

**TOPICS IN IMBALANCED DATA CLASSIFICATION:
ADABOOST AND BAYESIAN RELEVANCE VECTOR
MACHINE**

A Dissertation presented to
the Faculty of the Graduate School
at the University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

by

WENYANG WANG

Drs. Dongchu Sun and Zhuoqiong He, Dissertation Supervisors

MAY, 2020

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

**TOPICS IN IMBALANCED DATA CLASSIFICATION:
ADABOOST AND BAYESIAN RELEVANCE VECTOR
MACHINE**

presented by Wenyang Wang,

a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Dongchu Sun

Dr. Zhuoqiong He

Dr. Tieming Ji

Dr. Athanasios Christou Micheas

Dr. Shawn Ni

ACKNOWLEDGMENTS

I am very grateful to my advisors, Dr. Dongchu Sun and Dr. Zhuoqiong He, for their guidance, inspiration, and patience to make this project possible. I would like to thank my committee members Drs. Shawn Ni, Tieming Ji, and Athanasios Christou Micheas for their constructive suggestions and discussion on my research.

I would also like to thank the other faculty members in the Department of Statistics for providing all the excellent courses from which to learn. Many thanks to Judy Dooley, Abbie Van Nice-Booher, and Kathleen Maurer for their assistance in the department. I also thank Dr. Larry Ries for his helpful teaching assistant work advice. Thanks to my fellow graduate students from the University of Missouri. Their support and help make my life so enjoyable at Columbia.

Finally, I wish to acknowledge all the love and encouragement of my parents and wife. It is to them that I dedicate this work. Many love to my daughter, who makes my life complete and motivated.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF TABLES	viii
LIST OF FIGURES	x
ABSTRACT	xi
CHAPTER	
1 Introduction	1
1.1 Imbalanced Data Problem	1
1.2 Machine Learning Approach	4
1.3 Relevance Vector Machine Approach	7
1.4 Outline of the Dissertation	9
2 The Improved AdaBoost Algorithms for Imbalanced Data Classification	11
2.1 Introduction	11
2.2 The Enhanced AdaBoost Algorithm	14
2.2.1 The Enhanced AdaBoost	14
2.2.2 Properties of the Enhanced AdaBoost Algorithm	16
2.2.3 Choices of β and k	20
2.3 The Reinforced AdaBoost Algorithm	22
2.4 Numerical Studies	26
2.4.1 Simulation Studies Based on Gaussian Data	29

2.4.2	Simulation Study Based on Uniform Data	33
2.4.3	Real Data Studies	37
2.5	Conclusion Comments	42
3	Fully Bayesian Analysis of the Relevance Vector Machine Classification for Imbalanced Data	44
3.1	Introduction	44
3.1.1	Support Vector Machine with Kernel Functions	45
3.1.2	Adaptive Rejection Sampling Method	47
3.2	RVM Classification	52
3.3	Generic Bayesian RVM Classification Algorithm	59
3.4	Fully Hierarchical Bayesian RVM Classification Algorithm	62
3.5	Numeric Studies	66
3.5.1	Simulation Data Studies	66
3.5.2	Real Data Studies	69
3.6	Conclusion Comments	70
4	Fully Bayesian Analysis of the Relevance Vector Machine Classification with Probit Link Function	72
4.1	Introduction	72
4.1.1	The Probit Link Functions	73
4.2	Generic Bayesian PRVM Classification Algorithm	74
4.3	Fully Hierarchical Bayesian PRVM Classification Algorithm	79
4.4	Numeric Studies	80
4.4.1	Simulation Data Studies	81

4.4.2	Real Data Studies	83
4.5	Comparison Between the Bayesian RVM and PRVM	83
4.5.1	Elapsed Programming Time	84
4.5.2	Model Selection	85
4.6	Conclusion Comments	87
5	Discussion and Future Research	89
5.1	Discussion of Chapter 2	89
5.2	Discussion of Chapter 3	91
5.2.1	Generative vs. Discriminative Models	91
5.2.2	The Gaussian Mixture Model	92
5.3	Discussion of Chapter 4	95
5.3.1	0 – 1 Loss Function	96
5.3.2	Cost-sensitive Loss Function	97
5.3.3	Cost-sensitive Bayes Classifier	99
5.4	Comparison Between Boosting and Kernel Methods	103
 APPENDIX		
A	Proofs of Theorems in Chapter 2	110
B	Proofs of Theorems in Chapter 3	113
C	Proofs of Lemmas in Chapter 4	116
D	Ratio of Uniforms Sampling Method	118
E	Proofs of Theorems in Chapter 5	121
	BIBLIOGRAPHY	125

4.4.2	Real Data Studies	83
4.5	Comparison Between the Bayesian RVM and PRVM	83
4.5.1	Elapsed Programming Time	84
4.5.2	Model Selection	85
4.6	Conclusion Comments	87
5	Discussion and Future Research	89
5.1	Discussion of Chapter 2	89
5.2	Discussion of Chapter 3	91
5.2.1	Generative vs. Discriminative Models	91
5.2.2	The Gaussian Mixture Model	92
5.3	Discussion of Chapter 4	95
5.3.1	0 – 1 Loss Function	96
5.3.2	Cost-sensitive Loss Function	97
5.3.3	Cost-sensitive Bayes Classifier	99
5.4	Comparison Between Boosting and Kernel Methods	103
 APPENDIX		
A	Proofs of Theorems in Chapter 2	110
B	Proofs of Theorems in Chapter 3	113
C	Proofs of Lemmas in Chapter 4	116
D	Ratio of Uniforms Sampling Method	118
E	Proofs of Theorems in Chapter 5	121
	BIBLIOGRAPHY	125

VITA 136

LIST OF TABLES

Table	Page
2.1	Summation of $D_t(i)$ in Different Conditions of y_i and $h_t(\mathbf{x}_i)$ for AdaBoost. 15
2.2	Upper Boundary of k in the Enhanced AdaBoost Algorithm. 22
2.3	Machine Learning Algorithms in AdaBoost Numeric Studies. 27
2.4	Confusion Matrix. 27
2.5	The Results of Simulated Gaussian Data in AdaBoost Numeric Studies ($b = 10$). 31
2.6	The Results of Simulated Gaussian Data in AdaBoost Numeric Studies ($b = 5$). 33
2.7	The Results of Simulated Uniform Data in AdaBoost Numeric Studies ($b = 20$). 36
2.8	The Results of Seismic-bumps Data in AdaBoost Numeric Studies ($b =$ 15.13). 39
2.9	The Results of Glass-2 Data in AdaBoost Numeric Studies ($b = 11.59$). 39
2.10	The Results of New-thyroid-1 Data in AdaBoost Numeric Studies ($b =$ 5.14). 40
2.11	The Results of Ecoli-1 Data in AdaBoost Numeric Studies ($b = 3.36$). 42

3.1	The Criteria for Classification Evaluation in RVM Studies.	55
3.2	The Results of Simulated Data in Bayesian RVM ($b = 1$).	67
3.3	The Results of Simulated Data in Bayesian RVM ($b = 2$).	68
3.4	The Results of Simulated Data in Bayesian RVM ($b = 2.5$).	68
3.5	The Results of Simulated Data in Bayesian RVM ($b = 5$).	68
3.6	The Results of Simulated Data in Bayesian RVM ($b = 10$).	69
3.7	The Results of Real Datasets in Bayesian RVM. ^a	70
4.1	The Results of Simulated Data in PRVM ($b = 1$).	81
4.2	The Results of Simulated Data in PRVM ($b = 2$).	82
4.3	The Results of Simulated Data in PRVM ($b = 2.5$).	82
4.4	The Results of Simulated Data in PRVM ($b = 5$).	82
4.5	The Results of Simulated Data in PRVM ($b = 10$).	83
4.6	The Results of Real Datasets in PRVM. ^a	84
4.7	Epalsed Programming Time ^a of Bayesian RVM and PRVM Models. ^b	85
4.8	Maximum Likelihood Value of Bayesian RVM and PRVM Models. . .	87
5.1	Risk Matrix for Binary Classification under 0 – 1 Loss Function. . . .	96
5.2	Risk Matrix for Binary Classification under Cost-sensitive Loss Function.	98
5.3	The Results of Glass-1 Data($b = 1.82$).	105
5.4	The Results of Iris-0 Data($b = 2.00$).	105
5.5	The Results of Newthyroid-1 Data($b = 5.14$).	106
5.6	The Results of Glass-6 Data($b = 6.38$).	106
5.7	The Results of Ecoli-0345 Data($b = 9.00$).	107
5.8	The Results of Glass-2 Data($b = 11.59$).	107

LIST OF FIGURES

Figure	Page
1.1 Principal Issues Leading to Imbalanced Data Problem.	4
1.2 Machine Learning Classifiers Summary.	6
2.1 Distribution of $H(x)$ in Original AdaBoost and Enhanced AdaBoost. .	19
2.2 Simulated Gaussian Data with $b = 10$ in AdaBoost Numeric Studies.	30
2.3 Simulated Uniform Data with $b = 20$ in AdaBoost Numeric Studies. .	35
3.1 The Tangential Function $w_k(x)$ at s_k in ARS Method.	50
3.2 $w_n(x)$ Based on Two Support Points in ARS Method.	51
3.3 Simulated Data for Original RVM Classification ($N_+ = N_- = 3$). . . .	58
3.4 Convergence Plot of \mathbf{w} in Original RVM Classification Algorithm. . .	59
3.5 Simulated Gaussian Data for Bayesian RVM.	67
4.1 Logistic and Probit Link Functions.	73
4.2 Sampling From a Truncated Normal Distribution.	77

ABSTRACT

This research has three parts addressing classification, especially the imbalanced data problem, which is one of the most popular and essential issues in the domain of classification.

The first part is to study the Adaptive Boosting (AdaBoost) algorithm. AdaBoost is an effective solution for classification, but it still needs improvement in the imbalanced data problem. This part proposes a method to improve the AdaBoost algorithm using the new weighted vote parameters for the weak classifiers. Our proposed weighted vote parameters are determined not only by the global error rate but also by the classification accuracy rate of the positive class, which is our primary interest. The imbalanced index of the data is also a factor in constructing our algorithms. The numeric studies show that our proposed algorithms outperform the traditional ones, especially regarding the evaluation criterion of the $F - 1$ Measure. Theoretic proofs of the advantages of our proposed algorithms are presented.

The second part treats the Relevance Vector Machine (RVM), which is a supervised learning algorithm extended from the Support Vector Machine (SVM) based on the Bayesian sparsity model. Compared with the regression problem, RVM classification is challenging to conduct because there is no closed-form solution for the weight parameter posterior. The original RVM classification algorithm uses Newton's method in optimization to obtain the mode of weight parameter posterior, then approximates it by a Gaussian distribution in Laplace's method. This original model would work, but it just applies the frequency methods in a Bayesian framework. This part first proposes a Generic Bayesian RVM classification, which is a pure Bayesian

model. We conjecture that our algorithm achieves convergent estimates of the quantities of interest compared with the nonconvergent estimates of the original RVM classification algorithm. Furthermore, a fully Bayesian approach with the hierarchical hyperprior structure for RVM classification is proposed, which improves the classification performance, especially in the imbalanced data problem.

The third part is an extended work of the second one. The original RVM classification model uses the logistic link function to build the likelihood, which makes the model hard to conduct since the posterior of the weight parameter has no closed-form solution. This part proposes the probit link function approach instead of the logistic one for the likelihood function in RVM classification, namely PRVM (RVM with the Probit link function). We show that the posterior of the weight parameter in our model follows the multivariate normal distribution and achieves a closed-form solution. A latent variable is needed in our algorithm to simplify the Bayesian computation greatly, and its conditional posterior follows a truncated normal distribution. Compared with the original RVM classification model, our proposed one is another pure Bayesian approach and it has a more efficient computation process. For the prior structure, we first consider the Normal-Gamma independent prior to propose a Generic Bayesian PRVM algorithm. Furthermore, the Fully Bayesian PRVM algorithm with a hierarchical hyperprior structure is proposed, which improves the classification performance, especially in the imbalanced data problem.

Chapter 1

Introduction

1.1 Imbalanced Data Problem

Class imbalance is one of the most severe and challenging problems in machine learning, especially in classification. The class imbalanced situation frequently appears in many fields such as facial detection (e.g., Yue (2017) and Huang et al. (2019)), financial fraud detection (e.g., Chan et al. (1998), Bian et al. (2016), and Makki et al. (2019)), network intrusion detection (e.g., Miahe et al. (2019)), software defect prediction (e.g., Bennin et al. (2017) and Song et al. (2018)), and oil exploration (e.g., Kubat et al. (1998), Haixiang et al. (2016), and Geng et al. (2018)).

A binary imbalanced data can be described as follows: Let $\mathbf{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ be the training data for a classification problem, where $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$, \mathbf{X} is a subset of l -dimensional vector space, and the response $y_i \in \{-1, 1\}$ indicates two classes, $i = 1, \dots, n$. Define $\mathbf{S}^+ = \{(\mathbf{x}_i, y_i) \in \mathbf{S} : y_i = 1, i = 1, \dots, n\}$ is the

positive or minority class; $\mathcal{S}^- = \{(\mathbf{x}_i, y_i) \in \mathcal{S} : y_i = -1, i = 1, \dots, n\}$ is the negative or majority class. The class types, {minority, majority} and {positive, negative} are used to describe $\{\mathcal{S}^+, \mathcal{S}^-\}$.

Definition 1.1. *Let $|\mathbf{A}|$ denote the number of the elements in a set \mathbf{A} . Define $N_p = |\mathcal{S}^+|$ and $N_n = |\mathcal{S}^-|$ are the numbers of samples in the positive class and negative class, respectively. The imbalanced degree of data is defined as $b = N_n/N_p$.*

If $N_n > N_p$, this imbalanced condition in the different classes is called the imbalanced data problem. b is a positive number larger than 1 in imbalanced data. There is no explicit definition of the imbalanced data only based on b since other factors can affect the classification performance such as the dimensions of data, the overlap situation, and the total sample size (see Sun et al. (2007)). Typically, when the accuracy rate of \mathcal{S}^+ is significantly lower than that of \mathcal{S}^- because of the skewed structure of the data, we treat this as the imbalanced data problem. Six significant factors are known to cause an imbalanced data problem (see Lopez et al. (2013)):

- (a) Overlap between classes;
- (b) Borderline instances;
- (c) Areas with small disjuncts;
- (d) Noisy data;
- (e) Low density and lack of information in the training data;
- (f) Possible difference in the distributions of the training and the test data.

Lee et al. (2017) indicated that these factors could be grouped into three principal issues: (1) class overlap ((a) and (b)), (2) small disjuncts ((c) and (d)), and (3) data shift ((e) and (f)). Figure 1.1 illustrates these three situations. Each of the sub-figures in Figure 1.1 includes 30 black points of the negative class and 20 blue points

of the positive class indicating $b = 1.5$ and the bias of the data is not severe. Overlap, as shown in Figure 1.1 (1), involves two classes that are not completely separated, especially when almost all the positive class points are located on the borderline of the negative class. Figure 1.1 (2) shows a small disjunct problem occurring when several small clusters of the positive points are located inside of the negative class region. As can be seen in Figure 1.1 (3), the data shift means the distributions of the training and test data are different. Even when b is small such as 1.5 in Figure 1.1, the classification accuracy of \mathcal{S}^+ would be degraded when class overlap, small disjuncts, or data shift happens.

Another factor, data size, could affect the performance of classification. Predicting the sample size required for classification is a very challenging issue in Statistical Machine Learning. Typically, a large data size helps improve the performance of classifiers. However, increasing the data size is not the panacea for classification. There is an upper boundary for the accuracy rate of classifiers when we increase the training data size (see Figueroa et al. (2012)). Unlimited increasing the data size is useless, and people desire the proper amount of training data, which is dependent on many different aspects of the experiment. In classification, especially in the high-dimensional data classification, enough data points are necessary for building the borderlines to break the curse of dimensionality. But in the imbalanced data problem, more data near the borderline are needed to balance the skewness between two classes. Adding more data away from the borderline cannot help to solve the imbalanced data problem. Because there is not any guideline to indicate the information of data near the borderline, it is impossible to improve the performance of classifiers through the increasing data size.

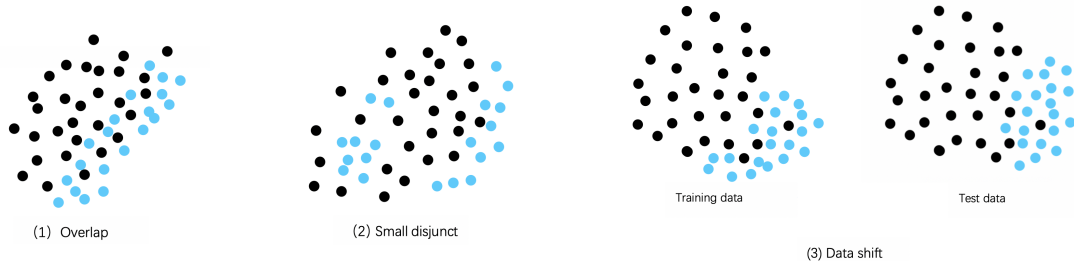


Figure 1.1: Principal Issues Leading to Imbalanced Data Problem.

Under the balanced situation, $N_n \approx N_p$, many algorithms are available for classification such as linear classifiers, Support Vector Machines, decision trees, random forest, and so on (see Alpaydin (2009)). For the imbalanced situation, although the algorithms mentioned above still can obtain an excellent global accuracy rate, the classification accuracy rate of \mathcal{S}^+ can be low. A worse result is that the classifier might identify the positive samples as noise and ignore them in the training process. If \mathcal{S}^+ is our particular interest, we need some other specific algorithms for the classification. In a cancer diagnostic where the cancer cases are quite rare compared with the healthy populations. People desire an optimal classifier that gives the maximum accuracy rate of the patients' category without sacrificing the global accuracy rate much. Only the binary imbalanced class problem is studied in this whole project. Multiple-class issues can be solved based on the one-versus-all scheme (see Schapire & Singer (1999)).

1.2 Machine Learning Approach

In the machine learning field, data level and algorithm level methods are two typical approaches (see Weiss (2004) and Kotsiantis et al. (2006)) to solve the imbalanced

class problem. The former is a pre-processing data method (e.g., Batista et al. (2004) and Hulse et al. (2007)), where resampling is utilized frequently. The basic idea of the data level method is to delete the instances in \mathcal{S}^- or increase the instances in \mathcal{S}^+ to change the data sizes of the two classes and relieve the imbalanced situation before the training process. Although the data-level approach is simple, the random under-sampling or over-sampling makes the pre-processing dataset different from the raw data. Due to this limitation, the data level method is used less often. On the other hand, the algorithm level method focuses on training the classifier directly with the original data. The most common ones are Cost-sensitive Learning proposed in Cheng (2016) and Ensemble Schemes, including Boosting (see Viola et al. (2002) and Chawla et al. (2003)) and Bagging (see Galar et al. (2016)). Cost-sensitive Learning assumes that the misclassified costs have prior information to enhance the impact of the minority class. The limitation is that the prior information regarding the costs is hard to find. Also, if the minority class samples are sparse, Cost-sensitive Learning may not construct an appropriate decision boundary (see Lopez et al. (2012)). Figure 1.2 summarizes the popular classifiers for balanced data and imbalanced data.

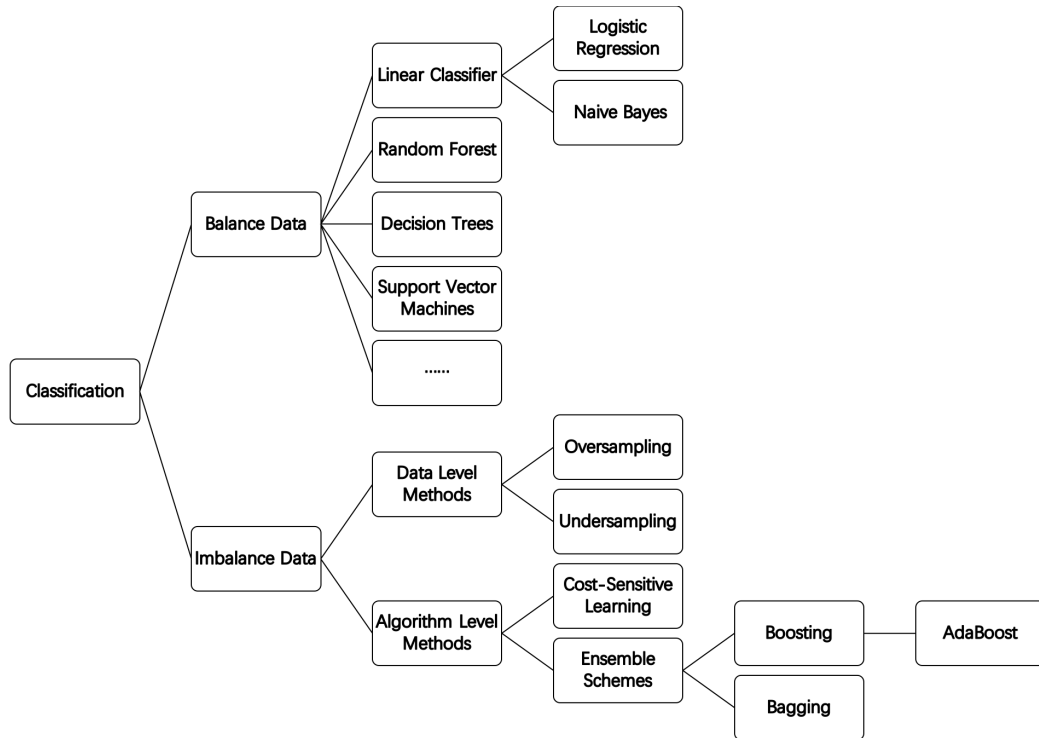


Figure 1.2: Machine Learning Classifiers Summary.

This project studies Adaptive Boosting (AdaBoost) algorithm, which is the most popular one in Boosting. Boosting is a kind of Ensemble Schemes and it can combine a series of weak classifiers that are only a little better than a random guess to construct a robust classifier. The Boosting approach is a resultful method to design the classification algorithm when it is hard to construct a single strong classifier alone. AdaBoost is the typical one in Boosting. More details and further improvement of AdaBoost are discussed in Chapter 2.

1.3 Relevance Vector Machine Approach

Relevance Vector Machine (RVM) proposed by Tipping (2000) is a Kernel method algorithm. The kernel method has become popular with the proposal of the Support Vector Machine (SVM) by Cortes & Vapnik (1996) and motivated the rapid growth of the classification algorithms (e.g., Sain (1996) and Shawe et al. (2004)). SVM is designed based on the Structural Risk Minimization (SRM) and it projects the linearly indivisible data in the high dimensional linear separable space. Boser et al. (1992) indicated that SVM constructs an optimal separating hyperplane as the classification borderline, which can obtain the maximum distance between two classes for a binary dataset.

Although SVM has been the leading method in classification, it still has a few limitations indicated by Tipping (2001):

(1) The numbers of kernels and parameters increase linearly with the growth of the training data size. The free parameters in SVM usually require the cross-validation for the optimizations. SVM would be inefficient in a large data case;

(2) The classification result is not a probability and does not provide an accuracy rate.

Tipping (2000) and Tipping (2001) originally proposed the Relevance Vector Machine (RVM) based on SVM to solve the above two limitations by conducting a sparsity prior to the weight parameter \mathbf{w} . Afterward, a fast sequence RVM algorithm was proposed in Tipping (2003), and it speeded up the training process significantly. Later, Thayananthan (2006) extended the RVM algorithm to multiple-output regression and multiple-output classification. RVM has obtained more applications in text image recognition (e.g., Silva & Ribeiro (2006)), image classification (e.g., Demir &

Erturk (2007)), time series analysis (e.g., Nikolaev & Tino (2005)), mechanical fault diagnosis (e.g., Wu et al. (2011)) and electric demand forecasting (e.g., Liu et al. (2004)). However, the most consequential and influential work related to RVM is in the regression field. RVM classification is hard to be conducted in a pure Bayesian framework because there is no closed-form solution for the posterior of weight parameter \mathbf{w} . The original RVM classification applies Newton's method to seek the mode of posterior of \mathbf{w} , then approximates it with a normal distribution in the Laplace's method instead of directly sampling \mathbf{w} from its posterior. The original RVM classification was built in a Bayesian framework but conducted with frequency methods. This project studies the original RVM classification and concludes the shortcomings of it. A Bayesian RVM model is proposed in Chapter 3, which keeps the same likelihood and weight parameter prior to the original RVM. But Bayesian RVM does the sampling process directly from the weight parameter posterior instead of the frequency methods in the original one. Therefore, a hierarchical structure is considered for the Bayesian RVM to improve the classification performance in the imbalanced data problem. Chapter 4 re-defines the likelihood of RVM by the probit link function instead of the logistic one in the original RVM and we propose the PRVM model. Benefiting from a latent variable, this new likelihood produces a concise weight parameter posterior, which follows a multivariate normal distribution and has the closed-form solution. PRVM has similar classification performances compared with the Bayesian RVM, but PRVM is more succinct and the programming is also more efficient in the case of code running time.

1.4 Outline of the Dissertation

Chapter 2 discusses the machine learning approach for imbalanced data classification problems. It studies and improves the AdaBoost algorithm. In this chapter, we first propose an improved AdaBoost algorithm, namely the Enhanced AdaBoost, to improve the classification performance using the new weighted vote parameters for the weak classifiers in constructing the algorithm. Theoretic proofs are given to support the advantages of it compared with the original one. Then, another improved AdaBoost, the Reinforced AdaBoost algorithm, is proposed. It solves a restriction in the Enhanced AdaBoost and is a comprehensive solution for the AdaBoost algorithm improvement. Some numeric studies are finally conducted to support our proposed algorithms' preponderance in practice.

Chapter 3 first studies the original Relevance Vector Machine classification model. We indicate the original one is not a pure Bayesian approach and has several theoretic shortcomings. Then, this chapter shows that the posterior of the weight parameter in RVM has the desired log-concave property so that we can do the sampling directly through the Adaptive Rejection Sampling method. We propose this pure Bayesian model, namely Bayesian Relevance Vector Machine (Bayesian RVM). Next, a hierarchical prior structure is employed for Bayesian RVM to improve the classification performance in the imbalanced data classification problem. Finally, the numeric studies are conducted to apply our proposed algorithms into practice.

Chapter 4 extends the work of Chapter 3. The original RVM does not have a closed-form solution in the posterior of the weight parameter since the logistic link function makes the likelihood function a complex expression. In this chapter, we propose the probit link function to construct the likelihood function instead of the

logistic one. The probit link function is similar to the logistic one, but it helps to construct a succinct model. We show that the RVM with probit link function (PRVM) is much more efficient and readable compared with the original one. The weight parameter in PRVM follows a multivariate normal distribution, which can be sampled quickly. The same hierarchical prior structure in Chapter 3 is studied again to solve the imbalanced data problem. The same simulated datasets and real datasets, as in Chapter 3, are employed in this chapter to evaluate the performance of PRVM in practice. Finally, we make a model comparison between the Bayesian RVM and the PRVM.

