LOGISTIC REGRESSION METHODS FOR CLASSIFICATION OF IMBALANCED DATA SETS

SANTI PUTERI RAHAYU

DOCTOR OF PHILOSOPHY

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's full name : SANTI PUTERI RAHAYU

Date of birth : 15 JANUARY 1975

Title : LOGISTIC REGRESSION METHODS FOR CLASSIFICATION OF IMBALANCED DATA SETS

Academic Session : 2011/2012

I declare that this thesis is clarified as :

☐ **CONFIDENTIAL** (Contain confidential information under the Official Secret Act 1972)*

☐ **RESTRICTED** (Contain restricted information as specified by the organization where research was done)*

☑ **OPEN ACCESS** I agree that my thesis to be published as online open access (Full text)

I acknowledge that Universiti Malaysia Pahang reserve the rights as follows:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies for the purpose of research only
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's signature) (Signature of Supervisor)

A0080024 PROF. DR. JASNI MOHAMAD ZAIN

New IC / Passport Number Name of Supervisor

Date : 25/09/2012 Date : 27/9/2012

NOTES: If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentially and restriction

LOGISTIC REGRESSION METHODS FOR CLASSIFICATION OF IMBALANCED DATA SETS

SANTI PUTERI RAHAYU

Thesis submitted in fulfilment of the requirements
for the award of the degree of
Doctor of Philosophy in Computer Science

Faculty of Computer System and Software Engineering
UNIVERSITI MALAYSIA PAHANG

SEPTEMBER 2012

STATEMENT OF AWARD FOR DEGREE

Thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy (Computer Science)

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy in Computer Science.

Signature :

Name of Supervisor: Prof. Dr. Jasni Mohamad Zain

Position :

Date : 27/9/2012

STUDENT'S DECLARATION

I hereby declared that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature : 

Name : Santi Puteri Rahayu

ID Number : PCC07002

Date : 25/09/2012

In the name of Allah,
The most Gracious, the Most Merciful.

And I (Allah) created not the jinn and mankind
except that they should worship Me (Alone)
(QS. 51: 56)

Dedicated specially to Da'wah ila Allah SWT,
my beloved husband (Juwari) and my children (Joesan's juniors: Izdihar, Taqy, Faiza, …)
my beloved parents (Ibunda Widiarti and alm. Ayahanda Marsudi Djojosoemarto),
my beloved grand parents (alm. Eyang kakung Khoiron and Eyang putri Hafidah),
my beloved brothers and sister (Bagus N. Wibowo, Dian Saraswati and W. Candra N.),
my beloved nephew and niece (N. M. Darmawan, S. Widya P, …)
my beloved grand children (Lathifah, ...)
and my beloved big family.

ACKNOWLEDGEMENTS

All praise and thanks are Allah's, the Lord of the 'Alamin (mankind, jinn and all that exist). Alhamdulillah, thanks God, to make this research possible.

I would to express my deep gratitude to Prof. Dr. Jasni Mohamad Zain for the freedom to determine the path of my PhD Research and for having served as my advisor during my study in University Malaysia Pahang (UMP), Malaysia. I would also like to thank Prof. Dr. Abdullah Embong for his advice and wisdom.

I would like to thank Prof. Sabira K., Prof. Dr. Siti Mariyam binti Shamsuddin and Dr. Tutut Herawan for their valuable input, comments and suggestions. My appreciation also goes to all my 'teachers', either formal or informal.

I would like to acknowledge UMP for giving me a financial support during my study. I would also like to acknowledge Department of Statistics, Institut Teknologi Sepuluh Nopember (ITS) Surabaya (Indonesia) for giving me a chance to further study in UMP.

I would like to thank Dr. Juwari Purwo Sutikno for his patience and invaluable help specially in providing basic of Matlab programming in this research and in editing layout of this thesis. I would like also to thank Dr. Anwaruddin Hisyam for taking time to edit English in this thesis. My appreciation also goes to all researchers for their useful researches which are the references of this research.

I would like to thank my colleagues (specially for Santi Wulan Purnami and Wibawati) and my friends (specially for Sita Fitriana, Dewi Anggoro, Amik Purbowati, Rini Susanah, Mulyati Ajisman, Wanti Utami, Ambikka and Mamiek Setyaningsih) for their attention, invaluable support and help during my study.

Finally, I would like to give my warm thanks to my husband, my children, my parents, my parents in law, my grandparents, my brothers-sisters and my big family for their love, wisdom, understanding, patience and strong support during my study.

ABSTRACT

Classification of imbalanced data sets is one of the important researches in Data Mining community, since the data sets in many real-world problems mostly are imbalanced class distribution. This thesis aims to develop the simple and effective imbalanced classification algorithms by previously improving the algorithms performance of general classifiers i.e. Kernel Logistic Regression Newton-Raphson (KLR-NR) and Regularized Logistic Regression NR (RLR-NR) which are Logistic Regression (LR)-based methods. Both LR-based methods have strong statistical foundation and well known classifiers which have simple solution of unconstrained optimization problem in performing the good performance as well as Support Vector Machine (SVM) which is determined as state-of-the art classifier in Kernel methodology and Data Mining community. However, the imbalanced LR-based methods are not extensively developed such as imbalanced SVM-based methods. Hence, it is required to develop effective imbalanced LR-based methods to be widely used in data mining applications.

Numerical results have showed that the use of Truncated Newton method for KLR-NR and RLR-NR which respectively resulted in Newton Truncated Regularized KLR (NTR-KLR) and NTR RLR (NTR-LR), is effective in handling the numerical problems on the huge matrix of *linear system of Newton-Raphson update rule* i.e. the training time and the singularity problem. These results can be seen as further explanation on the success of Truncated Newton method in TR-KLR and TR Iteratively Re-weighted Least Square (TR-IRLS) algorithm respectively, because of the equivalence of iterative method used by these algorithms. Moreover, only with the use of simple solution of unconstrained optimization problem, numerical results have demonstrated that proposed NTR-KLR and proposed NTR-LR respectively have comparable classification performance with RBFSVM (SVM with Radial Basis Function Kernel).

The imbalanced problem of both proposed general classification algorithms which is the limitation of accuracy performance specifically in classifying on the minority class has motivated this research to improve their classification performance on imbalanced data sets. In general, numerical results have showed that the use of adapted Modified AdaBoost methods for NTR-KLR and NTR-LR which respectively resulted in AdaBoost NTR Weighted KLR (AB-WKLR) and AB NTR Weighted RLR (AB-WLR) is significantly successful in improving the accuracy and stability performance of general classifiers i.e. NTR-KLR and NTR-LR respectively. The improvements on both error by *g-means* and *standard deviation* of *g-means* with *5*-Fold SCV could be achieved as high as more than 60. Furthermore, numerical results have demonstrated that proposed AB-WKLR and proposed AB-WLR respectively have comparable performances with AdaBoostSVM in classifying imbalanced data sets, only with the use of simple solution of unconstrained weighted optimization problem. Thus, both proposed imbalanced LR-based methods is simple and effective for classification of imbalanced data sets and have promising results.

ABSTRAK

Pengelasan set data yang tidak seimbang adalah salah satu kajian yang penting dalam masyarakat perlombongan data, kerana set data yang digunakan dalam dunia sebenar kebanyakannya adalah pengagihan kelas tidak seimbang. Tesis ini bertujuan untuk membangunkan algoritma pengelasan tidak seimbang yang mudah dan berkesan dengan meningkatkan prestasi algoritma pengelas umum iaitu *Kernel Logistic Regression Newton-Raphson* (*KLR-NR*) dan *Regularized Logistic Regression NR* (*RLR-NR*) yang merupakan kaedah berasaskan *Logistic Regression* (*LR*). Kedua-dua *LR-based methods* mempunyai asas statistik yang kukuh dan terkenal sebagai pengelas yang mempunyai penyelesaian yang mudah dari *unconstrained optimization problem* dalam melaksanakan prestasi yang sama baik dengan *Support Vector Machine* (*SVM*) yang ditentukan sebagai *state-of-the-art* pengelas dalam metodologi Kernel dan masyarakat Perlombongan Data. Walau bagaimanapun, *imbalanced LR-based methods* tidak dibangunkan secara meluas seperti *imbalanced SVM-based methods*. Oleh itu, ia diperlukan untuk membangunkan *imbalanced LR-based methods* yang berkesan yang digunakan secara meluas dalam banyak aplikasi perlombongan data.

Keputusan berangka telah menunjukkan bahawa penggunaan kaedah *Truncated Newton* untuk *KLR-NR* dan *RLR-NR* yang masing-masing mengakibatkan *Newton Truncated Regularized KLR* (*NTR-KLR*) dan *NTR RLR* (*NTR-LR*), adalah berkesan dalam menangani masalah berangka pada matriks besar dari sistem linear *Newton-Raphson update rule* iaitu masalah masa latihan dan ketunggalan. Keputusan ini boleh dilihat sebagai penjelasan lanjut mengenai kejayaan kaedah *Truncated Newton* di *TR-KLR* dan *TR Iterative Re-weighted Least Square* (*TR-IRLS*) algoritma, kerana kesetaraan kaedah lelaran yang digunakan oleh algoritma-algoritma ini. Selain itu, dengan hanya menggunakan penyelesaian yang mudah dari *unconstrained optimization problem*, keputusan berangka telah menunjukkan bahawa cadangan *NTR-KLR* dan cadangan *NTR-LR* masing-masing mempunyai prestasi klasifikasi setanding dengan *RBFSVM* (*SVM* dengan *Radial Basis Function*).

Masalah tidak seimbang kedua-dua algoritma klasifikasi umum yang dicadangkan yang merupakan had prestasi ketepatan khususnya dalam mengklasifikasikan kelas minoriti telah mendorong kajian ini untuk meningkatkan prestasi klasifikasi mereka pada set data yang tidak seimbang. Secara umum, keputusan berangka telah menunjukkan bahawa penggunaan kaedah *adapted Modified AdaBoost* untuk *NTR-KLR* dan *NTR-LR* yang masing-masing mengakibatkan *AdaBoost NTR Weighted KLR* (*AB-WKLR*) dan *AB NTR Weighted RLR* (*AB-WLR*) adalah lebih berjaya dalam meningkatkan prestasi ketepatan dan kestabilan pengelas umum iaitu *NTR-KLR* dan *NTR-LR*. Peningkatan bermakna oleh kedua-duanya atas kesilapan *g-means* dan sisihan piawai *g-means* dengan 5-Lipat SCV boleh dicapai setinggi lebih daripada 60. Tambahan pula, keputusan berangka telah menunjukkan bahawa cadangan *AB-WKLR* dan cadangan *AB-WLR* masing-masing mempunyai persembahan yang setanding dengan *AdaBoostSVM* dalam mengklasifikasikan set data tidak seimbang, hanya dengan menggunakan penyelesaian yang mudah dari *unconstrained weighted optimization problem*. Oleh itu, kedua-dua cadangan imbalanced *LR-based methods* merupakan kaedah yang mudah dan berkesan untuk pengkelasan set data yang tidak seimbang dan mendapat keputusan yang menjanjikan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a	The optimal step length
α	Coefficient vector of Kernel Logistic Regression
β	Coefficient vector of Regularized Logistic Regression
c	Conjugacy enforcer
d	The search direction
dis	The distance of a sample from the separating hyperplane
dim	Number of attributes
D	Diagonal matrix of variance and weight vector
ε_1	Threshold of the difference of optimization function values
ε_2	The convergence threshold for Linear CG
ε_t	The weighted error of component classifier on *t*-th round
f	Linear function
F	Ensemble function
h_t	The weighted prediction of component classifier on *t*-th round
g	Gradient vector
H	Hessian matrix
K	Kernel matrix
k	Number of fold
k_{ij}	Cell of kernel matrix
$\mathbf{K}_1$	Kernel matrix with the bias term
$\mathbf{K}_2$	Matrix that consist of diagonal element: **K** and the bias term is not regularized
l	Likelihood function

L	Log-likelihood function
λ	Regularization parameter
n	Number of samples
n_σ	Number of σ during AB-WKLR iterations
n_λ	Number of λ during AB-WLR iterations
$\mathbf{p}$	Probability of given input
φ	Function to map the original data **x** in input space into feature space
q	Quadratic form
$\mathbf{r}$	Residual
$\mathbf{s}$	Vector of Newton direction
S_{gmeans}	Standard deviation of g-means values
σ	RBF Kernel parameter
T	the number of AdaBoost iterations,
$\boldsymbol{\theta}$	Vector of general parameter
$\mathbf{v}$	Variance vector
$\mathbf{V}$	Diagonal matrix of **v**
$\mathbf{w}$	Weight vector of training samples
$\mathbf{W}$	Diagonal matrix of **w**
ω	The importance factor of corresponding component classifier to an ensemble
$\mathbf{x}$	Input vector without bias term
$\mathbf{y}$	Vector of input label
y_{pred}	Predictions of AdaBoost classifier
Z_t	The normalization factor on *t*-th round
$\mathbf{Z}$	Vector of adjusted response

ζ Function of Bernoulli distribution

LIST OF ABBREVIATIONS

AB-WKLR	Adaptive Boosting Weighted Kernel Logistic Regression
AB-WLR	Adaptive Boosting Weighted Regularized Logistic Regression
AdaBoost	Adaptive Boosting
AdaBoostSVM	Adaptive Boosting Support Vector Machine
AUC	Area Under Receiver Operating Curve
CG	Conjugate Gradient
CM	Confusion Matrix
CV	Cross Validation
DEV	Deviance
IVM	Import Vector Machine
NR	Newton-Raphson
NRUR	Newton-Raphson update rule
NTR-LR	Newton Truncated Regularized Logistic Regression
NTR-KLR	Newton Truncated Regularized Kernel Logistic Regression
NTR-WKLR	Newton Truncated Regularized Weighted Kernel Logistic Regression
NTR-WLR	Newton Truncated Regularized Weighted Regularized Logistic Regression
GS	Grid Search
GSVM-RU	Granular Support Vector Machine-Repetitive Under-sampling
IRLS	Iteratively Re-Weighted Least Square
KLR	Kernel Logistic Regression
KLR-IRLS	Kernel Logistic Regression Iteratively Re-Weighted Least Square

KLR-NR	Kernel Logistic Regression Newton-Raphson
LCG	Linear Conjugate Gradient
MLE	Maximum Likelihood Estimation
NLL	Negative Log-Likelihood
NR	Newton-Raphson
RBF	Radial Basis Function
RE-WKLR	Rare Event Weighted Kernel Logistic Regression
RLR	Regularized Logistic Regression
RLR-IRLS	Regularized Logistic Regression Iteratively Re-Weighted Least Square
RLR-NR	Regularized Logistic Regression Newton-Raphson
SCV	Stratified Cross Validation
SDC	Smote with Different Cost
SMO	Sequential Minimization Organization
SMOTE	Synthetically Minority Over-sampling Technique
SVM	Support Vector Machine
TR-IRLS	Truncated Regularized Iteratively Re-Weighted Least Square
TR-KLR	Truncated Regularized Kernel Logistic Regression
WKLR	Weighted Kernel Logistic Regression
WLR	Weighted Regularized Logistic Regression
WLS	Weighted Least Square
WWBOOST-SVM	Weighting rule and Weakened Support Vector Machine based Boosting

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The interface of statistics, database technology, pattern recognition, machine learning, and other areas are termed as Data Mining. It is concerned with the analysis of large databases by using machine learning methods, in identifying previously unsuspected pattern which are of interest or value to the data. (Hand, 1998; Tan et al., 2005).

Classification is a supervised data mining task, which is a predictive task with qualitative outcome. In the last decade, it is found that, beside the evaluation of data in manual, the use of classifier system is also very important factor in helping expert to make decision, i.e. to identify pattern and make prediction. Classifier system can achieve a fast, objective, more detailed and accurate classification by minimizing possible errors due to fatigued or inexperienced expert. (Huang et al., 2007; Polat et al., 2007; West, 2000).

In the last decade, the resulting family of Kernel learning methods (Scholkopf and Smola, 2002; Shawe and Christianini, 2004) have frequently demonstrated state-of-the-art performance on a wide range of benchmark and real-world applications. Most of these kernel-based methods, however, are presented in the literature along with the Support Vector Machine (SVM) method. SVM (Vapnik, 1998; Vapnik, 2000), which was developed based on the theory of Structural Risk Minimization (SRM), is popular with its effectiveness in the Kernel Machine Learning and Data Mining Community,

such that it is considered as state-of-the-art algorithm for classifying non-linear binary data.

Beside SVM, Kernel Logistic Regression (KLR) (Roth, 2001; Zhu and Hastie, 2004; Zhu and Hastie, 2005) is one of the most important recent developments for classification task in Kernel-machine techniques. It is the Kernel version of Regularized Logistic Regression (RLR) (Minka, 2003; Zhang and Oles, 2001) classifier. The use of Kernel in KLR algorithm is to improve the generalization performance of RLR on overcoming the non-linear problem that has low-to-medium-dimensional data (Maalouf, 2009).

Meanwhile, RLR is the regularized version of Logistic Regression (LR) (Hosmer and Lemeshow, 2000; Dreitsel and Machado, 2002; Hastie et al., 2001; McCulagh and Nelder, 1989) which is the fundamental and well known statistical method for classification task. It is a classifier which is well applied to linear problem with high-dimensional data (Komarek and Moore, 2005). Hence, RLR is considered as state-of-the-art algorithm for linear discriminant data.

KLR and RLR have received more extensive research attention, since they have similar loss function with SVM (Patra et. al., 2008; Rahimi, 2006; Rennie, 2005; Zhang and Oles, 2001; Zhang et al., 2003; Zhu and Hastie, 2005). Furthermore, by using *total accuracy* metric, the classification performance of KLR is similar to non-linear SVM (Karsmaker et al., 2007), while the classification performance of RLR is comparably accurate to linear SVM (Zhang et al., 2003; Zhang and Oles, 2001). However, optimization of SVM needs to be solved with quadratic constrained optimization, while KLR and RLR only need to be solved by unconstrained optimization (Maalouf, 2009), although it also can be stated as constrained optimization problem (Karsmaker et al., 2007; Kerthi et al., 2005). In addition, unlike SVM, both classifiers naturally provide probability of classification membership (Zhu, 2003; Zhang et al., 2003).

Many problem domains require transparent reasoning as well as accurate classifier (Ridgeway et.al, 1998). Trust in a system is developed by the quality of the results (accuracy) and also by clear description of how they were derived (transparent

reasoning) (Swartout, 1983). Good accuracy enables correct assessments / diagnosis / treatment and thus avoiding any heavy losses associated with wrong prediction (Lahsasna et al, 2008; West, 2000). Transparency enables expert to understand the classification/decision process. The capability of classifier to describe its analysis often affects the end-user acceptance. In types of situation like these, LR-based methods, i.e. KLR and RLR, are appropriate methods.

In summary, LR-based methods have simple optimization function than SVM-based methods on performing comparable accuracy. Moreover, the transparency of LR-based methods is supported by providing the membership probability naturally. Furthermore, LR-based methods are well known methods and have strong statistical foundation. However, as further as limited knowledge, the LR-based methods have less extensive research than SVM-based methods on imbalanced classification problem. Hence, in order to take the advantages of LR-based methods and to give further contribution on the research of LR-based methods, this thesis aims to further develop the LR-based methods for solving the classification problems, either general or imbalanced problem.

1.2 PROBLEM STATEMENT AND MOTIVATION

This thesis interests to conduct study on two main problems of KLR and RLR. The problems can be stated as follows:

(i) Newton-Raphson (Rennie, 2003) is the most commonly method to solve the non-linear optimization problem of KLR and RLR. Newton-Raphson method iteratively solves the linear system of Newton-Raphson Update Rule (NRUR). As has been reported in literatures, however, the use of Newton-Raphson method for KLR and RLR has numerical problem that the huge Hessian matrix needs to be inverted (Lin et al., 2008; Zhu and Hastie, 2005). Due to the density of its matrices, their computation can be slow (Komarek, 2004; Karsmakers et al. 2007; Maalouf, 2009).

(ii) General classifiers, such as SVM, KLR and RLR, were developed and evaluated on the assumption that the data has balanced class distribution (Japkowicz,

2000; Maalouf, 2009). However, in many real-world problems, it was faced that the data sets have imbalanced class distribution. The class imbalance problem corresponds to domains for which one class is represented by a large number of examples while the other is represented by only a few (Guo and Viktor, 2004; Japkowicz, 2000). In the case of binary classification, data sets are said to be imbalanced, if the number of negative instances are heavily larger than the positive ones (Akbani et al., 2004; Maalouf, 2009). Commonly, for two-class classification of imbalanced data set, the negative class is the notation for the majority class, while the positive class is the notation for the minority class. In imbalanced classification problems, the minority class is the class of primary interest. As has been reported in literatures of Kernel learning, it seems difficult for general classifier algorithms, even though SVM, to detect regularities within the minority class on imbalanced data problems (Akbani et al, 2004; Maalouf, 2009). Therefore, they have good specificity, but poor sensitivity (Akbani et al., 2004; Maloouf, 2003). King and Zeng (2001c) stated similarly that when non-kernel of probabilistic method such as logistic regression, is used, it underestimates the probability of rare events, because it tends to be biased towards the majority class, which is the less important class. Recently, in relation to further development of KLR and RLR respectively, this thesis has confirmed the limitation performance of both general classification algorithms on imbalanced data sets. The report can be found in Chapter 4.

The motivation of this research is described as follows:

(i) Several methods have been proposed for solving the numerical problem of KLR and RLR. Detail analysis of those methods proposed will be reported in Chapter 2. In the last decade, the use of Truncated Newton methods are the most proposed methods on applying KLR and RLR. However, so far, the success of Truncated Newton method in both algorithms has not been totally explored. Therefore, this thesis intends to contribute further explanation on the success of Truncated Newton for KLR and RLR specifically on improving the algorithm performance of these both LR-based methods.

(ii) For solving the imbalanced classification problem, a number of methods have been proposed in literatures of Kernel learning. Discussion on the limitation of those methods will be reported in detail, in Chapter 2. Based on those methods proposed, in general, the research of imbalanced LR-based methods are not as many as the research of imbalanced SVM-based methods which have good accuracy performance. Furthermore, the imbalanced techniques used on LR-based methods have led their accuracy performances for classification of imbalanced data sets that still require an improvement. Hence, it is important to develop the effective imbalanced LR-based methods for solving the imbalanced classification problem of general LR-based methods.

1.3 THE APPROACHES

This research concerns on developing better general and imbalanced classification algorithms for KLR-NR and RLR-NR. Related to this concern, there are two main problems that must be handled in this thesis, as stated in the previous section. The approach for solving those problems can be described as follows:

(i) In order to develop the simple and effective of general classification algorithms for KLR-NR and RLR-NR respectively, this research proposes the implementation of Truncated Newton method. Among other Truncated Newton LR-based method, the simplicity and the effectiveness of Truncated Regularized KLR (TR-KLR) (Maalouf et al., 2010) and TR Iteratively Re-weighted Least Square (TR-IRLS) (Komarek and Moore, 2005) have inspired this research. TR-KLR is as accurate as, and much faster than, non-Linear SVM on small-to-medium size data sets of non-linear classification problem. Meanwhile, TR-IRLS is comparably accurate with, and faster than, Linear SVM on large size data sets of linear classification problem.

In general, the use of Truncated Newton method typically consists of truncated inner algorithm and outer algorithm (Nash, 2000). In TR-KLR and TR-IRLS, the use of Truncated Newton includes Linear Conjugate Gradient (CG) method (Gilbert, 2006; Nash and Sofer, 1996; Shewchuk, 1994) and Iteratively Re-

weighted Least Square (IRLS) procedure (Mc Cullagh and Nelder, 1989; Nabney, 1999; Hastie et al., 2001) for KLR and RLR respectively.

In summary, the approaches for solving the numerical problem of KLR-NR and RLR-NR can be explained as follows:

(a) It is necessary to keep the use of unconstrained optimization problem for KLR-NR and RLR-NR respectively. This optimization problem typically has simpler solution than the constrained ones.

(b) It is also necessary to keep the use of Linear CG method, as the truncated inner algorithm of Truncated Newton method for KLR and RLR respectively. This method has faster computation in approximating the Newton's solution.

(c) Instead of IRLS procedure as used by TR-KLR and TR-IRLS, this approach uses Newton-Raphson method as the outer algorithm of Truncated Newton method. Newton-Raphson and IRLS are equivalent method for KLR and RLR. In addition, Newton-Raphson method is mathematically simple, because IRLS procedure is a representation of Newton-Raphson method.

The use of Truncated Newton method for solving the numerical problem of KLR-NR and RLR-NR algorithm respectively results in proposed Newton TR-KLR (NTR-KLR) and proposed Newton TR RLR (NTR-LR) algorithm. Because of the equivalency between Newton-Raphson method and IRLS procedure, the accuracy performance of both proposed classifier can be expected to have similar performance for TR-KLR and TR-IRLS respectively. In addition, both proposed algorithms can be seen as the Newton version of TR-KLR and the Newton version of TR-IRLS algorithm. Hence, both proposed algorithms can be used to contribute further explanation on the success of Truncated Newton method in TR-KLR and TR-IRLS respectively.

Moreover, the development of both proposed algorithms can be seen as preliminary representation of idea stated by Komarek (2004) that whether the behaviour of Newton-Raphson and Linear CG combination would be identical to IRLS and Linear CG combination. In specific, development of proposed NTR-KLR algorithm can be seen also as preliminary representation of Kernel version to the Trust Region Newton RLR that was proposed by Lin et al. (2008).

(ii) In order to develop the effective imbalanced classification algorithms for NTR-KLR and NTR-LR respectively, this thesis proposes the use of Modified AdaBoost method (with some adaptations). This is motivated by the success of imbalanced SVM-based method i.e. Adaptive Boosting SVM (AdaBoostSVM) (Li et al., 2008) with the use of this imbalanced technique. AdaBoostSVM has much better performance than SVM on solving the imbalanced classification problem. The use of AdaBoost-based method (Freund and Schapire, 1997) typically contains ensemble method and component classifier. In AdaBoostSVM, the ensemble method used is Modified AdaBoost and the component classifier is SVM with Radial Basis Function (RBF) Kernel (RBFSVM).

Detail strategies for solving the imbalanced classification problem of general LR-based methods are described in the following:

a. It is necessary to keep the use of Modified AdaBoost (with some adaptations) as the ensemble method of proposed imbalanced LR-based methods. Boosting mechanism of Modified AdaBoost forces the component classifiers to focus on the misclassified samples from the minority class by increasing the weights of training data. This prevents the minority class from being consider as noise in the majority class and be wrongly classified on imbalanced problem.
b. Instead of SVM, this approach uses NTR-KLR and NTR-LR respectively as the component classifier of proposed imbalanced LR-based methods. As proposed previously, NTR-KLR and NTR-LR are representation of KLR-NR and RLR-NR with Truncated Newton method respectively. The similarity of loss function among NTR-KLR, NTR-LR and SVM, has led these classifiers can be expected to have comparable accuracy. In addition, with the use of unconstrained optimization problem, NTR-KLR and NTR-LR have simpler solution of optimization problem than SVM.

The implementation of adapted Modified AdaBoost ensemble method for solving the imbalanced classification problem of NTR-KLR and NTR-LR component classifier respectively are called as Adaptive Boosting NTR Weighted KLR (AB-WKLR) and AB NTR Weighted RLR (AB-WLR) algorithm. As further as limited

knowledge, Nishida and Kurita (2006) were the first researchers who applied Boosting method, i.e. LogitBoost, on sparse version of KLR, i.e. Import Vector Machine (IVM) (Zhu and Hastie, 2005), While Huang et al. (2005) was the first to employ classic AdaBoost method on Logistic Regression (LR) that used weighted least-squares as the objective function and batch gradient descent algorithm for its optimization.

Since there is similarity loss function between component classifiers used, the accuracy performance of the proposed algorithms can be expected as well as AdaBoostSVM in classifying the imbalanced data sets. Moreover, the comparable accuracy only requires to be obtained by the simple solution of unconstrained optimization problem.

1.4 OBJECTIVES AND SCOPES

The main objective of the research is to develop the simple and effective classification algorithms using LR-based methods.
The research objective can be stated in detail as follows:

(ii) To develop general classification algorithms, i.e. NTR-KLR and NTR-LR
(iii) To develop imbalanced classification algorithms, i.e. AB-WKLR and AB-WLR

The scope of this research covers the following:

(i) This thesis considers 2-class classification and the data sets used mostly are imbalance.
(ii) Proposed general classification algorithms are developed based on KLR-NR and RLR-NR algorithm respectively, while proposed imbalanced classification algorithms were developed based on NTR-KLR and NTR-LR algorithm respectively.
(iii) Proposed NTR-KLR and proposed AB-WKLR are applied on small-to-medium size of data sets, while proposed NTR-LR and proposed AB-WLR are employed on large size data sets.

1.5 CONTRIBUTIONS

The primary contributions of this research are as follows:

(i) NTR-KLR and NTR-LR algorithm were developed. Both proposed algorithms contribute to the study of KLR-NR and RLR-NR respectively, by providing the simple and effective general classification algorithms for KLR-NR and RLR-NR respectively with the use of Truncated Newton method. Both proposed algorithms are also provided specifically to conduct further explanation on the success of Truncated Newton method in TR-KLR and TR-IRLS respectively, since both proposed algorithms are equivalent to TR-KLR and TR-IRLS respectively. In general, both proposed algorithms contribute to the general classification research of LR-based methods.

(ii) AB-WKLR and AB-WLR algorithm were developed. Both proposed algorithms contribute to the research of KLR-NR and RLR-NR with Truncated Newton method respectively, by providing the simple and effective imbalanced classification algorithms for NTR-KLR and NTR-LR respectively with the use of adapted Modified AdaBoost method. In general, both proposed algorithms contribute to the imbalanced classification research of LR-based methods.

1.6 OUTLINE OF THE THESIS

This thesis is organized as follows. Chapter 2 gives extended reviews of TR-IRLS, TR-KLR, AdaBoost algorithms for SVM and some basic theories of numerical experiment. Chapter 3 describes the proposed algorithms and the research methodology. In chapter 4, several numerical results of experiment are reported and discussed. At the end, conclusions for this research and recommendations for the further work are given in chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter presents the reviews of General and Imbalanced Classification Research, including TR-IRLS, TR-KLR, Adaptive Boosting (AdaBoost) algorithms for SVM and some basic theories on conducting numerical experiment. These reviews are required as fundamental theory in order to propose new algorithm of KLR and RLR, on both the algorithmic level and in dealing with the imbalanced problems.

2.2 CLASSIFICATION

Globally, data mining tasks are divided into two categories, namely supervised and unsupervised task. As mentioned in Chapter 1, classification is a supervised data mining task on predicting categorical response.

In the last decade, there are many classification methods that have been proposed on general and imbalanced data assumption. Among other classification methods, the maturity of LR-based methods has motivated this thesis for exploring these methods as the simple and effective classifier to be widely used in data mining application, either on general or imbalanced data sets.

In order to develop better performance of general and imbalanced classification algorithms for LR-based methods i.e. KLR and RLR, it is important to study the limitation of related previous research. In the following, summary of the latest research of LR-based methods in relation with general and imbalanced data are reviewed. In

relation with imbalanced researches of LR-based methods, the imbalanced researches SVM-based methods also are reviewed in summary.

2.2.1 GENERAL CLASSIFICATION

Several methods have been proposed for solving the numerical problem of LR-based method. Keerthi et al. (2005) suggested the popular sequential minimal optimization (SMO) algorithm for KLR by developing dual optimization problem. Zhu and Hastie (2005) offered the Import Vector Machine (IVM) algorithm in taking the advantage of SVM sparsity into Iteratively Re-weighted Least Square (IRLS) procedure (Hastie et al., 2001; Mc Cullagh and Nelder, 1989; Nabney, 1999) for KLR. Karsmakers et al. (2007) incorporated a fixed-size approach based on the number of support vectors with the method of alternating descent for solving multi-class KLR problem by using Least Square SVM (LS-SVM) (Suyken and Vandewalle, 1999; Suyken et al., 2000) frame work and IRLS procedure. For large scale RLR problem, Zhang et al. (2003) used non-linear Conjugate Gradient (non-linear CG) on the targeted text classification tasks. Komarek and Moore (2005) proposed Truncated Regularized IRLS (TR-IRLS) for RLR that modified the IRLS procedure using Linear CG (Nash and Sofer, 1996; Shewchuk, 1994) method. In 2007, Truncated Newton Interior Point was proposed by Koh et al. (2007) for solving the large scale problem on RLR. Lin et al. (2008) proposed Trust Region Newton (TRN) method that employs constrained CG method to approximate the Trust Region solution of RLR. Maalouf et al. (2010) then suggested Truncated Regularized KLR (TR-KLR) algorithm that combine TR-IRLS with Kernel method for solving non-linear classification problem.

The aforementioned methods have a restriction. SMO for KLR (Keerthi et al., 2005), fixed size of KLR with LS-SVM frame work (Karsmaker et al., 2007) and Truncated Newton interior point (Koh et al., 2007) need to be solved by previously reformulating the unconstrained optimization problem as the constrained one. IVM (Zhu and Hastie) has complex solution, specifically when employ to data set which has large number of attributes. The inner algorithm of Truncated Newton method in Trust Region RLR (Lin et al., 2008) is constrained Linear CG. Hence, it has also complex solution. The method which was proposed by Zhang et al. (2003) applied the

combination of Newton-Raphson method with non-linear CG method which is slower than Linear CG method. The outer algorithm of Truncated Newton method in TR-KLR (Maalouf et al., 2010) and TR-IRLS (Komarek and Moore, 2005) is IRLS procedure. It is required previously to restate the Newton-Raphson formula as the Weighted Least Square (WLS) problem. Based on aforementioned methods, in the last decade the use of Truncated Newton methods are the most proposed methods on applying KLR and RLR respectively. However, the effectiveness of Truncated Newton method for these LR-based methods has not been totally explored, specifically on improving their algorithms performances.

