to the proposed Border Sampling techniques, which are used as a filter sample selection method, we also developed a new Meta learning technique, called Cascading Customized Couple, which assumes a novel wrapped sample selection method to scale up individual learners for classification. The technique and its potential applications are introduced as follows.

1.1 Border Identification and Border Sampling

This section overviews how we developed our new technique by extending the previously proposed Border Identification technique.

The role of the training patterns located on the border lying close to the boundary separating samples of various classes has been studied in previous research [22][27][55]. The results show that a neural network trained with border patterns performs worse on the training set, but significantly better on the test set than one trained on the class cores [27]. Related research is called Border Identification, denoted as BI, which refers to as a technique that identifies border points in a labelled training set for a supervised learning task in machine learning area. As we can see, BI can be constructed as a sample selection technique, and the resulting sample can be used as a reduced training set for training a classifier. The BI technique is expected to help learners focus on the most relevant portion of the training set, and thus help learn a successful classifier. In this research, we re-investigate the role of training patterns on the border for reduction of training set size with respect to other common induction algorithms.

However, the traditional BI suffers from the limitation that it only identifies partial borders while it was observed that a full border consists of near borders and far borders. Therefore, it suffers from loss of information such that the resulting sample identified by BI is inadequate for use by training classifiers. As a result, we develop a new BI method, called Border Identification in Two Stages (BI_2), to identify a full border from a labelled training set to overcome the limitation of the traditional BI.

In addition, it is also shown that the borders identified by a BI method have high uncertainty from the perspective of Optimal Bayesian Learning Theory. As such, it is, thus, insufficient for classification because many induction algorithms build classifiers within this framework. A feasible

method is to add new borders from the remaining data. The new borders are more certain than the initially identified borders, and they are combined together as the resulting sample.

As a result, in this thesis, we begin by developing a new technique called Border Sampling, denoted as BS, which identifies proper border points from a labelled training set for a supervised learning task in Machine Learning by extending the traditional BI.

Next, there are several crucial questions that we should strictly define and answer to complete our investigation:

- How can BS be adapted for use on multiclass domains?
  - Why is pairwise border sampling ideal for border sampling?
- Can we establish a theoretical foundation of BS?
  - How can we formally define the concepts of border and redundant points?
- Can BS be made feasible on very large datasets?
  - How can we improve its time efficiency?
- Can BS be made to converge consistently?
  - How can we design an effective method for algorithm convergence: star convex collapse versus geometric computation?
- Can BS be used as a general sample selection technique for either a filter sample selection or wrapped sample selection?
  - In particular, could it be adapted as a wrapped sample selection for dealing with the Class Imbalance Problem (CIP) [5][12][13]?

1.2 Multiclass Domains

This section discusses how we developed Border Identification on multi-class domain by borrowing the same strategy as that used in induction algorithms for classification.

Learning on a binary domain is usually straightforward. For example, Support Vector Machine (SVM) is originally designed on a binary domain. Similarly, the BS technique, such as BI_2, is primarily developed on a binary domain. However, many practical applications define multiclass domains containing multiple class labels rather than only binary domains containing two class labels for supervised learning tasks.

In previous research, there is the class binarization method, e.g., one-against-one and one-against-all, etc, for classification, e.g., SVM, on multi-class domains [88]. Further, previous research has shown that pairwise Bayes classification is reduced to regular Bayes classification, and it is also true for Naïve Bayes [83].

As a result, the first question is straightforward. We can assume the one-against-one or the one-against-all for BS on multi-class domains by identifying each binary border from each pair of classes, and use the pairwise borders by combining all binary borders for the resulting borders. Further, pairwise Naïve Bayes built on the pairwise borders is reduced to a regular Naïve Bayes directly built on the resulting borders while Naïve Bayes with the one-against-all cannot be reduced to a regular Naïve Bayes due to the inaccuracy of probability estimation [83].

1.3 Theoretical Issues

In this section, we lay out the theoretical foundation for our approach. We develop Border Sampling techniques by overcoming the limitation of traditional Border Identification techniques. BS tends to find an effective

sample from any labelled training set. Its main advantage is that it is independent of classifiers or learners such that many classical learning algorithms can build successful classifiers on the resulting sample without loss of performance as compared with those classifiers built on the full training sets [61].

As a result, it is indispensible to establish a theoretical foundation about BS in advance of our further research on this new technique because a theoretical foundation help justify this novel technique.

For the second question, in Section 1.1, the theoretical foundation consists of the follow two aspects:

- How to formally define border and redundant points in a labelled training set ? It is not straightforward to correctly define border and redundant points. An improper definition can lead to a loss of information. It is shown that previous research is unsuccessful to establish a theoretical foundation for Border Identification.
- How to prove that an algorithm designed in terms of the formal definitions can correctly identify borders and remove redundant points? It is expected that redundant data points can be removed from training sets without any loss of information.

In general, Border Identification techniques are divided into three categories:

- Similarity distance method for the traditional Border Identification [22][27],
- Active Learning method for selecting border points from labelled and unlabeled data [15][21], and
- Voronoi diagram method for identifying borders by building a Voronoi diagram [23].

We adopt a similarity distance method to define borders and redundancy by specifying a similarity measure after we investigate the performances of three categories. According to our initial observation about the traditional Border Identification technique, in Section 3.1, we formally define a full border consisting of near borders and far borders, and redundant points, in Section 5.1 of Chapter 5. These definitions actually describe a new algorithm, called Border Identification in Two Stages.

1.4 Scalability

In this section, we survey how we solved the scalability of the proposed Border Sampling technique for sample selection on large datasets by incorporating it with the state of the art Markov Chain Monte Carlo technique.

Despite the advantages of BI_2 for identifying a full border and PBS for augmenting borders, both computations are still infeasible on large datasets since it is a quadratic learning algorithm. For example, it cannot process the Letter dataset from the UCIKDD repository [39] efficiently while the dataset contains ten thousand instances.

Recent research has focused on learning tasks on large datasets [49][72]. However, there exist some vital drawbacks in this research. For example, within the classification branch of machine learning, Progressive Sampling techniques [49][72][93], denoted as PS, are subject to a failure in converging to an optimal sample. Active Learning or Semi-Supervised Learning techniques [15][21][87] suffer from the same difficulty as PS to converge to an optimal sample with high bias of the selected learner.

A natural way is to adopt the standard Markov Chain Monte Carlo [3], denoted as MCMC, to scale up PBS on large datasets. Beyond the varied MCMC techniques, the state of the art Coupling From The Past [73], denoted

as CFTP, can produce an exact sample while the standard MCMC techniques cannot guarantee to converge to a stationary distribution [3][73].

For the third question, in Section 1.1, we proposed a novel Coupling Markov Chain Monte Carlo technique to scale up PBS for BS on large datasets, thus denoted as CPBS, by borrowing the main idea behind CFTP.

1.5 Algorithmic Convergence

In this section, we summarize how we will consider the algorithmic convergence of the proposed Border Sampling technique, which assumes the MCMC technique to scale up sample selection on large datasets. We improve the algorithmic convergence by adopting a novel geometric propagation method.

Efficiency and effectiveness are two crucial issues for BS on large datasets. BS can be very scalable on large datasets by assuming a MCMC technique, particularly the CFTP. One of the main problems is how to find an effective method for convergence detection in BS.

Borders are those data points lying on the boundary among classes. They can be simply defined as the nearest neighbours of some data points from other classes. Those data points identified by BI_2 on a subsample from a large population are referred to as local borders. As we know, local borders can fast evolve into global borders by assuming a MCMC technique. This means that the local nearest neighbour of a data point eventually evolves into a global nearest neighbour. It is known that the computation of nearest neighbours turns out to be a geometric computation.

For the fourth question, we first connect a geometric computation with BS. We define the geometry of a data point as its nearest neighbours belonging to another class. Because the nearest neighbour of a data point in a subsample sampled from a large population is different from the nearest

neighbour in the large population, the geometry of a data point is divided into local and global geometries. It is expected that a local geometry finally become a global geometry by incorporating an effective geometric computation in BS.

1.6 Wrapped Border Sampling

In this section, we explain that we considered a wrapped sample method to scale up individual classifiers and address the Class Imbalance Problem by separation of domains, which is regarded as a novel Meta learner and another type of border identification as well.

The Class Imbalance Problem [12][13], denoted as CIP, is one of the main problems that induction algorithms are subjected to in many practical applications, especially in the case where class distributions are highly skewed. It leads to induction algorithms unexpectedly overestimating the majority class while under-estimating the minority class.

With regard to the fifth question, in Section 1.1, for a wrapped border sampling on the CIP we note that our current research on BS based on BI_2 can be regarded as a filter sample selection technique performing prior to training time. We chose to test the filter sampling capability of Border Sampling for supervised learning tasks. It can be shown that BS as a filter can help produce an effective sample for learning successful classifiers. This is possible because BS is learner independent. Theoretically, it is believed that it can be used as a wrapped sample selection method.

In many learning tasks, a wrapped sample selection method turns out to be preferable to a filter sample selection method. For example, the decision tree induction algorithm, e.g., C4.5 [74], uses the information gain as a splitting threshold for building the nodes of the tree so as to separate the original domain into sub-domains. Instead of this implicit way for a wrapped sample

selection in C4.5, some simple sampling techniques such as under-sampling and over-sampling are usually used as an explicit way for sample selection.

In the context of the CIP [12][13], traditional basic sampling techniques such as under-sampling and over-sampling have the vital flaw of unavoidably suffering from loss of information. Recent research has a tendency to assume ensemble learning techniques wrapped with basic sampling techniques for the Class Imbalance Problem by enhancing individual classifiers [14][33][64]. As a result, instead of developing a possible method to wrap BS based on BI_2 in modeling, intuitively, we are encouraged to develop a novel Meta classifier with a novel wrapped sample selection method in order to enhance the performance of classifiers faced with the CIP even more.

In sum, we are interested in a new method consisting of the following two crucial techniques to enhance individual classifiers, and thus help address the CIP:

- using a wrapped sample selection method for training a successful classifier without any loss of information
- using an ensemble learning technique to improve individual classifiers

As a result, we propose a novel Meta learning method, called Cascading Customized Couple (CCC), to improve the performance of individual classifiers. CCC achieves its goal by building a couple of individual classifiers, which are called Customized Classifiers (CC), and are customized on its own sub-domain created by defining proper separations of the original domain.

Because a sub-domain is a separation of the original domain, and is created in terms of training errors output by a previously built CC, this separation is a wrapped sample selection method. Further, training errors or

misclassifications output by the first component in CCC are usually located at the region lying close to the decision boundary of the component. Therefore, these training errors are also regarded as borders, which are slightly different from those borders, which are close to the class boundary, identified by BI_2.

As a result, we developed two kinds of Border Sampling techniques: BS on class boundary by using BI_2 as a filter method and BS on decision boundary by defining the separation as a wrapped method in CCC. As we can see, BS on decision boundary is model-dependent while BS on class boundary is model-independent. Both are believed to be necessary as two effective sample selection methods for supervised learning tasks.

We compare CCC with previously proposed Meta learning techniques such as Bagging, AdaBoost, and MultiBoostAB, which have been successfully applied to many learning tasks. In particular, we show that CCC outperforms these Meta Learning techniques to scale up Naïve Bayes and other Bayesian classifiers such as those Naïve Bayes-like classifiers, e.g., AODE, AODEsr, HNB, and WAODE [47][92][100][101] while these Meta learners suffer from the same difficulty to enhance these Bayesian classifiers.

1.7 Contribution

Our main contributions consist of the following techniques and results:

- We first proposed a new method to identify a full border points in a labelled training set, in Chapter 3. The new method is called Border Identification in Two Stages, denoted as BI_2, for border identification by avoiding the limitation of the traditional Border Identification method.
- We proposed a new sampling technique for augmenting border points because the initial border points have high uncertainty for

adequate learning, in Chapter 4. The new sampling technique is called Progressive Border Sampling, denoted as PBS, for sample selection in supervised learning by incorporating BI_2 with the previously proposed Progressive Sampling technique, denoted as PS.

- We discussed an effective method for BS on multi-class domains because BS on multi-class domains is not a trivial issue, in Chapter 4. As a result, we adopt two possible strategies for BS on multi-class domains by borrowing ideas of class binaritization methods, which are originally used for classification. It is shown that the pairwise border sampling is preferable to the border sampling with the one-against-all on multi-class domains according to a theoretical analysis about pairwise Naïve Bayes [83].
- We formally established a theoretical foundation for Border Sampling, in Chapter 5. Basically, a labelled training set defines a latent full border, which consists of near borders and far borders. BS can efficiently and effectively remove all redundant data and maintain informative data. As a result, BS can produce an effective sample from the original training set.
- We proposed a novel method, which is called Coupling Markov Chain Monte Carlo for scaling up Progressive Border Sampling, denoted as CPBS, in Chapter 5. CPBS incorporates PBS with Coupling Markov Chain Monte Carlo by borrowing ideas behind the Coupling From The Past (CFTP).
- We developed an effective geometric computation for enhancing the convergence detection in CPBS, in Chapter 6. The proposed method is called Tripled Markov Chain Monte Carlo for Progressive Border Sampling (TPBS), which is an alternative

method of CPBS for scaling up PBS on large datasets by assuming the effective geometric computation for enhancing the algorithmic convergence.

- We proposed a new ensemble learning technique with a novel wrapped sample selection method for improving individual classifiers, in Chapter 7. The new ensemble technique is called Cascading Customized Classification (CCC), which is a couple of a new kind of classifiers, called Customized Classifier. The new wrapped sample selection method is regarded as Border Sampling on decision boundary by defining the separation in CCC instead of Border Sampling on class boundary by using BI_2.

1.8 Organization

The remainder of the thesis is organized as follows.

In Chapter 2, we review the previous research related to our work;

In Chapter 3, we start to introduce our new method for Border Identification, called Border Identification in Two Stage, denoted as BI_2, by analyzing an example;

In Chapter 4, we propose Progressive Border Sampling technique by incorporating Progressive Sampling technique with BI_2 to augment borders, and discuss possible strategies for BS on multi-class domains;

In Chapter 5, we establish a theoretical foundation of Border Sampling techniques, and develop a novel method for scalability of BS on large datasets by incorporating a state of the art Coupling Markov Chain Monte Carlo technique with PBS as an oracle;

In Chapter 6, we develop an alternative method of CPBS through effective geometric computation for the algorithmic convergence;

In Chapter 7, we develop a new ensemble learning method, called Cascading Customized Couple for scaling up individual classifiers based on a novel wrapped sample selection method, which is directly used for modeling and expected to help tackle the CIP;

In Chapter 8, we report on our experimental results to justify our new techniques proposed in this thesis.

Finally, we draw our conclusion and describe future work in Chapter 9.

First of all, the flow of the dissertation, as shown in Figure 1.1, gives us a first glance at the organization of the dissertation.

Chapter 2

Preliminary

In this chapter, we review previous research related to our work. It consists of the following aspects:

- Traditional Border Identification techniques for sample selection in supervised learning
- Progressive Sampling techniques, which are learner-independent sampling techniques for scaling up induction algorithms on large datasets in supervised learning.
- Instance selection, which is regarded as a general framework of sample selection in supervised learning. In this thesis, both instance selection and sample selection are alternatively used for convenience if no confusion occurs.
- Class Imbalance Problem (CIP), which is a problem that many classifiers suffer
- Cascade Learning and AdaBoost, which are two kinds of ensemble learning techniques for improving individual classifiers.

2.1 Traditional Border Identification

2.1.1 Definition of a Border in the BI context

In Duch's work [22], the *borders* of a vector or data point x are the sets of k nearest neighbours from this point to each of the classes x does not belong to. Therefore, data points which are far from borders are viewed as redundant and all new data points x can be unambiguously classified by the borders

close to x. The definitions of Duch's borders can be illustrated by a synthesized dataset with three categories: circle class, diamond class, and square class, as shown in Figure 2.1.

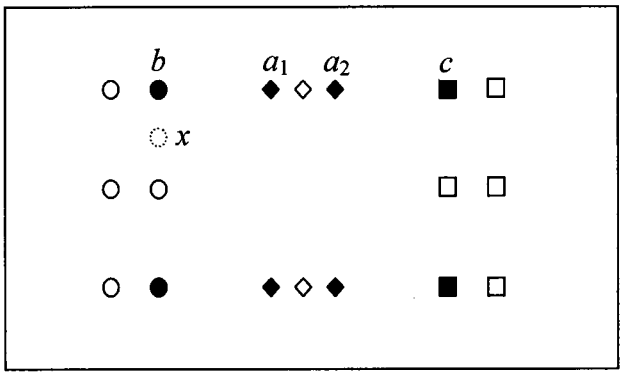

Figure 2.1. Duch's borders in a synthesized dataset with three categories.

Notes: The borders of a_1 are the nearest neighbours b and c from the circle class and the square class, respectively; The borders of a_1 are the same as those of a_2; all solid dots are borders while other dots are redundant points; an input x can be classified as the class b into the circle class because it is close to b.

In Foody's work [27], a border training set contain patterns drawn from different classes but which are close together in feature space and thereby expected to lie near the classification decision boundary. Therefore, these border training patterns can achieve an accurate classification. Further, the Mahalanobis distance, $d(x,y) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}}$, where the covariance matrix is diagonal and σ_i is the standard deviation of x_i, is used as a measure of the typicality of that pattern to the class.

We formalize the previous definitions about borders as follows.

Definition 2.1. Duch's reference set [22].

Given a data point p in a labelled training set D, *class borders* with respect to p are all k nearest neighbours of p from the other classes. The *border* or reference set of D consists of all borders for all data points in D.

Definition 2.2. Foody's borders [27].

Given a training set D and a sufficiently small threshold δ, any pair of data points $p, q \in D$ are *border training patterns* if $dist(p, q) \leq \delta$, where p and $q \in D$ have different class labels, and *dist* is a distance function with respect to a specified distance metric, e.g, Mahalanobis. A *border* training set consists of border training patterns.

As we can see, both definitions are based on a similarity distance metric, and borders can be obtained by computing nearest neighbours from between classes.

2.1.2 Traditional Border Identification algorithms

We describe previously proposed traditional BI techniques, which all are based on similarity distance metrics.

The first algorithm, denoted as Duch 1 [22], as shown in Figure 2.2, starts from the whole training set D and removes those vectors or instances $x \in$ D that have all k nearest vectors, denoted as P, from the same class, i.e., C(x, P) is true if all data points in P have the same class label as x has; otherwise, it is false. Further, the number k decreases from the maximum number K′ of nearest neighbours to the minimum number of nearest neighbours K, e.g., 10, where K < K′.

The second algorithm, denoted as Duch 2 [22], as shown in Figure 2.3, selects good reference vectors or borders for each vector or instance $x \in$ D by determining its k nearest vectors from each class that is different from C(x) and moving these instances to the reference set or border, where C(x) returns the class label of x.

It is easy to analyze their computational complexities, i.e., $O(n^2)$, where n is the number of instances in the training set D.

However, it is not easy to define K and K′ in Duch 1. We have no prior knowledge for properly setting K and K′. Duch 2 suffers from the same difficulty as Duch 1 with respect to defining a proper K. In addition, we suggest that Duch 2 is preferable to Duch 1 because Duch 2 is more consistent with Defintion 2.1 than Duch 1.

According to Definition 2.2, the Foody algorithm for BI can be described in Figure 2.4. The main problem is that we have to derive a proper scale for the measure threshold δ. Moreover, it is possible that there are different δ for different classes.

By comparison, Foody's algorithm is more similar to Duch 2's algorithm than to Duch 1's algorithm. Both are regarded as the *traditional BI technique* for Border Identification in supervised learning.

```
Duch 1 algorithm
Input      D: a training set
           K: the minimum number of nearest neighbors,
           K': the maximum number of nearest neighbors
Output     B: a border of D
begin
1     R = ∅
2     for i = K' to K
3           foreach x ∈ D
4                P = kNN(x, D, K)
                      //return the K nearest neighbors of x
5                if(C(x, P))
6                   R = R ∪ {p}
7           i--
8     B = D - R
9     return B
end
```

Figure 2.2. Duch 1 algorithm for Border Identification.

```
Duch 2 algorithm
Input      D: a training set
           K: the number of nearest neighbors
Output     B: a border of D
begin
1     B = ∅, C = partition(D), C = [C1,…, Ck]
2     for ∀i, j, where i < j, Ci ≠ ∅, Cj ≠ ∅
3           Pi = ⋃p∈Ci kNN(p, Cj, K)
4           Pj = ⋃p∈Cj kNN(p, Ci, K)
5           B = B ∪ Pi, B = B ∪ Pj
6     return B
end
```

Figure 2.3. Duch 2 algorithm for Border Identification.

```
Foody algorithm
Input      D: a training set
           δ: A sufficient small measure threshold
Output     B: a border of D
begin
1     B = ∅, C = partition(D), C = [C1,…, Ck]
2     for ∀i, j, where i < j, Ci ≠ ∅, Cj ≠ ∅
3           foreach x ∈ Ci
4               foreach y ∈ Cj
5                   if(dist(x, y) ≤ δ
6                       B = B ∪ {x} ∪ {y}
7     return B
end
```

Figure 2.4. Foody algorithm for Border Identification.

2.2 Progressive Sampling

Progressive Sampling (PS) [72] can maximize the accuracy of a model by learning on a small sample from the original large population. The standard PS [72] starts with a small sample and generates progressively larger ones until the model's accuracy no longer improves. Therefore, PS is also one of the possible sample selection techniques [49][72] for a supervised learning task.

There are two main components in PS: the sampling schedule and the convergence detection. The sampling schedule is denoted as $S = \{n_0, n_1, n_2,\ldots, n_k\}$, where each n_i is an integer that specifies the size of the ith sample and for $i < j$, $n_i < n_j$, $n_i < |D|$.

There are three different schedules considered in PS:

(I) The Static schedule [49], where $S_s = \{n_{\min}\}$. The value $n_{\min}$ can be calculated by a statistical similarity, e.g., Chi-square test, if the small sample with a single size $n_{\min}$ has the same data distribution as the full training set.

(II) The Arithmetic schedule [49], where $S_a = n_0 + i \times n_\delta = \{n_0, n_0 + n_\delta, n_0 + 2n_\delta,\ldots, n_0 + kn_\delta\}$, where $i = 1, 2, 3, \ldots, k$; n_0 is the initial sample size and n_δ is an increment.

(III) The Geometric schedule [72], where $S_g = a^i n_0 = \{n_0, an_0, a^2 n_0, a^3 n_0,\ldots, a^k n_0\}$, n_0 and a are constants, and $a = 2$ is the default for large datasets. $i = 1, 2, 3,.., k$.

The behaviours of the different aims of PS are described by a learning curve, denoted as acc(n), which is referred to as a curve of accuracy with respect to sample size n, and it can be created by a base learning algorithm. The learning curve is essential for convergence detection.

The learning curve in PS can be approximately fitted by the power law [49], which is given by

$acc(n) = a - bn^{-\alpha}$, where a, b, and α can be fit by an optimization method; n is the size of the sample.

The occurrence or vicinity of the peak of a learning curve is detected as the point of convergence, which corresponds to an optimal sample.

On the other hand, the *linear regression with local sampling* (LRLS) [72] technique computes the slope of the learning curve in terms of sample size n_i

and local samples in the neighbourhood of n_i samples, and detects convergence if the slope is sufficiently close to zero.

For the static schedule, in this thesis, n_{min} is simply defined as the size of augmented borders identified by the proposed PBS, in Section 4.2 of Chapter 4, i.e., $n_{min} = |\text{Border}|$.

The constants n_0 and n_δ in arithmetic and geometric PS as well as n_{LRLS} for LRLS can be given by

$$n_0, n_\delta, n_{LRLS} = \begin{cases} 10, \text{if} |D| < 1000 \\ 100, \text{if} |D| \geq 1000 \end{cases}$$

Therefore, the schedule in the geometric PS with LRLS can be dynamically given by

$S_g = a^i n_0 = \{n_0, an_0, an_0 + n_{LRLS}, a^2 n_0, a^2 n_0 + n_{LRLS}, a^3 n_0, a^3 n_0 + n_{LRLS}, \ldots, a^k n_0, a^k n_0 + n_{LRLS}\}$

Because a power law for acc(n) [49] is not expected to be correctly fit to a learning curve, in our research, the convergence detection for PS is determined by three successive points n_1, n_2, n_3 in the learning curve for capturing the peak of the learning curve as the convergence point n_2, i.e, the middle point, where given $\delta = 0.005$, we have

$\text{acc}(n_2) - \text{acc}(n_1) \geq 0 \wedge |\text{acc}(n_3) - \text{acc}(n_2)| < \delta$,

where $n_1 < n_2 < n_3$.

2.3 Instance Selection

The Instance selection problem refers to choosing a subset of data to achieve the original purpose of a data mining application [63]. Research on sampling, sample selection, and instance selection has a similar purpose although they might be a little different in different applications. In this section, we discuss the related topics in the general framework of instance selection. For

convenience, we use the term 'sample selection' and 'instance selection' alternatively, in this thesis, without any confusion.

2.3.1 Purpose

The initial purpose for instance selection is to scale up data mining algorithms. For example, instance selection for reduction of training set help fast classification in Instance Based learning (IBL), e.g., IBk algorithm [96]. An ideal sampling or instance selection technique can be incremental and model independent for all learning tasks [63]. In detail, we describe three main aspects related to an ideal instance selection technique as follows:

- Model-independence. The resulting sample should be learner-independent, and thus can be used for training any classifier without loss of information. As we can see, PS is a learning schema, which can be used for fast training of any classifier by producing a small sample [72]. It is regarded as a model-independent method for sample selection while there are still many methods, which are model-dependent for Instance-Based Learning [96].
- Scalability. A method should be scalable on large datasets. A linear or super-linear instance selection method is preferred. Further, an efficient instance selection approach should be able to become parallelized. Unfortunately, the previous methods usually have a quadratic time complexity [96], and cannot be easily parallelized.
- Incremental learning. After the resulting sample is obtained from the original large population, and new additional data is added in, sample selection on the new additional data never access the original whole population, and proceeds in the same process as the previous process on the original whole population. A precise incremental learning for

sample selection is difficult to achieve. Therefore, an approximate incremental learning [44] would be a compromised goal.

2.3.2 Previous Approaches

Some previously proposed instance selection techniques for reduction of training sets in Instance-Based Learning are introduced as follows.

The Condensed Nearest Neighbour rule (CNN) algorithm [37][96] finds a subset S of the training set T by a) randomly selecting one instance belonging to each output class from T and putting them in S; b) each instance in T is classified only using S. If an instance is misclassified, it is added to S; c) the process is repeated until there are no instances in T that are misclassified.

The Edited Nearest Neighbour Rule (ENN) [96] is a backward elimination method which a) supposes the initial subset S is the same as T; b) removes each instance in S that does not satisfy its agreement. That is, ENN reduces training sets by removing noise, which cannot be correctly classified by their k nearest neighbours, e.g., $k = 3$ in the original paper.

The Repeated Editing Nearest Neighbour rule (RENN) [96] can remove more noise than ENN. Because the noise removal might lead to a new source of noise, RENN repeatedly removes noise until no noise of this kind is found.

DROP3.1, in this thesis, is a variant of the Decremental Reduction Optimization Procedure 3 (DROP3) [96] and Iterative Case Filtering (ICF) [9], thus denoted as DROP3.1, which is used for removing redundant data points. DROP3.1 first executes ENN to remove noise from the original training set T, and sorts the resulting instances S by distances to their nearest neighbours belonging to the other classes in S, and then removes redundant points, which can be classified by their k nearest neighbours, e.g., $k = 5$ in

this thesis, in S with a high probability p, e.g., $p \geq 0.8$ in this thesis, without the redundant points.

Other approaches such as IB1, IB2 and IB3 [1] are a series of Instance-based Learning algorithms. IB1 is simply the 1-NN algorithm as a baseline; IB2 is quite similar to CNN while IB3 is similar to ENN.

2.3.3 Prototype Reduction Shema

Samples from training sets are also regarded as prototypes [11]. Instance selection is thus also regarded as a prototype reduction. We discuss recent research for prototype reduction as follows.

Prototype reduction schema (PRS) research can be divided into parametric and nonparametric methods. For example, traditional Vector Quantization (VQ) [98] and CNN [37] are regarded as two nonparameteric methods while LVQ 3, as dicussed below, is a parametric method.

Further, PRS can be also divided into the following three categories with respect to the way post-processing is handled:

- Seletion method, which only selects some of the existing data points as the prototypes [37];
- Creation and modification method, which creates new prototypes from the original prototypes [11][98]. For example, Vector Quantization (VQ) techniques [98] such as VQ1 and VQ2 are a modification schema, and have been shown to be an effective approach for data reduction.
- Creating and adjusting, which learns optimal positions of these prototypes from initially chosen prototypes toward the class boundary using the Learning Vector Quantization 3(LVQ3) algorithm because they yield a higher performance [51];

2.3.3.1 Learning Vector Quantization 3 (LVQ3)

One of the most recent researches, called Hybrid_PRS [51], is a hybrid method by creating and adjusting prototypes in two steps: first, select or create initial prototypes by any one of the converntial reduction methods, e.g., CNN; second, learning optimal postitions of the initial prototypes with an LVQ3 algorithm. The principle of LVQ3 is described as follows.

In LVQ3, given two code-book vectors (initial prototypes) m_i and m_j, for any prototype x, x falls into a window w if

$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > \left(\frac{1-w}{1+w}\right)$, where $w \in (0, 0.5]$, or generally $w \in [0.2, 0.3]$, d_i and d_j are Euclidean distances between x to m_i and m_j, respectively.

If x and m_j belong to the same class, and x and m_i belong to different classes, the updating rule is defined by

$m_i(t+1) = m_i(t) - \alpha(t)\,[x(t) - m_i(t)]$, i.e., m_i migrates toward x;

$m_j(t+1) = m_j(t) + \alpha(t)\,[x(t) - m_j(t)]$, i.e., m_j migrates outward x ;

If x, m_i, and m_j belong to the same class, the updating rule is defined by

$m_k(t+1) = m_k(t) - \varepsilon(t)\alpha(t)\,[x(t) - m_k(t)]$;

where $\varepsilon(t)$ and $\alpha(t)$ are called the learning rate and relative learning rate, respectively; $\varepsilon \in [0.1, 0.5]$.

Therefore, the performance of LVQ3 depends on the following factors:

- initial prototypes
- learning rate α and
- relative learning rate ε, and
- Number of time steps $T \in [50..200]$, and in general, we have

$$\alpha(t) = \alpha(0)\frac{T}{t+T}$$

As we can see, LVQ3 is an approximate method for PRS because it is bias to the window size w, its updating rules for empirical estimation, and the

number of iterations *T*. Neverthless, the Hybrid_PRS can enhance PRS by learning optimal parameters with LVQ3 using Placement and Optimzing sets, which are generated from the original training set [51].

2.3.3.2 Recursive PRS for large datasets

Because PRS suffers from low efficiency on large datasets, the recursive method, called Adaptive Recusive Parition (ARP) [52], for scaling up PRS is first introduced. The main idea consists of the following three aspects:

- The data set is subdivided recursively into smaller subsets;
- Converntional PRS processes the smaller subsets of data points that effectively are sampled from the entire space to yield subsets of prototypes.
- The prototypes which result from each subset are coalesced, and processes again by the PRS to yield more refined prototypes.

The ARP_PRS algorithm is described in Figure 2.5. As we can see, the algorithm is recursive for a tree-structure search.

```
Algorithm ARP_PRS
Input T: training set,
Output S: resulting training set
    Recursive_PRS(T, S, K, J)
End
Procedure Recursive_PRS(T, O, K, J)
Input:  T: training set
        K: a specified size of smallest set
        J:  number of subsets
Begin
    if |T| < K
        PRS(T, O)
        return O
    else
        Si = partition(T), where i = 0,..., J
        for(i = 0;i< J; i++)
            Recursive_PRS(Si, Oi)
        T = ∪ Oi, i = 0, ..., J – 1
        Recursive_PRS(T, O)
End
```

Figure 2.5. Adaptive Recursive Patition (ARP) for PRS algorithm.

2.3.4 Noise and Tomek links

Noise and Tomek links [86][96] are regarded as important patterns to influence the performance of classification. Because they are located at the class boundary, removing noise in Tomek links helps smooth the decision boundary so as to improve the generalization of classifiers [55][96].

A *Tomek link*, which is related to borderline and noise, can be defined as a pair of x and y as follows.

If no z exits such that

$$Dist(x,z) < Dist(x,y)$$

or

$$Dist(y,z) < Dist(y,x)$$

Noise in training sets can be defined as a data point with the wrong label [55][86][102], i.e., one that resides on the wrong side of the class boundary. Noise can usually be found in Tomek links. As we can see, those examples or data points in Tomek links are either borderline or noises.

2.3.5 Combination of continuous and nominal variables for distance metrics

Many instance selection methods work with a distance metric. Practical applications unavoidably contain continuous nominal variables. Therefore, a distance metric that combines continuous and nominal variables is crucial in instance selection.

Some of previous methods can be described as follows.

In Value Difference Metric (VDM) [12][16][96] for nominal variables, a matrix consisting of all v_i is given by

$$\delta(v_1,v_2) = \sum_{i=1}^{c} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k$$

or

$$\delta(v_1, v_2) = \sum_{i=1}^{c} \left| \frac{1}{p(v_1)} p(v_1 \mid c_i) - \frac{1}{p(v_2)} p(v_2 \mid c_i) \right|^k$$

where v_1 and v_2 are the two corresponding nominal feature values. C_1 is the total number of occurrences of feature value v_1, and C_{1i} is the number of occurrences of nominal feature value v_1 for class i. A similar convention can also be applied to C_{2i} and C_2. k is a constant, usually set to 1.

The distance between two feature vectors is given by:

$$\text{dist}(x, y) = w_x w_y \sum_{i=1}^{N} \delta(x_i, y_i)^r$$

where $r = 1$ yields the Manhattan distance, and $r = 2$ yields the Euclidean distance. w_x and w_y are the exemplar weights in the modified VDM.

As a heterogeneous metric, Heterogeneous Value Difference Metric, denoted as HVDM [95], between two data points x and y is given by

$$\text{HVDM}(x, y) = \sqrt{\sum_{i=1}^{m} d_i^2(x_i, y_i)}$$

where each data point has m attributes, i.e., x_i, y_i, $i = 1, \ldots, m$; and $d_i()$ is defined as

$$d_i(x, y) = \begin{cases} 1, & \text{if } x_i \text{ or } y_i \text{ is missing} \\ \text{VDM}(x_i, y_i), & \text{if attribute i is nominal} \\ \text{diff}(x_i, y_i), & \text{if attribute i is continuous} \end{cases}$$

As a result, VDM and diff in HVDM are suggested to be normalized [95]. We emphasize two points as follows:

Firstly, in HVDM, the distance will be 1 if any corresponding attribute value is missing;

Secondly, these distance metrics are not suggested directly to be used as a similarity distance metric for BI. They should be transformed by using a reverse function such that the similarity is equal to 1 if the distance is 0; the

similarity is equal to 0 if the distance is 1. That is, the original distance from a distance metric must fall in [0, 1].

The distance metrics assumed in this thesis for Border Sampling are similar to that defined in VDM and HVDM. For more discussion, see Section 3.5.

The distance metric defined in Smote-NC [12] for the Class Imbalance Problem is a little different from the metrics as discussed above. It is given by

$$Dist = \sum_{i=1}^{n_1} (x_{1i} - x_{2i})^2 + n_2 \mathrm{Med}^2,$$

where Med is calculated as the median of the standard deviations of continuous features of the minority class for a difference of a nominal feature with different values; n is the number of nominal features with different values.

2.4 Class Imbalance Problem

We are curious to know how a sample selection technique helps the Class Imbalance Problem (CIP). In this section, we describe some preliminaries about the CIP.

2.4.1 Definition of the Problem

The Class Imbalance Problem [12][13], denoted as CIP, is one of the main problems that induction algorithms are subjected to. This problem occurs in many practical applications, especially in the case where class distributions are highly skewed.

The CIP can be referred to as the situation in which learning algorithms unexpectedly classify some classes correctly with higher accuracy than others [13]. In other words, an induction algorithm unexpectedly overestimates the majority class while it under-estimates the minority class.

There are several possible causes for the CIP. Because the costs of procuring different training data are different [93] in many practical applications, class distributions in training data are highly skewed. A learning algorithm tends to build a classifier with high accuracy by correctly classifying more instances of the majority class and ignoring the errors of the minority class [13]. As a result, it is not expected that the learned classifier has a poor accuracy for predicting the minority class. In addition, it is also suggested that there exists some complicated concepts in data even though the data is not imbalanced with respect to the entire class distributions. These complicate concepts can be small disjunctions which still cause the CIP [48].

For example, a training data set from Chase Manhattan Bank contains 500,000 transactions with 20 percent of fraudulent transactions [10]. The class distribution, i.e., 20/80, is highly skewed. In this example, we would be required to build a classifier to detect fraudulent transactions.

Accuracy has generally been used for the evaluation of modeling. However, it has been shown that Accuracy is inappropriate for the evaluation of classification on the CIP [26]. With respect to the CIP, the Area Under Receiver Operating Characteristic (ROC) Curve, denoted as AUC, has been shown to be more stable for estimating the performance of classifiers than others, e.g., Accuracy, F-score, G-Mean, etc [26].

It is often seen that the cost of misclassifications on the minority class is greater than that of misclassifications on the majority class. Cost-sensitive learning studies on how a learner builds a classifier with a lesser cost of misclassifications when the costs of misclassifications of positive cases and negative cases are different [18][103]. However, there is no reason to treat CIP and Cost-sensitive Learning as two entirely different kinds of problems [20][93][102].