Chapter 2

The Improved AdaBoost Algorithms for Imbalanced Data Classification

2.1 Introduction

The AdaBoost formulated by Freund and Schapire (1997) is a popular Boosting algorithm in Ensemble Schemes. It trains a series of weak classifiers iteratively based on the weighted data. The outputs of all weak classifiers are combined into a weighted summation that presents the final boosted classifier. A review of the AdaBoost algorithm is stated below.

Algorithm 2.1. The Original AdaBoost Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$. Choose the number of weak classifiers T .

0. Let $t = 1$ and initialize the sample weight $D_t(i) = 1/n$, $i = 1, \dots, n$;

1. Find the weak classifier $h_t(\mathbf{x}) \in \{-1, 1\}$ minimizing the weighted error rate $\epsilon_t = \sum_{i: h_t(\mathbf{x}_i) \neq y_i} D_t(i)$;

2. Calculate the weak classifier weight $\alpha_1^t = \frac{1}{2} \log\{(1 - \epsilon_t)/\epsilon_t\}$;

3. Update $D_{t+1}(i) = D_t(i) \exp\{-\alpha_1^t y_i h_t(\mathbf{x}_i)\}$, $i = 1, \dots, n$, renormalize it as $D_{t+1}(i) = D_{t+1}(i) / \sum_{j=1}^n D_{t+1}(j)$;

4. If $t = T$, stop the iteration, else let $t = t + 1$ and return to Step **1**.

Output. The final strong classifier is $C_1(\mathbf{x}) = \text{sign}\{H_1(\mathbf{x}) - M_1\}$, where M_1 is the threshold, and

$$H_1(\mathbf{x}) = \sum_{t=1}^T \alpha_1^t h_t(\mathbf{x}).$$

A weak classifier $h_t(\mathbf{x})$ presents a map from \mathbf{X} to $\{-1, 1\}$. When $h_t^*(\mathbf{x})$ is any map from \mathbf{X} to \mathbf{R} , we can use $h_t(\mathbf{x}) = \text{sign}(h_t^*(\mathbf{x}))$ to transfer $h_t^*(\mathbf{x})$ to a weak classifier. When $h_t^*(\mathbf{x}_i)$ is zero, randomly set \mathbf{x}_i as the positive or negative class. α^t is the corresponding weighted vote parameter of the weak classifiers. Note that α_1^t in Step **2** is chosen based on the method of minimizing the exponential loss function (see Rojas (2009)), which implies that the more accurate a weak classifier is, the larger vote weight it has in constructing the final strong classifier. Step **3** presents the weight of the samples, $D_t(i)$. If Sample (\mathbf{x}_i, y_i) is misclassified by $h_t(\mathbf{x})$, then $D_{t+1}(i) > D_t(i)$

in $h_{t+1}(\mathbf{x})$. On the other hand, if Sample (\mathbf{x}_i, y_i) is classified correctly by $h_t(\mathbf{x})$, then $D_{t+1}(i) < D_t(i)$ in $h_{t+1}(\mathbf{x})$ (see Freund and Schapire (1999)). So in AdaBoost, every weak classifier would get updated by focusing on the misclassified samples in the previous iteration to improve the global classification accuracy. By highlighting the misclassified samples and combining all the weak classifiers to build a strong one, AdaBoost is regarded as a popular and powerful algorithm in classification.

If the positive class \mathbf{S}^+ is our primary interest, the AdaBoost algorithm should be improved because it uses the same $D_t(i)$ for all samples in both \mathbf{S}^+ and \mathbf{S}^- . Also, it chooses the α_1^t only based on the global error rate ϵ_t . In recent years, several improved AdaBoost algorithms have been proposed for the imbalanced class problem. Typically, they include adjusting $D_t(i)$ or α_1^t . AdaCost (see Fan et al. (1999)) and Cost-sensitive AdaBoost (see Zhou et al. (2017) and Tao et al. (2019)) adjusted $D_t(i)$ by adding a higher misclassification cost to the minority class. They assimilated the Cost-sensitive Learning into the AdaBoost but these methods have a similar limitation as of the Cost-sensitive Learning. Adjusting α_1^t is a mighty improvement without apparent shortcomings. Li et al. (2009) proposed AD AdaBoost with a new α^t to improve the AdaBoost algorithm and received a convincing result in the object detection problem. In this chapter, we first propose an Enhanced AdaBoost algorithm which is based on the AD AdaBoost but designed for the imbalanced class problem by adjusting α_1^t . When the imbalanced index b is large, the Enhanced AdaBoost can obtain an excellent classification result. However, b is not the only factor to define the imbalanced data. Recall Figure 1.1, when b is small, we still could face an imbalanced problem. So we propose the Reinforced AdaBoost, which is another improved AdaBoost algorithm. The Reinforced AdaBoost can perform better when

the imbalanced index b is relatively small but needs an additional iteration compared with the Enhanced AdaBoost. Similar to AD AdaBoost, our improved α^t includes not only the global error rate but also the classification accuracy rate of \mathbf{S}^+ . Also, the imbalanced index b is considered in our proposed algorithms. The final classifier with our proposed weight parameters α^t can focus on \mathbf{S}^+ to increase the criterion of $F - 1$ Measure, which is the evaluation criterion we are most interested in, without losing the global accuracy rate.

2.2 The Enhanced AdaBoost Algorithm

2.2.1 The Enhanced AdaBoost

We follow the notations, $D_t(i)$ and ϵ_t in **Algorithm 2.1**, b in **Definition 1.1**.

Definition 2.1. Define $K_t = \sum_{i:y_i=1} D_t(i)$, $P_t = \sum_{i:y_i=1, h_t(\mathbf{x}_i)=1} D_t(i)$, and $\gamma_t = P_t/K_t$.

For every $h_t(\mathbf{x})$, K_t is the summation of weights of all samples in \mathbf{S}^+ , P_t means the summation of weights of the correctly classified samples in \mathbf{S}^+ , γ_t is a measure of the classification ability of $h_t(\mathbf{x})$ in \mathbf{S}^+ . Note that $\gamma_t \in [0, 1]$, $b \in (1, \infty)$, and $\epsilon_t \in [0, 0.5]$ since it is assumed that all the classifiers have to be better than a random guess. The summation of sample weights under different conditions of y_i and $h_t(\mathbf{x}_i)$ are presented in Table 2.1. The global error rate ϵ_t is the summation of the false positive rate $A_{1,-1}^t$ and the false negative rate $A_{-1,1}^t$, $\epsilon_t = A_{1,-1}^t + A_{-1,1}^t$. $K_t = A_{1,\cdot}^t$ and $P_t = A_{1,1}^t$ are based on **Definition 2.1**. The proportion of $A_{1,1}^t$ in $A_{1,\cdot}^t$ is presented by $\gamma_t = P_t/K_t$. Note that in the t^{th} step of the iterations, the number of correctly classified samples in \mathbf{S}^+ is $N_p \gamma_t$.

Table 2.1: Summation of $D_t(i)$ in Different Conditions of y_i and $h_t(\mathbf{x}_i)$ for AdaBoost.

	$h_t(\mathbf{x}_i) = 1$	$h_t(\mathbf{x}_i) = -1$	Row Total
$y_i = 1$	$A_{1,1}^t = \sum_{i:y_i=1,h_t(\mathbf{x}_i)=1} D_t(i)$	$A_{1,-1}^t = \sum_{i:y_i=1,h_t(\mathbf{x}_i)=-1} D_t(i)$	$A_{1,\cdot}^t = \sum_{i:y_i=1} D_t(i)$
$y_i = -1$	$A_{-1,1}^t = \sum_{i:y_i=-1,h_t(\mathbf{x}_i)=1} D_t(i)$	$A_{-1,-1}^t = \sum_{i:y_i=-1,h_t(\mathbf{x}_i)=-1} D_t(i)$	$A_{-1,\cdot}^t = \sum_{i:y_i=-1} D_t(i)$
Column Total	$A_{\cdot,1}^t = \sum_{i:h_t(\mathbf{x}_i)=1} D_t(i)$	$A_{\cdot,-1}^t = \sum_{i:h_t(\mathbf{x}_i)=-1} D_t(i)$	1

Now we propose the Enhanced AdaBoost with $D_t(i)$, ϵ_t , γ_t , and b defined above.

Algorithm 2.2. The Enhanced AdaBoost Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$. Choose the number of weak classifiers T , the parameters β and k . $b = N_n/N_p$ is determined by the training data.

0. Let $t = 1$ and initialize the sample weight $D_t(i) = 1/n$, $i = 1, \dots, n$;

1. Find the weak classifier $h_t(\mathbf{x}) \in \{-1, 1\}$ minimizing the weighted error rate

$$\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i).$$

Let $\gamma_t = \frac{A_{1,1}^t}{A_{1,\cdot}^t}$, if $\gamma_t > \frac{1}{2}$, ϵ_t should satisfy $\epsilon_t < \frac{1}{2}\{1 - (2\gamma_t - 1)/(b + 1)\}$, else repeat this step;

2. Calculate the weak classifier weight

$$\alpha_2^t = \frac{1}{2} \log\{(1 - \epsilon_t)/\epsilon_t\} + k \exp\{\beta(2\gamma_t - 1)\}, \quad (2.1)$$

where k and β are two parameters, $k > 0$;

3. Update $D_{t+1}(i) = D_t(i) \exp\{-\alpha_1^t y_i h_t(\mathbf{x}_i)\}$, $i = 1, \dots, n$, renormalize it as

$$D_{t+1}(i) = D_{t+1}(i) / \sum_{j=1}^n D_{t+1}(j);$$

4. If $t = T$, stop the iteration, else let $t = t + 1$ and return to Step 1.

Output. The final strong classifier is $C_2(\mathbf{x}) = \text{sign}\{H_2(\mathbf{x}) - M_2\}$, where M_2 is the threshold, and

$$H_2(\mathbf{x}) = \sum_{t=1}^T \alpha_2^t h_t(\mathbf{x}).$$

2.2.2 Properties of the Enhanced AdaBoost Algorithm

In (2.1), α_2^t includes two important parameters ϵ_t and γ_t , which means that α_2^t considers not only the global error rate but also the classification ability of \mathbf{S}^+ (see Li et al. (2007)). In Step 1 of **Algorithm 2.2**, b plays an important role and constructs a restriction for the global error rate ϵ_t when $\gamma_t > \frac{1}{2}$. This restriction is weak when b is large and should be satisfied easily in a severe imbalanced data problem.

Fact 2.1. *Define*

$$H_a(\mathbf{x}) = k \sum_{t=1}^T h_t(\mathbf{x}) \exp[\beta(2\gamma_t - 1)], \quad (2.2)$$

$H_1(\mathbf{x})$ and $H_2(\mathbf{x})$ in **Algorithm 2.1** and **2.2** satisfy $H_2(\mathbf{x}) = H_1(\mathbf{x}) + H_a(\mathbf{x})$.

Note that $C_2(\mathbf{x})$ in **Algorithm 2.2** is a monotone function of $H_2(\mathbf{x})$. Improving the performance of $C_2(\mathbf{x})$ is equivalent to adjusting $H_2(\mathbf{x})$. The idea is to improve the classification accuracy rate by making the value of $H_2(\mathbf{x})$ for \mathbf{S}^+ increase and the value of $H_2(\mathbf{x})$ for \mathbf{S}^- decrease compared to $H_1(\mathbf{x})$, respectively. Figure 2.1 is a frequency histogram of $H_1(\mathbf{X})$ and $H_2(\mathbf{X})$ based on the simulated Gaussian data

part in the numeric study section with $b = 10$, $\beta = 0.5$, and $k = 0.0009$. In Figure 2.1, $H_2(\mathbf{x})$ of \mathbf{S}^+ and \mathbf{S}^- moves further away than the $H_1(\mathbf{x})$ does, which means the Enhanced AdaBoost would result in more accurate classification.

Fact 2.2. *In (2.2), $H_a(\mathbf{x})$ satisfies*

$$\begin{aligned}\sum_{i:y_i=1} H_a(\mathbf{x}_i) &= N_p k \sum_{t=1}^T (2\gamma_t - 1) \exp\{\beta(2\gamma_t - 1)\}, \\ \sum_{i:y_i=-1} H_a(\mathbf{x}_i) &= N_p k \sum_{t=1}^T \{(2\gamma_t - 1) + (2\epsilon_t - 1)(b + 1)\} \exp\{\beta(2\gamma_t - 1)\}.\end{aligned}$$

Note that for \mathbf{S}^+ and \mathbf{S}^- , we have

$$\begin{aligned}\sum_{i:y_i=1} h_t(\mathbf{x}_i) &= N_p \gamma_t \cdot 1 + N_p (1 - \gamma_t) \cdot (-1) \\ &= N_p (2\gamma_t - 1),\end{aligned}$$

$$\begin{aligned}\sum_{i:y_i=-1} h_t(\mathbf{x}_i) &= \{N_n - [(N_n + N_p)\epsilon_t - (N_p - N_p \gamma_t)]\} \cdot (-1) + \\ &\quad [(N_p + N_n)\epsilon_t - (N_p - N_p \gamma_t)] \cdot 1 \\ &= 2\{(N_p + N_n)\epsilon_t - (N_p - N_p \gamma_t)\} - N_n.\end{aligned}$$

Fact 2.2 follows from **Fact 2.1**. We consider three situations of γ_t : (a) $\gamma_t \in [0, \frac{1}{2})$, (b) $\gamma_t = \frac{1}{2}$, and (c) $\gamma_t \in (\frac{1}{2}, 1]$. It is often that γ_t is larger than 0.5 but in some steps, especially in the last few steps of iterations, γ_t might be smaller than 0.5 if the distribution of \mathbf{S}^+ is sparse or the small disjunct situation is serious.

Theorem 2.1. (a) For $\gamma_t \in [0, \frac{1}{2})$,

$$\sum_{i:y_i=-1} H_a(\mathbf{x}_i) < \sum_{i:y_i=1} H_a(\mathbf{x}_i) < 0, \quad (2.3)$$

(b) For $\gamma_t = \frac{1}{2}$,

$$\sum_{i:y_i=-1} H_a(\mathbf{x}_i) < \sum_{i:y_i=1} H_a(\mathbf{x}_i) = 0, \quad (2.4)$$

(c) For $\gamma_t \in (\frac{1}{2}, 1]$, assume that

$$\epsilon_t < 0.5(1 - \frac{2\gamma_t - 1}{b + 1}) < 0.5, \quad (2.5)$$

we have

$$\sum_{i:y_i=-1} H_a(\mathbf{x}_i) < 0 < \sum_{i:y_i=1} H_a(\mathbf{x}_i). \quad (2.6)$$

Proof. See Appendix A1. □

In all three situations of **Theorem 2.1**, $\sum_{i:y_i=-1} H_a(\mathbf{x}_i) < \sum_{i:y_i=1} H_a(\mathbf{x}_i)$ always holds to improve the ability to distinguish \mathbf{S}^+ and \mathbf{S}^- , but (c) does a better job than (a) and (b). In (c), $H_2(\mathbf{x})$ of \mathbf{S}^+ and \mathbf{S}^- move toward opposite directions compared to $H_1(\mathbf{x})$ because of adding $H_a(\mathbf{x})$. However, in (a) and (b), $H_2(\mathbf{x})$ moves to the negative direction compared to $H_1(\mathbf{x})$, $H_2(\mathbf{x})$ of \mathbf{S}^- has a larger movement than that of \mathbf{S}^+ . Fortunately, (c) happens more frequently in the training process. Combine all $h_t(\mathbf{x})$ with α_2^t as the weights to construct the final classifier $C_2(\mathbf{x})$. \mathbf{S}^+ and \mathbf{S}^- would be separated toward the opposite directions. Figure 2.1 gives a refined

interpretation of this. It indicates that $H_2(\mathbf{x})$ has a further movement than $H_1(\mathbf{x})$ in the right direction. The two curves of $H_1(\mathbf{x})$ intersect around zero. If we choose the intersection point as the threshold and the curves beyond the threshold point in each class cause the misclassification. But the two curves of $H_2(\mathbf{x})$ are separated entirely. Choose any point between the gap of the two curves as the threshold, we could obtain a perfect classifier.

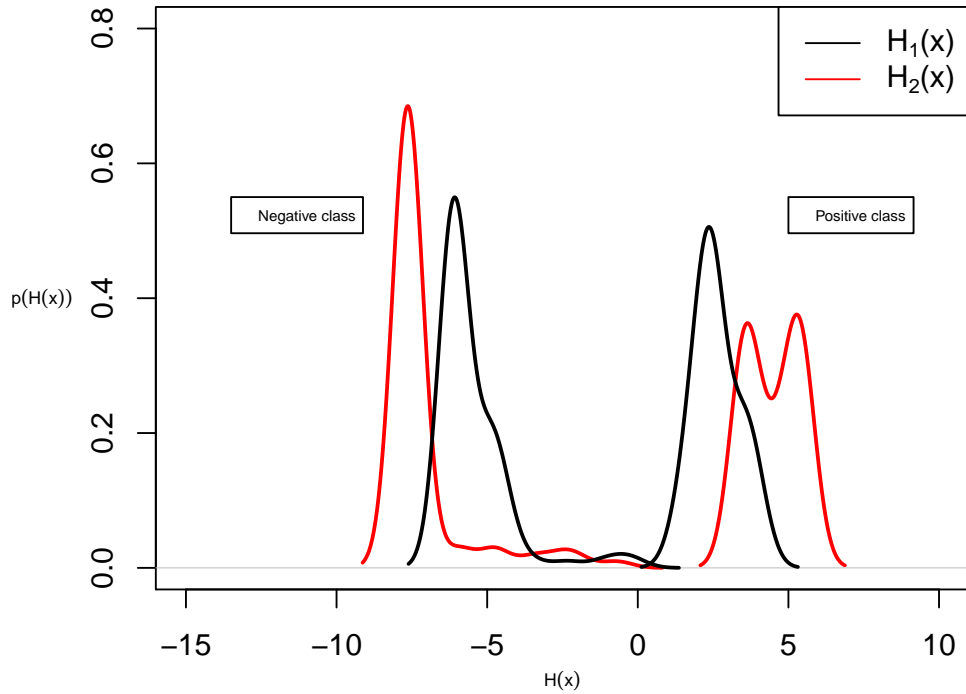


Figure 2.1: Distribution of $H(x)$ in Original AdaBoost and Enhanced AdaBoost.

2.2.3 Choices of β and k

The Enhanced AdaBoost requires pre-specification of k and β , $k > 0$. $H_a(\mathbf{x})$ consisting of k and β contributes to $H_2(\mathbf{x})$ to improve the classification performance compared to $H_1(\mathbf{x})$. The parameters k and β are important to ensure that the Enhanced AdaBoost can not only increase the classification accuracy rate for \mathbf{S}^+ but also keep the global error rate low.

In the Enhanced AdaBoost, we need α_2^t to be an increasing function of γ_t . So that under the same ϵ_t , the weak classifier will have a larger vote weight if it has a stronger classification ability for \mathbf{S}^+ . Express this condition as

$$\frac{\partial}{\partial \gamma_t} \alpha_2^t > 0, \quad (2.7)$$

which implies

$$\beta > 0. \quad (2.8)$$

We generally choose $\beta = 0.5$.

The parameter k is a positive parameter, which decides the adjusting part in α_2^t . A large k can make the stretching effect more obvious. But the classification ability of \mathbf{S}^- would decrease by a lot if we increase k unadvisedly. A larger k could make the first term of (2.1) less important. So it is important to determine a proper value of k . Define $Z_t = \sum_{j=1}^n D_{t+1}(j)$, $t = 1, \dots, T - 1$. Schapire et al. (1999) proved that for

AdaBoost, the global training error satisfies:

$$\frac{1}{n} |\{i : H(\mathbf{x}_i) \neq y_i\}| \leq \prod_{t=1}^{T-1} Z_t, \quad (2.9)$$

where $|\cdot|$ means the number of elements in the set, $i = 1, \dots, n$.

For $t = 1, \dots, T-1$, the upper boundary of the global training error should decrease to receive a more accurate final classifier when a new weak classifier is added, which means that we need $Z_t < 1$ (see Li et al. (2007)). In the Enhanced AdaBoost, Harrington (2012) showed that Z_t is:

$$Z_t = (1 - \epsilon_t) \exp\{-\alpha_2^t\} + \epsilon_t \exp\{\alpha_2^t\}. \quad (2.10)$$

By solving the inequality $Z_t < 1$,

$$0 < k < \frac{0.5 \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)}{\exp\{\beta(2\gamma_t - 1)\}}. \quad (2.11)$$

The upper boundary of k in (2.11) is a decreasing function of ϵ_t . Recall (2.5), in the most frequent case, we rewrite (2.11) as:

$$0 < k \leq \frac{0.5 \log \left(\frac{1 + \frac{2\gamma_t - 1}{b+1}}{1 - \frac{2\gamma_t - 1}{b+1}} \right)}{\exp\{\beta(2\gamma_t - 1)\}} < \frac{0.5 \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)}{\exp\{\beta(2\gamma_t - 1)\}},$$

where $\gamma_t \in (\frac{1}{2}, 1]$.

We recommend to set $\beta = 0.5$. b is specified when the training data is given. We only consider $\gamma_t \in (\frac{1}{2}, 1]$ for computational simplicity. Given $\beta = 0.5$, $b =$

(2, 5, 10, 20, 50, 100), and $\gamma_t = (0.505, 0.6, 0.7, 0.8, 0.9, 1.0)$, the upper boundaries of k can be obtained in Table 2.2. Here we keep all the numbers rounded to the lowest value with four decimal places. For example, in the first part of the simulated Gaussian data study, $b = 10$. We choose $\beta = 0.5, k = 0.0009$ based on Table 2.2 for the new weak classifier weights α_2^t . This choice can ensure (2.9) holds and the global error rate remains low in our proposed algorithm.

Table 2.2: Upper Boundary of k in the Enhanced AdaBoost Algorithm.

$\gamma_t \backslash b$	2	5	10	20	50	100
0.505	0.0033	0.0016	0.0009	0.0004	0.0001	0.0001
0.6	0.0604	0.0301	0.0164	0.0086	0.0035	0.0017
0.7	0.1098	0.0546	0.0297	0.0155	0.0064	0.0032
0.8	0.1501	0.0743	0.0404	0.0211	0.0087	0.0044
0.9	0.1831	0.0899	0.0488	0.0255	0.0105	0.0053
1.0	0.2102	0.1020	0.0552	0.0289	0.0118	0.0060

2.3 The Reinforced AdaBoost Algorithm

The Enhanced AdaBoost has the restriction of ϵ_t in (2.5), including b and γ_t . The restriction would be weak when the value of b is large, so that Step 1 in **Algorithm 2.2** does not need to be repeated. But b is not the only factor to define imbalanced data. Assume we have a dataset with $b = 2$. In some particular training iteration of the Enhanced AdaBoost, $\gamma_t = 0.99$, the restriction of ϵ_t in (2.5) is $0 \leq \epsilon_t < 0.3367$, which is a small subinterval of $0 \leq \epsilon_t < 0.5$. This situation often happens for a

small b and Step 1 in **Algorithm 2.2** would be repeated to seek for a stronger $h_t(\mathbf{x})$ which satisfies the restriction. The worst situation is that under small b , there is no such $h_t(\mathbf{x})$ minimizing ϵ_t and satisfying the restriction of ϵ_t in (2.5). So we need to propose a new α^t without any restriction to obtain the same further separation of $H_t(\mathbf{x})$ for \mathbf{S}^+ and \mathbf{S}^- as α_2^t does. Note that if $0.5(1 - \frac{2\gamma_t - 1}{b + 1}) \leq \epsilon_t < 0.5$, γ_t has to be in the interval of $(0.5, 1]$. Now we propose another improved AdaBoost algorithm by considering a new modified parameter α_3^t called the Reinforced AdaBoost.

Algorithm 2.3. The Reinforced AdaBoost Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$. Choose the number of weak classifiers T , the parameters β and k . $b = N_n/N_p$ is determined by the training data.

0. Let $t = 1$ and initialize the sample weight $D_t(i) = 1/n$, $i = 1, \dots, n$;

1. Find the weak classifier $h_t \in \{-1, 1\}$ minimizing the weighted error rate

$$\epsilon_t = \sum_{i: h_t(\mathbf{x}_i) \neq y_i} D_t(i), \text{ let}$$

$$\gamma_t = \frac{A_{1,1}^t}{A_{1,\cdot}^t};$$

2.1. When (1) $0 \leq \gamma_t \leq \frac{1}{2}$ or (2) $\frac{1}{2} < \gamma_t \leq 1$ and $\epsilon_t < \frac{1}{2}\{1 - (2\gamma_t - 1)/(b + 1)\}$, calculate the weak classifier weight

$$\alpha^t = \alpha_2^t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) + k \exp\{\beta(2\gamma_t - 1)\};$$

2.2. When $\frac{1}{2} < \gamma_t \leq 1$ and $\frac{1}{2}\{1 - (2\gamma_t - 1)/(b + 1)\} \leq \epsilon_t$, calculate the weak

classifier weight

$$\alpha^t = \alpha_3^t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \left\{ \frac{\exp(\gamma_t - 0.5) + 0.5}{0.5 - \epsilon_t} + \frac{0.5(b + 1)}{\gamma_t - 0.5} \right\}; \quad (2.12)$$

3. Update $D_{t+1}(i) = D_t(i) \exp\{-\alpha_1^t y_i h_t(\mathbf{x}_i)\}$, $i = 1, \dots, n$, renormalize it as

$$D_{t+1}(i) = D_{t+1}(i) / \sum_{j=1}^n D_{t+1}(j);$$

4. If $t = T$, stop the iteration, else let $t = t + 1$ and return to Step 1.

Output. The final strong classifier is $C_3(\mathbf{x}) = \text{sign}\{H_3(\mathbf{x}) - M_3\}$, where $H_3(\mathbf{x}) = \sum_{t=1}^T \alpha^t h_t(\mathbf{x})$, M_3 is the threshold.

Here $A_{1,1}^t$ and $A_{1,\cdot}^t$ follow from Table 2.1. Define $H_3'(\mathbf{x}) = \sum_{t=1}^{T'} \alpha_3^t h_t(\mathbf{x})$, T' is the number of weak classifiers that have the weighted vote parameters of α_3^t . We can obtain **Fact 2.3** based on **Fact 2.1** and **2.2**.

