

การตั้งค่าความผิดพลาดใหม่แบบวนซ้ำเพื่อปรับปรุงฟังก์ชันค่าใช้จ่ายของโครงข่ายประสาท
สำหรับการเรียนรู้ข้อมูลแบบไม่ดูล

นายพิรศุขม์ รุ่งจรัสแสง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2554

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)

เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย

The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository (CUIR)

are the thesis authors' files submitted through the Graduate School.

ITERATIVE ERROR RE-ESTABLISHMENT TO IMPROVE NEURAL NETWORK
COST FUNCTION FOR IMBALANCED DATA LEARNING

Mr. Perasut Rungcharassang

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Applied Mathematics and Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2011

Copyright of Chulalongkorn University

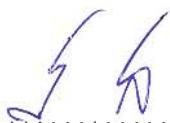
Thesis Title Iterative Error Re-establishment to improve Neural Network
 Cost Function for Imbalanced Data Learning
By Mr. Perasut Rungcharassang
Field of Study Applied Mathematics and Computational Science
Thesis Advisor Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial
Fulfillment of the Requirements for the Master's Degree



..... Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE



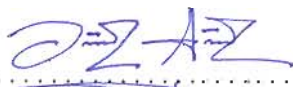
..... Chairman
(Assistant Professor Krung Sinapiromsaran, Ph.D.)



..... Thesis Advisor
(Professor Chidchanok Lursinsap, Ph.D.)



..... Examiner
(Associate Professor Suchada Siripant)



..... External Examiner
(Chularat Tanprasert, Ph.D.)

พีรศุภมภ์ รุ่งจรัสแสง : การตั้งค่าความผิดพลาดใหม่แบบวนซ้ำเพื่อปรับปรุงฟังก์ชันค่าใช้จ่ายของโครงข่ายประสาทสำหรับการเรียนรู้ข้อมูลแบบไม่ดุล. (Iterative Error Re-establishment to Improve Neural Network Cost Function for Imbalanced Data Learning) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร. ชิดชนก เหลือสินทรัพย์, 38 หน้า.

ในงานวิจัยนี้ได้นำเสนอ ขั้นตอนวิธีการสอนใหม่เพื่อเพิ่มความแม่นยำของกลุ่มส่วนน้อยสำหรับปัญหาการเรียนรู้ข้อมูลไม่ดุล ขั้นตอนวิธีนี้ได้จากการสังเกตว่าสาเหตุของค่าความแม่นยำต่ำเนื่องจากอิทธิพลของพจน์ค่าคลาดเคลื่อน ซึ่งคำนวณได้จากผลต่างของเป้าหมายกับข้อมูลส่งออกที่แท้จริงทั้งหมดยกกำลังสอง โดยข้อมูลในกลุ่มส่วนน้อยเหล่านั้นในฟังก์ชันค่าใช้จ่าย เพื่อแก้ปัญหานี้ฟังก์ชันค่าใช้จ่ายจะถูกตั้งค่าใหม่ที่แต่ละรอบขึ้นอยู่กับค่าผิดพลาดของข้อมูลในกลุ่มส่วนน้อยและกลุ่มส่วนมาก ข้อมูลใดๆที่คำนวณค่าผิดพลาดได้น้อยกว่า 0.05 จะถูกคัดออกไปจากฟังก์ชันค่าใช้จ่าย ข้อมูลที่เหลือให้ใส่กลับเข้าไปในฟังก์ชันค่าใช้จ่าย ขั้นตอนวิธีการสอนใหม่นี้ถูกเปรียบเทียบกับวิธีเลเวนเบิร์ก-มาร์ควอดท์และวิธีลาโมบุสท์ บนชุดข้อมูลมาตรฐาน 15 ชุด จากผลการทดลองแสดงให้เห็นถึงความแม่นยำของวิธีการนี้ที่สูงขึ้นกว่าวิธีการอื่นๆใน 13 ตัวอย่าง พร้อมทั้งมีความเร็วในการสอนเร็วขึ้นอีกด้วย

ภาควิชา คณิตศาสตร์ ลายมือชื่อนิสิต *พีรศุภมภ์ รุ่งจรัสแสง*
 และวิทยาการคอมพิวเตอร์
 สาขาวิชา คณิตศาสตร์ประยุกต์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก *C. L. S.*
 และวิทยาการคณนา
 ปีการศึกษา 2554

5272462723 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE


KEYWORDS : NEURAL NETWORK / LM / IMBALANCED DATA / COST FUNCTION

PERASUT RUNGCHARASSANG : ITERATIVE ERROR RE-ESTABLISHMENT TO IMPROVE NEURAL NETWORK COST FUNCTION FOR IMBALANCED DATA LEARNING. ADVISOR : PROF. CHIDCHANOK LURSINSAP, Ph.D., 38 pp.

A new training algorithm to enhance the accuracy of minority class in imbalanced data learning problem was proposed. This algorithm is based on the observation that the cause of lower accuracy is due to the domination of the error terms, i.e. the square of difference between the target and the actual output, computed by those data in majority class in the cost function. To resolve this domination, our cost function is re-established at each epoch based on the errors of the data in minority and majority classes. Any datum whose corresponding term in the cost function produces an error less than 0.05 is removed from cost function. Otherwise, it is put back into the cost function. Our algorithm adopting multilayer perceptron and Levenberg-Marquardt (LM) as the learning algorithm was compared with classical LM and the recent algorithm RAMOBoost on 15 well-known benchmarks. The experimental results of our approach produced higher accuracy than the other approaches in 13 cases with faster training speed.

Department : .. Mathematics and Student's Signature : 

.. Computer Science

Field of Study : .. Applied Mathematics Advisor's Signature : 

.. And Computational Science

Academic Year : 2011

ACKNOWLEDGEMENTS

In the completion of my Master Thesis, I am very grateful to Professor Dr. Chidchanok Lursinsap, my advisor for his advice and ideas. Without his great suggestion, this study could not be completed.

Sincere thanks to Associate Professor Suchada Siripant, Assistant Professor Dr. Krung Sinapiromsaran, and Dr. Chularat Tanprasert, my thesis committee, for their good questions and suggestions. Furthermore, I would like to give my sincere thanks to Assistant Professor Dr. Montri Maleewong from Kasetsart University for recommended to further study at Chulalongkorn University.

I would like to thank Graduate Study Scholarship, Faculty of Science, Chulalongkorn University for partial support of this work and Dr. Sheng Chen for an original code of RAMOBoost algorithm.

Moreover, I would like to thank my friends and groups in Advanced Virtual and Intelligent Computing (AVIC) Center for their support and giving me some important advice about programming in my thesis. Last but not least, I am very grateful to my family who is the most important supporter.

CONTENTS

	Page
ABSTRACT IN THAI	iv
ABSTRACT IN ENGLISH	v
ACKNOWLEDGEMENTS	vi
CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER	
I INTRODUCTION	1
1.1 Imbalanced Problem	1
1.2 Scope of The Study	2
1.3 Thesis Overview	2
II RELATED WORKS	4
2.1 Sampling Techniques	4
2.2 Synthetic Minority Over-sampling Technique (SMOTE)	4
2.3 Ranked Monority Over-sampling in Boosting (RAMOBoost)	5
2.4 Levenberg-Marquardt Algorithm (LM)	6

III RE-ESTABLISHED COST FUNCTION TRAINING ALGORITHM (ReCoFT)	8
3.1 Brief Description about ReCoFT	8
3.2 Relation between The Number of Hidden Neurons and Pattern Re- moved	12
3.3 Concept of ReCoFT Algorithm in Two Dimensions	14
IV EXPERIMENTS AND RESULTS	19
4.1 Performance Measures	19
4.2 Data Set Description	20
4.3 Selection Ratio between Training and Testing	20
4.4 Experiments and Results	20
4.5 Comparison of Training Time	23
4.6 Removed Pattern Description	24
4.7 The Number of Removed Data	25
V CONCLUSION	36
REFERENCES	37
VITA	38

LIST OF TABLES

Table		Page
4.1	The confusion matrix and the meanings of TP, FP, TN, and FN. . .	19
4.2	The original data sets are set into the imbalanced data sets in order to use in the experiments. The same as the data in RAMOBoost, 2010.	21
4.3	The number of removed and retained data at the final epoch. . . .	22
4.4	The comparison of the average training time (seconds) of each algorithm.	24
4.5	The interval of the number of hidden neurons that help considering.	25
4.6	The digits in the lower ($max^{(minor)}$) and the upper boxes ($max^{(major)}$)	26

LIST OF FIGURES

Figure	Page
3.1	The interval of 0.05 of the output value of temporarily removed patterns based on the sigmoid function. 9
3.2	(a) The initial weights are randomly set . (b) Areas are consistent with the pattern removed conditions. 15
3.3	(a) Shows pattern removed following the condition in algorithm. (b) Uses the remaining pattern to adjust the new weights. (c) Shows the original pattern with the new weights. 16
3.4	(a) Shows the removed pattern responding to the weights. (b) Adjusts the weights from retained pattern. (c) Shows the original pattern with the new weights. 17
3.5	The concept of ReCoFT in two dimensions. Patterns p and q represent the data in majority and minority classes, respectively. (a) Shows pattern removed responding to the weights. (b) Adjusts the weights from retained pattern again. (c) Shows the original pattern with the final weights that can separate patterns p and q, finishes this algorithm. 18
4.1	The comparison of class accuracy between 70:30 and 50:50 from the algorithm (ReCoFT) for each data set. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes. 27
4.2	The number of removed data at different epochs during the training session of each data set. (a) Spambase. (b) Wine. (c) German. . . . 28
4.3	The number of removed data at different epochs during the training session of each data set. (a) Vehicle. (b) Segment. (c) Page. 29
4.4	The number of removed data at different epochs during the training session of each data set. (a) Satimage. (b) Vowel. (c) Abalone. . . . 30

Figure	Page
4.5 The number of removed data at different epochs during the training session of each data set. (a) Glass. (b) Yeast. (c) Letter.	31
4.6 The number of removed data at different epochs during the training session of Shuttle data set.	32
4.7 The percentage of the number of data removed from the algorithm (ReCoFT) for each data set at a final epoch. (a) Each class. (b) Number of cut data in every class.	33
4.8 The comparison of class accuracy from the algorithm (ReCoFT), LM, and RAMOBoost for each data set. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes.	34
4.9 The comparison of class accuracy from the algorithm (ReCoFT) and RAMOBoost for eight data sets. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes. .	35

CHAPTER I

INTRODUCTION

1.1 Imbalanced Problem

Classifying a given set of data into two groups based on their targets is the most common problem in several applications such as pattern recognition, prediction, and machine intelligence. In case of multi-target problem, the problem itself can be transformed into the problem of classifying data into two groups. In this study, the binary classification problem is concentrated. Generally, the accuracy of classification depends on cross-validation and measuring the learning performance in terms of a cost function during 'training session' so that the learning system can achieve a high testing accuracy.