2.2.2 IMBALANCED CLASSIFICATION

General classification algorithm, such as SVM, TR-KLR and TR-IRLS were developed under balanced data assumption, such that they have limited performance when applied on imbalanced data sets which is the most type of data in many real-world problems. Examples of imbalanced problem include the oil spills detection in satellite radar images (Kubat et al., 1998), micro array data clustering (Pearson et al., 2004), document categorization (del Castillo and Serrano, 2004), word mispronunciation (Busser and Daelemans, 1999), credit risk assessment (Lai et al., 2006; Yu et al., 2006), credit card fraud detection (Chan and Stolfo, 1998; Fawcett and Provost, 1997), queues series (Tsoucas, 1992), international conflicts (King and Zeng, 2001a), state failure (King and Zeng, 2001b), tornadoes detection (Trafalis et al., 2003), landslides susceptibility (Bai et al., 2008; Eeckhaut et al., 2006), telecommunication equipment failures (Weiss and Hirsh, 2000), train derailments (Quigley et al., 2007), and other imbalanced problems.

The latest research on data mining in relation to imbalanced data can be mapped in general into Algorithm Level Techniques, Data Level Techniques and Kernel-based Techniques (Maalouf, 2009). The extensive research of Kernel-based techniques in the last decade, have inspired this thesis to study the effectiveness and the limitation of imbalanced techniques used.

A number of methods have been proposed in literatures of Kernel learning in dealing with the imbalanced problem (Chawla, 2004). Yan et al. (2003) proposed the SVM ensembles method that is another example of the use of advanced sampling in SVM. Tang et al. (2009) proposed the Granular Support Vector Machines-Repetitive Under-sampling (GSVM-RU) algorithm that combines classification algorithm of Granular Support Vector Machines (GSVM) (Tang et al., 2005) and under-sampling method (Kubat and Matwin, 1997). Wu and Chang (2005) proposed an algorithm of Kernel Boundary Aligment (Cristianini et al., 2006) that adjusts the kernel matrix to fit the training data. Akbani et al. (2004) combined SMOTE (Synthetic Minority Over-sampling Technique) (Chawla et al., 2002) and Different Cost (Veropoulos, 1999) as the SMOTE Difference Cost (SDC) for SVM, Li et al. (2005 and 2008) designed the strategy of parameter adjusting in AdaBoost algorithm for SVM specifically, i.e. AdaBoostSVM, Wang et al. (2010) proposed a new loss function for SVM to adjust SVM solution by taking into account the sample sizes of the two class, Maalouf and Trafalis (2011) suggested Rare Event Weighted Kernel Logistic Regression (RE-WKLR) algorithm that combines TR-KLR (Maalouf et al., 2010), weighting and bias correction. Meanwhile, in probabilistic non-kernel methodology, King and Zeng (2001a) proposed Rare Event Logistic Regression (RE-LR) that combines weighting and bias correction, and Owen (2007) developed Infinitely Imbalanced Logistic Regression (IILR) by reformulating the likelihood unconstrained optimization problem as the Gaussian Mixture Model (GMM) optimization problem.

Most of those aforementioned methods were applied for the imbalanced SVM-based methods which resulted in good accuracy performance for classification on imbalanced data sets. However, these methods typically need to be solved by constrained optimization problem which has complex solution. IILR (Owen, 2007) has also complex optimization problem. RE-WKLR (Maalouf and Trafalis, 2011) and its non-kernel version i.e. RE-LR (King and Zeng, 2001a) are the imbalanced LR-based methods. Both classifiers keep the use of unconstrained optimization problem which only requires to be solved by simple solution. However, the imbalanced techniques used have led their accuracy performances for classification of imbalanced data sets that still require to be improved. Furthermore, based on aforementioned methods in general, the

number of research of imbalanced LR-based methods are not as many as that of imbalanced SVM-based methods.

2.3 RLR-IRLS AND KLR-IRLS WITH $y \in (0,1)$

Logistic Regression (LR) is the fundamental method of *classification* which is predictive task whose qualitative labels. Supposing there are n pairs of training data ($\mathbf{x}_i$, y_i), where $\mathbf{x}_i$ is input vector with dimension d (number of features) for i^{th} instance and corresponding label y_i. Considering a binary or two-class classification problem $y_i \in \{0,1\}$, for every instance $\mathbf{x}_i$, the label is either y_i =0 or y_i=1, i=1,2.., n. The instance $\mathbf{x}_i$ belongs to class 1 with probability $p(y_i=1|\mathbf{x}_i)$ and it belongs to class 0 with probability $p(y_i=0|\mathbf{x}_i) = 1 - p(y_i=1|\mathbf{x}_i)$ (Hosmer and Lemeshow, 2000).

In binary LR, it is required to model the posterior probability of two classes via linear functions (Hastie et al., 2001).

$$f(\mathbf{x}_i) = \mathbf{X}_i\boldsymbol{\beta} \tag{2.1}$$

where $\boldsymbol{\beta}$ denotes the coefficient vector with size (dim+1) x 1, including the bias term, while $\mathbf{X}_i = [\mathbf{x}_i\ 1]$.

The output of linear function, $f(\mathbf{x}_i)$, can be interpreted as an estimate posterior probability of two class by the use of logit transformation to eq. (2.1),

$$\ln\left\{\frac{\pi(\mathbf{x}_i)}{1-\pi(\mathbf{x}_i)}\right\} = f(\mathbf{x}_i) \tag{2.2}$$

Such that its results in $\pi(\mathbf{x}_i) = p(y_i=1|\mathbf{x}_i,\boldsymbol{\beta})$

$$p_i = \frac{\exp(f(\mathbf{x}_i))}{1+\exp(f(\mathbf{x}_i))}$$

$$= \frac{1}{1+\exp(-f(\mathbf{x}_i))} \tag{2.3}$$

$$1-\pi(\mathbf{x}_i) = 1 - p(y_i=1|\mathbf{x}_i,\boldsymbol{\beta})$$
$$= p(y_i=0|\mathbf{x}_i,\boldsymbol{\beta})$$
$$= \frac{1}{1+\exp(f(\mathbf{x}_i))} \quad (2.4)$$

Logistic response function on eq. (2.3) maps the values of linear model, $f(\mathbf{x})$, to the range [0,1] as can be displayed on Fig. 2.1.

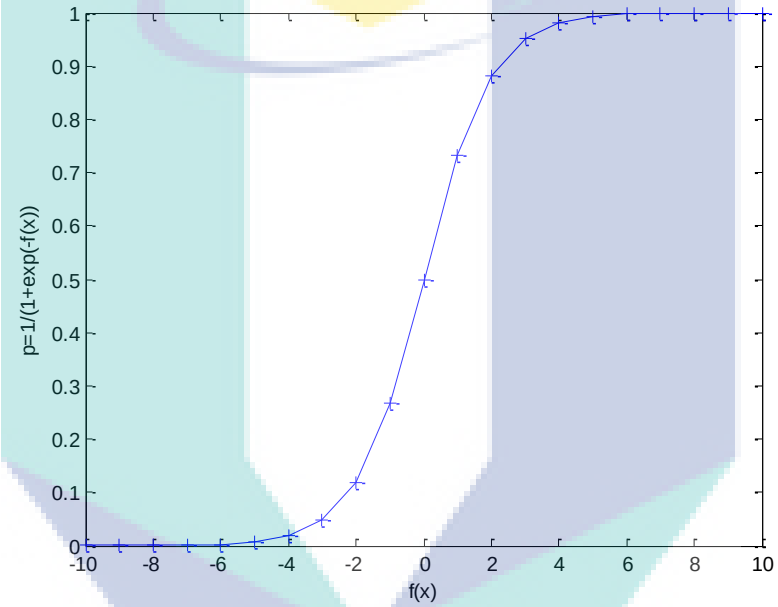


Figure 2.1: Logistic Response Function

Moreover, the logistic response function implicitly places $f(\mathbf{x}) = 0$ as a separating hyperplane between class 0 and class 1, and the classification rule will be described in subsection 2.3.1.

2.3.1 Regularized optimization function of RLR and KLR

For those pairs ($\mathbf{x}_i$, y_i) where y_i=1, the conditional probability is $\pi(\mathbf{x}_i)$, and for those pairs where y_i=0, the conditional probability is $1-\pi(\mathbf{x}_i)$. Therefore, ($\mathbf{x}_i$, y_i) is assumed to follow Bernoulli distribution (Hosmer and Lemeshow, 2000).

$$\xi(x_i) = \pi(\mathbf{x}_i)^{y_i}\left[1-\pi(\mathbf{x}_i)\right]^{1-y_i} \tag{2.5}$$

LR models are usually fit by Maximum Likelihood Estimation (MLE) (Garthwaite et al., 2002, Hogg, 1994) method. Since the observations are assumed to be independent, the likelihood function is obtained as the product of the terms given in eq. (2.6) as follows;

$$l(\boldsymbol{\beta}) = \prod_{i=1}^{n}\left(\pi(\mathbf{x}_i)\right)^{y_i}\ \left(1-\pi(\mathbf{x}_i)\right)^{1-y_i} \tag{2.6}$$

The principle of maximum likelihood states that estimate of $\boldsymbol{\beta}$ is the value which maximizes the expression in eq. (2.6). MLE estimates of LR can be found by maximizing the log-likelihood optimization function and set its derivatives to zero. However, it is mathematically easier to work with the log of eq. (2.6). This expression, *the log likelihood*, is defined as,

$$\begin{aligned} L(\boldsymbol{\beta}) = \log l(\boldsymbol{\beta}) &= \sum_{i=1}^{n}\left\{y_i \log[\pi(\mathbf{x}_i)] + (1-y_i)\log[1-\pi(\mathbf{x}_i)]\right\} \\ &= \sum_{i=1}^{n}\left\{y_i \log\left[\frac{\exp(\mathbf{X}_i\boldsymbol{\beta})}{1+\exp(\mathbf{X}_i\boldsymbol{\beta})}\right] + (1-y_i)\log\left[\frac{1}{1+\exp(\mathbf{X}_i\boldsymbol{\beta})}\right]\right\} \\ &= \sum_{i=1}^{n}\left\{y_i(\mathbf{X}_i\boldsymbol{\beta}) - \log[1+\exp(\mathbf{X}_i\boldsymbol{\beta})]\right\} \end{aligned} \tag{2.7}$$

Over-fitting training data may arise in LR, especially when the data are very high dimensional and/or sparse (Zhang and Yang, 2003). Quadratic regularization (Hoerl and Kennard, 1973) is one of most popular methods to control the bias-variance trade-off (Geman et al., 1992), by introducing a penalty on large fluctuations of MLE estimate. Hence, in order to avoid the over-fitting problem and to obtain better generalization, it is necessary to add quadratic regularization to the log likelihood function such that we have regularized log likelihood. (Maalouf, 2009; Park and Hastie, 2008)

RLR

In RLR, the regularized optimization problem can be stated as (Lee and Silvapulle, 1988; Le Cessie, S. and Van Houwelingen, 1992).

$$L(\boldsymbol{\beta}) = \mathbf{1}^T\left((\mathbf{y}.(\mathbf{X}\boldsymbol{\beta})) - \log(1+\exp(\mathbf{X}\boldsymbol{\beta}))\right) - \frac{\lambda}{2}\left\|\boldsymbol{\beta}^T\boldsymbol{\beta}\right\| \tag{2.8}$$

where :

y is vector of labels with dimension *n* x 1

β is vector of coefficient with dimension (*d*+1) x 1, including the bias term.

X is matrix [**x 1**]

$f(\mathbf{x}) = \mathbf{X}\boldsymbol{\beta}$

λ = regularization parameter ($\lambda > 0$), where bias term is not regularized

The Hessian matrix of eq. (2.8) is positive definite, so the regularized function of RLR is convex optimization function (Boyd and Vandenberghe, 2004; Lin et al., 2008). Hence, there is only one solution which is global minimum.

The loss function or the deviance, DEV, for RLR is given by formula

$$\begin{aligned} DEV(\boldsymbol{\beta}) &= -2\log l(\boldsymbol{\beta}) \\ &= -2L(\boldsymbol{\beta}) \end{aligned} \tag{2.9}$$

KLR

The optimization function of KLR can be obtained by kernelizing the optimization function of RLR on eq. (2.8), based on the Representer Theorem (Cawley and Talbot, 2005; Cawley and Talbot, 2008; Scholkopf et al., 2002);

$$\boldsymbol{\beta} = \varphi(\mathbf{x})\boldsymbol{\alpha} \tag{2.10}$$

where $\varphi(\mathbf{x})$ is a function to map the original data **x** in input space into feature space, in order to convert the non-linear relation into linear relation.

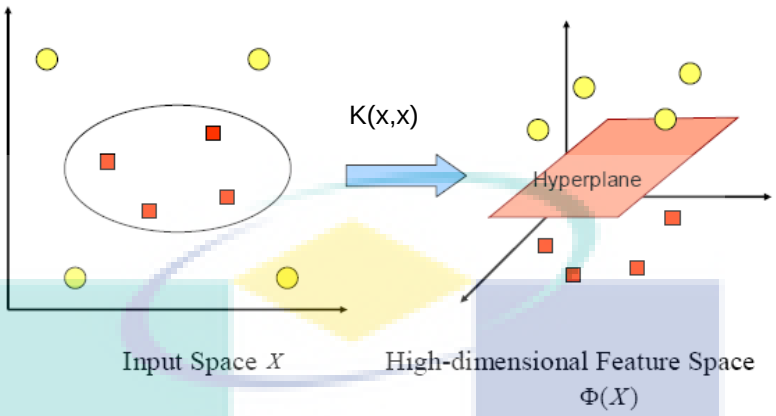


Figure 2.2: Kernel trick

Source: Nugroho et al. (2003)

The dot product in feature space can be expressed in terms of input vectors through the kernel function. The Kernel function is a transformation function that must satisfy Mercer's necessary and sufficient conditions (Mercer, 1970). A kernel function must be expressed as inner product and must be positive semi-definite. The decision function (logit model) of KLR, *f(x)*, can be expressed in the form,

$$\begin{aligned} f(\mathbf{x}) &= \beta^T \varphi(\mathbf{x}) \\ &= \varphi(\mathbf{x})^T \varphi(\mathbf{x})\boldsymbol{\alpha} \\ &= \mathbf{K}\boldsymbol{\alpha} \end{aligned} \tag{2.11}$$

where **K** is a Kernel matrix. Each cell of Kernel matrix, k_{ij}, is an inner product between individuals *i* and *j* that holds a measure of similarity (Tenenhaus et al., 2007).

By substituting eq. (2.10) and eq. (2.11) on eq. (2.8), the regularized optimization function of KLR can be defined as (Katz et al., 2006; Maalouf et al. 2010).

$$L(\boldsymbol{\alpha}) = \mathbf{1}^T\left((\mathbf{y}.(\mathbf{K}_1\boldsymbol{\alpha})) - \log(1+\exp(\mathbf{K}_1\boldsymbol{\alpha}))\right) - \frac{\lambda}{2}\left\|\boldsymbol{\alpha}^T\mathbf{K}_2\boldsymbol{\alpha}\right\| \quad (2.12)$$

where :

y is vector of label with dimension *n* x 1

α is vector of coefficient with dimension (*d*+1) x 1, including the bias term.

$\mathbf{K}_1$ is matrix [**K 1**]

$f(\mathbf{x}) = \mathbf{K}_1\boldsymbol{\alpha}$

λ = regularization parameter

$\mathbf{K}_2$ is matrix $\begin{bmatrix}\mathbf{K} & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix}$ (the bias term is not regularized)

K is kernel matrix of size *n* x *n*. Two of the most popular kernel functions are Linear kernel and RBF kernel.

Similar to RLR, the regularized function of KLR is convex optimization function (Cawley and Talbot, 2008), with the deviance, DEV, (Maalouf et al., 2010).

$$\begin{aligned} DEV(\boldsymbol{\alpha}) &= -2\log l(\boldsymbol{\alpha}) \\ &= -2L(\boldsymbol{\alpha}) \end{aligned} \quad (2.13)$$

Minimizing the deviance is equivalent to maximizing the log-likelihood (Hosmer and Lemeshow, 2000). As the non-linearity of MLE estimates, minimizing the deviance of RLR and KLR requires numerical method, such as the Iteratively Re-weighted Least Squares (IRLS) method, in order to find MLE estimates.

Once the optimal MLE estimates for RLR and KLR are found, classification of given instance, $\mathbf{x}_i$, is carried according to the following rules;

$$\begin{aligned} \hat{y}_i &= 1, \text{ if } \hat{f}(\mathbf{x}_i) \geq 0 \text{ or } \hat{p}_i \geq 0 \\ &= 0, \text{ otherwise} \end{aligned} \quad (2.14)$$

2.3.2 IRLS method for RLR and KLR

The IRLS method is an algorithm that iteratively solves the linear system of Weighted Least Squares (WLS) problem. The IRLS method is a representation of Newton-Raphson method (Hastie et al. 2001; Nabney, 1999; Roth, 2001);

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} + \mathbf{s}^{(t)} \\ &= \boldsymbol{\theta}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)}\end{aligned} \tag{2.15}$$

where t is the iteration index, while $\mathbf{g}$ and $\mathbf{H}$ are the gradient and the Hessian which is achieved by differentiating the regularized NLL function with respect to θ.

RLR with IRLS method (RLR-IRLS)

The IRLS method for RLR can be defined as (Hastie et al., 2001; Park et al., 2008);

$$\begin{aligned}\boldsymbol{\beta}^{(t+1)} &= \boldsymbol{\beta}^{(t)} + \mathbf{s}^{(t)} \\ &= \boldsymbol{\beta}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)} \\ &= \boldsymbol{\beta}^{(t)} - \left(\frac{\partial^2 L(\boldsymbol{\beta})}{\partial(\boldsymbol{\beta})\partial(\boldsymbol{\beta}^T)}\right)^{-1}\left(\frac{\partial L(\boldsymbol{\beta})}{\partial(\boldsymbol{\beta})}\right) \\ &= \boldsymbol{\beta}^{(t)} - \left(-\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{X} - \lambda\mathbf{I}\right)^{-1}\left(\mathbf{X}^T\left(\mathbf{y} - \mathbf{p}^{(t)}\right) - \lambda\boldsymbol{\beta}^{(t)}\right)\end{aligned} \tag{2.16}$$

where $\mathbf{V}$ = diag($\mathbf{p}$.(1-$\mathbf{p}$)) with size n x n

Such that the linear system of WLS problem on using IRLS method for RLR can be written as;

$$\begin{aligned}\left(\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{X} + \lambda\mathbf{I}\right)\boldsymbol{\beta}^{(t+1)} &= \mathbf{X}^T\mathbf{V}^{(t)}\left(\mathbf{X}\boldsymbol{\beta}^{(t)} + \mathbf{V}^{-1(t)}\left(\mathbf{y} - \mathbf{p}^{(t)}\right)\right) \\ &= \mathbf{X}^T\mathbf{V}^{(t)}\mathbf{Z}^{(t)}\end{aligned} \tag{2.17}$$

where **Z** is known as adjusted response.

KLR with IRLS method (KLR-IRLS)

The IRLS method for KLR can be expressed as (Cawley and Talbot, 2008; Roth, 2001;);

$$\begin{aligned}\boldsymbol{\alpha}^{(t+1)} &= \boldsymbol{\alpha}^{(t)} + \mathbf{s}^{(t)} \\ \boldsymbol{\alpha}^{(t+1)} &= \boldsymbol{\alpha}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)} \\ &= \boldsymbol{\alpha}^{(t)} - \left(\frac{\partial^2 L(\boldsymbol{\alpha})}{\partial(\boldsymbol{\alpha})\partial(\boldsymbol{\alpha}^T)}\right)^{-1}\left(\frac{\partial L(\boldsymbol{\alpha})}{\partial(\boldsymbol{\alpha})}\right) \\ &= \boldsymbol{\alpha}^{(t)} - \left(-\mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{K}_1 - \lambda\mathbf{K}_2\right)^{-1}\left(\mathbf{K}_1{}^T\left(\mathbf{y}-\mathbf{p}^{(t)}\right) - \lambda\mathbf{K}_2\boldsymbol{\alpha}^{(t)}\right) \qquad (2.18)\end{aligned}$$

Therefore, the linear system of WLS on using IRLS method for KLR can be stated as;

$$\begin{aligned}\left(\mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{K}_1 + \lambda\mathbf{K}_2\right)\boldsymbol{\alpha}^{(t+1)} &= \mathbf{K}_1{}^T\mathbf{V}^{(t)}\left(\mathbf{K}_1\boldsymbol{\alpha}^{(t)} + \mathbf{V}^{-1(t)}\left(\mathbf{y}-\mathbf{p}^{(t)}\right)\right) \\ &= \mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{Z}^{(t)} \qquad (2.19)\end{aligned}$$

The Hessian matrix of RLR and KLR are dense, such that the iterative computation could become unacceptably slow. It is the numerical problem on using the IRLS method for RLR and KLR in order to find the WLS solution. (Komarek, 2004; Maalouf, 2009)

2.4 RLR-IRLS AND KLR-IRLS WITH TRUNCATED NEWTON METHOD

For solving large scale data of nonlinear optimization problem, such as MLE, Truncated-Newton method is a suitable method. Truncated Newton method contains a doubly iterative method: an outer iteration and an inner iteration. Komarek and Moore (2005) was the first to propose the implementation of Truncated Newton, by Linear Conjugate Gradient (Linear CG) approach, as the inner iteration, to approximate the WLS solution on using IRLS method, that as the outer iteration, for search the MLE of RLR. As mentioned earlier, the proposed method is called TR-IRLS. The early stopping

of Linear CG iteration is referred as the Truncated Newton method with accompanying convergence guarantees (Komarek and Moore, 2005; Lewis et al., 2006). Inspired by the effectiveness of TR-IRLS for solving large scale data of classification problems, Maalouf et al. (2010) then proposed to combine the speed of TR-IRLS with the accuracy generated by the use of kernel method for solving non-linear classification problems that resulted in TR-KLR.

Linear CG method is almost always used as an inner iterative algorithm in a Truncated Newton method, such as TR-IRLS or TR-KLR. CG method is an optimal iterative method for solving a positive-definite linear system **AP** = **b**. It means that the i^{th} iteration of $\mathbf{P}_i$ minimizes the associated quadratic function, $Q(\mathbf{P}) = \frac{1}{2}\mathbf{P}^T\mathbf{AP} - \mathbf{P}^T\mathbf{b}$ (Nash, 2000). Recent studies have showed that the Conjugate Gradient (CG) method provides better results to estimate RLR model than any other numerical methods (Malouf, 2002; Minka, 2003). CG only requires computation of matrix-vector products such that simplifying the computation. The use of CG method has an advantage that it guarantees convergence in at most *n* steps (Lewis et al., 2006). Linear CG is the application of CG to find the optimal value of quadratic form. The numerical problem of IRLS method for RLR and KLR, respectively, can be solved by the use of Linear CG method to quadratic form of WLS linear system (Komarek and Moore, 2005; Maalouf et al. 2010).

In general, TR-IRLS and TR-KLR algorithm respectively contains two loops. First algorithm represents the outer loop (main algorithm) that finds the MLE by using IRLS method. The main algorithm is terminated when the relative difference of optimization function is no larger than a specified threshold, ε_1. Second algorithm represents the truncated inner loop that solves the linear system of WLS problem by using the Linear CG method to approximate the WLS solution on first algorithm. This algorithm is terminated when the square residual is no greater than a specified threshold, ε_2.

2.4.1 TR-IRLS: RLR-IRLS with Truncated Newton method

In TR-IRLS, solving the linear system of WLS problem on eq. (2.17) for RLR-IRLS by using the approach of Linear CG, as the truncated inner algorithm, is equivalent to minimizing the quadratic form,

$$\frac{1}{2}\boldsymbol{\beta}^{T^{(t+1)}}\left(\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{X}+\lambda\mathbf{I}\right)\boldsymbol{\beta}^{(t+1)}-\boldsymbol{\beta}^{T^{(t+1)}}\left(\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{Z}^{(t)}\right) \tag{2.20}$$

For Algorithm 2.1.1 (RLR-IRLS), the maximum number of iterations for RLR-NR is set to 30, while for Algorithm 2.1.2 (Linear CG for RLR-IRLS), the maximum number of iterations for Linear CG is set to 200 iterations. Moreover, threshold of the difference of optimization function values for RLR-NR in Algorithm 1 is set to 0.01 (ε_1) and the convergence threshold for Linear CG in Algorithm 2 is set to 0.005 (ε_2). The success to control a trade off between convergence speed and accuracy has been shown by the use of these default parameter values. (Komarek and Moore., 2005; Maalouf, 2009).

Algorithm 2.1 TR-IRLS

Algorithm 2.1.1. RLR-IRLS (Outer loop)

Input : **X**, **y**, λ

Initialize : $\boldsymbol{\beta}^{(1)}$, $DEV^{(1)}$

Output : $\boldsymbol{\beta}^{(\tau+1)}$

Do while $\left|\frac{DEV^{(t+1)}-DEV^{(t)}}{DEV^{(t+1)}}\right|>\varepsilon_1$

For t = 1 to *max RLR-IRLS iterations*

(1) Compute probability: $\mathbf{p}^{(t)} = 1./(1+\exp(-\mathbf{X}\boldsymbol{\beta}^{(t)}))$

(2) Compute variance: $\mathbf{V}^{(t)} = \text{diag}(\mathbf{p}^{(t)}.(1-\mathbf{p}^{(t)}))$

(3) Compute WLS solution, $(\mathbf{X}^T\mathbf{V}\mathbf{X}+\lambda\mathbf{I})\boldsymbol{\beta}^{(\tau+1)}=(\mathbf{X}^T\mathbf{V}\mathbf{Z})$

(4) Compute $DEV^{(t+1)}$

End For

where ε_1=0.01, *max RLR- IRLS iterations* = 30

Algorithm 2.1.2 Linear CG for RLR-IRLS (Truncated inner loop)

Input : $\mathbf{A}= (\mathbf{X}^T\mathbf{V}\mathbf{X}+\lambda\mathbf{I})$ and $\mathbf{b}=(\mathbf{X}^T\mathbf{V}\mathbf{Z})$

Initialize: $\beta^{(1)}$, $\mathbf{r}^{(1)}=\mathbf{b}$, $\mathbf{d}^{(1)}= \mathbf{r}^{(1)}$

Output : $\beta^{(\tau+1)}$

Do while $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length: $a^{(\tau)} = \mathbf{r}^{T(t)}\mathbf{r}^{(t)} / (\mathbf{d}^{T(t)}\mathbf{A}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution: $\beta^{(\tau+1)} = \beta^{(\tau)} + \mathbf{a}^{(\tau)}\mathbf{d}^{(t)}$

(3) Update the residual: $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\mathbf{A}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer: $\mathbf{c}^{(t)} = \mathbf{r}^{T(t+1)}\mathbf{r}^{(t+1)} / \mathbf{r}^{T(t)}\mathbf{r}^{(t)}$

(5) Update the search direction: $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 200

2.4.2 TR-KLR: KLR-IRLS with Truncated Newton method

In TR-KLR, solving the linear system of WLS problem in eq. (2.19) for KLR-IRLS by using the approach of Linear CG is equivalent to minimizing the quadratic form,

$$\frac{1}{2}\boldsymbol{\alpha}^{T(t+1)}\left(\mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{K}_1 + \lambda\mathbf{K}_2\right)\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^{T(t+1)}\left(\mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{Z}^{(t)}\right) \qquad (2.21)$$

Except the value of ε_1, all setting of TR-KLR are similar to TR-IRLS algorithm above. For TR-KLR, threshold of the difference of optimization function values in Algorithm 2.2.1 is set to 2.5 (ε_1). The use of these default parameter values have shown adequate accuracy and also maintained good convergence speed. However, to obtain better accuracy in some cases, it may be advisable to make this threshold smaller (Maalouf et al., 2010)

Algorithm 2.2 TR-KLR

Algorithm 2.2.1 KLR-IRLS (Outer loop)

Input: **X**, $\mathbf{K_1}$, **y**, λ, σ

Initialize:$\boldsymbol{\alpha}^{(1)}$, $DEV^{(1)}$,

Output : $\boldsymbol{\alpha}^{(\tau+1)}$

Do while $\left|\frac{DEV^{(t+1)} - DEV^{(t)}}{DEV^{(t+1)}}\right| > \varepsilon_1$

For t = 1 to *max KLR-NR iterations*

(1) Compute probability: $\mathbf{p}^{(t)} = 1./(1+\exp(\mathbf{y}.\mathbf{K}_1\boldsymbol{\alpha}^{(t)}))$

(2) Compute variance : $\mathbf{V}^{(t)} = \text{diag}(\mathbf{p}^{(t)}.(1-\mathbf{p}^{(t)}))$

(3) Compute WLS solution : $(\mathbf{K}_1^{\mathrm{T}}\mathbf{V}\mathbf{K}_1+\lambda\mathbf{K}_2)\boldsymbol{\alpha}^{(t+1)}=(\mathbf{K}_1^{\mathrm{T}}\mathbf{V}\mathbf{Z})$

(4) Compute $DEV^{(t+1)}$

End For

where ε_1=2.5, *max KLR- NR iterations* = 30

Algorithm 2.2.2 Linear CG for KLR-IRLS (Truncated inner loop)

Input : $\mathbf{A}= (\mathbf{K}_1^{\mathrm{T}}\mathbf{V}\mathbf{K}_1+\lambda\mathbf{K}_2)$ and $\mathbf{b}=(\mathbf{K}_1^{\mathrm{T}}\mathbf{V}\mathbf{Z})$

Initialize: $\boldsymbol{\alpha}^{(1)}$, $\mathbf{r}^{(1)}=\mathbf{b}$, $\mathbf{d}^{(1)}= \mathbf{r}^{(1)}$

Output : $\boldsymbol{\alpha}^{(\tau+1)}$

Do while $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length : $a^{(\tau)} = \mathbf{r}^{\mathrm{T}(t)}\,\mathbf{r}^{(t)}/\,(\mathbf{d}^{\mathrm{T}(t)}\,\mathbf{A}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution : $\boldsymbol{\alpha}^{(\tau+1)} = \boldsymbol{\alpha}^{(\tau)} + \mathbf{a}^{(\tau)}\,\mathbf{d}^{(t)}$

(3) Update the residual : $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\,\mathbf{A}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer : $\mathbf{c}^{(t)} = \mathbf{r}^{\mathrm{T}(t+1)}\,\mathbf{r}^{(t+1)} / \mathbf{r}^{\mathrm{T}(t)}\mathbf{r}^{(t)}$

(5) Update the search direction : $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\,\mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 200

2.5 ADAPTIVE BOOSTING METHOD

AdaBoost (Freund and Schapire, 1997) is the most popular Boosting method (Freund, 1993; Freund, 1995; Schapire, 1990; Schapire, 1992). Beside Bootstraap Agregating (Bagging) (Breiman, 1998), Boosting is the commonly used ensemble method (Opiz and Maclin, 1999), which is one of the major recent developments in machine learning and data mining community for classification task (Friedman et al., 2000). Ensemble method is a technique to combine the moderate predictions of the multiple component classifiers to produce a single classifier (an ensemble) which has generally highly accurate (strong) prediction such that better, i.e. low bias and variance (Bauer and Kohavi; Friedman et al. 2000; Oza, 2001; Oza and Russel, 2001; Schapire et al., 1998), than any of the individual classifier making up the ensemble. AdaBoost creates an ensemble, collection of component classifiers, by maintaining a set of weights over training data and adaptively adjusting these weights after each Boosting iterations (Li et al., 2008).

Supposing there are *n* pairs of training data ($\mathbf{x}_i, y_i$) with $y_i \in (-1,1)$, AdaBoost combines many component classifiers to develop an ensemble as follows (Zhou and Wei, 2009);

$$F(\mathbf{x}) = \sum_{t=1}^{T} \omega_t h_t(\mathbf{x}) \tag{2.21}$$

where $h_t(\mathbf{x})$ is the weighted prediction of component classifier, $h_t(\mathbf{x}_i) \in (-1,1)$

ω_t is the importance factor indicating the contribution of corresponding component classifier to an ensemble ($\omega_t > 0$)

T is the number of AdaBoost iterations, i.e. the number of component classifiers in ensemble

Moreover, given the instance $\mathbf{x}_i$, predictions of AdaBoost classifier can be found by

$$y_{pred}(\mathbf{x}_i) = sign(F(\mathbf{x}_i)) \tag{2.22}$$

Thus, the classification rule of AdaBoost classifier is determined by a weighted majority vote of T component classifiers that have moderate total accuracy values, where each ω_t is a weight assigned to each $h_t(\mathbf{x})$. The voting process removes the uncorrelated error of moderately accurate component classifiers such that it leads to better generalization performance of AdaBoost. Moderately accurate component classifiers often have larger diversity (larger disagree with each other or lower correlated error) than those component classifiers which are very accurate (Li et al., 2008; Shin and Soghn, 2005).

The AdaBoost algorithm estimates ω_t in a stepwise manner. Detail algorithm of AdaBoost is given below (Freund and Schapire, 1997).