Fact 2.3. *We have the equations:*

$$\begin{aligned} \sum_{i:y_i=1} H_1(\mathbf{x}_i) &= (\gamma_t - 0.5) \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) N_p, \\ \sum_{i:y_i=-1} H_1(\mathbf{x}_i) &= \{(\gamma_t - 0.5) + (\epsilon_t - 0.5)(b + 1)\} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) N_p, \\ \sum_{i:y_i=1} H_3'(\mathbf{x}_i) &= (\gamma_t - 0.5) \left\{ \frac{\exp(\gamma_t - 0.5) + 0.5}{0.5 - \epsilon_t} + \frac{0.5(b + 1)}{\gamma_t - 0.5} \right\} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) N_p, \\ \sum_{i:y_i=-1} H_3'(\mathbf{x}_i) &= \{(\gamma_t - 0.5) + (\epsilon_t - 0.5)(b + 1)\} \left\{ \frac{\exp(\gamma_t - 0.5) + 0.5}{0.5 - \epsilon_t} + \frac{0.5(b + 1)}{\gamma_t - 0.5} \right\} \\ &\quad \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) N_p. \end{aligned}$$

Theorem 2.2. *Assume that*

$$\gamma_t \in \left(\frac{1}{2}, 1\right] \text{ and } 0.5\left(1 - \frac{2\gamma_t - 1}{b + 1}\right) \leq \epsilon_t < 0.5, \quad (2.13)$$

we have

$$\sum_{i:y_i=1} H'_3(\mathbf{x}_i) > \sum_{i:y_i=1} H_1(\mathbf{x}_i), \quad (2.14)$$

$$\sum_{i:y_i=-1} H'_3(\mathbf{x}_i) \leq \sum_{i:y_i=-1} H_1(\mathbf{x}_i). \quad (2.15)$$

Proof. See Appendix A2. □

Theorem 2.2 indicates that under $\frac{1}{2}\{1 - (2\gamma_t - 1)/(b + 1)\} \leq \epsilon_t$ with our proposed α_3^t , $H(\mathbf{x})$ of \mathcal{S}^+ and \mathcal{S}^- still can obtain a further separation in positive and negative directions than the AdaBoost does, respectively. (2.14) is a strict inequation, which means the positive class can obtain a strict increase in the classification result. The Reinforced AdaBoost eliminates the restriction of global error in the Enhanced AdaBoost and is a comprehensive solution to improve the AdaBoost algorithm for the imbalanced data problem. But the additional iteration in the definition of α^t in Step **2** of the Reinforced AdaBoost makes it more complex compared with the Enhanced AdaBoost. So when b is small, the Reinforced AdaBoost is the preferential choice. Otherwise, the Enhanced AdaBoost is powerful enough for the imbalanced data classification.

2.4 Numerical Studies

In this performance evaluation, we apply several algorithms, including our two proposed ones, the Enhanced AdaBoost and the Reinforced AdaBoost, into two kinds of simulated datasets and four real datasets. Because the goal of this chapter is to improve the AdaBoost algorithm, we choose three kinds of original AdaBoost algorithms with different weak classifiers as the comparisons. On the other hand, Support Vector Machines (SVMs) is a popular classifier. Especially, SVMs with Radial Basis Function (RBF) kernel has the congenital advantage in Gaussian data classification. The RBF kernel is defined as

$$K_{RBF}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2),$$

where γ defines how far the influence of a single training example reaches. So SVMs with RBF kernel is under our consideration. Table 2.3 summarizes all the algorithms we use.

Evaluation criteria play a crucial role in assessing the performance of a classifier. Before we give the evaluation criteria, the definition of the Confusion Matrix for the binary classifier is needed in Table 2.4. Kohavi and Provost (1998) proposed the Confusion Matrix containing the basic information about actual and predicted classification done by a classifier.

Table 2.3: Machine Learning Algorithms in AdaBoost Numeric Studies.

Name	Description	Parameters
SVMs	Standard single Support Vector Machines with RBF kernel	γ_{best} is obtained by grid searching in $2^{[-20:20]}$
Ada-DT	AdaBoost using Decision Tree as weak classifiers	
Ada-LSVMs	AdaBoost using linear kernel SVMs as weak classifiers	
Ada-RSVMs	AdaBoost using RBF kernel SVMs as weak classifiers	γ_{best} is obtained by grid searching in $2^{[-20:20]}$
En-Ada	Enhanced AdaBoost using Decision Tree as weak classifiers	
Re-Ada	Reinforced AdaBoost using Decision Tree as weak classifiers	

Table 2.4: Confusion Matrix.

		Prediction		Row Total
		Positive	Negative	
Truth	Positive	TP = True Positive rate	FN = False Negative rate	P
	Negative	FP = False Positive rate	TN = True Negative rate	N
Column Total		P'	N'	1

The common evaluation criteria based on the Confusion Matrix are:

- *Global Accuracy* = $TP + TN$;
- *Global Error Rate* = $1 - \text{Global Accuracy} = 1 - (TP + TN)$;

- *Sensitivity* = *Recall* = TP/P ;
- *Specificity* = TN/N ;
- *Precision* = TP/P' .

In the imbalanced data problem, our goal is to keep the global error low and improve the classification accuracy rate of \mathbf{S}^+ . *Sensitivity* (*Recall*) and *Precision* are more important for accessing the classification performance of \mathbf{S}^+ . *F - 1 Measure* (see Lewis and Gale (1994)) integrates *Sensitivity* and *Precision* as an average, defined as:

- $F - 1 \text{ Measure} = \frac{2 \times \textit{Precision} \times \textit{Sensitivity}}{\textit{Precision} + \textit{Sensitivity}}$.

F - 1 Measure represents a harmonic mean between *Sensitivity* and *Precision*. The harmonic mean of two numbers is closer to the small one. So a high *F - 1 Measure* can ensure *Sensitivity* and *Precision* are both high.

The Receiver Operating Characteristic (ROC) curve is created by plotting the true positive rate (TP) on the Y-axis against the false positive rate (FP) on the X-axis at various threshold settings. The area under the curve (*AUC*) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. *AUC* provides a single measure of a classifier's performance based on the ROC curve. Large *AUC* means the classifier performs well.

We choose five from all seven criteria above. *Global Accuracy* and *AUC* are used to evaluate the global classification ability. *Sensitivity* and *F - 1 Measure* are used as the evaluation criteria to compare the classification ability for \mathbf{S}^+ in different algorithms. *Specificity* is used to assess the classification ability of \mathbf{S}^- . All the results of the criteria in this section are kept rounded to four decimal places.

2.4.1 Simulation Studies Based on Gaussian Data

The simulated training dataset is:

$$\mathbf{X}_{ij} = \begin{pmatrix} X_{ij1} \\ X_{ij2} \end{pmatrix} \stackrel{iid}{\sim} \mathbf{N}_2(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (2.16)$$

where $i = -1, 1$ and $j = 1, \dots, n_i$.

$$\boldsymbol{\mu}_{-1} = \begin{pmatrix} 7 \\ 8 \end{pmatrix}, \boldsymbol{\mu}_1 = \begin{pmatrix} 13 \\ 15 \end{pmatrix}, \boldsymbol{\Sigma}_{-1} = \begin{pmatrix} 10 & 3 \\ 3 & 8 \end{pmatrix}, \boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}. \quad (2.17)$$

All the variables in $\mathbf{X}_{-1,j}$ and $\mathbf{X}_{1,j}$ have the class labels -1 (Majority) and 1 (Minority), respectively. We consider two cases based on $b = 10$ and $b = 5$.

2.4.1.1 $b = 10$

We set the training sample sizes of the positive and negative class to be $(n_{-1}, n_1) = (500, 50)$. The test dataset has the same distributions as the training data, but the test data size is $(n_{-1}^*, n_1^*) = (100, 10)$. The scatter plots of this simulated data are given in Figure 2.2.

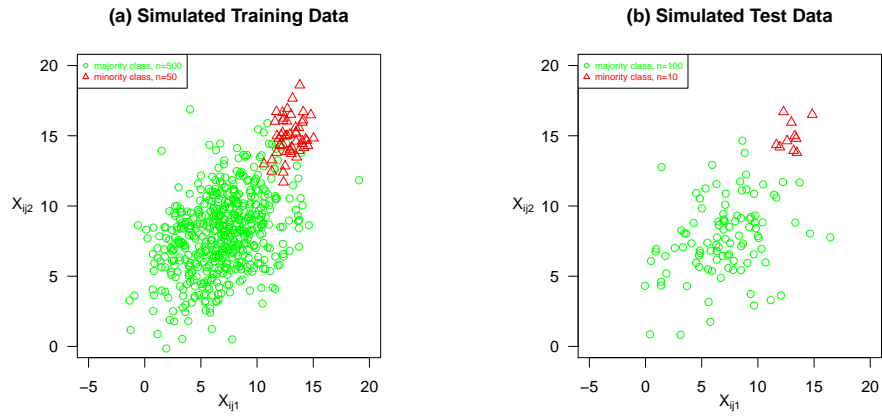


Figure 2.2: Simulated Gaussian Data with $b = 10$ in AdaBoost Numeric Studies.

For each simulation of the data, we conduct all the algorithms in Table 2.3 to compare the criteria of *Sensitivity*, *Specificity*, *F - 1 Measure*, *AUC*, and *Global Accuracy*. We repeat the experiments 100 times for every algorithm to reduce the randomness impact of the data simulation. Table 2.5 displays the mean values and standard deviation values (shown in brackets) of 100 repeated results.

Table 2.5: The Results of Simulated Gaussian Data in AdaBoost Numeric Studies ($b = 10$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.8931 (0.1051)	0.9903 (0.0109)	0.8968 (0.0761)	0.9417 (0.0527)	0.9815 (0.0135)
Ada-DT	0.8485 (0.1188)	0.9873 (0.0116)	0.8575 (0.0890)	0.9717 (0.0599)	0.9747 (0.0154)
Ada-LSVMs	0.8778 (0.1145)	0.9739 (0.0210)	0.8233 (0.0524)	0.9259 (0.0497)	0.9651 (0.0129)
Ada-RSVMs	0.9822 (0.1024)	0.9693 (0.0092)	0.8636 (0.0352)	0.9801 (0.0082)	0.9712 (0.0085)
En-Ada	0.9833 (0.0157)	0.8834 (0.0779)	0.9153 (0.0900)	0.9259 (0.0541)	0.9185 (0.0376)
Re-Ada	0.9840 (0.0574)	0.8902 (0.0685)	0.9159 (0.0767)	0.9262 (0.0576)	0.9213 (0.0568)

The Reinforced AdaBoost has the largest $F - 1$ Measure and *Sensitivity*. Notably, $F - 1$ Measure obtains an obvious improvement in both of our proposed algorithms compared to others. So our proposed algorithms outperform others regarding the classification performance of the positive class.

The SVMs with RBF kernel has the largest *Specificity* and *Global Accuracy* and the AdaBoost with RBF kernel SVMs has the largest *AUC*. The phenomenon of SVMs performing well makes sense since as a reliable classifier, SVMs can obtain an excellent result for the classification if we are interested in the global accuracy. Also,

the data we use is Gaussian data, all the algorithms related to SVMs with RBF kernel should perform well instinctively. Our proposed algorithms are both AdaBoost with the decision tree as the weak classifiers and they show better classification results in the positive class also obtain persuasive results from the global perspective.

The Reinforced AdaBoost has a similar result to the Enhanced AdaBoost. The reason is that $b = 10$ is a large value and the theoretical improvement in the Reinforced AdaBoost is not obvious compared to the Enhanced AdaBoost. We use the same simulated data but change $b = 5$ and repeat the experiment in the next case.

2.4.1.2 $b = 5$

Now the training data and test data are kept in the same distribution as the previous case but $(n_{-1}, n_1) = (500, 100)$ for the training data, and $(n_{-1}^*, n_1^*) = (100, 20)$ for the test data. The compared algorithms and criteria are the same as the last experiment. The results of 100 repeated experiments for the situation of $b = 5$ are shown in Table 2.6. All the results in Table 2.6 are similar to the results in Table 2.5. But the criteria differences between the Reinforced AdaBoost and Enhanced AdaBoost are more obvious in this study. This result supports the claim that the Reinforced AdaBoost outperforms the Enhanced AdaBoost when b is relatively small.

Table 2.6: The Results of Simulated Gaussian Data in AdaBoost Numeric Studies ($b = 5$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.9510 (0.0447)	0.9824 (0.0131)	0.9232 (0.0364)	0.9667 (0.0227)	0.9771 (0.0127)
Ada-DT	0.9005 (0.0695)	0.9805 (0.0141)	0.9011 (0.0507)	0.9405 (0.0356)	0.9672 (0.0167)
Ada-LSVMs	0.9663 (0.0425)	0.9621 (0.0267)	0.9003 (0.0520)	0.9642 (0.0291)	0.9628 (0.0207)
Ada-RSVMs	0.9736 (0.0168)	0.9672 (0.0181)	0.9226 (0.0396)	0.9804 (0.0127)	0.9716 (0.0155)
En-Ada	0.9810 (0.0534)	0.9111 (0.0567)	0.9341 (0.0213)	0.9402 (0.0867)	0.9402 (0.0789)
Re-Ada	0.9830 (0.0123)	0.9154 (0.0345)	0.9401 (0.0364)	0.9451 (0.0234)	0.9433 (0.0364)

2.4.2 Simulation Study Based on Uniform Data

The simulated training dataset is:

$$\mathbf{X}_{ij} = \begin{pmatrix} X_{ij1} \\ X_{ij2} \end{pmatrix} \stackrel{iid}{\sim} \text{Uniform}_2(\mathbf{a}_i, \mathbf{b}_i), \quad (2.18)$$

where $i = -1, 1$ and $j = 1, \dots, n_i$.

$$\mathbf{a}_{-1} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \mathbf{b}_{-1} = \begin{pmatrix} 10 \\ 12 \end{pmatrix}, \mathbf{a}_1 = \begin{pmatrix} 4 \\ 12 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 6 \\ 13 \end{pmatrix}. \quad (2.19)$$

All the variables in $\mathbf{X}_{-1,j}$ and $\mathbf{X}_{1,j}$ have the class labels -1 (Majority) and 1 (Minority), respectively. We set $b = 20$, the training sample sizes of the positive and negative class are $(n_{-1}, n_1) = (1000, 50)$.

The test dataset is:

$$\mathbf{X}_{ij}^* = \begin{pmatrix} X_{ij1}^* \\ X_{ij2}^* \end{pmatrix} \stackrel{iid}{\sim} \mathbf{Uniform}_2(\mathbf{a}_i, \mathbf{b}_i) + \boldsymbol{\epsilon}_{ij}, \quad (2.20)$$

where $i = -1, 1$, $j = 1, \dots, n_i^*$, and

$$\boldsymbol{\epsilon}_{ij} = \begin{pmatrix} \epsilon_{ij1} \\ \epsilon_{ij2} \end{pmatrix} \stackrel{iid}{\sim} \mathbf{N}_2(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i). \quad (2.21)$$

Where

$$\boldsymbol{\mu}_{-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\mu}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma}_{-1} = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}, \boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}. \quad (2.22)$$

The test data size is $(n_{-1}^*, n_1^*) = (200, 10)$. The scatter plots of this simulated data are given in Figure 2.3.

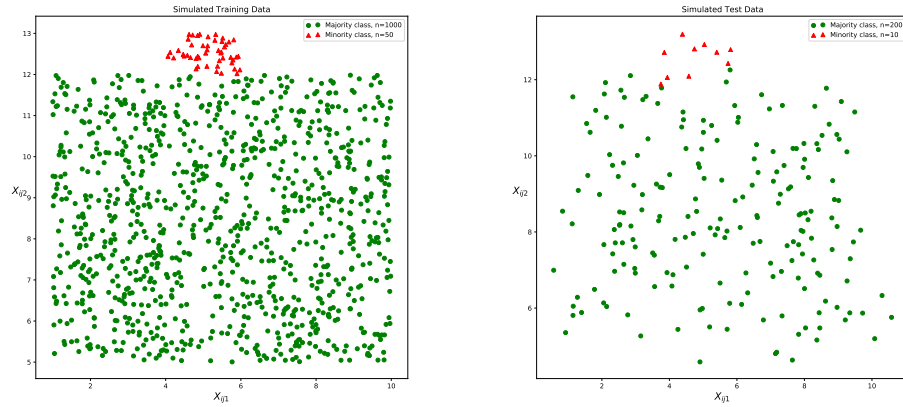


Figure 2.3: Simulated Uniform Data with $b = 20$ in AdaBoost Numeric Studies.

Except for the different sample sizes, the test data has an additional Gaussian error term compared to the training data. The training data points are located in two disjoint squares without overlap between the two classes, but the test data would have the overlap because of the error term. In Figure 1.1 (3), this situation is referred to as a Data Shift problem, in the sense that the distributions of the training data and test data are different. This situation often occurs when a classifier is trained based on standard and clean training data. However, the test data comes from the real practice that would have randomness and is not the same as the training data.

Table 2.7: The Results of Simulated Uniform Data in AdaBoost Numeric Studies ($b = 20$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.9198 (0.0906)	0.9941 (0.0053)	0.9027 (0.0623)	0.9570 (0.0449)	0.9906 (0.0060)
Ada-DT	0.9030 (0.0888)	0.9829 (0.0082)	0.8068 (0.0743)	0.9429 (0.0444)	0.9791 (0.0087)
Ada-LSVMs	0.9594 (0.0827)	0.9369 (0.0397)	0.6309 (0.1448)	0.9482 (0.0345)	0.9380 (0.0357)
Ada-RSVMs	0.9386 (0.0824)	0.9913 (0.0068)	0.8901 (0.0687)	0.9650 (0.0410)	0.9888 (0.0073)
En-Ada	0.9801 (0.0431)	0.9256 (0.0234)	0.9404 (0.0245)	0.9446 (0.0345)	0.9424 (0.0253)
Re-Ada	0.9812 (0.0634)	0.9353 (0.0353)	0.9465 (0.0245)	0.9364 (0.0465)	0.9467 (0.0253)

Table 2.7 contains the classification results for this simulated Uniform data. The results are still similar to Table 2.5 and Table 2.6, but the $F - 1$ Measure obtains a more significant improvement in the Reinforced AdaBoost and the Enhanced AdaBoost compared to other algorithms. Our proposed AdaBoost algorithms improve the classification ability for positive class apparently without losing the global classification ability.

2.4.3 Real Data Studies

Four real binary datasets are studied and their imbalanced index b are 15.13, 11.59, 5.14, and 3.36. Since all the datasets are imbalanced, each dataset is randomly split in half as the training part and half as the test part to avoid the situation that there are no or too few numbers of positive samples in the training or test data.

2.4.3.1 Seismic-bumps data study

The Seismic-bumps data with $b = 15.13$ is obtained from the University of California, Irvine, Machine Learning Repository. This dataset is about the seismic hazard in mining activity. Mining activity is always connected with the occurrence of dangers, which are called mining hazards. Seismic hazard is a special and dangerous case in mining hazards. The Seismic-bumps dataset is intensely imbalanced with only 170 samples of the positive class among a total of 2584 samples and it contains 18 explanatory variables including seismic hazard assessment obtained by different methods, seismic energy recorded by different types of equipment, the number of seismic bumps in different energy range and so on. The hazardous state is the negative class defined as that there is a high energy seismic bump occurring in the next shift, which is indicated by $y = -1$. The non-hazardous state is the positive class defined as that there is no high energy seismic bump occurring in the next shift, which is indicated by $y = 1$.

Table 2.8 displays the Seismic-bumps data experiment. It shows the great performance of our proposed algorithms for the classification of this dataset. Except for the *specificity*, all evaluation criteria of the Reinforced AdaBoost and the Enhanced AdaBoost perform better than other algorithms, especially for the $F - 1$ Measure.

Another interesting fact is the results based on the Enhanced AdaBoost and the Reinforced AdaBoost are the same. Because $b = 15.13$ is a considerable number in this dataset and γ_t should be relatively small in this high-dimensional data, the improvement part in the Reinforced AdaBoost compared to the Enhanced AdaBoost does not work.

2.4.3.2 Glass-2 data study

The Glass-2 dataset is obtained from the Knowledge Extraction Based on Evolutionary Learning (KEEL) Dataset Repository. This dataset is originally from the USA Forensic Science Service and intensely imbalanced with only 17 samples of the positive class among a total of 214 samples, which means $b = 11.59$. The Glass-2 dataset has eight explanatory variables, including the refractive index, the different chemical elements content and so on. The positive class is labeled as the glass is non-float processed and it is indicated by $y = 1$. The negative class is defined as the glass is made by other processed methods and it is indicated by $y = -1$.

Table 2.9 displays the Glass-2 data experiment. It shows similar results as the Seismic-bumps data results. Except for the *specificity*, all evaluation measures of our proposed algorithms perform better than other algorithms, especially for the $F - 1$ Measure. Compared to the Enhanced AdaBoost, the Reinforced AdaBoost has a slight improvement.

Table 2.8: The Results of Seismic-bumps Data in AdaBoost Numeric Studies ($b = 15.13$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.0000	1.0000	0.0000	0.5	0.9320
Ada-DT	0.1047	0.9873	0.1636	0.5460	0.9273
Ada-LSVMs	0.0000	0.9992	0.0000	0.4996	0.9312
Ada-RSVMs	0.6279	0.6675	0.2030	0.6477	0.6648
En-Ada	0.9211	0.9755	0.9601	0.9101	0.9533
Re-Ada	0.9211	0.9755	0.9601	0.9101	0.9533

Table 2.9: The Results of Glass-2 Data in AdaBoost Numeric Studies ($b = 11.59$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.0000	1.0000	0.0000	0.5	0.8972
Ada-DT	0.0909	0.9896	0.1538	0.5402	0.8972
Ada-LSVMs	0.0000	1.0000	0.0000	0.5000	0.8970
Ada-RSVMs	0.1818	0.9063	0.1818	0.5440	0.8318
En-Ada	0.8962	0.8327	0.8554	0.8948	0.8617
Re-Ada	0.9056	0.9087	0.9074	0.9047	0.9072

2.4.3.3 New-thyroid-1 data study

The New-thyroid-1 dataset is also obtained from the KEEL Dataset Repository. This dataset is original from the Garavan Institute in Sydney, Australia. It includes the

information of thyroid patients and was rearranged by KEEL to be an imbalanced binary dataset. The New-thyroid-1 dataset is imbalanced with 35 positive samples among a total of 215 samples, which indicates the imbalanced index is $b = 5.14$. This dataset contains 5 explanatory variables, including the levels of different hormones like Thyroxin, Triiodothyronine and so on. The positive class is labeled as Hyperthyroidism, which is indicated by $y = 1$. The negative class is defined as non-Hyperthyroidism, which is indicated by $y = -1$.

Table 2.10 displays the results of the New-thyroid-1 data experiment. It shows the satisfactory performance of our proposed algorithms compared to others for this data classification problem. All the evaluation criteria of the Reinforced AdaBoost and the Enhanced AdaBoost perform better than other algorithms. Because $b = 5.143$ is a relatively small number in this dataset, the improvement part in the Reinforced AdaBoost compared to the Enhanced AdaBoost works.

Table 2.10: The Results of New-thyroid-1 Data in AdaBoost Numeric Studies ($b = 5.14$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.2632	1.0000	0.4167	0.6316	0.8704
Ada-DT	0.7895	0.9888	0.8571	0.8891	0.9537
Ada-LSVMs	0.8421	0.9888	0.8889	0.9154	0.9630
Ada-RSVMs	0.8421	0.9663	0.8421	0.9042	0.9444
En-Ada	0.9881	0.9495	0.9673	0.9888	0.9680
Re-Ada	0.9888	1.000	0.9944	0.9944	0.9943

2.4.3.4 Ecoli-1 data study

The Ecoli-1 dataset is also obtained from the KEEL Dataset Repository. This dataset initially comes from the Institute of Molecular and Cellular Biology at Osaka University. It includes the information of the cellular localization sites of proteins and is rearranged by KEEL as an imbalanced binary dataset. The Ecoli-1 dataset is imbalanced with 77 positive samples among a total of 336 samples, which indicates the imbalanced index is $b = 3.36$. This dataset contains seven explanatory variables, including the signal sequence recognition scores by different methods, the presence of charge on N-terminus and so on. The positive class is labeled as the inner membrane without a signal sequence, which is indicated by $y = 1$. The negative class is defined as other situations, which is indicated by $y = -1$.

Table 2.11 displays the results of the Ecoli-1 data experiment. It shows the Reinforced AdaBoost has the largest $F - 1$ Measure and Global Accuracy. Compared to the Enhanced AdaBoost, the relatively small value of $b = 3.36$ makes the improvement part in Reinforced AdaBoost play a significant role. The differences between our two proposed algorithms are more obvious in this case.

Table 2.11: The Results of Ecoli-1 Data in AdaBoost Numeric Studies ($b = 3.36$).

Algorithms	Evaluation Measures				
	Sensitivity	Specificity	F-1 Measure	AUC	Global Accuracy
SVMs	0.8889	0.9242	0.8205	0.9066	0.9167
Ada-DT	0.8611	0.9091	0.7848	0.8851	0.8988
Ada-LSVMs	0.0000	1.0000	0.0000	0.5000	0.7857
Ada-RSVMs	0.9444	0.8636	0.7727	0.9040	0.8810
En-Ada	0.9322	0.9437	0.9383	0.9280	0.9379
Re-Ada	0.9205	0.9706	0.9457	0.8371	0.9441

Based on the numeric studies, our two proposed algorithms outperform the traditional ones in imbalanced data classification. With large imbalanced index b , our proposed algorithms show excellent performance. The difference between the Enhanced AdaBoost and the Reinforced AdaBoost is more obvious in the case of the small value of b and they would have the same performance in a large b case such as the Seismic-bumps data study.

2.5 Conclusion Comments

Class imbalanced problem is a critical issue in many fields and raises considerable hardship for classification. To address the challenge of the imbalanced class problem, the Enhanced AdaBoost and Reinforced AdaBoost are proposed in this chapter to improve the Adaboost algorithm. The innovation point is an adjustment of the weighted vote parameters of weaker classifiers α^t , which includes the global error rate

and the positive class accuracy rate. In addition, our algorithms consider the imbalanced index b , to improve the performance of classification in the imbalanced class problem.

Numerical studies of two kinds of simulated datasets and four real datasets compare the proposed algorithms and four other traditional algorithms. Our algorithms outperform in the positive class, especially regarding $F - 1$ Measure, and do not lose the global accuracy rate. When b is large, the Enhanced AdaBoost is a powerful algorithm for the imbalanced data classification. But if b is relatively small, the Reinforced AdaBoost is recommended to be used for the imbalanced data problem to obtain a better result.

Chapter 3

Fully Bayesian Analysis of the Relevance Vector Machine Classification for Imbalanced Data

3.1 Introduction

In statistics, Relevance Vector Machine (RVM) is an algorithm that uses a Bayesian model to obtain parsimonious solutions for regression and probabilistic classification. The RVM has an identical functional form to the Support Vector Machine (SVM), but provides probabilistic classification. Compared to SVM, the Bayesian formulation of the RVM avoids the set of free parameters of the SVM. RVM uses an Expectation Maximization (EM)-like learning method and is therefore not a pure Bayesian model. This chapter studies the original RVM model. Two proposed Bayesian RVM models are given to complete the RVM framework.