In real applications such that the application of classification in bioinformatics, the number of data in both classes is not equal. The class with larger amount of data is called *the majority class* and the other is called *the minority class*. For example, the amount of data of patients having cancer is usually less than the amount of data of healthy patients. Obviously, the accuracy of classification of the larger class is always higher than that of the smaller class because the cost function deployed during 'training session' measures the difference between the actual outputs and targets of both the minority and the majority classes. Hence, the error from the larger class must obviously dominate the error of the smaller class. This problem has been studied under the name of *imbalanced problem*. There are few solutions to this problem [1] [2] [3] [4] [5]. These approaches were based on the concepts of over-sampling and under-sampling of the training data set.

In several papers, *the minority class* and *the majority class* are called *the positive class* and *the negative class*, respectively. The ratio between the minority and the majority classes may have several ranges from 49:51, 1:100 may be to 1:100,000.

This study proposes *Re-established Cost Function Training Algorithm* (ReCoFT) as a new training algorithm. The concept started from a simple idea that the correctly predicted data will not be used in the learning algorithm in the next iteration. Only the incorrectly predicted data are retained. ReCoFT reduces terms for calculating the cost function and also eliminates the unnecessary data to the learning algorithm. The strategy obviously differs from the others' [1] [2] [3] [4]. Because those methods are based on re-sampling and the data are randomly increased or decreased during the learning algorithm. But ReCoFT temporarily and dynamically removes some data during the training session under determined conditions. The number of removed data depends on the cost function and ReCoFT's conditions. Implicitly, when the number of input neurons is reduced, the running time is also reduced.

1.2 Scope of The Study

In this study, the performance of the algorithm was evaluated on 15 well-known data sets described in Chapter IV. Each data set was modified into the binary classification (the minority and the majority classes) and randomly divided into two parts, 70% of the minority and the majority classes were for training and the other 30% data were for testing. The experiments used a 3-layer neural network with one output and Levenberg-Marquardt as the learning algorithm. The sigmoid function was used as the activation function.

The objective of the study is to present a new training algorithm to enhance the accuracy of minority class in the imbalanced problem.

1.3 Thesis Overview

Chapter II summarizes the related works to deal with the imbalanced problem. Levenberg-Marquardt algorithm is proposed as the learning algorithm in the ex-

periments. Chapter III explains the concept of the proposed algorithm and the conditions for re-establishing the cost function. Chapter IV discusses the experiments and results. Chapter V concludes the study.

CHAPTER II

RELATED WORKS

2.1 Sampling Techniques

Over-sampling and under-sampling are techniques in order to adjust a balanced distribution of data. Over-sampling is randomly increasing data in minority class to balance distribution. Under-sampling is randomly decreasing data in majority class in order to balance distribution. But both techniques have some problem because the over-sampling ends up as an over-fitting for classification. There is a possibility to remove the important data in majority class for the under-sampling.

2.2 Synthetic Minority Over-sampling Technique (SMOTE)

One of the over-sampling techniques widely applied is SMOTE [1] [2]. SMOTE is the pre-processing process to deal with the original data set in order to balance the class distribution between the minority and the majority classes. After that, the balanced data set is used to classify with classifiers. SMOTE uses a uniform distribution to synthesize data in the minority class. This method finds the k -nearest neighbor of each data in the minority class and generates the position of synthetic samples on the line segments between an original data in the minority class and its nearest neighbor.

2.3 Ranked Minority Over-sampling in Boosting (RAMOBoost)

RAMOBoost [3] combines the over-sampling technique of ADASYN [4] and a boosting technique. SMOTE uses a uniform distribution to over-sample synthetic data in the minority class but ADASYN uses a density distribution to over-sample data in the minority class. This implied that the over-sampling in each k -nearest group is not equal. RAMOBoost adapts synthetic data generation in a learning algorithm by considering the ratio around nearest neighbors of each minority data from training set. This method does not generate data for the minority class if there are not the majority data in their k -nearest neighbors. If there are many majority data in that group, minority data will be proportionally generated. In part of a boosting technique, RAMOBoost uses an iterative learning in order to adjust the weights at each boosting iteration to shift the decision boundary between the minority and the majority classes.

The objective of RAMOBoost is to reduce an error from the imbalanced data and to adapt the input data from the data distribution. RAMOBoost algorithm is divided into two parts. First part is a data modification, the input of data sets is normalized in the interval $[0,1]$. After that, find the k -nearest neighbors of the minority class following ADASYN algorithm to generate data for the minority class. Second part is a learning algorithm. The modified data from the first part are used to be the input of the classifier for adjusting the weights of the networks.

RAMOBoost uses multilayer perceptron (MLP) as the learning classifier. The MLP is set as follows the number of hidden neurons is set to four. RAMOBoost studies on the binary classification problem. The number of output neurons is set to two. The activation function uses the sigmoid function.

RAMOBoost's experiments use 19 data sets from UCI machine learning repository [6] and ELENA project. Since the original data sets are multi-class, they were modified into 2-class imbalanced data.

From above, RAMOBoost and this algorithm (ReCoFT) use MLP as the learning classifier and also concentrate on the binary classification problem. Thus

the performance of ReCoFT will be compared with RAMOBoost and classical Levenberg-Marquardt.

2.4 Levenberg-Marquardt Algorithm (LM)

LM [7] is an algorithm for adjusting the weights based on the Jacobian matrix of an error between its target and its actual output. Generally, the cost function for evaluating the training performance is defined as follows.

$$F(\mathbf{w}) = \sum_{p=1}^P \left[\sum_{k=1}^K (t_{kp} - o_{kp})^2 \right] \quad (2.1)$$

F is defined as a function of the sum square error with respect to all weights $\mathbf{w} = [w_1 w_2 \dots w_N]^T$. N is the number of the weights. P is the number of the patterns. K is the number of the output nodes. t_{kp} is a target and o_{kp} is an actual output value. If the network has only one output neuron, then the cost function can be shortly rewritten as in equation (2.2)

$$F(\mathbf{w}) = \sum_{p=1}^P (t_p - o_p)^2 \quad (2.2)$$

$F(\mathbf{w})$ can be written in terms of the following dot product of error vectors.

$$F(\mathbf{w}) = \mathbf{E}^T \mathbf{E} \quad (2.3)$$

where $\mathbf{E} = [e_1 e_2 \dots e_P]^T$ and $e_p = t_p - o_p$ where $p = 1, \dots, P$. From equation (2.3), its Jacobian matrix is defined as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \dots & \frac{\partial e_1}{\partial w_N} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \dots & \frac{\partial e_2}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_P}{\partial w_1} & \frac{\partial e_P}{\partial w_2} & \dots & \frac{\partial e_P}{\partial w_N} \end{bmatrix} \quad (2.4)$$

All weights can be adjusted by using the Jacobian matrix \mathbf{J} and the error vector \mathbf{E} in the following equation.

$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - (\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I})^{-1} \mathbf{J}^T \mathbf{E} \quad (2.5)$$

Here, α is a learning rate and \mathbf{I} is an identity matrix. $\mathbf{w}^{(new)}$ and $\mathbf{w}^{(old)}$ are the current values of the weights and the values after being updated, respectively. To reduce the oscillation effect, Amir and his team [8] proposed a method to adjust the value of α by using the following equations.

$$\alpha^{(new)} = \begin{cases} \alpha^{(old)} \cdot \beta & \text{if } F(\mathbf{w}^{(new)}) < F(\mathbf{w}^{(old)}) \\ \alpha^{(old)} / \beta & \text{otherwise} \end{cases} \quad (2.6)$$

β is a constant and $0 < \beta \leq 1$.

CHAPTER III

RE-ESTABLISHED COST FUNCTION TRAINING ALGORITHM (ReCoFT)

3.1 Brief Description about ReCoFT

Since the cause of imbalanced problem is due to the unequal numbers of terms in the minority and the majority classes, some terms belonging to the majority class must be temporarily and dynamically removed as well as re-included in the cost function during the training session. By performing these processes, the activation value from the majority class will be expected not to dominate the value from the minority class and the separating hyperplanes located will be pulled towards the space in between two classes.

When starting the training session, all data are involved in the cost function. At each epoch, the learning algorithm retains some minimum amount of data so that the separating hyperplanes in forms of vectors can be fixed in the space without any non-deterministic elements in the vectors. Each datum is selectively and temporarily removed from the cost function. A new cost function based on the remaining data is re-established to adjust all weights. A datum is removed from the cost function if the difference between its computed output value and its target is greater than a defined constant.