Algorithm 2.3 AdaBoost

Input : **x**, **y**

Initialize: The weight of training samples $\mathbf{w}_t(i)$, where $\mathbf{w}_1(i)$= 1/n; i = 1,2,..n

Output : the class prediction, $y_{pred}(\mathbf{x}) = sign\left(\sum_t^T \omega_t h_t(\mathbf{x})\right)$

Do while ($t < T$)

(1) Get the model, f_t, and then get the weighted prediction, h_t, by performing component classifier on weighted training data

(2) Calculate the weighted error of h_t : $\varepsilon_t = \sum_{i=1}^{n} w_t(i),\ y_i \neq h_t(\mathbf{x})$

and choose h_t with minimal error

(3) Set the weight of h_t: $\omega_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

(4) For i = 1 to n

Update the weight of training samples:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} x \begin{cases} \exp(\omega_t), & y_i \neq h_t(\mathbf{x}_i) \\ \exp(-\omega_t), & y_i = h_t(\mathbf{x}_i) \end{cases}$$

Z_t is normalization factor

End For

End For

As mentioned earlier in this section, AdaBoost maintains and adaptively adjust a set of weights over training data after each Boosting iterations. Hence, component classifier should be designed for taking into account the weight distribution in training process. All the weights $w_t(i)$ are kept as probability distribution such that data are processed according to a probability distribution.

In practice, there are two ways to process training data and the corresponding probability distribution (Zhou and Wei, 2009):

(i) A component classifier can be trained using corresponding weights on training data directly by using weighting technique.

(ii) Corresponding weights can not be applied on algorithm of component classifier directly, so that re-sampling is used to bypass the difficulty. The training data are re-sampled according to the probability distribution, and these re-sampled data are used to train a component classifier.

Algorithm of AdaBoost can be described as follows: at the beginning of iteration, component classifier is trained by using weight distribution on training data. Initially, weights of training data are set equally to $\frac{1}{n}$ ($w_1(i)$). The weighted prediction of component classifier (h_t) that has minimal weighted error (ε_t) is selected.

$$\varepsilon_t = \sum_{i=1}^{n} w_t(i),\ y_i \neq h_t(\mathbf{x}) \tag{2.23}$$

The weighted error should be slightly less than 0.5 (50%). Weight ($w_t(i)$) is added in ε_t, when samples are misclassified $\left(y_i h_t(\mathbf{x}_i) < 0\right)$. The importance of component classifier is then evaluated by

$$\omega_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t} \tag{2.24}$$

Since ε_t should be slightly less than 0.5, ω_t will be greater than zero.

In AdaBoost algorithm, the weights, $w_t(i)$, is updated using the rule: weights of training samples which are misclassified $\left(y_i h_t(\mathbf{x}_i) < 0\right)$ by current component classifier will be increased while weights of training data which are correctly classified $\left(y_i h_t(\mathbf{x}_i) > 0\right)$ will be decreased. The formula to update weights can be written as,

$$w_{t+1}(i) = \frac{w_t(i)\exp\left(-\omega_t y_i h_t(\mathbf{x}_i)\right)}{Z_t} \tag{2.25}$$

where $Z_t = \sum_{i=1}^{n} w_i \exp(-\omega_t y_i h_t(\mathbf{x}_i)$ is the normalization factor such that $w_{t+1}(i)$ is a probability distribution, $\sum_{i}^{n} w_{t+1}(i) = 1$. AdaBoost processes component classifier repeatedly until T iterations. Finally, an ensemble is combined linearly by these trained component classifiers with corresponding weights.

The effect of weight update rule in AdaBoost algorithm is to reduce the training error of an ensemble during Boosting iterations (Iyer, 1999). It can be shown that the training error drops exponentially (Schapire and Singer, 1999):

$$\frac{1}{n}\left|i : \left\{y_{pred}(\mathbf{x}_i) \neq y_i\right\}\right| \leq \frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i F(\mathbf{x}_i)\right) = \prod_{i=1}^{n} Z_t \tag{2.26}$$

By unravelling the update rule in eq. (2.25), it is found that,

$$= \frac{\exp\left(-y_i \sum_{t=1}^{T} \omega_t h_t(\mathbf{x}_i)\right)}{n\prod_{t=1}^{T} Z_t}$$

$$= \frac{\exp\left(-y_i F(\mathbf{x}_i)\right)}{n\prod_{t=1}^{T} Z_t} \tag{2.27}$$

Moreover, if $y_i \neq y_{pred}(\mathbf{x}_i)$ then $y_i F(\mathbf{x}_i) \leq 0$ implying that $\exp(-y_i F(\mathbf{x}_i)) \geq 1$, thus

$$\lfloor y_{pred}(\mathbf{x}_i) \neq y_i \rfloor \leq \exp(-y_i F(\mathbf{x}_i)) \tag{2.28}$$

The inequality in eq. (2.26) can be described by combining eq. (2.27) and eq. (2.28)

$$\frac{1}{n}\sum_{i=1}^{n}\lfloor y_{pred}(\mathbf{x}_i) \neq y_i \rfloor \leq \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i F(\mathbf{x}_i)) = \sum_{i=1}^{n} w_{t+1}(i)\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t$$

In order to minimize training error rapidly, eq. (2.26) suggests that h_t and ω_t should be chosen on round t to minimize the normalization factor

$$Z_t = \sum_{i=1}^{n} w_i \exp(-\omega_t y_i h_t(\mathbf{x}_i)) \tag{2.29}$$

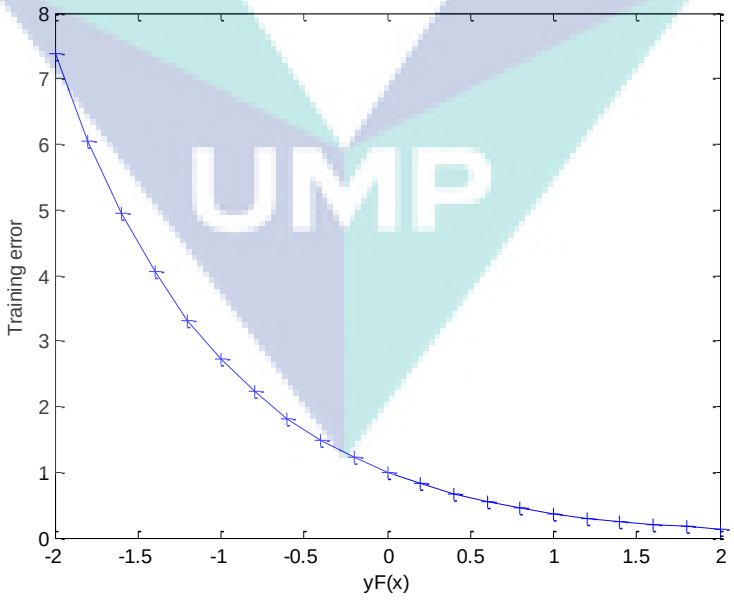


Figure 2.3: Training error of AdaBoost

(a) Finding ω_t

We attempt to minimize Z_t by finding ω_t as follows,

$$\frac{dZ_t}{d\omega_t} = -\sum w_t(i) y_i h_t(\mathbf{x}_i) \exp(-y_i \omega_i h_t(\mathbf{x}_i) = 0$$

$$-\sum_{i:y_i = h_t(x)} w_t(i) \exp(-\omega_t) + \sum_{i:y_i \neq h(x)} w_t(i) \exp(\omega_t) = 0$$

$$-(1-\varepsilon_t)\exp(-\omega_t) + \varepsilon_t \exp(\omega_t) = 0$$

$$\omega_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

The smaller error of h_t (ε_t), the larger the importance of component classifier (ω_t) (Fig. 2.4). This indicates that a component classifier that has slightly "stronger" discriminating power plays a more important role in the ensemble for making decision.

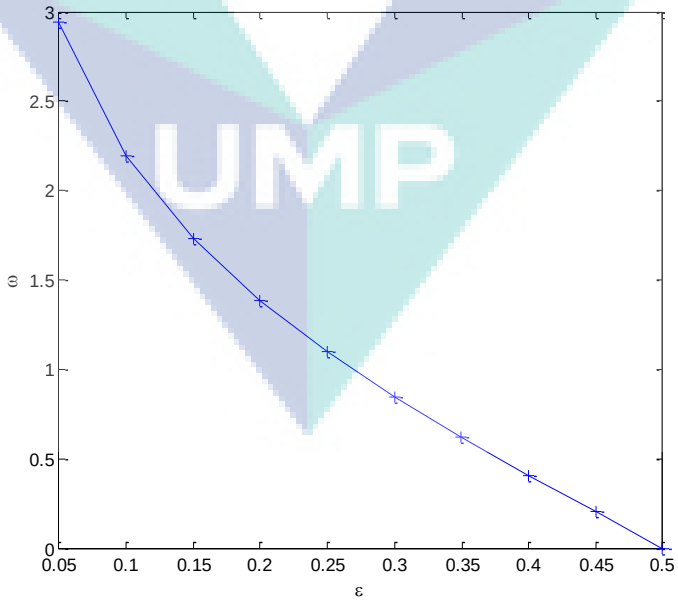


Figure 2.4: Plot ω vs ε

(b) Choosing h_t

Z_t is minimized by selecting h_t with minimal weighted error (ε_τ). Justification of weighted error minimization can be described from eq. (2.29) as follows

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} w_i \exp(-\omega_t y_i h_t(\mathbf{x}_i)) \\ &= \sum_{i:y_i=h_t(x)} w_t(i)\exp(-\omega_t) + \sum_{i:y_i\neq h(x)} w_t(i)\exp(\omega_t) \\ &= (1-\varepsilon_t)\exp(-\omega_t) + \varepsilon_t \exp(\omega_t) \end{aligned} \tag{2.30}$$

By finding ω_t as has been described above, the upper bound on training error (eq. 2.26) is simplified to

$$\prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \tag{2.31}$$

where γ_t measures how much better the h_t's prediction compare to the random, that is $\varepsilon_t = \frac{1}{2} - \gamma_t$. This property means that AdaBoost is able to improve in efficiency if any component classifiers have weighted error slightly lower than the worst case error $\frac{1}{2} - \gamma_t$.

2.6 ADABOOST ALGORITHM FOR SVM

Analysis of boosting techniques shows that Boosting is tied to the choice of component classifier (Joshi et al., 2000; Joshi et al., 2001). This section describes two AdaBoost algorithms that are specifically designed for Support Vector Machine (SVM) with RBF kernel parameter, i.e. AdaBoostSVM and WwBoost-SVM. Both algorithms are the foundation on developing proposed AdaBoost algorithms for DTR-KLR and DTR-LR respectively.

SVM was developed from the theory of Structural Risk Minimization (Vapnik, 1998 and 2000) that has drawn considerable attentions in various research areas. By using a kernel trick to map the training samples from an input space to a high-dimensional feature space, SVM finds an optimal separating hyperplane in the feature spaces with solving constrained optimization problem. SVM also uses a regularization parameter (C) to control its model complexity and training error.

The total accuracy value of SVM classifier is often high and cannot meet the requirement on a component classifier given in AdaBoost that needs to be only slightly better than 50%. Due to the fact that SVM is strong classifier (highly accurate classifier), AdaBoost is not expected to improve the performance of SVM and sometimes they even worsen the performance (Wickramaratna et al., 2001). The characteristic of AdaBoost causes SVM to concentrate too much on few very hard samples or outliers at the expense of the majority of the training samples, causing over-fitting problem, hence resulting in unproductive or counterproductive behaviour of AdaBoost.

By employing SVM with RBF kernel parameter (σ), i.e. RBFSVM, as the component classifier in standard AdaBoost, Li et al. (2005 and 2008) also confirmed the similar problems when applying a single σ to all of SVM component classifier. They conducted experiments by setting the value of σ, since the performance of SVM largely depends on the σ value if a roughly suitable C is given (although SVM cannot learn well when a very low value of C is used) (Valentini and Dietterich, 2004). Having too large value of σ, the classification accuracy of SVM is often less than 50% (too weak classification) and cannot meet the requirement on a component classifier given in AdaBoost. On the other hand, a smaller σ often makes the SVM component classifier stronger, causing highly correlated errors among component classifiers and moreover, too small σ can even make SVM over-fit the training data. A single best parameter σ may be found for SVM component classifiers by using model selection techniques such as k-fold or leave-one-out cross-validation. However, the process of model selection is time-consuming and should be avoided if possible. Hence, it seems that SVM

component classifiers do not perform optimally if only one single value of σ is used during AdaBoost iterations.

2.6.1 AdaBoostSVM

In order to benefit from boosting SVM, based on concept of the idea suggested by Valentini and Dietterich (2004) that AdaBoost with heterogeneous SVMs could work well, Li et al. (2008) proposed AdaBoost algorithm which is specifically designed for SVM with RBF kernel parameter by adaptively adjusting the performance of SVM component classifier during AdaBoost iterations. Therefore, it can meet the requirement on a component classifier given in AdaBoost that needs a set of moderately accurate component classifiers. Their proposed AdaBoost algorithm for SVM resulted in AdaBoostSVM (Li et al., 2008).

The performance of SVM in AdaBoostSVM can be adjusted by simply changing the value of σ during AdaBoost iterations, based on analysis of parameter influence on SVM performance. In detail, it can be described as follows:

(i) In certain range of testing data, a larger σ often leads to a reduction in classifier complexity but at the same lowers the classification performance.

(ii) A smaller σ often increases the learning complexity and leads to higher classification performance in general.

Since SVM is a strong classifier, Li et al. (2008) weakened appropriately the SVM component classifier, as suggested by Dietterich (2000). Hence, in AdaBoostSVM, a relatively large σ, which corresponds to SVM with relatively weak learning ability, is preferred. Generally, algorithm of AdaBoostSVM as mentioned in Algorithm 2.4 can be described as follows:

(i) Initially a large value is set to $\sigma(\sigma_{\max})$. Then, SVM with this σ is trained as many cycles as possible as long as more than half accuracy can be obtained.

(ii) Otherwise, this σ value is decreased slightly (σ_{step}) to increase the learning capability of SVM to help it achieve more than half accuracy. This process is continued until the σ is decreased up to the given minimal value (σ_{ini}).

Algorithm 2.4 AdaBoostSVM

Input : **x**, **y**, *C*

Initialize: The weight of training data $\mathbf{w}_t(i)$, where $\mathbf{w}_1(i)$= 1/n; i = 1,2,..*n*

The initial σ, σ_{max}; the minimal σ, σ_{min}; the step of σ, σ_{step}

Output : the class prediction, $y_{pred}(x) = sign\left(\sum_{t}^{T} \omega_t h_t(x)\right)$

Do while ($\sigma > \sigma_{min}$)

(1) Get the decision function, f_t, and then get the weighted prediction, h_t, by performing SVM on weighted training data.

(2) Calculate the weighted error of h_t: $\varepsilon_t = \sum_{i=1}^{n} w_t(i), y_i \neq h_t(\mathbf{x})$

(3) If $\varepsilon_t > 0.5$, decrease σ value by σ_{step} and go back to (1)

(4) Set the weight of h_t: $\omega_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

(5) For i = 1 to n

Update the weight of training data:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} x \begin{cases} \exp(\omega_t), & y_i \neq h_t(\mathbf{x}_i) \\ \exp(-\omega_t), & y_i = h_t(\mathbf{x}_i) \end{cases}$$

Z_t is normalization factor

End For

End For

where σ_{max} is set as the scatter radius of the training samples in the input samples (is set as about 10 to 15 times of σ_{min}

σ_{min} is set as the average minimal distance between any two training samples

σ_{step} is set to value within 1-3

C is empirically set as a value within 10 - 100

In AdaBoostSVM, σ value is decreased slightly in order to prevent the new SVM from being too strong for the current weighted training samples, and thus moderately accurate SVM component classifiers are obtained.

2.6.2 WwBoost-SVM

Wang and Li (2007) focused on designing a new weighting rule in addition to above adjusting parameter strategies, which are proposed by Li et al. (2008). Hence, their proposed algorithm is called WwBoost-SVM (new Weighting rule and Weakened SVM-based Boosting).

When decreasing the RBF kernel parameter in AdaBoostSVM process, the possibility of the over-fitting on hard data or outliers is increasing, such that boosting SVM tends to be unproductive or counterproductive. In order to avoid over-fitting problem, the new weighting rule is designed to prevent hard data or outliers from being assigned very high weight. The new weighting rule for AdaBoostSVM pays separated attention to different type of data rather than only pays much attention to erroneous data, as follows:

(i) Data correctly classified and lying far from the separating hyperplane are considered as ‘easy data’, thus their weights will be decreased more than other correctly classified data.

(ii) Data correctly classified but lying near the separating hyperplane are considered as ‘critical data’, thus their weights will be decreased less than other correctly classified data.

(iii) Data misclassified but lying near the separating hyperplane are also considered as ‘critical data’, thus their weights will be increased more than other misclassified data.

(iv) Data misclassified and lying far from the separating hyperplane are considered as ‘hard data or outliers’, thus their weights will be increased less than other misclassified data.

In SVM, the distance of a sample from the separating hyperplane is measured by $dis(\mathbf{x}_i, y_i) = |y_i f_t(\mathbf{x}_i)|$, with f_t is the decision function and y_i is the class label of input data $\mathbf{x}_i$. Data $\mathbf{x}_i$ is misclassified if $y_i f_t(\mathbf{x}_i) < 0$ and is correctly classified if $y_i f_t(\mathbf{x}_i) > 0$. In new weighting rule, Wang and Li (2007) reset weights of correctly classified by [1,H] and reset the weights of correctly classified data by [L,1], where $H = \exp(\omega_t)$ is a higher limit value and $L = \exp(-\omega_t)$ is a lower limit value. Weights of data with low value of $dis(\mathbf{x}_i, y_i)$ increase more than data with high value of $dis(\mathbf{x}_i, y_i)$, while weights of data with high value of $dis(\mathbf{x}_i, y_i)$ decrease more than data with low value of $dis(\mathbf{x}_i, y_i)$. The weight of the misclassified data farthest from the hyperplane and the weight of the correctly classified data nearest from the hyperplane are almost unchanged.

The new weighting rule which is proposed by Wang and Li (2007) can be expressed as

$$w_{t+1}(i) = w_t(i) x g(y_i f_t(\mathbf{x}_i)) \tag{2.32}$$

Where

$$g(y_i f_t(\mathbf{x}_i)) = \begin{cases} \exp(\omega_t) + \eta(1 - \exp(\omega_t))(1 - \exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) \le 0 \\ \exp(-\omega_t) + \eta(1 - \exp(-\omega_t))(\exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) > 0 \end{cases}$$

with $0 \le \eta \le 1$ (when $\eta = 0$, we get the classical weighting rule)

In $g(y_i f_t(\mathbf{x}_i))$, the first term is the classical weighting error that only pays attention to the weighted error. The second term in $g(y_i f_t(\mathbf{x}_i))$ is the new weighting rule that takes into account both the weighted error and the distance of data from the separating hyperplane. Both terms are considered equally in WwBoost-SVM algorithm, such that we use fixed value of η $(\eta = 1)$.

Basic steps of the WwBoost-SVM algorithm are described below. In WwBoost-SVM algorithm, the value of RBF kernel parameter is adaptively adjusted as proposed in AdaBoostSVM (Li et al., 2008) and the new weighting rule is used as proposed by Wang and Li (2007). At first, the optimization problem of SVM is modified, so that it can directly deal with weights distribution which is generated by WwBoost algorithm.

Algorithm 2.5 WwBoost-SVM

Input : **x**, **y**, C

Initialize: The weight of training data $\mathbf{w}_t(i)$, where $\mathbf{w}_1(i)$= 1/n; i = 1,2,..n

The initial σ, σ_{max}; the minimal σ, σ_{min}; the step of σ, σ_{step}

Output : the class prediction, $y_{pred}(x) = sign\left(\sum_{t}^{T} \omega_t h_t(x)\right)$

Do while ($\sigma > \sigma_{min}$)

For t = 1 to *max iteration for each* σ

(1) Get the decision function, f_t, and then get the weighted prediction, h_t, by performing Weighted SVM (SVM with **w**).

(2) Calculate the weighted error of h_t: $\varepsilon_t = \sum_{i=1}^{n} w_t(i), y_i \neq h_t(\mathbf{x})$

(3) If $\varepsilon_t > 0.5$, decrease σ value by σ_{step} and go back to (1)

(4) Set the weight of h_t: $\omega_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

(5) For i = 1 to n

Update the weight of training data:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} x \begin{cases} \exp(\omega_t) + (1-\exp(\omega_t))(1-\exp(y_i f_t(\mathbf{x}_i))), y_i f_t(\mathbf{x}_i) \leq 0 \\ \exp(-\omega_t) + (1-\exp(-\omega_t))(\exp(y_i f_t(\mathbf{x}_i))), y_i f_t(\mathbf{x}_i) > 0 \end{cases}$$

Z_t is normalization factor

End For

End For

where σ_{max} is set as the scatter radius of the training samples in the input samples (is set as about 10 to 15 times of σ_{min}

σ_{min} is set as the average minimal distance between any two training samples

σ_{step} is set to 1

C is empirically set as a value within 10 - 100

max iterations for each σ = 30

The algorithm of WwBoost-SVM continues until the value of RBF kernel parameter is decreased to given minimal value or reaches the ensemble size.

2.7 *k*-Fold Stratified Cross Validation

On imbalanced data sets, standard (unstratified) cross validation (CV) (Christmann et al., 2005) might sample the data such that there are folds with no minority examples. When standard CV was used on imbalanced data, there might be different distributions between training (the available data from which predictive tasks are constructed) and testing data fold (the resulting model performance and accuracy are assessed using data). This problem is referred as *sample selection bias* (Maalouf, 2009). Stratified Cross Validation (SCV) (Akbani et al., 2004; Diamantidis et al., 2000; Li et al., 2008) is a variant of Cross Validation (CV) where the class distribution in each fold is approximately the same as in the initial data set in order to avoid the *sample selection bias* problem.

The steps of *k*-Fold SCV:

(i) The data set is divided into *k* disjoint sets (folds)

(ii) Each fold is once used as the test data whereas the other *k* - 1 folds are put together to form the training data.

Partitions observations into a randomized training and a testing data fold is stratified. The stratification was conducted by using the information of class proportion in the initial dataset, such that both training and testing data folds have roughly the same class proportions as in the initial data set.

In this thesis, we set *k* = 10 for balanced data set and *k* = 5 for imbalanced data sets.

2.7.1 Evaluation Criterion

The main criterion on evaluating the performance of the classifier is the *accuracy* (Christmann et al., 2005). There are several metrics to measure the accuracy. This thesis considers two main metrics, i.e. *total accuracy* and *g-means*.

Total accuracy is the most commonly used metric to asses the accuracy of the classifier (Maalouf, 2009), while *g-means* is a type of evaluation metric which was suggested for measuring the accuracy performance of classifier on imbalanced problems (Akbani et al., 2004; Kubat and Matwin, 1997; Li et al., 2008; Weiss, 2004). In this research, *total accuracy* and *g-means* are measured based on the Confusion Matrix (CM) of binary class classification problem (Table 2.1).

Table 2.1: CM of binary class

Actual	**Predicted**	
	Positive	**Negative**
Positive	a_{11}	a_{12}
Negative	a_{21}	a_{22}

where :

a_{11} = the number of correctly classified positive instances

a_{12} = the number of incorrectly classified positive instances

a_{21}= the number of incorrectly classified negative instances

a_{22} = the number of correctly classified negative instances

Total accuracy is measured as proportion of the total number of correctly classified positive and negative instances, which can be written as,

$$total\ accuracy = \frac{a_{11} + a_{22}}{a_{11} + a_{12} + a_{21} + a_{22}} \tag{2.33}$$

G-means is measured based on sensitivity (the accuracy of positive class) and specificity (the accuracy of negative class) metrics,

$$g-means=\sqrt{sensitivity.specificity} \tag{2.34}$$

$$sensitivity=\frac{a_{11}}{a_{11}+a_{12}} \tag{2.35}$$

$$specificity=\frac{a_{22}}{a_{21}+a_{22}} \tag{2.36}$$

Total accuracy and *g-means* are measured by using *k*-Fold SCV method. The average of *total accuracy* and *g-means* values on *k* testing fold are used as the metric of accuracy on evaluating the performance of classifier with the given parameters (Christmann et al., 2005; Hsu et al., 2003 (updated 2010)). KLR has two parameters, i.e. Regularization and RBF kernel, i.e. (λ ,σ), while RLR has one parameter, i.e. regularization parameter (λ). This thesis also measures the average of *sensitivity* and *specificity* by using *k*-Fold SCV method, to give an even better idea of the performance of the classifier.

This research also measures the *standard deviation* of *total accuracy* and *g-means* respectively, as a result of the use of *k*-Fold SCV, as the second criterion on evaluating the performance of classifier. Standard deviation measures the stability of classifier on resulting in *total accuracy* and *g-means* values during *k*-Fold SCV. Therefore, the classifier has good performance if it has high *total accuracy* or *g-means* values and low *standard deviation* value.

2.7.2 Model Selection

Model selection (parameter search) (Cawley and Talbot, 2008) is the process to find the optimal value of classifier's parameter in order to achieve the optimal generalization performance of classifier. The main criterion is the *accuracy* of classifier, as explained in sub section 2.6.1. In KLR, model selection is the process to find the best combination of Regularization and RBF Kernel parameters (λ ,σ). Meanwhile, in RLR, model selection is the process to obtain optimal value of regularization parameter (λ).

Grid Search (GS) and *k*-Fold SCV method (Christmann et al., 2005; Diamantidis et al., 2000; Hsu et al., 2003 (updated 2010); Huang et al., 2007) are used on performing

the *model selection* in this thesis. GS method is used in order to determine a number of pairs of (λ ,σ).

The *model selection* on KLR is conducted as follows:

(i) Determining the grid range of (λ ,σ) and the grid step.

The grid values were used in this study i.e. $\lambda = e^{c_{1i}}$ and $\sigma = e^{c_{2i}}$.

The coefficients, i.e. c_{1i} and c_{2i}, are equidistant points (with the grid step = -0.5) which spreads over respectively with the grid range (2,-7) and (6,-3).

(ii) By using *k*-Fold SCV (as explained on sub section 2.6), estimate the average of *total accuracy* or *g-means* values for each pair of (λ ,σ).

(iii) The pair of (λ ,σ) with the best of the average of *total accuracy* or *g-means* on the *k* test fold, is determined as optimal parameters.

The *model selection* on RLR is similar to KLR, but only using regularization parameter (λ).

CHAPTER 3

PROPOSED ALGORITHMS AND RESEARCH METHODOLOGY

3.1 INTRODUCTION

Theory of proposed algorithms and its methodology are explained in this chapter. Referring to research objective of the thesis in Chapter 1, proposed algorithms consist of general classification algorithms (proposed NTR-KLR and NTR-LR algorithm) and imbalanced classification algorithms (proposed AB-WKLR and AB-WLR), while research procedures and numerical experiment design in achieving the research objective are included in research methodology.

3.2 PROPOSED NTR-KLR AND PROPOSED NTR-LR ALGORITHM

This section describes the development of proposed algorithms, i.e. proposed NTR-KLR and proposed NTR-LR. The description of KLR Newton-Raphson, RLR Newton-Raphson and Truncated Newton method are included in this section.

3.2.1 KLR Newton-Raphson and RLR Newton-Raphson with $y \in (-1,1)$

The goal of classification task is to estimate a classification rule (decision function) from n pairs of training data ($\mathbf{x}_i$, y_i), where $\mathbf{x}_i$ is input vector with dimension d (number of features) for i^{th} instance and corresponding label y_i. Considering two-class classification problem $y_i \in \{-1,1\}$, the label is either y_i =-1 or y_i=1, i=1,2.., n, for every instance $\mathbf{x}_i$. Therefore when given a new input $\mathbf{x}_i$, a class label can be assigned to it (Zhu, 2003).

(i) Classification rule of KLR and RLR

The classification rule of KLR and RLR can be estimated using (Zhu, 2003; Zhu and Hastie, 2005)

$$sign(p_1(\mathbf{x}) - 0.5) \quad \text{or} \tag{3.1}$$

$$sign\left(\log \frac{p_1(\mathbf{x})}{1-p_1(\mathbf{x})}\right) = sign(f(\mathbf{x})) \tag{3.2}$$

Considering two-class classification problem, general conditional probability of class membership for KLR and RLR (Minka, 2003;Zhu and Hastie, 2005) can be stated as,

$$p(Y = y_i \mid \mathbf{X} = \mathbf{x}_i) = \frac{1}{1+\exp(-y_i . f(\mathbf{x}))} \tag{3.3}$$

Hence, probability of instance $\mathbf{x}_i$ that belongs to class 1 becomes

$$p(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i) = 1 - p(y_i = -1 \mid \mathbf{X} = \mathbf{x}_i)$$

$$p_1(\mathbf{x}) = \frac{1}{1+\exp(-f(\mathbf{x}))} \tag{3.4}$$

Conditional probability of class membership for KLR can be written as

$$p(Y = y_i \mid \mathbf{X} = \mathbf{x}_i) = \frac{1}{1+\exp(-y_i \mathbf{K}_1 \boldsymbol{\alpha}^{(t+1)})} \tag{3.5}$$

Hence,

$$p_1(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{K}_1 \boldsymbol{\alpha}^{(t+1)})} \tag{3.6}$$

And then conditional probability for RLR,

$$p(Y = y_i \mid \mathbf{X} = \mathbf{x}_i) = \frac{1}{1 + \exp\left(-y_i . \mathbf{x}\beta^{(t+1)}\right)} \tag{3.7}$$

Hence,

$$p_1(\mathbf{x}) = \frac{1}{1 + \exp\left(-\mathbf{x}\boldsymbol{\beta}^{(t+1)}\right)} \tag{3.8}$$

Therefore, the main problem to estimate the classification rule of RLR and KLR is how to find the estimate of linear model, $f(\mathbf{x})$, which will be explained later in this sub section.

(ii) Regularized optimization function of RLR and KLR

In Logistic Regression (LR), supposing this research has random samples of n pairs of training data $(\mathbf{x}_i, y_i)$. Considering binary classification problem, $(\mathbf{x}_i, y_i)$ is assumed as identical, independent and follows Bernoulli distribution (y_i=0,1) with input $\mathbf{x}_i$. (Hosmer and Lemeshow, 2000)

Probability density function of $(\mathbf{x}_i, y_i)$ can be written as,

$$\xi(\mathbf{x}_i) = \pi(\mathbf{x}_i)^{y_i}\left[1 - \pi(\mathbf{x}_i)\right]^{1-y_i} \tag{3.9}$$

where :

$$\pi(\mathbf{x}_i) = P(y_i = 1 \mid \mathbf{x}_i) = \frac{\exp(\mathbf{X}_i\boldsymbol{\beta})}{1 + \exp(\mathbf{X}_i\boldsymbol{\beta})}$$

$$= \frac{1}{1 + \exp(-\mathbf{X}_i\boldsymbol{\beta})},$$

$$1 - \pi(\mathbf{x}_i) = P(y_i = 0 \mid \mathbf{x}_i) = \frac{1}{1 + \exp(\mathbf{X}_i\boldsymbol{\beta})}$$

The Maximum Likelihood Estimation (MLE) of $\boldsymbol{\beta}$ can be solved by minimizing the Negative Log Likelihood (NLL) function, which is the optimization function of LR.

$$-\log L(w) = -\log \prod_{i=1}^{n} (\pi(\mathbf{x}_i))^{y_i} \quad (1-\pi(\mathbf{x}_i))^{1-y_i}$$

$$= -\sum_{i=1}^{n} \{y_i \log[\pi(\mathbf{x}_i)] + (1-y_i)\log[1-\pi(\mathbf{x}_i)]\}$$

$$= \sum_{i=1}^{n} \{y_i \log(1+\exp(-\mathbf{X}_i\boldsymbol{\beta}) + (1-y_i)\log(1+\exp(\mathbf{X}_i\boldsymbol{\beta})\}$$

Instead of $y_i \in (0,1)$, by assuming $y_i \in (-1,1)$, the NLL function becomes (Zhang, 2010).

$$= \sum_{i=1}^{n}{}^{y_i=\pm 1} \log(1+\exp(-y_i.\mathbf{X}_i\boldsymbol{\beta})$$

$$L = 1^T \log(1+\exp(-y_i.f(\mathbf{x}_i))) \tag{3.10}$$

In order to avoid over-fitting problem on the training data by controlling the bias variance trade-off (Geman et al., 1992), it is necessary to give penalty (to be regularized) on the fluctuation of MLE estimates (Cawley and Talbot, 2008). This study uses the Ridge Regularization, which is added to the NLL function (Hoerl and Kennard, 1970; Minka, 2003; Zhang et al., 2003; Zhang and Oles, 2001; Zhu, 2003). It resulted in quadratically-regularized NLL for RLR and KLR respectively.

RLR

The Regularized version of LR has optimization function,

$$L(\boldsymbol{\beta}) = 1^T \log(1+\exp(-\mathbf{y}.(\mathbf{X}\boldsymbol{\beta})) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{3.11}$$

where :

y is vector of labels with dimension *n* x 1

β is vector of coefficient with dimension (*d*+1) x 1, including the bias term.

X is matrix [**x 1**]

$f(\mathbf{x}) = \mathbf{X}\boldsymbol{\beta}$

λ = regularization parameter, with bias term is not regularized

KLR

The optimization function of KLR can be obtained by kernelizing the optimization function of RLR in eq. (3.11), based on the representer theorem (Scholkopf et al. 2002, Zhu and Hastie, 2005)

$$\boldsymbol{\beta} = \varphi(\mathbf{x})\boldsymbol{\alpha} \tag{3.12}$$

where $\varphi(\mathbf{x})$ is a function to map the original data **x** in input space into feature space in order to convert the non-linear relation into linear relation.