3.1.1 Support Vector Machine with Kernel Functions

In classification, we are given a set of input data $\mathbf{S}^{train} = \{(\mathbf{x}_1^{train}, y_1^{train}), (\mathbf{x}_2^{train}, y_2^{train}), \dots, (\mathbf{x}_n^{train}, y_n^{train})\}$ along with the corresponding class label, $y_i^{train} \in \{-1, 1\}$. From the training data, we wish to learn a model of the dependency of the class label on the inputs with the objective of making accurate predictions for the unseen values of \mathbf{x} . A very successful approach to the classification is the Support Vector Machine (SVM). For the linearly separable data, SVM constructs an optimal separating hyperplane as the classification borderline, which can obtain the maximum distance between two classes for a binary dataset. When we do not have a linearly separable set of training data, the Kernel trick comes handy. The idea is mapping the non-linear separable dataset into a higher-dimensional space where we can find a hyperplane that can separate the samples. If we use a mapping function that maps the data into a higher-dimensional space, the decision rule of SVM will depend on the dot products of the mapping function for different samples. The kernel function is employed here to reduce the complexity of finding the mapping function and defines the inner product in the transformed space. Vapnik (1998) and Scholkopf et al. (1999) proposed the output of SVM for an arbitrary data point \mathbf{x}_0 can be expressed as a weighted summation of the form

$$f(\mathbf{x}_0; \mathbf{w}) = \sum_{i=1}^n w_i k(\mathbf{x}_0, \mathbf{x}_i^{train}) + w_0, \quad (3.1)$$

where $k(\cdot, \cdot)$ is the Kernel function, w_0 and w_i are weight parameters, $i = 1, 2, \dots, n$. Note that \mathbf{x}_0 can be a training data point or a test data point. The Radial Basis

Function (RBF) Gaussian kernel, namely

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\gamma^2}\right),$$

is used throughout this chapter. RBF Gaussian Kernel is the most popular Kernel in Statistics and Machine Learning fields. The SVM output for the training data \mathbf{S}^{train} can be expressed as the matrix form

$$f(\mathbf{x}^{train}; \mathbf{w}) = \mathbf{K}^{train} \mathbf{w}, \quad (3.2)$$

where $\mathbf{w} = (w_0, w_1, \dots, w_n)^T$ is the weight parameter and

$$\begin{aligned} \mathbf{K}^{train} &= (K_1^{train}, K_2^{train}, \dots, K_n^{train})^T \\ &= \begin{pmatrix} 1 & k(\mathbf{x}_1^{train}, \mathbf{x}_1^{train}) & k(\mathbf{x}_1^{train}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_1^{train}, \mathbf{x}_n^{train}) \\ 1 & k(\mathbf{x}_2^{train}, \mathbf{x}_1^{train}) & k(\mathbf{x}_2^{train}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_2^{train}, \mathbf{x}_n^{train}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & k(\mathbf{x}_n^{train}, \mathbf{x}_1^{train}) & k(\mathbf{x}_n^{train}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_n^{train}, \mathbf{x}_n^{train}) \end{pmatrix} \end{aligned} \quad (3.3)$$

Note that given the training data \mathbf{S}^{train} , the kernel matrix \mathbf{K}^{train} is fixed. The classification goal is to obtain $sign(f(\mathbf{x}^{train}; \hat{\mathbf{w}})) = \mathbf{y}^{train}$, where $\hat{\mathbf{w}}$ is the proper estimate of \mathbf{w} , $sign(z) = 1$ if $z \geq 0$, and $sign(z) = -1$ if $z < 0$.

3.1.2 Adaptive Rejection Sampling Method

Sampling plays an important role in statistics. Sampling from the conventional distributions can be done directly by statistics software like R , but it is hard to do the sampling from the unconventional distributions. Robert & Casella (2004) proposed the Rejective Sampling method to conduct the sampling of unconventional distributions. It samples from a proposed conventional distribution and sets a ratio to decide the acceptance or rejection of this sampling value. But the Rejection Sampling method needs an upper boundary to restrict the proposed conventional distribution and people do not have a certain approach to determine this boundary. The original idea of the Adaptive Rejection Sampling (ARS) method was proposed by Gilks & Wild (1992). It can determine the certain upper boundary of the unconventional distributions and has a high acceptance rate for the sampling process. For distributions whose probability density functions are log-concave, the Adaptive Rejection Sampling (ARS) method is powerful and efficient.

3.1.2.1 Rejection Sampling method

The Rejection Sampling method is a typical Monte Carlo Sampling method. When the aim distribution $X \sim p_X(x)$ is not suitable for direct sampling, the Rejection Sampling method employs a proposal distribution $Y \sim g_Y(y)$, which can produce the sampling values quickly. The basic idea is to sample a random value y' from the proposal distribution, then accept y' as the sample of aim distribution $p_X(x)$ with the probability of $p_X(y')/(M g_Y(y'))$, where $1 < M < \infty$ is a constant.

Algorithm 3.1. The Rejection Sampling Method

Input. The sample size N and the aim distribution $p_X(x)$.

0. Determine the proposal distribution $g_Y(y)$ and constant M , let $i = 1$;

while $i \leq N$ **do:**

1. Sample $u \sim U(0, 1)$, $y_i \sim g_Y(y)$;

2. If $u < p_X(y_i)/(M g_Y(y_i))$ then $x_i = y_i$; else repeat Steps **1** and **2**;

3. $i = i + 1$;

Output. $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ are the sample values.

Although the Rejection Sampling method works, it would produce inaccurate results and the process is inefficient sometimes. First, if the aim distribution $p_X(x)$ has peak value in some intervals, the Rejection Sampling method may result in the inclusion of samples that should not have been accepted. Also, when the dimension of the aim distribution increases, the ratio of $p_X(y_i)/g_Y(y_i)$ convergence to 0 with N increasing. This would result in that a useful sample is rejected before it is produced. The most difficult thing is to find the proper proposal distribution $g_Y(y)$ and the bounded constant M .

3.1.2.2 Adaptive Rejection Sampling Method

For a better sampling performance in practice, we need a proposal distribution closer to the aim one. Gilks & Wild (1992) proposed the Adaptive Rejection Sampling method idea. It exercises a series of envelope functions to do the sampling. If one

sample value is rejected and it will be included to construct a more compact envelope function. First, we give the definition of concavity and convexity of functions.

Definition 3.1. *If $f(x)$ is continuous on $[a, b]$ and the second derivative exists.*

- (1) *When $f''(x) > 0$ on (a, b) , $f(x)$ is convex on $[a, b]$;*
- (2) *When $f''(x) < 0$ on (a, b) , $f(x)$ is concave on $[a, b]$.*

A necessary assumption of the ARS method is that the aim distribution is log-concave. If the aim distribution function is $p_X(x)$ defined on $\mathbb{D} \subseteq \mathbb{R}$, based on the Definition 3.1, let $V_X(x) = -\log(p_X(x))$ and $V_X''(x) > 0$ always holds on \mathbb{D} . ARS method needs a serial support points $s_1 < s_2 < \dots < s_m$ to construct the envelope function. The more support points there are, the higher acceptable rate the sampling process will have at the cost of efficiency. In Figure 3.1, let $w_k(x)$ be the tangential function of $V_X(x)$ at support point s_k :

$$w_k(x) = V_X'(s_k)(x - s_k) + V_X(s_k), \quad (3.4)$$

where $k = 1, 2, \dots, m$. We obtain m tangential functions based on the support points.

$$W_n(x) = \max\{w_1(x), w_2(x), \dots, w_m(x)\}. \quad (3.5)$$

Because $V_X(x)$ is the convex function on \mathbb{D} , and $w_k(x)$ are the tangential functions at s_k , where $k = 1, 2, \dots, m$. So $W_n(x) \leq V_X(x)$. Figure 3.2 shows the $W_n(x)$ based on two support points. After transformation, we have a envelope function

$$\exp(-W_n(x)) \geq \exp(-V_X(x)) = p_X(x). \quad (3.6)$$

Then a piecewise proposal function is obtained based on $\exp(-W_n(x))$:

$$\pi_n(x) = c_n \exp(-W_n(x)), \quad (3.7)$$

where $c_n = (\int_{\mathbb{D}} \exp(-W_n(x)) dx)^{-1}$ is the regularization constant. The basic idea of ARS method is to first sample the random values u from $U(0, 1)$, x' from $\pi_n(x)$. If $u < \frac{p_X(x')}{\exp(-W_n(x'))}$, we accept x' as the sample value from $p_X(x)$. Otherwise, we add x' into the support points set S_n to obtain $S_{n+1} = S_n \cup x'$, which will construct a more compact $W_{n+1}(x)$. Repeat this step until we have enough acceptable samples.

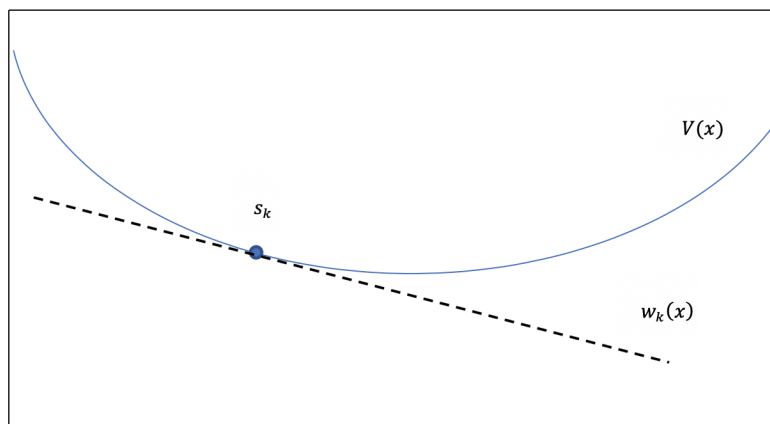


Figure 3.1: The Tangential Function $w_k(x)$ at s_k in ARS Method.

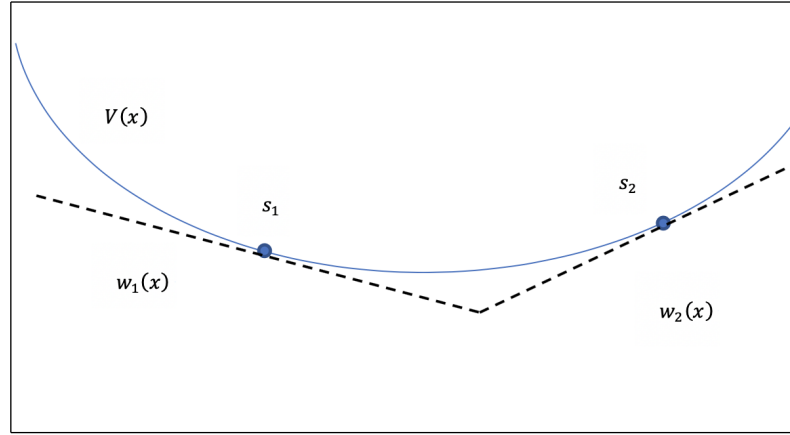


Figure 3.2: $w_n(x)$ Based on Two Support Points in ARS Method.

Algorithm 3.2. The Adaptive Rejection Sampling Method

- Input.** The sample size N , the aim distribution $p_X(x)$.
- 0.** Let $i = 1$ and determine the support points set S_i ;
- while** $i \leq N$ **do:**
- 1.** $V_X(x) = -\log(p_X(x))$ and construct the tangential functions of $V_X(x)$ based on the points in S_i ;
 - 2.** Sample $u \sim U(0, 1)$, $x' \sim \pi_n(x) \propto \exp(-W_n(x))$;
 - 3.** If $u < \frac{p_X(x')}{\exp(-W_n(x'))}$, $x_i = x'$ and $S_{i+1} = S_i$; Else, $S_i = S_i \cup x'$ and return to Step **1.**
- Output.** $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ are the sample values.
-

Thereafter, several improved ARS methods were proposed. Gilks (1995) proposed the MABS method, which combines the Metropolis-Hastings and ARS methods. But

this approach produces a Markov chain, which makes the samples are related to each other. Gorur & Teh (2009) proposed a new ARS method which can also solve log-convex distribution sampling. It divides the distribution function into several sections based on the concavity and convexity then conduct the sampling in every single section. Zhang (2017) summarized all the existing ARS methods and published the *AdapSamp* package in *R*. In this project, we use *AdapSamp* :: *rARS* function in *R* to conduct the Adaptive Rejection Sampling method.

3.2 RVM Classification

Relevance Vector Machine (RVM) is a Bayesian treatment for the output of the Support Vector Machine (SVM). This project only focuses on the RVM classification. It applies the Bernoulli distribution to the output of SVM in (3.1) and constructs the probability density function $p(\mathbf{y}|\mathbf{x})$ for the classification problems. The logistic link function is a continuous probability distribution. Its cumulative distribution function is the logistic link function. The logistic distribution is defined as

$$g_{logis}(t; \mu, s) = \frac{e^{-(t-\mu)/s}}{s(1 + e^{-(t-\mu)/s})^2}. \quad (3.8)$$

The logistic distribution receives its name from its cumulative distribution function, which is an instance of the family of logistic functions. The cumulative distribution function of the logistic distribution is also a scaled version of the hyperbolic tangent, which is the CDF of standard logistic distribution:

$$G_{logis}(t; \mu = 0, s = 1) = \frac{1}{1 + e^{-t}}. \quad (3.9)$$

The logistic sigmoid link function is used to map $f(\mathbf{x}; \mathbf{w})$ into $[0, 1]$. The likelihood of the training data set is

$$\begin{aligned}
p(\mathbf{y}^{train}|\mathbf{w}) &= \prod_{i=1}^n G_{logis}(f(\mathbf{x}_i^{train}; \mathbf{w}))^{\frac{1+y_i^{train}}{2}} [1 - G_{logis}(f(\mathbf{x}_i^{train}; \mathbf{w}))]^{\frac{1-y_i^{train}}{2}} \\
&= \prod_{i=1}^n \left(\frac{1}{1 + \exp(-K_i^{train}\mathbf{w})} \right)^{\frac{1+y_i^{train}}{2}} \times \\
&\quad \left(\frac{\exp(-K_i^{train}\mathbf{w})}{1 + \exp(-K_i^{train}\mathbf{w})} \right)^{\frac{1-y_i^{train}}{2}}, \tag{3.10}
\end{aligned}$$

where K_i^{train} and \mathbf{w} are defined in (3.2) and (3.3). RVM classification introduced a zero-mean Gaussian prior distribution over \mathbf{w} , namely

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{s=0}^n \mathcal{N}(w_s|0, \alpha_s^{-1}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{A}^{-1}), \tag{3.11}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_n)^T$, $\mathbf{A} = \text{diag}(\alpha_0, \alpha_1, \dots, \alpha_n)$, α_s is the hyperparameter associated with weight w_s , and $s = 0, 1, 2, \dots, n$. This prior helps to obtain the sparsity constraint. Compared with SVM, RVM classification has fewer relevant vectors because of the sparsity prior. The Bayesian model provides a posterior distribution for \mathbf{w} as

$$p(\mathbf{w}|\mathbf{y}^{train}, \boldsymbol{\alpha}) = \frac{p(\mathbf{y}^{train}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})}{\int p(\mathbf{y}^{train}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w}} = \frac{s(\mathbf{w})}{p(\mathbf{y}^{train}|\boldsymbol{\alpha})}, \tag{3.12}$$

where $s(\mathbf{w}) = p(\mathbf{y}^{train}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})$, which implies that

$$p(\mathbf{w}|\mathbf{y}^{train}, \boldsymbol{\alpha}) \propto s(\mathbf{w}). \tag{3.13}$$

The limitations of SVM are solved by RVM in the Bayesian framework. The original RVM classification obtained $\hat{\mathbf{w}}$, which is the estimation of \mathbf{w} , by maximizing $s(\mathbf{w})$. The classification function for the training data \mathbf{S}_{train} is

$$y_*^{train} = \text{sign} \left(\frac{1}{1 + \exp(-\mathbf{K}^{train} \hat{\mathbf{w}})} - \frac{1}{2} \right), \quad (3.14)$$

where \mathbf{K}^{train} is defined in (3.3). A test data can be defined as $\mathbf{S}^{test} = \{(\mathbf{x}_1^{test}, y_1^{test}), (\mathbf{x}_2^{test}, y_2^{test}), \dots, (\mathbf{x}_m^{test}, y_m^{test})\}$, where $\mathbf{x}_j^{test} \in \mathbf{X} \subseteq \mathbf{R}^l$, \mathbf{X} is in the same vector space as the training data. The response $y_j^{test} \in \{-1, 1\}$ indicates two classes, $j = 1, \dots, m$. In the imbalanced data problem, we define $\mathbf{S}_+^{test} = \{(\mathbf{x}_j^{test}, y_j^{test}) \in \mathbf{S}^{test} : y_j^{test} = 1, j = 1, \dots, m\}$ and $\mathbf{S}_-^{test} = \{(\mathbf{x}_j^{test}, y_j^{test}) \in \mathbf{S}^{test} : y_j^{test} = -1, j = 1, \dots, m\}$. The classification function for a test data \mathbf{S}_{test} is

$$y_*^{test} = \text{sign} \left(\frac{1}{1 + \exp(-\mathbf{K}^{test} \hat{\mathbf{w}})} - \frac{1}{2} \right), \quad (3.15)$$

where

$$\begin{aligned} \mathbf{K}^{test} &= (K_1^{test}, K_2^{test}, \dots, K_m^{test})^T \\ &= \begin{pmatrix} 1 & k(\mathbf{x}_1^{test}, \mathbf{x}_1^{train}) & k(\mathbf{x}_1^{test}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_1^{test}, \mathbf{x}_n^{train}) \\ 1 & k(\mathbf{x}_2^{test}, \mathbf{x}_1^{train}) & k(\mathbf{x}_2^{test}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_2^{test}, \mathbf{x}_n^{train}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & k(\mathbf{x}_m^{test}, \mathbf{x}_1^{train}) & k(\mathbf{x}_m^{test}, \mathbf{x}_2^{train}) & \dots & k(\mathbf{x}_m^{test}, \mathbf{x}_n^{train}) \end{pmatrix}. \end{aligned} \quad (3.16)$$

Eight criteria listed in Table 3.1 are used to evaluate the performance of algorithms

in the RVM research.

Table 3.1: The Criteria for Classification Evaluation in RVM Studies.

Training Data Global Accuracy Rate	$r_g^{train} = \frac{ y^{train}=y_*^{train} }{n}$
Training Data Positive Class Accuracy Rate	$r_p^{train} = \frac{ y^{train}=y_*^{train} \& y^{train}=1 }{n_p}$
Same Size Test Data Global Accuracy Rate	$r_g^{test} = \frac{ y^{test}=y_*^{test} }{n}$
Same Size Test Data Positive Class Accuracy Rate	$r_p^{test} = \frac{ y^{test}=y_*^{test} \& y^{test}=1 }{n_p}$
Smaller Size Test Data Global Accuracy Rate	$r_g^{stest} = \frac{ y^{stest}=y_*^{stest} }{n^s}$
Smaller Size Test Data Positive Class Accuracy Rate	$r_p^{stest} = \frac{ y^{stest}=y_*^{stest} \& y^{stest}=1 }{n_p^s}$
Larger Size Test Data Global Accuracy Rate	$r_g^{ltest} = \frac{ y^{ltest}=y_*^{ltest} }{n^l}$
Larger Size Test Data Positive Class Accuracy Rate	$r_p^{ltest} = \frac{ y^{ltest}=y_*^{ltest} \& y^{ltest}=1 }{n_p^l}$

The calculations of r_g^{test} and r_p^{test} use the same-sized test data as the training data, $n^{train} = n^{test} = n, n_p^{train} = n_p^{test} = n_p$. Smaller-sized and larger-sized test data are used for the calculations of $(r_g^{stest}, r_p^{stest})$ and $(r_g^{ltest}, r_p^{ltest})$, which means $n^s < n < n^l, n_p^s < n_p < n_p^l$. The simulation data studies use all these eight criteria. The real data studies only apply $r_g^{train}, r_p^{train}, r_g^{test}$, and r_p^{test} because it is hard to obtain different-sized real test data. All the test data sets in this paper keep the same imbalance index b as the training data, namely

$$\frac{|\mathbf{S}_-^{test}|}{|\mathbf{S}_+^{test}|} = \frac{|\mathbf{S}_-^{train}|}{|\mathbf{S}_+^{train}|} = b.$$

The original RVM classification algorithm is stated as follows:

Algorithm 3.3. The Original RVM Classification Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize \mathbf{w} and $\boldsymbol{\alpha}$ to obtain the started values \mathbf{w}^1 and $\boldsymbol{\alpha}^1$, calculate

$$\begin{aligned} h &= \nabla_{\mathbf{w}} \log g(\mathbf{w}) = \boldsymbol{\Phi}^\top (\mathbf{y} - [\sigma(\phi_1 \mathbf{w}), \dots, \sigma(\phi_n \mathbf{w})]^\top) - \mathbf{A}\mathbf{w}, \\ \mathbf{H} &= -\nabla \nabla_{\mathbf{w}} \log g(\mathbf{w}) = \boldsymbol{\Phi}^\top \mathbf{B}\boldsymbol{\Phi} + \mathbf{A}, \end{aligned} \quad (3.17)$$

where \mathbf{B} is a $(n+1) \times (n+1)$ diagonal matrix with diagonal elements $b_{ii} = \sigma(\phi_i \mathbf{w}) [1 - \sigma(\phi_i \mathbf{w})]$;

1. Fix $\boldsymbol{\alpha}$ and update \mathbf{w} with

$$\mathbf{w}^{t+1} = \mathbf{w}^t + (\mathbf{H})^{-1} h|_{\mathbf{w}=\mathbf{w}^t}; \quad (3.18)$$

2. Fix \mathbf{w} and update $\boldsymbol{\alpha}$ with

$$\alpha_s^{t+1} = \frac{\gamma_s^t}{\mathbf{w}_s^2}, \quad (3.19)$$

where $\gamma_s^t = 1 - \alpha_s^t \mathbf{H}_{ss}$, $s = 0, 1, 2, \dots, n$;

3. Repeat setps **1** and **2** until suitable convergence and obtain \mathbf{w}_0 , the mode of \mathbf{w} ;

Output. The final estimation of \mathbf{w} is $\mathbf{w}_{\text{MP}} = \mathbf{H}^{-1}\Phi^T \mathbf{B}\mathbf{y}|_{\mathbf{w}=\mathbf{w}_0}$.

Note that \mathbf{w}_{MP} , the maximum posterior of \mathbf{w} , is obtained by the Laplace's Method in Tipping (2000), which approximates a normal distribution with the mean value \mathbf{w}_0 to the posterior of \mathbf{w} . Xu et al. (2007), Pal & Foody (2012) and Braun et al. (2012) concluded that RVM is better than SVM in the fields of classification and regression. They also showed that the conduction speed of RVM is faster than SVM. But the original RVM classification algorithm still has several shortcomings:

1. Step 1 in **Algorithm 3.3** is to maximize the numerator $s(\mathbf{w})$ in (3.10); Step 2 is obtained by maximizing the denominator $p(\mathbf{y}^{\text{train}}|\boldsymbol{\alpha})$ in (3.10). This iteration process cannot ensure the maximum of posterior, $p(\mathbf{w}|\mathbf{y}^{\text{train}}, \boldsymbol{\alpha})$;

2. Laplace's Method is used to approximate $p(\mathbf{w}|\mathbf{y}^{\text{train}}, \boldsymbol{\alpha})$ as a normal distribution with the mean value \mathbf{w}_0 . Although Bishop and Tipping (2000) indicated that the posterior of \mathbf{w} is approximately normally distributed, the Laplace's method is still not stable under a strong normal distribution assumption;

3. The suitable convergence criterion is cryptic. An original RVM classification convergence study is stated as follows:

The simulated training dataset follows (2.16) and (2.17). Choose $N_+ = N_- = 3$, Figure 3.3 indicates this simulated training data.

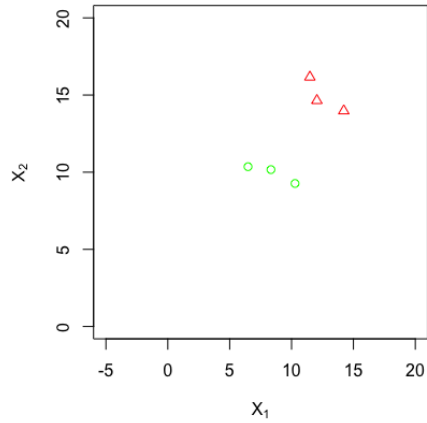


Figure 3.3: Simulated Data for Original RVM Classification ($N_+ = N_- = 3$).

Run the **Algorithm 3.3** 50000 iterations and check the parameter convergence in Figure 3.4. The plots take out 5000 burn-in and the red lines indicate 0. Although the classification accuracy rate is 100% in this case, we cannot determine that the parameter obtains a suitable convergence.

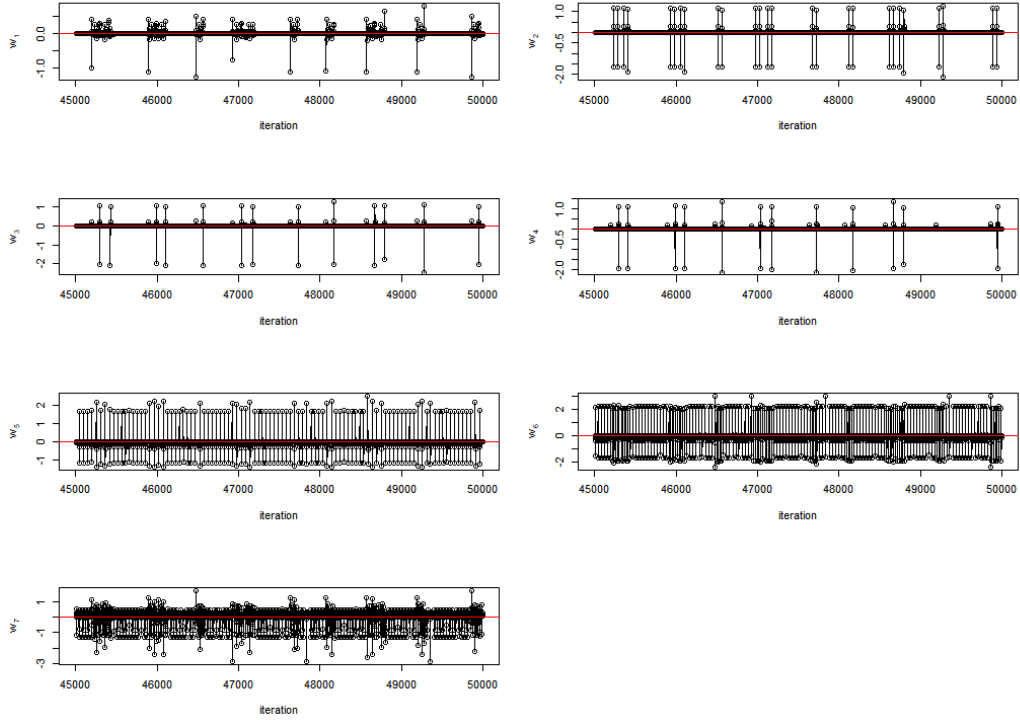


Figure 3.4: Convergence Plot of \mathbf{w} in Original RVM Classification Algorithm.