A 3-layer perceptron with LM as the learning algorithm with one output is used in this study. Let o_p and t_p be the output and the target of the input pattern p , respectively. Pattern p is removed or retained from the cost function under the following conditions.

1. **If** $(t_p - o_p)^2 \geq 0.05^2$, **then** retain pattern p in the cost function in equation (2.2).

2. **If** $(t_p - o_p)^2 < 0.05^2$, **then** remove pattern p in the cost function in equation (2.2).

Since the sigmoid function is used as the activation function. The meaning of 0.05 of the prediction error is illustrated in Fig 3.1. It shows the interval of the output value of temporarily removed patterns and retained patterns.

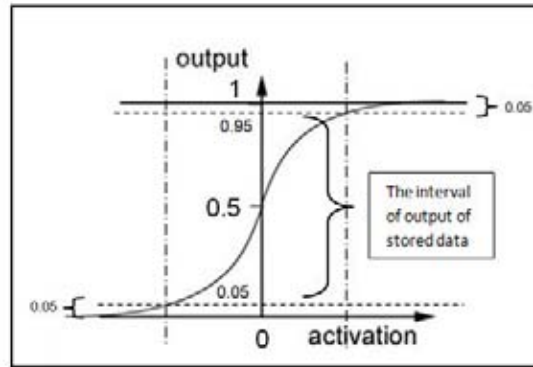


Figure 3.1: The interval of 0.05 of the output value of temporarily removed patterns based on the sigmoid function.

The minimum number of data in each class that must be preserved in order to bind a separating hyperplane in between two classes can be derived from the following observation. Any n -dimensional hyperplane can be represented by this equation.

$$a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_n \cdot x_n + b = 0 \quad (3.1)$$

where b is a constant and a_i is weight for input x_i . However, equation (3.1) can be rewritten by dividing each term with a_1 . Define $a'_{i-1} = \frac{a_i}{a_1}$, $2 \leq i \leq n$ and $b' = \frac{b}{a_1}$. Hence, equation (3.1) becomes

$$x_1 + a'_1 \cdot x_2 + \cdots + a'_{n-1} \cdot x_n + b' = 0 \quad (3.2)$$

To find the values of a'_i , for $1 \leq i \leq n-1$, and b' , at least n data points are needed from each class to fix the location of each hyperplane. The constant n is equal to the dimensions of data space. Let H be the number of hidden neurons. The

number of data that must be retained in each class during the training session is set to $n \cdot H$. Suppose the number of data in training minority class is equal to m and that of training majority class is equal to M at the beginning. Let $min^{(minor)}$ and $min^{(major)}$ be the minimum number of retained data in minority and majority classes, respectively. The detail of the proposed algorithm is given as follows.

Re-established Cost Function Training Algorithm

1. Initialize the values of α , β , H , and \mathbf{w} .
2. Let $\mathbf{D}^{(minor)}$ denote a set of data in minority class being involved.
3. Let $\mathbf{D}^{(major)}$ denote a set of data in majority class being involved.
4. Let $\mathbf{D}^{(temp)}$ denote an empty set.
5. Put all data in minority class in $\mathbf{D}^{(minor)}$.
6. Put all data in majority class in $\mathbf{D}^{(major)}$.
7. Set $min^{(minor)} = min^{(major)} = n \cdot H$.
8. Compute $F(\mathbf{w})$ and adjust \mathbf{w} only once.
9. **If** $m < n \cdot H$ **then**
10. set $min^{(minor)} = m$.
11. **If** $M < n \cdot H$ **then**
12. set $min^{(major)} = M$.
13. **For** $1 \leq i \leq max_iteration$ **do**
14. Set $\mathbf{D}^{(temp)} = \mathbf{D}^{(minor)}$.
15. **If** $|\mathbf{D}^{(minor)}| \geq min^{(minor)}$ **then**
16. Set $\mathbf{D}^{(minor)} = \phi$.
17. **For** each pattern $p \in$ minority class **do**
18. **If** $(t_p - o_p)^2 \geq 0.05^2$ **then**
19. Put pattern p into $\mathbf{D}^{(minor)}$.
20. **Endif**
21. **Endfor**

22. **If** $|\mathbf{D}^{(minor)}| \geq \min^{(minor)}$ **then**
 23. $\mathbf{D}^{(minor)} = \mathbf{D}^{(temp)}$.
 24. **Endif**
 25. **Endif**
 26. Set $\mathbf{D}^{(temp)} = \mathbf{D}^{(major)}$.
 27. **If** $|\mathbf{D}^{(major)}| \geq \min^{(major)}$ **then**
 28. Set $\mathbf{D}^{(major)} = \phi$.
 29. **For** each pattern $p \in$ majority class **do**
 30. **If** $(t_p - o_p)^2 \geq 0.05^2$ **then**
 31. Put pattern p into $\mathbf{D}^{(major)}$.
 32. **Endif**
 33. **Endfor**
 34. **If** $|\mathbf{D}^{(major)}| \geq \min^{(major)}$ **then**
 35. $\mathbf{D}^{(major)} = \mathbf{D}^{(temp)}$.
 36. **Endif**
 37. **Endif**
 38. Re-establish the cost function $F(\mathbf{w})$ based on
 39. data in $\mathbf{D}^{(minor)}$ and $\mathbf{D}^{(major)}$.
 40. Compute $\Delta \mathbf{w} = -(\mathbf{J}^T \mathbf{J} + \alpha \mathbf{I})^{-1} \mathbf{J}^T \mathbf{E}$.
 41. **If** $F(\mathbf{w}^{(old)} + \Delta \mathbf{w}) < F(\mathbf{w}^{(old)})$ **then**
 42. $\mathbf{w}^{(new)} = \mathbf{w}^{(old)} + \Delta \mathbf{w}$.
 43. Adjust $\alpha^{(new)} = \alpha^{(old)} \cdot \beta$.
 44. Set $\mathbf{w}^{(old)} = \mathbf{w}^{(new)}$.
 45. **Elseif** $F(\mathbf{w}^{(old)} + \Delta \mathbf{w}) > F(\mathbf{w}^{(old)})$ **then**
 46. Adjust $\alpha^{(new)} = \alpha^{(old)} / \beta$.
 47. **Elseif** $F(\mathbf{w}^{(old)} + \Delta \mathbf{w}) = F(\mathbf{w}^{(old)})$ **then**
 48. Terminate learning process.
 49. **Endif**
 50. **Endfor**

The cost function in equation (2.2) is re-established during the learning process. Let $\mathbf{D} = \mathbf{D}^{(minor)} \cup \mathbf{D}^{(major)}$. The re-established cost function can be rewritten in terms of sets $\mathbf{D}^{(minor)}$ and $\mathbf{D}^{(major)}$ as follows.

$$F(\mathbf{w}) = \sum_{p \in \mathbf{D}} (t_p - o_p)^2 \quad (3.3)$$

3.2 Relation between The Number of Hidden Neurons and The Removed Patterns

From ReCoFT algorithm, step 9-12 state that if $m < n \cdot H$ then set $min^{(minor)} = m$ otherwise set $min^{(minor)} = n \cdot H$ and if $M < n \cdot H$ then set $min^{(major)} = M$ otherwise set $min^{(major)} = n \cdot H$. Consequently, ReCoFT algorithm obtains that

$$min^{(minor)} = \min\{m, n \cdot H\} \quad (3.4)$$

and

$$min^{(major)} = \min\{M, n \cdot H\} \quad (3.5)$$

Definitely, $m < M$ and $n \neq 0$. The interval of the number of hidden neurons is divided into 3 cases as follows.

Case 1 Let $m > n \cdot H$

$$\because M > m > n \cdot H$$

$$\therefore M > n \cdot H$$

$$H < \frac{m}{n}$$

$$min^{(minor)} = n \cdot H$$

$$min^{(major)} = n \cdot H$$

Case 2 Let $M < n \cdot H$

$$\because m < M < n \cdot H$$

$$\therefore m < n \cdot H$$

$$H > \frac{M}{n}$$

$$\min^{(minor)} = m$$

$$\min^{(major)} = M$$

Case 3 Let $m < n \cdot H$

$$\text{and } M > n \cdot H$$

$$\frac{m}{n} < H < \frac{M}{n}$$

$$\min^{(minor)} = m$$

$$\min^{(major)} = n \cdot H$$

Since $\min^{(minor)}$ and $\min^{(major)}$ are the minimum numbers of retained data in the minority and the majority classes, respectively. If they are equal to m and M , respectively, it means that no patterns in any class will be removed. When H is selected by $H < \frac{m}{n}$ (Case 1), it is possible to remove pattern both the minority and the majority classes. Carefully, H should not be selected by $H > \frac{M}{n}$ (Case 2) because there are no patterns in the minority and the majority classes which are removed in the learning algorithm. Removing pattern is the most important part in this algorithm (ReCoFT). If H is selected by $\frac{m}{n} < H < \frac{M}{n}$ (Case 3), the algorithm will not remove any pattern in the minority class. But it is possible to remove pattern in the majority class.

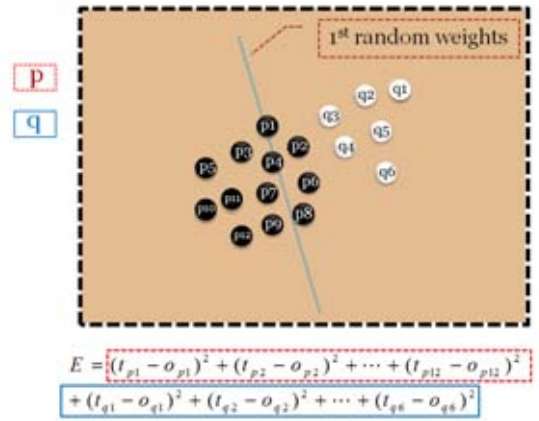
Notice that in Case 3, when H is selected by $H = \lceil \frac{m}{n} \rceil$, this algorithm obtains the number of removed data in the majority class as least as possible. And when H is selected by $H = \lfloor \frac{M}{n} \rfloor$, it obtains the number of removed data in majority class as much as possible.