Rather than defining the feature space explicitly, it is instead defined by a kernel function that evaluates the inner product between the images of input vectors in the feature space and must be positive semi-definite (Mercer, 1970). Hence, the decision function (logit model) of KLR, *f(x)*, can be expressed in the form,

$$\begin{aligned} f(\mathbf{x}) &= \beta^T \varphi(\mathbf{x}) \\ &= \varphi(\mathbf{x})^T \varphi(\mathbf{x})\boldsymbol{\alpha} \\ &= \mathbf{K}\boldsymbol{\alpha} \end{aligned} \tag{3.13}$$

where **K** is a Kernel matrix (Tenenhaus et al., 2007).

By substituting *f*(**x**) and **β** on equation (3.11), the regularized optimization function of KLR is obtained as,

$$L(\boldsymbol{\alpha}) = \mathbf{1}^T \log\left(1+\exp\left(-\mathbf{y}.(\mathbf{K}_1\boldsymbol{\alpha})\right)\right) + \frac{\lambda}{2}\boldsymbol{\alpha}^T\mathbf{K}_2\boldsymbol{\alpha} \tag{3.14}$$

where :

y is vector of label with dimension *n* x 1

$\boldsymbol{\alpha}$ is vector of coefficient with dimension (d+1) x 1, including the bias term.

$\mathbf{K}_1$ is matrix [**K 1**]

$f(\mathbf{x}) = \mathbf{K}_1\boldsymbol{\alpha}$

λ = regularization parameter

$\mathbf{K}_2$ is matrix $\begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}$ (the bias term is not regularized)

K is Kernel matrix of size n x n, which in this study, we considered to employ the universal Kernel (Hsu et al., 2010), i.e. Radial Basis Function (RBF) Kernel,

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

The regularized function of LR and KLR are convex optimization function (Boyd and Vandenberghe, 2004; Cawley and Talbot, 2008; Lin et al., 2008), such that there is only one solution which is global minimum. MLE estimates of RLR and KLR can be found by minimizing the regularized NLL functions. Since the MLE estimates have non-linear form, the equation can not be solved analytically, such that iterative technique, such as Newton-Raphson method, must be used. (Minka, 2003; Park et. al, 2008; Zhu and Hastie, 2005).

(iii) The relationship among SVM, KLR and RLR

Globally, the regularized function estimation problem contains two parts: a loss function and a regularization term,

$$L(\theta) = l(\mathbf{y}.f(\mathbf{x})) + \frac{\lambda}{2}\|f(\mathbf{x})\|^2 \qquad (3.15)$$

Several researchers have noted the relationship between the SVM and regularized optimization problem (Hastie et al., 2001). Fitting an SVM is equivalent to minimizing the regularized optimization problem that uses the Hinge loss function,

$$L(\theta) = (1 - \mathbf{y}.f(\mathbf{x}))_+ + \frac{\lambda}{2}\|f(\mathbf{x})\|^2 \tag{3.16}$$

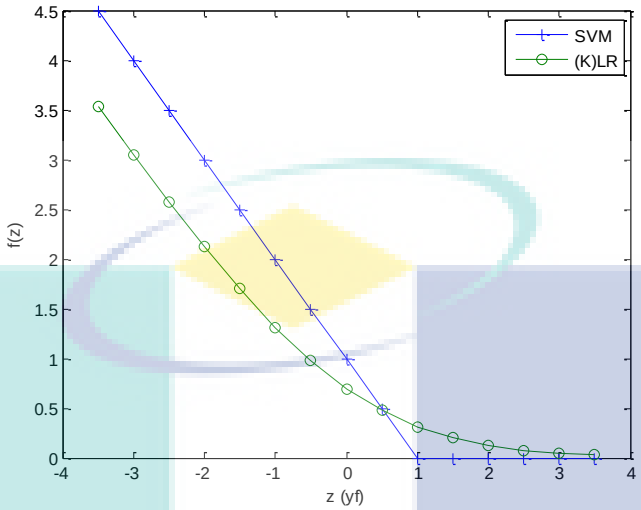


Figure 3.1: Loss functions of SVM, KLR and RLR

If the Hinge loss function is replaced by the NLL loss function, the problem becomes the regularized optimization problem of KLR (Zhu and Hastie, 2005),

$$L(\theta) = \log(1 + \exp(-\mathbf{y}.f(\mathbf{x})))_+ + \frac{\lambda}{2}\|f(\mathbf{x})\|^2 \tag{3.17}$$

Without the use of Kernel function, the regularized optimization of KLR becomes the RLR problem (Park et al., 2008). Hence, both classifiers have the same loss function.

If the Hinge loss function is plotted along with the NLL loss function, it can be seen that the NLL has similar shape to that of SVM (Fig. 3.1).The Hinge and NLL loss function are all Bayes consistent and margin-maximizing loss function. Because of the similarity between both loss functions, the fitted function of KLR and RLR performs similarly to the SVM (Zhang et al., 2003; Zhang and Oles, 2001; Zhu and Hastie, 2005;). However, as aforementioned, SVM needs to be solved with constrained regularized optimization problem, while KLR and RLR only need to be solved with unconstrained regularized optimization problem (Maalouf, 2009).

(iv) Newton-Raphson method for KLR and RLR

The Newton-Raphson method is an algorithm that iteratively solves the linear system of Newton-Raphson update rule (NRUR) in order to estimate the Newton direction (Lin et al., 2008), as can be seen in the Newton-Raphson formula,

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} + \mathbf{s}^{(t)} \\ &= \boldsymbol{\theta}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)}\end{aligned} \tag{3.18}$$

where t is the iteration index. $\mathbf{s}^{(t)}$ is the Newton direction, which is the solution of the linear system of Newton-Raphson update rule,

$$\mathbf{H}^{(t)}\mathbf{s}^{(t)} = -\mathbf{g}^{(t)} \tag{3.19}$$

The gradient (**g**) and the Hessian (**H**) are achieved respectively by differentiating the regularized NLL function with respect to θ.

RLR with Newton-Raphson method (RLR-NR)

The iterative method of Newton-Raphson for RLR has the form

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \mathbf{s}^{(t)}$$

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)}$$

$$= \boldsymbol{\beta}^{(t)} - \left(\frac{\partial^2 L(\boldsymbol{\beta})}{\partial(\boldsymbol{\beta})\partial(\boldsymbol{\beta}^T)}\right)^{-1}\left(\frac{\partial L(\boldsymbol{\beta})}{\partial(\boldsymbol{\beta})}\right) \quad \backslash$$

For RLR, the linear system of Newton-Raphson update rule can be expressed as

$$\left(\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{s}^{(t)} = \left(-\mathbf{X}^T(y.\mathbf{p}) + \lambda\boldsymbol{\beta}\right) \tag{3.20}$$

KLR with Newton-Raphson method (KLR-NR)

The iterative method of Newton-Raphson for KLR can be written as

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \mathbf{s}^{(t)}$$

$$\begin{aligned}\boldsymbol{\alpha}^{(t+1)} &= \boldsymbol{\alpha}^{(t)} - (\mathbf{H}^{(t)})^{-1}\mathbf{g}^{(t)} \\ &= \boldsymbol{\alpha}^{(t)} - \left(\frac{\partial^2 L(\boldsymbol{\alpha})}{\partial(\boldsymbol{\alpha})\partial(\boldsymbol{\alpha}^T)}\right)^{-1}\left(\frac{\partial L(\boldsymbol{\alpha})}{\partial(\boldsymbol{\alpha})}\right)\end{aligned} \tag{3.21}$$

The linear system of Newton-Raphson update rule of KLR, becomes

$$\left(\mathbf{K}_1{}^T\mathbf{V}^{(t)}\mathbf{K}_1 + \lambda\mathbf{K}_2\right)\mathbf{s}^{(t)} = \left(-\mathbf{K}_1{}^T(\mathbf{y}.\mathbf{p}) + \lambda\mathbf{K}_2\boldsymbol{\alpha}\right) \tag{3.22}$$

where $\mathbf{V}$ = diag($\mathbf{p}$.(1-$\mathbf{p}$)) with size n x n

As mentioned in Chapter 1, there is a numerical problem to find the approximate Newton direction on using the linear systems of Newton-Raphson update rule. The numerical problem is the huge matrix to be inverted, which is a linear system of n equations and n variables. (Lin et al., 2008; Zhu and Hastie, 2004; Zhu and Hastie, 2005;)

3.2.2 KLR-NR and RLR-NR with Truncated Newton method

Truncated Newton methods are a family of suitable methods for solving large scale data of non-linear optimization problem (Nash, 2000). A solid convergence theory has been derived for the methods. A Truncated Newton method consists of doubly iterative method: an outer iteration for the non-linear optimization problem (such as MLE in this research) and an inner iteration for the Newton equation. Before the solution to the Newton equation is obtained, the inner iteration is typically stopped or "truncated". At each iteration of Truncated Newton method, the current estimate of the solution is updated, i.e. a step is computed, by approximately solving the Newton

equation of an iterative algorithm. For large scale data of non-linear optimization problem, the result of Truncated Newton methods typically has a collection of powerful, flexible and adaptable tools.

This research proposes the use of Truncated Newton for KLR and RLR respectively, by keeping the use of Linear CG as the truncated inner algorithm, and the origin Newton Raphson method as the outer algorithm. Linear CG finds the approximate Newton direction by solving the numerical problem on using the linear system of Newton-Raphson update rule.

The numerical problem of Newton-Raphson method for RLR and KLR can be solved by the use of Linear CG method to quadratic form of Newton-Raphson update rule (Lin et al., 2008),

$$q^{(t)}(s) = \mathbf{g}^{T(t)}\mathbf{s}^{(t)} + \frac{1}{2}\mathbf{s}^{T(t)}\mathbf{H}^{(t)}\mathbf{s} \tag{3.23}$$

The use of Linear CG that simply requires matrix-vector products simplifies the computation of the linear system, such that time required in each iteration of KLR-NR and RLR-NR algorithms respectively to be fast (Komarek and Moore, 2005). In order to avoid the long computation that Linear CG may suffer from, the number of CG iterations is limited on approximating the Newton direction (Maalouf, 2009). Truncated Newton method accommodates the need for a "trade off "between convergence speed and accurate Newton direction.

Similar to TR-KLR (Maalouf et al., 2010) and TR-IRLS (Komarek and Moore, 2005) algorithm, NTR-KLR and NTR-LR algorithm respectively consist of two loops, i.e. outer and inner loops. Main algorithm represents the outer loop which is iterations to find the Newton direction by using the iterative method of Newton-Raphson. When the relative difference of optimization function is no larger than a specified threshold, ε_1, then the iteration is terminated. Second algorithm represents the inner loop which is iterations to find the approximation of the Newton direction by using the Linear CG

method. When the square residual is no greater than a specified threshold, ε_2, then the iteration is terminated.

The choices of parameter values (ε and setting of number of iteration) for both proposed algorithms mostly are considered appropriate following the previous research on Truncated Newton methods for KLR and RLR (Komarek and Moore, 2005; Malouf et al., 2010). The specified threshold values are considered to sufficiently reach good accuracy and convergence speed at the same time. The parameters will be given in detail in the proposed Algorithms 1 and 2 below.

KLR-NR with Truncated Newton method (Proposed NTR-KLR)

The numerical problem of Newton-Raphson method for KLR can be solved by the use of Linear CG method, as the Truncated Newton to quadratic form of Newton-Raphson update rule,

$$q^{(t)}(s) = \left(-\mathbf{K}_1{}^T(\mathbf{y}.\mathbf{p}) + \lambda \mathbf{K}_2 \boldsymbol{\alpha}\right)^{T(t)} \mathbf{s}^{(t)} + \frac{1}{2}\mathbf{s}^{T(t)}\left(\mathbf{K}_1 \mathbf{V}^{(t)} \mathbf{K}_1 + \lambda \mathbf{K}_2\right)^{(t)} \mathbf{s}^{(t)} \qquad (3.24)$$

Similar to TR-KLR, the maximum number of iterations for Algorithm 1 (KLR-NR) is set to 30 and the threshold of the difference of optimization function value is set to 2.5 (ε_1). For Algorithm 2 (Linear CG for KLR-NR), the convergence threshold is set to 0.005 (ε_2). Unlike TR-KLR, the maximum number of iterations for Linear CG in NTR-KLR is set to 1000 iterations for accommodating the complexity of data used.

Algorithm 3.1 The proposed NTR-KLR

Algorithm 1. KLR-NR (Outer loop)
Input: **X**, $\mathbf{K_1}$, **y**, λ, σ
Initialize:$\boldsymbol{\alpha}^{(1)}$, $L^{(1)}$,
Output : $\boldsymbol{\alpha}^{(\tau+1)}$

Do While $\left|\frac{L^{(t+1)} - L^{(t)}}{L^{(t+1)}}\right| > \varepsilon_1$

For t = 1 to *max KLR-NR iterations*

(1) Compute probability: $\mathbf{p}^{(t)} = 1./(1+\exp(\mathbf{y}.\mathbf{K}\boldsymbol{\alpha}^{(t)}))$

(2) Compute variance: $\mathbf{V}^{(t)} = \text{diag}(\mathbf{p}^{(t)}.(1-\mathbf{p}^{(t)}))$

(3) Compute $\mathbf{g}^{(t)}$ dan $\mathbf{H}^{(t)}$ of KLR

(4) Compute NR update rule solution: $\mathbf{H}^{(t)}\,\mathbf{s}^{(t)} = -\mathbf{g}^{(t)}$

(5) Compute $\boldsymbol{\alpha}^{(\tau+1)}$ by NR: $\boldsymbol{\alpha}^{(\tau+1)} = \boldsymbol{\alpha}^{(\tau)} + \mathbf{s}^{(t)}$

(6) Compute $L^{(t+1)}$

End For

where ε_1=2.5, *max KLR- NR iterations* = 30

Algorithm 2. Linear CG (Inner loop)

Input : $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$ **of KLR-NR**

Initialize: $\mathbf{s}^{(1)}$, $\mathbf{r}^{(1)}$=-**g**, $\mathbf{d}^{(1)}$= $\mathbf{r}^{(1)}$

Output : $\mathbf{s}^{(\tau+1)}$

Do While $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length: $a^{(\tau)} = \mathbf{r}^{T(t)}\,\mathbf{r}^{(t)} / (\mathbf{d}^{T(t)}\,\mathbf{H}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution: $\mathbf{s}^{(\tau+1)} = \mathbf{s}^{(\tau)} + \mathbf{a}^{(\tau)}\,\mathbf{d}^{(t)}$

(3) Update the residual: $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\,\mathbf{H}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer: $\mathbf{c}^{(t)} = \mathbf{r}^{T(t+1)}\,\mathbf{r}^{(t+1)} / \mathbf{r}^{T(t)}\mathbf{r}^{(t)}$

(5) Update the search direction: $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\,\mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 1000

RLR-NR with Truncated Newton method (Proposed NTR-LR)

The numerical problem of Newton-Raphson method for RLR can be solved by the use of Linear CG method to quadratic form of Newton-Raphson update rule,

$$q^{(t)}(s) = \left(-\mathbf{X}^T(\mathbf{y}.\mathbf{p}) + \boldsymbol{\lambda\beta}\right)^{T(t)}\mathbf{s}^{(t)} + \frac{1}{2}\mathbf{s}^{T(t)}\left(\mathbf{X}^T\mathbf{V}^{(t)}\mathbf{X} + \lambda\mathbf{I}\right)^{(t)}\mathbf{s}^{(t)} \quad (3.25)$$

All settings of NTR-LR are similar to TR-IRLS algorithm, except setting for the regularization parameter. TR-IRLS sets a fixed value of regularization parameter, while NTR-LR sets an optimal value of regularization parameter with Grid Search method and 5-Fold SCV.

Algorithm 3.2 The Proposed NTR-LR

Algorithm 1. RLR-NR (Outer loop)

Input : **X**, **y**, λ

Initialize : $\boldsymbol{\beta}^{(1)}$, $L^{(1)}$

Output : $\boldsymbol{\beta}^{(\tau+1)}$

Do While $\left|\frac{L^{(t+1)} - L^{(t)}}{L^{(t+1)}}\right| > \varepsilon_1$

For t = 1 to *max RLR-NR iterations*

(1) $\mathbf{p}^{(t)} = 1./(1+\exp(\mathbf{y}.\mathbf{x}\boldsymbol{\beta}^{(t)}))$

(2) Compute variance : $\mathbf{V}^{(t)} = \text{diag}(\mathbf{p}^{(t)}.(1-\mathbf{p}^{(t)}))$

(3) Compute $\mathbf{g}^{(t)}$ dan $\mathbf{H}^{(t)}$ of RLR

(4) Compute NR update rule solution : $\mathbf{H}^{(t)}\mathbf{s}^{(t)} = -\mathbf{g}^{(t)}$

(5) Compute $\boldsymbol{\beta}^{(\tau+1)}$ by NR : $\boldsymbol{\beta}^{(\tau+1)} = \boldsymbol{\beta}^{(\tau)} + \mathbf{s}^{(t)}$

(6) Compute $L^{(t+1)}$

End For

where ε_1=0.01, *max RLR- NR iterations* = 30

Algorithm 2. Linear CG (Inner loop)

Input : $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$ of RLR-NR

Initialize: $\mathbf{s}^{(1)}$, $\mathbf{r}^{(1)}$=-**g**, $\mathbf{d}^{(1)}$= $\mathbf{r}^{(1)}$

Output : $\mathbf{s}^{(\tau+1)}$

Do while $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length : $a^{(\tau)} = \mathbf{r}^{T(t)}\,\mathbf{r}^{(t)} / (\mathbf{d}^{T(t)}\,\mathbf{H}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution : $\mathbf{s}^{(\tau+1)} = \mathbf{s}^{(\tau)} + \mathbf{a}^{(\tau)}\,\mathbf{d}^{(t)}$

(3) Update the residual : $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\,\mathbf{H}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer : $\mathbf{c}^{(t)} = \mathbf{r}^{T(t+1)}\,\mathbf{r}^{(t+1)} / \mathbf{r}^{T(t)}\mathbf{r}^{(t)}$

(5) Update the search direction : $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\,\mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 200

3.3 PROPOSED AB-WKLR AND PROPOSED AB-WLR ALGORITM

The development of imbalanced classification algorithms for NTR-KLR and NTR-LR, i.e. proposed AB-WKLR and proposed AB-WLR respectively, is explained in this section. In general, both proposed algorithms consist of Weighted version of general classification algorithms (NTR Weighted KLR and NTR Weighted RLR) and adapted Modified AdaBoost methods. In relation to the development of imbalanced classification algorithms, study on the imbalanced problem of general classification algorithms and the proper use of evaluation metric is reported previously in sub section 3.3.1.

3.3.1 Study on the imbalanced problem and the proper use of evaluation metric

Table 3.1 summarizes the numerical results of NTR-KLR and NTR-LR by maximizing *total accuracy* value, in order to study the proper use of *g -means* metric as evaluation metric on imbalanced classification problem as compared to *total accuracy*. *Total accuracy* is the most common metric to evaluate the accuracy of classifier. As displayed in Table 3.1 and Fig. 3.2, *total accuracy* metric evaluates the performance of NTR-KLR classifier properly on Australia data set, which is balanced data. It can be seen that balanced data have almost equal value of *sensitivity* value and *specificity*. *Total accuracy* metric places the same weight, on the majority and minority classes, as observed in Fig. 3.2. This result has confirmed the proper use of *total accuracy* metric on balanced data, because it is theoretically developed under balanced data assumption. However, *total accuracy* metric has value towards to the majority class at the expense of the minority class on other data sets, which are imbalanced data, as shown in Table

3.1 and Fig. 3.2. It can be seen more clearly in Balance2 (5) and LetterImg26 (9) data sets.

Table 3.1: Summary of NTR-KLR and NTR-LR
by maximizing *total accuracy* value with 5-Fold SCV

No.	Name of Data set	Optimal Parameter		Class Accuracy		Evaluation Metrics	
		λ_{opt}	σ_{opt}	Minority (+) (*Sensitivity*)	Majority (-) (*Specificity*)	*Total Accuracy*	*g-means*
NTR-KLR							
1	Australia	**Exp(0)**	**Exp(2)**	0.8763	0.8748	**0.8754**	0.8755
2	Parkinson	**Exp(-6.5)**	**Exp(1)**	0.8800	0.9965	**0.9436**	0.9195
3	Glass7	**Exp(-5)**	**Exp(1)**	0.8667	0.9946	**0.9766**	0.9257
4	ImgSegment1	**Exp(-3)**	**Exp(0)**	0.9879	0.9995	**0.9978**	0.9937
5	Balance2	**Exp(-6)**	**Exp(1)**	**0.02**	**1**	**0.9232**	0.0632
6	Car3	**Exp(-5)**	**Exp(0)**	0.8275	0.9976	**0.9907**	0.9073
NTR-LR							
7	GammaImg	**Exp(0.5)**	-	0.5906	0.9000	**0.7912**	0.7291
8	Shuttle2to7	**Exp(-3)**	-	0.9186	0.9862	**0.9717**	0.9518
9	LetterImg26	**Exp(-3)**	-	**0.6757**	**0.9951**	**0.9833**	0.8199

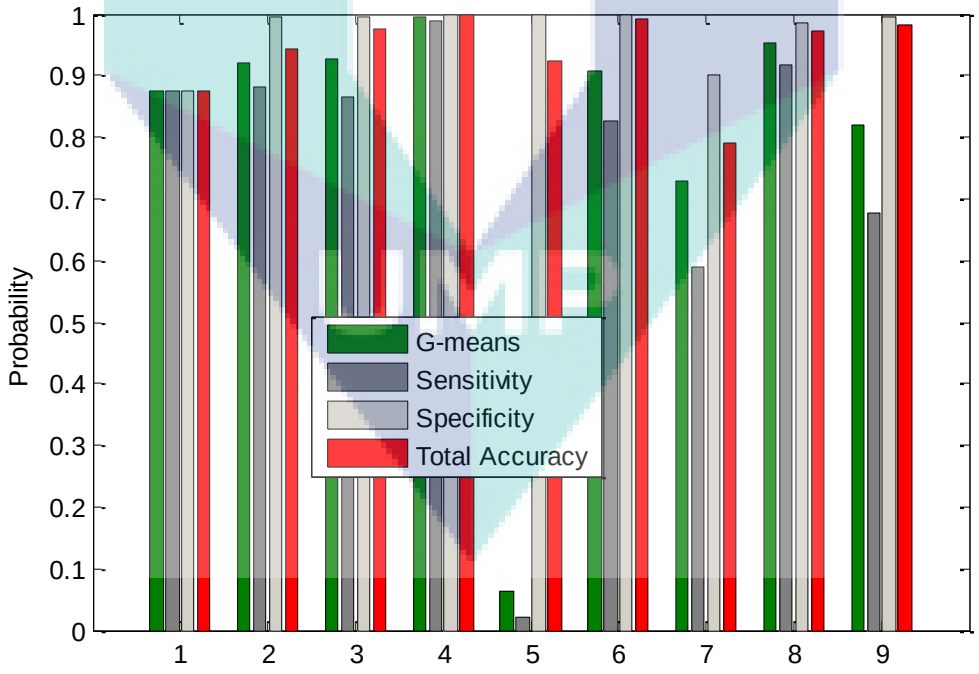


Figure 3.2: Comparison between *g-means* and *total accuracy* metrics on imbalanced problem

Table 3.1 and Fig. 3.2 show that *g-means* metric evaluates the accuracy of NTR-KLR such as *total accuracy* metric, on balanced data. Moreover, *g-means* metric evaluates the accuracy of NTR-KLR and NTR-KLR properly on eight data sets of imbalanced problem. It is displayed that the *g-means* metric accommodates minority and majority classes equally, as it is measured at the geometric mean value between *sensitivity* and *specificity* value. These results have confirmed the proper use of *g-means* metric that was suggested by Kubat and Matwin (1997) and has been used also by several researchers for evaluating the accuracy of classifier on imbalanced problem. As this thesis focuses on imbalanced problem research, it is considered to use *g-means* as the main evaluation metric for classifier performance on imbalanced problem, by maximizing the *g-means* value with 5-Fold SCV. The summary of results can be observed in Table 3.2.

NTR-KLR and NTR-LR by maximizing *g-means* value with 5-Fold SCV (Table 3.2), have the same parameters and accuracy result, such as by maximizing total accuracy value with 5-Fold SCV (Table 3.1), with exception to GammaImg data set. The similarity of parameter and accuracy results in explaining the influence of parameters to the classification performance of NTR-KLR and NTR-LR either by maximizing *total accuracy* or *g-means* metric will be studied and discussed later in the sub section 3.3.3.

It is indicated in Table 3.2 and Figure 3.3, that NTR-KLR and NTR-LR tend to classify everything as negative class which is majority class, such that they have almost perfect *specificity* values on classifying imbalanced problems. However, it seems not easy for both algorithms, which are general classifiers to identify the pattern of minority class. Therefore, they have poor *sensitivity* values. These results are similar to that achieved by Akbani et al. (2004) but under different optimization function.

Table 3.2: Summary of NTR-KLR and NTR-LR
by maximizing *g-means* value with 5-Fold SCV

No.	Name of Data set	Optimal Parameter		Class Accuracy		*g-means* ($S_{g\text{-}means}$)
		λ_{opt}	σ_{opt}	Minority (+) (*Sensitivity*)	Majority (-) (*Specificity*)	
	NTR-KLR					
1	Parkinson	**Exp(-6.5)**	**Exp(1)**	0.8800	0.9655	**0.9195 (0.0716)**
2	Glass7	**Exp(-5)**	**Exp(1)**	0.8667	0.9946	**0.9257 (0.0732)**
3	ImgSegment1	**Exp(-3)**	**Exp(0)**	0.9879	0.9995	**0.9937 (0.0065)**
4	Balance2	**Exp(-6)**	**Exp(1)**	0.0200	1	**0.0632 (0.1414)**
5	Car3	**Exp(-5)**	**Exp(0)**	0.8429	0.9982	**0.9073 (0.0509)**
	NTR-LR					
6	GammaImg	**Exp(-0.5)**	-	0.5908	0.8999	**0.7291 (0.0071)**
7	Shuttle2to7	**Exp(-3)**	-	0.9186	0.9862	**0.9518 (0.0048)**
8	LetterImg26	**Exp(-3)**	-	0.6757	0.9951	**0.8199 (0.0115)**

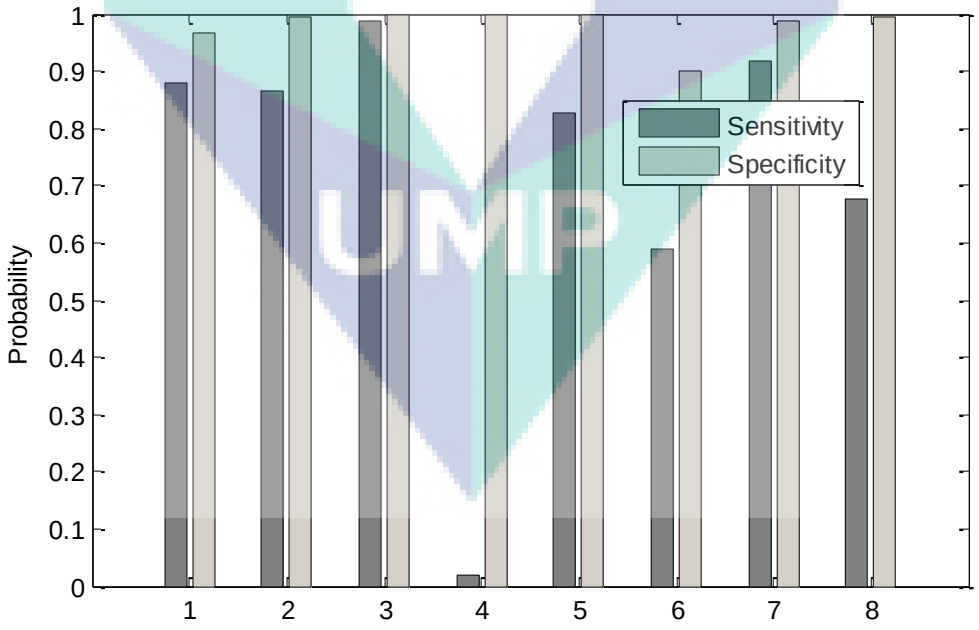


Figure 3.3: Performance of *sensitivity* and *specificity* on imbalanced problem

It can be observed in this study that the minority class is the important class that needs to be focused by NTR-KLR and NTR-LR on imbalanced problem research, since it presents poor *sensitivity* value. This thesis proposes a modification of NTR-KLR and NTR-LR algorithm respectively which are designed specifically for imbalanced classification problem. The proposed algorithms unavoidably sacrifice the *specificity* value in improving the *sensitivity* value. Hence, unlike RE-WKLR classifier (Maalouf et al., 2011), this research considers to use *g-means* as the proper metric to evaluate the accuracy performance of proposed classifiers on imbalanced problem in order to accommodate *sensitivity* and *specificity* value equally. The development of imbalanced NTR-KLR and NTR-LR algorithm respectively will be started to explain in sub section 3.3.2.

3.3.2 NTR Weighted KLR and NTR Weighted RLR

In order to improve the performance of Boosting NTR-KLR and NTR-LR using adapted Modified AdaBoost methods respectively, it is required to modify the Negative Log-Likelihood (NLL) function on the regularized optimization function of KLR and RLR into Weighted NLL, such that they can directly deal with weight distributions, which are generated by modified AdaBoost algorithm for solving the imbalanced problem. King and Zheng (2001) and Maalouf et al. (2011) proposed the use of weighted loss function for solving the imbalanced classification problem in RLR and TR-KLR under different scheme.

This research proposes the use of Weighted NLL loss function on the regularized optimization function of KLR and RLR, hence KLR becomes a Weighted KLR (WKLR) and RLR becomes a Weighted RLR (WLR). The use of Weighted NLL on the regularized optimization function of WKLR and RLR, for instance NLL loss function, resulted in the regularized weighted optimization function of WKLR and WLR.

The regularized weighted optimization function of WKLR and WLR

The regularized weighted optimization function of WKLR can be written as,

$$L(\boldsymbol{\alpha}) = \mathbf{w}^T \log(1 + \exp(-\mathbf{y}.(\mathbf{K}_1\boldsymbol{\alpha}))) + \frac{\lambda}{2}\boldsymbol{\alpha}^T\mathbf{K}_2\boldsymbol{\alpha} \quad (3.26)$$

On WLR, the regularized weighted optimization function can be stated as,

$$L(\boldsymbol{\beta}) = \mathbf{w}^T \log(1 + \exp(-\mathbf{y}.(\mathbf{X}\boldsymbol{\beta}))) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} \quad (3.27)$$

where **w** = weight distribution with dimension *n* x 1

The gradient and the Hessian of WKLR and WLR are derived by differentiating the regularized weighted NLL function with respect to α and β respectively. Similar to NTR-KLR and NTR-LR, WKLR and WLR use Truncated Newton method in order to approximate the Newton direction of WKLR and WLR in using the Newton-Raphson method.

WKLR and WLR with Truncated Newton method

For WKLR, the linear system of Newton-Raphson update rule can be stated as

$$\left(\mathbf{K}_1\mathbf{D}^{(t)}\mathbf{K}_1 + \lambda\mathbf{K}_2\right)\mathbf{s}^{(t)} = \left(-\mathbf{K}_1{}^T\mathbf{W}(\mathbf{y}.\mathbf{p}) + \lambda\mathbf{K}_2\boldsymbol{\alpha}\right) \quad (3.28)$$

while the linear system of Newton-Raphson update rule for WLR becomes

$$\left(\mathbf{X}^T\mathbf{D}^{(t)}\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{s}^{(t)} = \left(-\mathbf{X}^T\mathbf{W}(\mathbf{y}.\mathbf{p}) + \lambda\boldsymbol{\beta}\right) \quad (3.29)$$

where **D** = diag(**v**.**w**); **v** = **p**.(1-**p**)

W =diag(**w**)

Solving the linear system of Newton-Raphson update rule of WKLR through the approach of Linear CG, as the truncated inner method, is equivalent to minimizing the quadratic form,

$$q^{(t)}(s) = \left(-\mathbf{K}_1{}^T\mathbf{W}(\mathbf{y}.\mathbf{p}) + \lambda\mathbf{K}_2\boldsymbol{\alpha}\right)^{T(t)}\mathbf{s}^{(t)} + \frac{1}{2}\mathbf{s}^{T(t)}\left(\mathbf{K}_1\mathbf{D}^{(t)}\mathbf{K}_1 + \lambda\mathbf{K}_2\right)^{(t)}\mathbf{s}^{(t)} \quad (3.30)$$

Meanwhile, solving the linear system of Newton Raphson Update Rule of WLR by the approach of Linear CG, as the truncated inner method for WLR, is equivalent to minimizing the quadratic form,

$$q^{(t)}(s) = \left(-\mathbf{X}^T\mathbf{W}(\mathbf{y}.\mathbf{p}) + \lambda\boldsymbol{\beta}\right)^{T(t)}\mathbf{s}^{(t)} + \frac{1}{2}\mathbf{s}^{T(t)}\left(\mathbf{X}^T\mathbf{D}^{(t)}\mathbf{X} + \lambda\mathbf{I}\right)^{(t)}\mathbf{s}^{(t)} \quad (3.31)$$

The use of Truncated Newton method for WKLR resulted in NTR-WKLR, while the use of Truncated Newton for WLR resulted in NTR-WLR. Because of the similarity to the NTR-KLR and NTR-LR algorithm, detail algorithm of WKLR and WLR will be displayed next in the sub section 3.3.3.