3.3 Generic Bayesian RVM Classification Algorithm

In this section, we propose a Generic Bayesian RVM classification method with the likelihood and prior for \mathbf{w} in (3.8) and (3.9). Our Generic RVM classification algorithm samples the parameter directly from the posterior instead of Newton's method in **Algorithm 3.3**. A Gamma hyperprior is called for each α_s and it yields a Student- t marginal prior for \mathbf{w} when $\boldsymbol{\alpha}$ is integrated out. The Gamma hyperprior is

$$(\alpha_s | a, b) \sim \text{Gamma}(\alpha_s | a, b), \quad (3.20)$$

the marginal prior for w_s is

$$p(w_s) = \int p(w_s|\alpha_s)p(\alpha_s)d\alpha_s = \frac{b^a\Gamma(a + \frac{1}{2})}{(2\pi)^{\frac{1}{2}}\Gamma(a)}(b + w_s^2)^{-(a+\frac{1}{2})}, \quad (3.21)$$

where $s = 0, 1, 2, \dots, n$. The marginal prior for the vector \mathbf{w} is a product of independent Student- t distributions in (3.21). Fokoué et al. (2011) indicated that this density induces more sparsity pressure than the LASSO prior. Recall that the posterior of \mathbf{w} is

$$p(\mathbf{w}|\mathbf{y}^{train}, \boldsymbol{\alpha}) \propto \exp(-\frac{1}{2}\mathbf{w}^T \mathbf{A}\mathbf{w}) \prod_{i=1}^n \left(\frac{1}{1 + \exp(-\phi_i^{train}\mathbf{w})} \right)^{\frac{1+y_i^{train}}{2}} \times \left(\frac{\exp(-\phi_i^{train}\mathbf{w})}{1 + \exp(-\phi_i^{train}\mathbf{w})} \right)^{\frac{1-y_i^{train}}{2}}, \quad (3.22)$$

where $s = 0, 1, 2, \dots, n$. Although this posterior has no closed-form, it has the desirable log-concave property from a computational perspective.

Theorem 3.1. *The conditional posterior of w_s , $p(w_s|w_k, \mathbf{y}^{train}, \alpha_s)$ is log-concave, where $k = 0, 1, \dots, s - 1, s + 1, \dots, n$.*

Proof. See Appendix B1. □

With this log-concavity, drawing samples from the posterior of w_s becomes possible

with the ARS method. The posterior of α_s is

$$\begin{aligned}
p(\alpha_s | w_s, a, b) &\propto \alpha_s^{\frac{1}{2}} \exp\left(-\frac{1}{2}\alpha_s w_s^2\right) \cdot \alpha_s^{a-1} \exp(-b\alpha_s) \\
&= \alpha_s^{a+\frac{1}{2}-1} \exp\left[-\left(b + \frac{1}{2}w_s^2\right)\alpha_s\right] \\
&\propto \text{Gamma}\left(a + \frac{1}{2}, b + \frac{1}{2}w_s^2\right).
\end{aligned} \tag{3.23}$$

The following pseudo-code is implemented to perform the Generic Bayesian RVM classification.

Algorithm 3.4. The Generic Bayesian RVM Classification Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize \mathbf{w} and $\boldsymbol{\alpha}$ to obtain the started values \mathbf{w}^t and $\boldsymbol{\alpha}^t$. Choose (a, b) , the number of burn-in B , and the number of iterations T ;

- 1.** Fix $\boldsymbol{\alpha}^t$, draw a new \mathbf{w}^{t+1} according to (3.22);
- 2.** Fix \mathbf{w}^{t+1} , draw a new $\boldsymbol{\alpha}^{t+1}$ according to (3.23);
- 3.** Repeat steps 1 and 2 until suitable convergence is obtained by T iterations;

Output. The final estimation of \mathbf{w} is $\hat{\mathbf{w}} = (T - B)^{-1} \sum_{t=B+1}^T \mathbf{w}^t$.

Algorithm 3.4 is a strict Bayesian method and conducted by the Gibbs sampling method, which could obtain more stable parameter estimates than the original RVM classification regarding parameter convergence. Although **Algorithm 3.4** builds a Bayesian framework for the RVM classification and ends up yielding a sparsity rep-

resentation, the complete freedom of α given to the parameters makes it difficult to find the unique solution because the number of parameters grows with the sample size. This is a typical case of the Neyman-Scott problem proposed in Neyman et al. (1948). Fokoué et al. (2011) indicated that in RVM’s context, the Neyman-Scott problem means that the prior of (3.11) makes the estimate of \mathbf{w} not consistent. A dimension reduction via the coefficient structure can solve this problem.

3.4 Fully Hierarchical Bayesian RVM Classification Algorithm

This section follows the hierarchical prior structure in Fokoué et al. (2011) but applied to RVM classification instead of the regression problem. One of the main contributions of Fokoué et al. (2011) is to add another layer random-coefficient structure for prior of α , which reduces the parameter dimensions. The dimensions reduction can give consistent estimates of \mathbf{w} . This Fully Bayesian method could relate α_s ’s with the coefficient parameter and enhance the inner connection of parameters. This section, compared with Fokoué et al. (2011), makes some improvements. Only n dimensions of the parameters were considered in Fokoué et al. (2011), the error term of the parameters, w_0 and α_0 , were ignored. This project considers all $n + 1$ dimensions in the parameters. In the numeric study part, Fokoué et al. (2011) specified all the hyperparameters and only sampled \mathbf{w} and α in the Gibbs sampling process. The numeric studies in this project run the full Gibbs sampling iterations, including all the parameters. Recall the prior for w_s is

$$p(w_s|\alpha_s) = \mathcal{N}(w_s|0, \alpha_s^{-1}). \tag{3.24}$$

Reparametrize $\boldsymbol{\alpha}$ as $\boldsymbol{\eta} = (\eta_0, \eta_1, \dots, \eta_n)$, where $\eta_s = \log(\alpha_s)$, and $s = 0, 1, 2, \dots, n$. Fokoué et al. (2011) defined the hyperprior for $\boldsymbol{\eta}$ is

$$\boldsymbol{\eta} \sim \mathcal{N}_{n+1}(\mu \mathbf{1}_{n+1}, \tau^2 \boldsymbol{\Sigma}_{n+1}), \quad (3.25)$$

where $\boldsymbol{\Sigma}_{n+1} = (1 - \rho) \mathbf{I}_{n+1} + \rho \mathbf{1}_{n+1} \mathbf{1}_{n+1}^\top$, \mathbf{I}_{n+1} is an identity matrix, and $\mathbf{1}_{n+1}$ is a vector with all values of 1. Note that ρ should remain in the interval of $(0, 1)$. The interpretation of ρ is to maintain the trade-off between absolute freedom of α_s 's when ρ is close to 0 and the total tightness of α_s 's when ρ is close to 1. τ^2 should be relatively large because sparsity is still an important goal in RVM classification. The value of ρ indicates the relative contribution of the joint effects between all the α_s 's, the value of τ^2 controls the magnitude of information in $\boldsymbol{\alpha}$. Based on their expected effect, we propose the constant priors for ρ and μ , a conjugate prior for τ^2 , namely

$$p(\rho) = \text{Uniform}(0, 1), \quad p(\mu) = \text{Uniform}(0, 1), \quad \text{and} \quad p(\tau^{-2}) = \text{Gamma}(c, d). \quad (3.26)$$

Since we only add a new layer to the prior, the Fully conditional posterior for \mathbf{w} remains unchanged as (3.22). For the joint posterior of $\boldsymbol{\alpha}$, we can reach it through its reparametrized version $\boldsymbol{\eta}$, $\boldsymbol{\eta} = \log(\boldsymbol{\alpha})$,

$$\begin{aligned} p(\boldsymbol{\eta} | \text{others}) &\propto p(\mathbf{w} | \boldsymbol{\alpha}(\boldsymbol{\eta})) p(\boldsymbol{\eta} | \mu, \rho, \tau^2) \\ &\propto \left(\prod_{s=1}^{n+1} \frac{1}{\sqrt{2\pi}} e^{\eta_s/2} \right) \exp \left(-\frac{1}{2} \sum_{s=1}^{n+1} e^{\eta_s} w_s^2 \right) \exp \left\{ -\frac{1}{2\tau^2(1-\rho)} \cdot \right. \\ &\quad \left. \sum_{s=1}^{n+1} (\eta_s - \mu)^2 + \frac{\rho}{2\tau^2(1-\rho)(1+n\rho)} \left[\sum_{s=1}^{n+1} (\eta_s - \mu) \right]^2 \right\}. \quad (3.27) \end{aligned}$$

It seems hard to draw samples for this posterior but it also has the desired log-concave property. The ARS method can be applied again.

Theorem 3.2. *The conditional posterior of η_s , $p(\eta_s | \text{others})$ is log-concave.*

Proof. See Appendix B2. □

The prior for ρ allows us to write

$$\begin{aligned}
p(\rho | \text{others}) &\propto p(\rho)p(\boldsymbol{\eta} | \mu, \rho, \tau^2) \\
&\propto \frac{1}{(1-\rho)^{\frac{n}{2}}(1+n\rho)^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2\tau^2(1-\rho)} \sum_{s=1}^{n+1} (\eta_s - \mu)^2 + \right. \\
&\quad \left. \frac{\rho}{2\tau^2(1-\rho)(1+n\rho)} \left[\sum_{s=1}^{n+1} (\eta_i - \mu) \right]^2 \right\}. \tag{3.28}
\end{aligned}$$

The method of Ratio of Uniforms is used to sample from this conditional posterior. See Appendix D for more details about the Ratio of Uniforms sampling method. The posterior of μ is

$$\begin{aligned}
p(\mu | \text{others}) &\propto p(\mu)p(\boldsymbol{\alpha} | \mu, \rho, \tau^2) \\
&\propto \exp \left\{ -\frac{1}{2\tau^2(1-\rho)} \sum_{s=1}^{n+1} (\eta_s - \mu)^2 + \frac{\rho}{2\tau^2(1-\rho)(1+n\rho)} \cdot \right. \\
&\quad \left. \left[\sum_{s=1}^{n+1} (\eta_i - \mu) \right]^2 \right\} \\
&\propto \exp \left\{ -\frac{n+1}{2\tau^2(1+n\rho)} \left(\mu - \frac{\sum_{s=1}^{n+1} \eta_s}{n+1} \right)^2 \right\} \\
&\propto \mathcal{N} \left(\frac{\sum_{s=1}^{n+1} \eta_s}{n+1}, \frac{\tau^2(1+n\rho)}{n+1} \right). \tag{3.29}
\end{aligned}$$

For τ^2 , we have

$$\begin{aligned}
p(\tau^{-2} | \text{others}) &\propto p(\tau^{-2})p(\boldsymbol{\eta} | \mu, \rho, \tau^2) \\
&\propto (\tau^{-2})^{c-1} \exp(-d\tau^{-2})(\tau^{-2})^{\frac{n+1}{2}} \exp\left(-\frac{1}{2\tau^2}(\boldsymbol{\eta} - \mu\mathbf{1}_{n+1})^\top \cdot \right. \\
&\quad \left. \boldsymbol{\Sigma}^{-1}(\boldsymbol{\eta} - \mu\mathbf{1}_{n+1})\right) \\
&\propto \text{Gam}\left(c + \frac{n+1}{2}, d + \frac{1}{2}\left\{\frac{1}{1-\rho}\sum_{s=1}^{n+1}(\eta_s - \mu)^2 - \right. \right. \\
&\quad \left. \left. \frac{\rho}{(1-\rho)(1+n\rho)}\left[\sum_{s=1}^{n+1}(\eta_s - \mu)\right]^2\right\}\right). \tag{3.30}
\end{aligned}$$

The samples of μ and τ^2 are easy to obtain from their special close-forms. Based on the above derivation of full conditional posteriors, we have an alternative algorithm:

Algorithm 3.5. Fully Hierarchical Bayesian RVM Classification Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize \mathbf{w} , $\boldsymbol{\alpha}$, μ , ρ and τ^2 to obtain the started values \mathbf{w}_t , $\boldsymbol{\alpha}_t$, μ_t , ρ_t and τ_t^2 . Choose (c, d) , the number of burn-in B , and the number of iterations T ;

- 1.** Fix other parameters and draw a new \mathbf{w}_{t+1} according to (3.22);
- 2.** Fix other parameters and draw a new $\boldsymbol{\alpha}_{t+1}$ according to (3.27);
- 3.** Fix other parameters and draw a new ρ_{t+1} according to (3.28);
- 4.** Fix other parameters and draw a new μ_{t+1} according to (3.29);
- 5.** Fix other parameters and draw a new τ_{t+1}^2 according to (3.30);

6. Repeat steps 1-5 until suitable convergence is obtained by T iterations;

Output. The final estimation of \mathbf{w} is $\hat{\mathbf{w}} = (T - B)^{-1} \sum_{t=B+1}^T \mathbf{w}_t$.

3.5 Numeric Studies

3.5.1 Simulation Data Studies

The simulated Gaussian datasets have the same distribution as (2.16), (2.17). We chose five kinds of sizes, $(n_p, n_n) = (30, 30), (15, 30), (12, 30), (6, 30), (3, 30)$, to illustrate the performance of different algorithms in different-sized data. $b = 1, 2, 2.5, 5, 10$ for these five cases and a larger b indicates a more imbalanced dataset. Following Figure 3.5 shows the training data sets.

We run the **Algorithm 3.4** and **3.5** with $T = 10000, B = 500, (a, b) = (1, 1/999)$, and $(c, d) = (1, 1/999)$. The evaluation criteria come from Table 3.1. For both **Algorithm 3.4** and **3.5**, we repeat the experiments 100 times for every case in Figure 3.5 to reduce the randomness impact of the data simulation. Table 3.2–3.6 display the mean values and standard deviation values (shown in the bracket) of 100 repeated results, the larger accuracy rate is indicated by boldface.

The simulation studies show that for the balanced data and mildly imbalanced data as $b = 1, 2, 2.5$, **Algorithms 3.4** and **3.5** perform similarly. But for the seriously imbalanced data as $b = 5, 10$, **Algorithm 3.5** outperforms **3.4** significantly. Especially for $b = 10$, the accuracy rates for the positive class are almost zero under **Algorithm 3.4**, but **Algorithm 3.5** improves the classification performance in this

case.

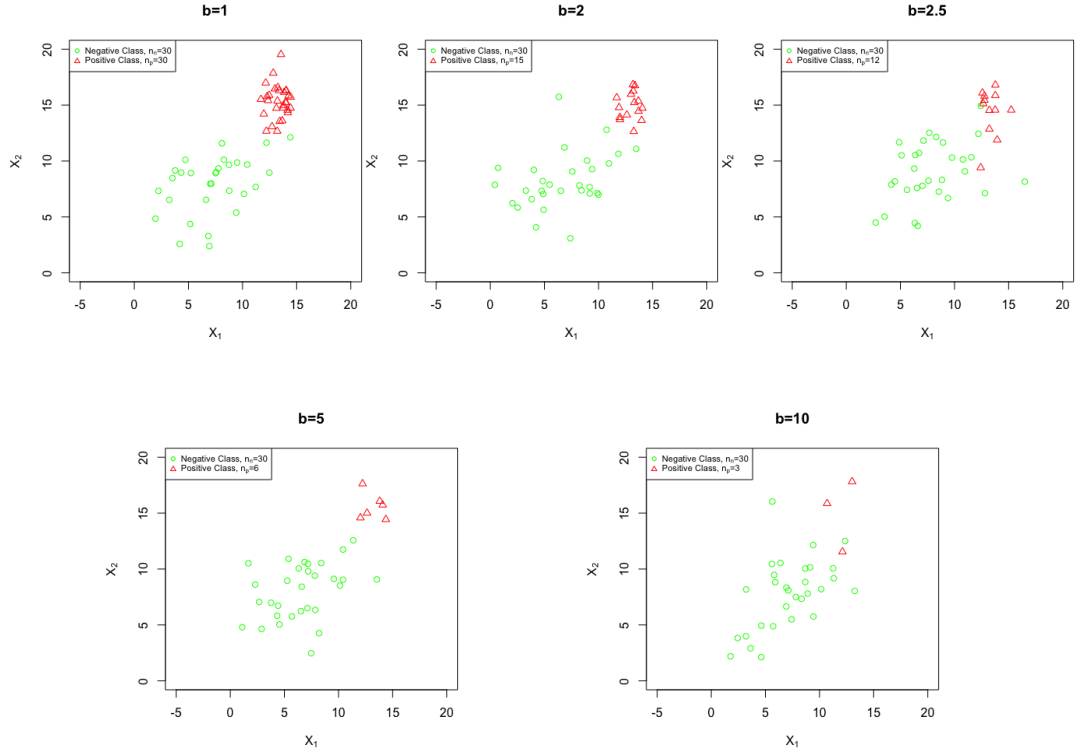


Figure 3.5: Simulated Gaussian Data for Bayesian RVM.

Table 3.2: The Results of Simulated Data in Bayesian RVM ($b = 1$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 3.4	0.9823 (0.0148)	0.9710 (0.0254)	0.9780 (0.0306)	0.9732 (0.0209)	0.9993 (0.0047)	0.9980 (0.0080)	0.9980 (0.0141)	0.9996 (0.0022)
Algorithm 3.5	0.9770 (0.0170)	0.9678 (0.0328)	0.9705 (0.0390)	0.9674 (0.0300)	0.9993 (0.0067)	1.0000 (0.0000)	0.9990 (0.0100)	0.9998 (0.0016)

^a ($n_n^{train} = 30, n_p^{train} = 30$), ^b ($n_n^{test} = 30, n_p^{test} = 30$), ^c ($n_n^{stest} = 10, n_p^{stest} = 10$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 90$)

Table 3.3: The Results of Simulated Data in Bayesian RVM ($b = 2$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 3.4	0.9796 (0.0257)	0.9791 (0.0235)	0.9773 (0.0418)	0.9757 (0.0147)	0.9680 (0.0690)	0.9693 (0.0542)	0.9760 (0.0870)	0.9698 (0.0378)
Algorithm 3.5	0.9760 (0.0236)	0.9822 (0.0214)	0.9740 (0.0443)	0.9808 (0.0138)	0.9707 (0.0616)	0.9767 (0.0477)	0.9700 (0.0823)	0.9798 (0.0322)

^a ($n_n^{train} = 30, n_p^{train} = 15$), ^b ($n_n^{test} = 30, n_p^{test} = 15$), ^c ($n_n^{stest} = 10, n_p^{stest} = 5$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 45$)

Table 3.4: The Results of Simulated Data in Bayesian RVM ($b = 2.5$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 3.4	0.9693 (0.0296)	0.9745 (0.0294)	0.9729 (0.0416)	0.9732 (0.0186)	0.9375 (0.1015)	0.9403 (0.0920)	0.9550 (0.1088)	0.9433 (0.0580)
Algorithm 3.4	0.9679 (0.0376)	0.9710 (0.0286)	0.9650 (0.0482)	0.9731 (0.0171)	0.9383 (0.1212)	0.9325 (0.0860)	0.9275 (0.1390)	0.9414 (0.0541)

^a ($n_n^{train} = 30, n_p^{train} = 12$), ^b ($n_n^{test} = 30, n_p^{test} = 12$), ^c ($n_n^{stest} = 10, n_p^{stest} = 4$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 36$)

Table 3.5: The Results of Simulated Data in Bayesian RVM ($b = 5$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 3.4	0.9600 (0.0213)	0.9567 (0.0222)	0.9133 (0.0229)	0.8500 (0.0118)	0.7062 (0.0464)	0.7467 (0.1770)	0.7600 (0.2141)	0.6688 (0.1331)
Algorithm 3.5	0.9614 (0.1675)	0.9568 (0.2042)	0.9348 (0.2041)	0.8933 (0.3171)	0.7101 (0.2334)	0.7811 (0.4003)	0.7805 (0.5152)	0.7600 (0.2289)

^a ($n_n^{train} = 30, n_p^{train} = 6$), ^b ($n_n^{test} = 30, n_p^{test} = 6$), ^c ($n_n^{stest} = 10, n_p^{stest} = 2$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 18$)

Table 3.6: The Results of Simulated Data in Bayesian RVM ($b = 10$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 3.4	0.9973 (0.2144)	0.9091 (0.0091)	0.9091 (0.0000)	0.9104 (0.1763)	0.0900 (0.0288)	0.0000 (0.0000)	0.0000 (0.0000)	0.0100 (0.0320)
Algorithm 3.5	0.9503 (0.0357)	0.9148 (0.0417)	0.9118 (0.0758)	0.9187 (0.0332)	0.5233 (0.3914)	0.1667 (0.2485)	0.2200 (0.4163)	0.1922 (0.2118)

^a ($n_n^{train} = 30, n_p^{train} = 3$), ^b ($n_n^{test} = 30, n_p^{test} = 3$), ^c ($n_n^{stest} = 10, n_p^{stest} = 1$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 9$)

3.5.2 Real Data Studies

Six binary real data sets are studied in this paper and the imbalance index b changes from 1.82 to 11.59. The datasets are obtained from the Knowledge Extraction based on Evolutionary Learning (KEEL) Dataset Repository (see Jesús Alcalá-Fdez et al. (2011)). KEEL is an open-source Java software tool used for data discovery tasks. It includes plenty of datasets that can be used for imbalanced data problem studies. The detailed descriptions of every dataset can be found on the website of KEEL. For every dataset, we randomly split the positive and negative classes, where half is the training part, the other half is the test part. **Algorithm 3.4** and **3.5** are applied to all the datasets. Table 3.7 lists some basic information of the datasets and the classification results of the four criteria. The real data studies show that **Algorithm 3.5** indeed outperforms **3.4**, especially for seriously imbalanced datasets when we are interested in the positive class.

Table 3.7: The Results of Real Datasets in Bayesian RVM.^a

Dataset	b	Dimension	Total Data Size	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
glass1	1.82	9	214	0.6449	0.6449	0.0000	0.0000
				0.6542	0.6449	0.0263	0.0000
iris0	2.00	4	150	1.0000	1.0000	1.0000	1.0000
				1.0000	1.0000	1.0000	1.0000
newthyroid1	5.14	5	215	0.8318	0.8333	0.0000	0.0000
				0.8999	0.8333	0.0588	0.0000
glass6	6.38	9	214	0.8679	0.8611	0.0000	0.0000
				0.8679	0.8611	0.0000	0.0000
ecoli0345	9.00	7	200	0.8600	0.9000	0.0000	0.0000
				0.9100	0.9000	0.1000	0.0000
glass2	11.59	9	214	0.9245	0.9167	0.0000	0.0000
				0.9379	0.9300	0.0133	0.0133

^a Results in the table are listed by **Algorithm 3.4** on the top, **Algorithm 3.5** on the bottom.

3.6 Conclusion Comments

Two Bayesian RVM classification algorithms are proposed in this chapter and they make two-fold contributions. The first Generic Bayesian RVM algorithm conducts a pure Bayesian RVM classification algorithm compared to the original RVM classification method and makes it possible to sample the weight parameter directly from the posterior. On the other hand, the Fully Hierarchical Bayesian algorithm follows the hyperprior structure in Fokoué et al. (2011) but is applied to the classification problem to improve the classification performance compared to the Generic one, especially in the imbalanced data problem. We have provided the theoretical justification of the

log-concavity for the conditional posterior of some parameters, which helps to set up a fast and stable sampling process. The simulated data studies use the data from the same distribution in Chapter 2 but with different imbalance indexes. The experiment results show the favorable performance of our two proposed algorithms and indicate that the Fully Hierarchical Bayesian RVM algorithm can classify the seriously imbalanced data more strongly than the Generic one. The real data studies explore six datasets and check the performance of our two proposed algorithms in practice. They both perform well and the Fully Hierarchical one indeed outperforms the Generic one in imbalanced data when we are more interested in the positive class.

Chapter 4

Fully Bayesian Analysis of the Relevance Vector Machine Classification with Probit Link Function

4.1 Introduction

The original RVM classification is hard to conduct in practice because the posterior of the weight parameter has no closed-form solution. The previous chapter shows an approach that addresses this issue by doing the sampling directly from the posterior based on the log-concave property. In this chapter, we propose the probit link function to form a new likelihood function in RVM instead of the logistic one in the original algorithm. Benefiting from a latent variable, this new likelihood function can lead a more concise posterior, which follows a multivariate normal distribution.

4.1.1 The Probit Link Functions

A probit model is a type of regression, where the dependent variable has two values and the independent variable is $(-\infty, +\infty)$. The purpose of the probit model is to estimate the probability that the observations with particular characteristics will fall into a specific category, so it is popular for the binary classification problem. The probit link function $G_{probit}(x)$ is used to map $f(\mathbf{x}; \mathbf{w})$ into $[0, 1]$. $G_{probit}(x)$ is defined as

$$G_{probit}(t) = \Phi(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz. \quad (4.1)$$

Following Figure 4.1 shows the logistic and probit link functions. The logistic one has slightly flatter tails. The probit curve approaches the axes more quickly than the logistic curve. In binary classification problems, they are the same in the application.

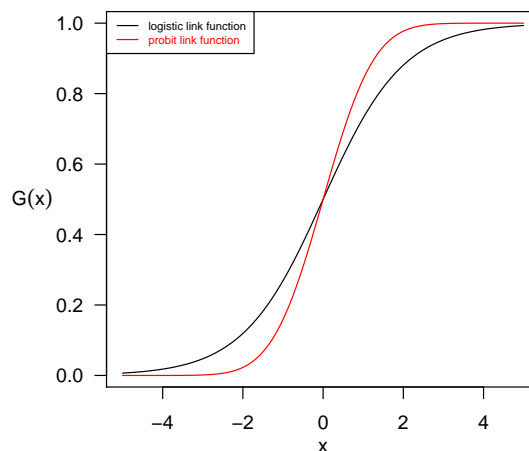


Figure 4.1: Logistic and Probit Link Functions.

4.2 Generic Bayesian PRVM Classification Algorithm

The Bernoulli probability of every data point is

$$p_i = R_i^{\frac{1+y_i^{train}}{2}} (1 - R_i)^{\frac{1-y_i^{train}}{2}}, \quad (4.2)$$

where $R_i = G_{probit}(K_i^{train} \mathbf{w})$, K_i^{train} and \mathbf{w} are defined in (3.2) and (3.3). The likelihood of the training data set is

$$\begin{aligned} P(\mathbf{y}^{train} | \mathbf{w}) &= \prod_{i=1}^n p_i \\ &= \prod_{i=1}^n R_i^{\frac{1+y_i^{train}}{2}} (1 - R_i)^{\frac{1-y_i^{train}}{2}} \\ &= \prod_{i=1}^n G_{probit}(K_i^{train} \mathbf{w})^{\frac{1+y_i^{train}}{2}} (1 - G_{probit}(K_i^{train} \mathbf{w}))^{\frac{1-y_i^{train}}{2}}. \end{aligned} \quad (4.3)$$

Following Albert and Chib (1995), we bring in a latent variable $\boldsymbol{\mu}$ for the probit link function:

$$\begin{aligned} \boldsymbol{\mu} &= (\mu_1, \mu_2, \dots, \mu_n)^T, \\ \mu_i &\stackrel{indep.}{\sim} \mathcal{N}(K_i \mathbf{w}, 1). \end{aligned} \quad (4.4)$$

We can show

$$\begin{aligned}
R_i &= G_{probit}(K_i \mathbf{w}) \\
&= \int_{-\infty}^{K_i \mathbf{w}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz \\
&= \int_0^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\mu_i - K_i \mathbf{w})^2\right) d\mu_i \\
&= P(\mu_i > 0).
\end{aligned} \tag{4.5}$$

Note that $1 - R_i = P(\mu_i \leq 0)$. Rewrite the likelihood function, including the latent variable

$$P(\mathbf{y}^{train} | \mathbf{w}, \mu) = \prod_{i=1}^n (\mathbf{1}_{(\mu_i > 0)})^{\frac{1+y_i^{train}}{2}} (\mathbf{1}_{(\mu_i \leq 0)})^{\frac{1-y_i^{train}}{2}} \phi(\mu_i - K_i \mathbf{w}). \tag{4.6}$$

Follow the original RVM classification to introduce a zero-mean Gaussian prior distribution over \mathbf{w} , namely

$$p(\mathbf{w} | \boldsymbol{\alpha}) = \prod_{s=0}^n \mathcal{N}(w_s | 0, \alpha_s^{-1}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{A}^{-1}),$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_n)^T$, $\mathbf{A} = \text{diag}(\alpha_0, \alpha_1, \dots, \alpha_n)$, α_s is the hyperparameter associated with weight w_s , and $s = 0, 1, 2, \dots, n$. A Gamma hyperprior is called for each α_s . The Gamma hyperprior is

$$(\alpha_s | a, b) \sim \text{Gamma}(\alpha_s | a, b).$$

The full posterior is

$$p(\mathbf{w}, \mathbf{y}^{train}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = (2\pi)^{-\frac{n+1}{2}} |\mathbf{A}|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w}\right) \prod_{i=1}^n (\mathbf{1}_{(\mu_i > 0)})^{\frac{1+y_i^{train}}{2}} (\mathbf{1}_{(\mu_i \leq 0)})^{\frac{1-y_i^{train}}{2}} \cdot \phi(\mu_i - K_i \mathbf{w}).$$

The conditional posterior of \mathbf{w} is

$$p(\mathbf{w} | \mathbf{y}^{train}, \boldsymbol{\alpha}, \boldsymbol{\mu}) \propto \exp\left(-\frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w}\right) \prod_{i=1}^n \phi(\mu_i - K_i \mathbf{w}). \quad (4.7)$$

Lemma 4.1. *The conditional posterior of \mathbf{w} follows a multivariate normal distribution*

$$p(\mathbf{w} | \mathbf{y}^{train}, \boldsymbol{\alpha}, \boldsymbol{\mu}) \propto \mathcal{N}(\hat{\mathbf{w}}, \mathbf{V}^{-1}), \quad (4.8)$$

where $\mathbf{V} = \mathbf{A} + \mathbf{K}^T \mathbf{K}$, $\hat{\mathbf{w}} = \mathbf{V}^{-1} \mathbf{K}^T \boldsymbol{\mu}$.