3.3 Concept of ReCoFT Algorithm in Two Dimensions

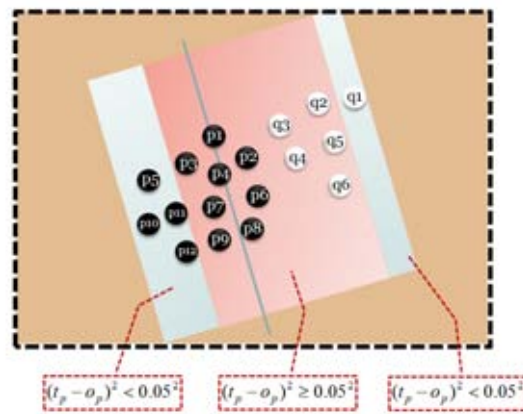
To explain the concept of ReCoFT algorithm, an example in two dimensions is used to illustrate how the algorithm works. All data points and their weights in Fig 3.2 are assumed without experimental values. There are 6 data in the minority class and 12 data in the majority class in the assumed data set.

The data in the majority class are named as p1 to p12 and the data in the minority class are named as q1 to q6. The terms of the cost function are presented under the figures. In Fig 3.2(a), the terms of the cost function in the red dash box are the terms of the majority class and in the blue line box are the terms of the minority class. In two dimensions, the weights of network that connect between the input layer and the hidden layer are in linear equations. The number of hyperplanes is equal to the number of hidden neurons in two dimensions. In this illustration, the number of hidden neurons is set to one (one straight line).

Firstly, the initial weights are randomly set as shown in Fig 3.2(a). There are 18 data being involved in the cost function at the first times. Fig 3.2(b) shows the assumed areas, pink and blue areas, under the condition for re-establishing the cost function. The data in the pink area are retained and used in the learning algorithm in the next iteration. So the data in the blue areas are removed as shown in Fig 3.3(a). After the data are temporarily removed, the cost function becomes to 13 terms. The remaining data will be trained in the next iteration in order to adjust the weights with LM algorithm shown in Fig 3.3(b). In next step in Fig 3.3(c), the error for the updated weights with respect to the original data (all 18 data) is computed. In Fig 3.4(a), there are 8 temporarily removed data. Therefore the cost function reduces to 10 terms such that 6 terms are from majority class and 4 terms are from minority class. In Fig 3.4(b), the new weights are calculated from the retained data. Then, finding an error for the updated weights with respect to the original data is shown in Fig 3.4(c). After that, the algorithm removes 11 terms out of 18 terms that are correctly predicted following the conditions. Consequently, the new cost function in this iteration remains 7 terms as shown in Fig 3.5(a). Fig 3.5(b) and Fig 3.5(c) show the two final iterations.

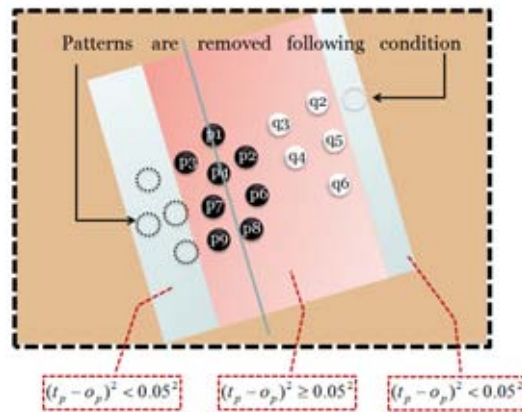


(a)

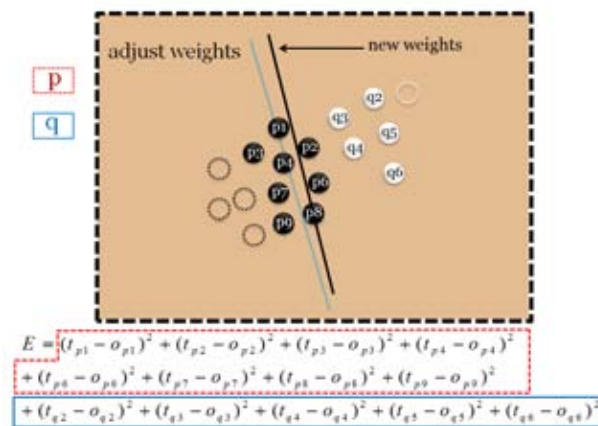


(b)

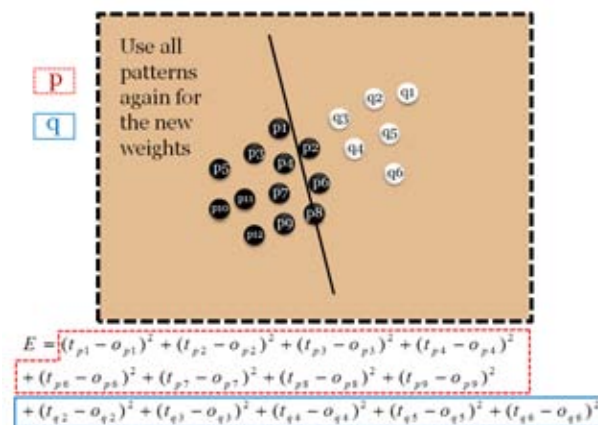
Figure 3.2: (a) The initial weights are randomly set. (b) Areas are consistent with the pattern removed conditions.



(a)

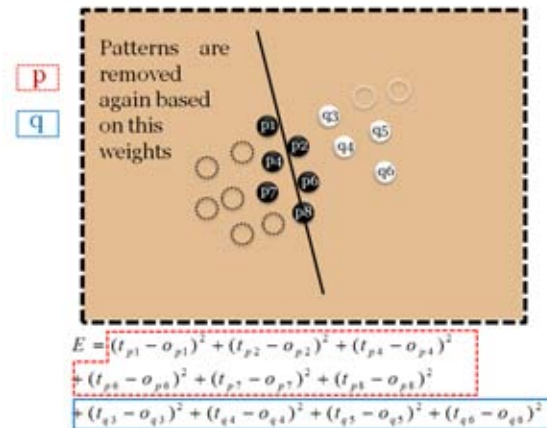


(b)

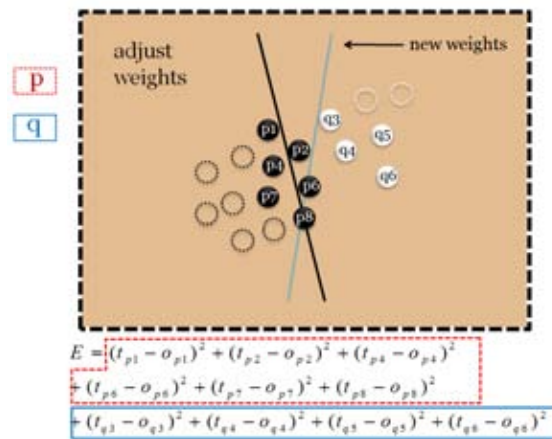


(c)

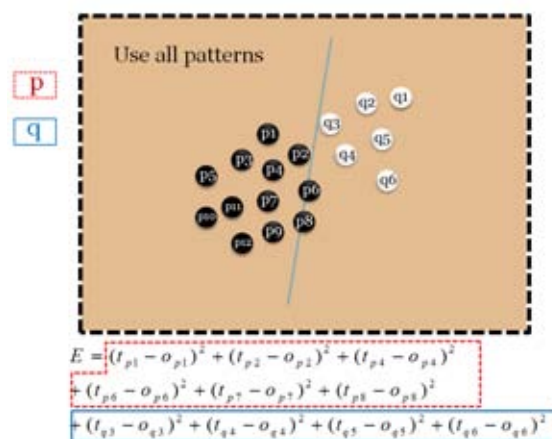
Figure 3.3: (a) Shows pattern removed following the condition in algorithm. (b) Uses the remaining pattern to adjust the new weights. (c) Shows the original pattern with the new weights.



(a)

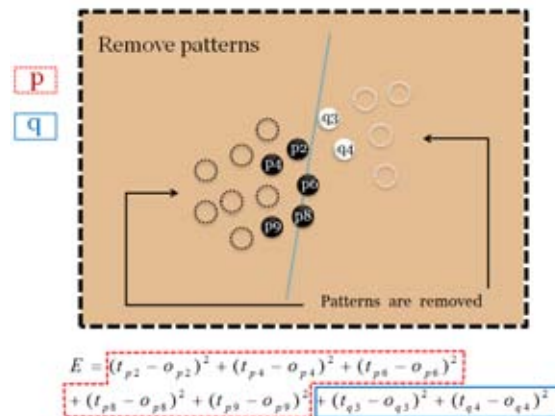


(b)

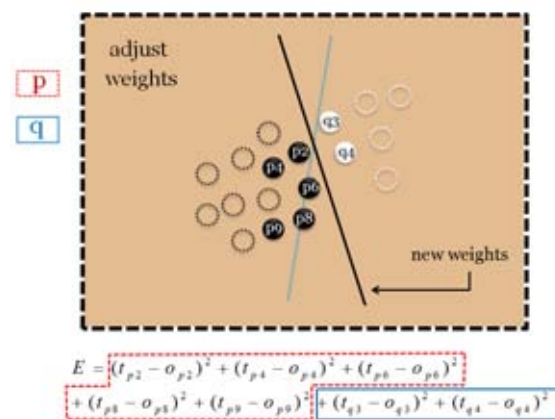


(c)

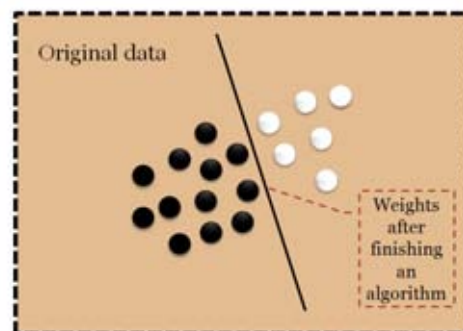
Figure 3.4: (a) Shows the removed pattern responding to the weights. (b) Adjusts the weights from retained pattern. (c) Shows the original pattern with the new weights.