3.3.3 NTR-WKLR and NTR-WLR with adapted Modified AdaBoost method

This sub section contains the description of NTR-WKLR, NTR-WLR and The Adaptations of Modified AdaBoost method. The main adaptations of Modified AdaBoost method for NTR-KLR and NTR-LR classifier respectively were conducted by determining the strategies of parameter adjusting on applying the Modified AdaBoost, based on the parameters influence of NTR-KLR and NTR-LR classifiers respectively. The setting of parameters is adjusted appropriately during the process of adapted Modified AdaBoost methods iterations. In relation to this, study on the influence of parameters to the classification performance of NTR-KLR and NTR-LR respectively is reported previously.

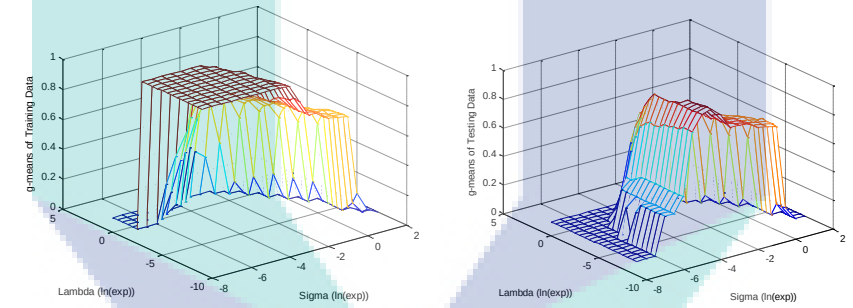
(i) Study on the influence of parameters to classification performance

This sub section describes the study on influence of parameters to classification performance of NTR-KLR and NTR-RLR which can be seen as representation of KLR and RLR classifier, but under Truncated Newton method. This study is important for

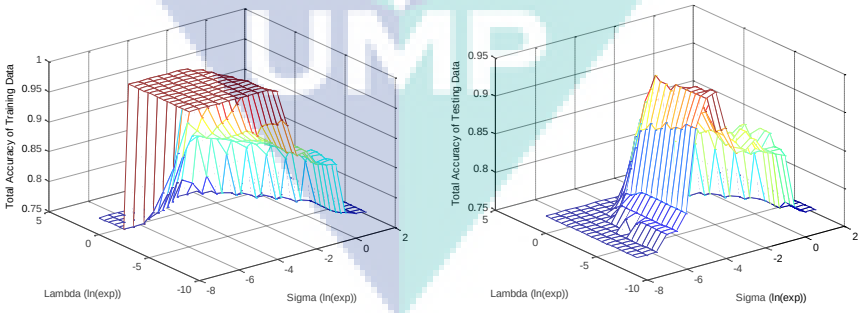
determining the strategy of parameter adjusting during the process of adapted Modified AdaBoost. The classification performance of NTR-KLR and NTR-LR are measured by using *g-means* and *total accuracy* metric.

The influence of parameters (λ,σ) to classification performance of NTR-KLR

This research considers using NTR-KLR with RBF kernel. Parameters of NTR-KLR include the regularization parameter (λ) and the RBF kernel parameter, Gaussian width (σ). Fig. 3.4 –3.8 shows that the variation of σ leads to larger variation of classification performance than variation of λ. This means that the kernel parameter, σ, is regarded more important than the regularization parameter, λ, on changing the classification performance of NTR-KLR, by *total accuracy* and *g-means* metric.

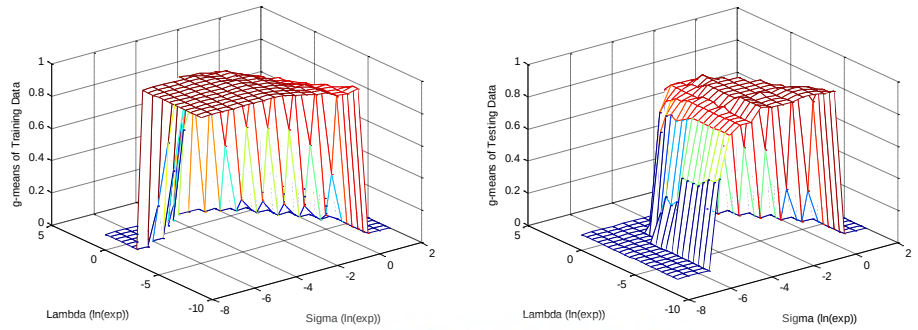


(a) *g-means*

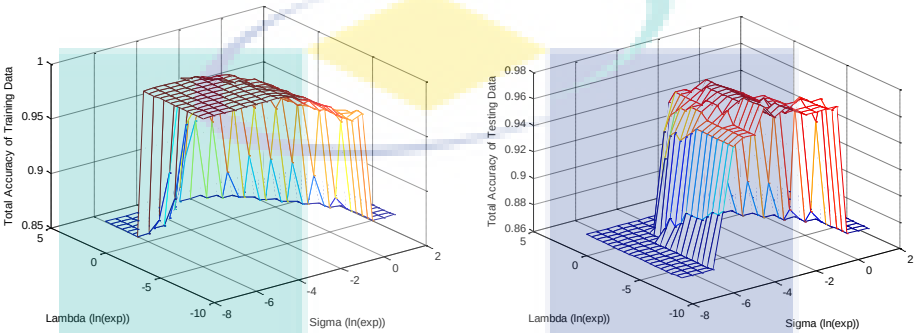


(b) *total accuracy*

Figure 3.4: The influence of parameter using *Parkinson* data set

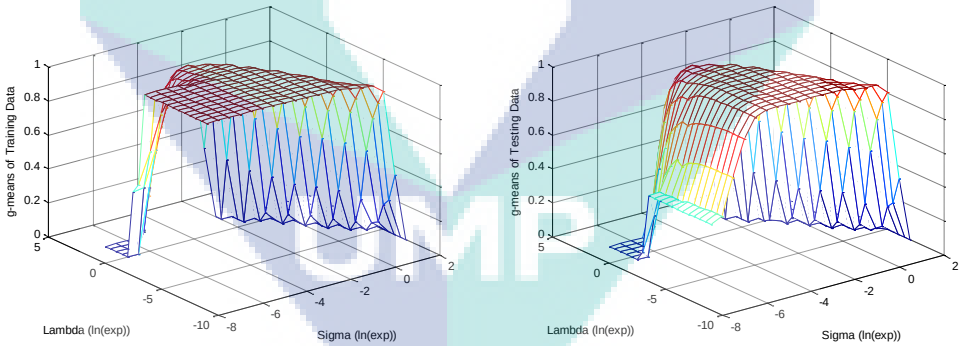


(a) *g-means*

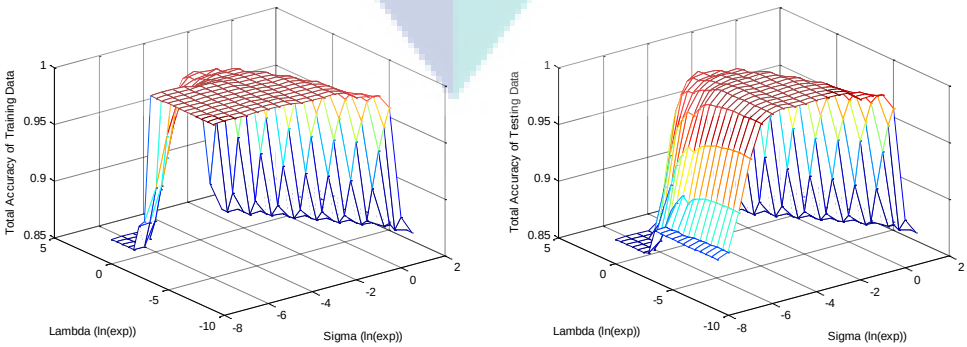


(b) *total accuracy*

Figure 3.5: The influence of parameter using *Glass7* data set

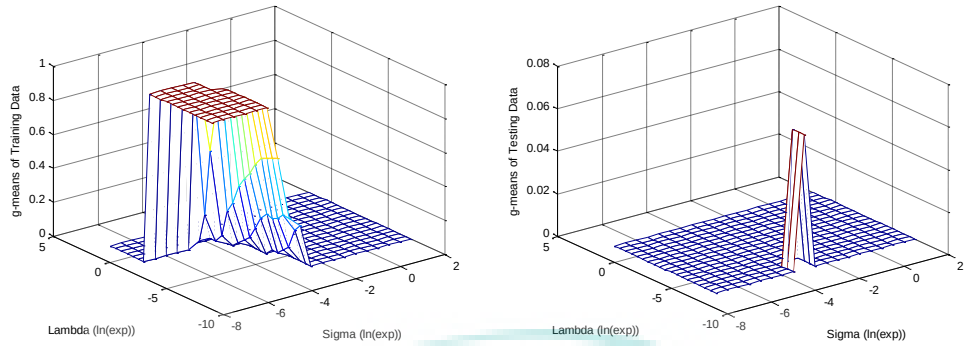


(a) *g-means*

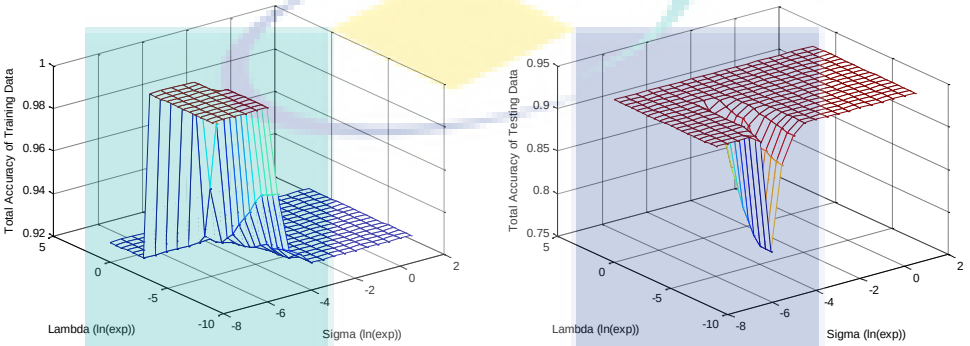


(b) *total accuracy*

Figure 3.6: The influence of parameter using ImgSegment1 data set

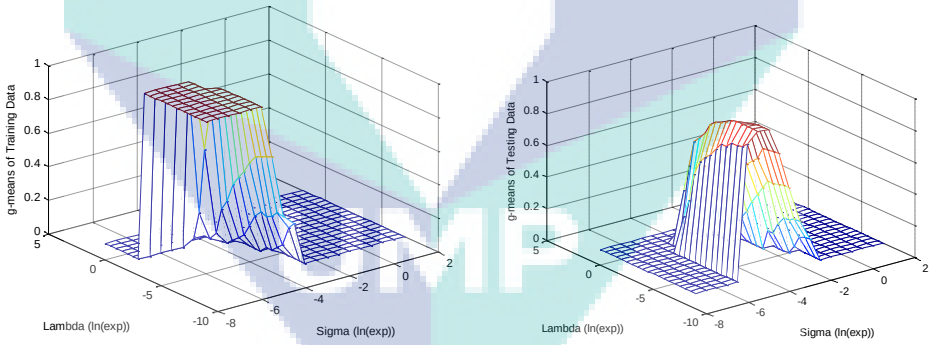


(a) *g-means*

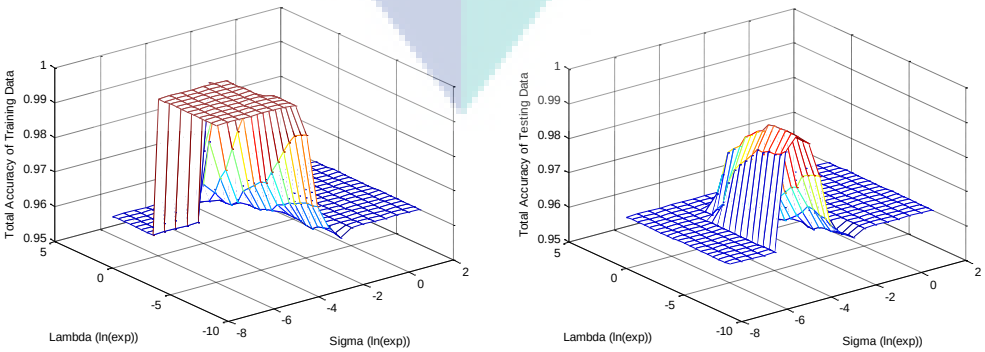


(b) *total accuracy*

Figure 3.7: The influence of parameter using *Balance2* data set



(a) *g-means*



(b) *total accuracy*

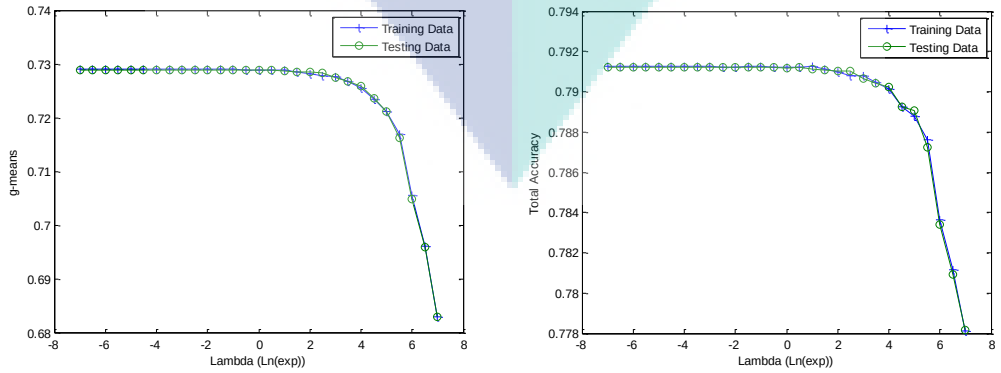
Figure 3.8: The influence of parameter using *Car3* data set

It can be seen also in Fig. 3.4 – 3.8 that the classification performance on training data (left side) decreases with higher σ when using a roughly suitable λ. Meanwhile the classification performance on testing data (right side) decreases after reaching a high value region. Unlike on training data, the classification performance on testing data increases until reaching a high value region. This fact supports the conjecture of over fitting with small value of σ. The value of λ has similar behaviour to σ, that the classification performance of NTR-KLR decreases with higher λ. Zhu and Hastie (2005) has realized this for Import Vector Machine (IVM) scheme which is representation of KLR with import vector. However, the variation of λ leads to smaller variation of classification performance than variation of σ. Hence, λ is regarded less important than σ on changing the classification performance of NTR-KLR.

Generally, it is demonstrated that *g-means* values has similar behaviour to *total accuracy* values in explaining the influence of parameters to classification performance of NTR-KLR, with exception to testing data of Balance2 data set.

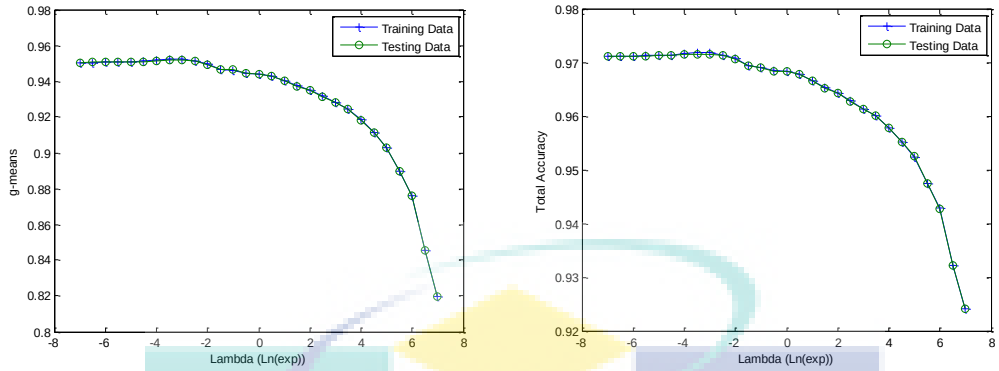
The influence of parameter (λ) to classification performance of NTR-LR

It is shown in Fig. 3.9 – 3.11 that the classification performance of NTR-LR decreases with higher regularization parameter (λ).



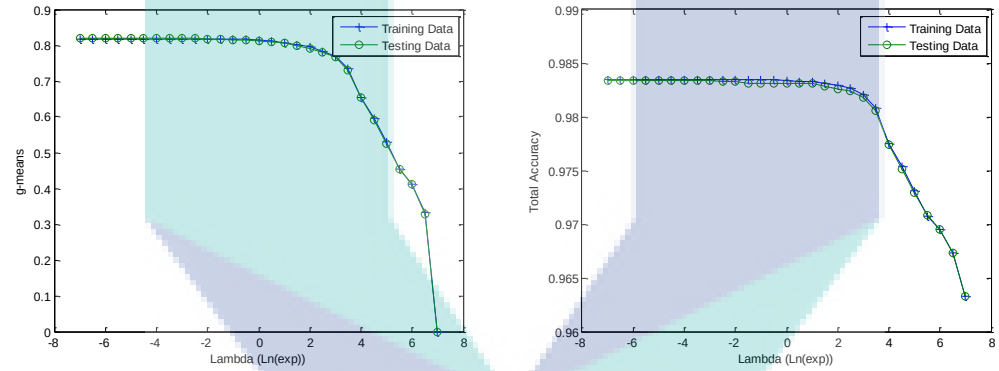
(a) *g-means* (b) *total accuracy*

Figure 3.9: The influence of parameter using *GammaImg* data set



(a) *g-means* (b) *total accuracy*

Figure 3.10: The influence of parameter using *Shuttle2to7* data set



(a) *g-means* (b) *total accuracy*

Figure 3.11: The influence of parameter using *LetterImg26* data set

The classification performance of NTR-LR changes slightly along low value of λ. Higher value of λ then leads to larger variation of classification performance. As on NTR-KLR, it is displayed that *g-means* values have similar behaviour to the *total accuracy* values, as shown on Fig. 3.9 – 3.11.

The analysis of parameters influence to classification performance of NTR-KLR and NTR-LR by *total accuracy* metric is used to employ the proper tuning of parameters on applying the algorithms of adapted Modified Adaboost for NTR-KLR and NTR-LR. Meanwhile, the similarity influence in this study has explained the

similarity of parameter and accuracy results by maximizing *g-means* and *total accuracy* metric in determining the optimal performance of NTR-KLR and NTR-LR classifier, as has been shown in sub section 3.3.1. In addition, *g-means* value provides geometric mean of *sensitivity* and *specificity* value such that it can evaluate properly on imbalanced classification problem rather than *total accuracy*, as has been confirmed in sub section 3.3.1.

(ii) Adaptations of Modified AdaBoost method

The adaptations of Modified AdaBoost algorithm for NTR-KLR and NTR-LR which are highly accurate (strong) classifiers respectively were conducted in order to benefit from Boosting mechanism. The adaptations can be explained briefly as follows:

(i) The adaptations adjust the parameter values of NTR-KLR and NTR-LR respectively, during the process of AdaBoost iterations to achieve a set of moderately accurate classifiers in order to benefit from Boosting NTR-KLR and NTR-LR respectively. These adaptations are similar to that strategy used in AdaBoostSVM (Li et al., 2008).

(ii) It uses the new weighting rule of AdaBoost algorithm, instead of the classic algorithm, in order to prevent the over-fitting problem. The new weighting rule which was proposed by Wang and Li (2007) is specifically designed for maximum margin classifiers such as SVM, KLR and RLR.

(iii) The adaptations use criteria 'perfect' (weighted error by *total accuracy* is zero), beside criteria no better than random (weighted error by *total accuracy* is 0.5 or more) or a fixed number of iterations, in adjusting the parameter values of NTR-KLR and NTR-LR during AdaBoost iterations. These criteria have been proposed by Meir et al. (2003), for AdaBoost algorithm with fixed parameter value.

The adaptation of Modified AdaBoost for NTR-KLR classifier resulted in adapted Modified AdaBoost I, while the adaptation for NTR-KLR classifier resulted in adapted Modified AdaBoost I. The algorithms of adapted Modified AdaBoost are applied to NTR-WKLR and NTR-WLR respectively.

(iii) The implementations of adapted Modified AdaBoost to NTR-WKLR and NTR-WLR

The implementations of adapted Modified AdaBoost to NTR-WKLR and NTR-WLR component classifier respectively resulted in proposed AB-WKLR and proposed AB-WLR. Both proposed algorithms mainly consist of two loops, adapted Modified AdaBoost and weighted component classifier (NTR-WKLR or NTR-WLR). First main loop represents the outer loop which is adapted Modified AdaBoost iterations to generate weight adaptively and to ensemble a set of weighted component classifier. Starting from large parameter value, corresponding to component classifier with weak learning ability (have low *total accuracy*), it is trained as long possible to meet criteria "perfect accuracy" or ''less than half accuracy" or maximum iterations. Otherwise, we decrease the parameter value to increase the learning ability of component classifier. The process of adapted Modified AdaBoost continues until given minimal value of parameter. Second main loop represents the inner loop for AB-WKLR and AB-WLR which is weighted component classifier (NTR-WKLR or NTR-WLR) iterations that find the Newton direction approximation of WKLR and WLR respectively by using Truncated Newton approach. The weighted component classifier iterations are terminated when Linear CG iterations are terminated and the relative difference of optimization function is no larger than a specified threshold, ε_1. Linear CG iterations are terminated, when the square residual is no greater than a specified threshold, ε_2.

Proposed AB-WKLR: The use of adapted Modified AdaBoost I to NTR-WKLR

The strategy of parameter adjusting for AB-WKLR is designed in the following. Minimal value of RBF kernel parameter, σ_{min}, is determined as optimal parameter value of σ. It is obtained by doing selection model of NTR-KLR. The selection model is conducted with grid search method and 5-Fold SCV, by maximizing *total accuracy* value. Therefore $\sigma_{min} = \sigma_{opt}$ of NTR-KLR = $\exp(c_2)$.

The initial σ, maximal value of RBF kernel parameter, σ_{max}, are defined by formula, $\sigma_{max} = \exp(c_2 + d_u)$. Upper deviation parameter, d_u, is determined as number of

σ adjusting which is empirically set as a value within 0.5 – 6. We choose d_u that gives best generalization of AB-WKLR. The σ_{max} is decreased with σ_{step} =-0.5, by transforming the value of σ with previous log natural. The σ_{max} is decreased slightly to prevent the new NTR-KLR from being too strong for the current weight distribution. The maximum number iterations for AB-WKLR is set by calculating with formula, 30 x n_σ, where n_σ is defined as number of σ which is set as (2 x d_u) + 1.

As the classification performance of NTR-KLR is mainly affected by RBF kernel parameter, σ, a fixed value of regularization parameter (λ) can be employed. In order to achieve optimal generalization performance of AB-WKLR, λ is empirically set as an optimal value within exp(-3) to exp(-6) for all experiments. The optimal λ for AB-WKLR is conducted with grid search method and 5-Fold SCV by maximizing *g-means* value.

Algorithm 3.3 The Proposed AB-WKLR

Algorithm 1. Adapted Modified AdaBoost I (Outer loop)

Input : **x**, **y**, $\alpha^{(\tau)}$

Initialize: The weight of training data $\mathbf{w}_t(i)$, where $\mathbf{w}_1(i)$= 1/n; i = 1,2,..n

The initial σ, σ_{max}; the minimal σ, σ_{min}; the step of σ, σ_{step}

Output : the class prediction, $y_{pred}(\mathbf{x}) = sign\left(\sum_{t}^{T} \omega_t h_t(\mathbf{x})\right)$

Do while ($\sigma > \sigma_{min}$)

For t = 1 to *max iteration for each σ*

(1) Get the logit model, f_t, and then get the estimation of classification rule, h_t, by performing NTR-WKLR (NTR-KLR with **w)**.

(2) Calculate the weighted error of h_t: $\varepsilon_t = \sum_{i=1}^{n} w_t(i), y_i \neq h_t(\mathbf{x})$

(3) If $\varepsilon_t > 0.5$ or $\varepsilon_t = 0$, decrease σ value by σ_{step} and go back to (1)

(4) Set the weight of h_t: $\omega_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

(5) For i = 1 to n

Update the weight of training data:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} x \begin{cases} \exp(\omega_t) + (1 - \exp(\omega_t))(1 - \exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) \le 0 \\ \exp(-\omega_t) + (1 - \exp(-\omega_t))(\exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) > 0 \end{cases}$$

Z_t is normalization factor

End For

End For

where *max iterations for each* $\sigma = 30$

Algorithm 2. NTR-WKLR (Inner loop)

Algorithm 2a. Weighted KLR-NR (Outer iterations)

Input: $\mathbf{K}_1^{(t)}$,**y**, λ, $\mathbf{w}^{(t)}$, $\sigma^{(t)}$,

Initialize:$\boldsymbol{\alpha}^{(1)}$, $L^{(1)}$,

Output : $\boldsymbol{\alpha}^{(\tau+1)}$

Do while $\left| \frac{L^{(t+1)} - L^{(t)}}{L^{(t+1)}} \right| > \varepsilon_1$

For t = 1 to *max WKLR-NR iterations*

(1) Compute $\mathbf{p}^{(t)}$ = 1./(1+exp(**y**.$\mathbf{K_1}\boldsymbol{\alpha}^{(t)}$))

(2) Compute variance : $\mathbf{V}^{(t)}$ = diag($\mathbf{p}^{(t)}$.(1-$\mathbf{p}^{(t)}$))

(3) Compute $\mathbf{g}^{(t)}$ dan $\mathbf{H}^{(t)}$ of WKLR

(4) Compute NR update rule solution : $\mathbf{H}^{(t)}\,\mathbf{s}^{(t)} = -\mathbf{g}^{(t)}$

(5) Compute $\boldsymbol{\alpha}^{(\tau+1)}$ by NR : $\boldsymbol{\alpha}^{(\tau+1)} = \boldsymbol{\alpha}^{(\tau)} + \mathbf{s}^{(t)}$

(6) Compute $L^{(t+1)}$

End For

where ε_1=2.5, *max Weighted KLR- NR iterations* = 30

Algorithm 2b. Linear CG (Inner Iterations)

Input : $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$ of WKLR

Initialize: $\mathbf{s}^{(1)}$, $\mathbf{r}^{(1)}$=-**g**, $\mathbf{d}^{(1)}$= $\mathbf{r}^{(1)}$

Output : $\mathbf{s}^{(\tau+1)}$

Do while $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length : $a^{(\tau)} = \mathbf{r}^{T(t)}\,\mathbf{r}^{(t)} / (\mathbf{d}^{T(t)}\,\mathbf{H}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution : $\mathbf{s}^{(\tau+1)} = \mathbf{s}^{(\tau)} + \mathbf{a}^{(\tau)}\,\mathbf{d}^{(t)}$

(3) Update the residual : $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\,\mathbf{H}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer : $\mathbf{c}^{(t)} = \mathbf{r}^{T(t+1)}\ \mathbf{r}^{(t+1)}\ / \mathbf{r}^{T(t)}\mathbf{r}^{(t)}$

(5) Update the search direction : $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\ \mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 1000

Proposed AB-WLR: The use of adapted Modified AdaBoost II to NTR-WLR

Design of the parameter adjusting for AB-WLR is explained in the following. Similar to AB-WLR, selection model of NTR-LR with Grid Search method is conducted by maximizing *total accuracy* value with 5-Fold SCV, such that it gets λ_{min} = λ_{opt} of NTR-LR = exp(c_1).

The initial λ is determined by formula, λ_{max} = exp(c_1 + d_u), where d_u is defined as number of σ adjusting. In order to obtain best generalization for AB-WLR, d_u is empirically set as a value within 20 – 26. By transforming the value of λ with previous log natural, the λ_{max} is decreased with λ_{step} =-1. Different from AB-WKLR, the λ_{max} is decreased larger, because the variation of λ leads to smaller variation of classification performance. Similar to AB-WKLR, the maximum number of iterations for AB-WLR is set by calculated with formula, 30 x n_λ, where n_λ is determined as number of λ which is set as (2 x d_u) + 1. Evaluation of the effectiveness of AB-WLR classifier is performed by maximizing *g-means* value with 5-Fold SCV.

Algorithm 3.4 The Proposed AB-WLR

Algorithm 1. Adapted Modified AdaBoost II (Outer loop)

Input : **x**, **y**, $\beta^{(t)}$

Initialize: The weight of training data $\mathbf{w}_t(i)$, where $\mathbf{w}_1(i)$= 1/n; i = 1,2,..n

The initial σ, σ_{max}; the minimal σ, σ_{min}; the step of σ, σ_{step}

Output : the class prediction, $y_{pred}(\mathbf{x}) = sign\left(\sum_t^T \omega_t h_t(\mathbf{x})\right)$

Do while ($\lambda > \lambda_{min}$)

For t = 1 to *max iteration for each* λ

(1) Get the logit model, f_t, and then get the estimation of classification rule, h_t, by performing NTR-WLR (NTR-LR with **w**).

(2) Calculate the weighted error of h_t: $\varepsilon_t = \sum_{i=1}^{n} w_t(i),\ y_i \neq h_t(\mathbf{x})$

(3) If $\varepsilon_t > 0.5$ or $\varepsilon_t = 0$, decrease σ value by σ_{step} and go back to (1)

(4) Set the weight of h_t: $\omega_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

(5) For i = 1 to n

Update the weight of training data:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} x \begin{cases} \exp(\omega_t) + (1-\exp(\omega_t))(1-\exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) \leq 0 \\ \exp(-\omega_t) + (1-\exp(-\omega_t))(\exp(y_i f_t(\mathbf{x}_i))), & y_i f_t(\mathbf{x}_i) > 0 \end{cases}$$

Z_t is normalization factor

End For

End For

where *max iterations for each* λ = 30

Algorithm 2. NTR-WLR (Inner loop)

Algorithm 2a. Weighted RLR-NR (Outer iterations)

Input: **x**, **y**, $\lambda^{(t)}$, $\mathbf{w}^{(t)}$

Initialize:$\boldsymbol{\beta}^{(1)}$, $L^{(1)}$,

Output : $\boldsymbol{\beta}^{(\tau+1)}$

Do while $\left|\frac{L^{(t+1)} - L^{(t)}}{L^{(t+1)}}\right| > \varepsilon_1$

For t = 1 to *max WLR-NR iterations*

(1) Compute $\mathbf{p}^{(t)} = 1./(1+\exp(\mathbf{y}.\mathbf{X}\boldsymbol{\beta}^{(t)}))$

(2) Compute variance : $\mathbf{V}^{(t)} = \text{diag}(\mathbf{p}^{(t)}.(1-\mathbf{p}^{(t)}))$

(3) Compute NR update rule solution : $\mathbf{H}^{(t)}\mathbf{s}^{(t)} = -\mathbf{g}^{(t)}$

(5) Compute $\boldsymbol{\beta}^{(\tau+1)}$ by NR : $\boldsymbol{\beta}^{(\tau+1)} = \boldsymbol{\beta}^{(\tau)} + \mathbf{s}^{(t)}$

(6) Compute $L^{(t+1)}$

End For

where ε_1=0.01, *max Weighted RLR- NR iterations* = 30

Algorithm 2b. Linear CG (Inner Iterations)

Input : $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$ of WLR

Initialize: $\mathbf{s}^{(1)}$, $\mathbf{r}^{(1)}$=-$\mathbf{g}$, $\mathbf{d}^{(1)}$= $\mathbf{r}^{(1)}$

Output : $\mathbf{s}^{(\tau+1)}$

Do while $\mathbf{r}^T\mathbf{r} > \varepsilon_2$

For t=1 to *max Linear CG iterations*

(1) Compute the optimal step length : $a^{(\tau)} = \mathbf{r}^{T(t)}\,\mathbf{r}^{(t)} / (\mathbf{d}^{T(t)}\,\mathbf{H}\,\mathbf{d}^{(t)})$

(2) Update the approximate solution : $\mathbf{s}^{(\tau+1)} = \mathbf{s}^{(\tau)} + \mathbf{a}^{(\tau)}\,\mathbf{d}^{(t)}$

(3) Update the residual : $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - \mathbf{a}^{(t)}\,\mathbf{H}\,\mathbf{d}^{(t)}$

(4) Update A-Conjugacy enforcer : $\mathbf{c}^{(t)} = \mathbf{r}^{T(t+1)}\,\mathbf{r}^{(t+1)} / \mathbf{r}^{T(t)}\mathbf{r}^{(t)}$

(5) Update the search direction : $\mathbf{d}^{(t+1)} = \mathbf{r}^{(t+1)} + \mathbf{c}^{(t)}\,\mathbf{d}^{(t)}$

End For

where ε_2 =0.005, *max Linear CG iterations* = 200

3.4 RESEARCH METHODOLOGY

This section contains the description of Research Procedure and Design of Numerical Experiment.

3.4.1 RESEARCH PROCEDURES

This sub section describes procedures as shown in Fig. 3.12 which are required to achieve the research objective of the thesis as follows:

(i) Developing the NTR-KLR algorithm as the Newton version of TR-KLR algorithm, and NTR-LR algorithm, as the Newton version of TR-IRLS algorithm (it can be seen also as non-Kernel version of NTR-KLR algorithm) (Section 3.2). This step includes;

a. Defining the regularized Negative Log-Likelihood (NLL) function of KLR and RLR respectively by assuming $y \in (-1,1)$ instead of $y \in (0,1)$. Then, obtain the gradient and Hessian matrix, by differentiating the regularized NLL function with respect to α and β respectively.

b. Determining the linear system of Newton-Raphson update rule for KLR and RLR.

c. Apply Linear CG method to quadratic form of Newton-Raphson update rule, in order to find iteratively the approximation of Newton direction for KLR-NR and RLR-NR respectively.

d. Find, iteratively, the MLE estimate of α for KLR and the MLE estimate of β for RLR, by using Newton-Raphson method.