Proof. See Appendix C1. □

The conditional posterior of α_s is

$$\begin{aligned} p(\alpha_s | w_s, a, b) &\propto \alpha_s^{\frac{1}{2}} \exp\left(-\frac{1}{2} \alpha_s w_s^2\right) \cdot \alpha_s^{a-1} \exp(-b \alpha_s) \\ &= \alpha_s^{a+\frac{1}{2}-1} \exp\left[-\left(b + \frac{1}{2} w_s^2\right) \alpha_s\right] \\ &\propto \text{Gamma}\left(a + \frac{1}{2}, b + \frac{1}{2} w_s^2\right). \end{aligned} \quad (4.9)$$

The conditional posterior of μ_i is

$$\begin{aligned}
 p(\mu_i | \mathbf{w}, \mathbf{y}^{train}, \boldsymbol{\alpha}) &\propto (\mathbf{1}_{(\mu_i > 0)})^{\frac{1+y_i^{train}}{2}} (\mathbf{1}_{(\mu_i \leq 0)})^{\frac{1-y_i^{train}}{2}} \phi(\mu_i - K_i \mathbf{w}) \\
 &= \begin{cases} g_{R_i}(u_i) \mathbf{1}_{(u_i > 0)} & \text{if } y_i = 1 \\ g_{R_i}(u_i) \mathbf{1}_{(u_i \leq 0)} & \text{if } y_i = -1 \end{cases}, \quad (4.10)
 \end{aligned}$$

where $g_{R_i} = \phi(\mu_i - K_i \mathbf{w})$, $i = 1, 2, \dots, n$. This conditional posterior is a truncated normal distribution and the sampling process of it may be inefficient. When $K_i \mathbf{w}$ is far away from 0, one sampling process of (4.10) for $y_i = 1$ or $y_i = -1$ would have a low acceptable rate. Figure 4.2 shows a situation where we sample some negative values from a normal distribution with mean value of $K_i \mathbf{w} = 2$. Only the shaded area can satisfy our requirement and the sampling acceptable rate is low.

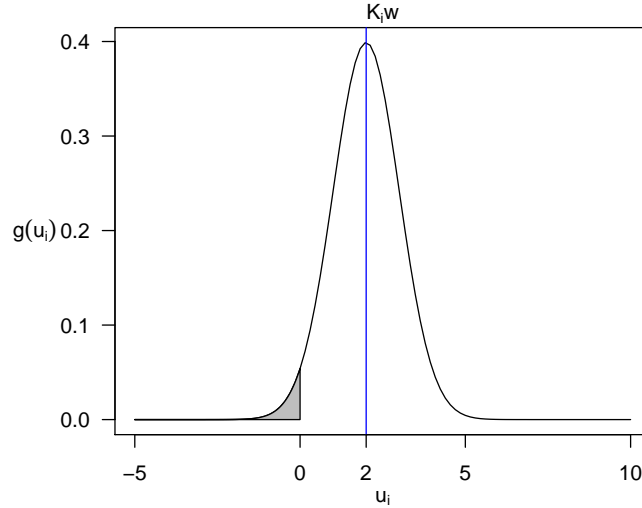


Figure 4.2: Sampling From a Truncated Normal Distribution.

C. R. Ren (2001) proposed following Lemma 4.2 with an 100% acceptable rate sampling method for this conditional posterior.

Lemma 4.2. *Let u be a uniform random variable on $(0, 1)$, the variable Z follows a normal distribution $Z \sim N(b, 1)$.*

(1) $D = b + \Phi^{-1}(u\Phi(-b))$ and $Z|Z \leq 0$ have the same distribution.

(2) $D = b + \Phi^{-1}(1 - u\Phi(b))$ and $Z|Z > 0$ have the same distribution.

Proof. See Appendix C2. □

For $i = 1, 2, \dots, n$, we can do the sampling of μ_i as follows based on Lemma 4.2:

(1) Sample

$$u \sim \text{uniform}(0, 1); \tag{4.11}$$

(2) If $y_i = 1$, calculate

$$\mu_i = K_i \mathbf{w} + \Phi^{-1}(1 - u\Phi(K_i \mathbf{w})); \tag{4.12}$$

(3) If $y_i = -1$, calculate

$$\mu_i = K_i \mathbf{w} + \Phi^{-1}(u\Phi(-K_i \mathbf{w})). \tag{4.13}$$

$\Phi(\cdot)$ is the the cumulative distribution function (CDF) of the standard normal distribution. The following pseudo-code is implemented to perform this Generic PRVM classification.

Algorithm 4.1. The Generic PRVM Classification Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize \mathbf{w} , $\boldsymbol{\alpha}$, and $\boldsymbol{\mu}$ to obtain the started values \mathbf{w}^t , $\boldsymbol{\alpha}^t$ and $\boldsymbol{\mu}^t$. Choose (a, b) , the number of burn-in B , and the number of iterations T ;

1. Fix $\boldsymbol{\alpha}^t$ and $\boldsymbol{\mu}^t$, draw a new \mathbf{w}^{t+1} according to (4.8);

2. Fix \mathbf{w}^{t+1} and $\boldsymbol{\mu}^t$, draw a new $\boldsymbol{\alpha}^{t+1}$ according to (4.9);

3. Fix $\boldsymbol{\alpha}^{t+1}$ and \mathbf{w}^{t+1} , draw a new $\boldsymbol{\mu}^{t+1}$ according to (4.11), 4.12, 4.13;

3. Repeat steps 1, 2 and 3 until suitable convergence is obtained by T iterations;

Output. The final estimation of \mathbf{w} is $\hat{\mathbf{w}} = (T - B)^{-1} \sum_{t=B+1}^T \mathbf{w}^t$.

Algorithm 4.1 is a more succinct and efficient algorithm compared with the original RVM and the Bayesian RVM. The conditional posteriors all have closed-form solutions and the sampling process is simple. For the imbalanced data problem, we again apply the hierarchical prior structure in Fokoué et al. (2011) to PRVM.

4.3 Fully Hierarchical Bayesian PRVM Classification Algorithm

Because we only change the hyperprior structure, the conditional posterior of \mathbf{w} keeps the same as (4.8). Other parameters' posteriors are the same as **Algorithm 3.5**. Based on the above derivations of full conditional posteriors, we have an alternative

algorithm:

Algorithm 4.2. The Fully Hierarchical Bayesian PRVM Classification Algorithm

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize \mathbf{w} , $\boldsymbol{\alpha}$, μ , m , ρ and τ^2 to obtain the started values \mathbf{w}_t , $\boldsymbol{\alpha}_t$, μ_t , m_t , ρ_t and τ_t^2 . Choose (a, b) , (c, d) , the number of burn-in B , and the number of iterations T ;

- 1.** Fix other parameters and draw a new \mathbf{w}_{t+1} according to (4.8);
- 2.** Fix other parameters and draw a new $\boldsymbol{\alpha}_{t+1}$ according to (3.27);
- 3.** Fix other parameters and draw a new μ_{t+1} according to (3.27);
- 4.** Fix other parameters and draw a new m_{t+1} according to (3.29);
- 5.** Fix other parameters and draw a new ρ_{t+1} according to (3.28);
- 6.** Fix other parameters and draw a new τ_{t+1}^2 according to (3.30);
- 7.** Repeat steps 1 – 5 until suitable convergence is obtained by T iterations;

Output. The final estimation of \mathbf{w} is $\hat{\mathbf{w}} = (T - B)^{-1} \sum_{t=B+1}^T \mathbf{w}_t$.

4.4 Numeric Studies

We use the same simulated datasets and real datasets as chapter 3 in this PRVM chapter for the numeric studies.

4.4.1 Simulation Data Studies

We run the **Algorithm 4.1** and **4.2** with $T = 5000$, $B = 500$, $(a, b) = (1, 1/999)$, and $(c, d) = (1, 1/999)$ on the simulated datasets. The evaluation criteria come from Table 3.1. For both **Algorithm 4.1** and **4.2**, we repeat the experiments 100 times for every case in Figure 3.5. Table 4.1–4.5 display the mean values and standard deviation values (shown in the bracket) of 100 repeated results, the larger accuracy rate is indicated by boldface. These simulation studies show that PRVM has a similar performance as the Bayesian RVM. For the seriously imbalanced data as $b = 5, 10$. Two algorithms of PRVM outperform the Bayesian RVM. Especially for the case of $b = 10$, the PRVM is significantly better than the Bayesian RVM.

Table 4.1: The Results of Simulated Data in PRVM ($b = 1$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 4.1	0.9990 (0.0071)	0.9570 (0.0947)	0.9570 (0.0833)	0.9592 (0.0914)	0.9987 (0.0094)	0.9607 (0.0802)	0.9700 (0.0735)	0.9687 (0.0593)
Algorithm 4.2	0.9876 (0.1701)	0.9317 (0.0760)	0.9600 (0.0568)	0.9478 (0.0750)	0.9993 (0.0067)	0.9267 (0.1350)	0.9730 (0.1060)	0.9311 (0.1254)

^a ($n_n^{train} = 30, n_p^{train} = 30$), ^b ($n_n^{test} = 30, n_p^{test} = 30$), ^c ($n_n^{stest} = 10, n_p^{stest} = 10$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 90$)

Table 4.2: The Results of Simulated Data in PRVM ($b = 2$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 4.1	0.9802 (0.1507)	0.9724 (0.0322)	0.9707 (0.0408)	0.9719 (0.0192)	0.9711 (0.2340)	0.9640 (0.0854)	0.9640 (0.0875)	0.9636 (0.0527)
Algorithm 4.2	0.9816 (0.1251)	0.9574 (0.0348)	0.9778 (0.0328)	0.9843 (0.0412)	0.9698 (0.1456)	0.9667 (0.0603)	0.9667 (0.0778)	0.9481 (0.0783)

^a ($n_n^{train} = 30, n_p^{train} = 15$), ^b ($n_n^{test} = 30, n_p^{test} = 15$), ^c ($n_n^{stest} = 10, n_p^{stest} = 5$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 45$)

Table 4.3: The Results of Simulated Data in PRVM ($b = 2.5$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 4.1	0.9637 (0.0091)	0.9700 (0.0350)	0.9586 (0.0542)	0.9637 (0.0330)	0.9401 (0.0905)	0.9400 (0.1038)	0.9050 (0.1667)	0.9367 (0.0934)
Algorithm 4.2	0.9651 (0.1967)	0.9619 (0.0526)	0.9464 (0.0512)	0.9619 (0.0302)	0.9411 (0.0975)	0.9458 (0.0729)	0.9375 (0.1111)	0.9431 (0.0833)

^a ($n_n^{train} = 30, n_p^{train} = 12$), ^b ($n_n^{test} = 30, n_p^{test} = 12$), ^c ($n_n^{stest} = 10, n_p^{stest} = 4$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 36$)

Table 4.4: The Results of Simulated Data in PRVM ($b = 5$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 4.1	0.9700 (0.1139)	0.9700 (0.0317)	0.9167 (0.0512)	0.9600 (0.0241)	0.7805 (0.1140)	0.8633 (0.1639)	0.8500 (0.2901)	0.8411 (0.1406)
Algorithm 4.2	0.9700 (0.0213)	0.9781 (0.0316)	0.9750 (0.0547)	0.9606 (0.0223)	0.7997 (0.1033)	0.8837 (0.1631)	0.9000 (0.2052)	0.8500 (0.1362)

^a ($n_n^{train} = 30, n_p^{train} = 6$), ^b ($n_n^{test} = 30, n_p^{test} = 6$), ^c ($n_n^{stest} = 10, n_p^{stest} = 2$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 18$)

Table 4.5: The Results of Simulated Data in PRVM ($b = 10$).

	$r_g^{train^a}$	$r_g^{test^b}$	$r_g^{stest^c}$	$r_g^{ltest^d}$	$r_p^{train^a}$	$r_p^{test^b}$	$r_p^{stest^c}$	$r_p^{ltest^d}$
Algorithm 4.1	0.9771 (0.1622)	0.9654 (0.0274)	0.9691 (0.0472)	0.9631 (0.0000)	0.7333 (0.0479)	0.7267 (0.2988)	0.7600 (0.4314)	0.7467 (0.2480)
Algorithm 4.2	0.9802 (0.0236)	0.9757 (0.0395)	0.9818 (0.0407)	0.9797 (0.0124)	0.7434 (0.2214)	0.8842 (0.1856)	0.9113 (0.1431)	0.8444 (0.1685)

^a ($n_n^{train} = 30, n_p^{train} = 3$), ^b ($n_n^{test} = 30, n_p^{test} = 3$), ^c ($n_n^{stest} = 10, n_p^{stest} = 1$), ^d ($n_n^{ltest} = 90, n_p^{ltest} = 9$)

4.4.2 Real Data Studies

Algorithm 4.1 and **4.2** are applied to all the six real datasets. The real data studies results in Table 4.6 show that **Algorithm 4.2** indeed outperforms **4.1**, especially for seriously imbalanced datasets. The performances of Bayesian RVM and PRVM are similar.

4.5 Comparison Between the Bayesian RVM and PRVM

From the numeric studies, we can conclude that the Bayesian RVM and PRVM models are similar for classification accuracy results. The only theoretic difference between them is the link functions for the likelihood. The Bayesian RVM uses the logistic link function, but the PRVM employs the probit one. It is still worth discussing more comparisons between them.

Table 4.6: The Results of Real Datasets in PRVM.^a

Dataset	b	Dimension	Total Data Size	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
glass1	1.82	9	214	0.6721 0.6890	0.6511 0.6669	0.0000 0.0214	0.0000 0.0108
iris0	2.00	4	150	1.0000 1.0000	1.0000 1.0000	1.0000 1.0000	1.0000 1.0000
newthyroid1	5.14	5	215	0.8710 0.9233	0.8541 0.8509	0.0000 0.0591	0.0032 0.0000
glass6	6.38	9	214	0.8901 0.8901	0.8611 0.8611	0.0000 0.0000	0.0000 0.0000
ecoli0345	9.00	7	200	0.8596 0.9274	0.9015 0.9016	0.0000 0.2977	0.0000 0.0000
glass2	11.59	9	214	0.9241 0.9400	0.9221 0.9452	0.0000 0.0159	0.0000 0.0231

^a Results in the table are listed by **Algorithm 4.1** on the top, **Algorithm 4.2** on the bottom.

4.5.1 Elapsed Programming Time

The Bayesian RVM model needs the ARS method to conduct the sampling process. The model has to conduct one sampling iteration for every dimension of \mathbf{w} . Also, we do not have a strategy to determine the suitable support values for the ARS sampling process, so the ARS method could be inefficient. PRVM can sample the whole vector \mathbf{w} directly from its posterior since it follows a multivariate normal distribution. Table 4.7 lists the elapsed programming time for these two models. We conduct every experiment on the simulated Gaussian datasets with 5000 iterations. Repeat 100 times and calculate the mean and standard deviation values listed in the Table 4.7.

The PRVM is significantly more efficient than the Bayesian RVM.

Table 4.7: Epalsed Programming Time ^a of Bayesian RVM and PRVM Models.^b

Data Size	30-30	30-15	30-12	30-6	30-3
Generic Bayesian RVM	87758.2994	40447.4915	38074.4325	14604.5066	16167.2713
	6419.9621	3565.7886	(2127.4783)	(1049.5469)	(3893.8587)
Generic PRVM ^c	236.5257 (1.5314)	45.1316 (1.5714)	42.4868 (8.0555)	41.1134 (3.1998)	40.6392 (6.0307)

^a Time is measured in seconds.

^b R Programmings are conducted on Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz.

^c AdapSamp::rARS is used for log-concave posterior sampling.

4.5.2 Model Selection

Many parameter estimation problems adopt likelihood function as the objective function. When enough training data are available, the accuracy of models can be improved continuously. However, at the cost of model complexity increases, it also brings up a widespread problem in machine learning, namely overfitting. Therefore, the problem of model selection seeks an optimal balance between the complexity of the model and the ability of the model describing the dataset. Many information criteria have been proposed to avoid the overfitting problem by adding a penalty for model complexity. We introduce two commonly used model selection methods: Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).

AIC is a standard to measure the goodness of model fitting. Akaike proposed it in 1974. It is based on the concept of entropy and provides a standard to balance the complexity of the model estimation and the goodness of model fitting. Generally,

AIC is defined as:

$$AIC = 2k - 2\ln(\hat{L}), \quad (4.14)$$

where k is the number of model parameters, and \hat{L} is the maximum value of the likelihood function. It is common to choose the model with minimum AIC.

BIC is similar to AIC and it is also used for model selection. Schwarz proposed it in 1978. The penalty term of BIC is larger than AIC since BIC also considers the number of samples. When the sample size is large, it can effectively prevent the situation of the model's complexity being too high.

$$BIC = k \ln(n) - 2\ln(\hat{L}), \quad (4.15)$$

where k is the number of model parameters, n is the number of samples, and \hat{L} is the maximum value of the likelihood function. Given the same data, the two RVM models in this project, Bayesian RVM and PRVM, have the same k and n . So we only need to focus on \hat{L} to compare them in the cases of AIC and BIC. Choose the Gaussian simulated datasets defined in (2.16) and (2.17) and repeat the process of seeking maximum likelihood value 100 times for every simulated dataset in the Bayesian RVM and PRVM. Table 4.8 shows the mean and standard deviation results. In the case of \hat{L} , the Bayesian RVM and PRVM are similar. But the PRVM seems a little preferred than the Bayesian RVM.

Table 4.8: Maximum Likelihood Value of Bayesian RVM and PRVM Models.

Data Size	30-30	30-15	30-12	30-6	30-3
Bayesian RVM	0.9999928	0.9999928	0.9999928	0.999927	0.9999927
	(1.0120×10^{-6})	(1.0120×10^{-6})	(1.0120×10^{-6})	(1.8935×10^{-6})	(1.0593×10^{-6})
PRVM	0.9999986	0.9999986	0.9999986	0.9999985	0.9999986
	(4.9653×10^{-7})	(5.3061×10^{-7})	(5.3061×10^{-7})	(4.1018×10^{-7})	(5.1192×10^{-7})

4.6 Conclusion Comments

Two RVM with the probit link function (PRVM) classification algorithms are proposed in this chapter. The posterior of the weight parameter in the original RVM has no closed-form solution, so it is hard to conduct. The intricate likelihood is the reason for this. The original RVM uses the logistic link function to construct the likelihood function, which leads to all the difficulties in the algorithm. Benefiting from the probit link function, the posterior of the weight parameter in PRVM follows a multivariate normal distribution. PRVM is a more compact algorithm, and its programming speed is significantly faster than the Bayesian RVM, which is the algorithm we proposed in the last chapter. The Fully Hierarchical PRVM follows the hyperprior structure in Chapter 3 to improve the classification performance in the imbalanced data problem.

A study of the comparison between Bayesian RVM and PRVM is conducted. The numeric studies show that these two models have similar classification accuracy results. For the severely imbalanced data, PRVM is significantly better than the Bayesian RVM. Also, PRVM is more efficient than the Bayesian RVM in the case

of programming time. From the perspective of model selection, PRVM is a little preferred than the Bayesian RVM in the cases of AIC and BIC.

Chapter 5

Discussion and Future Research

There are several possibilities in the future research for these two approaches to solve the imbalanced data problem. There is a major way to extend the results in Chapter 2 to multiple classes problem. For Chapter 3 and 4, Bayes classifier is deserved to be studied because it is the optimal one in all possible classifiers. If we consider the misclassified cost as the prior, this Cost-sensitive Bayes classifier approach is an admissible direction to continue this project.

5.1 Discussion of Chapter 2

In practice, we often need to deal with the multi-classification data. The disassembling method is the most commonly used in the generalization from binary to multi-classifications: the multi-classification problem is divided into many binary problems, and a classifier is trained for each binary problem. During the testing process, the results of these classifiers are integrated to obtain the final prediction result. Sup-

pose the dataset has N categories. According to the splitting strategy, the extended methods are divided into the following three categories:

(1) One vs. One (OvO)

Training: Randomly choose two pairs from N categories, which produces $N(N+1)/2$ binary classification tasks. Each binary task can be solved by the binary classification algorithm;

Test: All $N(N+1)/2$ classifiers are applied to an unseen sample and the class with the highest number of predictions is predicted by the combined classifier.

(2) One vs. Rest (OvR)

Training: For every category, treat each of them as the positive class and the others as the negative class. This process produces N binary classification tasks;

Test: Applying all N classifiers to an unseen sample and predicting the label, which the corresponding classifier reports the highest confidence score.

(3) Many vs. Many (MvM)

OvO and OvR are two special cases of MvM. In MvM, several categories are taken as the positive class and the other ones are the negative class. The following is the most common MvM technology: Error Correcting Output Codes (ECOC):

Coding: Separate N categories to M divisions. Choose m divisions as the positive class, the other $M - m$ divisions are the negative class. Repeat such a method K times and it produces K classifier;

Decoding: Applying all K classifiers to an unseen sample and predicting the label that the corresponding classifier reports the highest confidence score.

For imbalanced data problems, we can extend our improved AdaBoost algorithms to the multi-classes problem based on the above three options.

5.2 Discussion of Chapter 3

In classification, the generative model and discriminative model are the two typical approaches. The generative model learns the joint probability $p(\mathbf{x}, y)$ based on the training dataset \mathcal{S}_{train} . It predicts the label y_*^0 for the unseen data point \mathbf{x}^0 by calculating $p(y_*^0|\mathbf{x}^0)$ based on the Bayes rule. The discriminative model estimates $p(y|\mathbf{x})$ directly and learns a map from the input \mathbf{x} to the class label y . Gaussian mixture model (GMM) is the most popular one in the generative model and it has been applied in many classification tasks (e.g., Hastie & Tibshirani (1996)). Most of the Kernel methods are discriminative models. They include SVM (Cortes & Vapnik (1995)), proximal SVM (PSVM) (Fung & Mangasarian (2001) and Rifkin (2002)), and RVM (Tipping (2001)). To compare the generative model and discriminative model in classification, we study the GMM and RVM methods. The reasons why we prefer the discriminative models to the generative models are discussed in this part.

5.2.1 Generative vs. Discriminative Models

Define Θ as the parameters that need to be determined in the model. The straightforward way of estimating Θ is through Maximum Likelihood Estimation (MLE). Let $\hat{p}(\mathbf{x}|y)$ is the estimation of $p(\mathbf{x}|y)$, MLE classifiers seek $\Theta_g = \arg \max_{\Theta} R_g(\Theta)$, where

$$R_g(\Theta) = \prod_{i=1}^n \hat{P}(\mathbf{x}_i|y_i).$$

Ng & Jordan (2002) defined this as the generative model. On the other hand, the discriminative model seeks $\Theta_d = \arg \max_{\Theta} R_d(\Theta)$, where

$$R_d(\Theta) = \prod_{i=1}^n \hat{P}(y_i | \mathbf{x}_i).$$

Note that

$$\begin{aligned} R_d(\Theta) &= \prod_{i=1}^n \frac{\hat{P}(\mathbf{x}_i | y_i) \hat{P}(y_i)}{\hat{P}(\mathbf{x}_i)} \\ &= \prod_{i=1}^n \left(1 + \frac{\sum_{y_j \neq y_i} \hat{P}(\mathbf{x}_i | y_j) \hat{P}(y_j)}{\hat{P}(\mathbf{x}_i | y_i) \hat{P}(y_i)} \right)^{-1} \end{aligned}$$

In the discriminative case, the model minimizes the likelihood of competing classes $y_j \neq y_i$. Note that sometimes a generative model is more appropriate when we have a confident estimation of $p(\mathbf{x}|y)$. But in most cases, the discriminative models perform better (see Nadas et al. (1988) and Rubinstein & Hastie (1997)).

5.2.2 The Gaussian Mixture Model

The mixture model is a probability model that can be used to represent K sub-distributions in the population distribution. In other words, the mixture model represents the probability distribution of observed data in a population, which is a mixed distribution composed of K sub-distributions. The mixture model does not require the observed data to provide information about the sub-distributions to calculate the probability. When x is univariate, a Gaussian probability density function is

$$P(x|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where μ is the mean and σ is the standard derivation. When \mathbf{x} is multivariate, the Gaussian probability density function is

$$P(\mathbf{x}|\Theta) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)}{2}\right), \quad (5.1)$$

where μ is the mean vector, Σ is the covariance, D is the dimension of data.

Let K be the numbers of the Gaussian sub-distributions, $k = 1, 2, \dots, K$. α_k is the probability of the data belonging to k_{th} sub-distribution, where $\alpha_k \geq 0$, $\sum_{k=1}^K \alpha_k = 1$. $\phi(\mathbf{x}|\Theta_k)$ is the probability density function of k_{th} sub-distribution, it is the same as above (5.1). The latent variable γ_{jk} means the probability of \mathbf{x}_j belonging to k_{th} sub-distribution. The probability density function of GMM is

$$P(\mathbf{x}|\Theta) = \sum_{k=1}^K \alpha_k \phi(\mathbf{x}|\Theta_k). \quad (5.2)$$

We assume all the data are independent of each other. The likelihood function of GMM is

$$L(\Theta) = \prod_{j=1}^n P(\mathbf{x}_j|\Theta),$$

the log-likelihood function is

$$\log L(\Theta) = \sum_{j=1}^n \log P(\mathbf{x}_j|\Theta) = \sum_{j=1}^n \log \left(\sum_{k=1}^K \alpha_k \phi(\mathbf{x}|\Theta_k) \right). \quad (5.3)$$

The MLE method cannot be used here for the estimation of parameters. The EM algorithm (Dempster (1977)) is usually applied to seek the estimation of parameters

iteratively. EM algorithm seeks the low boundary of the likelihood function by the Jensen inequation, then maximizes this low boundary. The algorithm is stated as follows.