(a)



(b)



(c)

Figure 3.5: The concept of ReCoFT in two dimensions. Patterns p and q represent the data in majority and minority classes, respectively. (a) Shows pattern removed responding to the weights. (b) Adjusts the weights from retained pattern again. (c) Shows the original pattern with the final weights that can separate patterns p and q, finishes this algorithm.

CHAPTER IV

EXPERIMENTS AND RESULTS

4.1 Performance Measures

The confusion matrix shown in Table 4.1 is used in order to measure the performance of each classifier. TP, FP, TN and FN are the number of predictions and their meanings as the following

TP means *minority class* was **correctly** classified as *minority class*.

FP means *majority class* was **incorrectly** classified as *minority class*.

TN means *majority class* was **correctly** classified as *majority class*.

FN means *minority class* was **incorrectly** classified as *majority class*.

Table 4.1: The confusion matrix and the meanings of TP, FP, TN, and FN.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The accuracy of correct prediction of each class (the minority and the majority classes) was used to measure an efficiency in each methods.

The percentage of prediction minority correctly is defined as

$$\frac{TP}{TP + FN} * 100$$

The percentage of prediction majority correctly is defined as

$$\frac{TN}{TN + FP} * 100$$

The percentage of prediction both classes correctly is defined as

$$\frac{TP + TN}{TP + TN + FP + FN} * 100$$

4.2 Data Set Description

The data sets in the experiments were summarized in Table 4.2. 15 data sets from UCI machine learning repository [6] were used to compare the performance of each method in the experiments. There are many classes in the original data sets. Thus, the original data of each data were transformed into the imbalanced data. n represents the dimensions of each data set. The next column represents the number of the original data (or the number of pattern). The others represent the number of data in the minority and the majority classes, respectively.

4.3 Selection Ratio between Training and Testing

Importantly, each data set should not be divided less than 50% for training. So the ratio between training and testing were considered to compare their performance. From Fig 4.1, the left bars represent the performance of each data set whose 70% for training and 30% for testing. The other bars belong to data set whose 50% for training and 50% for testing. Fig 4.1(a), (b) and (c) show the accuracy of the minority, the majority and all classes, respectively. The ratio is set to 70:30 because the experiments show their performance are better than the 50:50 ratio.

4.4 Experiments and Results

From Section 4.3, 70% of data were for training and the other 30% of data were for testing in the experiments. Ten the experiments were conducted in 10 folds and all weights were initialized in to the same values in the experiments. The learning process was terminated if the number of epochs was more than 100 epochs or Δw was not changed. The initial values of α and β were set to 0.01 and 0.1,

Table 4.2: The original data sets are set into the imbalanced data sets in order to use in the experiments. The same as the data in RAMOBoost, 2010.

Dataset	n	# Data	Minority class (#)	Majority class (#)
Sonar	60	208	Class 'R' (97)	Class 'M' (111)
Spambase	57	4601	Spam email (1813)	Non-spam email (2788)
Ionosphere	34	351	'Bad radar' (126)	'Good radar' (225)
Wine	13	178	Class '1' (59)	Classes '2' and '3' (119)
German	24	1000	Bad credit (300)	Good credit (700)
Vehicle	18	846	'VAN' (199)	the others (647)
Segment	18	2310	'brickface' (330)	the others (1980)
PageBlocks	10	5473	the others (560)	'text' (4913)
Satimage	36	6435	Class '4' (626)	the others (5809)
Vowel	10	990	Class '0' (90)	the others (900)
Abalone	7	731	Class '18' (42)	Class '9' (689)
Glass	9	214	Class '6' (9)	the others (205)
Yeast	8	483	'POX' (20)	'CYT' (463)
Letter	16	20000	Letter 'Z' (789)	Letters 'A' to 'Y' (19211)
Shuttle_ trn	9	43500	'Fpv Close' (37)	the others (43463)

respectively. The number of hidden neurons was set to four in the preliminary experiments. However, the other numbers of hidden neurons were also experimented and discussed later.

For each data set, the cost function was re-established during the training session based on the different number of data in each class. Fig 4.2 - Fig 4.6 summarize the amount of data in each data set removed or cut from the original data set in both the minority and the majority classes. In each graph, the vertical axis denotes the amount of removed data and the horizontal axis denotes the order of training epoch. The digits in the upper box of each graph is the maximum number of removed data in the majority class but that of the lower box is the maximum number of removed data in minority class during the training session. Note that the number of removed training data is increased after a number of

epochs increases because the learning accuracy increases. In Table 4.3, the fourth column and the fifth column show the number of removed and retained data in each data set between the training minority and majority data at a final epoch, respectively.

Fig 4.7(a) and (b) show graphs that are the percentage of the number of removed data in one class and both classes, respectively. Obviously, the data in the majority class should be removed more than the data in the minority class. Notice that at the final epoch, ReCoFT algorithm reduced the number of used training data more than 50 percent as shown in Fig 4.7(b).

Table 4.3: The number of removed and retained data at the final epoch.

Data set	m	M	# of removed		# of retained		Final epoch
			m	M	m	M	
Sonar	68	78	0	0	68	78	100
Spambase	1269	1952	479	1719	790	233	100
Ionosphere	88	158	0	0	88	158	100
Wine	41	83	0	15	41	68	100
German	210	490	38	298	172	192	100
Vehicle	139	453	0	333	139	120	100
Segment	231	1386	149	1197	82	189	70
Page	392	3439	0	3059	392	380	83
Satimage	438	4066	0	3180	438	886	100
Vowel	63	630	14	577	49	53	90
Abalone	29	482	0	342	29	140	100
Glass	6	144	0	108	6	36	100
Yeast	14	324	0	288	14	36	26
Letter	552	13448	439	13408	113	40	100
Shuttle	26	30424	0	30387	26	37	37

Fig 4.8(a), (b) and (c) show the performance of each method based on the testing data in the minority, the majority, and both classes. For each data set, the first vertical bar denotes the accuracy of the algorithm. The accuracy of the LM and RAMOBoost are shown by the next two vertical stripes, respectively. Notice that the accuracy of the minority class obtained from the algorithm is lower than the accuracy of the LM and RAMOBoost in these eight data sets, i.e. Sonar, Ionosphere, Page, Satimage, Vowel, Abalone, Letter, and Shuttle. This is because accuracy was based on only four hidden neurons. However, the experiments are conducted to find the best number of hidden neurons to increase the accuracy.

Fig 4.9 shows the different numbers of hidden neurons for each data set to make the accuracy from the algorithm is higher than RAMOBoost, i.e. Sonar, Ionosphere, Page, Satimage, Vowel, Abalone, Letter, and Shuttle. The number of hidden neurons were varied in between 1 to 20. The separated results of the minority class, the majority class, and all classes are shown in Fig 4.9(a), (b) and (c). The number in each box means a number of hidden neurons used in each data set. However, in case of Page and Satimage data sets, the accuracy of the results is still lower than that of RAMOBoost.

4.5 Comparison of Training Time

All algorithms were run under the environment of Intel Core 2 Duo E6750@ with 2.66GHz and 2GB RAM. The training time was measured in second unit. Table 4.4 summarized the average training time of each algorithm, i.e. the algorithm (ReCoFT), LM, and RAMOBoost 20 (RAMO*). For RAMOBoost 20, the number of boosting iterations was set to 20 and its training time was the average training time of 20 iterations. Notice the training time of RAMOBoost was more than those of the other methods. If data sets are not very complex, the training time of ReCoFT and LM are quite close. But if data sets have many features or dimensions, obviously, ReCoFT spent less training time than those of the other methods in the following data sets : Spambase, Satimage, Letter, and Shuttle.

Table 4.4: The comparison of the average training time (seconds) of each algorithm.

Data set	Ours (ReCoFT)	LM	RAMOBoost 20 (RAMO*)
Sonar	3.2192	3.2890	95.8513 (4.7926)
Spambase	33.6292	76.2112	3130.9028 (156.5451)
Ionosphere	1.6825	1.4461	47.5490 (2.3774)
Wine	1.3176	1.4842	20.6254 (1.0313)
German	1.6588	2.3377	119.2663 (5.9633)
Vehicle	2.1309	2.0399	30.9601 (1.5480)
Segment	1.1302	7.0015	168.5494 (8.4275)
Page	1.2160	5.0667	159.9820 (7.9991)
Satimage	36.1011	57.5002	1155.1529 (57.7576)
Vowel	0.3625	0.4031	13.4512 (0.6726)
Abalone	0.2199	0.2670	6.3113 (0.3156)
Glass	1.2657	1.6796	20.7212 (1.0361)
Yeast	0.1740	0.3537	12.7315 (0.6366)
Letter	6.5254	31.7967	946.7490 (47.3374)
Shuttle	15.0442	30.5300	586.5496 (29.3275)

4.6 Removed Pattern Description

The interval of hidden neurons in Chapter III, section 3.2, is shown in Table 4.5. Since the number of hidden neurons was set to four in the preliminary experiments ($H = 4$), so each data set was concluded as follows. Spambase, German, Vehicle, Segment, Page, Satimage, Vowel, Abalone and Letter are in Case 1. Sonar is in Case 2. Ionosphere, Wine, Glass, Yeast and Shuttle are in Case 3. The relation between H and removed pattern of data sets in the experiments is summarized in Table 4.3.