Detail algorithm of NTR-KLR and NTR-LR respectively has been explained in sub section 3.2.2.

(ii) Develop the AB-WKLR algorithm, as the imbalanced NTR-KLR algorithm and AB-WLR algorithm, as the imbalanced NTR-LR algorithm (Section 3.3).

(a) Developing NTR-Weighted KLR (NTR-WKLR) and NTR-Weighted RLR (NTR-WLR) such that can directly deal with weight distributions which are generated by adaptation of modified AdaBoost algorithms respectively.

(1) Modifying the NLL loss function on regularized optimization function of KLR and RLR respectively as the Weighted NLL loss function. Hence, KLR model becomes Weighted KLR (WKLR) model and RLR becomes Weighted RLR (WLR). The regularized function of WKLR and WLR respectively is termed as Regularized Weighted NLL function.

(2) By using similar steps as on NTR-KLR and NTR-LR algorithm above, estimate WKLR and WLR model respectively using Truncated Newton method.

Detail algorithm of NTR-WKLR and NTR-WLR are included in algorithm of AB-WKLR and AB-WLR respectively in sub section 3.3.3 (iii).

(b) Developing the adaptations of Modified AdaBoost algorithm for NTR-KLR and NTR-LR which are highly accurate (strong) classifier respectively in order to benefit from Boosting mechanism (Sub section 3.3.3 (ii)).

The both adaptations of modified AdaBoost algorithm focus on determining the process of AdaBoost iterations by taking into account the parameter influence on classification performance of NTR-KLR and NTR-LR respectively.

Detail strategies of parameter adjusting during the process of adapted Modified AdaBoost iterations and algorithms of both adaptations are included in algorithm of AB-WKLR and AB-WLR respectively in sub section 3.3.3 (iii).

(c) Employing the adapted Modified AdaBoost I to NTR-WKLR and applying the adapted Modified AdaBoost II to NTR-WLR (Sub section 3.3.3 (iii)). This step includes;

(1) Train NTR-WKLR and NTR-WLR respectively then obtain the weighted prediction.

(2) Calculate the weighted error of NTR-WKLR and NTR-WLR respectively.

(3) Compute the weight of weighted prediction for NTR-WKLR and NTR-WLR respectively.

(4) Update the weight of training data for NTR-WKLR and NTR-WLR respectively.

Iterate step (1) – (4) by using the strategies of parameter adjusting respectively until finish, then predict by weighted majority vote.

Detail algorithm of AB-WKLR and AB-WLR respectively has been explained in sub section 3.3.3 (iii).

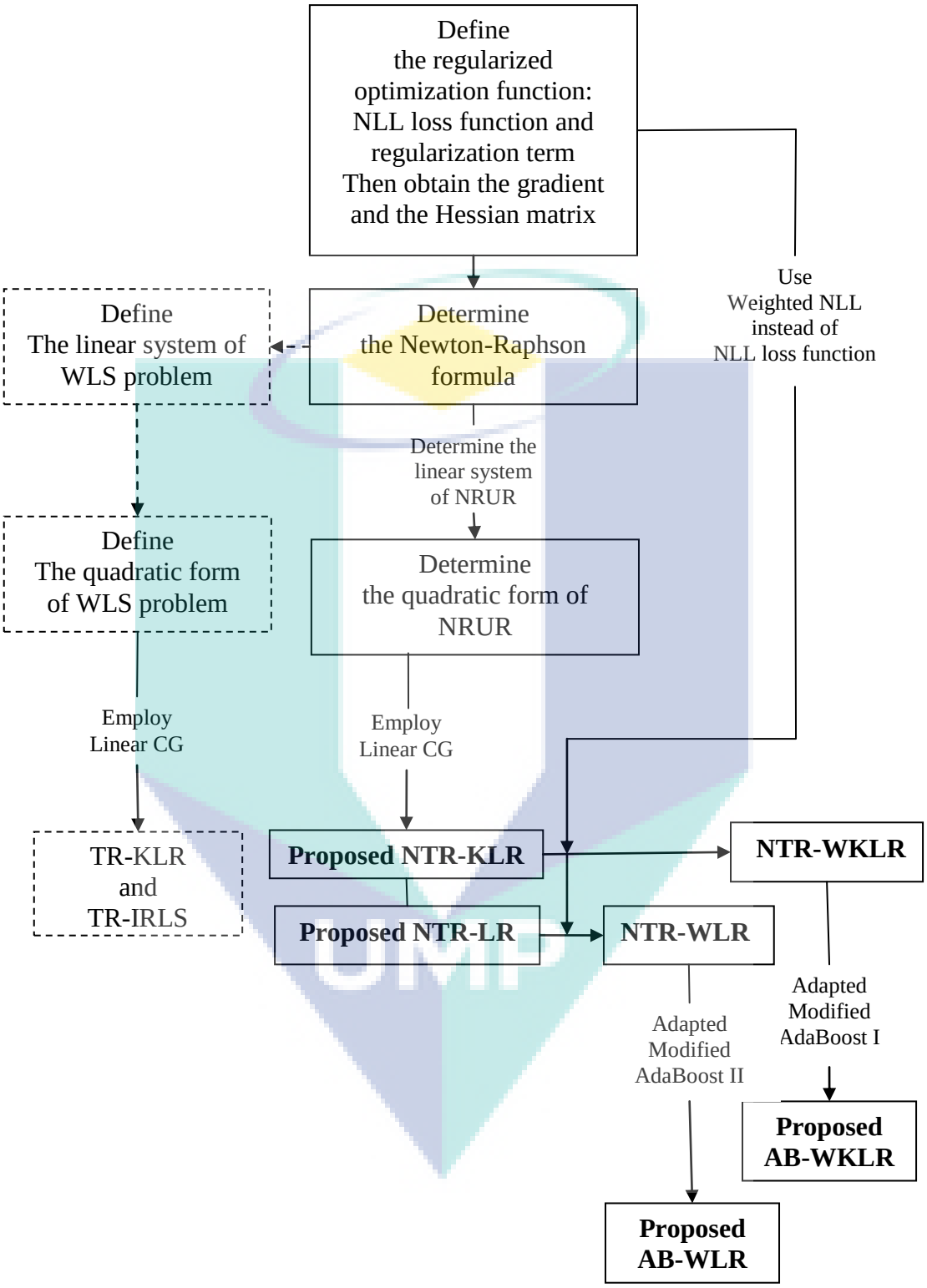


Figure 3.12: Research Procedure

3.4.2 DESIGN OF NUMERICAL EXPERIMENTS

This section includes explanation of data preparation and performance evaluation of several numerical experiments. The numerical experiments were conducted in order to evaluate the performance of the proposed algorithms. All of the computations were performed on a 2 GB RAM Computer, by using Matlab 7.

Data Preparation

This thesis uses several benchmark data sets from UCI Machine Learning (Frank and Asucion, 2010), in order to evaluate the performance of the proposed algorithms. Nine data sets with varying size, dimension and degrees of class imbalance were used in this research, as presented in Table 3.3.

Except Australia, Parkinson and GammaImg data sets, most of data sets originally have more than two classes. This research converted them into imbalanced two-class data sets by combining some classes with reference to the previous research (Akbani et al., 2004; Li et al., 2008). The suffix after each name of data sets indicates the class used as the positive (minority) class. Eight data sets are imbalanced, except Australia which is the reference base.

The complexity of data depends on the product of sample (n) and number of attributes (dim) (Komarek, 2004). The complexity of GammaImg, Shuttle2to7, and LetterImg26 data set are widely higher than remaining data sets, as displayed in Table 3.3. Hence, this thesis categorizes these data sets to be large, while the remains data sets are small-to-medium. This categorization was decided based on the size of complexity of data used on the previous researches (Komarek, 2004; Maalouf, 2009).

Table 3.3: General Profiles of Data Sets

No.	Name of Data set	Attribute (*dim*)	Sample (*n*)	Minority (+) Class	Majority (-) Class	Imbalanced	
1	Australia (Credit Risk)	14	690	307	383	1.25	Balanced
2	Parkinson	22	195	48	147	3.06	Imbalanced
3	Glass7 (Forensic)	9	214	29	185	6.38	Imbalanced
4	ImgSegment1 (Image Segmentation)	19	2310	330	1980	6	Imbalanced
5	Balance2 (Psychology)	4	625	46	576	12.52	Imbalanced
6	Car3	6	1728	69	1659	24.04	Imbalanced
7	GammaImg	10	19020	6688	12332	1.84	Imbalanced
8	Shuttle2to7	8	58000	12414	45586	3.67	Imbalanced
9	LetterImg26 (Letter Image)	16	20000	734	19266	26.25	Imbalanced

In the following, short overviews of the data sets are given.

Australia

This data set contains confidential data with title *Australian Credit Approval* that concerns on credit card application. This dataset is a good mix of attributes, i.e. continuous, nominal with small numbers values and nominal with larger numbers of values.

Parkinson

This data set contains attributes of particular voice measure to discriminate healthy people from those with Parkinson disease. A range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD), is composed in this dataset.

Glass7

This data set contains nine attribute in determining whether the glass was a type of headlamp or not. If it is correctly identified, it can be used as evidence in criminological investigation. This data set contains, in total, continuously valued.

ImgSegment1

This data set contains 19 continuous attributes with title *Image Segmentation Data*. The classification task is to predict whether the pixel was brickface image or not.

Balance2

This data set contains 4 attributes of psychological experimental results in order to discriminate whether the observation was balanced or not. This data set includes four numeric attributes.

Car3

This data set, which is named as Car Evaluation Database, contains 6 attribute for making decision whether the car is good or not. Six categorical attributes are included in this data set.

GammaImg

This data set contains 10 attributes to discriminate gamma image (signal) from the image of hadronic showers initiated by cosmics rays in the upper atmosphere (background). All of the data sets are continuous values.

Shuttle2to7

This data set came from NASA that contains 9 numeric attribute. Further description of this data set is not available.

LetterImg26

This data set contains 16 attributes. The classification task is to recognize a pixel as image of capital letter *Z* or not. All of attributes are numeric values.

All of data sets were randomized previously based on simple random sampling technique (Van-Hulse et al., 2007) using random number. The data sets then were normalized to ensure the similarity of numerical range, using formula

$$\hat{x}_{ji} = \frac{x_{ji} - \bar{x}_j}{s_{x_j}} \tag{3.32}$$

where i = 1,2,..n; j = 1,2, .. no. of attributes

s_{x_j} = standard deviation of attributes

The data sets were pre-processed to guarantee the quality of numerical results, i.e. accuracy and stability of the proposed classifiers (Han and Kamber, 2006).

Performance evaluation

Several numerical experiments are conducted to evaluate the performance of proposed methods respectively which can be described as follows (Fig. 3.13):

(i) Proposed NTR-KLR and Proposed NTR-LR

(a) The effectiveness of Truncated Newton method in NTR-KLR and NTR-LR algorithm was shown respectively in solving the numerical problem of KLR-NR and RLR-NR respectively.

It is demonstrated by plotting the training time to convergence versus the relative difference of optimization function for:

(1) NTR-KLR and KLR-NR on five small-to-medium size data sets.

(2) NTR-LR and RLR-NR on three large size data sets.

Although proposed NTR-KLR and proposed NTR-LR algorithm are general classifiers, all of the data sets used in determining the effectiveness of Truncated Newton method, are imbalanced.

(b) Evaluation of the effectiveness performance of proposed NTR-KLR and proposed NTR-LR classifier was conducted respectively by comparing both proposed algorithms to RBFSVM based on two criteria i.e. *g-means* and Area Under Receiver Operating Curve (AUC) value (Fawcett,2004).

.

(c) In relation to further development of NTR-KLR and NTR-LR respectively for classification of imbalanced data sets, the problem of general classifiers i.e. NTR-KLR and NTR-LR on classifying the imbalanced data sets was determined previously. It was evaluated respectively along with the study on

the proper use of *g-means* metric, as the accuracy metric on imbalanced classification problem. The study on the problem of NTR-KLR and NTR-LR on classifying the imbalanced data sets, totally, uses eight imbalanced data sets. Meanwhile the study on the proper use of *g-means* metric uses eight imbalanced data sets and one balanced data set as the reference base i.e. Australian data set.

(ii) Proposed AB-WKLR and Proposed AB-WLR.

(a) The effectiveness of adapted Modified AdaBoost method in AB-WKLR and AB-WLR algorithm was shown respectively in solving the imbalanced problem of NTR-KLR and NTR-LR respectively, as follows

(a.1) Compare AB-WKLR to NTR-KLR on five small-to-medium size data sets.

(a.2) Compare AB-WLR to NTR-LR on three large size data sets.

The comparison was conducted based on criteria: improvement in reducing error by *g-means*, *standard deviation* of *g-means* values during *5*-fold SCV and statistical significance using Paired *t* test (Montgomery, 1991).

(b) The effectiveness performance of proposed AB-WKLR and AB-WLR classifier on imbalanced classification problems was evaluated respectively in this thesis, by comparing both proposed algorithms to AdaBoostSVM based on two criteria i.e. *g-means* and AUC value.

The tradeoff between bias and variance in cross validation depends on the number of folds (Diamantidis et al., 2000). Hence, this thesis applies a moderate number of folds (*5*-Fold SCV) on performing the proposed algorithms.

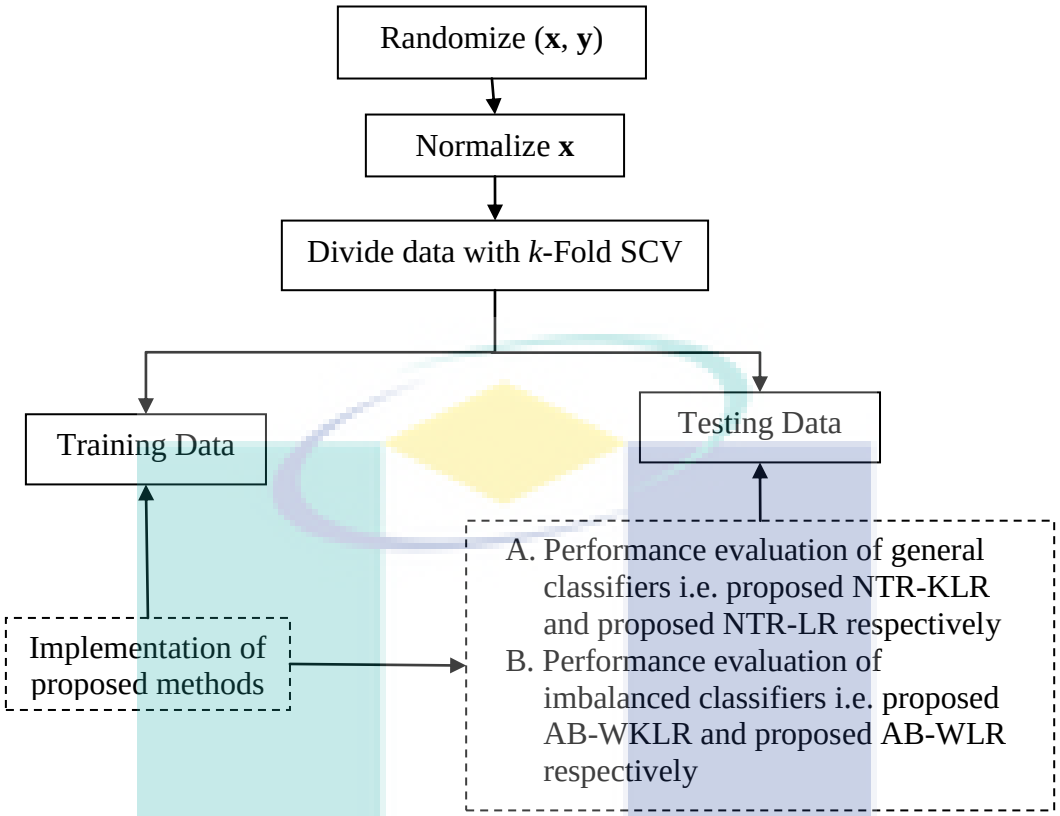


Figure 3.13: Performance evaluation

CHAPTER 4

NUMERICAL RESULTS AND DISCUSSION

4.1 INTRODUCTION

This chapter presents several numerical results along with the discussion on the performance evaluation of proposed general classification algorithms (NTR-KLR and NTR-LR) and proposed imbalanced classification algorithms (AB-WKLR and AB-WLR).

4.2 PROPOSED NTR-KLR AND NTR-LR: NUMERICAL RESULTS AND DISCUSSION

Several numerical results along with the discussion on the performance evaluation of NTR-KLR and NTR-LR algorithm respectively are presented in this section. In general, this section is divided into three sub sections, i.e. 4.2.1 Accuracy, stability and numerical convergence of proposed algorithms; 4.2.2 The effectiveness of Truncated Newton method in proposed algorithms; 4.2.3 Performances comparison of proposed algorithms to RBFSVM.

4.2.1 Numerical convergence, accuracy, stability and of NTR-KLR and NTR-LR

Table 4.1 displays iterations number and *g-means* value of NTR-KLR. It can be observed that the maximum number of iterations by Linear CG (LCG) during 5-Fold SCV did not reach 1000, which is the maximum number of iterations that is set for Linear CG, as the inner loop of NTR-KLR algorithm. The maximum number of iterations by Linear CG during 5-Fold SCV was reached on *ImgSegment1* and *Car3*

data sets which are the most complex data sets among five small-to-medium data sets on applying NTR-KLR algorithm.

Table 4.1: Iterations number and *g-means* value of NTR-KLR algorithm by maximizing *g-means* value with *5*-Fold SCV

Fold	No	Name of Data Set	No. of Iterations		*g-means*				
			NR	LCG	Park	Glass7	ImgS1	Blc2	Car3
1	1	Parkinson	2	[115,39]	0.9826				
	2	Glass7	2	[27,14]		0.8165			
	3	ImgSegment1	2	[468,153]			0.9911		
	4	Balance2	1	45				0	
	5	Car3	2	[777,134]					0.9593
2	1	Parkinson	2	[110,40]	0.8630				
	2	Glass7	2	[26,18]		0.9129			
	3	ImgSegment1	2	[557,189]			1		
	4	Balance2	1	47				0	
	5	Car3	2	[746,141]					0.8452
3	1	Parkinson	2	[133,33]	0.9154				
	2	Glass7	2	[41,20]		0.9129			
	3	ImgSegment1	2	[596,165]			1		
	4	Balance2	1	37				0.3162	
	5	Car3	2	[882,137]					0.8851
4	1	Parkinson	2	[117,41]	0.8367				
	2	Glass7	2	[50,17]		1			
	3	ImgSegment1	2	[626,167]			0.9847		
	4	Balance2	1	37				0	
	5	Car3	2	[774,137]					0.8864
5	1	Parkinson	2	[134,41]	1				
	2	Glass7	2	[47,14]		0.9864			
	3	ImgSegment1	2	[577,194]			0.9924		
	4	Balance2	1	52				0	
	5	Car3	2	[718,128]					0.9608
		Average of *g-means* (*S*-gmeans)			**0.9195 (0.0716)**	**0.9257 (0.0732)**	**0.9937 (0.0065)**	**0.0632 (0.1414)**	**0.9073 (0.0509)**

The more important is that the maximum number of iterations by iterative method of Newton-Raphson (NR) during 5-Fold SCV is 2. It did not reach to 30, which is the maximum iterations set for KLR-NR, as the main loop of NTR-KLR classifier. Thus, it is convergence guarantee for NTR-KLR algorithm. Table 4.1 also summarizes the *g-means* value in applying the NTR-KLR classifier during *5*-Fold SCV.

Similar to the implementation of NTR-KLR algorithm, the maximum number of iterations by RLR-NR and Linear CG in applying the NTR-LR algorithm did not reach

30 and 200, which are the maximum iterations set for RLR-NR and Linear CG (Table 4.2).

Table 4.2: Iteration number and *g-means* value of NTR-LR algorithm by maximizing *g-means* value with *5*-Fold SCV

Fold	No	Name of Data Set	No. of Iterations		*g-means*		
			NR	LCG	GmImg	Shuttle	LetImg
1	1	GammaImg	3	[11,11,11]	0.7270		
	2	Shuttle2to7	7	[11,11,11, 9,11,9,9]		0.9514	
	3	LetterImg26	8	[15,15,15, 14,14,14, 14,13]			0.8185
2	1	GammaImg	3	[11,11,11]	0.7339		
	2	Shuttle2to7	7	[11,11,9,9 ,11,9,9]		0.9573	
	3	LetterImg26	8	[15,15,14, 14,14,14, 14,13]			0.8144
3	1	GammaImg	3	[11,11,11]	0.7384		
	2	Shuttle2to7	7	[11,10,9,9 ,11,9,9]		0.9508	
	3	LetterImg26	8	[16,15,14, 14,13,14, 14,13]			0.8266
4	1	GammaImg	3	[11,11,11]	0.7261		
	2	Shuttle2to7	7	[10,11,9,1 1,11,9,9]		0.9481	
	3	LetterImg26	8	[15,15,14, 14,13,14, 14,13]			0.8350
5	1	GammaImg	3	[11,11,11]	0.7202		
	2	Shuttle2to7	7	[11,11,10, 11,11,9,9]		0.9515	
	3	LetterImg26	8	[15,15,15, 14,14,14, 14,12]			0.8051
Average of *g-means* (*S*-gmeans)					**0.7291 (0.0071)**	**0.9518 (0.0048)**	**0.8199 (0.0115)**

Those results are similar to that stated by Komarek and Moore (2005) that those numbers should not be reached. It is indicated on Table 4.2, during *5*-Fold SCV, eight was reached as the maximum number of iterations by RLR-NR, while the maximum iterations by LCG was 15 which is indicating the consistency with the convergence theory of CG method (Lewis et al., 2006).

This research also lists down the *g-means* values in applying NTR-KLR and NTR-LR classifier respectively during *5*-Fold SCV. They were measured along with standard deviation of g-means values in evaluating the accuracy of NTR-KLR and NTR-LR classifier and its stability (Table 4.1 and Table 4.2).

4.2.2 The effectiveness of Truncated Newton in NTR-KLR and NTR-LR

As aforementioned, the main numerical problem of KLR-NR and RLR-NR is the huge matrix that needs to be inverted, such that the computation to be slow and probably the matrix to be singular. In this sub section, several experiments were conducted in order to evaluate the effectiveness of Truncated Newton in proposed NTR-KLR and proposed NTR-LR algorithm on solving the numerical problem of KLR-NR and RLR-NR respectively.

The effectiveness of Truncated Newton method was determined by plotting the training time versus relative difference of optimization function for both proposed algorithms then compared to KLR-NR and RLR-NR algorithm respectively. The plots can be seen in Fig. 4.1 and Fig. 4.2.

Fig. 4.1 and Fig. 4.2 show that the proposed algorithms have performed faster than KLR-NR and RLR-NR respectively in decreasing the relative difference of optimization function. Moreover, NTR-KLR can handle the singularity problem as indicated in Fig. 4.1 (a.2, a.3 and a.4), on which KLR-NR cannot handle. It means that the use of the Truncated Newton method in NTR-KLR and NTR-LR algorithm respectively is effective in solving the numerical problem of KLR-NR and RLR-NR i.e. the singularity and the training time problem. Therefore, NTR-KLR outperforms KLR-NR on small-to-medium size data sets when the singularity and the training time problem exists, while NTR-LR has performed better than RLR in handling the training time problem on large size data sets. These results can be seen as further explanation to the effectiveness of Truncated Newton method in TR-KLR (Maalouf et al, 2010) and TR-IRLS (Komarek and Moore, 2005) algorithm respectively, because of the equivalence of iterative method used by these algorithms.

NTR-KLR

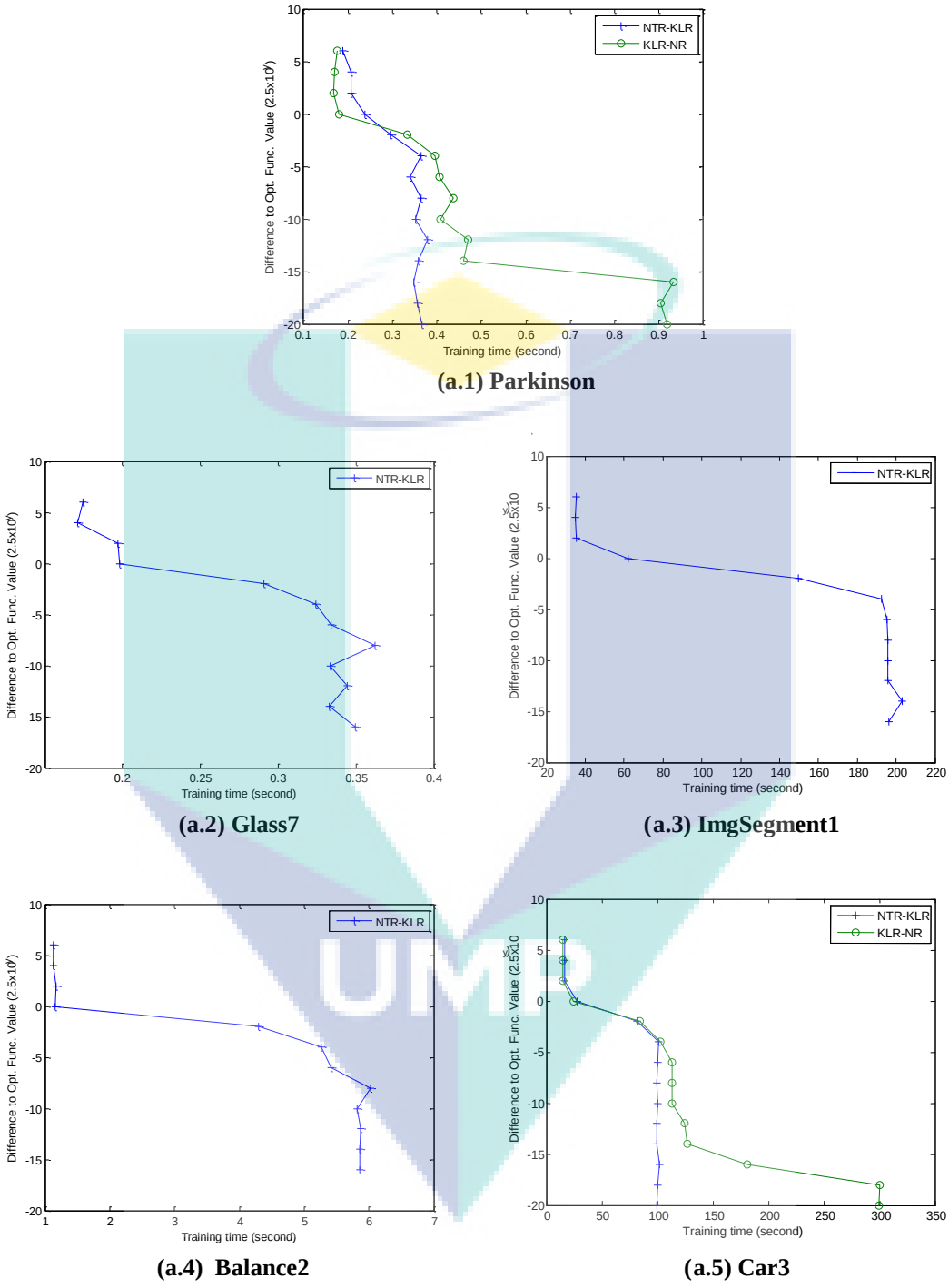
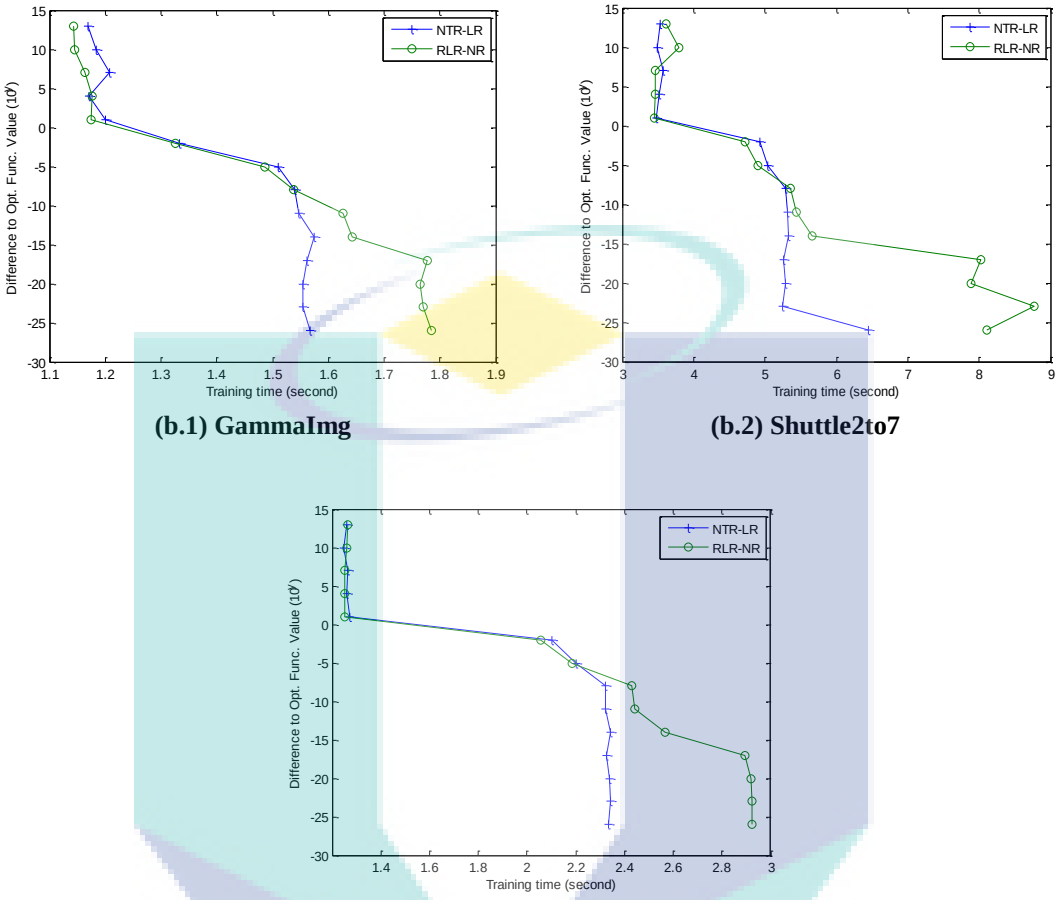


Figure 4.1: Comparison of algorithm performance between NTR-KLR and KLR-NR

NTR-LR



(b.1) GammaImg **(b.2) Shuttle2to7**

(b.3) LetterImg26

Figure 4.2: Comparison of algorithm performance between NTR-LR and RLR-NR

4.2.3 Performances Comparison of proposed NTR-KLR and proposed NTR-LR to RBFSVM

Furthermore, the summary of performance comparison between proposed classifiers with their related work (RBFSVM) is given in Table 4.3. The performances comparison was conducted based on criteria: *g-means* value and Area Under Receiver Operating Curve (AUC) (Fawcett, 2004).

Table 4.3: Summary of comparison: proposed NTR-KLR vs RBFSVM and NTR-LR vs RBFSVM

Data Set	*g-means*		AUC	
	Proposed NTR-KLR	**RBFSVM (Li et al., 2008)**	**Proposed NTR-KLR**	**RBFSVM (Li et al., 2008)**
Glass7	0.9257*	0.867	0.9723*	0.943
ImgSegment1	0.9937	0.995*	0.9999*	0.998
Car3	0.9073*	0	1*	0.631
	Proposed NTR-LR	**SVM (Li et al., 2008)**	**Proposed NTR-LR**	**SVM (Li et al., 2008)**
LetterImg26	0.8199*	0.818	0.990439*	0.933

It is found that NTR-KLR and RBFSVM have comparable value of *g-means* and AUC on *Glass7* and *ImgSegment1* data set. Hence, although NTR-KLR performs much better than RBFSVSVM on classifying *Car3* data set, both classifiers have comparable performance in general.

Meanwhile, NTR-LR and RBFSVM have comparable performance on LetterImg26 data set. In addition, NTR-LR has simple solution with the use of unconstrained optimization problem and without the use of Kernel function.

4.3 PROPOSED AB-WKLR and AB-WLR: NUMERICAL RESULTS and DISCUSSION

This section presents and discusses several numerical results on the performance evaluation of AB-WKLR and AB-WLR classifier on imbalanced data sets with varying degrees of imbalance. Three sub sections are included in this section: 4.3.1 Accuracy, stability and numerical convergence of proposed algorithms; 4.3.2 The effectiveness of adapted Modified AdaBoost methods in proposed algorithms; 4.3.3 Performances comparison of proposed algorithms to AdaBoostSVM.

4.3.1 Accuracy, stability and numerical convergence of AB-WKLR and AB-WLR

Table 4.4 displays the summary of AB-WKLR performance respectively on five imbalanced data sets. In detail, it contains σ adjusting strategies, optimal value of λ, accuracy values (*sensitivity*, *specificity*, *g-means*) and stability indicator (*standard*

deviation of *g-means*, $S_{g\text{-}means}$) with 5-Fold SCV. It can be observed in Table 4.4 that AB-WKLR classifier demonstrates best generalization performance on *ImgSegment1* data set. It performs best accuracy performance (highest *g-means* value) and best stability performance (lowest $S_{g\text{-}means}$).

Meanwhile, the summary of AB-WLR performances on three imbalanced data sets is displayed in Table 4.5. It consists of λ adjusting strategies, accuracy values (*sensitivity*, *specificity*, *g-means*) and stability indicator (*standard deviation* of *g-means*, $S_{g\text{-}means}$) with 5-Fold SCV. AB-WLR classifier performs best accuracy performance on LetterImg26 data set, while best stability performance of AB-WLR classifier was shown on Shuttle2to7 data set as indicated in Table 4.5.