Algorithm 5.1. EM Algorithm for Gaussian Mixture Model

Input. The training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbf{R}^l$ and $y_i \in \{-1, 1\}$, $i = 1, \dots, n$.

0. Let $t = 1$ and initialize all the parameters;

1. E-step: calculate the probability of \mathbf{x}_j coming from k sub-distribution:

$$\gamma_{jk} = \frac{\alpha_k \phi(\mathbf{x}_j | \Theta_k)}{\sum_{k=1}^K \alpha_k \phi(\mathbf{x}_j | \Theta_k)}, j = 1, 2, \dots, n, k = 1, 2, \dots, K;$$

2. M-step: update the parameters:

$$\begin{aligned} \mu_k &= \frac{\sum_{j=1}^n (\gamma_{jk} \mathbf{x}_j)}{\sum_{j=1}^n \gamma_{jk}}, k = 1, 2, \dots, K; \\ \Sigma_k &= \frac{\sum_{j=1}^n \gamma_{jk} (\mathbf{x}_j - \mu_k) (\mathbf{x}_j - \mu_k)^T}{\sum_{j=1}^n \gamma_{jk}}, k = 1, 2, \dots, K; \\ \alpha_k &= \frac{\sum_{j=1}^n \gamma_{jk}}{n}, k = 1, 2, \dots, K; \end{aligned}$$

3. Repeat steps 1 and 2 until suitable convergence is obtained;

Output. The final estimation of all parameters.

GMM needs a basic assumption: in every class, the data is normally distributed. Otherwise, a big dataset is needed for GMM to satisfy the Gaussian assumption.

Another important reason to use the discriminative model rather than the generative one is that we should solve the classification problem directly and never solve a more general problem as an intermediate step, such as estimating $p(\mathbf{x}|y)$ (see Vapnik (1998)). Considering the computational efficiency and matters such as the missing data, it seems that the discriminative classifiers are always to be preferred to the generative one. Also, the number of parameters in GMM increases linearly with the growth of data size. GMM performs badly in big data and the high-dimensional data cases. But the discriminative models, such as the SVM and RVM, can control the freedom of parameters by using the sparsity effect.

5.3 Discussion of Chapter 4

In statistical machine learning, the goal of classification is often to obtain a classifier based on the training data and predict an unobserved output value y based on an observed input vector \mathbf{x} from the test data. This requires us to estimate a functional relationship $h(\mathbf{x}) \approx y$ from a set of training data pairs of (\mathbf{x}, y) . Usually, the quality of the predictor $h(\mathbf{x})$ can be measured by a loss function $l(h(\mathbf{x}), y)$. Our goal is to find a predictor $h(\mathbf{x})$ so that the expected loss of $h(\cdot)$ given below is as small as possible:

$$L(h(\cdot)) = E_{\mathbf{X}, Y} l(h(\mathbf{x}), y), \quad (5.4)$$

where we use $E_{\mathbf{X}, Y}$ to denote the expectation with respect to the true underlying distribution of the data. Many of the loss functions consist of $h(\mathbf{x})y$. We give the following definition for this kind of loss functions.

Definition 5.1. Given a classification function $h(\cdot)$, the margin of a subject (\mathbf{x}_i, y_i) is defined as $h(\mathbf{x}_i)y_i$. All the loss functions consisting of the margin are called the *margin-based loss function*.

It is easy to see that a subject is classified correctly by $h(\cdot)$ if and only if its margin is positive.

5.3.1 0 – 1 Loss Function

The classification error of $h(\cdot)$ at a point (\mathbf{x}, y) is:

$$\ell_1(h(\mathbf{x}), y) = \begin{cases} 1, & \text{if } y = 1 \text{ and } h(\mathbf{x}) < 0 \\ 1, & \text{if } y = -1 \text{ and } h(\mathbf{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.5)$$

We transfer (2) to the following Table 5.1:

Table 5.1: Risk Matrix for Binary Classification under 0 – 1 Loss Function.

$\ell_1(y, h(\mathbf{x}))$	$h(\mathbf{x}) \geq 0$	$h(\mathbf{x}) < 0$
$y = 1$	0	1
$y = -1$	1	0

Given a set of training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, we need to find a $h(\mathbf{x})$ that minimizes the empirical misclassification loss:

$$\frac{1}{n} \sum_{i=1}^n \ell_1(h(\mathbf{x}_i), y_i), \quad (5.6)$$

(5.3) is an approximation to the minimization of the true classification error: $L_1(h(\cdot)) = E_{\mathbf{X}, Y} \ell_1(h(\mathbf{X}), Y)$.

5.3.2 Cost-sensitive Loss Function

In practice, the losses of misclassification in different classes are usually inequable. In the cancer detection case, the misdiagnosis of a cancer patient is more serious than the misdiagnosis of a healthy person. So a larger cost should be assigned to the cancer class. Let c_1 and c_{-1} be the cost of misclassification for two classes and the classification error of $h(\cdot)$ at a point (\mathbf{x}, y) is:

$$\ell_2(h(\mathbf{x}), y) = \begin{cases} c_1, & \text{if } y = 1 \text{ and } h(\mathbf{x}) < 0 \\ c_{-1}, & \text{if } y = -1 \text{ and } h(\mathbf{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.7)$$

Typically, the positive (minority) class is our interest. So $c_1 \geq c_{-1}$. The empirical misclassification loss is the average of the linear summation of $\ell_2(h(\mathbf{x}_i), y_i)$ and let $c_{-1} = 1$, $c = \frac{c_1}{c_{-1}}$. Rewrite (5.7) as

$$\ell_2(h(\mathbf{x}), y) = \begin{cases} c, & \text{if } y = 1 \text{ and } h(\mathbf{x}) < 0 \\ 1, & \text{if } y = -1 \text{ and } h(\mathbf{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

Here $c \geq 1$. (5.7) is equivalent to (5.8) when we focus on minimizing the empirical loss. Transfer (5.8) to the following Table 5.2:

Table 5.2: Risk Matrix for Binary Classification under Cost-sensitive Loss Function.

$\ell_2(y, h(\mathbf{x}))$	$h(\mathbf{x}) \geq 0$	$h(\mathbf{x}) < 0$
$y = 1$	0	c
$y = -1$	1	0

Given a set of training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, we wish to find a $h(\mathbf{x})$ that minimizes the empirical misclassification loss:

$$\frac{1}{n} \sum_{i=1}^n \ell_2(h(\mathbf{x}_i), y_i), \quad (5.9)$$

(5.9) is an approximation to the true classification error: $L_2(h(\cdot)) = E_{\mathbf{X}, Y} \ell_2(h(\mathbf{X}), Y)$. The 0 – 1 loss function is a special case of the Cost-sensitive loss function. When $c = 1$, these two loss functions are identical. We only consider the Cost-sensitive loss function in the rest of this part. Minimizing (5.9) is hard due to the nonconvexity of the classification loss function ℓ_2 . Typically, there are two approaches to solve the problem.

(1) To avoid dealing with (5.9), find the Bayes classifier directly which is the optimal one;

(2) Instead of minimizing (5.9), minimize a convex upper bound of the empirical loss. For example, AdaBoost in Chapter 2 employs the exponential loss function $\exp(-h(\mathbf{x})y)$.

5.3.3 Cost-sensitive Bayes Classifier

Assume \mathbf{S} is identical, independently distributed (*i.i.d.*) samples from the data space and the conditional in-class probabilities, $P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = 1)$ and $P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = -1)$ are given.

5.3.3.1 Bayes Classifier

Given a class label k , the prior is

$$P(\mathbf{Y} = k), \quad (5.10)$$

here $P(\mathbf{Y} = k) + P(\mathbf{Y} = -k) = 1$.

The likelihood is

$$P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = k) = \prod_{i=1}^n P(\mathbf{X}_i = \mathbf{x}_i|\mathbf{Y} = k). \quad (5.11)$$

The posterior is

$$\begin{aligned} & P(\mathbf{Y} = k|\mathbf{X} = \mathbf{x}) \\ &= \frac{P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = k)P(\mathbf{Y} = k)}{P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = k)P(\mathbf{Y} = k) + P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = -k)P(\mathbf{Y} = -k)}. \end{aligned} \quad (5.12)$$

The Bayes rule is to minimize the posterior risk $L_{pos}(h(\mathbf{x}))$. The test is:

$$H_0 : h(\mathbf{x}) \geq 0 \quad v.s. \quad H_a : h(\mathbf{x}) < 0.$$

We have

$$\begin{aligned}
L_{pos}(h(\mathbf{x}) \geq 0) &= 1 \cdot P(\mathbf{Y} = -1 | \mathbf{X} = \mathbf{x}) \\
&= \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1) + P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}, \\
L_{pos}(h(\mathbf{x}) < 0) &= c \cdot P(\mathbf{Y} = 1 | \mathbf{X} = \mathbf{x}) \\
&= \frac{c \cdot P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1) + P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}.
\end{aligned} \tag{5.13}$$

So $h(\cdot)$ should be

$$\begin{cases} h(\mathbf{x}) \geq 0, & \text{if } c \cdot P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1) \geq P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1), \\ h(\mathbf{x}) < 0, & \text{if } c \cdot P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1) < P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1). \end{cases} \tag{5.14}$$

Rewrite (5.14) as

$$\begin{cases} h(\mathbf{x}) \geq 0, & \text{if } c \geq \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1)} \\ h(\mathbf{x}) < 0, & \text{if } c < \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1)}. \end{cases} \tag{5.15}$$

Assume $h(\mathbf{x}) \in \{-1, 1\}$, rewrite (5.15) as:

$$\begin{aligned}
h_b(\mathbf{x}) &= \text{sign} \left(c - \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)P(\mathbf{Y} = -1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)P(\mathbf{Y} = 1)} \right) \\
&= \text{sign} \left(c - \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -1)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = 1)} \cdot \frac{P(\mathbf{Y} = -1)}{P(\mathbf{Y} = 1)} \right),
\end{aligned} \tag{5.16}$$

here $\text{sign}(A) = 1$ if $A \geq 0$, $\text{sign}(A) = -1$ if $A < 0$. $h_b(\mathbf{x})$ is the Bayes classifier under the Cost-sensitive loss function $\ell_2(h(\mathbf{x}), y)$. Next we give the definition of the Bayes

risk.

Definition 5.2. (Bayes Risk under Cost-sensitive loss function)

The Bayes risk is the minimum of the risk for all classifiers:

$$L_2^* = \inf_{h(\cdot)} L_2(h(\cdot)).$$

We can prove that the Bayes risk is achieved by the Bayes classifier $h_b(\mathbf{x})$.

Theorem 5.1. (Risk of Bayes Classifier)

$$L_2(h_b(\cdot)) = L_2^*$$

Proof. See Appendix E1. □

From Theorem 5.1, the Bayes classifier with the Cost-sensitive loss function is the optimal one. The performance of any given classifier can be evaluated in terms of how close its risk is to the Bayes risk. The Bayes classifier $h_b(\mathbf{x})$ depends on three factors: the cost factor c , the likelihood ratio $\frac{P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=-1)}{P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=1)}$, and the prior ratio $\frac{P(\mathbf{Y}=-1)}{P(\mathbf{Y}=1)}$.

(1) The implementer usually determines the cost factor c based on the working experience, where $c \geq 1$.

(2) Given the conditional in-class probabilities, the likelihood can be calculated directly based on the training data \mathbf{S} and we can obtain the likelihood ratio.

(3) The prior ratio needs to be estimated and we discuss this issue in the following subsection.

5.3.3.2 Prior Ratio Estimation

We set $p = P(\mathbf{Y} = 1)$ and $1 - p = P(\mathbf{Y} = -1)$. So the prior ratio is $\frac{1-p}{p}$. First, we consider the MAP estimation for the prior.

Theorem 5.2. *The Maximum A Posterior (MAP) estimate of $P(\mathbf{Y} = k)$ is*

$$P(\mathbf{Y} = 1) = \frac{n_+}{n}, P(\mathbf{Y} = -1) = \frac{n_-}{n}. \quad (5.17)$$

Proof. See Appendix E2. □

The prior ratio based on Theorem 5.2 is $\frac{n_-}{n_+}$. If we treat p as the parameter and bring in a prior for p . The probability of the class label is:

$$P(y_i|p) = p^{\frac{1+y_i}{n}} (1-p)^{\frac{1-y_i}{n}},$$

here $i = 1, 2, \dots, n$. The likelihood of y_i is:

$$\begin{aligned} P(y_{1:n}|p) &= \prod_{i=1}^n p^{\frac{1+y_i}{n}} (1-p)^{\frac{1-y_i}{n}} \\ &= p^{\frac{n+\sum_{i=1}^n y_i}{2}} (1-p)^{\frac{n-\sum_{i=1}^n y_i}{2}} \\ &= p^{n_+} (1-p)^{n_-}. \end{aligned}$$

We set a Beta prior for the parameter p :

$$P(p|a_p, b_p) = \text{Beta}(p; a_p, b_p) = \frac{1}{B(a_p, b_p)} p^{a_p-1} (1-p)^{b_p-1}.$$

The posterior of p given $y_i (i = 1, 2, \dots, n)$ is

$$\begin{aligned} P(p|y_{1:n}) &\propto P(y_{1:n}|p)P(p|a_p, b_p) \\ &= p^{n_+} (1-p)^{n_-} \frac{1}{B(a_p, b_p)} p^{a_p-1} (1-p)^{b_p-1} \\ &\propto \text{Beta}(p; n_+ + a_p, n_- + b_p). \end{aligned}$$

The estimation of p and $1 - p$ are $E(p|y_{1:n}) = \frac{n_+ + a_p}{n + a_p + b_p}$ and $E(1 - p|y_{1:n}) = \frac{n_- + b_p}{n + a_p + b_p}$. If we choose $a_p = b_p = \frac{1}{2}$, the estimation for the prior is

$$P(\mathbf{Y} = 1) = \frac{n_+ + \frac{1}{2}}{n + 1}, P(\mathbf{Y} = -1) = \frac{n_- + \frac{1}{2}}{n + 1}. \quad (5.18)$$

The corresponding prior ratio is $\frac{n_- + \frac{1}{2}}{n_+ + \frac{1}{2}}$. The estimation in (5.18) is a Laplace Smoothing of the estimation in (5.17) to avoid the zero-probability estimate. This approach is study-worthy and can provide the theoretically optimal classifier for the imbalanced data classification problem.

5.4 Comparison Between Boosting and Kernel Methods

Boosting and Kernel methods are two typical techniques for classification. They both have received considerable attention in recent years and many successful applications have been described in the literature. This project studies the Adaptive Boosting model in Chapter 2, which is the most famous Boosting algorithm. RVM is studied in Chapters 3 and 4. Several improved RVM algorithms are proposed to make the Kernel methods family flourish. Boosting and Kernel methods have something in common to justify their success, namely the margin. By using a kernel trick to map the training samples from an input space to a high dimensional feature space, the Kernel method finds an optimal separating hyperplane and uses a parameter to balance its model complexity and training error. To build the proper classification borderline, SVM and RVM tend to find the support points and relevant points, respectively. On the other

hand, Boosting tries to obtain the same goal indirectly by minimizing a cost function related to margin. Boosting is a general technique for improving the performance of any given classifier. It can effectively combine a number of weak classifiers, which are generally a little better than a random guess, into a strong classifier that can achieve an arbitrarily low error rate. In this part, we show several numeric studies based on the Boosting and Kernel methods in this project. A discussion of the curse of dimensionality in the Kernel method is stated. Some ideas to improve the Kernel methods are finally listed.

The classification results are similar in the simulated Gaussian data studies of Chapters 2, 3, and 4. For the high-dimensional real datasets, the significant differences of the classification results between the algorithms in this project are worthy of further discussion. Six real binary datasets from KEEL in Chapters 3 and 4 are used in this part. Three Boosting methods in Table 2.3: Ada-DT, En-Ada, and Re-Ada are studied. Four Kernel methods in Table 2.3, Algorithm 3.3, 4.2, and 4.3: SVMs, the Generic Bayesian RVM, the Generic PRVM, and the Fully Hierarchical PRVM are studied. The first four criteria in Table 3.1 are used for the evaluation of classifiers' performance.

Table 5.3: The Results of Glass-1 Data($b = 1.82$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	0.7042	0.6916	0.0552	0.0658
En-Ada	0.8318	0.6822	0.6316	0.6316
Re-Ada	0.9252	0.6636	0.8421	0.8421
SVMs	0.8131	0.7477	0.0553	0.0474
Generic Bayesian RVM	0.6449	0.6449	0.0000	0.0000
Generic PRVM	0.6721	0.6511	0.0000	0.0000
Fully Hierarchical PRVM	0.6890	0.6669	0.0214	0.0108

Table 5.4: The Results of Iris-0 Data($b = 2.00$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	1.0000	1.0000	1.0000	1.0000
En-Ada	1.0000	1.0000	1.0000	1.0000
Re-Ada	1.0000	1.0000	1.0000	1.0000
SVMs	1.0000	1.0000	1.0000	1.0000
Generic Bayesian RVM	1.0000	1.0000	1.0000	1.0000
Generic PRVM	1.0000	1.0000	1.0000	1.0000
Fully Hierarchical PRVM	1.0000	1.0000	1.0000	1.0000

Table 5.5: The Results of Newthyroid-1 Data($b = 5.14$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	0.9366	0.9815	0.0514	0.0253
En-Ada	0.9878	0.9907	0.7234	0.7524
Re-Ada	0.9453	0.9630	0.7934	0.8065
SVMs	0.8422	0.8426	0.0553	0.0556
Generic Bayesian RVM	0.8318	0.8333	0.0000	0.0000
Generic PRVM	0.8710	0.8541	0.0000	0.0032
Fully Hierarchical PRVM	0.9233	0.8509	0.0591	0.0000

Table 5.6: The Results of Glass-6 Data($b = 6.38$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	0.9448	0.9630	0.7891	0.8667
En-Ada	1.0000	0.9352	1.0000	0.8332
Re-Ada	1.0000	0.9352	1.0000	0.8322
SVMs	0.9906	0.9537	0.9286	0.6667
Generic Bayesian RVM	0.8679	0.8611	0.0000	0.0000
Generic PRVM	0.8901	0.8611	0.0000	0.0000
Fully Hierarchical PRVM	0.8901	0.8611	0.0000	0.0000

Table 5.7: The Results of Ecoli-0345 Data($b = 9.00$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	0.9711	0.9600	0.6574	0.6000
En-Ada	0.9800	0.9800	0.6316	0.6316
Re-Ada	0.9819	0.9700	0.6777	0.6173
SVMs	0.9000	0.9000	0.0000	0.0000
Generic Bayesian RVM	0.8600	0.9000	0.0000	0.0000
Generic PRVM	0.8596	0.9015	0.0000	0.0000
Fully Hierarchical PRVM	0.9274	0.9016	0.2977	0.0000

Table 5.8: The Results of Glass-2 Data($b = 11.59$).

Algorithms	Evaluation Measures			
	r_g^{train}	r_g^{test}	r_p^{train}	r_p^{test}
Ada-DT	0.8672	0.8519	0.1522	0.1111
En-Ada	0.9434	0.8981	0.2500	0.3333
Re-Ada	0.9434	0.8981	0.2500	0.3333
SVMs	0.9245	0.9167	0.0000	0.0000
Generic Bayesian RVM	0.9245	0.9167	0.0000	0.0000
Generic PRVM	0.9241	0.9221	0.0000	0.0000
Fully Hierarchical PRVM	0.9400	0.9452	0.0159	0.0231

It is obvious that the Boosting algorithms outperform the Kernel methods significantly. Note that all the Kernel methods in this project use the RBF Gaussian kernel, which is a local kernel, to construct the models. The reason for the unsatis-

factory performances is the curse of dimensionality for the local kernel methods (see Bengio et al. (2005)). The curse of dimensionality (see (Bellman (1961))) means the number of parameters is large concerning the number of training samples. There is a risk of over-fitting the training data, which means the poor generalization to classify new data correctly. Additionally, sensitivity to noise and computational complexity may increase with the dimension of data. This problem is known as the curse of dimensionality. For the local Kernel methods, the classification output $f(\mathbf{x}, \mathbf{w})$ is mostly determined by the neighbors of \mathbf{x} in the training set. In high-dimensional data case, the required number of neighborhoods could grow exponentially with the dimensionality of the data. We have to balance the bias-variance trade-off argument for classification models: if we make the neighbor regions smaller, bias is reduced and more complex functions can be represented. But the variance increases because there is not enough data used to determine the value of $f(\mathbf{x}, \mathbf{w})$ around \mathbf{x} , $f(\mathbf{x}, \mathbf{w})$ becomes less stable. This is the reason why the kernel functions methods perform badly in the high-dimensional real data studies.

To address this issue, Non-Local Means (NLM) can be used for RVM models in this project. NLM is a method introduced by Buades et al. (2005) and has become quite popular. The method was further enhanced for speed in subsequent works by Bilcu et al. (2007). The NLM is a weighted averaging process of the local kernels. This process can be written mathematically as

$$\hat{z}(\mathbf{x}_i) = \frac{1}{C_i} \sum_{j=1}^n k_{ij} y_j,$$

where $C_i = \sum_{j=1}^n k_{ij}$. k_{ij} is the kernel function such as the RBF Gaussian kernel:

$$k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\gamma^2}\right).$$

This kind of global kernels can solve the curse of dimensionality in the RVM models and this approach is one of the future research works.

Appendix A

Proofs of Theorems in Chapter 2

A1. Proof of Theorem 2.1.

It follows from **Fact 2.2** that

$$\begin{aligned}\sum_{i:y_i=-1} H_a(\mathbf{x}_i) &= N_p k \sum_{t=1}^T \{(2\gamma_t - 1) + (2\epsilon_t - 1)(b + 1)\} \exp\{\beta(2\gamma_t - 1)\} \\ &= \sum_{i:y_i=1} H_a(\mathbf{x}_i) + N_p k (b + 1) \sum_{t=1}^T (2\epsilon_t - 1) \exp\{\beta(2\gamma_t - 1)\}.\end{aligned}$$

Note that $N_p k (b + 1) \sum_{t=1}^T (2\epsilon_t - 1) \exp\{\beta(2\gamma_t - 1)\} < 0$ because of $0 \leq \epsilon_t < \frac{1}{2}$. So

$\sum_{i:y_i=-1} H_a(\mathbf{x}_i) < \sum_{i:y_i=1} H_a(\mathbf{x}_i)$ always holds. Consider three situations:

(a) When $\gamma_t \in [0, \frac{1}{2})$, it is obvious that $\sum_{i:y_i=1} H_a(\mathbf{x}_i) < 0$, this implies (2.3);

(b) When $\gamma_t = \frac{1}{2}$, (2.4) follows from $\sum_{i:y_i=1} H_a(\mathbf{x}_i) = 0$;

(c) When $\gamma_t \in (\frac{1}{2}, 1]$, under (2.5),

$$\begin{aligned}\sum_{i:y_i=1} H_a(\mathbf{x}_i) &= N_p k \sum_{t=1}^T (2\gamma_t - 1) \exp\{\beta(2\gamma_t - 1)\} > 0, \\ \sum_{i:y_i=-1} H_a(\mathbf{x}_i) &= N_p k \sum_{t=1}^T \{(2\gamma_t - 1) + (2\epsilon_t - 1)(b + 1)\} \exp\{\beta(2\gamma_t - 1)\} < 0.\end{aligned}$$

These imply (2.6).

A2. Proof of Theorem 2.2.

Under the assumption of (2.13), we can show that

$$\begin{aligned}& \sum_{i:y_i=1} H'_3(\mathbf{x}_i) - \sum_{i:y_i=1} H_1(\mathbf{x}_i) > 0 \\ \Leftrightarrow & \frac{\exp(\gamma_t - 0.5) + 0.5}{0.5 - \epsilon_t} + \frac{0.5(b + 1)}{\gamma_t - 0.5} - 1 > 0 \\ \Leftrightarrow & (b + 1) \frac{\exp(\gamma_t - 0.5) + 1}{\gamma_t - 0.5} > 1.\end{aligned}\tag{A.1}$$

Note that (A.1) always holds. So (14) gets proved. On the other hand, we have

$$\begin{aligned}& \sum_{i:y_i=-1} H'_3(\mathbf{x}_i) - \sum_{i:y_i=-1} H_1(\mathbf{x}_i) \leq 0 \\ \Leftrightarrow & \{(\gamma_t - 0.5) + (\epsilon_t - 0.5)(b + 1)\} \left\{ \frac{\exp(\gamma_t - 0.5) + 0.5}{0.5 - \epsilon_t} + \frac{0.5(b + 1)}{\gamma_t - 0.5} - 1 \right\} \leq 0 \\ \Leftrightarrow & \left\{ (b + 1) - \frac{0.5(b + 1)^2}{\gamma_t - 0.5} \right\} (0.5 - \epsilon_t)^2 - \left\{ (\gamma_t - 0.5) + (b + 1)\exp(\gamma_t - 0.5) \right\} \cdot \\ & (0.5 - \epsilon_t) + (\gamma_t - 0.5)\exp(\gamma_t - 0.5) + 0.5(\gamma_t - 0.5) \leq 0 \\ \Leftrightarrow & 0.5 - \epsilon_t \leq \frac{\gamma_t - 0.5}{b + 1} \\ \Leftrightarrow & 0.5\{1 - (2\gamma_t - 1)/(b + 1)\} \leq \epsilon_t.\end{aligned}\tag{A.2}$$

(A.2) always holds under assumption (2.13), so Theorem 2.2 gets proved.

Appendix B

Proofs of Theorems in Chapter 3

Appendix B1. Proof of Theorem 3.1.