Table 4.5: The interval of the number of hidden neurons that help considering.

Data set	n	m	M	m/n	M/n
Sonar	60	68	78	1.1	1.3
Spambase	57	1269	1952	22.3	34.2
Ionosphere	34	88	158	2.6	4.6
Wine	13	41	83	3.2	6.4
German	24	210	490	8.8	20.4
Vehicle	18	139	453	7.7	25.2
Segment	18	231	1386	12.8	77
PageBlocks	10	392	3439	39.2	343.9
Satimage	36	438	4066	12.2	112.9
Vowel	10	63	630	6.3	63
Abalone	7	29	482	4.1	68.8
Glass	9	6	144	0.7	16
Yeast	8	14	324	1.8	40.5
Letter	16	552	13448	34.5	840.5
Shuttle_ trn	9	26	30424	2.9	3380.4

4.7 The Number of Removed Data

In Fig 4.2 - Fig 4.6, the digits in the lower box and the upper box in each graph are the maximum number of removed data in the minority and the majority classes, respectively. Suppose the number of data in training minority class is equal to m and that of training majority class is equal to M at the beginning. Let $min^{(minor)}$ and $min^{(major)}$ be the minimum number of retained data in the minority and the majority classes, respectively. And let $max^{(minor)}$ and $max^{(major)}$ be the maximum number of removed data in the minority and the majority classes, respectively. Therefore, the digits in the lower box and the upper box are defined as follows.

$$max^{(minor)} = m - min^{(minor)} \quad (4.1)$$

$$max^{(major)} = M - min^{(major)} \quad (4.2)$$

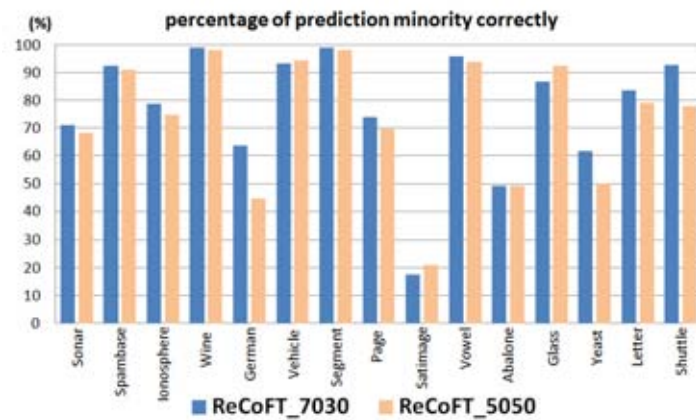
From equation (3.4) and equation (3.5) in Chapter III, section 3.2, equation (4.1) and equation (4.2) can be rewritten as given in equation (4.3) and equation (4.4).

$$max^{(minor)} = m - min\{m, n \cdot H\} \quad (4.3)$$

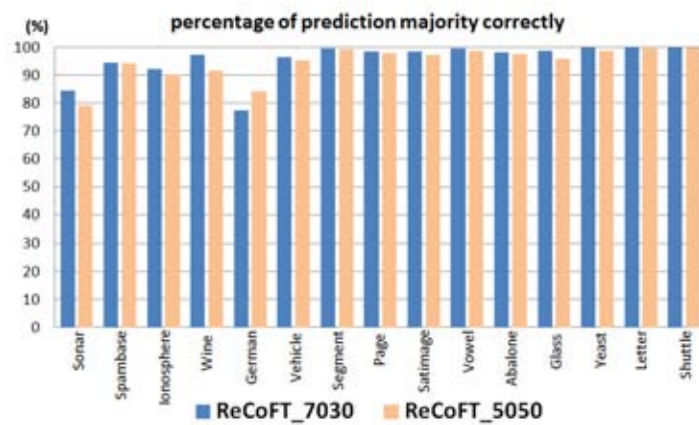
$$max^{(major)} = M - min\{M, n \cdot H\} \quad (4.4)$$

Table 4.6: The digits in the lower ($max^{(minor)}$) and the upper boxes ($max^{(major)}$).

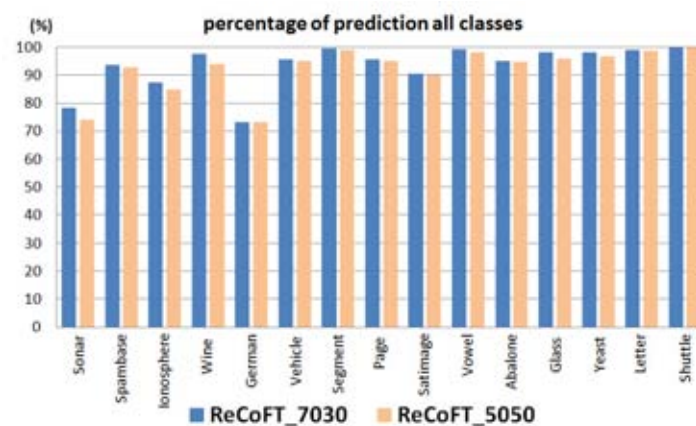
Data set	n	m	M	$n \cdot H$ ($H=4$)	$max^{(minor)}$	$max^{(major)}$
Sonar	60	68	78	240	0	0
Spambase	57	1269	1952	228	1041	1724
Ionosphere	34	88	158	136	0	22
Wine	13	41	83	52	0	31
German	24	210	490	96	114	394
Vehicle	18	139	453	72	67	381
Segment	18	231	1386	72	155	1310
PageBlocks	10	392	3439	40	352	3399
Satimage	36	438	4066	144	294	3922
Vowel	10	63	630	40	23	590
Abalone	7	29	482	28	1	454
Glass	9	6	144	36	0	108
Yeast	8	14	324	32	0	292
Letter	16	552	13448	64	450	13422
Shuttle_ trn	9	26	30424	36	0	30388



(a)

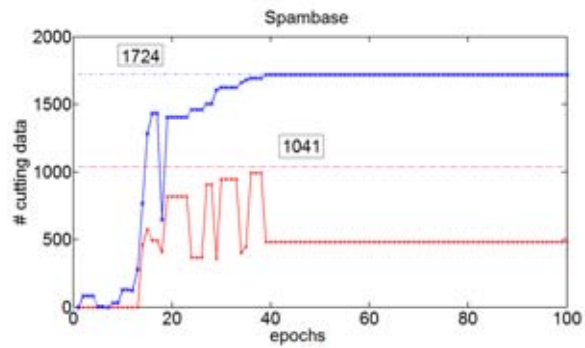


(b)

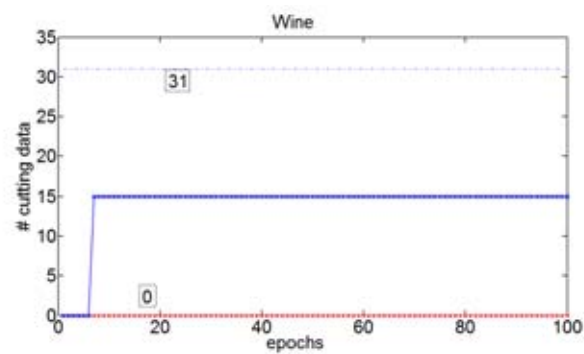


(c)

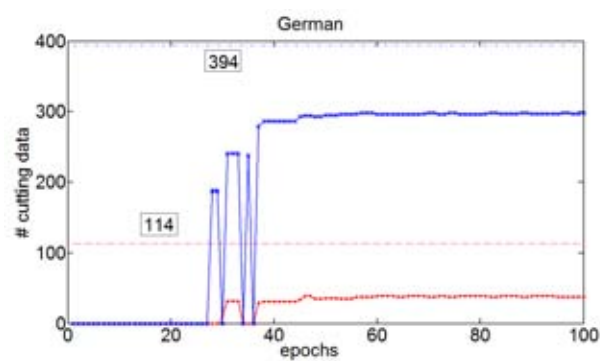
Figure 4.1: The comparison of class accuracy between 70:30 and 50:50 from the algorithm (ReCoFT) for each data set. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes.



(a)

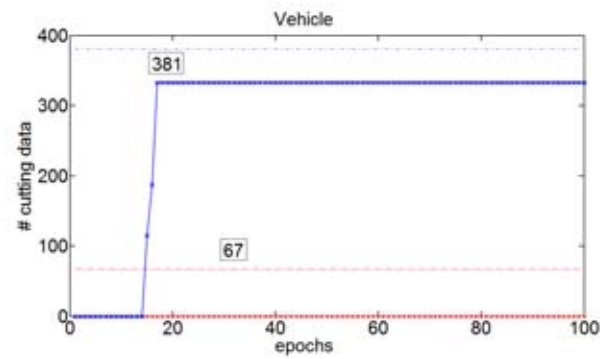


(b)

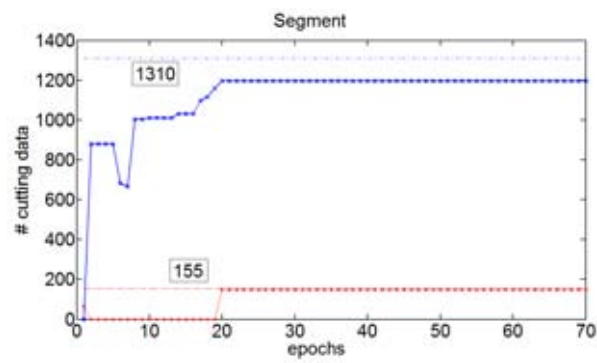


(c)

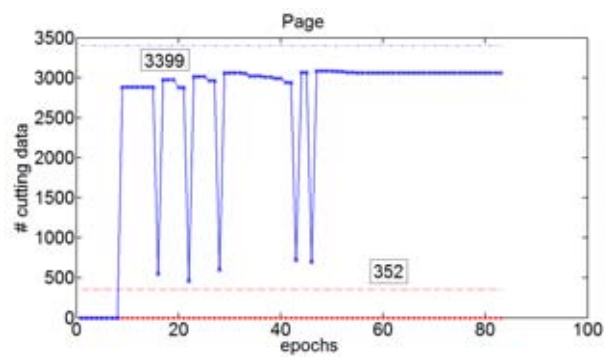
Figure 4.2: The number of removed data at different epochs during the training session of each data set. (a) Spambase. (b) Wine. (c) German.