In our fraudulent transaction example, as a complicated case, the cost of failing to detect different fraudulent transactions is not the same [10]. In this thesis, we explore the CIP by assuming a uniform cost between the minority and majority class.

2.4.2 Methodology

The methods for the CIP can be divided into two kinds of methods. The first one is regarded as *basic* methods by under-sampling and over-sampling. The second one is regarded as *ensemble* learning methods by incorporating basic methods with ensemble learning techniques.

2.4.2.1 Basic methods

Basic methods, e.g., under-sampling and over-sampling, etc, which were proposed in previous research for the CIP, have been successful in practice and in theory [46]. Random sampling techniques such as under-sampling and over-sampling are direct avenues to tackle the CIP. *Under-sampling* performs random sampling with replacement on the majority class such that the size of the resulting subsample from the majority class is equal to that of the original minority class. On the other hand, *over-sampling* performs random sampling on the minority class such that the size of the resulting subsample from the minority class is equal to that of the original majority class.

Further, other methods were developed by assuming the two random sampling techniques as follows.

One-Side Selection (OSS) [55] is an under-sampling technique for the CIP by a) specifying an initial subset S consisting of the minority class and one instance of the majority class; b) on the assumption that an instance satisfies an *agreement* if it agrees with the majority of its k-nearest neighbours due to

the same class; and thus using the Condensed Nearest neighbours (CNN) [37][96] to edit those instances in the majority class, and then move into S if they do not satisfy their agreement on S; c) the Tomek links are removed as noises, where a *Tomek link* is a pair of instances, which are the nearest neighbour of each other with different class labels.

Synthetic Minority Over-sampling Technique (Smote) is different from the traditional over-sampling for the CIP [12]. It creates synthetic data with no replacement for the minority class such that the CIP is eliminated due to the balanced class. The Smote tackles the majority class by using under-sampling. In a word, Smote with 5 nearest neighbour searching is used in the minority class for synthesis of data while under-sampling is used in the majority class for reduction.

2.4.2.2 Ensemble learning methods for the CIP

Basic methods such as under-sampling and over-sampling can be incorporated into ensemble learning for the CIP.

EasyEnsemble [64] is an ensemble learning algorithm with individual AdaBoost classifiers for the CIP. It uses under sampling on the majority class to produce the number of balanced sub-domains between the minority class and the majority class. AdaBoost classifiers with weaker classifiers, e.g., Decision Stumps, are built correspondingly on these balanced sub-domains. Further, these individual AdaBoost classifiers [29] are stacked in terms of their outputs as features of a Meta learner [64][97]. EasyEnsemble returns the Meta learner as the resulting classifier.

BalanceCascade is similar to EasyEnsemble except that successive individual AdaBoost classifiers are built on misclassifications produced in previously learned individual AdaBoost classifiers.

Geng et al., used AdaBoost for the CIP in straightforward way [33]. The individual AdaBoost classifier with weak classifier, e.g., Decision Stump, was built on the result sub-domains from a CIP domain by resampling on the majority class and incorporating each resulting sub-sample from the majority class with the minority class. This approach has been used for Web Spam detection.

Chen, et. al, [14] proposed the approach that uses balanced random forest (BRF) to learn imbalanced data. BRF adopts under sampling on majority class to obtain the number of balanced sub-samples on which corresponding individual random forest classifiers are built.

Recent research also shows that boosting techniques can be successful with Support Vector Machine for the CIP [90].

2.4.3 Remarks

As we can see, basic sampling methods, e.g., under-sampling or over-sampling, help learners build better classifiers by sampling between the minority class and the majority class. As a result, the selected sample guides learners to build their decision boundaries. Furthermore, all of the ensemble learning methods described above assume random sampling, especially under-sampling, in the related ensemble learning algorithms.

The main problem is that these methods are restricted to be used in binary domains. The extension to multi-class domains is not straightforward. On the other hand, we have no prior information whether a domain is class imbalanced or not. Further, it is possible that a learner is subject to the CIP while another learner is not on the same domain. As a result, the capability of classifiers is crucial to overcome the CIP.

2.5 Summary

In this thesis, we apply and develop a new BI technique for sample selection in supervised learning. The previous Border Identification techniques, Duch 1, Duch 2, and Foody's BI, are investigated.

Duch 2 is similar to Foody's BI. Both are similarity distance based methods for BI. The state of the art Progressive Sampling techniques, which are regarded as model-independent methods, were developed for learning on large datasets.

Sampling or sample selection is also discussed in a general framework of instance selection for supervised learning tasks. Most of the previously proposed instance selection techniques are used as training set reduction techniques for Instance-Based Learning. Recent research on instance selection focuses on three properties: model-independence, scalability, and incrementality [63].

The roles of noise and Tomek links are discussed although they are not emphasized in this thesis. The representation of an instance is involved in a combination of continuous, nominal, and missing values from its continuous and nominal attributes. It is crucial to precisely calculate a similarity distance metric for instance selection.

We did a survey on the Class Imbalance Problem (CIP), and described its definition and methodologies. We are curious to know whether an instance selection technique can be used to solve the CIP. As a result, we emphasize a Meta learning technique, called Cascade Learning, to solve the CIP. It is compared with another Meta learning technique AdaBoost.

Our current research does not try to solve the problem of noise in Tomek links. We do not take into account of incremental sampling in this thesis. The combination of continuous and nominal variable in a distance metric is a

nontrivial issue. We will show an example of how previously proposed methods suffer a lacuna in the following discussions.

Chapter 3

Border Identification in Two Stage

In this chapter, we show the limitation of traditional BI methods, and show how we counter this limitation by introducing the concept of a *full border*.

3.1 Full Border

As discussed above, see in Section 2.1, both Duch' borders and Foody's borders are based on a similarity distance metric. It is useful to define a uniform border for further discussion. We define a uniform border from our new insight.

In general, a *border* does not concretely exist in a training set. A *latent border*, called a border for short, is specified in a labelled training set by the set of data points lying close to the boundary [22][27][55].

With a similarity distance metric, informative data points are defined for delineating a border as follows.

Definition 3.1. Given a data point p in a training set, the *informative data points* of p are its nearest neighbours from the other class. A set of informative data points is a *border*.

This definition follows and extends Duch's borders [22] and Food y's borders [27]. Further, Duch 1, Duch 2, and Foody algorithms have to be adapted with Definition 3.1, and they are regarded as a traditional BI method to identify borders or informative points defined in Definition 3.1.

Example 3.1. Given a labelled synthesized binary training set, pictorially denoting two different classes as circles and squares, with 10 data points, as shown in Figure 3.1, we use the BI method to identify its border. For each

circle point, we find its informative data points. As a result, all informative data points of the circle class is $B_c = \{3, 4\}$. Similarly, all informative data points of the square class is $B_s = \{1, 2\}$. The border $B = B_c \cup B_s = \{1, 2, 3, 4\}$.

However, the traditional BI methods according to Definition 3.1 are unable to learn a full border. For example, in Figure 3.1, a learned classifier built on B might have low performance for predicting data points 5, 6, 7, and 8. We observe that the resulting border does not contain the data points 5 and 7. No boundary between 2 and 5 or 4 and 7 can be easily learned. The data at 5 and 6 or 7 and 8 are far from the others.

Further, we consider how 9 and 10 are close to the border. The data point at 1 on the border is the nearest point to 9. They are both in the circle class. Similarly, 3 on the border is the nearest point to 10. They are both in the square class.

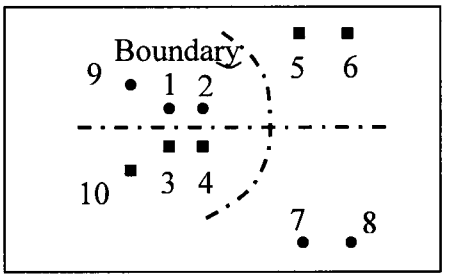

Figure 3.1. Border Identification in a synthesized training set.

Definition 3.2. A data point is *redundant* iff it is not on the identified border and is nearest to an informative data point with the same class label from the border.

As we can see, in Figure 3.1, the points 9 and 10 are two redundant data points while the points 5, 6, 7, 8 are neither informative data points nor redundant ones. For the points 5, 6, 7, and 8, we need to define a new border

among these data points for learning the hyperplane. We formalize this case by the following definitions.

Definition 3.3. A border identified by the BI method is defined as a *near* border. The border which cannot be directly identified by the BI method is called a *far* border. A *full* border consists of all near and far borders.

The simple definition of far border may allow developing an effective method to identify far borders. In practice, far borders, in our initial research, can be defined by exploring Example 3.2.

Example 3.2. Given the synthesized training set as shown in Figure 3.1, redundant data points 9 and 10 can be removed according to the near border $B_n = \{1, 2, 3, 4\}$. The remaining data points are neither the part of a border nor redundant data points. They are 5, 6, 7, and 8. As a result, the *far border* consisting of 5 and 7 can be effectively identified as informative data points of the border from the remaining data points. As a result $B = B_n \cup B_f = \{1, 2, 3, 4\} \cup \{5, 7\} = \{1, 2, 3, 4, 5, 7\}$.

3.1.1 Farther border

A far border can be a farther border to another already extended one. The *farther* border can be identified as informative data points of the previously extended border by removing its corresponding redundant data points. This is a recursive process. Simply stated, farther borders are also far borders.

For example, as discussed before, suppose we obtain the extended border $B = \{1, 2, 3, 4, 5, 7\}$. After removing its redundant data points 6 and 8, the remainder is empty. It shows that we cannot find anymore farther borders in this case.

3.1.2 Optimal border and bias

A border should be adequate for training an optimal classifier, and it is biased towards more informative data points than a narrow full border.

Instead of those definitions of Duch's borders and Foody's borders, see 2.1, and Definition 3.3 of a full border, an optimal border can be defined as follows.

Definition 3.4. An optimal border is an *augmented* full border, which consists of all augmented near and far borders for training a successful classifier.

However, we delay the related discussion how to achieve an augmented full border to Section 4.2 in Chapter 4.

3.1.3 Multi-class Domains

In Example 3.2, we describe a method for identifying a full border on a binary domain. For a multiclass domain, a full border can be found in a pairwise way which is similar to the oo strategy for classification [17][88], in learning algorithm for multiclass applications.

Suppose we have a training set with c classes. B_{ij} is a border between the class *i* and the class *j*. We have $B = \bigcup_{i \neq j} B_{ij}, i, j = 0, \ldots, c$, where those B_{ij} are not necessarily exclusive from each other. We further discuss BI on multiclass domains in Section 3.4.

3.2 Illustrations

We show the borders identified on two synthesized datasets as follows. Given the first synthesized data set, as shown in Figure 3.2(a), the BI with Radial kernel distance function [66], i.e., $e^{-d^2/2}$, where $d^2 = \sum \left(\frac{x_{ai} - x_{bi}}{\sigma_i} \right)^2$ for a Mahalanobis distance between two data points x_a and x_b by assuming independence among variables x_i, only identifies an incomplete border, as shown in Figure 3.2(b), while a full border is identified by our

new method BI_2 (see Section 3.3) in Figure 3.2(c), i.e., the informative data points indicated by the ovals were not identified by the BI. Figure 3.2(d) also shows the result of BI_2 with Cosine as a similarity measure.

On the second synthesized dataset representing a more complex XOR problem, as shown in Figure 3.3(a), the BI algorithm with Cosine similarity measure can only find an incomplete border, as shown in Figure 3.3(b), while the BI_2 with Cosine shows its complete capability to identify a full border in Figure 3.3(c) because those border points indicated by the ovals are identified. Figure 3.3(d) also shows the result by BI_2 with Radial Kernel distance function as a similarity measure.

It was observed that different distance measures in BI_2 have different effects. Previous research has shown that although the Cosine method is sensitive to the translation of distance, Cosine similarity measure normalizes naturally to the unit sphere [80]. The Cosine similarity measure is insensitive to the scale, and obtains more informative points in the class core [27]. On the other hand, RBF is insensitive to the translation, but sensitive to scale. No distance measure is superior to another one.

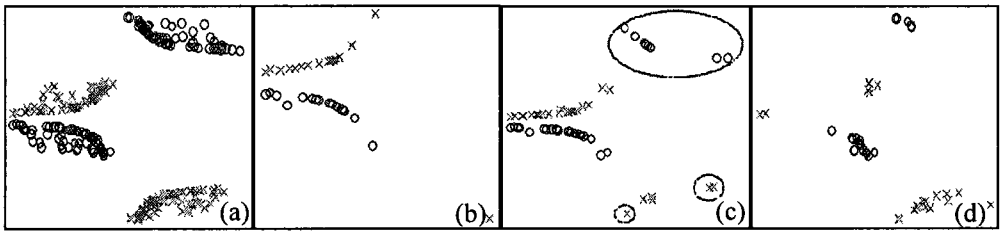

Figure 3.2. Border Identification by using BI and Radial Kernel.

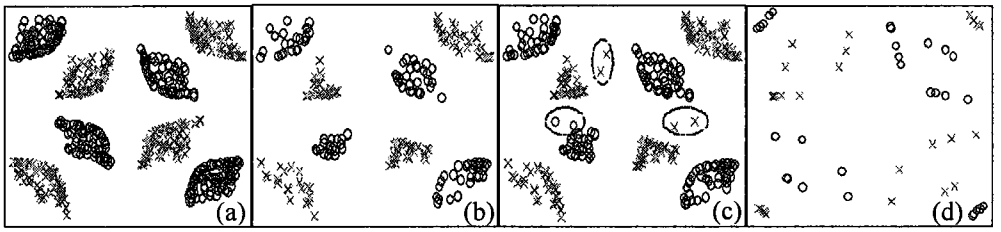

Figure 3.3. BI and Cosine. A complicate XOR problem.

Instead of analyzing and designing an optimal distance measure in BI_2, we chose the Cosine method for our initial experiments.

More details about combination of continuous, nominal, and missing values in BI_2 are discussed in Section 3.5.

3.3 BI_2 Algorithm

We propose the BI_2 algorithm, as shown in Figure 3.4, which assumes two stages to identify a full border for a binary domain. It has two inputs: two categories C_i and C_j, and it returns the resulting borders in B, which is initialized at Step 1.

At the first stage, BI_2 generates the near border between classes C_i and C_j at Step 2 and Step 3 by generating the informative data points of either C_i from C_j or C_j from C_i. 1stNN(p, C_x) at Steps 2 and 3 searches for all the 1-nearest neighbours of p from C_x in terms of a specified similarity metric. At Steps 4 and 5, B_{ij}, C_i, and C_j will be updated. At Step 6 and Step 7, BI_2 will identify far borders in C_i and C_j, with the remaining data points, respectively, by farBorder().

From Step 8 to Step 14, farBorder() in BI_2 initializes D′ with the set of the remaining data points D and the identified far border B_f with empty. B_k is one of its inputs, which is denoted as the resulting border corresponding to B_{ij} at Step 6 and Step 7.

The while loop between Step 9 and Step 13 first removes all the redundant data points in the remaining database using removeRedundant() at Step 10, i.e., $D' = \{x \mid \forall x \exists y, x \in D' \wedge y \in B_k \wedge y \in 1stNN(x, B_k) \wedge l(x) \neq l(y)\}$, where the $l(x)$ function returns the class label of x. As a result, those *tied* points are also informative data points, i.e., $l(x) = l(y')$. Step 11 is the only exit point of the while loop. It is activated on condition that D′ is empty. A far border B_f' is identified at Step 12 by generating the informative data points of B_k

from D′. At Step 13, the identified far borders B_f must be removed from the original D.

```
BI2 algorithm
Input     Ci, Cj: two classes
Output    Bij: the identified border between Ci and Cj;
begin
1    B = ∅
2    Pi = ⋃p∈Cj 1stNN(p, Ci)
3    Pj = ⋃p∈Ci 1stNN(p, Cj)
4    Bij = Bij ∪ Pi, Cj = Cj - Pi
5    Bij = Bij ∪ Pj, Ci = Ci - Pj
6    farBorder(Ci, Bij)
7    farBorder(Cj, Bij))
end
farBorder(D, Bk)
8    D′ = D; Bf = ∅;
9    while(true)
10       D′ = removeRedundant(D′, Bk)
11       if(D′ = ∅) break
12       B′f = ⋃p∈Bk 1stNN(p, D′)
13       Bk = Bk∪ B′f ; Bf = Bf ∪ B′f ; D′ = D′ - B′f
14   D = D - Bf
```

Figure 3.4. BI_2 algorithm for Border Identification.

The main procedures in BI_2 are several iterations of the 1stNN() and removeRedundant() procedures in the while loop of farBorder(). The time complexity of BI_2 can be analyzed as follows.

Suppose $n = |D|$ and $n_i = |C_i|$, $n = \sum n_i$. The time complexity of 1stNN() for two classes C_i and C_j is always $O(fn_in_j)$ for its informative data points, where f is the number of features in a training set.

The time complexity of removeRedundant() has an upper bound of $O(fn_in_j)$. The time complexity of the second stage at Step 6 and Step 7 can be computed by $O(frn_in_j)$, where r is the depth of the iteration in the while loop of farBorder().

Therefore, BI_2 has a time complexity of $O(frn^2)$ for the binary case. Empirically, r is bounded by a small number ($\ll n$). For example, r in Anneal case from UCIKDD repository [39] is 4. A theoretical analysis of the upper bound of r will not be discussed until Section 5.3.3.2.

Theorem 3.1. If p is identified by BI_2 as a redundant point with respect to any border B extended from the near border, then p is still redundant with respect to the full border.

Proof: suppose p is a redundant point identified with respect to the current borders B, and p' is the nearest neighbour of p from B. We can prove, see Section 5.1 in Chapter 5, that $\forall q$ from a class different from p's, we have $dist(p', p) < dist(p, q)$, where dist() is a specified distance metric. Hence, p is always redundant, and p cannot be identified as an informative point later on.

Theorem 3.1 is given as a justification for the correctness of BI_2. More details about theoretical issues pertaining to borders will be discussed in Section 5.1 in Chapter 5.

3.4 BI_2 on Multi-Class domains

Previous research has proposed the class binaritization methods, e.g., the pairwise or one-against-one method, and the one-against-all method, for classification on multi-class domains [17][38][88].

Similarly, we can use the same strategies for BI_2 on Multi-Class domains as follows:

- *pairwise method or one-against-one (oo)*
  BI_2 identifies borders on each pair of classes, and all resulting borders between two classes are unified together as a resulting full border, as shown in Figure 3.5.
- *one-against-all method (oa) or one-against-rest*

In each round, BI_2 identifies a border between a class and the rest of classes; the border from the class is added into the resulting full border, as shown in Figure 3.6

Because the oo and the oa methods might lead to different effects, we compare the two methods for BI as follows:

(a) Both assume that BI_2 will identify full borders between classes, and combine them together to create the resulting border;

(b) Both have a time complexity of $O(n^2)$;

(c) The oo method has an advantage over the oa method in that BI_2 in finding a border between each pair of classes only accesses the two classes. It might, thus, be more scalable than the oa method with respect to space complexity;

(d) However, (c) is not true if BI_2 frequently run on a small subsample resampled from a large population, see Section 5.3 in Chapter 5;

(e) The border identified by BI_2 with the oo might be a little larger than that by BI with the oa because borders from each class are obtained or *overestimated* by assuming BI_2 between the class and each of other k – 1 classes, where k is the number of classes.

(f) BI_2 with oa has a *variant* that identifies not only borders from each class but also additional borders from the rest of the classes. The variant is believed to have a potential advantage for classification on multi-class domains

```
BI-oo algorithm
Input      D: a labeled training set
Output     B: the identified border from D
begin
1     B = ∅, C = partition(D), C = [C1,…,Ck]
2     for ∀i,j, where i < j, Ci ≠ ∅, Cj ≠ ∅
3        Bij = BI2(Ci, Cj)
4        B = B ∪ Bij
5     return B
end
```

Figure 3.5. Pairwise BI algorithm based on BI_2 on Multi-Class domains.

```
BI-oa algorithm
Input      D: a labeled training set
Output     B: the identified border from D
begin
6     B = ∅, C = partition(D), C = [C1,…,Ck]
7     for I =0 to k, Ci ≠ ∅,
8        Cj = C - Ci
9        Bij = BI2(Ci, Cj)
10       Bi = Bij - Cj
11       B = B ∪ Bi
12    return B
end
```

Figure 3.6. BI with OA algorithm based on BI_2 on Multi-Class domains.

3.5 Similarity Distance Metrics for BI_2

An instance can be represented as a *vector*, which consists of all attribute-values pairs. However, the similarity distance computation becomes complex when values contain continuous, nominal, and missing values.

Many traditional similarity distance metrics [80][95] can be used in BI_2 for BS. However, there is a special treatment to apply for the combination of continuous, nominal, and missing values in BI_2.

The selected similarity measure is Cosine measure, which is favoured due to its natural normalization to the unit sphere [80][95]. Therefore, the explicit normalization is believed to be unnecessary [80].

In general, a data point x in a dataset D can be represented as an input vector $x = (x_1, x_2,\ldots, x_l, c)$, where $c \in C = \{c_1,\ldots c_j\}$ is a class label. The component x_i of x corresponds to a feature i, which has continuous, nominal, or missing values.

The Cosine similarity measure of two data points x_a and x_b, is given by

$$S_{Cosine}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^t \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \bullet \|\mathbf{x}_b\|_2} \quad (1)$$

Formally, suppose there are k features in the feature space while there is really $k + 1$ attributes in the dataset and the class attribute is not measured for S_{Cosine}.

According to Eq.(1),

(a) For $\mathbf{x}_a^t \mathbf{x}_b = \sum_{i=1}^{k} \mathbf{x}_a^i \mathbf{x}_b^i$,

If x_a^i and x_b^i are nominal, then

$$\mathbf{x}_a^i \mathbf{x}_b^i = \begin{cases} 1, \mathbf{x}_a^i \text{ and } \mathbf{x}_b^i \text{ are equal and not missing} \\ 1, \mathbf{x}_a^i \text{ and } \mathbf{x}_b^i \text{ are missing} \\ 0, \text{otherwise} \end{cases}$$

If $\mathbf{x}_a^i$ and $\mathbf{x}_b^i$ are continuous, then

$$\mathbf{x}_a^i \mathbf{x}_b^i = \begin{cases} \mathbf{x}_a^i \bullet \mathbf{x}_b^i, & \mathbf{x}_a^i \text{ and } \mathbf{x}_b^i \text{ are not missing} \\ \max_j^2 \{|\mathbf{x}_j^i|\}, & \mathbf{x}_a^i \text{ and } \mathbf{x}_b^i \text{ are both missing} \\ 0, & \text{otherwise} \end{cases}$$

(b) For $\|\mathbf{x}\|_2 = \sqrt{\sum (\mathbf{x}^i)^2}$,

$$(\mathbf{x}^i)^2 = \begin{cases} 1, & \mathbf{x}^i \text{ is nominal} \\ (\mathbf{x}^i)^2, & \text{continuous and not missing} \\ \max_j^2 \{|\mathbf{x}_j^i|\}, & \text{otherwise} \end{cases}$$

As we can see, if two corresponding nominal features have different values, their similarity is zero. In particular, the original missing values are not actually replaced with a known value for calculating the similarity. If

both attribute values are missing, they are only supposed to have the same value, e.g., the maximum value, empricially. We can assume the same methods to adapt other traditional distance metrics [80][95], e.g., Radial-based function (RBF), Pearson Coefficient, and Extended Jaccard similiarity, etc, for combination of continuous, nominal, and missing values in BI_2.

The method used in BI_2 for combination of continuous, nominal, and missing values is entirely different from the previous methods, e.g., Value Difference Metric (VDM) [95], for distance calculation. Empirically, the new method is very effective in BI_2 for Border Identification. There are two crucial differences for these combinations between metrics used in BI_2 and those used in the traditional BI as follows.

Firstly, two missing values should be regarded as equal values. As a result, their similarity distance should be zero, also see HVDM in Section 2.3.5;

Secondly, suppose we have a large population P, and we do resampling on P for a subsample S, where $S \subset P$. Given two instances p_1, $p_2 \in S$, a similarity distance method Dist() on either S or P is *invariant* if

$Dist_S(p_1, p_2) = Dist_P(p_1, p_2)$

An invariant similarity distance metric is important in statistical sampling techniques because it is not expected that the distances between two data points in two subsamples containing them are different.

However, not all similarity distance metrics are invariant. It can be shown that there is an abnormal case in that VDM between two different nominal values is 0 according to VDM, see Section 2.3.5,

$$\delta(v_1, v_2) = \sum_{i=1}^{c} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k$$

For example, as shown in Table 3.1, where we define a synthesized binary dataset (class labels 0 and 1) with 7 instances and an input space of two

feature a_1 and a_2, VDM between the tuple #0 and the tuple #7 in the whole dataset are equal to 0, i.e.,

$VDM_{0\text{-}7}(\#0, \#7) = \delta_1(0, 1) + \delta_2(0, 0)$

$= [|C_{11}/C_1 - C_{21}/C_2| + |C_{12}/C_1 - C_{22}/C_2|]_{a1} + [|C_{11}/C_1 - C_{21}/C_2| + |C_{12}/C_1 - C_{22}/C_2|]_{a2}$

$= (2/4 - 2/4) + (2/4 - 2/4) + 0 = 0$

However, the tuple #0 and the tuple #7 are actually different. Therefore, the tuple #0 has a different tuple as its nearest neighbours. This leads to much variance for modeling.

If we consider a subsample containing only two tuples #0 and #7, we have

$VDM_{0,7} = \delta_1(0, 1) + \delta_2(0, 0) = |1/1 - 0/1| + |0/1 - 1/1| = 2$

VDM is defined in terms of similar classification [96]. Because local statistics, e.g., C_i and C_{ij}, are used in the distance metric, it leads to a non-invariant distance metric, i.e., $VDM_{0\text{-}7} \neq VDM_{0,7}$

Table 3.1. A synthesized binary data with balanced classes.

#	a_1	a_2	c
0	0	0	0
1	1	0	0
2	0	1	1
3	1	1	1
4	0	1	0
5	1	1	0
6	0	0	1
7	1	0	1

For maintaining making an invariant distance metric, all statistics appearing in a distance metric should be global.

For example, in RBF, see Section 3.2,

$$\sum\left(\frac{x_{ai} - x_{bi}}{\sigma_i}\right)^2$$

σ_i should be a global statistic no matter whether two data points are in a subsample sampled from a large population.

3.6 Related work

Informally, the state-of-the-art methods for identifying borders can be divided into two categories: similarity distance-based method [22][27] and active learning-based method [15].

The similarity distance method can be directly used for identifying a border by scanning the whole training set. For each data point, the k-nearest neighbours from other classes are identified as data points on the border. BI can be also achieved by searching border points in a full Voronoi diagram by constructing the Voronoi diagram in which all adjacent data points are connected with each other [23].

Voronoi diagrams describe a structure in which all adjacent data points are connected with each other [4]. After a full Voronoi diagram is built on the data, the BI can be easily achieved by searching for border points, which are the nearest neighbours from the other class.

More details about BI based on Voronoi diagram and Active Learning are described as follows.

3.6.1 Voronoi Diagram

Let $S = \{p_1,\ldots, p_n\}$, where each p_i is called a site in the plane (2 dimensions). For two distinct sites $p, q \in S$, the dominance of p over q is defined as the subset of planes being closer to p than q, i.e.,

$$dom(p, q) = \{x \in R^2 \mid \delta(x, p) \leq \delta(x, q)\}$$

where δ denotes the Euclidean distance function [4].

dom(p, q) is a closed half plane bounded by the perpendicular bisector of the straight line of p and q. The region of a site $p \in S$ consists of all of the dominances of p over the remaining sites in S, i.e.,

$$V(p) = reg(p) = \bigcap_{q \in S-\{p\}} dom(p,q)$$

The Voronoi diagram V(S) consists of all partitions corresponding to all regions. For example a Voronoi diagram of 8 sites is shown in Figure 3.7 [4].

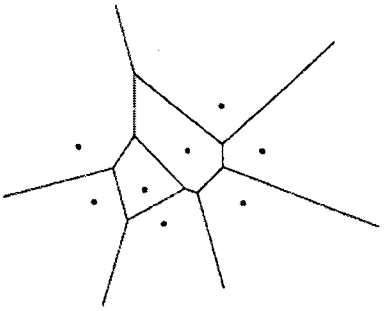

Figure 3.7. Voronoi diagram of 8 data points.

On the other hand, a Delaunay diagram, denoted as D(S), is the dual (an equivalent transformation) of the Voronoi diagram in that we put a Delaunay node (vertex) at each site p_i and we connect two sites p_i and p_j with a straight line segment if and only if the Voronoi cells $V(p_i)$ and $V(p_j)$ share a common boundary segment.

A generic definition of Voronoi diagrams in n-dimension, denoted n-Voronoi diagram can be described as follows.

For n points $p_1, \ldots, p_n$ of Euclidean d-space E^d, the Voronoi diagram [53] $V(p_1, \ldots, p_n)$ is a sequence $(V_1, \ldots, V_n)$ of convex polyhedra covering E^d, where V_i consists of all points of E^d that have p_i as a nearest point in the set $\{p_1, \ldots, p_n\}$, i.e.,

$$V_i = \left\{ x \in E^d \middle| \forall j \,\|\, x - p_i \,\| \leq \| x - p_j \,\| \right\} = \bigcap_{j \neq i} dom(p_i, p_j)$$

where

$dom(p_i, p_j) = \{ x \in E^d \mid < p_j - p_i, x > \leq \frac{1}{2}(\| p_j \|^2 - \| p_j \|^2) \}$ and <,> is a dot product.

Note that dom(p_i, p_j) is the closed halfspace which contains p_i and whose bounding hyperplane passes through the midpoint of the segment [p_i, p_j] and is perpendicular to that segment.

Previous research has proposed many algorithms to build Voronoi diagrams [4][28][35][60][70][78].

For example, the Divide and Conquer algorithm [78] splits data points in half in a plane recursively, and builds the diagram for each half, and then merges those halfspaces together. Because the merge can be efficiently done in time $O(n)$, a running time can be $T(n) \leq 2T(n/2) + O(n)$, i.e., $T(n) = O(nlogn)$. The algorithm uses $O(n)$ space for storing the diagrams.

On the other hand, building a Voronoi diagram with d-dimension can be more complicated than building a Voronoi diagram in a plane. It would require time $\Omega(n^{\lceil d/2 \rceil})$ in the worst case [23][53].

As mentioned above, after a Voronoi diagram is built, the BI is a simple task of searching for the nearest neighbours with different labels in the Voronoi diagram.

3.6.2 Active Learning-based methods

Active learning [15][87] studies how to label data from a massive unlabeled data and little labelled data to help learn a specific induction algorithm. The purpose is to reduce the cost of labelling in machine learning. It can be regarded as an indirect method, which has been applied to the problem of border identification by reducing the region of uncertainty with respect to learners [15].

In more details, active learning [15][87] and IB2 [2] are two classical learner-based methods for border identification. The active learning method works as follows: an ensemble learner is built on an initial small sample from the full training set, and this trained learner outputs borders according

to their uncertainty according to minimum margin or maximum information gain [1][65]; therefore, the identified borders are added into the initial sample for further training the ensemble learner; this process is repeated until a specified condition of termination is met, and the resulting borders can be achieved by the last learner outputting all borders.

The earlier instance selection technique, IB2 and IB3 [2], can be used for identifying borders in terms of misclassifications because these misclassifications are believed to be close to the class boundary while IB3 further removes noise.

The main problem is that these instance selection techniques are only designed for reducing training sets for Instance-Based Learning. IB2 failed to produce effective samples for training decision trees as compared with decision trees built on the full training set [2]. Active_Decorate [65] suffers from the same difficulty as IB2 to produce effective samples for training other classifiers, and authors have expressed that using Decorate [65] to pick examples for training Bagging, AdaBoost, and decision tree will be interesting.

As a result, border points are not directly defined in labelled data. Contrarily, border points are defined according to the decision boundary of a discriminator. Further, there is an essential difference between an active learning –based method and BI_2, i.e., BI_2 is a learner-independent method.

3.6.3 Hybrid_PRS

The Hybrid_PRS, as discussed in Section 2.3.3.1, is a recent technique that produces a small set of prototypes by creaing and adjusting prototypes by learning optimal position with LVQ3.

There are at least several differences between BI_2 and the Hybrid_PRS.

- BI_2 is a nonparametric method while Hybrid_PRS is a parametric method; therefore, BI_2 is more robust than Hybrid_PRS due to a low bias;
- BI_2 is straightforward for identifying borders. The Hybrid_PRS is a post-processing technique. Therefore, Hybrid_PRS is an indirect method;
- Their results are different. BI_2 identifies borders while Hybrid_PRS obtains an approximate set of data points close to the class boundary.
- BI_2 performs instance selection for any classifier while Hybrid_PRS only produces samples for Instance-Based Learning.

As a result, BI_2 is a better learner-independent instance selection method than the Hybrid_PRS.

3.6.4 Remarks

The BI with similarity distance metrics is more direct than the BI with Voronoi diagrams because the latter requires the construction of Voronoi diagram in advance. Especially, in cases of high dimensionality, building a Voronoi diagram is intractable. Although previous practical methods for building Voronoi diagrams in planes can be achieved in loglinear time complexity [78].

Previous research has applied Voronoi diagrams for Instance-Based Learning [23]. For example, the Nearest-neighbour editing algorithm [23] can be used to prune training sets by searching borders points in the Voronoi diagram. The main drawback is that one cannot add training data intelligently later because the pruning step requires knowledge of all the training data ahead of time, and its complexity of $O(d^3 n^{\lfloor d/2 \rfloor} \ln(n))$ [23] is still intractable for many practical applications.

Active learning methods for border identification are biased toward learners. Therefore, the resulting sample with respect to a specific learner cannot be used for other learners. The identified border points are not necessarily the same as those defined in labelled data by using the similarity distance-based method.

Therefore, the similarity distance-based method is preferred to the BI by using Voronoi diagram and active learning-based method.

3.7 Summary

Our research for Border Sampling (BS) is based on our observation that traditional BI suffers from the limitation of identifying a partial border. We illustrate the limitation of traditional BI and propose a new BI technique, called Border Identification in Two Stages (BI_2). As a result, a full border consisting of all near borders and far borders is extracted. The main advantage of BI_2 is that it can identify more informative points than traditional BI.

We show how BI_2 works on two synthesized datasets by using two similarity distance metrics, RBF and Cosine. Furthermore, two related issues are discussed, i.e., BI_2 on multi-class domains and similarity distance metrics for combination of continuous, nominal, and missing values.

BI can be achieved by building a Voronoi diagram or borrowing Active Learning techniques. However, these techniques are not straightforward for Border Identification because they need extra effort to build the Voronoi diagram or to employ Active Learning techniques. BI with similarity distance metrics is regarded as the traditional BI, and is preferable to other BI methods.

There are still several issues, which should be further analyzed.

First, we consider instance selection issues. We wonder if BI_2 technique can be used for sample selection for training classifiers in supervised learning. Border points have high uncertainty for discrimination, and it is believed that they are insufficient for Bayesian Learning theory. Therefore, we develop a novel method to enhance BI_2 for instance selection in supervised learning. We will answer this question in Section 4.2 in Chapter 4.

Second, we consider scalability. Although a method based on BI_2 can be used for instance selection in supervised learning, it is still infeasible in practical applications because it has quadratic time complexity. We should develop a novel method to scale up BI_2 for border sampling. This will be answered in Section 5.3 in Chapter 5 and Section 6.3 in Chapter 6.

Third, we consider the formalization of border identification. Border points are observed and described in this chapter by an example. Border points and redundant points should be strictly defined. A theoretical foundation should be established for further research. We answer this question in Section 5.1 in Chapter 5.

Fifth, we consider the possible application of BI_2 for Class Imbalance Problem (CIP). Previous research [13] has shown that simple sampling techniques, e.g., under-sampling and over-sampling, or synthesized methods, similar to Smote [12], can be used to handle the CIP. This question should be investigated. We answer this question in Chapter 7.

Chapter 4

Progressive Border Sampling

In this chapter, we further develop and enhance BI_2 as a general instance selection technique by incorporating Progressive Learning technique. This allows us to address the high uncertainty for discrimination of the border points selected by BI_2.

4.1 Problem and Discussion

BI_2 is proposed for border identification because it avoids the limitation of the traditional BI technique. The new BI technique is suggested as an alternative effective method for the reduction of training sets. However, border points identified by using BI_2 have high uncertainty for discrimination such that the resulting sampling cannot be sufficiently used for classification according to Bayesian Learning. We need a novel method to enhance the full border identified by using BI_2.

In a straightforward manner, we could identify more borders from the remaining data points by using BI_2 to augment the borders after the initial borders are identified by using BI_2. The main problem is how to define a stopping criterion such that the resulting borders are sufficient for training a successful classifier. A possible method for stopping criterion is to adopt Progressive Learning (PL) such as Progressive Sampling techniques for convergence detection.

Previous Progressive Sampling strategies (PS) [72] were shown to be able to maximize the accuracy of a model by learning on a small sample from the

original large population. Technically, standard PS [72] starts with a small sample and generates progressively larger ones until the model's accuracy no longer improves. The convergence detection is related to a learning curve, created by a base learning algorithm, and represented by a curve plotting accuracy versus sample size.

As a result, we can incorporate BI_2 with PS to learn an effective sample from the original training sets by borrowing the main idea behind the original PS. In detail, we propose a new technique called Progressive Border Sampling (PBS). It uses Progressive Sampling (PS) techniques to progressively learn sufficient borders for discrimination by assuming Border Identification in Two Stages (BI) method proposed in the previous chapter. The new method consists of the following two main aspects.

First, PBS can fully identify the latent border specified by the entire set of labelled cases and extract the data points from the border lying close to the class boundary by assuming BI_2.

Second, a sufficient border can be progressively learned until the convergence is detected by PBS. Therefore, a set of sufficient data points from the border that are considered particularly informative can be used as a new training set, from which a classifier is built by a learner.