It can be observed also in Table 4.4, AB-WKLR classifier has higher *sensitivity* value than *specificity* value on *Balance2* and *Car3* data set, while on other three data sets, AB-WKLR classifier performs higher on *specificity* values. On the other hand, AB-WLR classifier has higher *sensitivity* value than *specificity* value on *GammaImg* and *LetterImg26* data set, while AB-WKLR classifier performs higher *specificity* value on *Shuttle2to7* data set, as indicated in Table 4.5. Hence, *g-means* is the proper evaluation metric in accommodating the non-standard and imbalanced situations like these.

Table 4.4: Summary of AB-WKLR performance by maximizing *g-means* value with *5*-Fold SCV

No.	Name of Data set	Optimal Parameter	Class Accuracy		*g-means* ($S_{g\text{-}means}$)
		λ_{opt}	Minority (+) *Sensitivity*	Majority (-) *Specificity*	
1	**Parkinson**				
	σ_{min} =exp(1)	Exp(-5.5)	0.9246	0.94	**0.9320**
	σ_{max} =exp(1.5)				**(0.0433)**
	d_u = 0.5; σ_{step} = -0.5				
2	**Glass7**				
	σ_{min} =exp(1)	Exp(-6)	0.9622	1	**0.9808**
	σ_{max} =exp(5.5)				**(0.0157)**
	d_u = 4.5; σ_{step} = -0.5				
3	**ImgSegment1**				
	σ_{min} =exp(0)	Exp(-3)	0.9980	1	**0.9990**
	σ_{max} =exp(1)				**(0.0006)**
	d_u = 1; σ_{step} = -0.5				
4	**Balance2**				
	σ_{min} =exp(1)	Exp(-3.5)	0.9045	0.8356	**0.8655**
	σ_{max} =exp(7)				**(0.0752)**
	d_u =6; σ_{step} = -0.5				
5	**Car3**				
	σ_{min} =exp(0)	Exp(-3)	0.9940	0.9714	**0.9822**
	σ_{max} =exp(6)				**(0.0323)**
	d_u = 6; σ_{step} = -0.5				

Table 4.5: Summary of AB-WLR performance by maximizing *g-means* value with *5*-Fold SCV

No.	Name of Data set	Class Accuracy		*g-means* ($S_{g\text{-}means}$)
		Minority (+) *Sensitivity*	Majority (-) *Specificity*	
1	**GammaImg**			
	λ_{min} =exp(-0.5)	0.8123	0.7087	**0.7587**
	λ_{max} =exp(21.5)			**(0.0061)**
	d_u = 22; σ_{step} = -1			
2	**Shuttle2to7**			
	λ_{min} =exp(-3)	0.9665	0.9799	**0.9732**
	λ_{max} =exp(22)			**(0.0025)**
	d_u = 25; σ_{step} = -1			
3	**LetterImg26**			
	λ_{min} =exp(-3)			
	λ_{max} =exp(17)	0.9417	0.9414	**0.9415**
	d_u = 20;σ_{step} = -1			**(0.0093)**

Detail *g-means* values during 5-Fold SCV along with $S_{g\text{-}means}$ of AB-WKLR are summarized in Table 4.6. This table also presents the number of σ which is set and the number of iterations which was reached by AB-WKLR to converge.

Table 4.6: Number of σ, number of iterations and *g-means* value of AB-WKLR classifier by maximizing *g-means* value with 5-Fold SCV

Fold	No.		n_σ	Number of iterations	*g-means*				
					Park	Glass7	ImgS1	Blc2	Car3
1	1	Parkinson	2	2	0.9649				
	2	Glass7	10	152		1			
	3	ImgSegment1	3	3			0.9987		
	4	Balance2	13	172				0.9278	
	5	Car3	13	165					0.9955
2	1	Parkinson	2	2	0.8808				
	2	Glass7	10	140		0.9726			
	3	ImgSegment1	3	3			0.9987		
	4	Balance2	13	186				0.7472	
	5	Car3	13	193					0.9955
3	1	Parkinson	2	2	0.8983				
	2	Glass7	10	129		0.9864			
	3	ImgSegment1	3	5			1		
	4	Balance2	13	165				0.9151	
	5	Car3	**13**	**206**					0.9244
4	1	Parkinson	2	2	0.9322				
	2	Glass7	10	112		0.9586			
	3	ImgSegment1	3	5			1		
	4	Balance2	13	151				0.9022	
	5	Car3	13	147					0.9985
5	1	Parkinson	2	3	0.9837				
	2	Glass7	10	132		0.9864			
	3	ImgSegment1	3	4			1		
	4	Balance2	13	158				0.8351	
	5	Car3	13	145					0.9970
		Average of *g-means* ($S_{g\text{-}means}$)			**0.9320 (0.0433)**	**0.9808 (0.0157)**	**0.9990 (0.0006)**	**0. 8655 (0.0752)**	**0.9822 (0.0323)**

As explained in Chapter 3, the maximum number of iterations for AB-WKLR is set to 30 x n_σ. It can be observed in Table 4.6 that this number has never been reached by AB-WKLR. The maximum number of iterations was reached by AB-WKLR during 5-Fold SCV with 206 iterations. It was reached on *Car3* data set. It means that AB-WKLR classifier takes 206 iterations to converge therefore it consists of 206 NTR-WKLR component classifiers on final iteration.

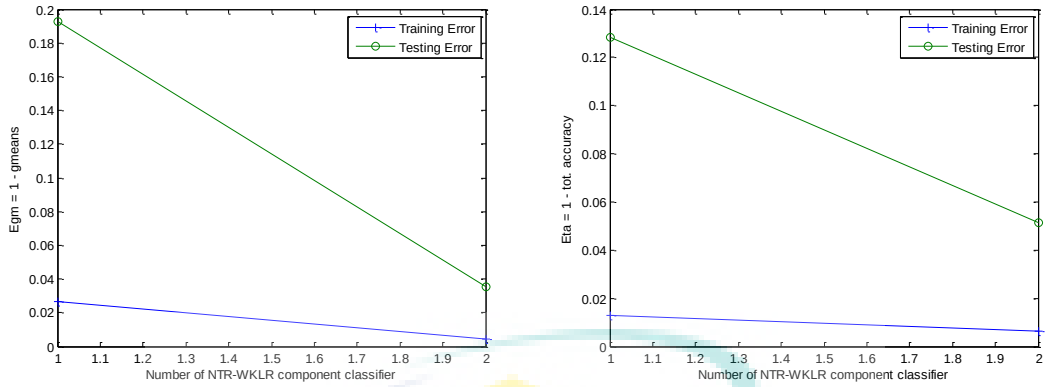
Table 4.7 provides detail *g-means* values during *5*-Fold SCV along with $S_{g\text{-}means}$ of AB-WLR classifier. The number of λ which is set and the number of iterations which was required by AB-WLR to converge are displayed also in Table 4.7.

Table 4.7: Number of λ, number of iteration and *g-means* value of AB-WLR classifier by maximizing *g-means* value with *5*-Fold SCV

Fold	No.	Name of Data Set	n_λ	No. of Iterations	*g-means*		
					GmImg	Shuttle	LetImg
1	1	GammaImg	23	151	0.7506		
	2	Shuttle2to7	26	289		0.9734	
	3	LetterImg26	21	253			0.9557
2	1	GammaImg	23	147	0.7673		
	2	Shuttle2to7	26	289		0.9759	
	3	LetterImg26	21	284			0.9326
3	1	GammaImg	23	145	0.7578		
	2	Shuttle2to7	26	253		0.9745	
	3	LetterImg26	21	250			0.9420
4	1	GammaImg	23	194	0.7569		
	2	Shuttle2to7	26	224		0.9691	
	3	LetterImg26	21	238			0.9433
5	1	GammaImg	23	213	0.7609		
	2	Shuttle2to7	26	122		0.9731	
	3	LetterImg26	21	256			0.9337
Average of *g-means* (*S-gmeans*)					**0.7587 (0.0061)**	**0.9732 (0.0025)**	**0.9415 (0.0093)**

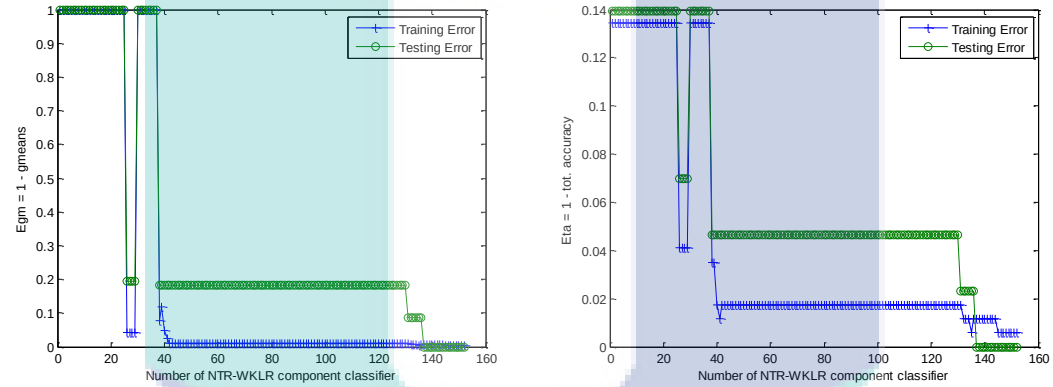
It can be observed in Table 4.7 that the maximum number of iterations for AB-WLR which is set to *30* x n_λ has never been reached by AB-WLR. The maximum number of iterations was reached by AB-WLR during *5*-Fold SCV with 289 iterations. It was reached on Shuttle2to7 data set. It means that, AB-WLR classifier contains 289 NTR-WLR component classifiers on final iteration.

Fig. 4.3 – Fig 4.7 show the error curves during AB-WKLR iterations, while the curves of error during AB-WLR iterations are displayed in Fig. 4.8 – Fig 4.10. In order to give better idea on the proper use of *g-means* metric, error curves by *g-means* and error curves by *total accuracy* are displayed separately.



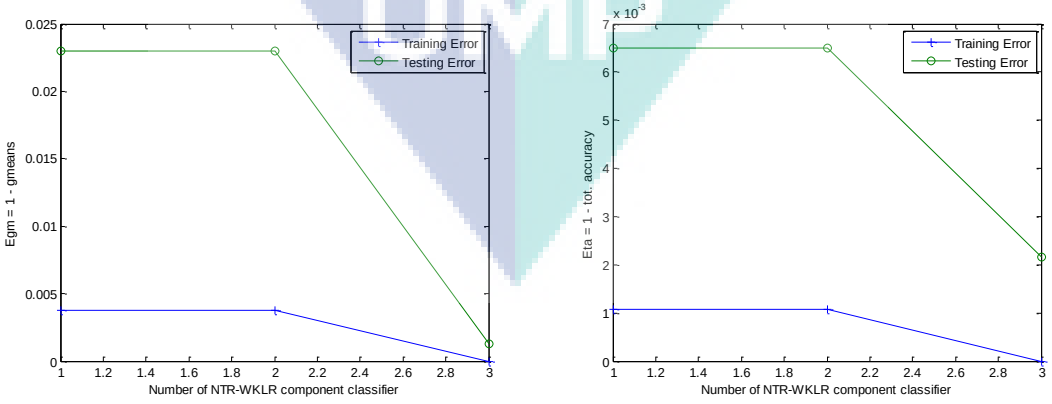
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.3: Error curve for AB-WKLR on first fold of *Parkinson* data set



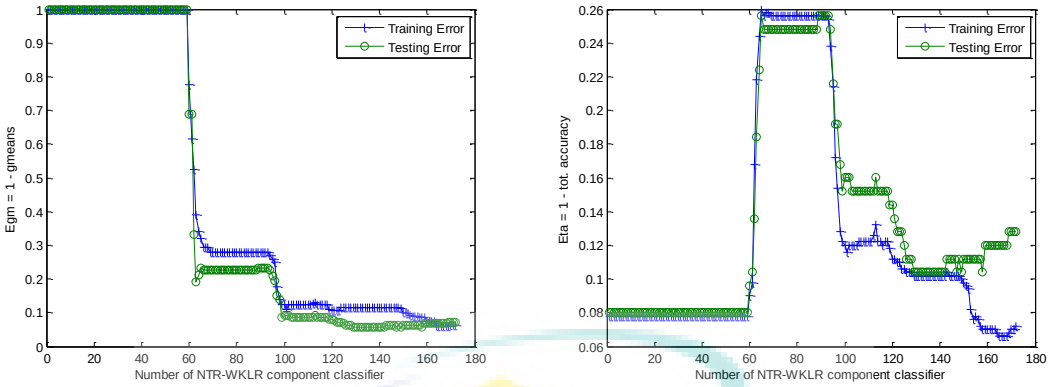
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.4: Error curve for AB-WKLR on first fold of *Glass7* data set



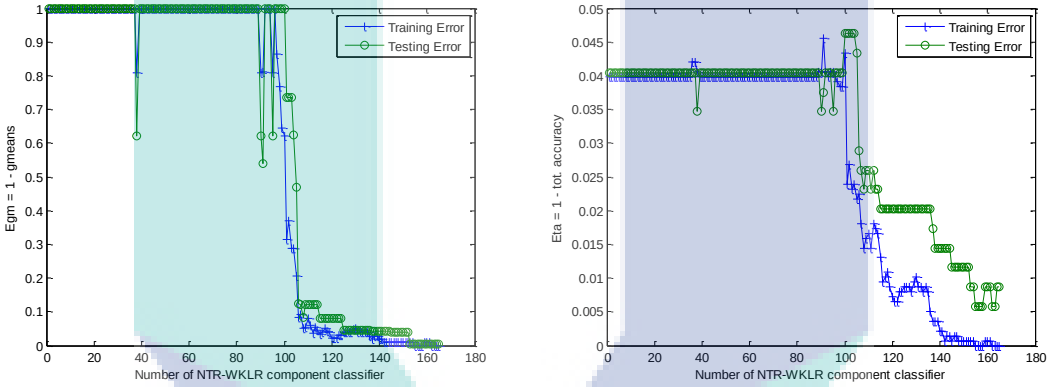
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.5: Error curve for AB-WKLR on first fold of *ImgSegment1* data set



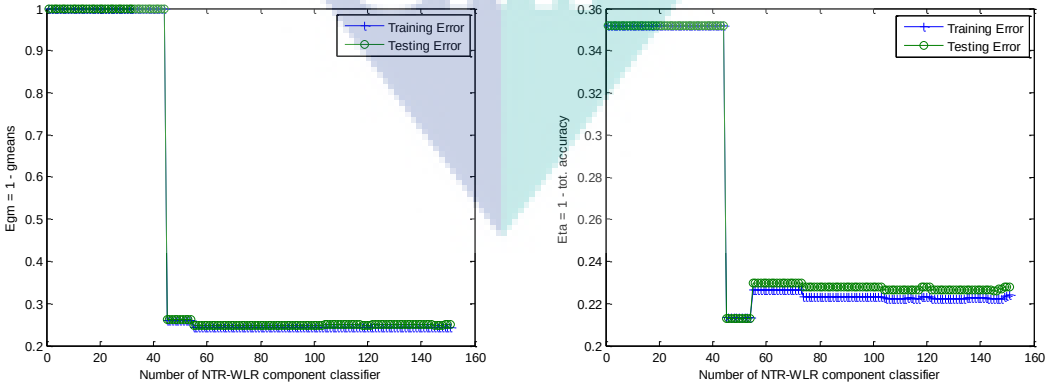
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.6: Error curve for AB-WKLR on first fold of *Balance2* data set



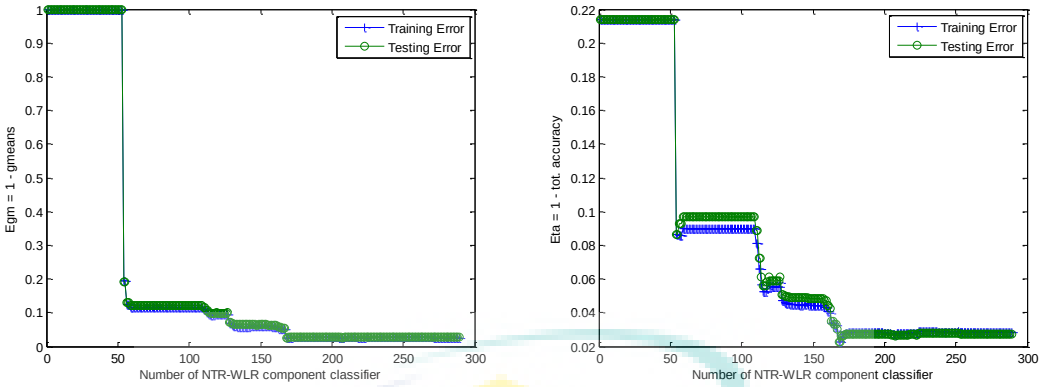
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.7: Error curve for AB-WKLR on first fold of *Car3* data set



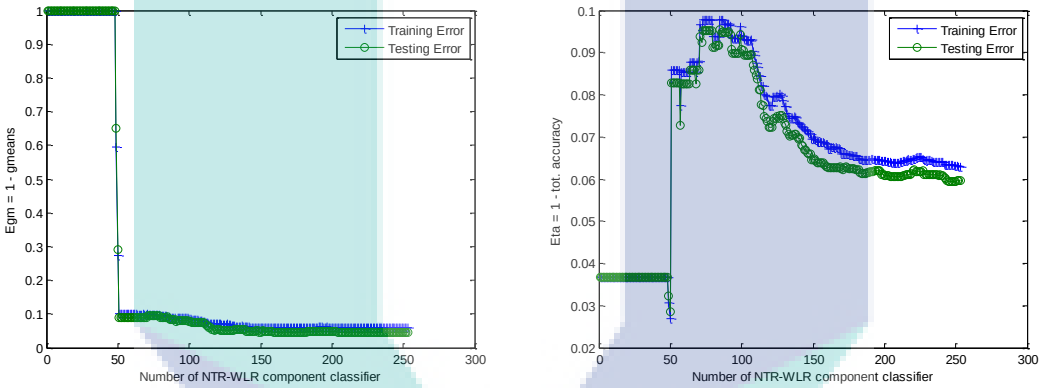
(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.8: Error curve for AB-WLR on first fold of *GammaImg* data set



(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.9: Error curve for AB-WLR on first fold of *Shuttle2to7* data set



(a) Error by *g-means* (b) Error by *total accuracy*

Figure 4.10: Error curve for AB-WLR on first fold of *LetterImg26* data set

It can be seen in Fig. 4.3 – Fig. 4.10 that training error by *g-means* metric decreases rapidly as NTR-WKLR or NTR-WLR component classifier was added. It means that training error by *g-means* metric drops fast to converge until final iteration of AB-WKLR and AB-WLR. This indicates the consistency with basic theoretical property of AdaBoost algorithm which uses error by *total accuracy* metric. Testing error follows the behaviour of training error similarly.

4.3.2 The effectiveness of adapted Modified AdaBoost in AB-WKLR and AB-WLR

The comparison results between AB-WKLR and NTR-KLR are summarized in Table 4.8 and Fig. 4.11. It can be seen that the accuracy and stability of AB-WKLR are

better than NTR-KLR on all five data sets, since AB-WKLR has better *g-means* and *standard deviations* of *g-means* ($S_{g\text{-}means}$).

Table 4.8: Summary of comparison results between AB-WKLR and NTR-KLR by maximizing *g-means* value with *5*-Fold SCV

No.	Name of Data set	*Sensitivity*		*Specificity*		*g-means* ($S_{g\text{-}means}$)	
		NTR-KLR	AB-WKLR	NTR-KLR	AB-WKLR	NTR-KLR	AB-WKLR
1	**Parkinson**	0.8800	**0.9246**	**0.9655**	0.94	0.9195 (0.0716)	**0.9320 (0.0433)**
2	**Glass7**	0.8667	**0.9622**	0.9946	**1**	0.9257 (0.0732)	**0.9808 (0.0157)**
3	**ImgSegment1**	0.9879	**0.9990**	0.9995	**1**	0.9937 (0.0065)	**0.9990 (0.0006)**
4	**Balance2**	0.02	**0.9045**	**1**	0.8356	0.0632 (0.1414)	**0. 8655 (0.0752)**
5	**Car3**	0.8275	**0.9940**	**0.9976**	0.9714	0.9073 (0.0509)	**0.9822 (0.0323)**

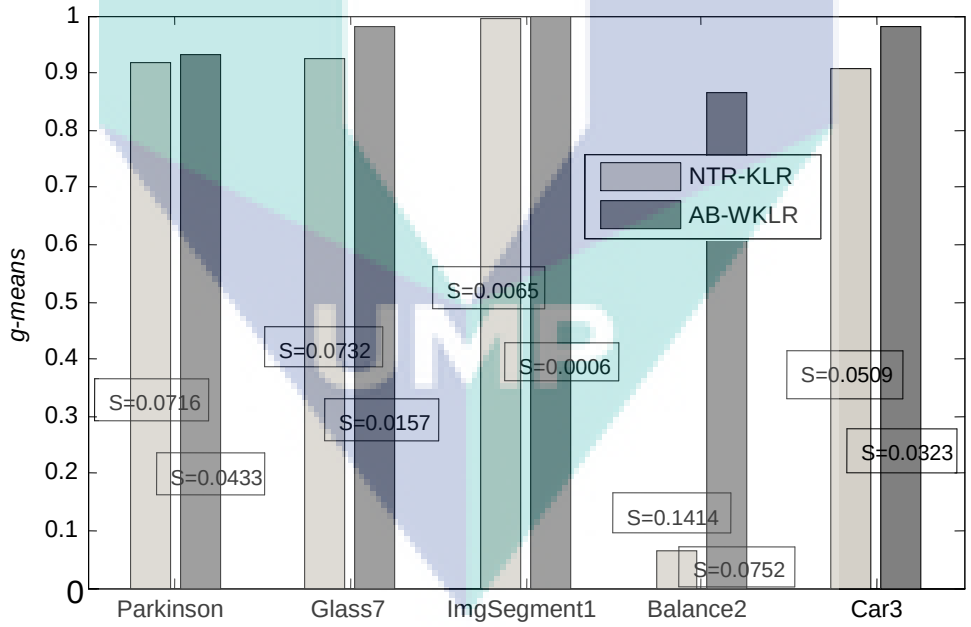


Figure 4.11: Comparison of *g-means* and $S_{g\text{-}means}$ between AB-WKLR and NTR-KLR

Table 4.9 and Fig. 4.12 provide the summary of comparison results between AB-WLR and NTR-LR. It can be observed that AB-WLR produces better *g-means* and

standard deviations of *g-means* ($S_{g\text{-}means}$), such that the accuracy and stability of AB-WLR is better than NTR-LR on all of three data sets.

Table 4.9: Summary of comparison results between AB-WLR and NTR-LR by maximizing *g-means* value with *5*-Fold SCV

No.	Name of Data set	*Sensitivity*		*Specificity*		*g-means* ($S_{g\text{-}means}$)	
		NTR-LR	AB-WLR	NTR-LR	AB-WLR	NTR-LR	AB-WLR
1	**GammaImg**	0.5908	**0.8123**	**0.8999**	0.7087	0.7291 (0.0071)	**0.7587 (0.0061)**
2	**Shuttle2to7**	0.9186	**0.9665**	**0.9862**	0.9799	0.9518 (0.0048)	**0.9732 (0.0025)**
3	**LetterImg26**	0.6757	**0.9417**	**0.9951**	0.9414	0.8199 (0.0115)	**0.9415 (0.0093)**

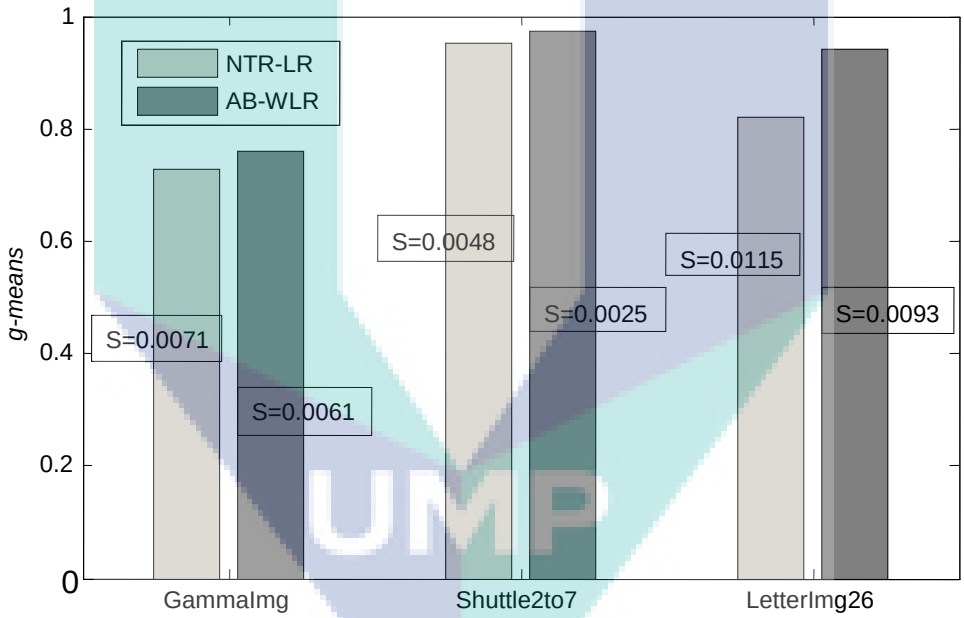


Figure 4.12: Comparison of *g-means* and $S_{g\text{-}means}$ between AB-WLR and NTR-LR

It can be observed also in Table 4.8 and Fig. 4.11 that AB-WKLR has better *sensitivity* value than NTR-KLR on all five imbalanced data sets, while AB-WLR performs better *sensitivity* value than NTR-LR on all three imbalanced data sets as indicated in Table 4.9 and Fig. 4.12. These results indicate the effectiveness of AB-WKLR and AB-WLR in solving the problem of minority class on imbalanced data sets, on which NTR-KLR and NTR-LR performed worse respectively. However, AB-WKLR

inevitably has worse *specificity* value on three of five imbalanced data sets, while AB-WLR produces worse *specificity* value on all three imbalanced data sets.

Table 4.10, Fig. 4.13 and Fig. 4.14 give the summary of AB-WKLR improvements to NTR-KLR. It can be observed that improvements on both error by *g-means* and *standard deviation* of *g-means* with *5*-Fold SCV could be as high as more than 90%. Hence, AB-WKLR is more accurate and more stable classifier than NTR-KLR in classifying on small-to-medium size of imbalanced data sets. Furthermore, AB-WKLR is useful specifically for data sets which have degrees of imbalance within 6 to 24, as indicated in Table 4.10.

Table 4.10: Summary of AB-WKLR improvements to NTR-KLR in reducing error by *g-means* and standard deviation of *g-means*

Name Of Data Set	Degree of Imbalanced	Average 5-Fold SCV Error (1 - *gmeans*)		Improvement on Reducing Error by *g-means* (%)	Standard Deviation of *g-means* ($S_{g\text{-}means}$)		Improvement on reducing $S_{g\text{-}means}$ (%)
		NTR-KLR	AB-WKLR		NTR-KLR	AB-WKLR	
Parkinson	**3.06**	0.0805	0.068	15.5280	0.0716	0.0433	39.5251
Glass7	**6.38**	0.0743	0.0192	74.1588	0.0732	0.0157	78.5519
ImgSegment1	**6**	0.0063	0.001	84.127	0.0065	0.0006	**90.7692**
Balance2	**12.52**	0.9368	0.1345	**85.6426**	0.1414	0.0752	46.8175
Car3	**24.04**	0.0927	0.0178	80.7983	0.0509	0.0323	36.5422

Largest contribution of AB-WKLR in improving the accuracy of NTR-KLR was shown on *Balanced2* data set, while largest contribution of AB-WKLR in improving the stability was shown on *ImgSegment1* data set as observed in Table 4.10, Fig. 4.13 and Fig. 4.14.

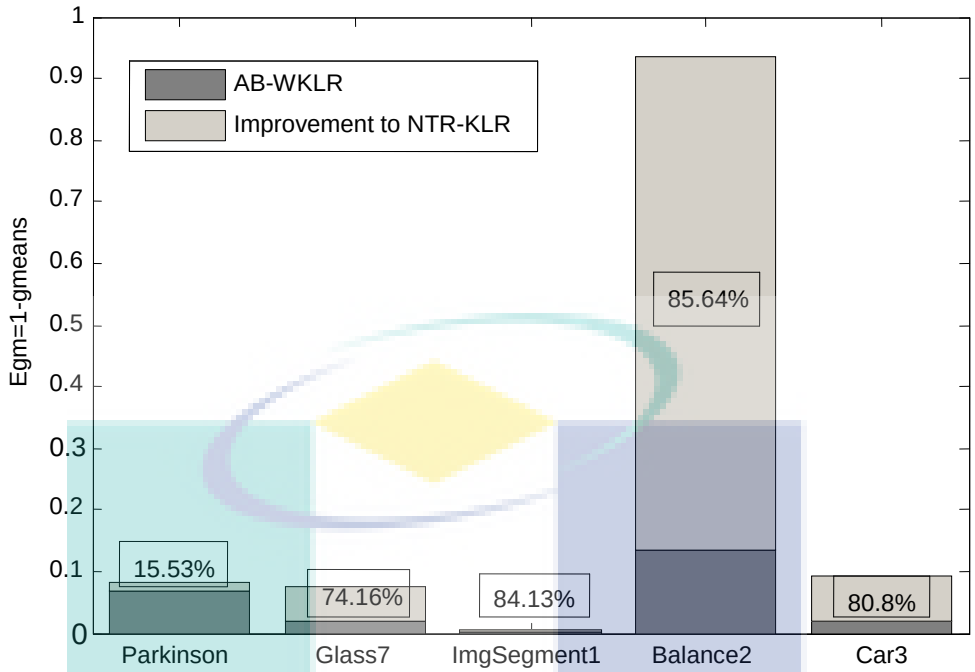


Figure 4.13: Improvements of AB-WKLR to NTR-KLR in reducing error by *g-means*

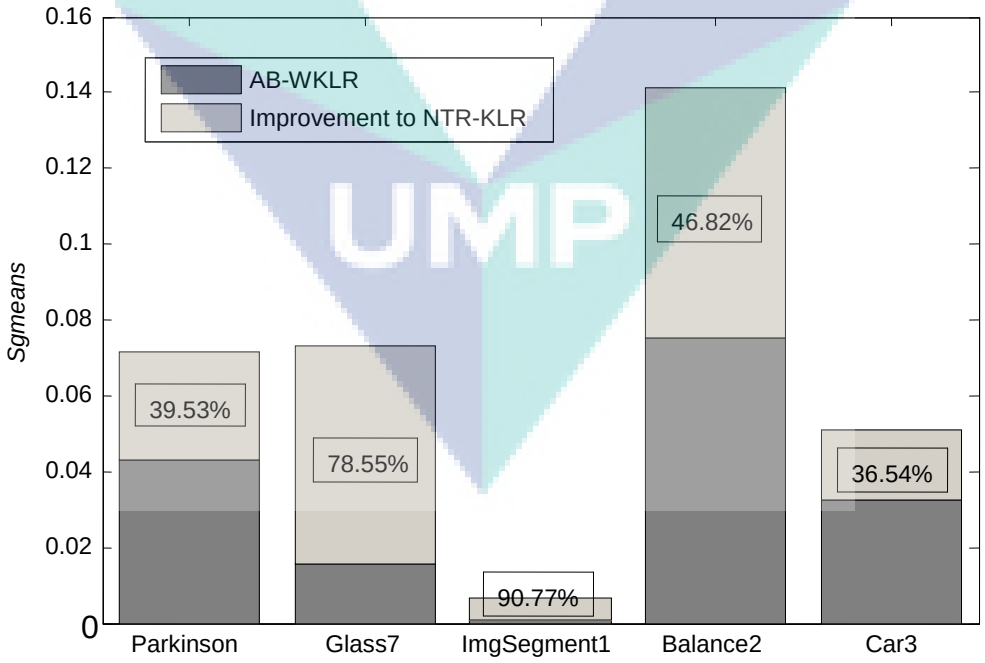


Figure 4.14: Improvements of AB-WKLR to NTR-KLR in reducing *standard deviation* of *g-means*

The summary of AB-WLR improvements to NTR-LR is provided in Table 4.11, Fig. 4.15 and Fig. 4.16. As compared to NTR-LR, it can be seen that AB-WLR is more accurate and more stable classifier in classifying on large size of imbalanced data sets, because its improvements on both error by *g-means* and standard deviation of *g-means* with 5-Fold SCV could be achieved more than 60%. Similar to AB-WKLR, AB-WLR is useful specifically for data sets which have high degrees of imbalance i.e. 3.6 and 26 in this research, as shown in Table 4.11.

Table 4.11: Summary of AB-WLR improvements to NTR-LR in reducing error by *g-means* and standard deviation of *g-means*

Name Of Data Set	Degree of Imbalanced	Average Error With 5-fold SCV (1 – *gmeans*)		Improvement on Reducing Error by *g-means* (%)	Standard Deviation of *g-means* ($S_{g\text{-}means}$)		Improvement on reducing $S_{g\text{-}means}$ (%)
		NTR-LR	AB-WLR		NTR-LR	AB-WLR	
GammaImg	**1.84**	0.2709	0.2413	10.9265	0.0071	0.0061	14.0845
Shuttle2to7	**3.67**	0.0482	0.0268	44.3983	0.0048	0.0025	**47.9167**
LetterImg26	**26.25**	0.1801	0.0585	**67.5180**	0.0115	0.0093	19.1304

Table 4.10, Fig. 4.13 and Fig. 4.14 indicate that AB-WLR has largest contribution in improving the accuracy of NTR-LR on *LetterImg26* data set, while largest contribution of AB-WKLR in improving the stability was showed in *Shutte2to7* data set.