(a)

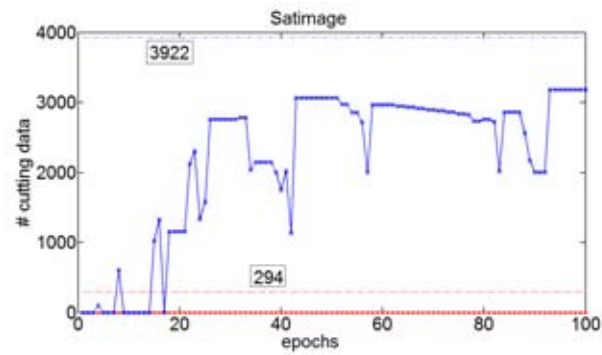


(b)

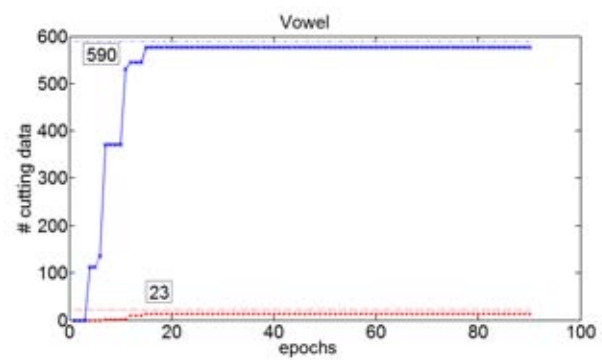


(c)

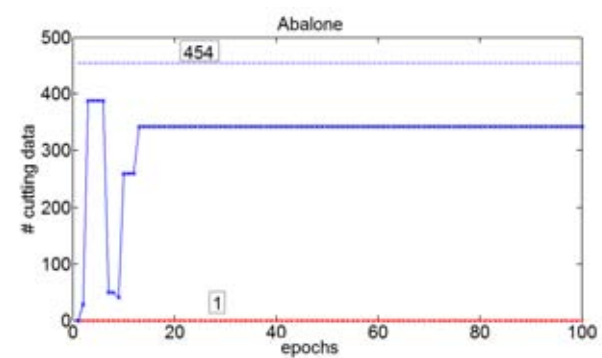
Figure 4.3: The number of removed data at different epochs during the training session of each data set. (a) Vehicle. (b) Segment. (c) Page.



(a)

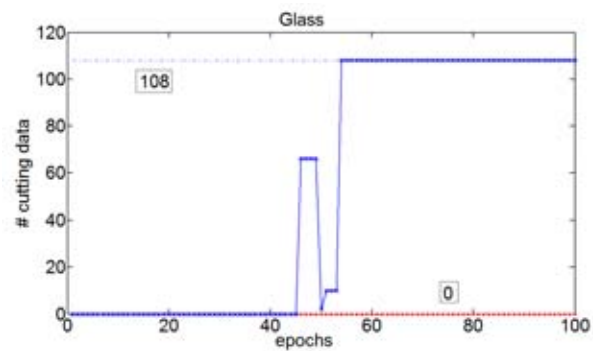


(b)

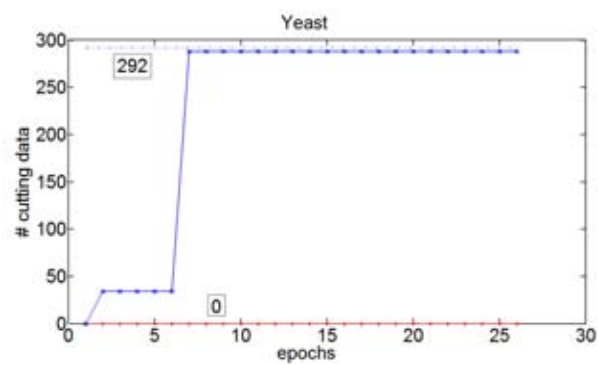


(c)

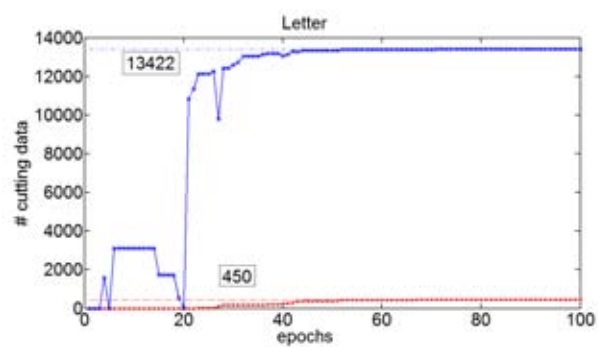
Figure 4.4: The number of removed data at different epochs during the training session of each data set. (a) Satimage. (b) Vowel. (c) Abalone.



(a)



(b)



(c)

Figure 4.5: The number of removed data at different epochs during the training session of each data set. (a) Glass. (b) Yeast. (c) Letter.

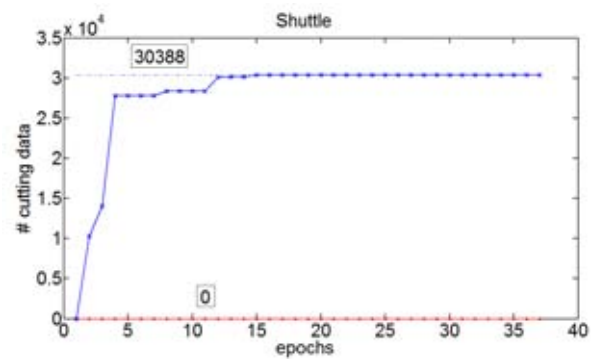
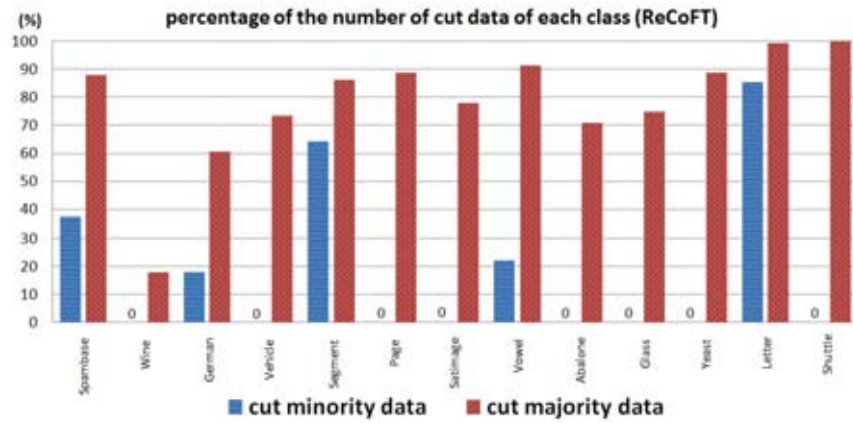
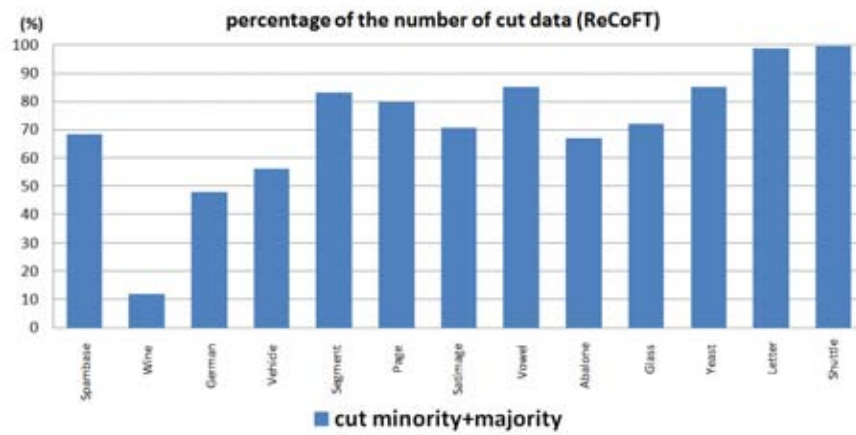


Figure 4.6: The number of removed data at different epochs during the training session of Shuttle data set.