PBS has, at least, two salient advantages. First, the data points on the new border generated by PBS help learners build better classifiers more effectively than other data points since PBS can easily detect its convergence to an effective sample. Second, the technique for border identification in the PBS is not necessarily related to any specific learning algorithm. The resultant border can be used for training better classifiers by many learners according to their competence.

4.2 Progressive Border Sampling algorithm

PBS algorithm is proposed to learn an effective sample for training classifiers. It utilizes the progressive learning technique [72], as mentioned above, to iteratively learn a small sample by identifying a full border, as described in Section 3.3, until it converges to an optimal border.

As shown in Figure 4.1, at Step 1, getClasset(D) performs a scan on D to partition the data and put data points into exclusive classes. From Step 2 to Step 11, PBS learns an effective sample by identifying pairwise borders between two classes in a binary way. B_{ij} at Step 3 is denoted as pairwise borders between C_i and C_j, and initialized with empty. C'_i and C'_j are the copies of C_i and C_j, respectively, and C_{ij} is their union used in the test at Step 7. At Step 6, PBS invokes BI_2 to identify a local full border in two stages given C_i and C_j, and the previously generated B_{ij}.

ValidateNBModel() in Step 7 validates B_{ij} by training a NB classifier on B_{ij} and testing it on C_{ij}. Acc describes a learning curve by recording the results of validation with accuracy.

At Step 8, IsConvergence() is used for convergence detection given the current point k and the history of validation in Acc. Its analysis and design are described in Section 4.3. Pairwise borders are defined with the previously generated B_{ij} at Step 9 if convergence occurs. Otherwise, the current B_{ij} is kept at Step 10. At Step 11, we obtain an effective sample B by performing the union of all the pairwise borders B_{ij}.

Because the time complexity of BI_2 is $O(frn_in_j)$ (see Section 3.3), the time complexity of the PBS can be obtained in a straightforward manner by sum of all $O(frn_in_j)$, i.e., $\sum_{i \neq j}(frtn_in_j) = frt(\sum n_in_j - (n_1^2+\ldots+n_c^2)) \leq frt(n^2 - n) = O(frtn^2)$, $(^2n_1+\ldots+n_c^2 \geq n)$, where r is the maximum depth of iteration in the second stage of BI_2 and t is the maximum number of trials in the loop of Step 5 for all local optimal borders in PBS. The value of t depends on the

convergence detection. According to the method defined for convergence at Step 8, empirically, PBS always converges to sufficient borders with a small number of trials. The average number of trials in the given benchmarks datasets from the UCIKDD repository is 4 while the maximum number of trials in PBS is never more than 6 for the resulting sample.

```
PBS algorithm
Input     D: a sample for training with c classes
Output    B
begin
1    B = ∅; C = getClassset(D), C = {Ci | i = 0, …, c}
2    for ∀i, j, where i < j, Ci ≠ ∅, and Cj ≠ ∅
3       Bij = ∅, C'i = Ci, C'j = Cj; Cij = Ci ∪ Cj
4       Acc[k] = 0, k = 0, 1, …, K, K = 100
5       while(true)
6           BI2(C'i, C'j, Bij)
7           Acc[k]=ValidateNBModel(Bij, Cij)
8           if(IsConvergence(k, Acc ))
9              Bij = old; break;
10          old = Bij, k++
11      B = B ∪ Bij
12   return B
end
```

Figure 4.1. PBS algorithm.

```
BI2 algorithm
Input      Ci, Cj: two classes
           Bij: the previously identified border between Ci
and Cj;
Update     Ci, Cj, and Bij
begin
1     Pi = ⋃p∈Cj 1stNN(p, Ci)
2     Pj = ⋃p∈Ci 1stNN(p, Cj)
3     Bij = Bij ∪ Pi, Cj = Cj - Pi
4     Bij = Bij ∪ Pj, Ci = Ci - Pj
5     farBorder(Ci, Bij)
6     farBorder(Cj, Bij))
end
farBorder(D, Bk)
7     D' = D; Bf = ∅;
8     while(true)
9        D' = removeRedundant(D', Bk)
10       if(D' = ∅) break
11       B'f = ⋃p∈Bk 1stNN(p, D')
12       Bk = Bk∪ B'f; Bf = Bf ∪ B'f; D' = D' - B'f
13    D = D - Bf
```

Figure 4.2. BI_2 algorithm in PBS.

4.3 Convergence detection

A base learning algorithm is needed for building a learning curve in Step 8 as shown in Figure 4.1. Naïve Bayes (NB) has a number of advantages and has been used for sampling [49]. We use NB with Gaussian estimator, thus denoted as GNB, instead of NB with Maximum Likelihood estimator, thus denoted as MNB [66] without any loss of generality.

Unlike PS, which builds a learning curve of accuracy, PBS builds a learning curve of Area Under ROC Curve (AUC) [26], as shown in Figure 4.3 and Figure 4.4, for convergence detection because AUC is more stable than Accuracy for evaluation of performance of classifiers, especially on imbalanced domains [26]. As a result, PBS tends to produce a balanced

sample from the original training set for training classifiers such that PBS potentially helps overcome the CIP in some learning tasks.

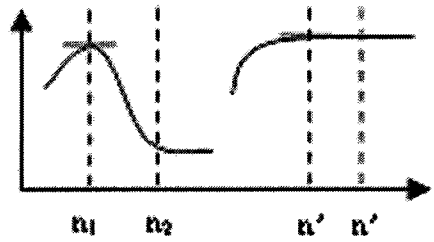

Figure 4.3. Learning Curves for convergence detection.

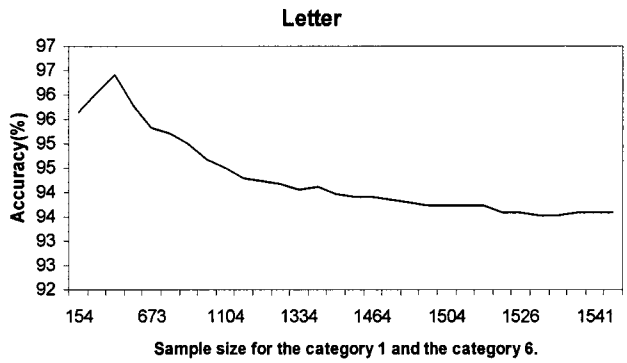

Figure 4.4. A Learning Curve of Naïve Bayes in Letter.

PBS intends to effectively manipulate the progress of the learning curve of Naïve Bayes, which is a linear machine [23][66]. Furthermore, this adaptive learning curve has been shown for PBS on the Letter dataset in the bottom graph of Figure 4.4, and shown for two general situations in Figure 4.3, and it is different from the traditional learning curve obtained with the power law [49][72].

In IsConvergence() at Step 8 of Figure 4.1, we define a rule with two points in a learning curve for convergence detection. That is, $Acc[n_2] - Acc[n_1] \leq 0$. For example, in Figure 4.3, n_2 and n'_2 are two convergent points corresponding to two learning curves.

4.4 Adaptive BI_2

For effectiveness, BI_2 in Section 3.3 has been adapted in PBS by introducing previously identified borders B_{ij} as a new input, as shown in Figure 4.2.

Further, it can be extended by separating B_{ij} into B_i and B_j, as shown in Figure 4.5, and re-writing Step 1 and Step 2 in BI_2 of Figure 4.2. The extended BI_2 in PBS has three inputs. Therefore, augmenting borders belonging to a class are based on the previously identified borders belonging to a different class if it has been obtained rather than only from the remaining data in the different class. Similarly, farBorder(PosData, PosB, NegB) for the second stage in BI_2, where PosData contains all remaining positive data, PosB contains all current positive borders, and NegB contains all current negative border, is used for identifying far borders from PosData with respect to PosB based on previously identified negative borders NegB. The details are omitted due to their simple implementation.

Empirically, the extended BI_2 is more effective than the original BI_2 for augmenting borders in PBS in practical cases.

BI_2 algorithm
Input C_i, C_j: two classes
B_i: the previously identified borders from C_i
B_j: the previously identified border from C_j
Update C_i, C_j, B_i, and B_j
begin
1 if($B_j = \varnothing$)
$P_i = \bigcup_{p \in C_j} 1stNN(p, C_i)$
else
$P_i = \bigcup_{p \in B_j} 1stNN(p, C_i)$
2 if($B_i = \varnothing$)
$P_j = \bigcup_{p \in C_i} 1stNN(p, C_j)$
else
$P_j = \bigcup_{p \in B_i} 1stNN(p, C_j)$
3 $B_i = B_i \cup P_i$, $C_i = C_i - P_i$
4 $B_j = B_j \cup P_j$, $C_j = C_j - P_j$
5 farBorder(C_i, B_i, B_j)
6 farBorder(C_j, B_j, B_i)
end

Figure 4.5. Adaptive BI_2 for Progressive Border Sampling.

4.5 Discussion

There are several comparisons between this new method and the original PS as follows.

First, the new method intends to produce an effective sample by augmenting borders identified by BI_2 while PS progressively learns a small sample from the original large population with an acceptable loss of information.

Second, the new method might be inefficient in large population due to the quadratic time complexity of BI_2 while the PS can be efficient by random sampling with a proper sampling schedule.

Third, both PBS and PS are regarded as learner-independent sample selection methods. The new method assumes BI_2 for sample selection while PS assumes random sampling.

4.6 Pairwise Border Sampling

Border sampling on multi-class domains is not a trivial issue. We need a theoretical discussion about how PBS works on multi-class domains.

The pairwise method is one of class binaritizaion methods for classification on multi-class domains. PBS can assume the same strategy for border sampling on multi-class domains. That is, PBS applies border sampling on each pair of classes, and all resulting subsamples from each pair of classes are unified together as the resulting sample. Because PBS successively builds a Naïve Bayes on each pair of classes, it actually builds a pairwise Naïve Bayes on all pairs of classes.

Previous research has shown that pairwise Naïve Bayes can be reduced to a standard Naïve Bayes while Naïve Bayes with the oa cannot be reduced [83]. This suggests that pairwise border sampling can be an effective method on multiclass domains. Especially, it is suggested that PBS can produce an effective sample for training a successful Naïve Bayes classifier.

4.6.1 Border Sampling for Bayes Learning

We first connect border sampling with Bayesian learning [23] to explain why border sampling can produce an effective sample for training a successful classifier.

Border sampling progressively learns an augmented border from borderlines with high uncertainty to redundant points with high certainty. Provided that we know the probability distribution $P(X)$ on a feature space X,

from the perspective of Bayesian learning, $x \in X$ has high uncertainty for discrimination if x is a border point, which can be identified by BS.

However, if x becomes redundant with respect to c_i, then $P(x|c_i) > P(x|c_j)$, where $i \neq j$. In other words, x has a higher certainty than border points belonging to the class c_i for discrimination. Therefore, x can be correctly classified.

As a result, it is suggested that borders should be augmented by adding new border points with higher certainty than initially identified border points, and augmented border points are sufficient for Bayesian Learning.

4.6.2 Class Binarization Methods

The goal of border sampling is to produce an effective sample from the original population in supervised learning no matter whether it is a binary domain or not. Previous fundamental theoretical results about class binarization methods for classification on multiclass domains provide a direct venue for border sampling on multiclass domains, see Section 4.7. We further discuss about PBS on multiclass domains by assuming class binarization methods as follows.

As a result, the pairwise strategy seems straightforward: border sampling populates border points between each pair of classes; all pairwise borders from each pair of classes are simply combined together. The oa strategy is referred to border sampling between each class and the rest of the class and the resulting borders are also simply combined together.

As a result, our primary motivation is to determine which strategy in border sampling is preferred to another one for training classifiers.

Formally, two class binarization methods, i.e., one-against-one (oo) and one-against-all (oa), for border sampling on multiclass domains can be described as follows.

oo method

It is also called the pairwise method. Border sampling with the oo strategy identifies the pairwise borders on each pair of classes. All obtained $\binom{k}{2}$ pairwise borders are combined together by a simple union as the resulting sample.

The pairwise strategy has been used for border sampling on multiclass domains in PBS algorithm [61][62], which has shown the consistent effectiveness in practical applications in most cases.

oa method

Border sampling with the oa strategy identifies individual borders b_i in each class by identifying a pairwise border b'_i between the class and the rest of classes. As a result, b_i can be obtained by retaining border points in class i out of b'_i. All obtained individual borders b_i, $i = 1,\ldots, k$ are combined together by a simple union as the resulting border.

```
PBS-oa algorithm
Input     D: a sample for training with c classes
Output    B
begin
1   B = ∅; C = getClassset(D), C = {Ci | i = 0, …, c}
2   for i = 0 to c do
3      if(Ci = ∅) continue
4      C'i = Ci, C'j = D - Ci; Cij = C'i ∪ C'j, Bij = ∅
5      Acc[0] = 0, k = 1, …, K, K = 100
6      while(true)
7        B'ij = BI2(C'i, C'j), Bij = Bij ∪ B'ij
         C'i = C'i - B'ij, C'j = C'j - B'ij
8        Acc[k] = ValidateNBModel(Bij, Cij)
9        if(acc[k] ≤ acc[k-1])
10          Bij = old; break;
11       old = Bij, k++
12     Bi = Bij ∩ Ci, B = B ∪ Bi
13  return B
end
```

Figure 4.6. PBS-oa: Progressive Border Sampling with one-against-all.

PBS with the oa, denoted as PBS-oa, can be described as follows.

Similar to the pairwise PBS, PBS-oa has a single input: a training set D. It returns the resulting augmented border points in B. The algorithm is initialized at Step 1, e.g., getClassset() loads data D and performs several basic statistics such as means and variances of variables for calculating nearest neighbors in BI_2 [61][62]. For loop beginning from Step 2 to Step 13 generates an augmented border B. Several variables are initialized. One of which, C_{ij} contains all data in D, and will be used for validation at Step 8 after border points in B_{ij} between C_i and the remaining data are identified at Step 7. Acc[] records a learning curve of Naïve Bayes. The while loop located between Step 6 and Step 11 is used for learning an augmented border. The border B'_{ij} between C_i and the remaining data C_j is identified by BI_2, and added into the augmented border B_{ij}. New border points in B'_{ij} have to be removed from C'_i and C'_j for the next iteration. A learning curve is built further at Step 8 by ValidateNBModel(). If the convergence is detected at the occurrence of a peak on the learning curve, the loop exists. Otherwise, the algorithm continues to augment border points. The learned sub-border B_{ij} between the class i and the remaining data has to be split for B_i which only includes border points contained in the class C_i. The resulting border B is obtained by combining all B_i's by a simple union.

4.6.3 Complexity Analysis

Given a labeled training data D with c classes, where $|D| = n$, $n = n_1 + \ldots + n_c$, for any two class c_i and c_j with n_i and n_j data points, respectively, PBS-oo computes the border points between c_i and c_j. Therefore, PBS-oo has a time complexity of

$$\sum_{i\neq j}^{c} n_i n_j = \sum_{i}^{c} \sum_{j}^{c} n_i n_j = \sum_{i=1}^{c} n_i \sum_{j\neq i}^{c} n_j = \sum_{i=1}^{c} n_i (n - n_i)$$

$$= n^2 - \sum_{i=1}^{c} n_i^2 = O(n^2)$$

For any class c_i with n_i data points, PBS-oa computes the border points between c_i and the rest of data with $n - n_i$ data points. Therefore, PBS with the oa has a time complexity

$$\sum_{i=1}^{c} n_i (n - n_i) = O(n^2)$$

As a result, PBS with either the oo or the oa has the same time complexity.

4.6.4 Pairwise Naïve Bayes for Validation

As discussed above, initially identified border points have high uncertainty, which might be insufficient for classification. Uncertainty can be overcome by progressively learning new border points based on the previously identified border points for an augmented border until this augmented border is sufficient for Bayesian Learning. However, how to decide the progress of augmentation and the condition of convergence to an effective sample is not a trivial issue.

As we know, Naïve Bayes is a simple implementation of Bayes classifiers, and thus it is also a Bayes classifier with the conditional independent assumption. Further, a pairwise Naïve Bayes is equivalent to a standard Naïve Bayes on a multi-class domain [83].

As a result, pairwise border sampling corresponds to pairwise Naïve Bayes classifier built on all pairs of classes from a multi-class domain. Because the pairwise Naïve Bayes built on each pair of classes is equivalent to the standard Naïve Bayes built on the whole training set, the resulting sampling obtained by the pairwise border sampling is effective for training a standard Naïve Bayes classifier. It is believed that the resulting sample also becomes effective for Bayesian Learning.

However, as discussed in Section 4.6.2, Naïve Bayes with the oa is not equivalent to standard Naïve Bayes due to the probability estimation [83]. As a result, PBS-oo seems more desirable than PBS-oa algorithm for Bayesian Learning.

4.7 A Brief Review of Previous Class Binarization Methods

In machine learning, some discriminative models, e.g., support vector machine (SVM), were originally designed for building binary classifiers on binary domains [17][40]. This requires an effective method to customize them for building multiclass classifiers on multiclass domains. Traditionally, there are two *class binarization* methods called one-against-one (oo) and one-against-all (oa) for this customization [30]. They have been successfully applied for either extending traditional binary classifiers from binary domains to multiclass domains or decomposing multiclass domains into an ensemble of binary domains for efficiently training [30].

4.7.1 One-Against-All

The one-against-all (oa) method determines n decision functions that separate one class from the remaining classes $D_i(x)$, $i = 1,\ldots, n$.

For example [17][88], for SVM, let the ith decision function, with the maximum margin or distance between two support vectors belonging to two different classes, respectively, that separates class i from the remaining classes be

$D_i(x) = w_i^t g(x) + b_i$

The hyperplane $D_i(x) = 0$ forms the optimal separating hyperplane and if the classification problem is separable, the training data belonging to class i satisfies $D_i(x) \geq 1$ and if those belonging to the remaining classes satisfy $D_i(x) \leq -1$.

The classification rule is given by

$$y_i = \arg\max_{i=1,\ldots,n} D_i(x)$$

4.7.2 One-Against-One

In one-against-one (oo) or pairwise classification we require a binary classifier for each possible pair of classes and the number of the total pairs is $n(n-1)/2$ for an n-class problem. The decision rule is defined by voting the majority of wins on pairwise classification.

For SVM on multiclass domains [88], the discriminating function for each pair of classes i and j is given by

$$D_{ij}(x) = w_{ij}^t g(x) + b_{ij}$$

where w_{ij} is a weight matrix; $g(x)$ is a map from an input space of x to a feature space; $D_{ij}(x) = - D_{ji}(x)$.

For the input x, we define the classification rule by the voting method called Max wins [30] as follows

$$y_i = \arg\max_{i=1,\ldots,c} D_i(x)$$

where $D_i(x) = \sum_{j\neq i, i=1}^{n} \text{sign}(D_{ij}(x))$.

Or for a tied input with the same votes for different classes

$$y_i = \arg\max_{i=1,\ldots,c} \sum_{j\neq i, j=1}^{c} D_{ij}(x)$$

On the other hand, the Directed Acyclic Graph SVM (DAGSVM) has the same training time as the oo. It uses a rooted binary DAG to define a class in the classification tasks [40][69]. Therefore, there is the advantage that it takes a less test time than the oo. A generalized Bradley-Terry model extends the paired individual comparisons to paired team comparisons [41].

Previous research has shown that with respect to SVM no method can compete with the oo in the training time and no method is statistically better

than the others [40]. However, Naïve Bayes with the oo is different from Naïve Bayes with the oa. The further discussion is followed in the next section.

4.7.3 Pairwise Naïve Bayes and Bayes classification

Some classifiers, e.g., SVM, are originally designed as binary classifiers while other classifiers, e.g., Naïve Bayes and Decision Tree, are directly designed on either binary or multiclass domains.

Our research is involved in a broad application of Naïve Bayes, e.g., Progressive Border Sampling in Chapter 4 and Cascading Customized Couple in Chapter 7. We further discuss Naïve Bayes as follows.

Given a training set with an probability distribution P, in supervised learning, Bayesian learning defines a classifier with a minimized error, i.e.,

$$y_i = c_i = \arg\max_{c_i \in C} P(c_i \mid x) \tag{4.1}$$

According to Bayes theorem, for estimating the right side in (4.1), we have

$$P(c_i \mid x) = \frac{P(x \mid c_i)Pr(c_i)}{P(x)} = \frac{P(x \mid c_i)Pr(c_i)}{\sum_{j}^{|C|} P(x \mid c_j)P(c_j)} \tag{4.2}$$

where the denominator does not change the result in (4.1). Therefore, from (4.2), Bayesian decision can be given by

$$y_i \leftarrow \arg\max_{c_i \in C} P(x \mid c_i)P(c_i) = \arg\max_{c_i \in C} P(a_1, a_2, ..., a_n \mid c_i)P(c_i) \tag{4.3}$$

Naïve Bayes assumes the probabilities of attributes a_1, a_2,…,a_n to be conditionally independent given the class c_i. Therefore, the right side of (4.3) becomes

$$P(x \mid c_i) = P(a_1, a_2, ..., a_n \mid c_i) = \prod_{j=1}^{n} P(a_j \mid c_i)$$

As we can see, a Naïve Bayes is a rudimentary implementation of Bayes classification.

On the other hand, the pairwise classification, which is also regarded as ensemble learning, transforms a multiclass domain with m class into $m(m - 1) / 2$ binary domains. Each binary domain consists of all examples from a pair of classes. A binary classifier is trained on each binary domain. For classification, an observation x is input to all binary classifiers, and the predictions of the binary classifiers are combined to yield a final prediction.

There has been a theoretical discussion about the pairwise Naïve Bayes classifiers, which is related to the pairwise Bayes classifier [83]. A pairwise probabilistic classifier is trained on each binary domain consisting of all examples or instances in either c_i or c_j, denoted as c_{ij}, to estimate probabilities $p_{ij} = P(c_i|x, c_{ij})$ and $p_{ji} = P(c_j|x, c_{ij}) = 1 - p_{ij}$ for voting. It has been shown that the resulting prediction from all binary classifiers by a linear combination of votes is equivalent to regular Bayes classification for class ranking [83].

Specifically, the unbiased weighted voting (wv) and unweighted voting (v) are two linear combination voting methods, and are defined by

$$y_{wv} = \arg\max_{c_i} \sum_{j \neq i} P(c_i \mid x, c_{ij}), \text{ and}$$

$$y_v = \arg\max_{c_i} \sum_{j \neq i} \left[P(c_i \mid x, c_{ij}) \right]_{\geq 0.5}, \text{ where } [x]_{\geq 0.5} = \begin{cases} 1, & x \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

A pairwise Bayes classifier with either the wv or the v methods is reduced to a regular Bayes classifier [83] for class ranking. In general, it is also true for a Naïve Bayes classifier because a Naïve Bayes is a naïve implementation of a Bayes classifier [83].

The oa classification splits a multiclass domain into m binary domains consisting of one class c_i, $i = 1 \ldots m$, from all other classes, and train these

binary classifiers using all examples of class c_i as positive examples and the examples of the union of all other classes $c_j = D - c_i$ as negative examples.

Although the oa classification can also be reduced to a regular Bayes classification, a Naïve Bayes classifier with the oa is not consistent with a regular Naïve Bayes for class ranking because the related probability estimates are not equivalent [83].

4.8 Model-Independence

We discuss the connection between the Border Sampling techniques and some existing classifiers.

Practitioners in Machine Learning have proposed many classifiers to learn the true classification functions, which describe the class boundary defined in a labeled training set. However, due to the time and space complexity of learning, these learners generally have some biases, and then obtain an approximate classification function.

For instance, Decision Trees such as C4.5 have a bias towards a small tree with low depth, and has a decision boundary parallel to the axes [66][74]; Naïve Bayes [23] is biased towards domains verifying the conditional independent assumption, and has a linear decision boundary. Support Vector Machine [89] is regarded as a learner for the maximum margin, and then it is believed that SVM has a lower bias than other learners while it is biased to the defined kernel function. Neverthless, it is also realized that SVM has a higher time complexity for learning than Naïve Bayes and Decision Tree. Instance-Base learning techniques such as IB1 [1] have a bias towards nearest neighbors. The situation can be worse if the distribution is skewed although the errors are tightly bounded [23].

On the other hand, it is desirable that a classifier outputs not only classification results under the zero-one loss function but also class

probability distributions. Therefore, a classifier can perform Bayesian learning by outputting probabilities rather than only decreasing the 1/0 loss.

Several viewpoints are supported and emphasized in this research. Given a labeled training set, the true class boundary has been defined although it is hard to learn due to the sample complexity with respect to the sample size and geometric property. Ideally, we have an optimal classifier, e.g., Bayesian classifier, to fix this true class boundary. Those existing classifiers are subject to a bias with respect to the true class boundary due to the difference between the decision boundary of classifiers and the class boundary of the Baysian classifier. In other words, all existing classifiers are an approximate of the Bayesian classifier and the form of the true class boundary is independent of the existing learners. It is also realized that Border Sampling is model-independent because it learns borders close to the true class boundary, and the resultant borders are used as input to any classifier rather than as a classification model.

Therefore, the Border Sampling technique has little bias to learn the true class boundary of a Bayesian Classifier, and it is model-independent as compared with previously proposed instance selection techiques such as CNN, which has the bias towards the nearest neighbors. The related discussion of the theorectical issue about BS will be introduced in Chapter 5, and our experiments in Chapter 8 will show its effectiveness.

Two synthetic datasets, as shown in Figures 3.2 and 3.3, are used to visualize how BS identifies borders. We also assume the same way to illustrate the proposed new techniques in following Chapters. In general, the resulting sample size and the recursive depth r for far borders in the BI_2 algorithm shown in Figure 3.4 and the number of trial in PBS can be used to describe the sample complexity of the training sets while it is hard to

visualize a real dataset with continuous variables and nominal variables in a high dimension.

It is a little surprising that the proposed instance selection technique has little bias to the true class boundary defined in a Bayesian classifier, and it only has a quadratic time complexity while previously proposed induction algorithms for Baysian learning usually suffer from high time and space complexities. It is also a satisfying result such that the new instance selection helps learn a better classifer.

4.9 Summary

BI_2 is a new technique for border identification on a labeled training set for a supervised learning task. However, initially identified borders have high uncertainty for discrimination, and thus are insufficient for Bayesian Learning. PBS progressively learns augmented borders by borrowing the ideas behind PS techniques. As we can see, PBS is a learner-independent sample selection technique as PS.

We have connected BS with Bayesian Learning to explain why PBS can produce an effective sample for training a successful classifier. The two main issues that we are concerned about are how to define its stopping criterion, and how to perform BS on a multi-class domain.

We have shown that these issues can be solved by building a pairwise Naïve Bayes classifier in PBS, and then defining the convergence condition as the occurrence of the peak of the learning curve obtained from a Naïve Bayes classifier built on each pair of classes.

In fact, PBS corresponds to a pairwise Naïve Bayes on a multi-class domain. It is shown that this pairwise Naïve Bayes is equivalent to a standard Naïve Bayes built on the whole training set. Therefore, the resulting sample obtained by PBS is effective to learn a better Naïve Bayes.

Both PBS-oo and PBS-oa have the same time complexity. The theoretical foundation suggests that PBS-oo is preferable to PBS-oa because the pairwise border sampling is effective while PBS-oa suffers from ineffectiveness due to the probability estimation for training a Naïve Bayes with oa.

Chapter 5

Border Sampling through Coupling Markov Chain Monte Carlo

The Progressive Border Sampling (PBS) is a quadratic learning algorithm, and thus it is still infeasible on large datasets.

In this chapter, our main purpose consists of the following two aspects:

- Formally establish a strict theoretical foundation for border sampling
- Develop a novel technique to scale up border sampling on large datasets according to the theoretical foundation

5.1 Theoretical Foundation

We begin to formalize the Border Sampling technique as follows.

5.1.1 Formal Definitions

Several functions are described as follows.

1NN(p): nearest neighbour function, which defines the single nearest neighbour of a data point p among all data points (the entire domain) according to a distance metric dist();

1NN(p, D'): extended nearest neighbour function, which defines the single nearest neighbour or the *informative data point* of a data point p in a subset D' of a domain D according to a distance metric dist();

$l(p)$: label function, which defines the label of the given data point p;

$C(p)$: a set of data points of the same category as p, denoted as C_p.

A training set contains redundancy, see Section 3.1. According to the discussion in Section 3.1, redundant data points can be defined as follows.

Definition 5.1. Given a labelled dataset D and its subset $B \subseteq D$, any point $p \in D - B$ is a *redundant data point* with respect to B if $p' = 1NN(p, B)$ and $l(p) = l(p')$. A set of redundant points R with respect to B, denoted as

$R(B, D) = \{p \mid \forall p \in D - B, \exists p' \in B, p' = 1NN(p, B) \text{ and } l(p) = l(p')\}$, denoted as R(B), without any confusion.

Definition 5.2. Given a labelled dataset D, the *full border* B of D can be defined recursively as follows:

1) $B = B \cup B_n$, where $B_n = \{q \mid \forall p \in D, \exists q \in D, q = 1NN(p, C_q) \text{ and } l(p) \neq l(q)\}$, called *near border*.
2) $B = B \cup B_f$, where $B_f = \{q \mid \forall p \in B, \exists q \in D, q = 1NN(p, C_q - B - R(B))\}$ and $l(p) \neq l(q)$, called *far border*.

We can show that a redundant data point with respect to the full border B is always near data points of the same category and far from data points of different categories.

Theorem 5.1. Given a redundant point p with respect to B, i.e., $p \notin B$, $p' = 1NN(p, B)$, and $l(p') = l(p)$. We have $\forall q \in B$, $dist(p, p') < dist(p, q)$.

Proof: Follows naturally from the definitions.

Theorem 5.2. Given a redundant point p with respect to a border B, i.e., $p' = 1NN(p, B)$ and $l(p') = l(p)$, we have $\forall q \in D$, $dist(p, p') < dist(p, q)$, where $l(p) \neq l(q)$.

Proof:

1) If $q \in B$, then $dist(p, p') < dist(p, q)$ according to Theorem 5.1.

2) We assume that $\exists q \in D - B$, $dist(p, q) < dist(p, p')$, where $l(q) \neq l(p)$. Further, we assume that $q' \in B$ is an informative point of p and $l(q) = l(q')$, which is defined by Def. 5.1, i.e., $q' = 1NN(p, C_q)$, where $l(p) \neq l(q)$. However, according to Theorem 5.1, $dist(p, q) < dist(p, q')$ and $q \neq q'$. This is contradicted by $q' = 1NN(p, C_q)$.

That is, a redundant point is always close to a border point belonging to the same class and is far from other data points belonging to different classes. In other words, it is inside its class, and is surrounded by its class borders.

The theoretical issues also include the correctness of BI_2.

Theorem 5.3. The correctness of BI_2 is established by Theorem 3.1 in Section 3.3.

5.1.2 Discussion

In Chapter 3, BI_2 is used for identifying a full border [61], i.e., in the first stage, the BI_2 identifies the near border between any two categories. In the second stage, BI_2 iteratively identifies new far borders in the two categories.

For example, a simple XOR function can be visualized by 4 labelled data points in 2D. The BI_2 can identify two near border points and two far border points from the XOR domain while the depth of the recursion for far border points is 1.

The time complexity for searching for the depth of far borders is $O(F)$, where F is the number of attributes, as shown in Section 5.3.3.2. Empirically, the depth of the recursion is bound with a small number («n) in many practical applications.

Because the full border identified by BI_2 is insufficient for discrimination in Bayesian Learning, Progressive Border Sampling (PBS) was proposed in Chapter 4 to progressively learn an augmented full border using the pairwise strategy [61][72] for multi-class domains such that the resulting border points can be used for training in supervised learning tasks [61].

PBS can be equivalent to BI_2 only for full border identification if we ignore the convergence detection step for the obtained augmented full border

obtained, and it can be regarded as a *forward selection* procedure for border sampling.

However, this quadratic algorithm is infeasible for border sampling on large datasets. In this chapter, we use PBS as an *oracle* for border sampling on large datasets by adopting a state-of the art Markov Chain Monte Carlo (MCMC) technique.

Further, according to theoretical foundation described in Section 5.1.1, we have the following fact.

Theorem 5.4. Given a distance metric, BI_2 searches for a complete set of borders in a binary domain.

Proof: according to Theorem 5.2, a redundant point is always close to a border point belonging to the same class and is far from other data points belonging to another class.

Therefore, a data point, which is close to those data points belonging to the same class and far from those data points belonging to the different classes, might be a border, which can be identified by using BI_2 and remains in the resulting borders, or a redundant point, which can be identified by using BI_2 and finally can be removed from the resulting borders according to Theorem 3.1 in Section 3.3. Therefore, BI_2 identifies a complete set of borders. □

5.2 A Brief Review of Coupling From The Past

The standard Markov Chain Monte Carlo (MCMC) is a sampling technique such that selecting the sample $x^{(i+1)}$ only depends on the sample $x^{(i)}$, where the superscript i is a nonnegative integer, and the chain is expected to converge to a stationary distribution π or a invariant measure, which values π_i sum to 1 and satisfy

$$\pi_i = \sum \pi_i p_{ij}$$

where the time-independent matrix p_{ij} describes a probability transferring from the state i to state j.

In general, the MCMC holds two properties: *irreducibililty* and *aperiodicity*. The irreducibility is regarded as the property that any state is reachable from a state in a Markov chain and the aperiodicity is regarded as the property that there is no circle of transitions among all state at regular times in a Markov chain.

The initial convergence time is called the *burn-in* time, which measures how quickly a Markov chain is not biased by the starting point $x^{(0)}$. The mixing rate measures how fast the Markov chain converges. Ideally, the stationary distribution of a good chain is reached quickly starting from an arbitrary position, i.e., it has low burn-in time and mixing rate.

Besides those characteristics defined in the standard MCMC, the Coupling From The Past (CFTP) is an exact sampling technique, which consists of the following three main components [3][73]:

oracle, which is a random map procedure which generates a subsample from the original population;

Composition of maps, which can be used to simulate the flow for many time-steps;

Convergence detection, which is used for ascertaining whether total coalescence has occurred, i.e., the state of Markov chain reaches to the stationary distribution.

The oracle can be used to produce maps f_{-1}, f_{-2}, f_{-3}..., f_{-N}, where N represents how far we have to go into the past, and is determined at run-time. We can define a composite map by $F_{-N}^{0} \stackrel{\text{def}}{=} f_{-1} \circ f_{-2} \circ f_{-3} \circ \ldots \circ f_{-N}$, and the composite map must bring in a *collapse*, which describes a specific and undesired subsample, with respect to some N.

5.3 CMCMC Strategy

We investigate a naïve strategy to incorporate the MCMC with border sampling. We then describe our new method by assuming Coupling From The Past (CFTP) for scaling up border sampling.

5.3.1 A General Strategy

A *general* strategy for the scalability of the oracle, i.e., PBS, on large datasets can be depicted as follows. Given a large training set D and the specified size N of a partition, we can obtain M subsamples, S_i, i = 1,..., M, where M = |D| / N, by stratified sampling. PBS can be executed as BI_2 to identify each local full border B_i on each subsample S_i, and the resulting border is given by $B = \cup_{i=1}^{M} B_i$. We iteratively employ this strategy on the previously generated border B until the iteration process terminates in terms of some stopping criterion.

This strategy is very similar to the ARP_PRS, which is a recursive method for PRS on large datasets, as introduced in Section 2.3.3.2. Theoretically, this method utilizes the Markov Chain Monte Carlo technique (MCMC) [3][71], which performs sampling by constructing a Markov Chain on a large population for an implied integral, i.e., in the oracle, $\int f(x)dx$, where $f(x)$ is defined as a function for estimating border points from a subsample.

Standard stratified sampling techniques are used for reducing the variance of the Monte Carlo estimate. A large population can be partitioned into M subsamples by simple stratified sampling with a specified partition size, called *sampling window*.

Because MCMC iteratively produces successive samples containing border points from the previously identified borders, these successive samples evolve from a large population with many redundant data points to a

small population with fewer redundant data points. Therefore, the MCMC process reduces the number of data points in the identified border.

5.3.2 Coupling MCMC

The naïve strategy for MCMC, as described above, is insufficient to converge to the stationary distribution π because we cannot guarantee the monotonicity of the states, i.e., $P(B^{(i)}) \leq P(B^{(i+1)})$, where i represents the ith state B of the Markov Chain.

Given a labeled dataset D, we can obtain the composite B containing border points identified by the oracle from subsamples defined by a specified sampling window in the naïve strategy. B does not contain sufficient border points while $D' = D - B$ does not purely contain redundant points. The naïve strategy can be used again on D' for new border points. B can be augmented by adding the new border points while D' is reduced by removing the new border points identified from it. As a result, the iterative procedure can cause D' to *collapse* into a specific subsample, which is believed to approximate to a star convex graph. For example, a simple XOR domain is thought to satisfy the collapsing condition for a star convex graph with 4 data points.

Geometrically, a graph S in R^n is called a *star convex* if there exists x_0 in S such that for all x in S the line segment from x_0 to x is in S. For example, the leftmost and rightmost graphs, as shown in Figure 5.1, are star convex.

However, the definition of a star convex graph, from the context of Machine Learning, is not straightforward. We shall give a new definition of a star convex in supervised learning until Definition 5.3.

The collapsing condition can be empirically defined by $c(c - 1) + 2$ by assuming that each pair of classes has at least two border points and one of them is a XOR domain, where c is the number of classes.