First, the logarithm of the posterior for \mathbf{w} is

$$\begin{aligned} l^{\mathbf{w}} &= \log p(\mathbf{w} | \text{others}) \\ &= \sum_{i=1}^n \left[\frac{1 + y_i^{\text{train}}}{2} \log \left(\frac{1}{1 + \exp(-\phi_i^{\text{train}} \mathbf{w})} \right) + \frac{1 - y_i^{\text{train}}}{2} \log \left(\frac{\exp(-\phi_i^{\text{train}} \mathbf{w})}{1 + \exp(-\phi_i^{\text{train}} \mathbf{w})} \right) \right] - \\ &\quad \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} + C \\ &= \sum_{i=1}^n \left[\frac{1 + y_i^{\text{train}}}{2} \log \left(\frac{1}{1 + \exp(-\sum_{s=0}^n \phi_{i,s}^{\text{train}} w_s)} \right) + \frac{1 - y_i^{\text{train}}}{2} \right. \\ &\quad \left. \log \left(\frac{\exp(-\sum_{s=0}^n \phi_{i,s}^{\text{train}} w_s)}{1 + \exp(-\sum_{s=0}^n \phi_{i,s}^{\text{train}} w_s)} \right) \right] - \frac{1}{2} \sum_{s=0}^n \alpha_s w_s^2 + C, \end{aligned}$$

where C is some constant. For any w_k in \mathbf{w} , we have

$$\begin{aligned}
l_k^{\mathbf{w}} &= \log p(w_k | \text{others}) \\
&= \sum_{i=1}^n \left[\frac{1 + y_i^{\text{train}}}{2} \log \left(\frac{1}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) + \frac{1 - y_i^{\text{train}}}{2} \right. \\
&\quad \left. \log \left(\frac{\exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) \right] - \frac{1}{2} \sum_{s=0, s \neq k}^n \alpha_s w_s^2 - \frac{1}{2} \alpha_k w_k^2 + C,
\end{aligned}$$

where $k = 0, 1, 2, \dots, n$. Then we calculate the first and second divergency,

$$\begin{aligned}
\frac{\partial}{\partial w_k} l_k^{\mathbf{w}} &= \sum_{i=1}^n \left[\frac{1 + y_i^{\text{train}}}{2} \left(\frac{\phi_{i,k}^{\text{train}} \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) + \right. \\
&\quad \left. \frac{1 - y_i^{\text{train}}}{2} \left(\frac{-\phi_{i,k}^{\text{train}}}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) \right] - \alpha_k w_k,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial w_k^2} l_k^{\mathbf{w}} &= \sum_{i=1}^n \left[\frac{1 + y_i^{\text{train}}}{2} \left(\frac{-\phi_{i,k}^{\text{train}^2}}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) + \right. \\
&\quad \left. \frac{1 - y_i^{\text{train}}}{2} \left(\frac{-\phi_{i,k}^{\text{train}^2} \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}{1 + \exp\left(-\sum_{s=0, s \neq k}^n \phi_{i,s}^{\text{train}} w_s - \phi_{i,k}^{\text{train}} w_k\right)}\right) \right] - \alpha_k \\
&< 0,
\end{aligned}$$

the log-concavity is proved.

Appendix B2. Proof of Theorem 3.2.

The conditional posterior of any η_k in $\boldsymbol{\eta}$ is

$$p(\eta_k \mid \text{others}) \propto \exp \left[\frac{\eta_k}{2} - \frac{1}{2} \exp(\eta_k) w_k^2 - \frac{(\eta_k - \mu)^2}{2\tau^2(1 - \rho)} + \frac{\rho(\eta_k - \mu)^2}{2\tau^2(1 - \rho)(1 + n\rho)} + \frac{\rho(\eta_k - \mu) \sum_{s=0, s \neq k}^n (\eta_s - \mu)}{\tau^2(1 - \rho)(1 + n\rho)} \right],$$

where $k = 0, 1, 2, \dots, n$. For a constant C , the log-posterior of η_k is

$$\begin{aligned} l_k^\eta &= \log p(\eta_k \mid \text{others}) \\ &= C + \frac{\eta_k}{2} - \frac{1}{2} \exp(\eta_k) w_k^2 - \frac{1 + (n - 1)\rho}{2\tau^2(1 - \rho)(1 + n\rho)} (\eta_k - \mu)^2 + \\ &\quad \frac{\rho(\eta_k - \mu) \sum_{s=0, s \neq k}^n (\eta_s - \mu)}{\tau^2(1 - \rho)(1 + n\rho)}. \end{aligned}$$

The second divergence is

$$\frac{\partial^2}{\partial \eta_k^2} l_k^\eta = -\frac{1}{2} \exp(\eta_k) w_k^2 - \frac{1 + (n - 1)\rho}{\tau^2(1 - \rho)(1 + n\rho)}.$$

Because $1 + (n - 1)\rho > 0$ for any $\rho \in (-1, 1)$, $\frac{\partial^2}{\partial \eta_k^2} l_k^\eta < 0$ always holds.

Appendix C

Proofs of Lemmas in Chapter 4

Appendix C1. Proof of Lemma 4.1.

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}^{train}, \boldsymbol{\alpha}, \boldsymbol{\mu}) &\propto \exp\left(-\frac{1}{2}\mathbf{w}^T \mathbf{A} \mathbf{w}\right) \prod_{i=1}^n \phi(\mu_i - K_i \mathbf{w}) \\ &= \exp\left[-\frac{1}{2}\left(\mathbf{w}^T \mathbf{A} \mathbf{w} + \sum_{i=1}^n (\mu_i - K_i \mathbf{w})^2\right)\right] \\ &\propto \exp\left[-\frac{1}{2}\left(\mathbf{w}^T \mathbf{A} \mathbf{w} + \sum_{i=1}^n ((K_i \mathbf{w})^2 - 2\mu_i K_i \mathbf{w})\right)\right] \\ &= \exp\left[-\frac{1}{2}\left(\mathbf{w}^T \mathbf{A} \mathbf{w} + \mathbf{w}^T \mathbf{K}^T \mathbf{K} \mathbf{w} - 2\mathbf{w}^T \mathbf{K}^T \boldsymbol{\mu}\right)\right] \\ &= \exp\left[-\frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}}) \mathbf{M} (\mathbf{w} - \hat{\mathbf{w}})^T\right] \\ &\propto \mathcal{N}_{n+1}(\hat{\mathbf{w}}, \mathbf{M}^{-1}), \end{aligned}$$

where $\mathbf{M} = \mathbf{A} + \mathbf{K}^T \mathbf{K}$, $\hat{\mathbf{w}} = \mathbf{M}^{-1} \mathbf{K}^T \boldsymbol{\mu}$.

Appendix C2. Proof of Lemma 4.2.

It is obvious that

$$\begin{aligned} D &= b + \Phi^{-1}(u\Phi(-b)) \\ \Rightarrow u(D) &= \frac{\Phi(D-b)}{\Phi(-b)} \\ \Rightarrow P(D) &= \frac{\partial u(D)}{\partial D} \mathbf{1}_{(0 \leq u \leq 1)} = \phi(D-b) \mathbf{1}_{(D \leq 0)}. \end{aligned}$$

It is similar process to prove another statement.

Appendix D

Ratio of Uniforms Sampling Method

In this section we introduce the ratio of uniforms method, which is a random number generation approach. This method was original proposed by Kinderman & Monahan (1977). Then Ripley (1987) further improved this method. Suppose that a bivariate random variable (U_1, U_2) is uniformly distributed and satisfies the following inequality:

$$0 \leq U_1 \leq \sqrt{g(U_2/U_1)},$$

where $g(x)$ is a any nonnegative function. So $X = U_2/U_1$ has a density function $f(x) = \frac{h(x)}{\int h(x)dx}$. The joint density of U_1 and U_2 , denoted by $f_{12}(u_1, u_2)$ is

$$f_{12}(u_1, u_2) = \begin{cases} k, & \text{if } 0 \leq u_1 \leq \sqrt{g(u_2/u_1)} \\ 0, & \text{otherwise} \end{cases},$$

where k is a constant number. Conduct the following transformation from (u_1, u_2) to (x, y) :

$$x = \frac{u_2}{u_1}, \quad y = u_1.$$

It is obvious that $u_1 = y, u_2 = xy$. So the Jacobian for this simple transformation is:

$$J = \begin{vmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} \\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ y & x \end{vmatrix} = -y.$$

Rewritten $f_{xy}(x, y)$ as:

$$f_{xy}(x, y) = |J|f_{12}(y, xy) = ky,$$

where $0 \leq y \leq \sqrt{g(x)}$. The marginal density of X , denoted by $f_x(x)$, is obtained as follows:

$$f_x(x) = \int_0^{\sqrt{g(x)}} f_{xy}(x, y)dy = \int_0^{\sqrt{g(x)}} kydy = k \left[\frac{y^2}{2} \right]_0^{\sqrt{g(x)}} = \frac{k}{2}g(x) = f(x),$$

where k is taken as $k = \frac{2}{\int g(x)dx}$. Thus, it is shown that $f_x(\cdot)$ is equivalent to $f(\cdot)$.

In practice, we need to choose the rectangle which encloses the area $0 \leq U_1 \leq \sqrt{g(U_2/U_1)}$ on the domain of (U_1, U_2) . The basic idea is to generate a uniform point in the rectangle, and reject the point which does not satisfy $0 \leq u_1 \leq \sqrt{g(u_2/u_1)}$. So in this method, we generate two independent uniform random draws u_1 and u_2 from

$U(0, b)$ and $U(c, d)$, respectively. The rectangle is given by:

$$0 \leq u_1 \leq b, \quad c \leq u_2 \leq d,$$

where b, c and d are given by:

$$b = \sup_x \sqrt{g(x)}, \quad c = - \sup_x x \sqrt{g(x)}, \quad d = \sup_x x \sqrt{g(x)}.$$

The sampling process is as follows (see Ripley (1987)):

- (1) Generate u_1 and u_2 independently from $U(0, b)$ and $U(c, d)$;
- (2) If $u_1^2 \leq h(u_2/u_1)$, set $x = u_2/u_1$. Else, return to (1).

Appendix E

Proofs of Theorems in Chapter 5

A1. Proof of Theorem 5.1.

Let $h(\cdot)$ be any classifier. We will show that

$$\begin{aligned} & c \cdot P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x}) + P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x}) \\ \geq & c \cdot P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x}) + P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x}). \end{aligned} \text{(E.1)}$$

$$\begin{aligned}
& P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = k, \mathbf{X} = \mathbf{x}) \\
&= 1 - P(h(\mathbf{X}) = \mathbf{Y} | \mathbf{Y} = k, \mathbf{X} = \mathbf{x}) \\
&= 1 - [P(\mathbf{Y} = k, h(\mathbf{X}) = k | \mathbf{X} = \mathbf{x})] \\
&= 1 - [E[\mathbf{1}_{\mathbf{Y}=k} \mathbf{1}_{h(\mathbf{X})=k} | \mathbf{X} = \mathbf{x}]] \\
&= 1 - [\mathbf{1}_{h(\mathbf{X})=k} E[\mathbf{1}_{\mathbf{Y}=k} | \mathbf{X} = \mathbf{x}]] \\
&= 1 - [\mathbf{1}_{h(\mathbf{X})=k} P[\mathbf{Y} = k | \mathbf{X} = \mathbf{x}]] \\
&= 1 - \left[\mathbf{1}_{h(\mathbf{X})=k} \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = k) P(\mathbf{Y} = k)}{P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = k) P(\mathbf{Y} = k) + P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = -k) P(\mathbf{Y} = -k)} \right].
\end{aligned}$$

Rewrite (5.14) and consider the following difference

$$\begin{aligned}
& [c \cdot P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x}) + P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x})] - \\
& [c \cdot P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x}) + P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x})] \\
&= c \cdot [P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x}) - P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = 1, \mathbf{X} = \mathbf{x})] + \\
& [P(h(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x}) - P(h_b(\mathbf{X}) \neq \mathbf{Y} | \mathbf{Y} = -1, \mathbf{X} = \mathbf{x})] \\
&= c \cdot \left[\frac{P(\mathbf{x} | \mathbf{Y} = 1) P(\mathbf{Y} = 1)}{P(\mathbf{x} | \mathbf{Y} = 1) P(\mathbf{Y} = 1) + P(\mathbf{x} | \mathbf{Y} = -1) P(\mathbf{Y} = -1)} (\mathbf{1}_{h_b(\mathbf{X})=1} - \mathbf{1}_{h(\mathbf{X})=1}) \right] + \\
& \left[\frac{P(\mathbf{x} | \mathbf{Y} = -1) P(\mathbf{Y} = -1)}{P(\mathbf{x} | \mathbf{Y} = 1) P(\mathbf{Y} = 1) + P(\mathbf{x} | \mathbf{Y} = -1) P(\mathbf{Y} = -1)} (\mathbf{1}_{h_b(\mathbf{X})=-1} - \mathbf{1}_{h(\mathbf{X})=-1}) \right] \\
&= \frac{c \cdot P(\mathbf{x} | \mathbf{Y} = 1) P(\mathbf{Y} = 1) - P(\mathbf{x} | \mathbf{Y} = -1) P(\mathbf{Y} = -1)}{P(\mathbf{x} | \mathbf{Y} = 1) P(\mathbf{Y} = 1) + P(\mathbf{x} | \mathbf{Y} = -1) P(\mathbf{Y} = -1)} (\mathbf{1}_{h_b(\mathbf{X})=1} - \mathbf{1}_{h(\mathbf{X})=1}) \\
&\geq 0
\end{aligned}$$

So that (5.14) always holds.

A2. Proof of Theorem 5.2.

The posterior is

$$\Pi(\boldsymbol{\theta}) = \prod_{i=1}^n P(\mathbf{X} = \mathbf{x}_i | \mathbf{Y} = y_i) P(\mathbf{Y} = y_i),$$

here $\boldsymbol{\theta}$ is all the parameters.

$$\begin{aligned} \pi(\boldsymbol{\theta}) &= \ln \Pi(\boldsymbol{\theta}) \\ &= \sum_{i=1}^n \ln P(\mathbf{Y} = y_i) + \sum_{i=1}^n \ln P(\mathbf{X} = \mathbf{x}_i | \mathbf{Y} = y_i) \end{aligned}$$

Let

$$\begin{aligned} \alpha_k &= P(\mathbf{Y} = y_i) \\ &= \begin{cases} \alpha_1, & \text{when } y_i = 1 \\ \alpha_{-1}, & \text{when } y_i = -1 \end{cases} \\ &= \prod_{k=1,-1} \alpha_k^{\mathbf{1}_{y_i=k}}. \end{aligned}$$

We know $\sum_{k=1,-1} \alpha_k = 1$. So

$$\begin{aligned} \pi(\boldsymbol{\theta}) &= \pi(\alpha_k, \lambda) \\ &= \sum_{i=1}^n \ln \left[\prod_{k=1,-1} \alpha_k^{\mathbf{1}_{y_i=k}} \right] + \lambda \left(\sum_{k=1,-1} \alpha_k - 1 \right) \\ &= \sum_{i=1}^n \sum_{k=1,-1} \mathbf{1}_{y_i=k} \ln(\alpha_k) + \lambda \left(\sum_{k=1,-1} \alpha_k - 1 \right) \end{aligned}$$

Let

$$\begin{aligned}\frac{\partial \pi(\alpha_k, \lambda)}{\alpha_k} &= \sum_{i=1}^n \frac{\mathbf{1}_{y_i=k}}{\alpha_k} + \lambda \\ &= 0,\end{aligned}$$

we have $\alpha_k = -\frac{\sum_{i=1}^n \mathbf{1}_{y_i=k}}{\lambda}$. Also $\sum_{k=1,-1} \alpha_k = 1$, we have $\lambda = -n$. So

$$\begin{aligned}\alpha_k &= -\frac{\sum_{i=1}^n \mathbf{1}_{y_i=k}}{\lambda} \\ &= \frac{\sum_{i=1}^n \mathbf{1}_{y_i=k}}{n}.\end{aligned}$$

Obtain the result $P(y_i = 1) = \frac{n_+}{n}$ and $P(y_i = -1) = \frac{n_-}{n}$.

Bibliography

- [1] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing* **17**, 255-287.
- [2] E. Alpaydin (2009), Introduction to machine learning.
- [3] J. Albert and S. Chib (1995). Bayesian residual analysis for binary response regression models. *Biometrika* **82(4)**, 747-769.
- [4] H. Akaike (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19(6)**, 716-723.
- [5] Bengio, Y., Delalleau, O., and Le Roux, N. (2005). The curse of dimensionality for local kernel machines. *Techn. Rep*, 1258.
- [6] Bellman, R. (1961). On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* **4(6)**, 284.

- [7] Buades, A., Coll, B., and Morel, J. M. (2005). A non-local algorithm for image denoising. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **Vol. 2**, 60-65.
- [8] Bilcu, R. C., and Vehvilainen, M. (2007). Fast nonlocal means for image denoising. *Digital Photography III* **Vol. 6502**, 65020R.
- [9] Y. Bian, M. Cheng, C. Yang, Y. Yuan, Q. Li, J. L. Zhao, and L. Liang (2016). Financial fraud detection: a new ensemble learning approach for imbalanced data. *PACIS 2016 Proceedings*, 315.
- [10] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, S. Mensah, and Mahakil (2017). Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering* **44 (6)**, 534-550.
- [11] G. E. Batista, R. C. Prati, and M. C. Monard (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter* **6 (1)**, 20-29.
- [12] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144-152.
- [13] Christopher M Bishop and Michael E Tipping (2000). Variational relevance vector machines. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 46-53.

- [14] Andreas Ch Braun, Uwe Weidner, and Stefan Hinz (2012). Classification in high-dimensional feature spaces assessment using SVM, IVM and RVM with focus on simulated enmap data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5(2)**, 436-443.
- [15] P. K. Chan and S. J. Stolfo (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, 164-168.
- [16] F. Cheng, J. Zhang, and C. Wen (2016). Cost-sensitive large margin distribution machine for classification of imbalanced data, *Pattern Recognition Letters* **80**, 107-112.
- [17] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer (2003). Smoteboost: Improving prediction of the minority class in boosting. *European Conference on Principles of Data Mining and Knowledge Discovery*, 107-119.
- [18] Corinna Cortes and Vladimir Vapnik (1995). Support-vector networks. *Machine Learning* **20(3)**, 273-297.
- [19] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* **39(1)**, 1-22.
- [20] Begüm Demir and Sarp Erturk (2007). Hyperspectral image classification using relevance vector machines. *IEEE Geoscience and Remote Sensing Letters* **4(4)**, 586-590.

- [21] Ernest Fokoué, Dongchu Sun, and Prem Goel (2011). Fully Bayesian analysis of the relevance vector machine with an extended hierarchical prior structure. *Statistical Methodology* **8(1)**, 83-96.
- [22] Y. Freund and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55 (1)**, 119-139.
- [23] Fung, G., and Mangasarian, O. L. (2001). Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software* **15(1)**, 29-44.
- [24] Y. Freund, R. Schapire, and N. Abe (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* **14**, 771-780.
- [25] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan (1999). Adacost: misclassification cost-sensitive boosting. *International Conference on Machine Learning*, 97-105.
- [26] Haixiang G., Yijing, L., Yanan, L., Xiao, L., and Jinling, L. (2016). BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. *Engineering Applications of Artificial Intelligence* **49**, 176-193.
- [27] X. Geng, Y.-Q. Zhu, and Z. Yang (2018). A novel classification method for class-imbalanced data and its application in microRNA recognition. *International Journal Bioautomation* **22(2)**.
- [28] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera (2016). Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets. *Information Sciences* **354**, 178-196.

- [29] Walter R Gilks and Pascal Wild (1992). Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41(2)**, 337-348.
- [30] Triguero, S. Gonzalez, J. M. Moyano, S. Garca, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesus, L. Snchez, and F. Herrera (2017). KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining. *International Journal of Computational Intelligence Systems* **10**, 1238-1249.
- [31] Casella, G., Robert, C. P., and Wells, M. T. (2004). Generalized accept-reject sampling schemes. *A Festschrift for Herman Rubin*, 342-347.
- [32] W. R. Gilks and P. Wild (1992). Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41(2)**, 337-348.
- [33] W. R. Gilks, N. G. Best, and K. K. C. Tan (1995). Adaptive rejection Metropolis sampling within Gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **44(4)**, 455-472.
- [34] C. Huang, Y. Li, C. L. Chen, and X. Tang (2019). Deep imbalanced learning for face recognition and attribute prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [35] Hastie, T., and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification and regression. *Advances in Neural Information Processing Systems*, 409-415.
- [36] P. Harrington (2012). Machine learning in action. *Manning Publications Co.*

- [37] Daniella Hubl, SBölte, S Feineis-Matthews, H Lanfermann, Andrea Federspiel, W Strik, F Poustka, and Thomas Dierks (2003). Functional imbalance of visual pathways indicates alternative face processing strategies in autism. *Neurology* **61(9)**, 1232-1237.
- [38] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano (2007), Experimental perspectives on learning from imbalanced data. *Proceedings of the 24th International Conference on Machine Learning*, 935-942.
- [39] M. Kubat, R. C. Holte, and S. Matwin (1998). Machine learning for the detection of oil spills in satellite radar images, *Machine Learning* **30 (2-3)**, 195-215.
- [40] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering* **30(1)**, 25-36.
- [41] R. Kohavi and F. Provost (1998). Glossary of terms special issue on applications of machine learning and the knowledge discovery process. *Machine Learning* **30**, 271-274.
- [42] Miroslav Kubat, Robert C Holte, and Stan Matwin (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning* **30(2-3)**, 195-215.
- [43] Heller, K., Teh, Y. W., and Gorur, D. (2009). Infinite hierarchical hidden Markov models. *Artificial Intelligence and Statistics*, 224-231.

- [44] A. J. Kinderman and J. F. Monahan (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software (TOMS)* **3(3)**, 257-260.
- [45] V. Lopez, A. Fernandez, S. Garcia, V. Palade, and F. Herrera (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences* **250**, 113-141.
- [46] W. Lee, C.-H. Jun, and J.-S. Lee (2017). Instance categorization by Support Vector Machines to adjust weights in AdaBoost for imbalanced data classification. *Information Sciences* **381**, 92-103.
- [47] V. Lopez, A. Fernandez, J. G. Moreno-Torres, and F. Herrera (2012). Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. *Expert Systems with Applications* **39 (7)**, 6585-6608.
- [48] C. Li, X. Ding and Y. Wu (2007). Revised AdaBoost algorithm-Ad AdaBoost [Chinese Mandrian]. *Jisuanji Xuebao/Chinese Journal of Computers* **30 (1)**, 103-109.
- [49] D. Lewis and W. Gale (1994). Training text classifiers by uncertainty sampling. *Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3-12.
- [50] Zunxiong, L., Deyun, Z., Qindong, S., and Zheng, X. (2004). Mid-term electric load prediction based on the relevant vector machine. *Journal of Xian Jiaotong University* **38(10)**, 1005-1008.

- [51] S. Makki, Z. Assaghir, Y. Taher, R. Haque, M.-S. Hacid, and H. Zeineddine (2019). An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access* **7**, 93010-93022.
- [52] M. O. Miah, S. S. Khan, S. Shatabda, and D. M. Farid (2019). Improving detection accuracy for imbalanced network intrusion classification using cluster-based under-sampling with random forests. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), IEEE*, 1-5.
- [53] Sikora M., Wrobel L. (2010). Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines. *Archives of Mining Sciences* **55(1)**, 91-114.
- [54] N Nikolaev and P Tino (2005). Sequential relevance vector machine learning from time series. *2005 IEEE International Joint Conference on Neural Networks* **2**, 1308-1313.
- [55] Ng, A. Y., and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 841-848.
- [56] Jerzy Neyman, Elizabeth L Scott, et al. (1948). Consistent estimates based on partially consistent observations. *Econometrica* **16(1)**, 1-32.
- [57] Mahesh Pal and Giles M Foody (2012). Evaluation of SVM, RVM and SMRL for accurate image classification with limited ground data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5(5)**, 1344-1355.

- [58] Rifkin, R. M. (2002). Everything old is new again: a fresh look at historical approaches in machine learning. *Doctoral dissertation in Massachusetts Institute of Technology*.
- [59] Rubinstein, Y. D., and Hastie, T. (1997). Discriminative vs Informative Learning. *International Conference on Knowledge Discovery and Data Mining* **5**, 49-53.
- [60] Rojas, R. (2009). AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting. *Freie University, Berlin, Tech. Rep.*
- [61] B. D. Ripley (1987). Regression techniques for the detection of analytical bias. *Analyst* **112(4)**, 377-383.
- [62] C. R. Ren (2001). Topics in Bayesian estimation: frequentist risks and hierarchical models for time to pregnancy. *Doctor of Philosophy Thesis, University of Missouri*.
- [63] Q. Song, Y. Guo, and M. Shepperd (2018). A comprehensive investigation of the role of imbalanced learning for software defect prediction. *IEEE Transactions on Software Engineering* **45 (12)**, 1253-1269.
- [64] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition* **40 (12)**, 3358-3378.
- [65] R. E. Schapire and Y. Singer (1999). Improved Boosting algorithms using confidence-rated predictions. *Machine Learning* **37 (3)**, 297-336.
- [66] Bernhard Schölkopf, Christopher JC Burges, and Alexander J Smola (1999). [B] Advances in kernel methods: support vector learning. *MIT Press*.

- [67] Catarina Silva and Bernardete Ribeiro (2006). Scaling text classification with relevance vector machines. *2006 IEEE International Conference on Systems, Man and Cybernetics* **5**, 4186-4191.
- [68] G. Schwarz (1978). Estimating the dimension of a model. *The Annals of Statistics* **6(2)**, 461-464.
- [69] X. Tao, Q. Li, W. Guo, C. Ren, C. Li, R. Liu, and J. Zou (2019). Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences* **487**, 31-56.
- [70] Shawe-Taylor, J., and Cristianini, N. (2004). Kernel methods for pattern analysis. *Cambridge University Press*.
- [71] Michael E Tipping (2000). The relevance vector machine. *The Advances in Neural Information Processing Systems*, 652-658.
- [72] Michael E Tipping (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* **1(Jun)**, 211-244.
- [73] Michael E Tipping and Anita C Faul (2003). Fast marginal likelihood maximization for sparse Bayesian models. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.
- [74] Arasanathan Thayananthan (2006). Template-based pose estimation and tracking of 3D hand motion. *Ph.D. Thesis, University of Cambridge*.
- [75] Vapnik, V. (1998). The support vector method of function estimation. *Nonlinear Modeling*, 55-85.

- [76] P. Viola and M. Jones (2002). Fast and robust classification using asymmetric AdaBoost and a detector cascade. *Advances in Neural Information Processing Systems*, 1311-1318.
- [77] G. M. Weiss (2004). Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter* **6 (1)**, 7-19.
- [78] Dinghai, W., Peilin, Z., and Yingtang, Z. (2011). Study on diesel engine faults diagnosis based on time frequency singular value spectrum and RVM. *Journal of Mechanical Strength* **33(3)**, 317-323.
- [79] Xiang-min, X., Yun-feng, M., Jia-ni, X., and Feng-le, Z. (2007). Classification performance comparison between RVM and SVM. *2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID)*, 208-211.
- [80] Yue, S. (2017). Imbalanced malware images classification: a CNN based approach. *arXiv:1708.08042*.
- [81] B. Zhou, T. Wang, M. Luo, and S. Pan (2017). An online tracking method via improved cost-sensitive AdaBoost. *2017 Eighth International Conference on Intelligent Control and Information Processing (ICICIP)*, 49-54.
- [82] Dong Zhang (2017). Adaptive sampling algorithms and its R package development. *Master Thesis, East China Normal University*.

VITA

Wenyang Wang was born on March 16, 1991, in Dalian, Liaoning Province of China. He graduated with a Bachelor of Science in Mathematics from the Dalian University of Technology in the summer of 2014. In the Fall of 2014, he joined a graduate program in the Department of Statistics at the University of Missouri, Columbia, USA. In the summer of 2016, he received a Master of Arts in Statistics and began his doctoral study with his advisors, Drs. Dongchu Sun and Zhuoqiong He. He has accepted a faculty position at Dalian Maritime University, China.

He was married to Jingwen Wang in December of 2018. Their daughter, Rita Xuehan Wang, was born in December 2019.