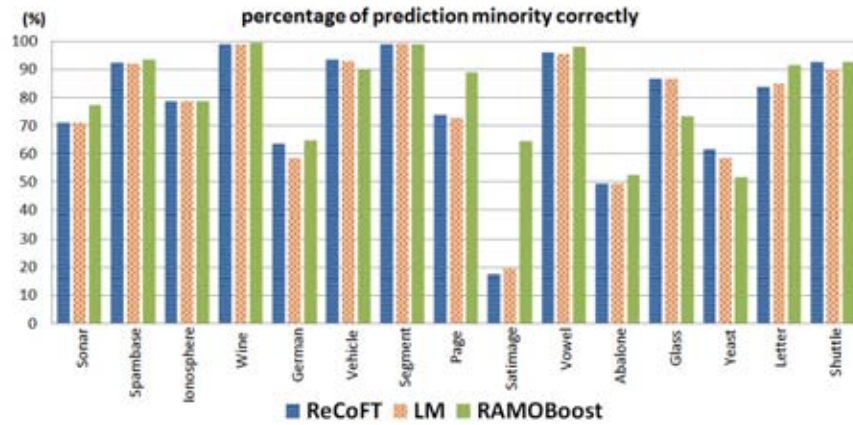


(a)

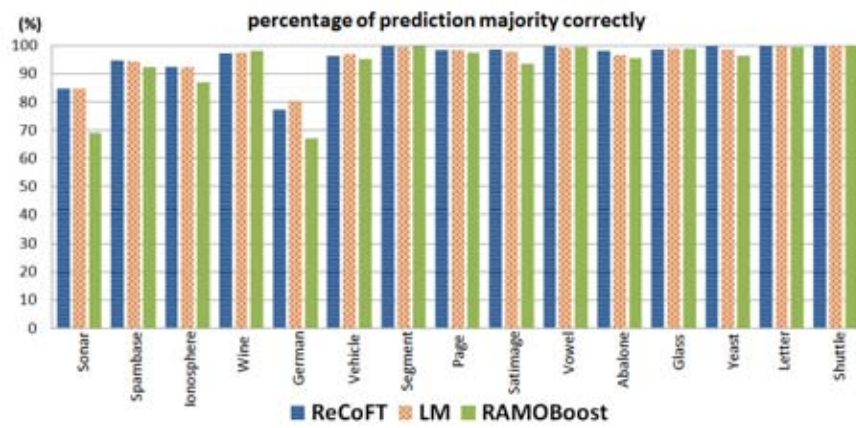


(b)

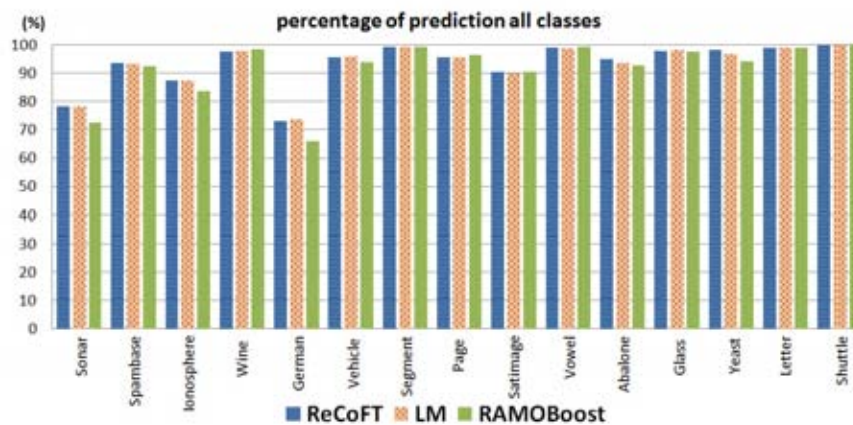
Figure 4.7: The percentage of the number of removed data from the algorithm (ReCoFT) for each data set at a final epoch. (a) Each class. (b) Number of cut data in every class.



(a)

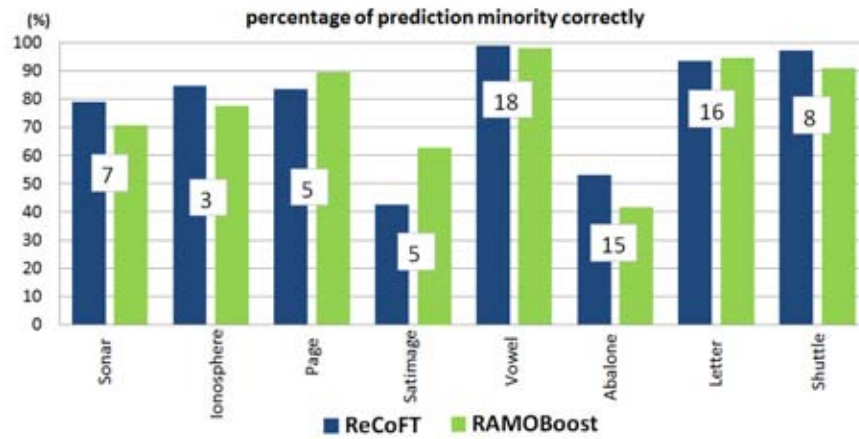


(b)

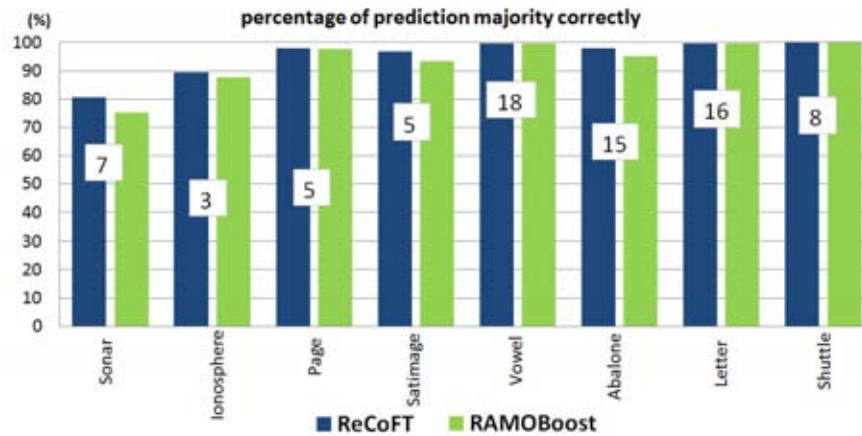


(c)

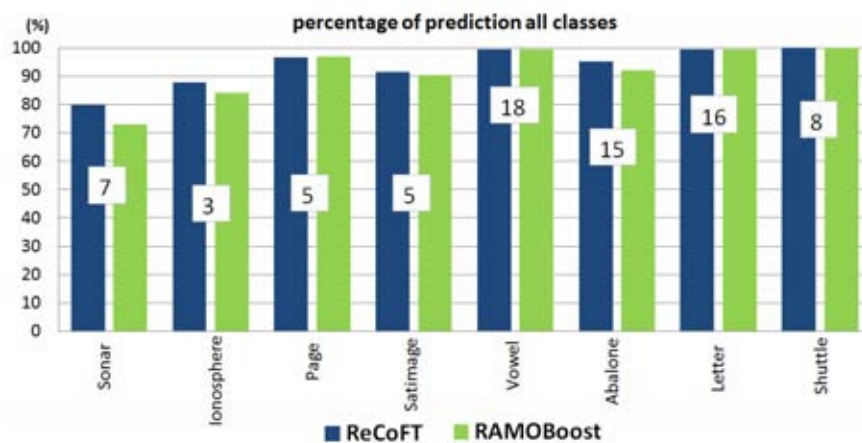
Figure 4.8: The comparison of class accuracy from the algorithm (ReCoFT), LM, and RAMOBoost for each data set. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes.



(a)



(b)



(c)

Figure 4.9: The comparison of class accuracy from the algorithm (ReCoFT) and RAMOBoost for eight data sets. (a) The accuracy of minority class. (b) The accuracy of majority class. (c) The accuracy of all classes.

CHAPTER V

CONCLUSION

A new training algorithm, named *Re-established Cost Function Training*, to enhance the accuracy of the minority class in imbalanced problem was introduced. The cost function was dynamically re-established by removing or putting the training data back into the function based on their closeness to their corresponding targets. The proposed algorithm was compared with the classical learning LM algorithm and the recent RAMOBoost algorithm which also concerned the class imbalanced problem. The algorithm achieved the higher accuracy in most cases within shorter training times.

The results confirmed that if the number of hidden neurons is unsuitably selected from each data set, the performance can be degraded. Unfortunately, the suitable number of hidden neurons cannot be explicitly determined on each data set. So the number of hidden neurons were varied within a defined interval until they get the best performance. Besides, there are several factors that affect to the performance of the algorithm. Those training set should not be used for the learning algorithm in the algorithm. Several learning times on the several training sets is one of the solution for those problem.

The algorithm (ReCoFT) can be used to preliminarily select the interval of the hidden neurons under the conditions of the algorithm in order to determine the behavior of removing data in the training session and also to limit the number of hidden neurons into narrow ranges.

For future work, Finding a new condition for select the data from both classes to increase efficiency in pattern removing should be emphasized.

REFERENCES

- [1] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. *Data Mining and Knowledge Discovery Handbook*, pages 875–886, 2010.
- [2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [3] Sheng Chen, Haibo He, and E.A. Garcia. Ramoboost: Ranked minority over-sampling in boosting. *Neural Networks, IEEE Transactions on*, 21(10):1624–1642, oct. 2010.
- [4] Haibo He, Yang Bai, E.A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328, june 2008.
- [5] Haibo He and E.A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, sept. 2009.
- [6] A. Frank and A. Asuncion. UCI machine learning repository : <http://archive.ics.uci.edu/ml>, 2010.
- [7] B.M. Wilamowski, S. Iplikci, O. Kaynak, and M.O. Efe. An algorithm for fast convergence in training neural networks. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 3, pages 1778–1782 vol.3, 2001.
- [8] Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi. Modified levenberg-marquardt method for neural networks training. *World Academy of Science, Engineering and Technology 6*, pages 40-48, 2005.

VITA

Name Perasut Rungcharassang
Date of Birth 30 April 1987
Place of Birth Bangkok, Thailand
Education B.Sc. (Mathematics),
Kasetsart University, 2008

Publication

Rungcharassang, P., and Lursinsap, C. Re-established Cost Function Training Algorithm to Enhance Accuracy of Minority Class in Imbalanced Data Learning, *Ninth International Joint Conference on Computer Science and Software Engineering (JCSSE) 2012*, pages 39 - 44, 2012.