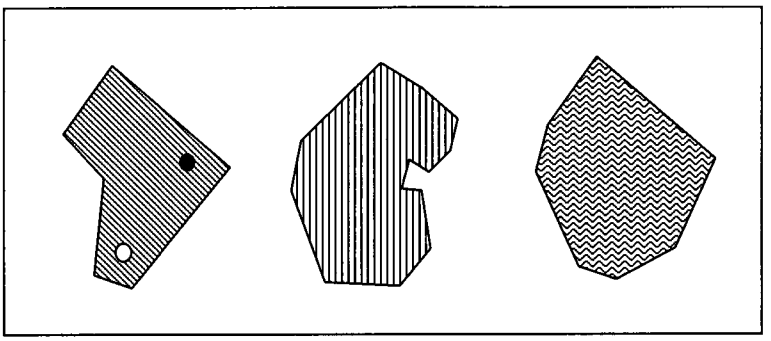

Figure 5.1. Two star convex graphs and a non-star convex graph.

Notes: The leftmost graph and the rightmost graph are star convex while the middle one is not. In the left graph, there exists a data point, e.g., solid ball point, such that for all data points in the graph, e.g., a circle point, the line segmentation connecting with each other remains in the graph. A graph, e.g., the leftmost, is not necessary to be convex.

Therefore, we construct two Markov Chains: the sequence of B for the sets of border points, called *B chain*, and the sequence of D′ for the sets of redundant points, called *R chain*. The iterative procedure can be regarded as a Coupling MCMC (CMCMC) consisting of the B chain and multiple R chains.

As we can see, if the condition of convergence, i.e., $n \leq c(c-1)+2$, in a R chain is satisfied for each local border, the corresponding subsample is empirically believed to be redundant enough as an approximate star convex group, which is defined as follows.

Definition 5.3. In the context of supervised learning, a *star convex* set S in a labelled dataset D is defined as a sample subspace delimited by the class boundary of D, where $\exists x_0 \in S$, such that $\forall x \in S$, the line segment x_0x is *contained* in S if x_0x does not cross the class boundary. A *star convex group* is a group of star convex sets, where each star convex belongs to the same class.

For example, a XOR domain with 4 data points is a star convex group with each point as a star convex.

However, a labelled dataset is not always a star convex group. In the context of Machine Learning, we are interested in some specific star convex sets which have a few border points. In particular, when border points are repeatedly removed from the original training set, the remaining sample may become a star convex group with fewer convex points than before.

Straightforward, we have the following theorem to connect a star convex with border points.

Theorem 5.5. In the context of supervised learning, given a binary domain D, if there are only two border points in D, then D is a star convex group with two star convex sets corresponding to the two classes.

Proof: we immediately know the two border points belong to different classes. $\forall x \in$ D, if x belongs to the class to which a border point x_0 belongs will be redundant, then the line segment x_0x lies in their class because x is close to x and far from data points in another class. Hence, each class is a star convex, and D is a star convex group. □

For illustration, a complicated XOR binary domain [61] is shown in Figure 5.2, where some states of the B chain and the R chain of CMCMC generated by the CPBS algorithm (see Section 5.3.3) are depicted.

Ordering Figure 5.2 from top to bottom and from left to right, we have

(a) Given a complicate XOR binary domain with 640 data points, the sampling window for stratified sampling is set to 100. The original dataset is the beginning state of the B chain, and it becomes the beginning state of the first R chain;

(b) A state of the first R chain;

(c) The state of the first R chain before the collapsing condition is satisfied;

(d) The collapsing state; the ending state through 7 states of the first R chain;

(e) The new (second) state of the B chain, which has 485 data points less than the original 640 data points, thus becomes the beginning state of the second R chain;

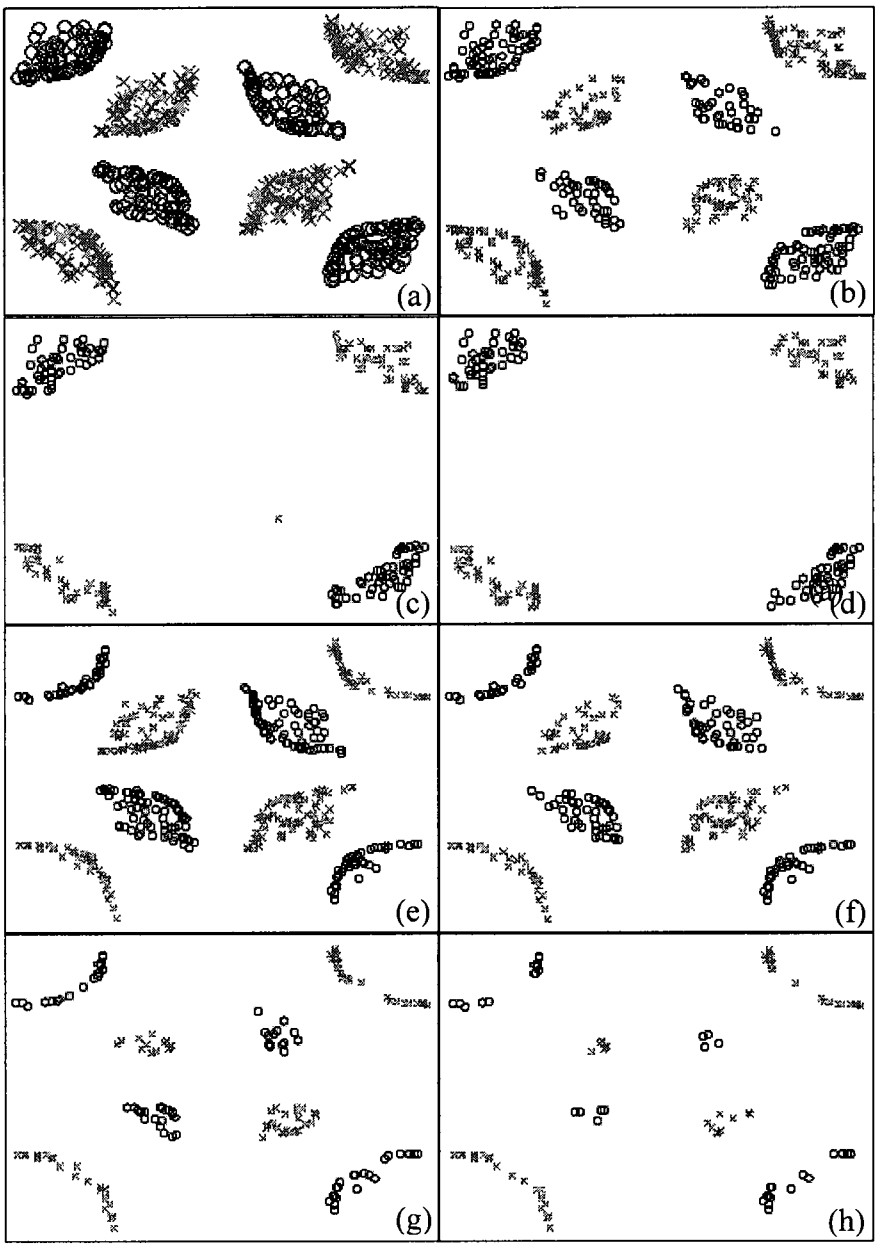

Figure 5.2. Two successive R chains of the CMCMC.

Figure 5.3. The third state of the B chain corresponding to h) in Figure 5.2.

(f) A state of the second R chain;

(g) The state of the second R chain before the collapsing condition is satisfied;

(h) The ending state after 4 states of the second R chain.

Correspondingly, the third state of the B chain is obtained, as shown in Figure 5.2, which is coalescent to (e), and leads to the occurrence of convergence. The resulting sample corresponding to (e) is returned through 3 states in B chain and 11 states in two R chains of the CMCMC.

As we can see, the divide and conquer strategy exhibited by this new technique produces an algorithm which can converge to an ideal sample by using a small sampling window.

5.3.3 CPBS Algorithm

According to the above discussion, we first propose a new method, called Coupling Markov Chain Monte Carlo-based PBS (CPBS), for border sampling on large datasets. Given its two inputs of the algorithm, the training set D and the sampling window W, it returns the result in Border. The algorithm produces the B chain in the while loop from Step 2 to Step 6 while the R chain is generated in the while loop from Step 11 to 21.

A linear machine, Naïve Bayes (NB), is used as a base learner for convergence detection of the B chain at Steps 4 and 5. NB has been

successfully used for progressive learning for convergence detection [49][61], i.e., ValidateNBModel() is used for building a NB classifier for estimating the current sample D', and the downward slop or the initial points of the plateau of the generated adaptive learning curve saved in LearningCurve is used as a convergent point.

```
CPBS algorithm
Input D: a training set, W: a specified window size
Output Border
begin
1    Border = ∅, i = 0..K, D' = D, LearningCurve[0] = 0;
2    while(true)
3       D' = Coupling(D', W)
4       LearningCurve[i+1] = ValidateNBModel(D', D)
5       if(LearningCurve[i+1]  ≤ LearningCurve[i])
        break;
6       i++
7    Border = D'
8    return Border
end
Procedure Coupling(D', W)
9    c = Number of class in D', cg = false
10   if(c ≤ 5) cg' = true else cg' = false
11   while(true)
12      B = ∅, N = ⌊|D'| / W⌋
13      S = StratifiedSampling(D', W), |S| = N
14      if(N = 1) cg = cg';
15      S' = ∅, collapsing = true
16      for(k = 0; k < N; k++)
17        Bk = PBS(S(k), cg), B = B ∪ Bk
18        S'  = S' ∪ (S(k) - Bk)
19        if(|Bk| > c(c - 1) + 2)) collapsing = false
20      if(collapsing ∨ cg) break;
21      D' = S'
22   return B
```

Figure 5.4. CPBS algorithm: Coupling Markov Chain Monte Carlo for scaling up Progressive Border Sampling.

The Coupling procedure generates a R chain in the while loop beginning at Step 11 while initially the condition of the forward selection of PBS is defined at Step 10. The floor function is used for specifying a sampling window at Step 12, and the stratified sampling technique helps reduce the

variance of MCMC at Step 13. PBS is used as an oracle for identifying local full borders at Step 17, and the algorithm tests the collapse of the R chain at Step 19 according to the collapsing condition. Initially, the Boolean variable cg, which is used for determining whether PBS performs convergence detection, is false at Step 9. It leads that the oracle performs as the BI_2 only for full border identification at Step 17. Because S will be shrunken at Step 18 in the while loop, when S is fitted in the sampling window W, PBS performs the forward selection for searching for an augmented full border.

5.3.3.1 Similarity measures

Generally, any distance metric or similarity measure can be used in the oracle of the CPBS for searching for border points, e.g., Radial-Based Function (RBF), Cosine, Euclidean distance or normalized Euclidean distance, Pearson Coefficient, Mahalanobis distance, and Extended Jaccard similarity [80], etc. However, their different effects have been observed in the oracle, e.g., RBF is biased to the class contour while Cosine is biased to the class core [61]. Instead of developing an ideal similarity, we empirically show that RBF helps the Monte Carlo integration in CPBS in most cases by conducting experiments.

5.3.3.2 Linear time complexity

Clearly, the space complexity of the CMCMC-PBS linearly increases. We analyze its time complexity as follows. Considering the two while loops in the CMCMC-PBS, the time complexity can be first simply given by $O(T\times K\times N/W\times C_0)$, where T is the number of tries for convergence detection in the while loop beginning at Step 2; K is the number of iterations in the while loop beginning at Step 11 in Coupling; N is the size of a given training set D and W is the sampling window; C_0 is the time complexity of the oracle

with the sampling window W, and $C_0 = O(T_0K_0FW^2)$, where F is the number of features; K_0 is the depth of the recursive far borders; T_0 is the number of tries for convergence detection [61]. Therefore, the time complexity of the CMCMC-PBS is given by

$$O(TT_0KK_0FWN + TFN) = O(TT_0KK_0FWN) \quad (5.1)$$

where the term O(TFN) [23][66] is the time complexity for learning a NB in the CMCMC-PBS. T_0 and K_0 are empirically analyzed in previous research by a bound with a small number («n) given a domain [61]. Especially, $T_0 = 1$ if the oracle runs as BI_2.

Because BI_2 assumes a pairwise strategy for border identification on multi-domains, K_0 has nothing to do with the number of classes.

As a result, the extended non-redundant XOR with 8 data points in 2D, as shown in Figure 5.4, is constructed as the worst case. The depth of the recursive far borders is 3. Further, we obtain an upper bound, i.e., $K_0 \leq 2F - 1$, by a constructed XOR of dimension F. It is just equal to the size of the boundary of an F-cube minus one. Empirically, K_0 is much smaller than F [61].

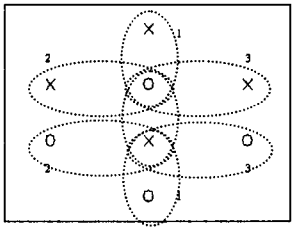

Figure 5.5. The extended XOR with 8 data points in 2D.

Notes: BI_2 first identifies two near borders located at the center of the graph. In the first recursion, it identifies two far border points indicated by the oval 1 based on the previously identified near borders; in the second recursion, it identifies two more far borders indicated by the oval 2 based on previously identified borders; in the third recursion, it identifies the last two far borders indicated by the oval 3.

On the other hand, Coupling searches for redundant points on the entire dataset by the oracle until a collapsing occurs. K is a little domain-related, e.g., redundancies, while it is related to collapsing test. But its small value («n) has been observed.

According to Eq. (5.1), CPBS is an efficient learning method in linear time complexity with respect to the sample size N for border sampling.

5.3.3.3 Convergence detections and collapsing

Naïve Bayes is used for convergence detection for the B chain from Step 2 to Step 6 of the CPBS algorithm. Empirically, it is not always effective to track the adaptive learning curve of this linear machine for convergence detection if random sampling is used. However, it has been shown that the effectiveness can be obtained by precisely border sampling [61].

If the oracle runs for forward selection with convergence detection by setting cg = true at Step 14, $T_0 > 1$. As a result, CPBS performs two convergence detections (T times for the CMCMC in the B chain and T_0 times in the oracle) and one collapsing (K times for the R chain).

The two convergence detections can be interactive somewhat. In some cases, the oracle's convergence detection can lead to the reduction of the CMCMC's convergence detection. We also emphasize that the CMCMC's backward elimination is more efficient for convergence than the oracle's forward selection in multiclass domains with high dimension.

Empirically, if the state in the R chain is fitted in the sampling window and the number of classes ≤ 5, the oracle performs convergence detection, e.g., in Figure 5.2.

5.3.3.4 Learning measures

The Area under ROC curve (AUC) is used for assessing the ranking in terms of separation of the classes while the ROC curve is drawn for rankers [26]. The AUC has been used for evaluating the performance of classifiers on class imbalanced domains because it is more stable than other learning measures, e.g., accuracy, for evaluation of performance of classifiers [26]. Therefore, the AUC is suggested as a learning measure for border sampling. As a result, the adaptive learning curve of NB is an AUC curve.

5.4 Comparison between CPBS and ARP_PRS

The ARP_PRS, as discussed in Section 2.3.3.2, is a recursive method, which is first introduced to scale up PRS on large datasets. As we can see, both CPBS and ARP_PRS assume the divide-conquer strategy to scale up the original base algorithms for instance selection on large datasets. Despite this, there are at least several critical differences between CPBS and ARP_PRS as follows:

- CPBS is a simple partition and iterative process while ARP_PRS assumes a tree-structure search; therefore, CPBS is more efficient than ARP_PRS because CPBS is not recursive and does not need much memory to trace the recursive process;
- CPBS promises the algorithm convergence to an effective sample while ARP_PRS does not promise the resulting sample to be effective; this is because ARP_PRS does not guarantee that only those useless data points are removed from each smaller subset, which is resampled from the original entire sample. Therefore, the ARP_PRS suffers from ineffectiveness of the resulting sample, which is obtained only by combining all smaller subsets;

- CPBS has linear time complexity while ARP_PRS is recursive with a higher time complexity.

5.5 Other Related Work

We propose CMCMC by adapting CFTP into border sampling. PBS is suggested as our oracle, and a state in the B chain corresponds to a composition map defined by those states in the related R chain. Collapsing can be observed by testing the occurrence of some star convex graph from the states of the R chain. A linear machine, i.e., Naïve Bayes, is assumed for convergence detection for the B chain.

Given a labelled dataset D, based on 1NN(.,.) and 1NN(.), respectively, $B_1 = \{q \mid \forall p \in D, \exists q \in D, q = 1NN(p, C_q)$ and $l(p) \neq l(q)\}$ and $B_2 = \{q \mid \forall p \in D, \exists q \in D, q = 1NN(p)$ and $l(p) \neq l(q)\}$ are not equivalent. As a result, 1NN(.) is subject to failure for defining a border while 1NN(.,.) should be used to define the near border. This observation is used for explaining the main difference between the border sampling techniques shown in BI_2 and the techniques for the reduction of training sets in those algorithms by the nearest neighbour editing rule [23], whose purpose is the reduction of training sets for Instance-Based Learning. We emphasize that the reduction of training sets should be one of the tasks of border sampling.

Furthermore, because some learners, e.g., Naive Bayes and Decision Tree, etc, can be very fast at learning a good classifier even though a large training set contains ten thousand examples, and some domains require sufficient examples to define their target concepts, reducing the sample size by simply selecting a small sample is not expected to reduce the computational cost without loss of performance. On the other hand, we claim that our current research for border sampling in supervised learning can be used for reducing the learning cost in semi-supervised learning and active learning.

Another related work is incremental learning, which can be regarded as a learning method for building a theory from examples available over time [34][81]. For example, incremental SVM for an online application has been studied [59]. Clearly, the CPBS can be easily used for incremental learning by suggesting and focusing on incremental sampling.

Assuming some optimal probability distribution, the Bayesian decision rule can define a Bayesian decision boundary by using a discriminating function $g_i(x) = p(x|w_i)p(w_i)$ [23]. Our research suggests that the new method tends to learn optimal class conditional probability distributions $p(x|w_i)$ by border sampling such that the related prior probability distribution $p(w_i)$ and Bayesian decision boundary can be obtained.

Because CPBS has a useful bias toward border data points lying close to the class boundary, it can produce an effective sample for training. Therefore, it also provides a promising treatment for tackling the class imbalance problem as compared with previous techniques for under-sampling and oversampling [26][55] by producing a possible balanced sample and by helping learn a successful classifier.

5.6 Summary

Recent research has focused on learning tasks applied to large datasets [49][72]. However, there exist some vital drawbacks in that research. For example, within the classification branch of machine learning, Progressive Sampling techniques (PS) [49][72] are subject to a failure in converging to an optimal sample with a bias toward a base learner. Active learning techniques or semi-supervised learning [15][21][87] suffer from the same difficulty as PS to converge to an optimal sample displays high bias toward the selected learner. Conversely, we believe that reducing the variance of the

data due to redundancies can help reduce the learning cost without loss of performance.

As a result, we incorporate CMCMC with PBS for border sampling on large datasets, and propose a new approach, called Coupling Markov Chain Monte Carlo-based for scaling up Progressive Border Sampling (CPBS), in which two interactive Markov chains, called the B chain for border points and the R chain for redundant data points, are defined according to CMCMC techniques, and the convergence detection for the B chain and the collapsing condition for the R chain are analyzed.

There are three main advantages to CPBS. First, it is independent of inductive algorithms as PBS itself. Therefore, it can learn an optimal sample by reducing the variance of the data due to redundancies from the original large population. Second, CPBS is a linear algorithm and can be efficient to converge to an effective sample by many small samples with a rapid mixing time related to CMCMC techniques. Therefore, it can be feasible to be used in practical applications. Third, CPBS is not restricted to being used in either small datasets or large datasets because it is not sensitive to the sampling window. At an extreme case, the whole training set is fitted in the sampling window.

We have established a theoretical foundation of the border sampling technique. There are still some issues needed to further our analysis:

- What are the pros and cons of convergence detection in CPBS by testing the collapse to a star convex group ?
- Can the border sampling technique be used for instance selection in a supervised learning task ?

These issues will be discussed overall in Chapter 6.

Chapter 6

Border Sampling through Tripled Markov Chain Monte Carlo

The crucial issue for the success of CPBS is its algorithmic convergence to an effective sample, especially on large datasts. Because it has been observed that CPBS suffers from a little failure in a few cases according to our initial experimental results, we further develop the Border Sampling technique on large datasets by introducing an alternative method for algorithmic convergence.

6.1 Problem and Discussion

CPBS implements the basic idea behind CFTP for fast border sampling from the original labelled training sets by creating two coupled Markov Chains, i.e., the B chain and the R chains. The main difficulty is how to define a convergence condition for R chains in CPBS.

We discuss the pros and cons of the algorithmic convergence of CPBS by using a synthesized dataset. Generally, CPBS can be described as follows.

First, a domain D is partitioned into many subsamples by stratification. CPBS identifies local borders on these subsamples by invoking the oracle PBS, which initially acts as BI_2, in its Coupling procedure; the composite of all local borders become a current full border B_i and a current redundant set R_i can be obtained by removing B_i from D. However, B_i does not contain purely border points while R_i does not contain purely redundant points. The trick is that the above process is repeated with R_i rather than B_i at the next

step, i.e., the process makes R_i to evolve into a state with higher redundancy than previous states;

Second, $D' = R_i$ can be a new domain substituting for the initial D or previous D′, and the process is repeated from the first step; therefore, D′ becomes further redundant with high probability until convergence occurs, i.e., $n \leq c(c-1)+2$ for all local borders [62], where n is the size of a local border and c is the number of classes in D, if we assume that each class contains a minimum of two data points. The resulting D′ tends to contain purely redundant points and $D - D'$ is regarded as the identified border B, which is returned to at the next step. The D′s constitute a R chain for redundant points.

Third, B still contains redundancies. The process is repeated with B substituting for the original D from the first step until a convergence occurs due to the descent of Naïve Bayes learning curve on all Bs. The Bs constitute a B chain for border points.

A large sample can always collapse into a star convex group consisting of the remaining redundant data if we continuously remove border points from the sample. However, the contrary is not true, i.e, redundant data points will not necessarily be formed into a star convex group. CPBS can be illustrated in the following example, which show more details than the example, which is shown in Figure 5.2, see Section 5.3.2.

Example 6.1. Given a synthesized dataset D with 8 classes, as shown in Figure 6.1, the algorithm, by setting a sampling window size W = 100 and a distance measure RBF, first produces redundant data points R_{11} from D by invoking Coupling, which directly satisfies the condition of convergence for R chain due to many redundant points, and which is believed to be sufficiently redundant. B_1 is obtained by removing R_{11} from D. Because B_1 does not satisfies the condition of convergence for B chain, the algorithm

continues to produce redundant data point R_{21}, R_{22}, and R_{23} from B_1 by invoking Coupling again until the condition of convergence for R chain is satisfied at R_{23}. B_2 is obtained by removing R_{23} from B_1. The condition of convergence for B chain is satisfied at B_2 due to the descent of the learning curve of Naïve Bayes on D, B_1, and B_2. B_1 is returned as the resulting sample B. The evolution of sample size is shown in Table 6.1 for explanation.

For example, in Table 6.1, $|B_1| = |D| - |R_{11}|$ because the first R chain converges to R_{11}, and $B_1 = D - R_{11}$. $|B_2| = |B_1| - |R_{23}|$ because the second R chain converges to R_{23}, and $B_2 = B_1 - R_{23}$. $B = B_1$ because B chain converge to B_1.

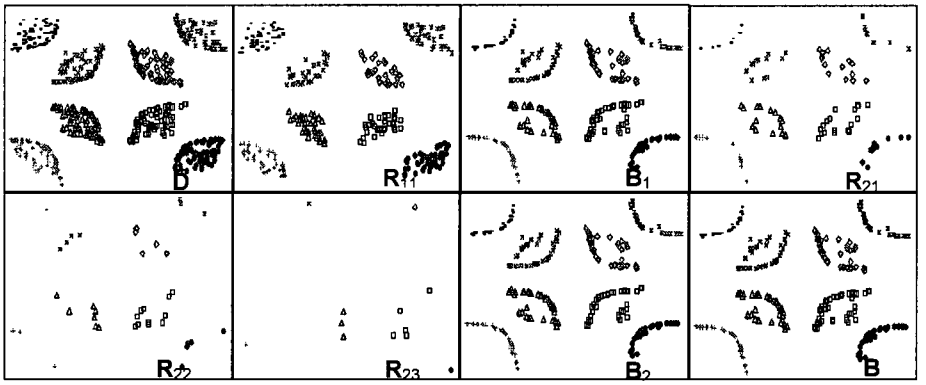

Figure 6.1. Border Sampling on a synthesized data by using CPBS.

Table 6.1. The evolution of sample size in coupled chains.

D	R_{11}	B_1	R_{21}	R_{22}	R_{23}	B_2	B
640	322	318	109	41	12	306	318

- R chain. All R_{ij}'s constitute the *i*th R chain. In this example, R_{11} and R_{23} are empirically computed as an approximate star convex group because only each subsample in the sampling window is tested as a star convex according to the condition of convergence for R chain, i.e., $n \leq \tau = c(c - 1) + 2$. The main problem is that redundant data points are not necessary to constitute a star convex group although a dataset can be collapsed to it by unceasingly removing border points from previously remaining data.

- B chain. The original domain D and all B_i's constitute B chain. The convergence point in B chain is defined by the peak of the learning curve of Naïve Bayes built on D and all B_i's. However, the convergence detection in B chain suffers some difficulty to augment border points for further optimization after the initial the convergence is detected.

6.2 Effective Geometric Computation

We introduce a new method for convergence detection in the R chains of CPBS. The method follows the propagation of nearest neighbours in border sampling.

We define the *geometry* of a data point as its nearest neighbours belonging to another class. The nearest neighbours of a data point from a subsample which is obtained from a large population might be different from those nearest neighbours of the data point from the large population. Therefore, the geometry of a data point is divided into *local* and *global* geometries with respect to subsamples and the original large population, respectively.

That is, the nearest neighbours of a data point from other classes in the original large population are considered as the global geometries. The nearest neighbours of a data point from other classes in a subsample of the population are considered as the local geometries. It is believed that the local geometry can evolve towards the global geometry fast by assuming a novel strategy, which can be formalized as follows.

6.2.1 Basic Definitions

The following concept is used for encapsulating the geometric information of data.

Definition 6.2. Given a set of data points S, the *geometry* of a data point $p \notin$ S with respect to S is defined as its nearest neighbour from S that comes from a category different from p's.

If we apply resampling to a large population T, the geometry of a data point p in a subsample of T is different from the geometry of p in T. Consequently, the geometries are divided into two categories: *local* and *global*. Further, a global geometry is a border point or a near border point according to the definition of a full border [62].

Given a subsample S of T, the local geometry of a data point with respect to S can be directly obtained from S. However, it might not be the global geometry of the data point. After the local geometry is computed from S, its global geometry can be dynamically updated if its local geometry is closer to p than the current global geometry is. However, if two data points with different classes are sufficiently far from each other, their global geometries are not expected to need updating.

There are two crucial assumptions related to the geometries of data points for border sampling.

Firstly, if B is a full border identified from a subsample S, then the remaining data points obtained by removing B from S, i.e., S – B, become further apart from those remaining data points belonging to different classes. As a result, their global geometries are not expected to get updated later;

Secondly, if the global geometries of data points are not updated any more, then it is believed that all data points are sufficiently far from those data belonging to different classes.

The above discussion suggests an effective geometric computation related to the propagation between a local geometry and a global geometry.

6.2.2 Geometric computation

Technically, the geometry of a data point consists of several components, as shown in Table 6.2. Geometric information is expected to be evolved and updated by a resampling technique from a local geometry to a global geometry. For example, if a localSimilarity is greater than the corresponding globalSimilarity, the globalNeighbour and globalSimilarity should be updated.

The GeometricPropagation procedure, as shown in Figure 6.2, is used for the geometric computation of a geometry, i.e., it is used for updating the geometry of p_1 given p_2. To implement the geometric computation in border sampling, it is sufficient to consider the 1-Nearest Neighbour procedure, i.e., 1stNN(.,.), in the PBS [61][62] because it is involved in the geometric computation. Hence, the original 1stNN(,.,) can be modified by incorporating GeometricPropagation with it, as shown in Figure 6.3.

As we can see, the GeometricPropagation procedure in getNearestNeighbour algorithm, as shown in Figure 6.3, is used for updating the geometry of p_1. If p_1 becomes further redundant, i.e., p_1 and p_2 are sufficiently far from each other, the geometry is not expected to be updated. Therefore, the first assumption finally leads to no geometry to be updated among all redundant data.

Table 6.2. The components of the geometry of a data point.

Components	Description
localNeighbour	the local geometry of a data point p
localSimilarity	the similarity between p and its local geometry
globalNeighbour	the global geometry of a data point p
globalSimilarity	the similarity between p and its global geometry

```
GeometricPropagation
input:    p1, p2: two data points
          s: similarity
output: updated
begin
     updated = false
     if(p1.localSimilarity < s)
        p1.localSimilarity = s
        p1.localNeighbour = p2
        if(p1.globalSimilarity < s)
           p1.globalSimilarity = s
           p1.globalNeighbour = p2
           updated = true
end
```

Figure 6.2. Geometric Propagation between two data points.

```
getNearestNeighbour
input:    p, a data point
          S, a subsample with the label different from p
output:   p0, the nearest neighbor of p from S
          updated, a Boolean variable
begin
     s1 = - ∞
     if(S = ∅)
        return null
     for each pi ∈ S
        s = getSimilarity(p, pi)
        if(s > s1)
          s1 = s;  p0 = pi
        if(s = 1)
          updated = GeometricPropagation(p, pi, s1);
          return p0
     updated = GeometricPropagation(p, pi, s1);
     return p0
end
```

Figure 6.3. The modified 1stNN procedure.

6.3 TPBS algorithm

We improve CPBS through the effective geometric computation, as shown in Figure 6.4. The new algorithm, called Tripled Markov Chain Monte Carlo for scaling up PBS, denoted as TPBS, initializes its variables at Step 1. The while loop starts from Step 2 to Step 7 for learning an augmented border B.

The BorderIdentification algorithm (see Figure 6.5) at Step 3 is executed to identify a new border B_i from D′. B is augmented by union with B_i at Step 4. Naïve Bayes is used for estimating B by building a Naïve Bayes classifier and testing on D at Step 5. The occurrence of a plateau in the learning curve is defined as the convergence point at Step 6. Otherwise, a new dataset D′ is generated by removing B_i from the previous D′ at Step 7 and the while loop continues.

BorderIdentification is an algorithm with three parameters for border identification on large datasets. W is the only parameter specified by the users of TPBS, and it is the same as that in CPBS [62]. From Step 2 to Step 8, BorderIdentification repeatedly uses the GCoupling procedure to progressively learn a border until the condition of convergence occurs at Step 6. If no convergence occurs, the identified border becomes a new dataset at Step 7, which will incur further removal of the redundancies at the next iteration. The resulting border is returned at Step 9. All identified borders constitute a B chain approaching a precise border.

GCoupling with two inputs is an improved Coupling procedure with respect to previous CPBS, which assumes the effective geometric computation. After the initial variables are set at Step 1, GCoupling defines a R chain for redundant data points in the while loop from Step 2 to Step 11. The Stratification procedure generates strata by partitioning D′ into N subsamples with sample propagation at Step 3.

```
TPBS algorithm
Input D, W
Output B
begin
1    B = ∅, i = 1..K, D' = D, LCurve[0] = 0;
2    while(true)
3       B_i = BorderIdentification(D', D, W)
4       B = B ∪ B_i
5       LCurve[i] = ValidateNBModel(B, D)
6       if(LCurve[i]  ≤ LCurve[i - 1])
          B = oB, break;
7       D' = D' - B_i, oB = B, i++
8    B = oB
9    return B
end
```

Figure 6.4. TPBS algorithm.

```
BorderIdentification algorithm
Input     D, a training set
          Test, test data
          W, the specified sampling window
Output B, a new border
begin
1   B = ∅, i = 1..K, D' = D, LCurve[0] = 0;
2   while(true)
3      B = GCoupling(D', W)
4      LCurve[i] = ValidateNBModel(B, Test)
5      if(LCurve[i] ≤ LCurve[i - 1])
6         break;
7      D' = B,
8      i++
9   return B
end
```

Figure 6.5. BorderIdentification algorithm.

In the for loop from Step 5 to Step 9, the algorithm identifies local border points from the subsample of each partition by using the oracle PBS, and then produces redundant points by removing local border points from the subsample. All redundant points are saved in S′. If there is no geometry to update on subsamples from all partitions at Step 10, the while loop exits and the algorithm converges to this point according to the second assumption. Otherwise, the algorithm sets a new dataset D′ for S′. Because S′ shrunked

due to border point removal, and the geometry of the data is effectively computed and updated in the while loop by invoking the oracle PBS, the condition at Step 10 can be always met in a limited number of iterations due to the shrinkage of S′ and the first assumption. At Step 13, the algorithm returns the resulting border, which is obtained by removing redundant data in S′ from the original large population D at Step 12.

```
GCoupling algorithm
input   D, training set
        W, the specified sampling window
output  B, identified border
1   D' = D, B = ∅
2   while(true)
3      S = Stratification(D', W),
                |S| = N, N = ⌊|D'| / W⌋
4      S' = [1..N], updated = false
5      for(k = 0; k < N; k++)
6         locUpdated = false
7         B_k = PBS(S[k], locUpdated)
8         S'[k]  = S[k] - B_k
9         if(locUpdated) updated = true
10     if(!updated) break
11     D' = S'
12  B = D - S'
13  return B
```

Figure 6.6. GCoupling algorithm.

6.3.1 Illustration

Given the synthesized data D, as shown in Figure 6.1, we illustrate, as shown in Figure 6.7, how the TPBS achieves border sampling on the synthesized data.

First, GCoupling successively produces a group of samples related to redundant data points, R_{111}, …, R_{114} (R_{114} is, actually, omitted, in Figure 6.7), and R_{115} , i.e., those S's, which are generated for redundant data at different repetitions in GCoupling, the initial dataset until the condition of convergence occurs, i.e., no propagation of nearest neighbours, and B_{11} = D

– R_{115} is returned into BorderIdentification. B_{11} still possibly contains redundancies. Therefore, BorderIdentification in Figure 6.6 invokes GCoupling again with B_{11} as input for further border points B_{12} and B_{13} until the convergence occurs, i.e., the descent of the learning curve of Naïve Bayes on B_{11}, …, B_{13}, and B_{13} is returned as A_1 in TPBS.

GCoupling corresponds to Coupling in the original CPBS. GCoupling also establishes two coupled Markov Chains, R = $\{R_{111}, R_{112}, R_{113}, R_{114}, R_{115}\}$ and B = $\{B_{11}, B_{12}, B_{13}\}$ while the condition of convergence of R is related to an effective geometric computation: no propagation in TPBS rather than collapsing to a star convex in the original CPBS.

Further, the resulting border points A_1 identified by BorderIdentification may contain high uncertainty, the algorithm in Figure 6.5 invokes BorderIdentification again to augment border points by identifying new border points, e.g., A_2 and A_3, from the remaining data points after removing the previously identified border points, e.g., A_1 and A_2. This progressive learning process in TPBS is repeated until the condition of convergence occurs, i.e., the descent of the learning curve of Naïve Bayes built on A_1, A_2, and A_3. A_2 is returned as the resulting sample because the descent point is A_3.

The evolution of the sample size of the related subsamples is reported in Table 6.3. As we can see, the sample size in either the R chain or the B chain decreases due to the removal of either non-redundancies or redundancies, respectively. Further, the sample size in all As increases due to the augmentation of the border.

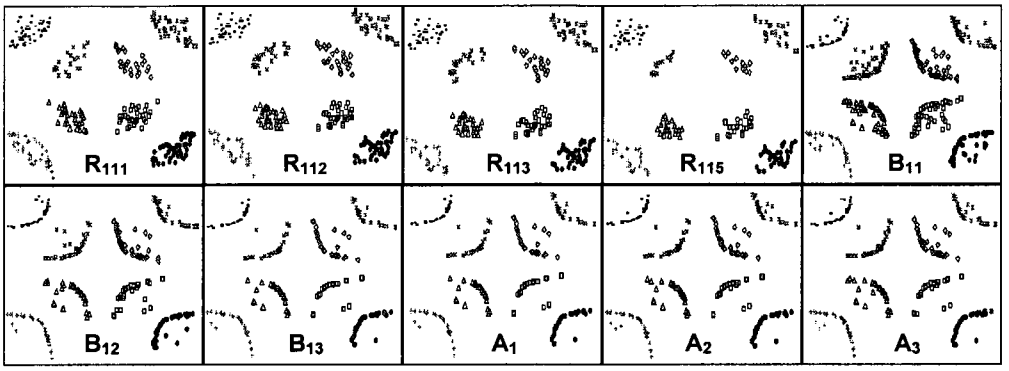

Figure 6.7. Border sampling on the synthesized data by using TPBS.

Table 6.3. The evolution of the sample size in the tripled chains.

R_{111}	R_{112}	R_{113}	R_{114}	**R_{115}**	B_{11}	B_{12}	**B_{13}**	A_1	**A_2**	A_3
314	288	259	226	203	437	304	238	238	266	298

6.3.2 Tripled Markov Chains

We can show three interactive Markov Chains built in the new algorithm. An R chain: a group of samples, e.g, R_{111}, ..., R_{115}, produced in GCoupling for redundant data points. Its condition of convergence is that no propagation of nearest neighbours occurs in all local subsamples. There is an assumption that all the global geometries are sufficiently computed such that all border points consisting of near borders and far borders are obtained if the convergence is detected. A B chain is a group of samples, e.g., B_{11},..., B_{13}, produced in BorderIdentification by further identifying border points; An A chain is a group of samples, e.g., A_1,...,A_3, produced in TPBS by augmenting border points.

On the other hand, the convergence detection of the R chain is achieved by detecting the propagation of nearest neighbours; the convergence of either the B chain or the A chains is achieved by detecting the initial descent point of their learning curves.

In addition, TPBS has a linear time complexity with respect to the sample size, i.e., $O(T_1T_2T_0KK_0FWN)$, where T_1 is the number of trials in TPBS. It is also the size of the A chain; T_2 is the number of trials in BorderIdentification.

It is also the size of the B chain; K is the number of repetitions in GCoupling, and then is equal to the size of the R chain. They can be analyzed similarly to the analysis in the CPBS [61][62].

6.4 Summary

CPBS scales up PBS for border sampling on large datasets by assuming Coupling Markov Chain Monte Carlo. It achieves border sampling on a large population by a subsample resampled from the large population. It is shown that CPBS is efficient and effective in most cases.

Our main concerns are how to reduce the effect of the small sampling window while converging quickly and consistently to an effective sample without any loss of information. Empirically, collapsing to a star convex group for convergence detection in R chains can be further improved. An alternative method is to assume geometric propagation among subsamples.

We first describe a geometric propagation when a resampling technique is applied. It actually describes the evolution of the nearest neighbours of instances from subsamples to the whole population. As a result, all global nearest neighbours can be precisely obtained, and this helps border sampling on a large population by building tripled Markov Chains.

Further research on border sampling on large populations is crucial for many supervised learning tasks. For example, we compare borders obtained by Border Sampling technique with Support vectors. Support vectors [77] can be those borders obtained by the border sampling technique in a noise-free training set. Support Vector Machine can be only built with support vectors in training sets according to the dual method, which transforms a solution from an input space to a sample space [89]. As a result, this suggests a possible application of Border Sampling for training SVM as a wrapped

sample selection method while we have developed BS as a filter sample selection method in previous chapters.

BS as a wrapped sample selection method has many salient advantages for training classifiers such as linear time complexity and model-independence. However, in Chapter 7, we show a novel wrapped sample selection method, which is regarded as model-dependence and a different kind of border sampling techniques for enhancing individual classifiers, and performs at training time even though BS can be employed.

Chapter 7

Cascading Customized Couple

The proposed Border Sampling techniques such as the CPBS and TPBS in the previous chapters can be considered to be filter sample selection methods. However, a wrapped sample selection method may be more desirable than a filter method in many practical applications since such techniques are suitable for use in improving individual classifiers. For the Class Imbalance Problem (CIP), we propose a new Meta learning technique, which adopts a novel wrapped sample selection method. In future work, the proposed Border Sampling technques as discussed in the previous chapters could be developed as wrapped methods for classification. We empasize that the wrapped sample selection method, which will be discussed in this chapter, is quite different from other wrapped methods.

7.1 Introduction

In Machine Learning, classifiers are subject to the Class Imbalance Problem (CIP)[12][13] in that a classification favors the majority class when class distributions are heavily skewed. As a result, the minority class is poorly classified and thus this is undesirable. Previous research has shown that an ensemble learner with basic sampling techniques, e.g., under-sampling and over-sampling, or a synthesized method [12], can be more effective than a basic sampling technique [14][33][64][90].

Basic sampling methods (see Section 2.4.2.1) intend to define an optimal ratio for a balanced class distribution in order to help learn a better classifier. Both basic sampling methods and ensemble learning approaches (see Section

2.4.2.2) that have been proposed so far focus on binary domains for the CIP. It is not straightforward to apply these approaches to multi-class domains. Therefore, we suggest the design of a novel ensemble learning technique helping tackle the CIP by assuming an effective wrapped sample selection to improve the performance of individual classifiers on either binary or multiclass domains.

Our goal consists of the following two tasks:

- Design a novel Meta learning technique. It is expected that the novel Meta learning technique can enhance individual classifiers directly on either binary or multiclass domains instead of only binary class imbalanced domains for helping tackle the CIP.
- Adapt a proper individual classifier to this novel Meta learning technique. It is unrealistic to improve any traditional classifier significantly by assuming the novel ensemble learning technique.

With respect to the first task, we investigate previously proposed Meta Learning techniques. Adaptive Boosting (AdaBoost) is known as a successful Meta learner in practical applications [29]. It improves any learning algorithm by repeatedly calling a weak classifier for tuning the weights of examples to help build successive components. Bootstrap aggregating (Bagging) [8] enhances individual classifiers by building diverse models for reducing bias and variance. MultiBoostAB [91] assumes the boosting technique in AdaBoost and wagging technique to enhance the original AdaBoost. In spite of their successes shown in previous research, our experimental results show that all of these Meta Learning techniques suffer from the same difficulty to improve a linear model such as Naïve Bayes [6][24][84][91].

With respect to the second task, Naïve Bayes is an ideal classifier, which can be learned fast, and exhibits a stable performance over many other

classifiers in many practical applications [19], especially in text classification [76]. Recent research has proposed Naïve Bayes-like classifiers such as Averaged One-Dependence Estimator (AODE) [92], which improve Naïve Bayes by relieving the conditional independence assumption that often is violated in practical applications. Naïve Bayes and those Naïve Bayes-like classifiers are properly chosen as base learners in the new Meta learner.

Cascade Learning techniques (CL) such as Stacking and Cascade Generalization [32][85][97] are a sort of Meta Learning techniques, which are distinct from other Meta Learning techniques. It emphasizes the close relationship among its components while other Meta Learning approaches intend to build diverse sub-models by basic sampling techniques for reducing a bias and variance. Because subsequent components in CL are built according to the outputs of previously built components, this brings in an opportunity to introduce a possible wrapped sample selection method for successfully training subsequent individual classifiers.

As a result, we first propose a novel Meta Learning method, called Cascading Customized Couple (CCC), to improve the performance of individual classifiers. CCC achieves its goal by building a couple of individual classifiers, called Customized Classifiers (CC). In particular, one of CCs in the couple is customized on its own sub-domain such that it is able to classify the input if it belongs to the sub-domain.

Further, because the sub-domain in the second CC in CCC is essentially a separation, which is created according to training errors output by the first CC, and these training errors tends to appear as borders lying close to the decision boundary of the first CC, the separation can be a sort of Border Sampling techniques at training time, and can be regarded as a wrapped sample selection method.

Accordingly, we connect this new Meta Learning technique developed in this chapter with Border Sampling techniques described in previous chapters, and we suggest the design of this new Meta Learning technique with a novel wrapped border sample selection method instead of using BS based on BI_2 to enhance individual classifiers so as to tackle the CIP.

The main advantage of this new Meta Learning technique, at least, consists of the following three aspects.

Firstly, it can enhance any traditional classifier. In particular, it can significantly scale up Naïve Bayes and Naïve Bayes-like classifiers for either classification or class ranking tasks while other Meta learning techniques suffer from a failure to enhance this sort of classifiers.

Secondly, it describes a novel wrapped sample selection method, which can be used to enhance individual classifiers without presetting an optimal ratio for a balanced class distribution no matter whether a binary or multiclass domain is given.

Thirdly, it is a simple classification model because it only contains two components, and it can be built in a linear time complexity.

7.2 Preliminary

7.2.1 Meta Learning and Cascade Learning

Bootstrap aggregating (Bagging) [8] builds a Meta learner by building diverse models on subsamples (or bags) obtained by sampling uniformly with replacement on the original training set. The size of each bag is the same as the original and the expected frequencies of samples in each bag follows a discrete Poisson distribution, which is used for describing the frequencies of samples in a fixed period of time.

Adaptive Boosting (AdaBoost) is a Meta Learning algorithm [29], which is a general method to improve any learning algorithm by repeatedly calling

a weak classifier. In each round, the weights of each incorrectly classified example are increased and the weights of each correctly classified example are decreased, so that the new classifier focuses on those examples. The resulting AdaBoost classifier H is defined as

$H(x) = \text{sign}(\sum \alpha_t h_t(x))$, where each individual classifier h_t can be regarded as functional features.

MultBoostAB [91] assumes the boosting technique in AdaBoost and wagging (sampling with different weights) technique to enhance the original AdaBoost by wagging a set of sub-committee of classifiers in that each sub-committee is formed by AdaBoost.

Traditionally, Cascade Learning [32][85][97] emphasizes a straightforward relationship among individual components. For example, previously proposed cascade learning techniques such as cascade generalization [32] and stacked generalization [85][97] build a set of classifiers on the original domain. They also output class probability distributions for each instance in the original domain. A new domain can be constructed from the original domain by using these class probability distributions as its feature values in Meta level.

To begin, we first give a general definition about CL as follows.

Definition 7.1. *Cascade learning* (CL) is an ensemble learning technique that learns a set of classifiers from the original domain by building each component on its own sub-domain which can be reconstructed in terms of the outputs of previously built component learners. Its decision rule is defined by a specified average voting schema.

The above definition of CL finds its roots in previous research related to cascade generalization [32] and stacked generalization [85][97]. Some Ideas similar to CL in Definition 7.1 have been raised in previous research. The Cascade-Correlation method learns the architecture of artificial neural

networks by assuming a strategy similar to CL [25]. A mixture of local experts [43] can be built on a partitioned domain by assuming the Expectation Maximization (EM) algorithm for the mixture [50]. Recursive Bayesian Classifiers (RBC) [56] is a suggestive schema that uses a hierarchy of probabilistic summaries instead of a single schema, i.e., RBC partitions instances and recursively builds simple NB on each partition. We develop a novel technique to implement this hierarchy in this thesis.

7.2.2 Naïve Bayes and Enhancement

Given a training set with a probability distribution P, in supervised learning, Bayesian learning defines a classifier with a minimized error, i.e.,

$$y_i = c_i = \arg\max_{c_i \in C} P(c_i \mid x) \equiv \arg\max_{c_i \in C} P(x|c_i)P(c_i)$$

$$= \arg\max_{c_i \in C} P(a_1, a_2, ..., a_n \mid c_i)P(c_i) \qquad (7.1)$$

NB [57][66] assumes the probabilities of attributes $a_1, a_2, \ldots, a_n$ to be conditionally independent given the class c_i. Therefore, the right side of (7.1) becomes

$$P(x \mid c_i) = P(a_1, a_2, ..., a_n \mid c_i) = \prod_{j=1}^{n} P(a_j \mid c_i)$$

NB can be trained in linear time with a number of parameters $P(a_j|c_i)$ by assuming a probabilistic distribution P in the training set. Under no prior knowledge, the prior class probabilities, $P(c_i)$ in (7.1), can be set to $1 / |C|$, and discarded without any impact on the final result. A NB is *uniform* if prior class probabilities are uniform. Therefore, an uniform NB intends to classify a new input by only using class conditional probabilities $P(a_j|c_i)$.

A NB classifier built on a binary domain is a linear machine while a NB classifier built on a multiclass domain defines a multi-hyperplane boundary among classes for classification with a specified decision rule.

NB is a stable classifier, and has a linear training time. Because it has exhibited a high performance over other classifiers in many applications despite a limitation of the learnability [99], a number of studies promise the improvement of NB by overcoming the restrictions of conditional independence assumption or assuming a Meta Learning technique. We summarize these methods into the following four categories:

- Select a subset of attributes such that they satisfy conditional independence assumption. For example, Selective Bayesian Classifiers (SBC) [58] uses forward selection in a greedy search to find an effective subset of attributes, with which a better Naïve Bayes is built. However, it has been also verified that searching for dependent attributes is not the best way to improve Bayesian classifiers [19].
- Combine a decision tree with Naïve Bayes. For example, NBTree [54] builds a local NB on each leaf of a decision tree.
- Build a simplified Bayesian network structure by allowing a simple relationship between attributes given a class [47][92][100][101]. For example, Tree Augmented Naïve Bayes (TAN) [31] extends tree-like Naïve Bayes, in which the class node directly points to all attribute nodes, and an attribute node has only one parent attribute. This remains an acceptable computational complexity.
- Use Meta Learning techniques to enhance individual Naïve Bayes. For example, Bagging, AdaBoost, and MultiBoostAB can be used to scale up NB although it often suffers from failures to boost NB [6][24][84][91] and NB-like classifiers according to our observations.

Because the conditional independence assumption is not expected to be satisfied in practical applications [19], previous research [47][92][100][101] has proposed Naïve Bayes-like (NB-like) classifiers for the enhancement of NB by relaxing the conditional independence assumption and maintaining a

linear training time with respect to the number of instances. Those techniques for building NB-like classifiers are regarded as the third category, and they are described typically as follows.

Aggregating One-Dependence Estimators (AODE) [92] achieves higher accuracy than NB by averaging over a constrained group of 1-dependence NB models built on a small space. AODE with Subsumption Resolution (AODEsr) [101] augments AODE by detecting the specialization-generalization relationship between two attribute values at classification time and deleting the generalization attribute value. Hidden Naïve Bayes (HNB) [100] constructs a hidden parent for each attribute. Weightily Averaged One-Dependence Estimators (WAODE) [47] weights the averaged 1-dependence classifiers by the conditional mutual information.

There are two main points in these NB-like classifiers. Firstly, they all describe a simple Bayesian structure for Bayesian Learning. For example, AODE and AODEsr define constrained parents while HNB defines a hidden parent node. Secondly, they are all designed for nominal attributes.

SBC, TAN, and NBTree are not regarded as NB-like classifiers. SBC involves in an intractable task searching for attribute dependences. TAN needs structure learning in the time complexity of $O(n^2 logn)$, and then has a higher time complexity $O(tn^2+kn^2v^2+n^2 logn)$ for training than NB-like classifiers such as AODE in the time complexity of $O(tn^2)$, where t is the number of instances, n is the number of attributes, k is the number of classes, and v is the average number of values for each attribute. NBTree performs an intensive tree structure learning for partitioning the sample space such that the running time is much longer than that for running either a decision tree or a Naïve Bayes [54].

Experimental results have shown that in most cases NB-like classifiers outperform SBC and NBTree for scaling up NB [47][92][100][101] and are

more efficient and effective than TAN. NB-like classifiers also outperform those Meta Learning techniques such as AdaBoost except for a few cases.

7.3 Cascading Customized Couple for Classification

We develop a new Meta learning technique, called Cascading Customized Couple (CCC) according to Definition 7.1.

7.3.1 Customized Classifiers

The main idea behind CCC is related to a new classifier, called *Customized Classifier* (CC).

For example, NB is a linear classifier defined in an input space from a training set [66]. It is expected that a training set can be separated into many small subsets such that the original training set can be more precisely classified by subsequent individual NB classifiers built on the partitioned training set. We describe several related concepts as follows.

Definition 7.2. A labelled training set can be regarded as a *domain*, which describes some domain knowledge in the training set. A subset of the training set can be regarded as a *sub-domain*. The *off-domain* of a sub-domain is defined as all of the other sub-domains in the training set.

Initially, examples in each original class naturally constitute a sub-domain. However, the original domain can be further divided into many *additional* sub-domains if necessary, and additional sub-domains can be labelled by *additional class labels*.

A domain is not always linearly separable, but it is expected that some sub-domain can be linearly separable from its off-domain. Therefore, a classifier can be customized in terms of a sub-domain and its off-domain for correct classification on its sub-domain.

Definition 7.3. A *Customized Classifier* (CC) is a classifier, which can classify an input in the related sub-domain, and can reject the classification on an input in its off-domain.

Definition 7.3 can be described as follows. A CC can output a class distribution of an input with respect to the original classes and the additional classes. For an input belonging to the off-domain, the CC intends to classify it by outputting a class membership probability 0 or an equal class membership probability for the original classes. This leads to a *rejection* of classification on the input by eliminating its effect on the final classification if an averaging vote rule is used. The principles of CC and CCC can be simply described in Example 7.1.

Example 7.1. Given a domain D with two original classes, c_1, c_2, without loss of generality, two CC classifiers, H_1 and H_2, are built from D, where H_1 has its sub-domain S_1 and its off-domain labelled by c_3; H_2 has its sub-domain S_2 and its off-domain labelled by c_4.

Given an input $x \in S_1$ with the label c_1, because x is in the sub-domain S_1 of H_1, H_1 can classify x. Suppose we have $P_1(c_1|x) = 0.6$, $P_1(c_2|x) = 0.3$, and $P_1(c_3|x) = 0.1$

Because x is not in the sub-domain S_2 of H_2, H_2 cannot correctly classify x. Instead, H_2 classifies x as its additional class c_4, Suppose we have

$P_2(c_1|x) = P_2(c_2|x) = 0.2$, and $P_2(c_4|x) = 0.6$, i.e., H_2 rejects classifying x as c_1 and c_2 because $P_2(c_1|x) = P_2(c_2|x)$. Therefore,

$p_1 = (P_1(c_1|x) + P_2(c_1|x) / 2 = 0.4$

$p_2 = (P_1(c_2|x) + P_2(c_2|x) / 2 = 0.25$

As a result, $c_1 = \arg\max_i (P_1(c_i \mid x) + P_2(c_i \mid x)/2)$, where $i = 1, 2$. □

Further, we have the following fact for a probability learning schema consisting of all CCs with a specific average voting rule.

Theorem 7.1. Given a training set D, which defines a original whole domain, if all customized classifiers $h_i = CC_i$, $i = 0,\ldots, k$, are properly defined on their sub-domains from the domain according to Definition 7.3, a probability learning schema H consisting of all built customized classifiers can correctly classify the whole domain.

Proof: Given any input x, provided that it belongs to a sub-domain D_i, there exists a h_i:CC_i built on D_i, which can correctly classify x as the label c_i, while other CC_j, $i \neq j, j = 0,\ldots, k$, cannot classify x because x does not belong to their sub-domains. Because $h_i\{p(c_i \mid x) > p(c_j \mid x)\}$ and $H\{P(c_i \mid x) > P(c_j \mid x)\}$, we have $H(x) = h_i(x)$ in terms of the average voting schema. Conversely, provided that $H(x) = c_i$, there is a h_i:CC_i which can classify it as c_i while other CC_js reject the classification on x according to Definition 7.3. Therefore, $H(x) = h_i(x)$. □

Theorem 7.2 shows that the probability learning schema consisting of all CC, which is defined in Definition 7.3, is a perfect classifier. However, we do not have an existing induction algorithm as an oracle for building those CCs. Instead, we can learn a couple of CCs by building any traditional classifier on the original domain and the changed training set consisting of its sub-domain containing examples labelled by original classes and the off-domain containing examples labelled by additional classes, respectively.

7.3.2 Learning Cascading Customized Couple

Given a training set D, the initial domain is the whole training set with original class labels. The first CC_0 is built on D by using a specified learning algorithm. CC_0 actually is a traditional classifier built on the original training set. The learning algorithm is done if CC_0 totally fits the training set D without misclassifications.

Otherwise, the misclassifications need to be further classified in the training set. To this end, we can add additional classes to label the corresponding correct classifications. As a result, the misclassification becomes a sub-domain, and all sub-domains in the additional classes are regarded as its off-domain. The original training set D becomes a new training set D_1 with the sub-domain containing the misclassifications and the additional classes corresponding to the classified examples. The second CC_1 is built on D_1 and the learning algorithm is end up with a couple of CC classifiers. Because CC_1 is built in terms of those outputs of CC_0, we say that they are cascaded with each other, and they are combined with each other to become a CCC classifier.

According to the above discussion, we propose the Cascading Customized Couple (CCC) induction algorithm to learn a CCC classifier, as shown in Figure 7.1.

The CCC algorithm builds a CCC classifier with its two input: the original training set D and a specified base induction algorithm L(). Because the algorithm performs labelling on D, the algorithm initially saves all original labels in the training set D by saveLabels() at Step 1 while it restores all original labels at Step 9 by restoreLabels() after it builds the couple of CC classifiers.

At Step 2, B is initialized as an empty set, which is used for collecting the resulting CC classifiers. The first CC learner h_i is built on D at Step 3 by using a traditional learning algorithm L(), e.g., NB() for learning a NB classifier. At Step 4, the misclassifications (training errors, also see 7.3.2.2) E of h_1 on D are computed, and the correct classifications CT on D are obtained by removing misclassifications E from D.

CCC algorithm
input *D*: original domain;
L: a specified base learner
output *H*:*CCC*, the resulting CCC classifier
begin
1 saveLabels(*D*)
2 $B = \varnothing$
// first CC
3 h_1 = L(*D*), $B = B \cup \{h_1\}$
4 $E = h_1(D)$, CT = *D* - *E*
// second CC
5 if(|CT| < |D|)
6 addClasses(CT, *D*, 0)
7 h_2 = L(*D*), $B = B \cup \{h_2\}$
8 $H(x) = \tilde{c} = \arg\max_{c \in C}\left(P''(H(x) = c)\right)$,

8.1 $$P''(H(x) = c) = \frac{P'(H(x) = c)}{\sum_{c' \in C} P'(H(x) = c')}$$

8.2 $$P'(H(x) = c) = \frac{1}{|B|}\sum_{h \in B} P(h(x) = c)$$

8.3 $$P(h(x) = c) = \frac{P(h(x) = c)}{\sum_{c' \in C'} P(h(x) = c')}$$

9 restoreLabels(*D*)
10 return *H*:CCC(B)
end.

Figure 7.1. Cascading Customized Couple induction algorithm.

At Step 5, if |CT| = |D|, then the first CC fits in D, CCC does not build the second CC for classifying training errors. Otherwise, the algorithm classifies those misclassifications contained in *E* from Step 6 to 7.

At Step 6, addClasses(), as shown in Figure 7.2, is used for adding additional class labels to the original domain D, and re-label those correct classifications obtained at Step 4 as the different additional classes in terms of their original class labels. In last phase, at Step 8, the learning algorithm defines a CCC classifier, which is an ensemble learner containing the resulting CC classifiers in *B*, with a modified averaged vote schema (see

7.3.2.3), where C is a set of original class labels while C' is a set of original class labels and additional class labels.

7.3.2.1 Additional Classes

The CCC induction algorithm defines additional classes on a training set for building cascaded CC classifiers. These additional classes help define the hyperplanes of a CC classifier to classify its sub-domain. Additional classes are defined by invoking the procedure addClasses(), as shown in Figure 7.2, where input i is used as an indicator variable for defining new additional classes.

```
addClasses algorithm
input   S: subdomain;
        i: beginning index
        D: original domain
output  D': a new domain with additional classes
begin
1  L = ∅, L'= ∅, j = 0, k = i
2  foreach p ∈ S
3     c = p.classLabel
4     if(c ∉ L)
5          L[j] = c      // current class label
6          L'[j] = c.k   // c.k is a new class label
7          p.classLabel = L'[j]
8          k++; j++
9     else
10         p.classLabel = L'[j'], where L[j'] = c
11 addClassLabels(L', D)
end.
Proc addClassLabels(L', D)
begin
12   i = 0; k = |D.classes|
13   foreach c = L'[i]
14      D.classes[k + i] = c
15      i++
end
```

Figure 7.2. addClasses Algorithm in CCC Algorithm.

From Step 2 to Step 10, the procedure re-labels each instance p in S with a new additional class label c.k. But p will be re-labelled with the same label

if the current label of p is the same as the current label of another instance. L recodes all current labels and L′ recodes all new additional class labels. Finally, at Step 11, the procedure extends the list of original class labels of D with new additional class labels.

7.3.2.2 Training Errors

The proposed learning algorithm needs to compute misclassifications at Step 4, as shown in Figure 7.1, and those misclassifications are regarded as *training errors* and are needed for further classification in CCC, also see the example in Section 7.3.3. For this reason, a base learner in CCC should avoid overfitting.

On the other hand, for a NB base classifier, because a prior class probability estimated from a training set might lead to a classification bias to the majority class, training errors will be overestimated. This suggests that a uniform NB is more proper than a traditional NB as a base learner in CCC by computing class conditional probabilities $P(a_j|c_i)$, as discussed in Section 7.2.2.

7.3.2.3 Average Decision Rule

Each individual classifier CC in a CCC computes a class probability distribution of all classes given an input x, i.e., $p(c_i|x)$, $i = 0, \ldots, c'$. It also includes the class probability distribution of additional classes.

A CCC classifier is an ensemble learner containing a couple of CC classifiers, in which the second CC intends to correctly classify their own sub-domains, and rejects a classification on an input in the corresponding off-domains by outputting 0 or the approximately equal membership probabilities for the original classes although it classifies the input in the off-domain as an additional class, much as described in Example 7.1.

Therefore, the crucial thing for a CC is that it is required to precisely output the class ranks on the original classes no matter whether it classifies an input as an additional class. That means that all resulting probabilities on additional classes given an input x will not be useful for a final classification.

This suggests that an average voting rule for a final prediction of the ensemble learner is preferable to a majority vote rule. The traditional average voting rule can be adapted in a CCC, and is defined at Step 8 in Figure 7.1, where C' is a set of class labels containing additional classes while C contains all original classes.

For each h, $P(h(x) = c)$ is a normalized among all $P(h(x) = c')$ in 8.3 of Figure 7.1, where $c' \in C'$, and this normalization is achieved in each h. In 8.2, $P'(h(x) = c)$ is a average of all $P(h(x) = c)$ of all h on an original class $c \in C$, and it will be again normalized in 8.1 for the maximum vote. That means that CCC performs the *double* classifications to yield a final decision.

An implementation of this average voting rule is shown in Figure 7.3, where the rule needs the number of original classes, denoted as numOriginalClasses, for the double classification; because two CC have different number of classes, the procedure extend() to make two arrays probs[] and dist[] the same size; normalize() is used for normalizing probabilities in probs[]; reset() is used for removing probabilities of additional classes such that normalize() at Step 11 is valid for maximum average voting using maxIndex() at Step 12.

```
// New Average Voting Rule for classification
classify algorithm
input   x: a new input
output  y: class label
begin
1  probs = classifiers(0).probability(x)
        // probs: class probability distribution for x
        // classifiers: all individual classifiers
2  reset(probs, numOriginalClass)
        // set a probability 0 for additional classes
3  s1 = size(probs), s = size(classifiers)
4  for k = 1 to s - 1
5       dist = classifiers(k).probablity(x), s2 = size(dist)
6       reset(dist, numOriginalClass)
7       if(s1 < s2)
8            probs = extend(probs, s2)
                  // extend the size of probs to s2
                  // and copy original values
9       probs[j] = probs[j] + dist[j], j = 0,…, s2
10 probs[j] = probs[j] / s, j = 0,…, size(probs)
11 normalize(probs)
12 return maxIndex(probs) // maximum index
end.
Proc reset(probs, size)
begin
     for(i = size; i< size(probs); i++)
        probs[i] = 0
end
```

Figure 7.3. New Combination Rule: Average Voting Rule in CCC.

7.3.2.4 Computational Complexity

There are several related subroutines in the CCC learning algorithm. saveLabels() at Step 1 and restoreLabels() at Step 9, as shown in Figure 7.1, can be simply implemented in a linear time complexity. Also, addClasses() adds additional class labels into *D* for those correctly classified examples in CT in a linear time complexity. A base learner is a traditional induction algorithm. For example, NB() has a linear training time O(*tn*), where *t* is the number of examples in the training set, *n* is the number of features in the input space [66].

CCC computes misclassifications at Step 4 by the learned CC classifier h_1. This can be achieved in a linear time complexity O(ktn), where k is the number of the original classes.

CCC only contains two components. As a result, the CCC learning algorithm with a base learner NB has a linear training time O(ktn), which has a lower order of magnitude than O($tn^2+kn^2v^2$) of HNB [100] or the time complexities of other NB-like classifiers.

Because CCC only changes class labels of examples for training cascaded CC classifiers, it requires an extra space O(t) for saving the original labels of examples. A customized NB classifier requires the space complexity of O(($k+k'$)$nv+t$) for training, where k' is the number of additional classes. Therefore, the space complexity of the resulting CCC with a base learner NB for a niminal case is O(($k+k'$)nv) while the space complexity for a continuous case is O(($k+k'$)n).

7.3.3 An Example

Given a dataset V with two classes, i.e., the square class (minority), denoted as '□', and the diamond class (majority), denoted as '◊', as shown in Figure 7.4(a), we show how CCC works on this synthesized dataset.

CCC with a base NB builds its first classifier. CCC runs the learned NB to correctly classify most examples, which are signed a minus class, denoted as a '–', and a solid dot class, denoted as '•', for the original diamond class and the original square class, respectively, as shown in Figure 7.4(b). The learner also misclassifies a few diamond examples and a few square examples. As we can see, the NB classifier as a linear classifier separates the original domain by a straight line.

CCC continues to build the second classifier by using the base learner and classifies the remaining misclassifications on the new sub-domain after all

correct classifications are re-labelled with additional class labels, i.e., '–' and '•' for those classified data points belonging to the diamond class and the square class, respectively.

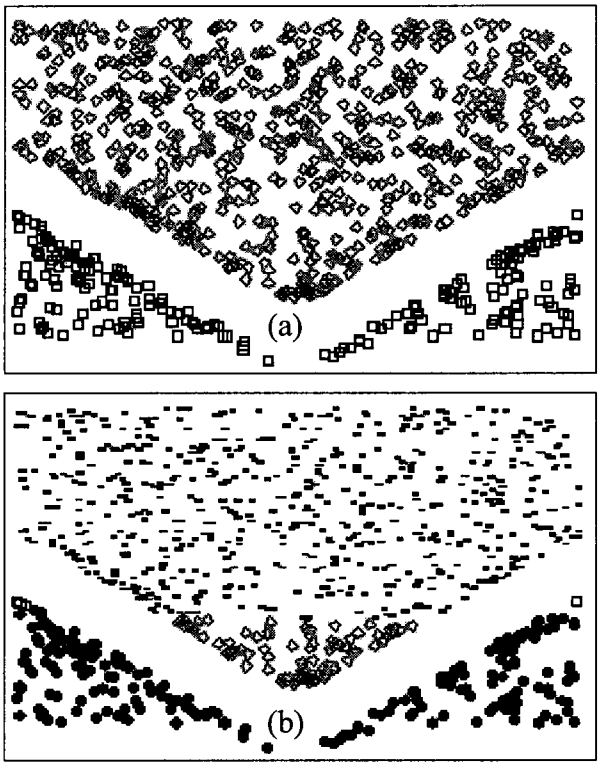

Figure 7.4. An Example of CCC with a base NB.

Finally, the resulting CCC is composed of two CC classifiers: the first one is essentially a traditional NB built on the original domain while the second one is customized on the sub-domain containing misclassifications output by the first one.

It is shown that CCC enhances NB by building multiple hyperplanes to separate the original domain with the original classes.

7.3.4 Border Sampling at training time

Additional classes are used for defining the separation of the original domain, and thus helping define sub-domains consisting of training errors output by the previously built CC.

According to CCC algorithm, as shown in Figure 7.1, the first CC is always built on the original training set. If CC does not fit the training set, the training errors of CC on the original training set are usually those border points lying close to the decision boundary, which is also adjacent to the class boundary, because the border points have high uncertainty for discrimination. A sub-domain will be defined on these training errors by labelling correct classifications as additional classes, and a successive CC is built on the separated domains to classify the sub-domain.

As a result, this separation of the original domain can be regarded as a different kind of Border Sampling techniques, which is wrapped in CCC and is different from BS based on BI_2. We can compare them as follows.

Firstly, both are regarded as novel Border Sampling techniques for sample selection in supervised learning. According to their performances, one is called *BS on class boundary* by using BI_2 while another one is called *BS on decision boundary* by defining the separation in CCC.

Secondly, BS on class boundary can be used as either a filter method, which performs prior to training time, or in theory a wrapped method for sample selection while BS on decision boundary is a wrapped sample selection method, which performs at training time.

Thirdly, BS on class boundary is model-independent while BS on decision boundary is model-dependent. It is believed that both are necessary as sample selection methods in supervised learning.

Empirically, CCC can enhance any base classifier in most cases by employing this kind of Border Sampling techniques. A plausible reason can be analyzed as follows.

Suppose that we have a domain D with two original classes c_1 and c_2, and a CCC is built on D.

On the one hand, if the first CC_1 misclassifies a new input x with a true label c_1, and x is located at the region close to the decision boundary of CC_1 without any loss of generality. Because CC_1 misclassifies x and x is a border close to the decision boundary of CC_1, we have $p_1(c_1|x) < p_1(c_2|x)$ and $p_1(c_1|x) - p_1(c_2|x) \geq -\delta$, where δ is a sufficiently small positive decimal fraction.

On average, the second CC_2 can correctly classify x, then we have $p_2(c_1|x) > p_2(c_2|x)$ and $p_2(c_1|x) - p_2(c_1|x) > \delta$.

With the decision rule in CCC, we have

$$P(\mathrm{H}(x)) = \arg\max_i (p_1(c_i \mid x) + p_2(c_i \mid x)).$$

Because $p_1(c_1|x) + p_2(c_1|x) - (p_1(c_2|x) + p_2(c_2|x))$

$= p_1(c_1|x) - p_1(c_2|x) + (p_2(c_1|x) - p_2(c_2|x)) > 0$

Therefore, CCC: $\mathrm{H}(x) = c_1$. That is, CCC, on average, correctly classifies those misclassifications output by the first CC_1, i.e., the base classifier.

On the other hand, if the first CC_1 correctly classifies a new input x with a true label c_1 with a probability δ. Therefore, we have $p_1(c_1|x) > p_1(c_2|x)$ and $p_1(c_1|x) - p_1(c_2|x) \geq \delta$, where δ is a positive decimal fraction.

On average, the second CC_2 can correctly classify x as an additional class label because the region which x is located at is separated into a sub-domain labelled as the additional class. Therefore, CC_2 assigns x to c_1 and c_2 with low probabilities, and then we have $p_2(c_1|x) \approx p_2(c_2|x)$ and $|p_2(c_1|x) - p_2(c_1|x)| < \delta$.

With the decision rule in CCC, we have

$P(\mathrm{H}(x)) = \arg\max_i (p_1(c_i \mid x) + p_2(c_i \mid x))$.

Because $p_1(c_1|x) + p_2(c_1|x) - (p_1(c_2|x) + p_2(c_2|x))$

$= p_1(c_1|x) - p_1(c_2|x) + (p_2(c_1|x) - p_2(c_2|x)) > 0$

Therefore, CCC: $\mathrm{H}(x) = c_1$. That is, CCC, on average, correctly classifies those correct classifications output by the first CC_1, i.e., the base classifier.

By combining the above two situations, we claim that in general CCC is able to enhance any classifier.

7.3.5 Discussion

There are several common points between AdaBoost and CCC algorithms. Both are ensemble learning techniques. Subsequent individual classifiers are built according to the outputs of previously built individual classifiers. Both are linear algorithms, and are not subject to overfitting if a proper base learner is selected.

However, we emphasize several crucial differences between an AdaBoost classifier and a CCC classifier as follows. (a) No w eight of samples is defined in CCC; (b) Because of additional classes, the modified averaging vote rule in CCC has the double classifications for a final decision; (c) Instead of using weights for more effort on misclassifications, CCC further classifies training errors output by the first CC by building additional classes to separate classified examples and misclassifications; (d) The functional features $h(x)$ in a CCC are more functional than the features $h(x)$ in AdaBoost due to the double classifications; (e) the number of components of H in AdaBoost is infinite while the number of components of H in CCC is two. We emphasize that both are regarded as two important learning techniques. Even CCC is not a classifier but a probability learning schema as compared with AdaBoost.

There are some connections between SVM and AdaBoost [29]. AdaBoost attempts to maximize the minimum margin of any training example [29]. Importantly, SVM utilizes the kernel trick to transform an input space into a feature space in a high dimension space [89]. A non-linear problem in the input space can be solved by building a linear classifier in the high dimension space. As a result, this suggests that CCC can be an ensemble of kernel machines. Individual kernel machines in CCC define some hyperplanes in the high dimension space.

```
getCC algorithm
input   D: original domain;
        L: a set of base learners
output  cc: the resulting CC
begin
1   auc0 = 0, cl0 = null
2   for each base learner cli ∈ L
3       cli.numOriginalClasses = getNumClasses(D)
4       cli.buildClassifier(D)
5       auci = AUC(cli, D)
6       if(auci > auc0)
7           auc0 = auci
8           cl0 = cli
9   return cl0
end.
```

Figure 7.5. getCC: build a CC using a dynamic selection strategy.

To select a proper kernel machine, CCC algorithm can be altered at Step 3 and Step 7, as shown in Figure 7.1, with a *dynamic selection* in that CCC selects the best base learner from a committee of several candidates in L. Because a high class ranking score is crucial for a proper base learner in CCC according to the discussion in Example 7.1 and 7.3.2.3, the AUC is chosen as a measure for the dynamic selection instead of an accuracy measure. The dynamic selection can be simply implemented, as shown in Figure 7.5, where *numOriginalClasses* representing the number of original classes in D is input into cl_i at Step 3.

7.4 Summary

We develop a novel Meta learning technique to enhance individual classifiers such that this new technique is expected to help address the CIP. Our preliminary research shows that this technique can be used to scale up Naïve Bayes classifiers on either binary or multiclass domains no matter whether they are class balanced or not.

Naïve Bayes (NB) [57] is a simple and stable classifier with a linear training time. It is deduced under the conditional independence assumption, and has demonstrated a surprising accuracy on many classification tasks [19] although the assumption is often violated in practical applications [54][56][76]. Current research has focused on the improvement of NB by developing Naïve Bayes-like (NB-like) classifiers which relax the assumption by learning a simple Bayesian structure and maintain a linear training time with respect to the number of examples in a training set [47][92][100][101].

Theoretically, a probability learning schema consisting of all CCs is a perfect classifier. The main problem is that we have no an existing induction algorithm for building a CC. The proposed novel Meta learning method, called Cascading Customized Couple (CCC), improves the performance of individual classifiers by building a couple of classifiers, both of which are called Customized Classifiers (CC). We show that CCC can enhance any base classifier.

There is an increasing interest to know a proper individual classifier, e.g., a uniform Naïve Bayes, used in CCC. In general, it is not suggested that CCC ensembles traditional classifiers which are subject to overfitting because CCC trains a successful classifier by estimating training errors.

CCC assumes a novel wrapped sample selection technique, which is different from a traditional sampling method, e.g., an under-sampling or an

over-sampling method. It is shown that this novel wrapped sample selection can be regarded as a kind of border sampling techniques, called Border Sampling on decision boundary by defining the separation in CCC. In addition, Border Sampling technique on class boundary by using BI_2 can be used as either a filter sample selection or a wrapped sample selection method although it is suggested to be used as a filter method for sample selection on either a small training set or a large population in this thesis.

We restrict our discussion to the comparison between CCC and other Meta Learning techniques such as Bagging, AdaBoost, and MultiBoostAB, and show that CCC is more successful than these Meta Learning techniques to scale up NB and NB-like classifiers. Both CCC and AdaBoost are regarded as two important learning techniques. Even CCC is not a real classifier as AdaBoost but a probability learning schema.

The couple of CC classifiers in CCC can be dynamically built by using different base learners according to their class ranking scores on sub-domains no matter whether base learners are a linear machine or not.

Chapter 8

Experiments

We describe our experiments to evaluate the proposed Border Sampling techniques as a filter sample selection method, and the proposed Cascading Customized Couple (CCC), which is a new Meta learning technique and is also regarded as a novel wrapped sample selection method.

Our main purposes are to extensively evaluate the effectiveness of the proposed techniques about Border Sampling and the CCC. In particular, the main experimental results are shown in Section 8.4.2 and Section 8.4.4 to justify the Coupling Markov Chain Monte Carlo for scaling up Progressive Border Sampling (CPBS) and CCC, respectively. In more detail, we describe our experimental results in the following order:

Firstly, we evaluated Progressive Border Sampling (PBS) for Border Sampling (BS) as compared with the traditional Border Identification (BI) for sample selection.

Secondly, we evaluated the CPBS for the scalability of PBS as compared with the Progressive Sampling (PS) for sample selection on large datasets.

Thirdly, we compared Tripled Markov Chain Monte Carlo for scaling up PBS (TPBS) with CPBS for sample selection, and compare TPBS with previously proposed instance selection techniques, which originally are designed for Instance-Based Learning (IBL).

Finally, we evaluated our new Meta learning technique CCC for scaling up Naïve Bayes classifier as compared with previously proposed Meta learning techniques such as Bagging and AdaBoost, and Naïve Bayes-like or One-Dependent-Estimation (ODE) classifiers for improving Naïve Bayes.

8.1 Datasets for Experiments

To evaluate the new techniques proposed in this thesis for Machine Learning, we chose 33 datasets in that one of them is obtained from a scientific application while others are those benchmark datasets from the UCIKDD repository [39].

Table 8.1. The characteristics of all 33 datasets for experiments.
Notes: 32 benchmark datasets (#0-#31) from the UCI repository and one synthesized dataset (#32) Explosion for Nuclear monitoring application; #attr, #ins, and #c are denoted as the numbers of attributes, instances, and classes, respectively.

No.	Datasets	#attr	#ins	#c
0	Anneal	39	898	5
1	Audiology	70	226	24
2	Autos	26	205	6
3	Balance-s	5	625	3
4	Breast-w	10	699	2
5	Colic	23	368	2
6	Credit-a	16	690	2
7	Diabetes	9	768	2
8	Glass	10	214	6
9	Heart-s	14	270	2
10	Hepatitis	20	155	2
11	Hypothyroid	30	3772	4
12	Ionosphere	35	351	2
13	Iris	5	150	3
14	kr-vs-kp	37	3196	2
15	Labor	17	57	2
16	Letter	17	20000	26
17	Lymph	19	148	4
18	Mushroom	23	8124	2
19	P-tumor	18	339	21
20	Segment	20	2310	7
21	Sick	30	3772	2
22	Sonar	61	208	2
23	Soybean	36	683	18
24	Splice	62	3190	3
25	Vehicle	19	846	4
26	Vote	17	435	2
27	Vowel	14	990	11
28	Waveform	41	5000	3
29	Zoo	18	101	7
30	Adult	15	48842	2
31	Shuttle	10	58000	7
32	Explosion	5	92630	2

For the scientific application, a possible method of explosion detection for the Comprehensive nuclear-Test-Ban-Treaty [82] consists of monitoring the amount of radioxenon in the atmosphere by measuring and sampling the activity concentration of Xe-131m, Xe-133, Xe-133m, and Xe-135 by radionuclide monitoring. Several samples were synthesized under different circumstances of nuclear explosions, and combined them with various measured levels of normal concentration backgrounds to synthesize a training dataset, called Explosion, for use with machine learning methods.

The characteristics of the these datasets are described in Table 8.1, where the columns are the names of the datasets, the number of attributes (#attr), the number of instances (#ins), the number of classes (#c).

8.2 Machine Learning Induction algorithms

We chose some classical induction algorithms to evaluate our new techniques for supervised learning in Machine Learning. These induction algorithms for building the corresponding classifiers are chosen from the Waikato Environment for Knowledge Analysis (Weka) tools [94].

In the experiments for evaluating the Progressive Border Sampling (PBS), we chose three traditional classifiers, which have been widely used for many practical applications, i.e., Naïve Bayes (NB), Support Vector Machine (SVM i.e., SMO [68]), and Decision Tree (DT, i.e., J48 [94]), which is an implementation of C4.5 [74] in Weka. We are mainly concerned about the effectiveness of PBS for sample selection. We compared PBS with the traditional BI (Duch 2, in Section 2.1.2) with respect to their effectiveness by evaluating the performance of NB, SVM [68], and DT [74], built on either the resulting sample obtained by PBS, on full training sets (Full), or on samples produced by the traditional BI (BI) algorithm.

All induction algorithms were run with their default settings, e.g., NB with Gaussian Estimator for continuous values (Maximum Likelihood Estimator for nominal values), SVM with polynomial of 1 for kernel function and constant C of 1 for soft margins, and DT with no reduced error pruning and no C4.5 pruning [74] and no Laplace smoothing.

In the experiments to evaluate the CPBS and TPBS, we selected the four learners: NB, DT, SVM, and IB1 [94], where IB1 was run with its default settings with a normalized Euclidean distance for IBL in Weka.

In the experiments to evaluate Cascading Customized Couple (CCC), we compared the CCC with previously proposed Meta classifiers: Bagging, AdaBoost, and MultiBoostAB, and we also compared the CCC with Naïve Bayes-like or One-Dependent-Estimation (ODE) classifiers including AODE, AODEsr, HNB, and WAODE for improving Naïve Bayes.

Table 8.2. Experiments with the proposed methods, previous approaches, and classifiers.

No.	Proposed methods	Previous approaches	Classifiers
0	PBS	Traditional BI	NB, SVM, DT
1	CPBS	PS: Static, Arith, Geo	NB, SVM, DT, IB1
2	TPBS	CNN, ENN, RENN, DROP3.1	NB, SVM, DT, IB1
3	CCC	SBC, TAN, NBTree, AODE, AODEsr, HNB, WAODE, AdaBoost, Bagging, MultiBoostAB	NB, SBC, TAN, NBTree, AODE, AODEsr, HNB, WAODE, Bagging, AdaBoost, MultiBoostAB

Notes: PBS: Progressive Border Sampling; CPBS: Coupling Markov Chain Monte Clarlo for Scaling up PBS; TPBS: Tripled Markov Chain Monte Carlo for Scaling up PBS; CCC: Cascading Customized Couple

Because those NB-like classifiers only works on nominal attributes and no missing values, the datasets were pre-processed by using the *ReplaceMissingValue* tool in Weka for missing values and using the unsupervised *Discretize* tool in Weka for discretizing continuous values. The

classifiers were built with their default settings, e.g., AODE with a frequencyLimit of 1, i.e., any attribute with values below this limit cannot be used as a parent, and so on.

All experiments, the corresponding methods, and classifiers are described in Table 8.2.

8.3 Methodology of Experiments

8.3.1 Statistical Test Methods

We used the paired t-test and the Wilcoxon signed rank test for significance testing on the results from two classifiers. The Wilcoxon signed-rank test or Wilcoxon test is a non-parametric statistical hypothesis test for two repeated measurements under the assumption of independence of the differences. It is an alternative to the paired t-test when these measurements cannot be assumed to be normally distributed.

Although the Area Under ROC Curve (AUC) value is equivalent to the Wilcoxon signed rank test or Wilcoxon-Mann-Whitney statistic [36] for observations from a classifier, the significance test by using the Wilcoxon test on the results observed from two classifiers is secure because the differences of AUC values from different classifiers are expected to be independent.

8.3.2 Validation

We conducted our experiments via the 10-fold cross validation (CV) to compare our new techniques with previous approaches. The results in the 10-CV are averaged for evaluation. The paired t-test and Wilcoxon test with 95% confidence level is used for significance tests about the averages between our proposed methods and other methods. In general, just as in Table 8.4, in a small column following each result, '*w*' and '*l*' denote that the

Progressive Border Sampling (PBS) wins and loses, respectively, against the corresponding method while an empty space represents a draw.

The AUC [26] was chosen as a performance measure because it is stable in the case of class imbalance domains [26]. Therefore, the statistical significance with respect to the obtained AUC was tested by using the paired t-test and the Wilcoxon signed rank test [26][36] at significance levels of 0.05.

8.4 Experimental Results

We reported experimental results for our new techniques proposed in this thesis.

8.4.1 Experimental Results on PBS

We conducted experiments on the first 30 benchmark datasets in Table 8.1. Progressive Border Sampling (PBS) was run with the similarity distance metric Cosine for BI_2. We used the new method, as discussed in Section 3.5, for the combination of continuous, nominal, and missing values in the distance metric.

The initial results for training set reduction on the 30 training sets are shown in Table 8.3, where the maximum number of trials (t) for augmenting borders in the PBS between two classes and the maximum depth (r) of iteration for far borders, the number of data points produced by PBS (#PBS), and the percent (%) of data points produced by PBS over the total number of instances (the ratio of #PBS over #ins) in the training set.

In general, a small fraction of #PBS over #ins is preferable. For instance, Mushroom is reduced to 15.02% while Vowel is reduced to 97.98%. These results show that Mushroom contains much redundancy (Hypothyroid contains the most redundancy) while Vowel contains a little redundancy. In addition, PBS reaches its algorithmic convergence among the 30 benchmarks

of UCI with a maximal trial value $t = 6$ on the Diabetes dataset and a minimal trial value $t = 2$ on the Audiology dataset. The depth of iterations for far borders reaches a maximum 4 in the Anneal dataset and a minimum 1 in the Mushroom dataset. The averages of these results are shown at the bottom of the table. As a result, empirically, t and r are bounded with small numbers which are much smaller than #ins. Therefore, PBS has a quadratic time complexity with respect to the number of instances in training sets by ignoring the impacts of t and r.

We compared PBS with the traditional BI, which implements the Duch 2 algorithm, as discussed in Section 2.1.2, with respect to their effectiveness by evaluating the performance of NB, SVM, and DT classifiers, which were built on either the resulting sample obtained by PBS, on full training sets (Full), or on the resulting sample produced by the traditional BI (BI) algorithm. The BI computes the nearest neighbours by using the same similarity distance metric Cosine as PBS and sets with $k = 1$ for the nearest neighbours without loss of generality.

Our experiments were run 2 times the 10-fold cross validation with different random seeds. In each run in the 10-fold cross validation, PBS and BI perform sample selection on the 9 folds, and the resulting sample is used for training classifiers, which are tested on the remaining fold as the holdout test set. The process is repeated for each fold. Statistical tests for the paired t-test at 95% confidence level between PBS and Full, and BI with respect to the performance of trained classifiers are reported in Table 8.4, where in a small column following each result, '*w*' and '*l*' denote that PBS wins and loses, respectively, against Full and BI while an empty space represents a draw.

We also compared PBS with the traditional BI for training set reduction with respect to elapsed time and the sample size. In Figure 8.1, we show

elapsed times of PBS and the traditional BI on the 30 benchmark datasets. In Figure 8.2, we show the sizes of far borders (Far) obtained by using PBS, the resulting samples obtained by using the traditional BI and PBS, and the original training set Full. It is shown that PBS, on average, needs a little more time (never 5 times slower) than th e BI for sample selection. On average, the sample size of PBS is at most 3 times larger than that of BI. For instance, in Mushroom, only 1143 points were identified by using PBS in 168 seconds as compared with 7311 points in Full.

To justify the effectiveness of PBS, we first analyzed one case where PBS does not seem to be producing an effective sample for training classical classifiers, as shown in Table 8.4. In Hypothyroid, PBS leads to poor accuracies of NB, SVM, and DT. We calculate AUC values for the three classifiers built by using PBS and Full on Hypothyroid: 0.9225/(l)0.9387(NB); 0.8241/(w)0.7036(SVM); 0.9618/0.9513(DT). According to the AUC values, PBS improves SVM and resembles Full for DT while it only a little degrades the performance of NB in this case.

The accuracy measure might not be a proper measure for evaluation on Hypothyroid, which has a highly imbalanced class distribution (3481:194:95:2) while AUC [55] have been used for evaluation of models on imbalanced domains. On the other hand, in Hypothyroid, the size of the resulting sample obtained by PBS is 462 with 164 informative data points identified on far borders as compared with 3394 samples in the full training set and 291 instances by the traditional BI. However, the performance of the classifiers built by using the BI is much poorer than that built by using PBS.

In sum, the PBS improves NB in most cases while PBS degrades SVM, somewhat, and DT with respect to accuracy in some cases as compared to the modeling on the full training sets. Furthermore, PBS outperforms the traditional BI methods overall, although PBS generally produces a little

larger samples than BI (1.5 times on average). The summary of the paired t-test on the 30 datasets are shown at the bottom of Table 8.4, where w\ \l denote win\draw\lose for PBS in each case.

Table 8.3. PBS for training set reduction on 30 benchmark datasets.

No.	Datasets	#ins	t/r	#PBS	%
0	Anneal	898	3/4	418	46.55
1	Audiology	226	2/2	211	93.36
2	Autos	205	3/3	187	91.22
3	Balance-s	625	3/1	215	34.40
4	Breast-w	699	3/2	257	36.77
5	Colic	368	3/2	257	69.84
6	Credit-a	690	3/2	452	65.51
7	Diabetes	768	6/2	574	74.74
8	Glass	214	3/3	177	82.71
9	Heart-s	270	3/3	200	74.07
10	Hepatitis	155	3/2	79	50.97
11	Hypothyroid	3772	4/4	548	**14.53**
12	Ionosphere	351	6/3	275	78.35
13	Iris	150	3/2	40	26.67
14	Kr-vs-kp	3196	4/2	2440	76.35
15	Labor	57	3/1	33	57.89
16	Letter	20000	5/2	18540	92.70
17	Lymph	148	3/2	121	81.76
18	Mushroom	8124	3/1	1220	**15.02**
19	P-tumor	339	3/3	326	96.17
20	Segment	2310	3/2	1343	58.14
21	Sick	3772	3/3	766	20.31
22	Sonar	208	4/2	174	83.65
23	Soybean	683	2/2	593	86.82
24	Splice	3190	4/2	2847	89.25
25	Vehicle	846	4/2	707	83.57
26	Vote	435	4/1	181	41.61
27	Vowel	990	3/2	970	**97.98**
28	Waveform	5000	4/3	4283	85.66
29	Zoo	101	21	58	57.43
	Average		3/2		65.47

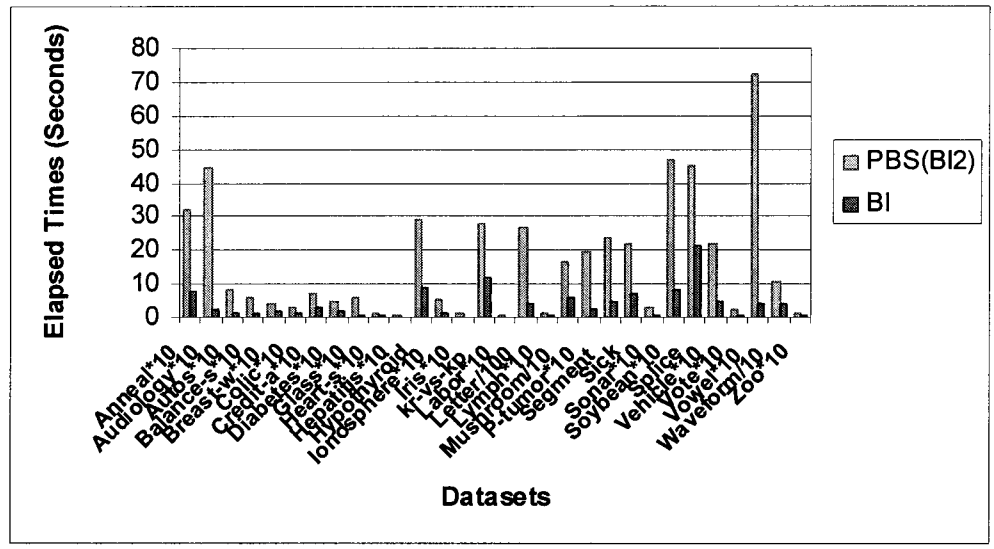

Figure 8.1. The scaled elapsed times of PBS and the traditional BI.
Notes: 30 benchmark datasets were chosen; as we can see, PBS needs a little more runing time than BI for far borders.

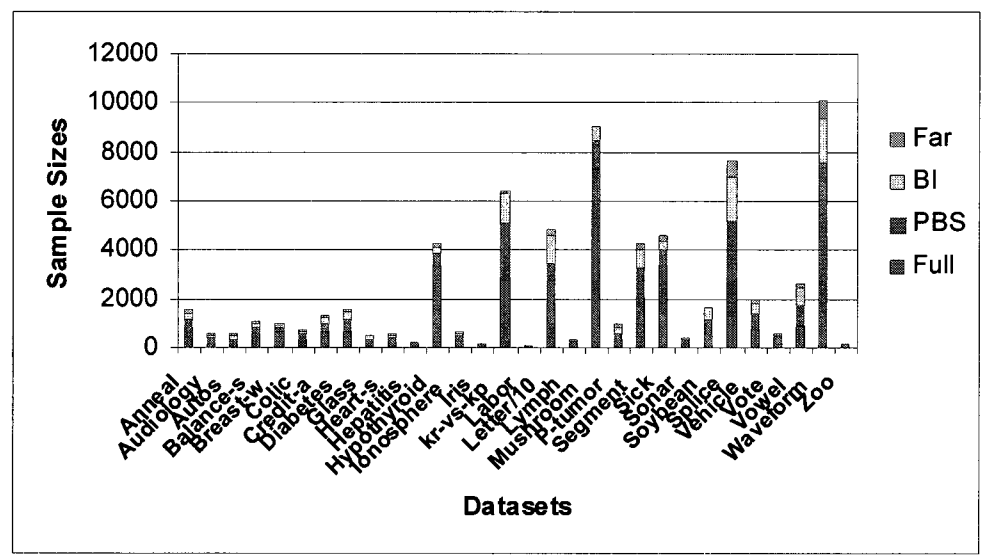

Figure 8.2. The sample sizes of resultant sample generated by Far, the traditional BI, PBS, and Full.
Notes: 30 benchmark datasets were chosen; the sample size by PBS may be a little larger than that by BI in some cases while it can be much smaller than the full size in many cases; there are always far borders in most cases.

Table 8.4. The performance (accuracy) of NB, SVM, and DT built on sample obtained by using PBS, Full, and BI.

Datasets	NB			SVM			DT		
	PBS	Full	BI	PBS	Full	BI	PBS	Full	BI
Anneal	86.81	86.42	82.51 *w*	95.66	96.94 *l*	87.30 *w*	98.33	98.55	95.32 *w*
Audiology	72.35	71.90	73.69	80.71	80.72	80.28	75.39	76.08	74.96
Autos	53.00	54.90	51.49	69.55	69.60	68.31	81.65	83.14	75.79 *w*
Balance-s	89.28	90.63	87.93	87.92	86.56 *w*	87.43	77.69	79.61 *l*	74.66 *w*
Breast-w	96.28	96.07	96.36	96.71	96.71	96.50	93.49	94.06	93.06
Colic	79.73	78.76	79.33	81.54	82.20	77.32 *w*	80.30	82.06	79.50
Credit-a	80.29	77.61 *w*	81.67 *l*	85.14	84.64	84.13 *w*	80.87	82.46	78.77
Diabetes	74.29	75.26	72.27	76.11	76.70	73.77 *w*	71.56	73.84	65.24 *w*
Glass	54.23	46.99 *w*	51.93	55.35	58.32	50.89 *w*	68.68	69.58	65.63
Heart-s	85.00	83.33 *w*	83.70	82.78	83.89	81.30	74.63	75.37	74.63
Hepatitis	85.08	83.19	85.40	83.15	84.54	78.27 *w*	77.33	78.54	70.65
Hypothyroid	70.76	95.32 *l*	45.27 *w*	84.40	93.64 *l*	47.77 *w*	94.96	99.54 *l*	92.14 *w*
Ionosphere	83.50	82.63	84.21	88.03	88.18	83.76 *w*	90.48	89.90	87.46 *w*
Iris	94.33	95.00	94.33	92.33	96.67 *l*	89.67	93.33	95.00	93.00
Kr-vs-kp	93.13	87.81 *w*	92.68 *w*	96.07	95.90	96.23	99.34	99.41	99.30
Labor	91.17	93.83	89.50	89.67	93.83	83.33 *w*	79.67	81.50	77.33
Letter	64.46	64.02 *w*	64.60	82.25	82.29	82.26	87.84	88.05	87.96
Lymph	82.43	82.79	83.10	87.17	86.86	88.21	79.45	74.12 *w*	73.33 *w*
Mushroom	98.18	95.78 *w*	97.94	99.56	100.0 *l*	98.72 *w*	99.97	100.0	99.78 *w*
P-tumor	49.71	50.00	49.71	47.50	48.24	46.17 *w*	42.05	43.37	42.49
Segment	79.48	80.24 *l*	81.93 *l*	92.49	92.86	91.88 *w*	95.24	96.73 *l*	95.48
Sick	94.55	92.74 *w*	90.26 *w*	95.77	93.88 *w*	87.08 *w*	95.68	98.79 *l*	90.51 *w*
Sonar	69.01	68.81	74.49	81.01	78.15	76.39 *w*	75.02	76.95	73.52
Soybean	92.97	92.90	92.89	93.41	93.41	93.41	90.92	90.63	90.84
Splice	95.61	95.44 *w*	95.99 *l*	93.34	93.40	93.12	92.49	92.57	92.32
Vehicle	46.40	45.32	49.17 *l*	73.35	74.59 *l*	72.11	72.64	71.87	70.50
Vote	93.57	90.23 *w*	94.25	95.74	95.74	94.59 *w*	95.74	95.85	94.14 *w*
Vowel	63.48	63.59	63.54	70.25	69.85	69.90	80.15	82.22 *l*	80.35
Waveform	79.75	79.97 *l*	79.77	86.37	86.48	86.42	74.84	75.12	74.71
Zoo	95.50	95.05	95.50	96.00	96.00	95.00	93.00	93.50	92.50
Average	79.81	79.88	78.85	84.64	85.36	81.38	83.76	84.61	81.86
t-test		9\18\3	4\22\4		2\23\5	15\15\0		1\24\5	10\20\0

8.4.1.1 Summary of PBS Results

A labelled training set contains a latent border. A proper border can be defined as an augmented border consisting of all near and far borders for supervised learning. Our proposed algorithm, the Progressive Border Sampling (PBS) technique, can progressively learn an effective sample by using BI_2, which overcomes the limitation of the traditional Border Identification (BI) method. It is shown that PBS can produce more effective

samples than the traditional BI for training classifiers. Our experimental results on 30 benchmark datasets from the UCIKDD repository show that PBS helps build classifiers similar to those built on full training sets in most cases (87 for win or draw versus 13 for lose) and overwhelmingly outperforms the traditional BI technique for the reduction of training sets.

8.4.1.2 Remarks about PBS Results

Border Sampling (BS) technique, e.g., PBS, learns a potential border close to the class boundary defined in a labelled training set. Therefore, it is only biased to the class boundary while traditional classifiers are biased to their own decision boundaries. This explains why BS is learner-independent.

The selected 30 benchmark datasets are regarded as small datasets in this experiment although some of them have ten thousands and more instances. Large datasets over one hundred thousands instances were not chosen in this experiment because PBS is a quadratic algorithm, and is impractical for sample selection on large datasets. On the other hand, Progressive Sampling (PS) has been used for sampling on large datasets. A comparison between Border Sampling and the PS will be discussed in Section 8.4.2.

PBS intends to produce an effective sample for any classifier. It is learner-independent while most of the previous approaches are designed for training set reduction in Instance-Based Learning (IBL) techniques such as the *k*-Nearest Neighbour classifier. We only compared PBS with the traditional BI for training common classifiers in supervised learning. A comparison between BS and previously proposed other sample selection techniques will not be discussed until Section 8.4.3.

A proper similarity distance metric is crucial for BS. We conducted experiments by using Cosine for border identification in PBS. It was

observed that other similarity metrics such as Radial Based Function (RBF) can also be successfully used in PBS.

However, it is also known that no single similarity metric is superior to other metrics in all cases. In general, Cosine and RBF can be used for PBS, and we assume that variables in RBF are independent of each other such that the distance metric can be efficiently computed. As a result, there is an interest to investigate and develop a proper distance metric used in PBS. This also suggests future work.

8.4.2 Experimental Results on CPBS

We investigated the proposed Coupling Markov Chain Monte Carlo for scaling up Progressive Border Sampling (CPBS) for sample selection on either small or large datasets for supervised learning. We compared the CPBS with the Progressive Sampling (PS) techniques including static (Static), arithmetic (Arith), and geometric PS with LRLS (Geo) for sample selection on large datasets, in particular, on Explosion for a practical application.

We conducted overall experiments on the 33 datasets, which are depicted in Table 8.1. CPBS with Radial Based Function (RBF) similarity metric selects samples from the original training sets with a specified sampling window.

To test the performance of CPBS with different sampling windows on either small datasets or large datasets, the datasets are divided into three groups, as shown in Table 8.5. The sizes of sampling windows on the datasets in the first group (#0-#21), the second group (#22-#26), and the third group (#27-#32, for large datasets) are set with 10, 100, and 1000, respectively.

The experiments were run 2 times on the datasets in the first group and the second group, and 1 times on large datasets in the third group via the 10-Cross Validation. Static was executed by resampling with replacement and the same class distribution, and the same sample size as that of the resulting sample identified by CPBS from the original training sets. Arith and Geo, on the other hand, were executed according to their specified schedules on each run, and the related results within the 10-Cross Validation were averaged for comparison.

The initial results for sample selection are reported in Table 8.5, where the columns are the names of the datasets, the number of instances (#ins), the average tries (T) in CPBS for convergence detection in the B chain, the average iterations (K) for collapsing test in the R chain, the number of data points selected from training sets by using CPBS (#CPBS), and the percent (%) of data selected by using CPBS over the original training sets.

Four inductive algorithms, as depicted in Table 8.2, were used for training classifiers on either the resulting samples generated by using CPBS, or the full training sets (Full), or those generated by using the previous approaches, i.e., Static, Arith, and Geo. The performances of these classifiers with respect to the Area Under ROC Curve (AUC) were used for evaluation between CPBS and PS techniques.

As we can see from our initial results, reported in Table 8.5, CPBS can select a small sample from the original training set after redundancy are removed. For example, CPBS produced samples with only 15.54 and 0.74 percents of the original training sets on Sick and Explosion, respectively, and it kept most of samples in the original training sets when little redundancy was found on Vowel and Splice.

The average tries T is 2 and 22 in Shuttle and Waveform, respectively. The average iteration K for collapsing test in Coupling is 1 and 70 in

Audiology and Shuttle, respectively. Therefore, the T and K are much smaller than the #ins.

Further, we show the efficiency and effectiveness of CPBS for sample selection on large datasets by comparing CPBS with Arith, Geo, and Full for building NB and DT while SVM and IB1 are ignored due to their intractability on large datasets.

Table 8.5. The sizes of B chain and R chain in CPBS on 33 benchmark datasets.

No.	Datasets	#ins	#CPBS	%	T	K
0	Anneal	898	418	51.72	3	2
1	Audiology	226	183	89.97	3	1
2	Autos	205	170	92.14	3	1
3	Balance-s	625	540	96.00	3	6
4	Breast-w	699	173	27.50	2	3
5	Colic	368	245	73.97	5	4
6	Credit-a	690	543	87.44	3	5
7	Diabetes	768	531	76.82	2	4
8	Glass	214	166	86.19	2	1
9	Heart-s	270	219	90.12	3	5
10	Hepatitis	155	66	47.31	3	3
11	Ionosphere	351	222	70.28	2	4
12	Iris	150	62	45.93	2	1
13	Labor	57	40	77.97	2	2
14	Lymph	148	117	87.84	4	3
15	P-tumor	339	295	96.69	3	1
16	Sonar	208	160	85.47	5	3
17	Soybean	683	603	98.10	3	1
18	Vehicle	846	690	90.62	3	1
19	Vote	435	162	41.38	16	5
20	Vowel	990	891	100.00	2	1
21	Zoo	101	75	82.51	2	1
22	Hypothyroid	3772	558	16.44	4	3
23	kr-vs-kp	3196	2434	84.62	10	6
24	Segment	2310	1883	90.57	3	5
25	Sick	3772	528	15.54	3	4
26	Splice	3190	2847	99.16	6	3
27	Letter	20000	16627	92.37	13	3
28	Mushroom	8124	3440	47.05	3	12
29	Waveform	5000	4256	94.58	22	7
30	Adult	48842	24665	56.11	2	13
31	Shuttle	58000	18254	34.97	2	70
32	Explosion	92630	620	0.74	2	24

We compared CPBS with the Arith for building NB and DT on Adult and Shuttle, as shown in Figure 8.3, where the resulting sample size by CPBS is denoted by the position of the vertical lines.

According to two graphs at the top of Figure 8.3, on Adult, Arith has a higher time cost for sampling than CPBS has after the queried sample size for NB or DT exceeds 6300 or 9400, respectively, while on Shuttle, Arith has a higher time cost for sampling than CPBS has after the queried sample size for NB or DT exceeds almost 6300 or 18700, respectively.

According to two graphs at the middle of Figure 8.3, no matter whether a large sample or a small sample is queried, however, Arith tends to degrade the performance of NB and DT as compared with CPBS. On Adult for NB, Arith approximately obtained the same performance as CPBS after it queried 3200 samples. However, Arith degraded the performance of DT on Adult as compared with CPBS. This suggests that CPBS outperforms the Arith for effective sample selection, and it is very competitive with the Full except one case in that CPBS degraded the performance of DT according to the AUC on Adult.

We also compared CPBS with Geo. The related results are omitted here due to the similar results. In sum, Geo can efficiently perform sampling while it is subject to failure in selecting an effective sample as compared with CPBS.

Finally, we summarized the results about CPBS, Full, and Static for building NB and DT on the datasets in the second group and the third group, as shown in Table 8.6, where '*w*' and '*l*' represent that CPBS wins and loses against the corresponding approaches, respectively, in terms of both the paired *t*-test and the Wilcoxon signed rank test at a significant level of 0.05.

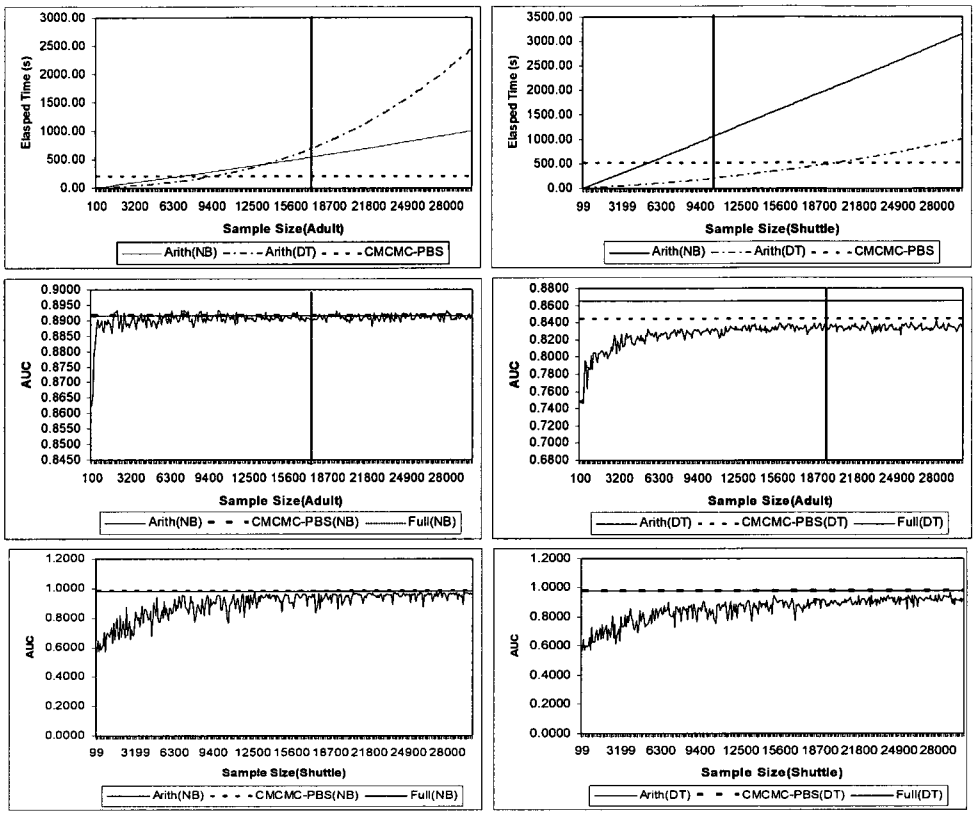

Figure 8.3. The comparison between CPBS and Arith about elapsed times and AUC for training NB and DT on Adult and Shuttle.

As a result, CPBS consistently outperforms Static for training NB and DT, and consistently outperforms the Full for training NB. It is very competitive with Full for DT except in the case on Adult for building DT.

The experimental results were also obtained on the first group, as shown in Table 8.7, where the average AUCs are shown at the bottoms of Table 8.7. As we can see, the CPBS can significantly improve NB in some cases while it ony degrades NB in the Balance-s and Inonsphere cases as compared with the Full according to the paired t-test. CPBS does not degrade DT in all cases, and CPBS significantly improves SVM and IB1 in the Anneal case while it maintains the performance of SVM and IB1 in the other cases as compared with Full.

Furthermore, CPBS outperforms the Static method to produce effective samples for building better classifiers, i.e., CPBS is better than this random method in most cases. CPBS only lost to Static in the Balance-s and Ionosphere cases for training IB1 and NB, respectively.

For another worse case, CPBS degrades the performance of DT on Adult as compared with Full. This suggests a further investigation on any possible improvement of the proposed algorithm.

Explosion is a scientifically synthesized domain. The experimental results on Explosion in Table 8.5 revealed that CPBS is superior to Static for training NB and DT while it is competitive with Full for training NB and DT, and it produces a quite small sample for training by removing much redundancy.

To investigate the possible effect of the sampling windows on the performance of CPBS, we repeated our experiments with incremental window sizes on the same datasets. The increments of sampling window sizes for sample selection using CPBS on the datasets in the first group, the second group, and the third group were set with 10, 100, and 100, respectively. As a result, we obtained other results, which correspond to different window sizes. Therefore, we can draw the AUC curves with respect to the sampling window sizes.

For example, as shown in Figure 8.4, we drew the curves of AUC with respect to NB, DT, SVM, and IB1, which were built by using CPBS with different window sizes on Anneal, and drew the curves of AUC with respect to NB and DT on Splice and Shuttle. As a result, we found a little impact of sampling windows on the resulting AUCs, and there is no negative effect from the sampling window on the performance of CPBS except for the case on Splice for building DT.

Table 8.6. Performance (AUC) of NB and DT built on the second and third groups by using CPBS, Full, and Static.

Datasets	NB			DT		
	CPBS	Full	Static	CPBS	Full	Static
Hypothyroid	.9378	.9399	.9267	.945	.9623	.9521
kr-vs-kp	.9812	.9521^{ww}	.9482^{ww}	.9987	.9983	.9925^{ww}
Letter	.9572	.9552^{ww}	.9546^{ww}	.9498	.9509	.9336^{ww}
Mushroom	.9994	.9981^{ww}	.9973^{ww}	.9999	1	1
Segment	.9779	.9779	.9764w	.9836	.9836	.9809
Sick	.922	.9271	.9238	.967	.9525	.9137^{ww}
Splice	.9947	.9944^{-w}	.9939^{ww}	.9515	.9531	.9444
Waveform	.9619	.9567^{w}	.9551^{w}	.828	.8255	.8156^{w}
Adult	.8915	.8914	.8916	.849	.8649^{ll}	.8307^{ww}
Shuttle	.9895	.9782^{ww}	.9383^{ww}	.9804	.9798	.9322^{-w}
Explosion	.5427	.4172^{ww}	.5272	.68	.6953	.5^{ww}
Average	**.9233**	.9080	.9121	.9212	.9242	.8905

Table 8.7. Comparison (AUC) between CPBS and Full, Static for training NB, DT, SVM, and IB1 on the first group.

Datasets	NB			DT			SVM			IB1		
	CPBS	Full	Static	CPBS	Full	Static	CPBS	Full	Static	CPBS	Full	Static
Anneal	0.9597	0.9599^{-l}	0.9514^{-w}	0. 8292	0.819	0.7899^{-w}	0.8484	0.8408^{-w}	0.8371^{-w}	0. 8213	0.7975^{ww}	0.8216
Audiology	0.7024	0.7017^{-w}	0.6987^{-w}	0. 6212	0.6196	0.6141	0.6436	0.6433	0.6243^{ww}	0.5993	0.6039	0.594
Autos	0.7225	0.7251	0.7025	0.735	0.7369	0.7159	0.7724	0.7733	0.7619	0.6933	0.6942	0.6618^{ww}
Balance-s	0.8738	0.8789^{l}	0.8445	0.6852	0.6721	0.7121	0.6648	0.6633	0.6642	0.675	0.6969	0.6897
Breast-w	0.9903	0.9879	0.9892	0.9421	0.9483	0.9372	0.963	0.9635	0.9625	0.9417	0.9485	0.9593^{-l}
Colic	0.8532	0.8372^{w}	0.8342	0.8487	0.8533	0.8171	0.7951	0.8095	0.7669	0.7673	0.7795	0.7518
Credit-a	0.8997	0.8982	0.8959	0.8294	0.8479	0.8427	0.8599	0.8572	0.8504	0.7998	0.8075	0.7986
Diabetes	0.8168	0.8174	0.8135	0.7574	0.7697	0.6868^{ww}	0.7131	0.7114	0.7143	0.6694	0.6637	0.6509
Glass	0.8111	0.8116	0.8083	0.7959	0.7924	0.7253^{ww}	0.7208	0.7305	0.7272	0.7331	0.7353	0.7302
Heart-s	0.8972	0.8981	0.8931	0.7947	0.759	0.7486	0.835	0.8313	0.8183	0.7588	0.7596	0.77
Hepatitis	0.8878	0.8797	0.8465	0.7276	0.7176	0.7223	0.7705	0.7474	0.7159	0.6671	0.658	0.6712
Ionosphere	0.9159	0.9390^{l}	0.9360^{l}	0.8803	0.8902	0.8664	0.8421	0.8464	0.8238	0.8311	0.8246	0.7800^{ww}
Iris	0.99	0.9893	0.9907	0.9667	0.9713	0.9647	0.96	0.9833	0.965	0.9667	0.9783	0.975
Labor	0.9646	0.9771	0.9771	0.8125	0.7854	0.8354	0.8792	0.8917	0.7958^{-w}	0. 8417	0.8479	0.8417
Lymph	0.8921	0.8922	0.8818	0.7303	0.7083	0.7064	0.8128	0.8105	0.7893	0.6927	0.6884	0.6697^{-w}
P-tumor	0.7613	0.7613	0.7581	0.6452	0.6469	0.6233^{-w}	0.715	0.7132	0.7077	0.5827	0.587	0.591
Sonar	0.8484	0.7984^{-w}	0.7862^{-w}	0. 7325	0.7631	0.6994	0.801	0.7721	0.7501^{-w}	0. 8692	0.8595	0.8130^{-w}
Soybean	0.9983	0.9983	0.9982	0.9743	0.9722	0.9525^{ww}	0.988	0.9881	0.9876	0.9674	0.968	0.9648
Vehicle	0.7498	0.7462	0.7393^{-w}	0. 7933	0.813	0.7663	0.8306	0.833	0.8243	0.7419	0.7404	0.7265
Vote	0.9887	0.974^{ww}	0.9741^{ww}	0.9745	0.9785	0.9577^{-w}	0.9571	0.9567	0.955	0.8938	0.9226	0.917
Vowel	0.9547	0.9547	0.9351^{ww}	0.9269	0.9269	0.8928^{ww}	0.9484	0.9483	0.9320^{ww}	0.9956	0.9956	0.9766^{ww}
Zoo	0.8917	0.8917	0.8821	0.7976	0.7976	0.7917	0.8048	0.8048	0.8095	0.8024	0.8024	0.803
Average	**0.8805**	0.8781	0.8698	**0.8091**	0.8086	0.7895	**0.8239**	0.8236	0.8083	0.7869	0.7891	0.7799

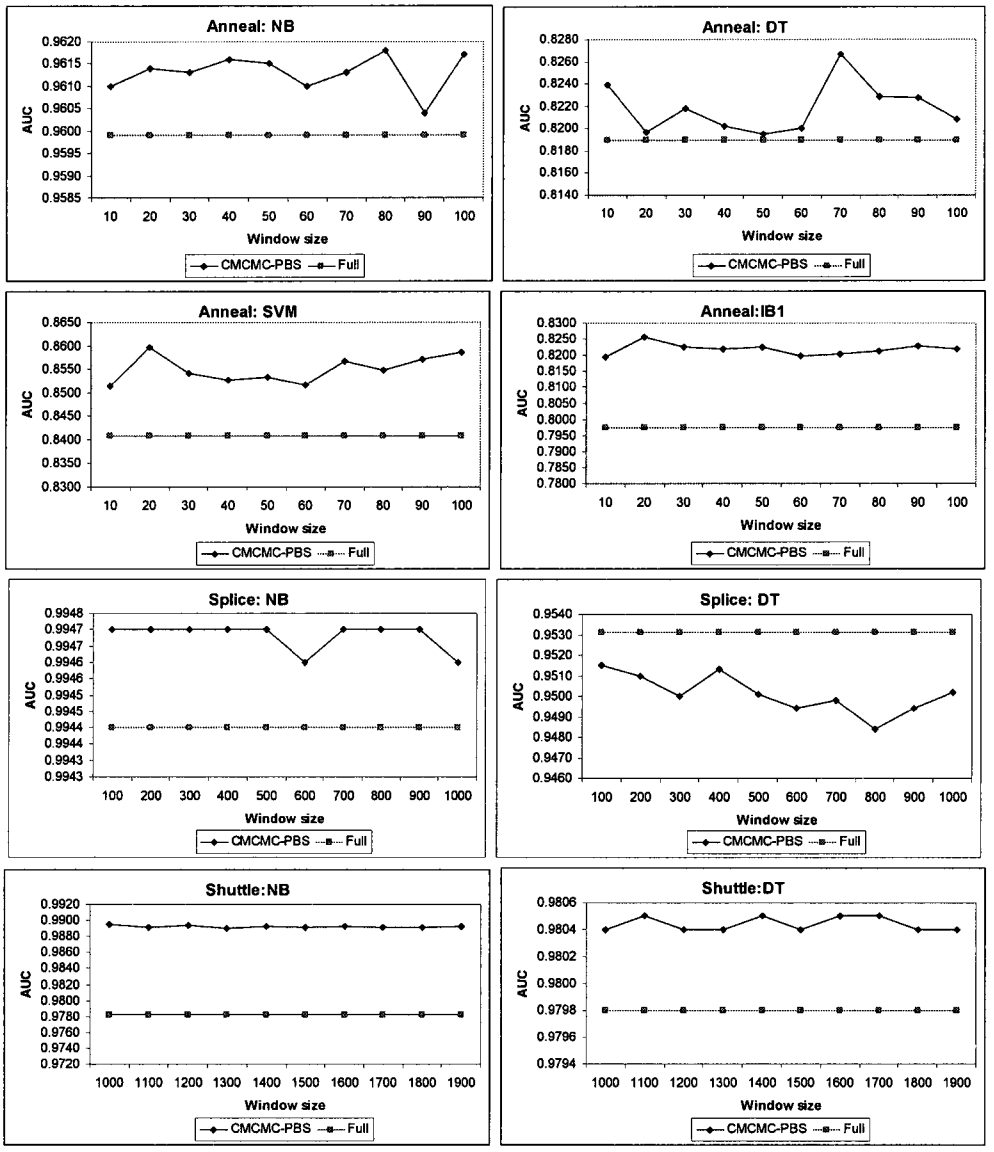

Figure.8.4. The effect of window size of CPBS.

8.4.2.1 Summary of CPBS Results

We discuss the scalability of the previously introduced CPBS for border sampling on large labelled datasets because PBS is infeasible on large datasets. CPBS incorporates PBS with the CMCMC technique. It can

efficiently produce an effective sample by learning many small samples in linear time complexity.

For example, CPBS is 60% and 40% faster than PBS for Border Sampling on Letter and Splice, respectively.

CPBS outperforms the previous three methods, Static, Arith, and Geo for sample selection although it has a little overhead over Arith and Geo. It is better than Full for training NB and SVM, and is very competitive with Full for training DT and IB1. There is no evidence to show a negative impact of the sampling window on the resulting samples for training. Therefore, CPBS is efficient and effective for sample selection in practical applications.

CPBS was applied to sample selection on Explosion, which is a scientifically synthesized dataset. The experimental results on Explosion revealed that CPBS is superior to Static (a random sampling method) for training NB and DT while it is competitive with the Full for training NB and DT with respect to the AUC scores.

8.4.2.2 Remarks about CPBS Results

As discussed in Section 2.3.3.2 and Section 5.4, the Adaptive Recursive Partition for Prototype Reduction Schema (ARP_PRS) is a recently proposed approach for scaling up the PRS on large datasets. In our recent research, we do not conduct any experiment for an empirical comparison between CPBS and the ARP_PRS for sample selection on large datasets. This work can be done in our future research.

CPBS for Border Sampling on large datasets suggests learning an integral probability distribution from the related small probability distributions by assuming the Coupling Markov Chain Monte Carlo technique. The technique is quite different from a traditional MCMC technique, which has a low possibility to converge to a stationary probability distribution. CPBS takes

into account the evolution of statistics from subsamples to a large population. The main problem is how to define the condition for convergence detection.

Our study attempts to establish a connection between CPBS and Coupling From The Past (CFTP). CFTP is a state of the art Markov Chain Monte Carlo technique, which suggests several Markov chains start at the same position in the past, and finally converge to the current same position. CPBS borrows the main idea behind CFTP for producing an effective sample in supervised learning.

8.4.3 Experimental Results on TPBS

We conducted experiments to compare TPBS with previous instance selection techniques, i.e., CNN, ENN, RENN, and DROP3.1, as well as CPBS for supervised learning.

Both TPBS and CPBS were run with Radial Based Function (RBF) and the sampling window size set with W = 100 (instead of W = 1000, used in experiments in Section 8.4.2). Four inductive algorithms, as depicted in Table 8.2, were used for training classifiers on either the resulting samples generated by TPBS, CPBS, or the full training sets (Full), or those generated by CNN, ENN, RENN, and DROP3.1. The performances of these classifiers with respect to the Area Under ROC Curve (AUC), which, on average, were obtained over 2 runs of the 10-Cross Validation, were used for evaluation between TPBS and other approaches.

Our experiments were conducted on 16 datasets, which were selected from Table 8.1. All datasets were divided into two groups, namely, 10 small and 6 large datasets, as shown in Table 8.8 and Table 8.9, respectively, where the columns #ins is the number of instances; #A, #B, and #R are the sizes of the A chain, B chains, and R chains in TPBS, respectively; #TPBS, #CPBS, #CNN, #ENN, #RENN, and #DROP3.1 are the sample sizes obtained by

using TPBS, CPBS, CNN, ENN, RENN, and DROP3.1, respectively; The % is represented as the ratio of #TPBS over #ins; The t-TPBS and t-CPBS are the elapsed times (secs.) of TPBS and CPBS for border sampling on large datasets, respectively.

Our initial experimental results for sample selection, reported in Table 8.8, show that TPBS produced a small sample from the original training set after redundancy was removed. For example, TPBS produced 566 samples and 171 samples from 898 * 0.9 = 808 and 629 samples in the original Anneal and Breast-w, respectively, while it kept most instances in Vowel because a little redundancy were found. In general, TPBS produces fewer samples than previous instance selection approaches except for DROP3.1, which consistently produces the fewest samples. However, TPBS produces a little more sample than CPBS except on Mushroom and Adult cases, as shown in Table 8.9.

The sizes of the A, B, and R chains of TPBS on the small datasets are less than 7 while the maximum size of the R chains is 11 in Mushroom, as shown in Table 8.8 and 8.9, respectively. Empirically, they are much smaller than #ins.

Table 8.8. The sample sizes of TPBS, CPBS, CNN, ENN, RENN, and DROP3.1 on 10 small datasets.

Datasets	#ins	#A	#B	#R	#TPBS	%	#CPBS	#CNN	#ENN	#RENN	#DROP3.1
Anneal	898	3	7	7	566	0.70	395	795	803	803	121
Audiology	226	3	5	4	195	0.96	168	181	159	149	91
Autos	205	2	5	3	180	0.98	171	181	157	151	112
Balance-s	625	4	7	6	495	0.88	448	559	510	510	142
Breast-w	699	2	7	7	171	0.27	147	629	614	614	32
Ionosphere	351	4	4	5	271	0.86	227	316	288	281	59
Vehicle	846	2	4	5	756	0.99	739	758	652	636	350
Vote	435	4	6	6	153	0.39	103	392	369	365	38
Vowel	990	2	5	4	891	1.00	855	879	886	886	526
Zoo	101	2	3	3	70	0.77	39	63	87	87	21
Average	538	3	5	5	**375**	0.78	329	475	452	448	**149**

Table 8.9. Elapse times of TPBS and CPBS on 6 large datasets.

Datasets	#ins	#A	#B	#R	#TPBS	#CPBS	t-TPBS(s)	t-CPBS(s)
Letter	20000	2	6	6	17942	15710	**684.2**	1595
Mushroom	8124	2	11	6	**2294**	5640	148	101
Waveform	5000	3	6	9	4482	4300	317.8	116.8
Adult	48842	2	3	7	**25936**	25975	208.1	78.3
Shuttle	58000	2	9	9	19418	18731	230.6	146.5
Explosion	92630	2	7	5	641	637	91.0	23.3
Average	38766	2	7	7	11786	11832	280	343.5

Further, we show the efficiency of TPBS for sample selection on large datasets by comparing TPBS with CPBS, as shown in Table 8.9. As we can see, TPBS spent a little more time than CPBS for sample selection on large datasets in most cases except for the Letter case.

Mushroom is a case that shows that CPBS may suffer from a difficulty for a sufficient reduction with a specified sampling window. Because of many redundancies in Mushroom, it is interesting to compare TPBS with CPBS for sample selection on large datasets with respect to different window sizes. We conducted an additional experiment to compare TPBS with CPBS for Border Sampling on Mushroom by incrementing the window sizes in TPBS and CPBS from 100 to 1000 with an increment of 100, as shown in Figure 8.5. The sample sizes obtained by TPBS fall between 2300 and 3800 while the sample sizes by CPBS fall between 3100 and 5800. After the window size is greater than 800, the resulting sample produced by TPBS has almost the same size as that by CPBS.

Therefore, TPBS produces smaller sample than CPBS with a small sampling window.

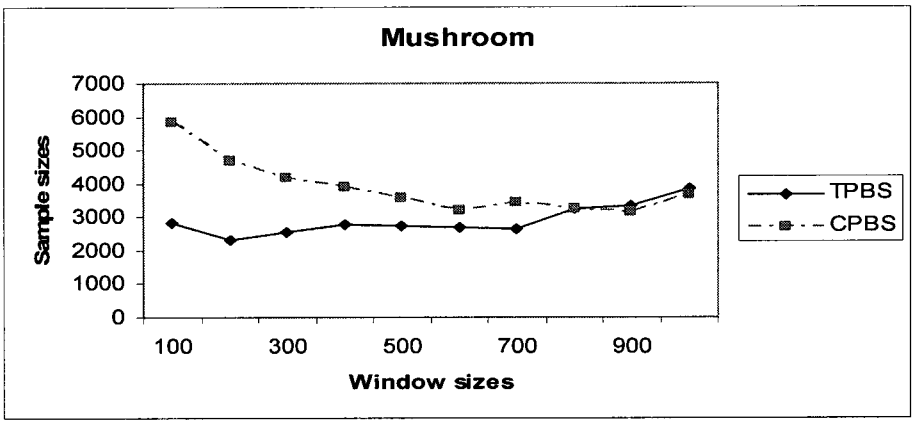

Figure 8.5. Sample Sizes of TPBS and CPBS on Mushroom with different window sizes.

We show the effectiveness of TPBS by comparing TPBS with Full, CPBS, ENN, RENN, and DROP3.1 for training classifiers: NB, DT, SVM, and 1NN (i.e., IB1) on the small datasets, reported in Table 8.10 and Table 8.11. The datasets are expressed by their abbreviations, and we use '*w*' and '*l*' to represent TPBS's wins and losses, respectively, against the corresponding methods in terms of the paired t-test (first) and Wilcoxon signed rank test (second) at significance levels of 0.05.

As we can see, TPBS helps learn NB in most cases, as shown in Table 8.10, and does not degrade the performances of DT, SVM, and 1NN built on the reduced samples in all cases as compared with Full. TPBS outperforms CPBS for training classifiers in all cases in terms of the paired *t*-test and in most cases in terms of the signed rank test. TPBS outperforms ENN, RENN, DROP3.1 for training the classifiers in most cases in terms of the paired t-test and Wilcoxon signed rank test.

ENN, RENN, and DROP3.1 are superior to TPBS for training 1NN in a few cases, as shown in Table 8.11. Several cases shown in boldface suggest that noise removal by using ENN can be helpful for training classifiers, especially for training 1NN. This means that these methods have a bias towards 1NN while they are inferior to TPBS for training other classifiers in

most cases. We conducted related experiments by incorporating ENN with TPBS for removing noise. As a result, TPBS is competitive with ENN and DROP3.1 for training 1NN on Anneal by obtaining a AUC score of 0.8231 instead of the original AUC score 0.7986.

We emphasize the experimental results about TPBS and CPBS for training two classifiers on large datasets, as shown in Table 8.12. As we can see, TPBS helps learn NB and does not degrade the performance of DT in all cases in terms of the paired *t*-test as compared with Full, and outperforms CPBS for training the two classifiers in all cases in terms of the paired *t*-test. On the other hand, the experimental results on Explosion revealed that both TPBS and CPBS are successful in this case.

Table 8.10. Comparison between TPBS and Full, CPBS, ENN, RENN, DROP3.1 for training NB and DT.

	NB						DT					
Dat.	TPBS	Full	CPBS	ENN	RENN	DROP3.1	TPBS	Full	CPBS	ENN	RENN	DROP3.1
Ann.	0.9612	0.9599 ◦	0.9618 •	0.9603 ◦	0.9602 ◦	0.9517 ◦◦	0.8156	0.819	0.8105	0.7985 ◦	0.7968 ◦	0.8275
Aud.	0.7019	0.7017	0.7007 ◦	0.6912 ◦◦	0.691 ◦◦	0.6876 ◦◦	0.6206	0.6196	0.6221	0.5937 ◦◦	0.5936 ◦◦	0.5976 ◦◦
Aut.	0.7256	0.7251	0.7245	0.6695 ◦◦	0.6681 ◦◦	0.6665 ◦◦	0.7374	0.7369	0.7372	0.7015	0.6927 ◦	0.6926 ◦
Bal.	0.8735	0.8789	0.8466 ◦◦	0.8303 ◦	0.8261 ◦◦	0.8035 ◦◦	0.6828	0.6721	0.6935	0.7196 •	0.7202 •	0.6929
Bre.	0.9936	0.9879 ◦◦	0.994	0.9923	0.9923	0.9853 ◦	0.934	0.9483	0.9208	0.9509	0.9512	0.9239
Ion.	0.9235	0.939 •	0.9039 ◦	0.9391 •	0.9377 •	0.8434 ◦◦	0.8821	0.8902	0.9028	0.8736	0.8709	0.7872 ◦◦
Veh.	0.7477	0.7462	0.7397 ◦	0.7441	0.7396 ◦	0.7077 ◦◦	0.8128	0.813	0.8161	0.8051	0.7915	0.7909
Vot.	0.9851	0.974 ◦◦	0.9911 •	0.9731 ◦◦	0.9726 ◦◦	0.9909	0.9695	0.9786	0.9614	0.9713	0.9666	0.9677
Vow.	0.9548	0.9547	0.9562	0.9538 ◦◦	0.9538 ◦◦	0.9671 ‖	0.9273	0.9269	0.9317	0.921	0.9206 ◦	0.9108 ◦
Zoo	0.8917	0.8917	0.8917	0.8845	0.8845	0.876	0.7952	0.7976	0.7976	0.7911	0.7911	0.792
Ave.	0.8759	0.8759	0.8710	0.8638	0.8626	0.8480	0.8177	0.8202	0.8194	0.8126	0.8095	0.7983

Table 8.11. Comparison between TPBS and Full, CPBS, ENN, RENN, DROP3.1 for training SVM and 1NN.

	SVM						1NN					
Dat.	TPBS	Full	CPBS	ENN	RENN	DROP3.1	TPBS	Full	CPBS	ENN	RENN	DROP3.1
Ann.	0.8375	0.8408	0.8432 •	0.8424	0.8421	0.8547	0.7986	0.7975	0.812 •	**0.823** ‖	**0.8245** ‖	**0.8209** ‖
Aud.	0.6434	0.6436	0.6454	0.6046 ◦◦	0.5988 ◦◦	0.6101 ◦◦	0.5985	0.6006	0.6035	0.5842 ◦◦	0.5849 ◦◦	0.5937
Aut.	0.7731	0.7734	0.7729	0.7366 ◦◦	0.7347 ◦◦	0.7329 ◦◦	0.6942	0.6942	0.6943	0.6519 ◦	0.653 ◦	0.6516 ◦
Bal.	0.6882	0.6678	0.648 ◦	0.7206 •	0.7245 •	0.7187 •	0.6655	0.6678	0.6722	**0.7046** ‖	**0.7109** ‖	0.6825
Bre.	0.965	0.9635	0.964	0.9681	0.9687	0.964	0.9459	0.9485	0.9243 ◦	0.962 •	0.9609 •	0.9247
Ion.	0.8475	0.8464	0.8387	0.8362	0.8233 ◦	0.6284 ◦◦	0.8246	0.8246	0.8265	0.796 ◦	0.7949 ◦	0.7895 ◦
Veh.	0.8318	0.8331	0.8321	0.818 ◦◦	0.8078 ◦◦	0.7893 ◦◦	0.7398	0.7404	0.7431	0.7281	0.724	0.7183
Vot.	0.9567	0.9567 ◦	0.9538 ◦	0.948 ◦	0.95 ◦	0.9407 ◦◦	0.9011	0.9273	0.9083	0.9246 ◦◦	0.9202 ◦◦	0.9144 ◦◦
Vow.	0.9482	0.9484	0.947	0.947	0.9466	0.9586 ‖	0.9956	0.9956	0.9956	0.9908 ◦◦	0.9905 ◦◦	0.9902
Zoo	0.8048	0.8048	0.806	0.8024 ◦	0.8024	0.8036	0.8024	0.8024	0.8048	0.7964	0.794	0.8128
Ave.	0.8296	0.8279	0.8251	0.8224	0.8199	0.8001	0.7966	0.7999	0.7985	0.7962	0.7958	0.7899

Table 8.12. Comparison (AUC) between TPBS and CPBS for building NB and DT.

	NB					DT				
Datasets	TPBS	Full		CPBS		TPBS	Full		CPBS	
Letter	0.9554	0.9552	ww	0.9552	ww	0.9506	0.9509		0.9509	
Mushroom	0.9981	0.9981	-w	0.9981	-w	1	1		1	
Waveform	0.9567	0.9567		0.9567		0.8253	0.8255		0.8255	
Adult	0.8909	0.8914	-l	0.8909		0.8631	0.8649		0.8483	ww
Shuttle	0.9858	0.9782	ww	0.9864	-l	0.9804	0.9798		0.9804	ww
Explosion	0.5091	0.4172	ww	0.534		0.7357	0.6953	-w	0.7453	
Average	0.8209	0.7902		0.7446		0.8432	0.8231		0.8481	

8.4.3.1 Summary of TPBS Results

Border Sampling (BS) is a new technique for instance selection. We developed an effective geometric computation which describes the propagation of nearest neighbours in BS. A new algorithm, called Tripled Markov Chain Monte Carlo for Progressive Border Sampling, denoted as TPBS, is proposed by incorporating this effective geometric computation with the Coupling Markov Chain Monte Carlo for Progressive Border Sampling (CPBS).

TPBS is regarded as an alternative method of CPBS for scaling up border sampling on large datasets by improving the algorithmic convergence. Our experimental results show that TPBS needs little more time to produce little more samples than CPBS in most cases while TPBS is more effective than CPBS to build classical classifiers. TPBS is quite scalable on large datasets similar to CPBS due to its approximate linear time complexity.

On average, TPBS produces fewer samples than CNN, ENN, and RENN except for DROP3.1. Further, TPBS outperforms previous instance selection approaches in producing effective samples for training several traditional classifiers while previous instance selection techniques have a bias towards 1NN learner.

8.4.3.2 Remarks about TPBS Results

The Border Sampling technique attempts to build several Markov Chains such that the resulting sample can approach to a stationary probability distribution. CPBS produces an effective sample from a large population by building coupled Markov Chains, i.e., B chain and R chains while TPBS builds tripled Markov Chains, i.e., A chain, B chains, and R chains. Both intend to scale up the original quadratic PBS on large datasets.

TPBS is regarded as an alternative method of CPBS by assuming an effective geometric computation for algorithmic convergence. TPBS makes a further effort for an effective Border Sampling whereas how to define effective Markov Chains remains a challenging task.

Border Sampling can be regarded as a learning technique for supervised learning. Essentially, BS can be incorporated into an induction algorithm for training a successful classifier.

According to our current research, we can describe the crucial elements of the Border Sampling technique as follows:

- Border Identification in Two Stages (BI_2). BI_2 algorithm improves the traditional BI technique by identifying a full border. A theoretical foundation has been established, as discussed in Chapter 5. It shows that BI_2 can remove all redundant data and produce a full border.
- A similarity distance metric. A new method for combining continuous, nominal, and missing values in a distance metric is discussed in Section 3.5. It is believed that a proper similarity distance metric for BI_2 is required to help reduce redundancy because a different similarity distance metric defines different borders.
- A class binaritization method for Border Sampling on multi-class domains. In this thesis, all experiments were conducted by assuming the pairwise strategy. However, there is no sufficient reason to reject

the one-against-all (oa) strategy in BS. BS with the oa method should be paid an extreme attention when we are concerned about a discriminative effect between a class and the rest of the classes. Our research suggests an unceasing scrutiny about the two strategies for BS on multi-class domains, especially when BS is used as a wrapped method for training classifiers.

8.4.4 Experimental Results on CCC

We conducted experiments on the first 30 benchmark datasets, as depicted in Table 8.1, to evaluate Cascading Customized Couple (CCC) for scaling up NB and NB-like classifiers.

In our experiments, we built two CCC classifiers: the first CCC ensembles a couple of CC classifiers consisting of only uniform NB classifiers, thus denoted as CCCNB, and the second CCC ensembles a couple of CC classifiers consisting of uniform NB or four NB-like classifiers, i.e., AODE, AODEsr, HNB, and WAODE, by assuming the dynamic selection described in Section 7.3.5.

We compared CCC with previously proposed 10 classifiers such as SBC, TAN, as depicted in Table 8.2, for scaling up NB and NB-like classifiers. In our experiments, all approaches ran 2 times via the 10-fold cross validation, and these previously proposed classifiers have been modified such that they can be used in CCC as follows:

- Each classifier can remember the number of original classes in the original domain by simply setting a new property *numOriginalClasses* representing the number of the original classes;
- The new average rule, as shown in Figure 7.3 in Section 7.3.2.3, has been added into the Vote classifier in Weka;

- A new capability for building a uniform NB was added into the original NB algorithm in Weka by setting a Boolean property *uniform* = *true*, and the classification rule in NB, which is defined in (7.1) in Section 7.2.2, has been modified such that a traditional NB become uniform.

We first show that CCCNB significantly enhances NB in most cases as compared with BNB(AdaBoost and NB), SBC, TAN, and NBTree, as shown in Table 8.13, where '*w*' and '*l*' represent that CCCNB wins and losses, respectively, against the corresponding methods in terms of the paired *t*-test (first) and Wilcoxon signed rank test (second) at significance levels of 0.05.

For example, in Balance-s, CCCNB significantly scales up NB, but is inferior to BNB for scaling up NB. In sum, as we can see, CCCNB consistently enhances NB in terms of paired *t*-test, and enhances NB in most cases in terms of the signed rank test as compared with Full; it outperforms BNB to scale up NB in most cases and only loses in three cases; in particular, it consistently outperforms SBC, which performs feature selection for NB, to scale up NB, and even outperforms TAN and NBTree, which attempt to learn a simple Bayesian structure and a tree structure, respectively, for scaling up NB in some cases.

Table 8.13. Comparison between CCCNB and several classifiers for scaling up NB.

Datasets	CCCNB	NB		BNB		SBC		TAN		NBTree	
Anneal	0.9603	0.9601		0.9379		0.9582		0.966	*-l*	0.9607	
Audiology	0.7038	0.7012	*-w*	0.6921	*-w*	0.6923	*-w*	0.7026		0.7012	
Autos	0.9221	0.9119	*-w*	0.9175		0.9244		0.9424	*ll*	0.9438	*-l*
Balance-s	0.9062	0.8307	*ww*	0.9606	*ll*	0.8307	*ww*	0.7943	*ww*	0.8307	*ww*
Breast-w	0.9939	0.9935		0.9807	*ww*	0.994		0.9883	*ww*	0.9935	
Colic	0.886	0.8394	*-w*	0.8141	*ww*	0.8508	*-w*	0.8488	*-w*	0.8814	
Credit-a	0.912	0.9197	*-l*	0.8801	*ww*	0.8693	*ww*	0.9163		0.9114	
Diabetes	0.8296	0.8311		0.785	*ww*	0.8296		0.8201		0.8187	
Glass	0.8246	0.7813	*ww*	0.7305	*ww*	0.8037		0.772	*ww*	0.8105	
Heart-s	0.9033	0.9108		0.8499	*ww*	0.8847		0.8753	*ww*	0.8917	
Hepatitis	0.8906	0.9056		0.8424	*-w*	0.8496	*-w*	0.8944		0.8423	*-w*
Hypothyroid	0.8994	0.8802	*ww*	0.864	*ww*	0.8497	*ww*	0.875	*ww*	0.8751	*ww*
Ionosphere	0.9542	0.9377		0.9485		0.927	*-w*	0.9819	*-l*	0.95	
Iris	0.9887	0.99		0.9777	*-w*	0.994		0.9893		0.9853	
kr-vs-kp	0.9829	0.9521	*ww*	0.9881	*ll*	0.9647	*ww*	0.9806	*-w*	0.9954	*ll*
Labor	0.9917	0.9917		0.9854		0.8312	*ww*	0.9312		0.9562	
Letter	0.9821	0.9691	*ww*	0.8897	*ww*	0.9706	*ww*	0.9913	*ll*	0.9854	*ll*
Lymph	0.9036	0.9002	*-w*	0.8983		0.8734	*ww*	0.8915		0.8943	
Mushroom	0.9998	0.9979	*ww*	1	*ll*	0.9998		1	*ll*	1	*ll*
P-tumor	0.7561	0.7544		0.6895	*ww*	0.7495		0.7505		0.7491	*-w*
Segment	0.9908	0.9835	*ww*	0.9834	*ww*	0.9889	*-w*	0.9964	*ll*	0.9915	
Sick	0.9809	0.9593	*ww*	0.9565	*ww*	0.9534	*ww*	0.9832		0.937	*ww*
Sonar	0.8493	0.8579		0.829		0.7393	*ww*	0.8375		0.807	
Soybean	0.9986	0.9983	*-w*	0.9906	*-w*	0.9977		0.9991	*-l*	0.9963	*-w*
Splice	0.9934	0.9944	*-l*	0.9863	*ww*	0.5095	*ww*	0.5095	*ww*	0.9944	*-l*
Vehicle	0.8469	0.8077	*ww*	0.794	*ww*	0.8207	*ww*	0.9124	*ll*	0.8598	
Vote	0.9893	0.9729	*ww*	0.973	*-w*	0.9591	*ww*	0.9881		0.9887	
Vowel	0.9746	0.9591	*ww*	0.9522	*ww*	0.9641	*ww*	0.9971	*ll*	0.9865	*ll*
Waveform	0.9574	0.9529	*ww*	0.8954	*ww*	0.9518	*ww*	0.9404	*ww*	0.9332	*ww*
Zoo	0.894	0.894		0.8857		0.8446	*ww*	0.8714	*-w*	0.894	
Average	**0.9222**	0.9113		0.8959		0.8792		0.8982		0.9122	

Table 8.14. CCC scales up NB and NB-like classifiers.

Datasets	CCC	NB	AODE	AODEsr	HNB	WAODE
Anneal	0.9653	0.9601 *-w*	0.961 *-w*	0.9651	0.9641	0.9648
Audiology	0.7069	0.7012 *ww*	0.7015 *ww*	0.7069	0.7044 *-w*	0.7071
Autos	0.9436	0.9119 *ww*	0.9349	0.9419	0.9451	0.9425 *-w*
Balance-s	0.8757	0.8307 *-w*	0.798 *ww*	0.7073 *ww*	0.8808	0.7073 *ww*
Breast-w	0.9907	0.9935	0.9941 *-l*	0.9927 *-l*	0.9905	0.9925
Colic	0.8557	0.8394	0.86	0.8594	0.8673	0.8596
Credit-a	0.912	0.9197	0.9244 *-l*	0.9166 *-l*	0.9096	0.9181 *-l*
Diabetes	0.8226	0.8311	0.8335 *-l*	0.8389 *ll*	0.8317 *ll*	0.8345 *-l*
Glass	0.8324	0.7813 *ww*	0.7867 *ww*	0.8502	0.8313	0.851
Heart-s	0.8839	0.9108 *ll*	0.9108 *ll*	0.9006 *-l*	0.8861	0.8947
Hepatitis	0.8869	0.9056	0.9063	0.8938	0.8961	0.8856
Hypothyroid	0.888	0.8802 *-w*	0.8733 *-w*	0.8916	0.8864	0.8925 *-l*
Ionosphere	0.9826	0.9377 *ww*	0.9749 *-w*	0.9831	0.9844	0.9834
Iris	0.9887	0.99	0.99	0.992	0.9867	0.9933
kr-vs-kp	0.9957	0.9521 *ww*	0.9744 *ww*	0.9805 *ww*	0.9821 *ww*	0.9859 *ww*
Labor	0.9646	0.9917	0.9917	0.9771	0.9583	0.9771
Letter	0.997	0.9691 *ww*	0.9941 *ww*	0.9958 *ww*	0.9948 *ww*	0.9963 *ww*
Lymph	0.8957	0.9002	0.8981	0.891	0.8953	0.8925
Mushroom	1	0.9979 *ww*	1	1	1	1
P-tumor	0.755	0.7544	0.7547	0.758	0.7557	0.7582
Segment	0.9975	0.9835 *ww*	0.9943 *ww*	0.9971 *-w*	0.9971	0.997 *-w*
Sick	0.9917	0.9593 *ww*	0.9719 *ww*	0.9843 *ww*	0.9836 *ww*	0.9894 *ww*
Sonar	0.9067	0.8579 *ww*	0.9041	0.8916	0.9142	0.8943
Soybean	0.999	0.9983 *ww*	0.9986 *ww*	0.9989	0.999	0.9988 *-w*
Splice	0.9958	0.9944 *ww*	0.9946 *-w*	0.9958	0.9654 *ww*	0.9956
Vehicle	0.9037	0.8077 *ww*	0.9013	0.8979 *-w*	0.9078	0.8991
Vote	0.9918	0.9729 *ww*	0.9873 *-w*	0.9872 *-w*	0.9884 *-w*	0.9886 *-w*
Vowel	0.9972	0.9591 *ww*	0.994 *ww*	0.9971	0.9974	0.9963 *ww*
Waveform	0.9615	0.9529 *ww*	0.9647 *ll*	0.9618 *-l*	0.9625	0.9614
Zoo	0.894	0.894	0.894	0.894	0.894	0.894
Average	**0.9261**	0.9113	0.9222	0.9216	0.9253	0.9217

Table 8.15. The comparison between CCC and Bagging for scaling up NB and NB-like classifiers.

Datasets	CCC	BgNB	BgAODE	BgAODEsr	BgHNB	BgWAODE
Anneal	0.9653	0.96 *-w*	0.9612 *-w*	0.964 *-w*	0.9638 *-w*	0.9646 *-w*
Audiology	0.7069	0.703 *ww*	0.7023 *ww*	0.7087	0.7028 *-w*	0.7067
Autos	0.9436	0.9132 *ww*	0.9337 *-w*	0.9462	0.9466	0.9455
Balance-s	0.8757	0.8187 *ww*	0.7898 *ww*	0.7037 *ww*	0.8747	0.7021 *ww*
Breast-w	0.9907	0.9935	0.9938	0.9931	0.9905	0.9931 *-l*
Colic	0.8557	0.8442	0.8628	0.8581	0.8666	0.8576
Credit-a	0.912	0.9197	0.9243 *-l*	0.9184	0	0 *-l*
Diabetes	0.8226	0.8312	0.8317	0.8353 *-l*	0.832 *ll*	0.8355 *ll*
Glass	0.8324	0.7783 *ww*	0.7944 *-w*	0.8444	0.8235	0.8385
Heart-s	0.8839	0.9083 *ll*	0.9103 *ll*	0.9028 *ll*	0.89	0.8986 *-l*
Hepatitis	0.8869	0.9053	0.9057	0.9003	0.8903	0.8998
Hypothyroid	0.888	0.8789 *-w*	0.872 *-w*	0.8901	0.8868	0.8927
Ionosphere	0.9826	0.9381 *ww*	0.9738 *-w*	0.9783 *-w*	0.9809	0.9797
Iris	0.9887	0.9887	0.988	0.9913	0.986	0.9907
kr-vs-kp	0.9957	0.9523 *ww*	0.9748 *ww*	0.9802 *ww*	0.9822 *ww*	0.9856 *ww*
Labor	0.9646	0.9917	0.9854	0.9708	0.9583	0.9771
Letter	0.997	0.9694 *ww*	0.9942 *ww*	0.996 *ww*	0.9948 *ww*	0.9964 *ww*
Lymph	0.8957	0.9004	0.8984	0.8924	0.8969	0.8913 *-w*
Mushroom	1	0.9979 *ww*	1	1	1	1
P-tumor	0.755	0.7532	0.7551	0.7562	0.7546	0.7554
Segment	0.9975	0.9838 *ww*	0.9943 *ww*	0.997 *ww*	0.9971	0.997 *-w*
Sick	0.9917	0.9596 *ww*	0.9723 *ww*	0.9852 *ww*	0.9842 *ww*	0.9893 *ww*
Sonar	0.9067	0.8602 *-w*	0.8982	0.873 *-w*	0.9132	0.8842
Soybean	0.999	0.9982 *ww*	0.9986 *ww*	0.9989	0.999	0.9988 *-w*
Splice	--					
Vehicle	0.9037	0.8088 *ww*	0.8981	0.8969 *-w*	0.9048	0.8975
Vote	0.9918	0.973 *ww*	0.9876 *-w*	0.9876 *-w*	0.9882 *-w*	0.9886 *-w*
Vowel	0.9972	0.9601 *ww*	0.9934 *ww*	0.9962 *ww*	0.9964 *-w*	0.9959 *ww*
Waveform	0.9615	0.9531 *ww*	0.965 *ll*	0.9628 *ll*	0.9625	0.9627 *ll*
Zoo	0.894	0.8893	0.894	0.894	0.894	0.894
Average	**0.9237**	0.9080	0.9191	0.9180	0.8917	0.8869

Table 8.16. Comparison between CCC and AdaBoost for scaling NB and NB-like classifiers.

Datasets	CCC	BNB	BAODE	BAODEsr	BHNB	BWAODE
Anneal	0.9653	0.9379	0.9239 *-w*	0.9205 *-w*	0.941 4	0.941
Audiology	0.7069	0.6921 *ww*	0.7015 *ww*	0.6913 *ww*	0.69 *ww*	0.6954 *ww*
Autos	0.9436	0.9175 *-w*	0.8456 *ww*	0.9341	0.9326	0.9455
Balance-s	0.8757	0.96 *ll*	0.7821 *ww*	0.7772 *ww*	0.9063 *-l*	0.7857 *ww*
Breast-w	0.9907	0.9807 *-w*	0.9642 *ww*	0.984	0.977 *ww*	0.9769 *ww*
Colic	0.8557	0.8141 *-w*	0.8077 *-w*	0.839	0.8207 *ww*	0.8318 *-w*
Credit-a	0.912	0.8801 *ww*	0.8359 *ww*	0.865 *ww*	0.8618 *ww*	0.8655 *ww*
Diabetes	0.8226	0.785 *ww*	0.7293 *ww*	0.7596 *ww*	0.7566 *ww*	0.7751 *ww*
Glass	0.8324	0.7264 *ww*	0.7819 *ww*	0.7905 *-w*	0.779 *ww*	0.7923 *-w*
Heart-s	0.8839	0.8499 *-w*	0.8069 *ww*	0.8462 *-w*	0.831 1 *ww*	0.8504 *-w*
Hepatitis	0.8869	0.8424	0.7595 *ww*	0.8301 *-w*	0.797 6 *-w*	0.8281 *-w*
Hypothyroid	0.888	0.8645 *ww*	0.8733 *-w*	0.8714 *-w*	0.869 9 *-w*	0.8747 *-w*
Ionosphere	0.9826	0.9485 *ww*	0.9128 *ww*	0.9799	0.9691 *ww*	0.9592 *-w*
Iris	0.9887	0.9777 *-w*	0.972 *-w*	0.9713 *-w*	0.972 *ww*	0.978 *-w*
kr-vs-kp	0.9957	0.9881 *ww*	0.9375 *ww*	0.9974 *-l*	0.9982 *ll*	0.998 *ll*
Labor	0.9646	0.9854	0.9542	0.9771	0.9656	0.9615
Letter	0.997	0.8897 *ww*	0.9586 *ww*	0.9862 *ww*	0.9874 *ww*	0.9891 *ww*
Lymph	0.8957	0.8983	0.8317 *-w*	0.8906	0.8593 *-w*	0.8945
Mushroom	1	1	1	1	1	1
P-tumor	0.755	0.6906 *-w*	0.7547	0.6595 *ww*	0.699 *ww*	0.6857 *ww*
Segment	0.9975	0.9831 *ww*	0.9729 *ww*	0.9948 *ww*	0.9944 *ww*	0.9946 *ww*
Sick	0.9917	0.9565 *ww*	0.9083 *ww*	0.9751 *ww*	0.9821 *-w*	0.9854 *-w*
Sonar	0.9067	0.829 *ww*	0.9041	0.8916	0.8776 *-w*	0.794 *ww*
Soybean	0.999	0.9906 *ww*	0.9779 *ww*	0.9898 *-w*	0.995 *-w*	0.9936 *-w*
Splice	--					
Vehicle	0.9037	0.7858 *ww*	0.7591 *ww*	0.845 *ww*	0.8566 *ww*	0.8595 *ww*
Vote	0.9918	0.973 *ww*	0.9466 *ww*	0.9806 *ww*	0.9752 *ww*	0.9762 *ww*
Vowel	0.9972	0.9522 *ww*	0.9687 *ww*	0.9937 *-w*	0.993 6	0.9942 *ww*
Waveform	0.9615	0.895 *ww*	0.8877 *ww*	0.9282 *ww*	0.9339 *ww*	0.9326 *ww*
Zoo	0.894	0.8857	0.8857	0.894	0.844 *ww*	0.8464 *ww*
Average	**0.9237**	0.8924	0.8739	0.8987	0.8989	0.8967

Table 8.17. Comparison between CCC and MultiBoostAB for scaling up NB and NB-like classifiers.

Datasets	CCC	MNB	MAODE	MAODEsr	MHNB	MWAODE
Anneal	0.9653	0.9045 *ww*	0.9333 *-w*	0.8751 *ww*	0.9083 *ww*	0.9074 *ww*
Audiology	0.7069	0.6606 *ww*	0.7015 *ww*	0.6737 *ww*	0.6835 *ww*	0.6824 *ww*
Autos	0.9436	0.9039 *-w*	0.8714 *ww*	0.8972 *-w*	0.9319	0.9334
Balance-s	0.8757	0.9399 *ll*	0.7819 *ww*	0.8113 *ww*	0.9049 *ll*	0.8475 *-w*
Breast-w	0.9907	0.9876	0.9709 *-w*	0.984	0.9787 *ww*	0.9779 *-w*
Colic	0.8557	0.834	0.8373	0.8359 *-w*	0.8479	0.8447
Credit-a	0.912	0.8956 *-w*	0.8402 *ww*	0.877 *ww*	0.8789 *ww*	0.8905 *ww*
Diabetes	0.8226	0.8035 *-w*	0.8024	0.7971 *ww*	0.7919 *ww*	0.8004 *-w*
Glass	0.8324	0.7394 *ww*	0.7817 *ww*	0.7974 *-w*	0.7699 *ww*	0.7707 *ww*
Heart-s	0.8839	0.8639	0.8211 *ww*	0.8717	0.8558 *ww*	0.8749
Hepatitis	0.8869	0.8462	0.796 *-w*	0.8572	0.8533 *-w*	0.8593
Hypothyroid	0.888	0.8636 *ww*	0.8733 *-w*	0.8659 *ww*	0.8626 *ww*	0.866 *ww*
Ionosphere	0.9826	0.948 *ww*	0.925 *ww*	0.9762	0.9613 *ww*	0.9524 *ww*
Iris	0.9887	0.983	0.9753 *-w*	0.9857	0.9783 *-w*	0.9887
kr-vs-kp	0.9957	0.968 *ww*	0.9866 *ww*	0.9905 *ww*	0.9899 *ww*	0.9932 *ww*
Labor	0.9646	0.9917	0.9542	0.9771	0.9354	0.9406
Letter	0.997	0.9071 *ww*	0.9863 *ww*	0.988 *ww*	0.9876 *ww*	0.9898 *ww*
Lymph	0.8957	0.8977	0.8468 *-w*	0.8751	0.8608 *-w*	0.892
Mushroom	1	0.9996 *ww*	1	1	1	1
P-tumor	0.755	0.6938 *-w*	0.7547	0.6866 *ww*	0.7233 *ww*	0.7074 *ww*
Segment	0.9975	0.9834 *ww*	0.9858 *ww*	0.9953 *-w*	0.9954 *ww*	0.9955 *ww*
Sick	0.9917	0.9608 *ww*	0.9123 *ww*	0.9742 *ww*	0.9786 *ww*	0.982 *-w*
Sonar	0.9067	0.8382 *ww*	0.9041	0.8916	0.8975	0.8385 *ww*
Soybean	0.999	0.9905 *-w*	0.978 *ww*	0.9885 *-w*	0.9913 *-w*	0.9923 *-w*
Splice	--					
Vehicle	0.9037	0.7926 *ww*	0.7733 *ww*	0.8788 *ww*	0.8839 *ww*	0.882 *ww*
Vote	0.9918	0.9598 *ww*	0.9498 *ww*	0.9798 *-w*	0.9756 *ww*	0.9825 *-w*
Vowel	0.9972	0.9393 *ww*	0.9791 *ww*	0.9918 *-w*	0.9914 *ww*	0.9937 *ww*
Waveform	0.9615	0.9007 *ww*	0.9331 *ww*	0.9332 *ww*	0.9391 *ww*	0.9396 *ww*
Zoo	0.894	0.8821	0.8917	0.894	0.831 *ww*	0.831 *ww*
Average	**0.9237**	0.8924	0.8878	0.9017	0.9030	0.9019

Secondly, we show that CCC can successfully scale up NB and NB-like classifiers, as shown in Table 8.14, and CCC is more successful than other Meta Learning techniques such as Bagging, AdaBoost, and MultiBoostAB for scaling up NB and NB-like classifiers, as shown in Table 8.15, 8.16, and 8.17, respectively.

For example, CCC built on kr-vs-kp ensembles a couple of CC classifiers consisting of uniform NB and NB-like classifiers by assuming dynamic selection strategy, and significantly enhances individual NB and NB-like classifiers, as shown in Table 8.14. Finally, as we can see, CCC usually significantly enhances NB and NB-like classifiers in most cases. In general, it is difficult to boost Bayes classifiers because they are believe to be much stable. This result shows that CCC can succeed in reducing the bias of these Bayes classifiers.

The experimental results for comparing CCC with Bagging for scaling up NB, AODE, AODEsr, HNB, and WAODE, thus denoted as BgNB, BgAODE, BgAODEsr, BgHNB, and BgWAODE, respectively, are shown in Table 8.15; the experimental results for comparing CCC with AdaBoost for scaling up NB, NB, AODE, AODEsr, HNB, and WAODE, thus denoted as BNB, BAODE, BAODEsr, BHNB, and BWAODE, respectively, are shown in Table 8.16; The experimental results for comparing CCC with MultiBoostAB for scaling up NB, AODE, AODEsr, HNB, and WAODE, denoted as MNB, MAODE, MAODEsr, MHNB, and MWAODE, respectively, are shown in Table 8.17.

In these experiments, we observed that NB-like classifiers built on the Splice case require too many parameters to build a Meta learner. We simply ignored their experimental results on Splice. From these results, CCC outperforms those Meta Learning techniques for scaling up NB and NB-like classifiers in most cases.

The averaged AUCs are displayed at the bottoms of Table 8.13, 8.14, 8.15, 8.16, and 8.17. The summary of statistical tests between CCCNB, CCC and all other approaches are reported in Table 8.18. As we can see, on average, CCC outperforms all other classifiers to successfully scale up NB and NB-like classifiers.

Table 8.18. Summary of statistical tests (win/draw/lose) between CCCNB, CCC and other approaches.

	Statistical tests	NB	BNB	SBC	TAN	NBTree
CCCNB	Paired-t	12\18\0	15\12\3	14\16\0	7\17\6	4\22\4
	Wilcoxon	17\11\2	20\7\3	20\10\0	10\11\9	7\17\6
CCC		NB	AODE	AODEsr	HNB	WAODE
	Paired-t	16\13\1	9\19\2	4\25\1	4\25\1	5\25\0
	Wilcoxon	19\10\1	14\11\5	7\18\5	6\23\1	9\18\3
		BgNB	**BgAODE**	**BgAODEsr**	**BgHNB**	**BgWAODE**
	Paired-t	15\13\1	8\19\2	6\21\2	3\25\1	5\22\2
	Wilcoxon	18\10\1	14\12\3	11\15\3	7\21\1	10\14\5
		BNB	**BAODE**	**BAODEsr**	**BHNB**	**BWAODE**
	Paired-t	16\13\1	19\11\0	11\19\0	16\13\1	14\14\1
	Wilcoxon	22\7\1	24\6\0	19\10\1	22\6\2	23\5\1
		MNB	**MAODE**	**MAODEsr**	**MHNB**	**MWAODE**
	Paired-t	15\13\1	16\13\0	12\17\0	19\9\1	15\14\0
	Wilcoxon	20\8\1	22\7\0	19\10\0	23\5\1	21\8\0

8.4.4.1 Summary of CCC Results

We conducted experiments on the first 30 benchmark datasets (the results for large datasets are omitted without any loss of generality) in Table 8.1 to show how CCC successfully scales up NB and those NB-like classifiers as compared with previously proposed Meta Learning techniques, i.e, Bagging, AdaBoost, and MultiBoostAB. All related traditional classifiers from Weka have been modified according to our discussion in Section 8.4.4 such that they can be used in CCC.

CCCNB, which is a CCC using the uniform NB as a base learner, outperforms SBC, which performs feature selection for NB, to scale up NB. This also verifies the previous research result that feature selection in SBC is not the best way to scale up NB [19].

CCCNB is also competitive with TAN and NBTree, which learn a simple Bayesian structure and a tree structure, respectively, to scale up NB in some cases. However, CCC does not intend to beat TAN, NBTree, and those NB-like classifiers.

Instead, CCC can successfully enhance NB and NB-like classifiers in most cases by building a couple of CC classifiers consisting of uniform NB and NB-like classifiers. Further, CCC outperforms previous proposed Meta Learning techniques to scale up NB and NB-like classifiers while these previous techniques are subject to a failure scaling up NB and NB-like classifiers.

8.4.4.2 Remarks about CCC Results

Although we compared CCC with other approaches such as BNB for scaling up NB and those NB-like classifiers, CCC does not intend to beat previous approaches for scaling up NB, but intends to enhance those Bayes classifiers. Even CCC is not a classifier but a probability learning framework.

Because AdaBoost with NB is also effective to scale up NB in some cases such as Balance-s in Table 8.13, it suggests that CCC can further improved by incorporating it with boosting techniques such as AdaBoost. In addition, although our experiments were conducted on discrete cases, CCC is not limited to discrete domains for scaling up individual classifiers.

Chapter 9

Conclusion and Future Work

In this research, we consider two important issues: sampling selection and the Class Imbalance Problem (CIP) in supervised learning. Correspondingly, we attempt to answer two questions:

- Can we find an effective sample from a given training set for training a successful classifier?
- Can we develop a novel sample selection technique to tackle the CIP?

To answer the first question, we developed a new sample selection technique, which is called Border Sampling (BS); to answer the second question, we developed a novel Meta Learning technique, which is called Cascading Customized Couple (CCC).

We concluded our research results in this thesis by summarizing our main work as follows.

9.1 BI_2 and PBS

To answer the first question, we re-investigated the traditional Border Identification (BI) technique. As a result, we observed that traditional BI [22][27] suffers from a limitation to identify partial borders, and thus developed a new method, called Border Identification in Two Stages (BI_2), to identify a full border consisting of near borders and far borders.

However, borders identified by BI_2 have a high uncertainty for discrimination, and it is insufficient to be used for training a successful

classifier according to Bayesian Learning theory [23] because traditional induction algorithms build classifiers within this learning framework.

As a result, we developed a new algorithm, called Progressive Border Sampling (PBS), to progressively augment borders for training any classifier by borrowing the main ideas behind the previously proposed Progressive Sampling (PS) technique. PBS produces more effective samples than the traditional BI for training classifiers. Our experimental results on 30 benchmark datasets from the UCIKDD repository show that the PBS helps build classifiers similar to those built on full training sets in most cases, and PBS overwhelmingly outperforms the traditional BI technique for training set reduction.

9.2 Border Sampling

After we achieved an initial success in developing a new learner-independent sample selection technique, we summarized this technique by naming it Border Sampling (BS) technique. Both BI_2 and PBS are two preliminary algorithms in BS.

Further, we have the following five questions corresponding to the initial five questions, see Section 1.1, related to BS:

- How is BS used in multi-class domains?
- How can we formally define border and redundancy in a labelled training set so as to establish a theoretical foundation of Border Sampling?
- How can PBS be scaled up to large datasets?
- How can we assure that BS converges to an effective sample from subsamples resampled from a large population ?
- Can BS be used as a wrapped sample selection for the CIP ?

9.2.1 Multiclass Domains

For the first question, we investigated traditional class binaritization methods, i.e., one-against-one and one-against-all, which are originally used for classification. As a result, similarly, we can also assume the one-against-one (oo) and the one-against-all (oa) for Border Sampling on multi-class domains. For example, PBS with oo strategy, denoted as PBS-oo, produces augmented borders on each pair of classes on a multi-class domain and builds a Naïve Bayes on the resulting borders. The learning curve of the Naïve Bayes is used for convergence detection. In total, this leads to a pairwise Naïve Bayes built on the multi-class domain. Previous theoretical result shows that this pairwise Naïve Bayes is equivalent to a standard Naïve Bayes built on the whole training set.

Although both PBS-oo and PBS-oa have the same time complexity, the theoretical foundation suggests that PBS-oo is preferable to PBS-oa because the pairwise border sampling is effective while PBS-oo suffers from ineffectiveness due to the probability estimation for training a Naïve Bayes with oa.

9.2.2 Theoretical Foundation

For the second question, we first formally define border and redundant points by avoiding the limitation of the traditional Border Identification, which only identifies partial borders. The formal definitions of border and redundant points are recursively described such that they actually correspond to an algorithm, which has a quadratic time complexity.

According to our definitions of border and redundant points, we show that BI_2 identifies all border points and remove all redundant points. Further, redundant points are located at the centers of classes, and surrounded by borders. We also show that all redundant points are close to those borders

belonging to the same class and far from those borders belonging to different classes.

9.2.3 CPBS

For the third question, we proposed a new algorithm, called Coupling Markov Chain Monte Carlo (CMCMC) for scaling up Progressive Border Sampling (PBS), denoted as CPBS, on large datasets. CPBS borrows the main idea behind Coupling From The Past (CFTP), by building coupled Markov Chains for scaling up PBS. As a result, it can produce an effective sample by learning subsamples resampled from a large population with a specified sampling window in a linear time complexity with respect to the number of instances in the population.

Our experiments on 33 datasets show that CPBS outperforms the previous three learner-independent methods, Static, Arith, and Geo for sample selection. It is even better than training on the Full dataset for NB and SVM, and it is very competitive with training on the Full dataset for training Decision Trees (DT) [74] and IB1 [1], and no evident result show a negative impact on the size of the sampling window on the resulting samples for training classifiers.

9.2.4 TPBS

For the fourth question, we designed a novel effective geometric computation which describes the propagation of the nearest neighbours in BS. A new algorithm, called Tripled Markov Chain Monte Carlo for scaling up Progressive Border Sampling, thus denoted as TPBS, is proposed by incorporating this effective geometric computation with CPBS for enhancing the algorithm convergence.

As a result, TPBS helps learn better NB in most cases while it does not degrade the performance of DT, SVM, and 1NN as compared with Full on

whole training sets. Further, TPBS as an alternative method of CPBS exhibits more effectiveness and stability than CPBS for instance selection with respect to different sampling window sizes. TPBS is very scalable on large datasets similar to CPBS due to its approximate linear time complexity, and can be more efficient than CPBS in a few cases.

On average, TPBS produces slightly larger samples than CPBS, and produces smaller samples than CNN, ENN, and RENN [96] except for the DROP3.1. Even if TPBS produces a sample with a size approximately equal to that obtained by other instance selection approaches, the resulting samples are quite different. TPBS outperforms previous approaches in producing effective samples for training several traditional classifiers while previous instance selection techniques have a bias towards Instance Based Learning (IBL) [1], e.g., IB1 and k-Nearest Neighbours (kNN).

The fundamental elements in Border Sampling techniques include as follows:

- Border Identification in Two Stages (BI_2), which identifies a full border consisting of near and far borders by avoiding the limitation of traditional Border Identification techniques such as Duch 2, which is discussed in Section 2.1.2.
- A proper similarity distance metric, which can properly combine continuous, nominal, and missing attribute values in the metric, as discussed in Section 3.5. Our current research only adopts Cosine or RBF by assuming variable independences. A proper similarity distance metric helps identify effective samples fast because a different metric defines a different border in a feature space;
- Class binaritization methods for BS on multi-class domains. Both one-against-one (oo) and one-against-all (oa) have the same time complexity for Border Sampling although the oo is preferable to the

oa according to the previous theoretical analysis about pairwise Naïve Bayes.

The advanced elements in Border Sampling techniques also include:

- Progressive Border Sampling (PBS), which augments borders for an effective sample;
- Coupling Markov Chain Monte Carlo, which is used for scaling up PBS on large datasets, thus denoted as CPBS, and its alternative method, which is called Tripled Markov Chain Monte Carlo, thus denoted as TPBS, for enhancing the algorithmic convergence;
- Cascading Customized Couple (CCC), which assumes a novel wrapped sample selection method for scaling up individual classifiers. Therefore, Border Sampling techniques are divided into two categories: BS on class boundary by using BI_2 and BS on decision boundary by defining the separation in CCC;

9.3 Cascading Customized Couple

For the fifth question, Border Sampling based on BI_2 is suggested to be used as a filter sample selection in this thesis although theoretically it can be used as a wrapped sample selection method. To address the Class Imbalance Problem (CIP), we investigated previous approaches, which assume ensemble learning techniques with basic sampling techniques e.g., under-sampling and over-sampling, for tackling the CIP. The main problem is that previous approaches for the CIP only work on binary domains. Our analysis suggests that the capability of classifiers is crucial for tackling the CIP.

As we can see, there are two main goals towards the enhancement of individual classifiers:

Firstly, we are interested in a new wrapped sample selection method, which is used in induction algorithms, for training a successful classifier;

Secondly, we would develop a novel Meta learning technique to tackle the CIP by enhancing individual classifiers. Previously proposed ensemble learning techniques with basic sampling methods for the CIP suffer from loss of information, and suffer from a failure to scale up NB and NB-like classifiers.

Finally, a novel Meta learning technique, Cascading Customized Couple (CCC), is proposed to improve individual classifiers by building a couple of a new kind of classifier, called Customized Classifiers (CC). Not only can CCC maintain or significantly improve NB, but CCC can also maintain or improve those NB-like classifiers in most cases according to our experiments conducted on benchmark datasets from the UCIKDD repository. Arguably, it is entirely different from AdaBoost. CCC is more successful than AdaBoost for improving NB and NB-like classifiers.

Further, CCC is suggested to build a successful ensemble learner by a dynamic selection when it works with different base learners. Both CCC and AdaBoost techniques are two important learning techniques. Even CCC can accept AdaBoost as a base learning in many learning tasks.

9.4 Contributions

Our main contributions in this thesis consist of the following two aspects. Firstly, we proposed a new sample selection technique, called Border Sampling (BS) on class boundary, which consists of BI_2, PBS, CPBS, and TPBS algorithms, for supervised learning tasks. Secondly, we proposed a new Meta learning technique, called Cascading Customized Couple (CCC), by building a couple of a new kind of classifiers, called Customized Classifiers (CC), by assuming a novel wrapped sample selection method, called Border Sampling on decision boundary, for improving individual classifiers.

BI_2 identifies a full border consisting of near borders and far borders, which are more informative than the border obtained by traditional BI [22][27]. PBS can produce an effective sample from the original training set by borrowing the main idea behind a previously proposed Progressive Sampling (PS) [49][72][93] technique to augment the borders. Both CPBS and TPBS can be used for scaling up PBS on large datasets by incorporating Coupling Markov Chain Monte Carlo (CMCMC) with PBS in CPBS and incorporating an effective geometric computation into Tripled Markov Chain Monte Carlo in TPBS.

We define a Border Sampling framework consisting of two categories: BS on class boundary by using BI_2 and BS on decision boundary by defining the separation in CCC. The former is suggested to be used as a filter sample selection method in this thesis although theoretically it can also be used as a wrapped sample selection. The latter is a new wrapped sample selection method used in CCC.

CCC is a different solution for improving individual classifiers as compared with AdaBoost, and is regarded as a crucial technique toward tackling the Class Imbalance Problem (CIP). It is entirely different from previous Meta learning techniques, and exhibits a potential high performance over previous Meta learning techniques for scaling up Naïve Bayes and Naïve Bayes-like classifiers.

9.5 Future Work

It is believed that BS can be applied to reducing learning cost in active learning [15][87] and semi-supervised learning [21] because it enjoys a lot of advantages such as the learner-independence and linear time complexity.

Before BS can be applied to other learning tasks, there are demands to answer whether TPBS can be enhanced to efficiently produce a smaller

sample than CPBS by using a proper similarity distance metric, and whether TPBS can be used for incremental sampling. One issue that needs to be resolved in this context is the fact that the proposed Border Sampling techniques still suffer, from time to time, from a little weakness in algorithmic convergence.

The choice of an appropriate similarity distance metric is another crucial issue for BS. For the time being, we do not know how a different similarity metric makes an impact on BS. These issues will have to be clarified in future.

Although BS has been discussed as a filter sample selection method in this thesis, we also wonder how BS can be used as a wrapped sample selection for training a successful classifier. Because Support Vector Machine can be only built with support vectors in training sets according to the dual method [89], this suggests a possible application of Border Sampling on class boundary for training SVM as a wrapped sample selection method.

CCC and AdaBoost are two important techniques to scale up individual classifiers. CCC is regarded as a probability learning schema for scaling up any classifier, and thus also accepts a Meta learner such as AdaBoost as a base learner when it has been observed that AdaBoost is effective in some cases. How to combine CCC and AdaBoost becomes our future work.

Bibliography

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. *Proc. of 15th Intl. Conf. on Machine Learning*, 1-10, 1998.

[2] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37-66, 1991.

[3] C. Andrieu, N. D. Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50: 5–43, 2003.

[4] F. Aurenhemmer. Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure. *ACM Computing survey*, 23(3): 345-405, 1991.

[5] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explorations*, 6(1):20-29, 2004.

[6] E. Baur and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105-139, 1999.

[7] C. Blake and C. Merz. UCI repository of Machine Learning Databases. http://www.ics.uci.edu/~mlearn/~MLRepository.html. Department of Information and Computer Science, University of California, Irvine.

[8] L. Breiman. Bagging Predictors. *Machine Learning*, 24(3):123-140, 1996.

[9] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining Knowledge Discovery*, 6(2):153–172, 2002.

[10] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of Knowledge Discovery and Data Mining*, 164-168, 1998.

[11] C.L. Chang. Finding prototypes for nearest neighbour classifiers. *IEEE Transaction on Computers*, 23(11): 1179-1184, 1974.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[13] N. V. Chawla, N. Japkowicz, and A. Kolcz. Editorial to the special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations*. 6(1):1-6, 2004.

[14] C. Chen, A. Liaw, L. Breiman. *Using Random Forest to Learn Imbalanced Data*. Technical Report. Berkeley, Department of Statistics, University of California, 2004.

[15] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129-145, 1996.

[16] S. Cost and S. Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10(1):57–78, 1993.

[17] R. Debnath, N. Takahide, and H. Takahashi. A decision based one-against-one method for multi-class support vector machine. *Pattern Analysis and Applications*, 7(2): 164-175. July, 2004.

[18] P. Domingos. MetaCost: a general method for making classifiers cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 155–164, 1999.

[19] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the sample Bayesian classifier. *Machine Learning* 29: 103-130, 1997.

[20] C. Drummond and R. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats oversampling. In *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets*, 2003.

[21] G. Druck, C. Pal, X. Zhu, and A. McCallum. Semi-supervised Classification with Hybrid generative/Discriminative Methods. In *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, 280-289, 2007.

[22] W. Duch. Similarity based methods: a general framework for classification, approximation and association. *Control and Cybernetics*, 29 (4):937-968, 2000.

[23] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* A Wiley Intersience Publication (2000).

[24] C. Elkan. *Boosting and Naïve Bayesian Learning.* Technical Report CS97-557, University of California, Davis, 1997.

[25] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, 2: 524–532, 1990.

[26] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. http://www.hpl.hp.com/-personal/TomFawcett/papers/ index.html, 2003.

[27] G. M. Foody. Issues in Training Set Selection and Refinement for Classification by a Feedforward Neural Network. In *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, 1:409-411, 1998.

[28] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica,* 2(2):153-174, 1987.

[29] Y. Freund and R. E. Schapire. A short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, 1999.

[30] J. H. Friedman. *Another approach to polychotomous classification.* Technical report, Stanford Univ, 1996.

[31] N. Friedman, D. Geiger, and M. Goldszmith. Bayesian network classifiers. *Machine Learning* 29: 131-163, 1997.

[32] J. Gama, P. Brazdil. Cascade generalization. *Machine Learning* 41: 315-343, 2000.

[33] G. Geng, C. Wang, Q. Li, L. Xu, and X. Jin. Boosting the performance of Web Spam detection with ensemble under-sampling classification. *IEEE Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 583-587, 2007.

[34] C. Giraud-Carrier. A Note on the Utility of Incremental Learning. *AI Communications*, 13: 215 – 223, 2000.

[35] P. J. Green and R. Sibson. Computing Dirichlet tessellations in the plane. *The Computert* J, 21:168-173, 1977.

[36] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 146:29-36, 1982.

[37] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14,:515–516, 1968.

[38] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

[39] S. Hettich and S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science. 1999.

[40] C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Trans on Neural Networks*, 13:415-425, 2002.

[41] T. Huang, R. C. Weng, and C. Lin. Generalized Bradley-Terry models and multi-class probability estimations. *Journal Machine Learning Research*, 7:85-115, 2006.

[42] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*-11, 1999.

[43] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive Mixtures of Local Experts. In *Neural Computation*, 3:79-97, 1988.

[44] A. Jain and D. Nikovski. Incremental exemplar learning schemes for classification on embedded devices. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases*, 11, 2008.

[45] N. Japkowicz, C. Myers and M. Gluck. A Novelty Detection Approach to Classification. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* 518-523, 1995.

[46] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):203-231, 2002.

[47] L. Jiang and H. Zhang. Weightily Averaged One-Dependence Estimators. In: *Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence*, 970-974, 2006.

[48] T. Jo and N. Japkowicz. Class imbalances versus small disjuncts. *SIGKDD Explorations*, 6(1):40-49, 2004.

[49] G. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 367-370, 1996.

[50] M. Jordan and R. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation* 6, 181-214, 1994.

[51] S.-W. Kim, B.J. Oommen, Enhancing prototype reduction schemes with LVQ3-type algorithms, *Pattern Recognition*, 36(5):1083–1093, 2003.

[52] S.-W. Kim, B.J. Oommen, Enhancing prototype reduction schemes with recursion: a method applicable for "large" datasets, *IEEE Transactions on Systems, Man, and Cybernetics—Part B SMC*, 34 (3): 1384–1397, 2004.

[53] V. Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv Math*, 34:75-80, 1980.

[54] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International conference on Knowledge Discovery and Data Mining*, 202-207, 1996.

[55] M. Kubat and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *Proc. 14th International Conference on Machine Learning*, 179-186, 1997.

[56] P. Langley. Induction of recursive Bayesian classifiers. In *Proceedings of the 8th European Conference on Machine Learning*, 153-164, 1993.

[57] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, 223-228, 1992.

[58] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, 399-406, 1994.

[59] P. Laskov, C. Gehl, S. Kr¨uger, K. M¨uller. Incremental Support Vector Learning: Analysis, Implementation and Applications. *Journal of Machine Learning Research*, 7:1909– 1936, 2006.

[60] C. L. Lawson. Generation of a triangular grid with applications to contour plotting. *Tech. Memorandum 299*, California Inst. Tech. Jet Propulsion Lab, 1972.

[61] G. Li, N. Japkowicz, T. J. Stocki, and R. K. Ungar. Full Border Identification for Reduction of Training Sets. In *Proceedings of the 21st Canadian Conference in Artificial Intelligence*, 203-215, 2008.

[62] G. Li, N. Japkowicz, T. J. Stocki, and R. K. Ungar. Border sampling through Markov chain Monte Carlo. *International Conference on Data Mining*, 393-402, 2008.

[63] H. Liu and H. Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6:115–130, 2002.

[64] X. Liu, J. Wu, and Z. Zhou. Exploratory Under-Sampling for Class-Imbalance Learning. *International Conference on Data Mining*, 965-969, 2006.

[65] P. Melville and R. J. Mooney. Diverse Ensembles for Active Learning. *Proceedings of the 21th International Conference on Machine Learning*, 584-591, 2004.

[66] T. Mitchell. *Machine Learning*. McGraw-Hill Companies, Inc (1997).

[67] A. Ng and M. Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and Naïve Bayes. In: Dietterich TG, Becker S, Ghahramani Z (eds) *NIPS*, 841-848, 2001.

[68] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods - support vector learning*, 185-208, 1999.

[69] J. C. Platt, N. Christianini, and J. Shawe-Tayor. Large magin DAG's for multiclass classification. In *Advance in Neural Information Processing System*, 12:547-553, 2000.

[70] D. F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer- Verlag, Berlin, 1985.

[71] W. H. Press and G. R. Farrar. Recursive Stratified Sampling for Multidimensional Monte Carlo Integration, *Computers in Physics*, 4:190-195, 1990.

[72] F. Provost, D. Jensen, and T. Oates. Efficient Progressive Sampling. *The 5th International Conference on Knowledge Discovery and Data mining*, 23-32, 1999.

[73] J. G. Propp and D. B. Wilson. Coupling from the past: a user's guide. In D. Aldous and J. G. Propp (Eds.), *Microsurveys in Discrete Probability*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 181-192, 1998.

[74] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

[75] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum. Classification with hybrid generative/discriminative models. In S. Thrun, L. Saul , & B. Sch¨olkopf (Eds.), *Advances in neural information processing systems 16*, 2003.

[76] J. Rennie, L. Shih, J. Teevan, D. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *Proceedings of International Conference on Machine Learning*, 616-623, 2003.

[77] B. Scholkpf, J. Platt, J. Shawe-Taylor, and A. Smola. *Estimating the support of a high-dimension distribution.* Technical report, Microsoft Research, MSR-TR-99-87, 1999.

[78] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual IEEE Sympostum on FOCS*, 151-162, 1975.

[79] T. J. Stocki, X. Blanchard, R. D'Amours, R. K. Ungar, J. P. Fontaine, M. Sohier, M. Bean, T. Taffary, J. Racine, B. L. Tracy, G. Brachet, M. Jean, and D. Meyerhof. Automated radioxenon monitoring for the comprehensive nuclear-test-ban treaty in two distinctive locations: Ottawa and Tahiti. J. *Environ.Radioactivity*, 80:305-326, 2005.

[80] A. Strehl and J. Ghosh. Value-based customer grouping from large retail data-sets. In *Proceedings of SPIE Conference on Data Mining and Knowledge Discovery,* Orlando, 4057:33-42, 2000.

[81] M. I. A. Strens. Evolutionary MCMC sampling and optimization in discrete spaces. In *Proceedings of the 20th International Conference on Machine Learning*, 736-743, 2003.

[82] J. D. Sullivan. The comprehensive test ban treaty. *Physics Today* 151, 1998.

[83] J. Sulzmann, J. Fürnkranz, and E. Hüllermeier. On Pairwise Naive Bayes Classifiers. In *Proceedings of the 18th European Conference on Machine Learning*, 658-665, 2007.

[84] K. Ting and Z. Zheng. A study of Adaboost with naive Bayesian classifiers: w eakness and improvement. *Computational Intelligence*, 19(2):186-200, 2003.

[85] K. Ting and I. Witten. Issues in Stacked Generalization. *Journal of Artificial Intelligence Research 10*, 271-289, 1999.

[86] I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):769–772, 1976.

[87] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[88] D. Tsujinishi, Y. Koshiba, and S. Abe. Why pairwise is better than one-against-all or all-at-once. *Proc. International Joint Conference on Neural Networks* 1:693-698, 2004.

[89] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[90] B. X. Wang and N. Japkowicz. Boosting Support Vector Machines for Imbalanced Data Sets. *Journal of Knowledge and Information Systems*, 4994:38-47, 2008.

[91] G. I. Webb. MultiBoosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159-196, 2000.

[92] G. I. Webb, J. Boughton, and Z. Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5-24, 2005.

[93] G. M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19: 315-354, 2003.

[94] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[95] D. R. Wilson and T. R. Martinez. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, 6:1-34, 1997.

[96] D. R. Wilson and T. R. Martinez. Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, 38:257–286, 2000. Kluwer Academic Publishers. Printed in The Netherlands.

[97] D. Wolpert. Stacked generalization. In *Neural Networks*, 5:241-260, 1992.

[98] Q. Xie, C.A.Laszlo, and R. K.Ward. Vector quantization techniques for nonparameteric classifiers design. *IEEE Transactions Pattern Anal. Math. Intell.*, 19(1):1326-1330, 1993.

[99] H. Zhang and C. X. Ling. Learnability of Augmented Naïve Bayes Nominal Domains. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 617-623, 2001.

[100] H. Zhang, L. Jiang, and J. Su. Hidden Naive Bayes. In: *Twentieth National Conference on Artificial Intelligence*, 919-924, 2005.

[101] F. Zheng, G. I. Webb. Efficient lazy elimination for averaged-one dependence estimators. In: *Proceedings of the 23th International Conference on Machine Learning*, 1113-1120, 2006.

[102] Z. H. Zhou and X. Y. Liu. Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. *IEEE Transactions on Knowledge and Data Engineering*, 63-77, 2005.

[103] Z. H. Zhou and X. Y. Liu. On Multi-Class Cost-Sensitive Learning. In *Proceedings of the 21th National Conference on Artificial Intelligence*, 567-572, 2006.

[104] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, 912-919, 2003.