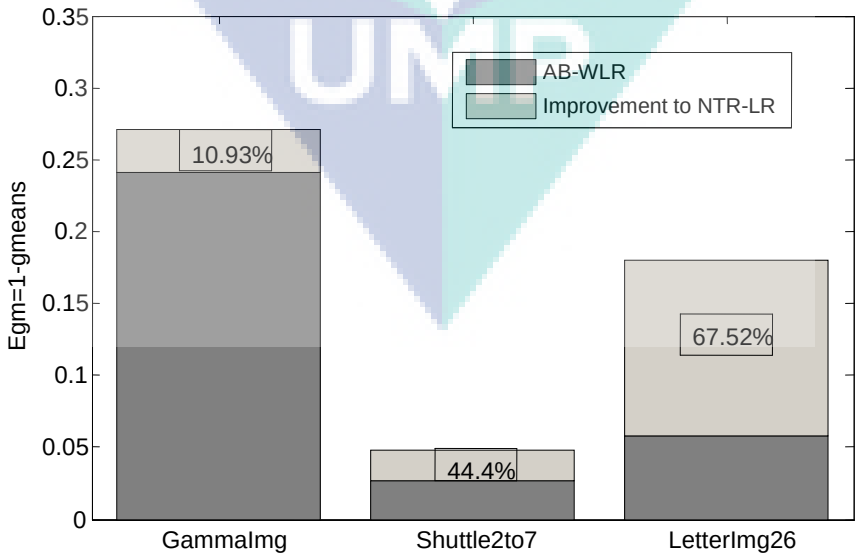


Figure 4.15: Improvement of AB-WLR to NTR-LR in reducing error by *g-means*

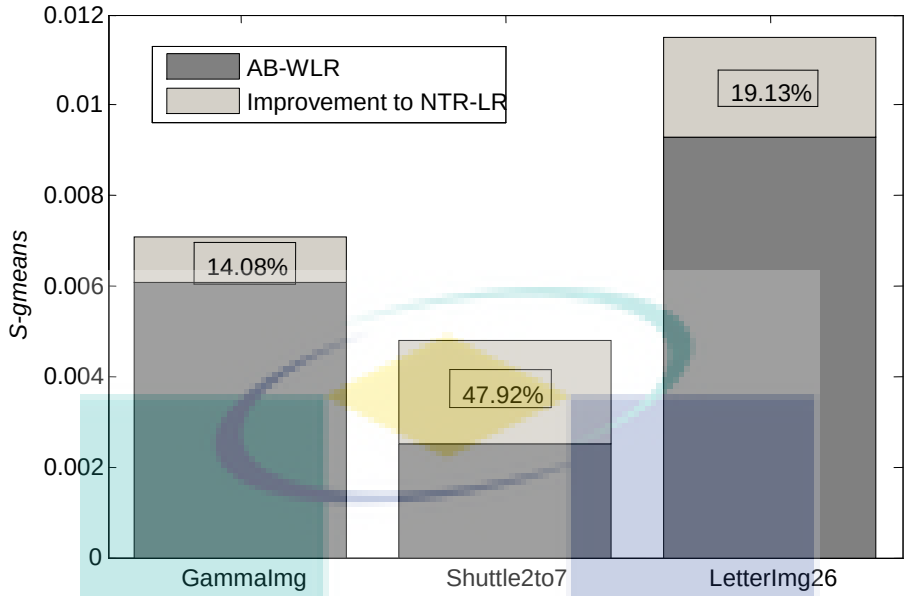


Figure 4.16: Improvement of AB-WLR to NTR-LR in reducing *standard deviation* of *g-means*

In general, the effectiveness of adapted Modified Adaboot methods in AB-WKLR and AB-WLR algorithm respectively has been tested statistically, as presented in Table 4.12. Statistical significance was conducted by comparing the accuracy performance of AB-WKLR to NTR-KLR and AB-WLR to NTR-LR using Paired-Samples *t*-test (α=0.05) (Montgomery, 1991).

Table 4.12: Summary of statistical significances: AB-WKLR vs NTR-KLR and AB-WLR vs NTR-LR

Data Set	Average of *g-means* With 5-fold SCV ($S_{g\text{-}means}$)		Paired-Samples *t*-test (α=0.05)	
	NTR-KLR	**AB-WKLR**	***p*-value**	**Statistical Significance**
Parkinson	0.9195 (0.0716)	**0.9320** (**0.0433**)	.599	No
Glass7	0.9257 (0.0732)	**0.9808** (**0.0157**)	.223	No
ImgSegment1	0.9937 (0.0065)	**0.9990** (**0.0006**)	.124	No
Balance2	0.0632 (0.1414)	**0. 8655** (**0.0752**)	0	Yes
Car3	0.9073 (0.0509)	**0.9822** (**0.0323**)	.035	Yes

Table 4.12: Continued

Data Set	Average of *g-means* With 5-fold SCV ($S_{g\text{-}means}$)		Paired-Samples *t*-test (α=0.05)	
	NTR-KLR	AB-WKLR	*p*-value	Statistical Significance
GammaImg	0.7291	**0.7587**	0.01	Yes
Shuttle2to7	0.9518	**0.9732**	0	Yes
LetterImg26	0.8199	**0.9415**	0	Yes
TOTAL				(5 Yes, 3 No)

The results of statistical tests in Table 4.12 describe that the use of Modified AdaBoost methods in AB-WKLR and AB-WLR algorithm respectively is effective significantly on 5 data sets out of total 8 data sets, since their p-value < 0.05 (α).

4.3.3 Performances Comparison of proposed AB-WKLR and proposed AB-WLR to AdaBoostSVM

Table 4.13 displays the summary of performances comparison between the proposed algorithms with their related work i.e. AdaBoostSVM. It can be observed that AB-WKLR and AdaBoostSVM have comparable value of *g-means* and AUC on three imbalanced data sets. AB-WLR and AdaBoostSVM also performs similarly on *LetterImg26* data set.

Although AB-WKLR performs slightly better than AdaBoostSVM in classifying three imbalanced data sets, both classifiers have comparable performance in general. In addition, NTR-KLR component classifier used in AB-WKLR has simple solution of unconstrained optimization problem. On the other hand, with the use of unconstrained optimization problem and without the use of Kernel function, AB-WLR has simpler solution than AdaBoostSVM in performing comparable performance on LetterImg26 data set.

Table 4.13: Summary of comparison: between proposed algorithms and AdaBoostSVM

Data Set	*g-means*		AUC	
	Proposed AB-WKLR	**AdaBoostSVM (Li et al., 2008)**	**Proposed AB-WKLR**	**AdaBoostSVM (Li et al., 2008)**
Glass7	0.9808*	0.885	1*	0.963
ImgSegment1	0.9990*	0.965	1*	0.991
Car3	0.9822*	0.975	1*	0.997
	Proposed AB-WLR	**AdaBoostSVM (Li et al., 2008)**	**Proposed AB-WLR**	**AdaBoostSVM (Li et al., 2008)**
LetterImg26	0.9415	0.945*	0.9968*	0.985

4.4 SUMMARY

This chapter has reported and analyzed the performances of proposed classification algorithms i.e. general classification algorithms (NTR-KLR and NTR-LR) and imbalanced classification algorithms (AB-WKLR and AB-WLR) based on several numerical experiments. The effectiveness of the proposed classification algorithms and the method used was evaluated. The numerical convergence, accuracy and stability of proposed classification algorithms were provided previously. Furthermore, the limitation of proposed general classification algorithms was also determined, in relation to the development of proposed imbalanced classification algorithms.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 INTRODUCTION

The conclusion which related to Research Objective of the thesis is given in this chapter, based on numerical results and discussion in Chapter 4. The recommendations for further work are described then in this chapter.

5.2 CONCLUSIONS

In general, this thesis has developed proposed general and imbalanced classification algorithm. The numerical converge of the proposed classification algorithms have been provided also respectively, along with their accuracy and stability. Proposed general classification algorithms respectively are NTR-KLR and NTR-LR, while proposed imbalanced classification algorithms respectively are AB-WKLR and AB-WLR.

5.2.1 NTR-KLR and NTR-LR

Numerical results have shown that the use of Truncated Newton method in NTR-KLR and NTR-LR algorithm is effective in handling the numerical problem of KLR-NR and RLR-NR respectively on the huge matrix of *linear system of Newton-Raphson update rule* i.e. the training time and the singularity problem. These results can be seen as further explanation on the success of Truncated Newton method in TR-KLR (Maalouf et al., 2010) and TR Iteratively Re-weighted Least Square (TR-IRLS) (Komarek and Moore, 2005) algorithm respectively, because of the equivalence of

iterative method used by these algorithms. Moreover, only with the use of unconstrained optimization problem which has simple solution, NTR-KLR and NTR-LR respectively have comparable performance with SVM which is determined as state-of-the-art classifier in Kernel methodology and Data Mining community. However, numerical results have confirmed that the performances accuracy of both proposed general classifiers is limited when applied on imbalanced data sets, specifically in classifying the minority class.

5.2.2 AB-WKLR and AB-WLR

Numerical results have demonstrated that the use of adapted Modified AdaBoost methods in AB-WKLR and AB-WLR algorithm respectively has performed significantly effective performance in improving the accuracy and stability performances of general classifiers i.e. NTR-KLR and NTR-LR respectively, on imbalanced data sets. The improvements on both error by *g-means* and *standard deviation* of *g-means* with 5-Fold SCV could be achieved as high as more than 60 to 90%.

Furthermore, numerical results have shown that AB-WKLR and AB-WLR respectively have comparable performances with AdaBoostSVM in classifying imbalanced data sets. In addition, both proposed imbalanced classification algorithms employ the weighted version of NTR-KLR (NTR-WKLR) and the weighted version of NTR-LR (NTR-WLR) component classifier respectively which have simple solution of unconstrained weighted optimization problem.

5.3 RECOMMENDATIONS

A number of recommendations for future works may enhance the promising results which have been demonstrated in this thesis. Those are outlined as follows:

a. Combine or replace the Newton-Raphson method with other method, as the outer algorithm of Truncated Newton method in approximating the solution of Newton-Raphson update rule to search the MLE for KLR and RLR respectively.
b. Extend the algorithm of NTR-KLR, NTR-LR, AB-WKLR and AB-WLR respectively for multi-class data sets problem.

c. Instead of Grid Search with *k*-Fold SCV, explore the use of other model selection methods, in order to obtain shorter time in process of model selection.

REFERENCES

Akbani, R., Kwek, S. and Japkowicz, N. 2004. Applying support vector machines to imbalanced datasets. *Lecture Notes in Computer Science.* **3201**: 39–50.

Bai, S.B., Wang, J., Zhang, F.Y., Pozdnoukhov, A. and Kanevski, M., 2008. Prediction of landslide susceptibility using logistic regression: a case study in Bailongjiang river basin, China. *Proceedings of The Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 4, pp. 647-651.

Batista, G., Prati, R. C. and Monard, M. C. 2004. A study of the behavior of several methods for balancing machine learning training data. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-SIGKDD Explorations*, *6*(1), 20–29.

Bauer, E. and Kohavi, R. An Empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*.

Boyd S.and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge: University Press.

Breiman, L. 1996. Bagging Predictor. *Machine Learning*. **24**: 123–140.

Busser, B., Daelemans, W. and Bosch, A., 1999. Machine learning of word pronunciation: the case against abstraction. *Proceedings of the Sixth European Conference on Speech Communication and Technology*, Eurospeech99, pp. 2123-2126.

Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*. **16**: 321-357.

Chawla, N. V., Japkowicz, N. and Kolcz, A. 2004. Editorial: special issue on learning from imbalanced data sets. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-SIGKDD Explorations*, *6*(1), 1–6.

Cristianini, N., Kandola, J., Elisseeff, A. and Shawe-Taylor, J. 2006. On kernel target alignment. *Studies on Fuzziness and Soft Computing.* 194: 205-256.

Christmann, A., Luebke, K., Marcos, M.G. and Ruping, S. 2005. *Determination of hyper-parameters for kernel based classification and regression*. Technical Reports-38, SFB 475, University of Dortmund.

Cawley G.C. and Talbot N.L.C. 2005. The evidence framework applied to sparse kernel logistic regression. *Neurocomputing*. **64**:119-135.

Cawley G.C. and Talbot N.L.C. 2008. Efficient Approximate Leave-One-Out Cross-Validation for Kernel Logistic Regression. *Machine Learning*. **71**: 243–264.

Chan, P.K. and Stolfo, S.J., 1998. Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. AAAI Press, pp. 164_168.

Del Castillo, M.D. and Serrano, J.I., 2004. A multistrategy approach for digital text categorization from imbalanced documents. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-ACMSIGKDD Explorations: Special Issue on Learning from Imbalanced Datasets*, pp. 39–70.

Diamantidis, N.A., Karlis, D. and Giakoumakis, E.A. 2000. Unsupervised stratification of cross-validation for accuracy estimation. *Artificial Intelligence*. 1**16**: 1–16.

Dietterich, T.G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning. **40** (2): 139–157.

Dreitsel, S and Machado, L.O. 2002. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*. **35**: 352-359.

Eeckhaut, M.V.D., Vanwalleghem, T., Poesen, J., Govers, G., Verstraeten, G. and Vandekerckhove, L., 2006. Prediction of landslide susceptibility using rare events logistic regression: a case-study in the Flemish Ardennes (Belgium). *Geomorphology*. **76**(3-4): 392-410.

Fawcett, T. 2004. *ROC graphs : Notes and practical considerations for researchers*. Technical report, HP Laboratories, MS 1143, 1501 Page Mill Road, Palo Alto CA 94304, USA.

Fawcett, T. and Provost, F., 1997. Adaptive fraud detection. *Data Mining and Knowledge Discovery*. **1** (3): 291–316.

Frank, A. and Asucion, A. 2010. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml/datasets.html?format=&task=cla&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table (2008)

Freund, Y. 1993. *Data filtering and distribution modelling algorithms for machine learning*. PhD thesis. University of California, Santa Cruz.

Freund, Y. 1995. Boosting a weak learning algorithm by majority. Information and Computation. **121**(2): 256-285.

Freund, Y. and Schapire, R. 1997. A Decision Theoritic Generalization of On-Line Learning and a application to Boosting. *Journal of Computer and System Sciences.* **55**(1): 119-139

Friedman, J., Hastie, T. and Tibshirani, R. 2000. Additive Logistic Regression: a Statistical view of Boosting. *Annals of Statistics*. **28**: 337-407.

Garthwaite, P., Jolliffe, I. and Byron, J. 2002. Statistical Inference. Oxford: University Press.

Geman, S., Bienenstock, E. and Doursat, R. 1992. Neural networks and the bias/variance dilemma. *Neural Computation.* **4**(1): 1–58.

Gilbert, J. R. (2006) http://www.cs.ucsb.edu/~gilbert/cs140Win2009/cgproject (2010).

Guo, H. and Viktor, H.L., 2004. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Minin-g ACMSIGKDD Explorations: Special Issue on Learning from Imbalanced Datasets*, pp. 30–39.

Han, J. and Kamber, M. 2006. *Data Mining: Concepts and Techniques*. 2nd ed. Morgan Kaufmann Publishers.

Hand, D.J. 1998. Statistics and More. *The American Statistician*. **5**(2): 112-118.

Hastie, T., Tibshirani, R. and Friedman, J. 2001. *The Element of Statistical Learning*. Canada: Springer-Verlag.

Hoerl, A. and Kennard**,** R. 1970. Ridge Regression: Biased Estimation for nonorthogonal problems. *Technometrics*. **12**: 55 – 67.

Hogg, Robert V.and Craig, A. T.1994. *Introduction to Mathematical Statistics.* 5th Edition. Prentice-Hall.

Hosmer, D.W. and Lemeshow, S., 2000. *Applied Logistic Regression*. Second edition. Wiley.

Hsu, C.W., Chang, C.C., and Lin, C.J. 2003 (Last updated: April 15, 2010). A practical guide to support vector classification. Technical report. Department of Computer Science, National Taiwan University. www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (2008)

Huang, C.L., Chen, M.C., Wang, C.J. 2007. Credit Scoring with a Data Mining Approach based on Support Vector Machines. *Expert System with Applications*. **33**: 847-856.

Huang, Y., Erdogmus, D, Santosh, M. and Pavel, M. 2005. Boosting Linear Logistic Regression for Single Trial ERP Detection in Rapid Serial Visual Presentation Tasks. *Proceedings of the 28th IEEE EMBS Annual International Conferenc*e.

Iyer, R.D. 1999. *An Efficient Boosting Algorithm for Combining Preferences*. Master thesis, Carnegie Mellon University.

Japkowicz. N. 2000. Learning from imbalanced data sets: A comparison of various strategies, Learning from imbalanced data sets. The AAAI Workshop 10-15. Menlo Park, CA: AAAI Press. Technical Report WS-00-05.

Joshi, M.V., Kumar, V. and Agarwal, R.C. 2001. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *Proceedings of the IEEE International Conference on Data Mining*. pp. 257–264.

Joshi, M.V., Agarwal, R.C. and Kumar, V. 2002: Predicting rare classes: can boosting make any weak learner strong? *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 297–306.

Karsmaker, P., Pelckmans, K. and Suykens, J.A.K. 2007. Multi-class Kernel Logistic Regression : a fixed sized implementation. *Proceedings of International Joint Conference on Neural Networks*.

Katz, M., Krüger, S.E., Schafföner, M., Andeli´c, E., Wendemuth, A. 2006. Speaker identification and verification using support vector machines and sparse Kernel logistic regression. *Lecture Notes in Computer Science*, **4153**: 176-184.

Keerthi, S.S., Duan, K.B., Shevade, S.K. and Poo, A.N. 2005. A fast dual algorithm for kernel logistic regression. Machine Learning. **61**(1-3): 151–165.

King, G. and Zeng, L. 2001a. Explaining rare events in international relations. *International Organization*. **55**(3): 693-715.

King, G. and Zeng, L. 2001b. Improving forecast of state failure. *World Politics.* **53**(4): 623-658.

King, G., Zeng, L., 2001c. Logistic regression in rare events data. *Political Analysis.* **9**: 137-163.

Komarek, P. 2004. *Logistic regression for data mining and high-dimensional classification*. Ph.D. thesis, Carnegie Mellon University.

Komarek P. and Moore A. 2005. *Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity*. Tech. report, TR-05-27, Robotics Institute, Carnegie Mellon University.

Kubat, M. and Matwin, S. 1997. Addressing the curse of imbalanced training sets: one-sided selection. *Proceedings of the Fourteenth International Conference on Machine Learning*. pp. 179–186.

Kubat, M., Holte, R.C. and Matwin, S., 1998. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, pp. 195-215.

Lai, K.K., Yu, L., Zhou, L.G. and Wang. S.Y. 2006. Credit risk evaluation with least square support vector machine, *Lecture Notes in Artificial Intelligence*. **4062**: 490-495.

Lewis, J.M., Lakshmivarahan, S. and Dhall, S. 2006. *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge: University Press.

Lee, A. & Silvapulle, M. 1988. Ridge estimation in logistic regression, *Communications in Statistics, Simulation and Computation.* **17**: 1231-1257.

Le Cessie, S. & Van Houwelingen, J. 1992. Ridge estimators in logistic regression, *Applied Statistic.* **41**: 191-201.

Li, X., Wang, L. and Sung, E. 2005. A study of AdaBoost with SVM based weak learners. *International Joint Conference on Neural Networks*, pp.196-201.

Li, X., Wang, L. and Sung, E. 2008. AdaBoost with SVM-based component classifiers. *Engineering Applications of Artificial Intelligence*. **21**: 785–795

Lin, C., Weng, R.C. and Keerthi, S.S. 2008. Trust Region Newton Method for Large-scale Logistic Regression. *Journal of Machine Learning Research*. **9**: 627-650.

Maalouf, M., 2009. *Robust weighted kernel logistic regression in imbalanced data and rare event.* Ph. D Thesis. Oklahoma University.

Maloof, M.A. 2003. Learning when data sets are imbalanced and when costs are unequal and unknown. *Proceedings of the International of Conference on Machine Learning (ICML'2003).*

Maalouf, M. and Trafalis, T.B. 2011. Robust weighted kernel logistic regression in imbalanced and rare events data. *Computational Statistics & Data Analysis*. **55**(1)**:** 168-183.

Maalouf, M., Trafalis, T.B., and Adrianto, A. 2010. Kernel logistic regression using truncated Newton method. *Computational Management Science*. **28**: 1-14.

Malouf, R. 2002. A comparison of algorithms for maximum etropy parameter estimation. *Proceedings of Conference on Natural Language Learning*, pp. 1-7.

McCullagh, P., Nelder, J.A. 1989. *Generalized Linear Models*, vol. 37: Monographs on Statistics and Applied Probability. second ed.. London: Chapman & Hall.

Meir, R. and Ratch, G. 2003. An Introduction to Boosting and Leveraging. *Advanced Lectures on Machine Learning-Lecture Note in Artificial Intelegence*. **2600**: 118-183.

Mercer, J. 1909. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London A.* **209**: 415–446.

Minka, T.P. 2003 (revised 2007). A comparison of numerical optimizers for logistic regression. Tech.Report, Depaartment of Statistics, Carnegie Mellon University. http://research.microsoft.com/en-us/um/people/minka/papers/logreg/minka-logreg.pdf (2010)

Montgomery, D.C. (1991) Design and analysis of experiment. 3rd ed. Singapore: John Wiley & Sons.

Nabney, I.T. 1999. Efficient training of RBF networks for classification. *Proceedings of the Nineth International Conference on Artificial Neural Networks*, **1**: 210–215.

Nash, S. G. 2000. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*. **124**(1-2):45-59.

Nash, S. G. and Sofer, A. 1996. Linear and non-Linear Programming. Mc-Graw-Hill.

Nishida, K. and Kurita, T. 2006. Kernel Feature Selection to Improve Generalization Performance of Boosting Classifiers. *Proceedings of The 2006 International Conference on Image Processing, Computer Vision, & Pattern Recognition.*

Nocedal, J. and Wright, S. 1999. *Numerical Optimization*. New York: Springer.

Nugroho, A.S., Witarto, A.B. and Handoko, D. 2003. Application of Support Vector Machine in Bioinformatic. *Proceeding of Indonesian Scientific Meeting in Central Japan*.

Opitz, D., Maclin, R., 1999. Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*. **11**: 169–198.

Owen, A. 2007. Infinitely imbalanced logistic regression. *Journal of Machine Learning Research*. **8**: 761–773.

Oza, N. C. 2001. *Online ensemble learning*. Ph.D.thesis. Department of Electrical Engineering and Computer Science. University of California, Berkeley.

Oza, N. C. and Russell, S. 2001. Online bagging and boosting. *Artificial Intelligence and Statistics.* pp. 105-112.

Park, C.Y., Koo, J.Y., Kim, P.T. and Lee, J.W. 2008. Stepwise feature selection using generalized logistic loss. *Computational Statistics and Data Analysis*. **52**: 3709–3718.

Park, M.Y. and Hastie, T., 2008. Penalized logistic regression for detecting gene interactions. *Biostatistics* **9**(1): 30–50.

Patra, S., Shanker, K. and Kundu, D. 2008. Sparse maximum margin logistic regression for credit scoring. *Eighth IEEE International Conference on Data Mining.*

Pearson, R., Goney, G. and Shwaber, J. 2003. Imbalanced clustering for microarray time-series. *Proceedings of the International of Conference on Machine Learning* (*ICML'2003).*

Polat, K. and Gunes, S. 2007. Breast Cancer Diagnosis using Least Square Support Vector Machine. *Digital Signal Processing.* **17**: 694-701.

Quigley, J., Bedford, T. and Walls, L., 2007. Estimating rate of occurrence of rare events with empirical Bayes: a railway application. *Reliability Engineering & System Safety*. **92**(5): 619-627.

Rahimi, Ali. 2006. The similarity between the logit loss and the SVM loss. citeseerx.ist.psu.edu/viewdoc/download?doi=*10.1.1.120.3097*

Rennie, J.D.M. 2003. Newton's method. http://people.csail.mit.edu/jrennie/writing/newton.pdf (2010)

Rennie, J.D.M. 2005. Maximum margin logistic regression. http://people.csail.mit.edu/jrennie/writing/mmlr.pdf (2010)

Ridgeway, G.D., Madigan, D., Richardson, T. and O'Kane, J. 1998. Interpretable Boosted Naïve Bayes Classification. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 101-104.

Roth, V. 2001. Pobabilistic Discriminative Kernel Classifiers for Multi-Class Problems. *Lecture Notes in Computer Science*, **2191**: 246 – 253.

Schapire, R.E. 1990. The strength of weak learnability. *Machine Learning*. 5(2): 197-227.

Schapire, R.E. 1992. *The design and analysis of efficient learning algorithms.* MIT Press.

Schapire, R.E., Freund, Y., Batlett, P. and Lee, W.S., 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*. **26** (5): 1651-1686.

Schapire, R.E. and Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*. **37** (3): 297–336.

Shawe-Taylor, J. and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis.* Cambridge: University Press.

Scholkopf, B. and Smola, A.J. 2002. *Learning With Kernel—Support Vector Machines Regularization Optimization and Beyond*. Cambridge: MIT Press.

Scholkopf, B., Herbrich, R. and Smola, A.J. 2002. A Generalized Representer Theorem. *Lecture Note in Computer Scince* . **2375**:416-426.

Shin, H.W. and Sohn, S.Y. 2005. Selected tree classifier combination based on both accuracy and error diversity. *Pattern Recognition*. **38**: 191–197.

Shewchuk, J. R. 1994. An introduction to the conjugate gradient method without the Agonizing Pain. Technical Report CS-94-125, Carnegie Mellon University, Pittsburgh.

Suykens, J. A. K. and Vandewalle. J. 1999. Least squares support vector machine classifiers. *Neural Processing Letters*. **9**(3):293-300.

Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., and Vanderwalle, J. 2002. *Least Squares Support Vector Machines*, World Scientific Publishing.

Swartout, W. 1983. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence*. **21**: 285-325.

Tan, P. N, Steinbach, M. and Kumar, V. 2005. *Introduction to Data Mining.* USA: Addison-Wesley.

Tang, Y., Zhang, Y.Q., Chawla, N.V. and Krasser, S. 2009. SVMs modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernectics*. Part B. **39**(1): 281–288.

Tang, Y., Jin, B. and Zhang, Y.Q. 2005. Granular support vector machines with association rules mining for protein homology prediction. *Artificial Intelligence in Medicine*. **35**(1-2): 121–134

Tenenhaus A., Giron A., Viennetc, E., Bérab, M., Saportad, G. and Fertil, B. .2007. Kernel logistic PLS: A tool for supervised nonlinear dimensionality reduction and binary classification. *Computational Statistics & Data Analysis.* **51**:4083 – 4100

Trafalis, T.B., Ince, H. and Richman, M.B., 2003. Tornado detection with support vector machines. *International Conference on Computational Science.* pp. 289-298.

Tsoucas, P., 1992. Rare events in series of queues. *Journal of Applied Probability*. **29**: 168-175.

Valentini G. and Dietterich,T.G. 2004. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. *Journal of Machine Learning Research*. **5**: 725–775.

Van-Hulse, J., Khosghoftaar, M.T. and Napolitano, A. 2007. Experimental perspectives on learning from imbalanced data. *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pp. 935–942.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

Vapnik V. 2000. *The nature of statistical learning theory*. 2nd ed. NewYork: Springer.

Veropoulos, K., Campbell, C. and Cristianini, N., 1999. Controlling the sensitivity of support vector machines. *Proceedings of the International Joint Conference on Artificial Intelligent*, pp. 55–60.

Wang, S., Jiang, W., and Tsui, K.L. 2008. Adjusted support vector machines based on a new loss function. *Ann Oper Res*. **174**(1): 83-101.

Wang, Y. and Lin, C.D. 2007. Learning by Bagging and Adaboost based on Support Vector Machine. *Proceedings of* the *5th IEEE International Conference on Industrial Informatics*. pp. 663-668

Weiss, G. M. 2004. Mining with rarity: a unifying framework. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-SIGKDD Explorations*, **6**(1): 7–19.

Weiss, G.M. and Hirsh, H., 2000. Learning to predict extremely rare events. *Proceedings of the AAAI'2000*, pp. 64-68.

Wickramaratna, J., Holden, S.B. and Buxton, B.F. 2001. Performance degradation in boosting. *Proceedings of the Second International Workshop on Multiple Classifier Systems*. pp. 11–21.

West, D. 2000. Neural network credit scoring models. *Computer and Operations Research*. **27**: 1131-1152.

Wu, G. and Chang, E.Y. 2005. Kba: kernel boundary alignment considering imbalanced data distribution. *IEEE Transactions on Knowledge and Data Engineering*. **17** (6): 786–795.

Yan, R., Liu, Y., Jin, R., Hauptmann, A. 2003. On predicting rare classes with svm ensembles in scene classification. Proceedings of the *IEEE International Conference on Acoustic, Speech and Signal Processing*, pp. 21–24.

Yu, L., Lai, K.K. and Wang, S.Y.2006. Credit risk assesment with least square fuzzy support vector machine. Proceedings of the *Sixth IEEE International Conference on Data Mining - Workshop (ICDMW'06).*

Zhang, J.. Iterative methods for optimization. http://learning.stat.purdue.edu/wiki/_media/courses/sp2011/598g/minimization.pdf (2010)

Zhang, J., Jin, R., Yang, Y. and Hauptmann, A.G., 2003. Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. *Proceedings of the Twentieth International Conference on Machine Learning*, ICML-2003.

Zhang, J. and Yang, Y. 2003. Robustness of Regularized Linear Classification Methods in Text Categorization. *Proceedings of SIGIR 2003: The Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Zhang, T. and Oles, F.J. 2001. Text categorization based on regularized linear classification methods. *Information Retrieval*. **4**:5-31.

Zhou, M. and Wei, H. 2009. Constructing Weak Learner and Performance Evaluation in AdaBoost. *Proceeding of the International Conference on Computational Intelligence and Software Engineering* (*CiSE*), pp. 1-4.

Zhu, J.2003. *Flexible Statistical Modelling*. PhD Thesis, Stanford University.

Zhu, J. and Hastie, T. 2004. Classification of Gene microarrays by Penalized Logistic Regression. *Biostatistics* **5**(3): 427-443.

Zhu J. and Hastie T. 2005. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*. **14**(1):185-205.

APPENDIX A

LIST OF PUBLICATIONS

A.1. Journal

1. Rahayu, S. P., Purnami, S.W., Embong, A., Jasni Mohamad Zain. Kernel Logistic Regression-Linear for Leukemia Classification using High Dimensional Data. Jurnal Ilmiah Teknologi Informasi (JUTI), Terakreditasi B. 7(3): 143-148. 2009
2. Rahayu, S. P., Jasni Mohamad Zain, Embong, A., Purnami, S.W. Logistic Regression Methods with Truncated Newton method. *Applied Mathematics and Computation, Elsevier. (submitted)*
3. Rahayu, S. P., Jasni Mohamad Zain, Embong, A., Logistic Regression Methods for Classification of Imbalanced Data Sets, *Journal of Computational Statistics and Data Analysis, Elsevier. (submitted)*

A.2 International Conference

1. Rahayu, S. P., Purnami, S.W., Embong, A. Applying Kernel Logistic Regression in Data Mining to Classify Credit Risk, *Proceeding of 3rd International Symposium on Information Technology (ITSIM)*, Kuala Lumpur, August 2008.
2. Rahayu, S. P., Embong, A. Purnami, S.W. Credit Risk Classification using Kernel Logistic Regression with optimal parameter. 10th *International Conference on Information Science, Signal Processing and Their Application (ISSPA)*, Kuala Lumpur, May 2010.
3. Rahayu, S. P., Jasni Mohamad Zain, Embong, A., Juwari, Purnami, S.W., Logistic regression methods with truncated newton method. *International Conference on Advances Science and Contemporary Engineering 2012. (accepted)*

.

A.3 National Conference

1. Rahayu, S. P. , Embong, A. Overview of Random Forest : Effective Ensemble Method in Modern Data Mining. *Proceedings of the National Conference on Software Engineering and Computer Systems (NaCSES2007)*, August 2007, Kuantan, Malaysia.

APPENDIX B

THE INFLUENCE OF PARAMETER

TO CLASSIFICATION PERFORMANCE OF NTR-KLR

1. Parkinson

a.1 *G-means* values on training data

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.08	0.16
4.5	0	0	0	0	0	0	0	0	0	0	0	0.14	0.55	0.64	0.67	0.68	0.68	0.69	0.69
4	0	0	0	0	0	0	0	0	0	0.16	0.55	0.64	0.67	0.68	0.69	0.7	0.69	0.69	0.7
3.5	0	0	0	0	0	0	0	0.17	0.56	0.64	0.67	0.68	0.7	0.7	0.7	0.7	0.71	0.71	0.71
3	0	0	0	0	0	0.14	0.59	0.65	0.66	0.69	0.7	0.72	0.72	0.73	0.73	0.75	0.75	0.76	0.76
2.5	0	0	0	0.09	0.54	0.65	0.68	0.7	0.72	0.73	0.74	0.75	0.77	0.77	0.76	0.76	0.76	0.77	0.77
2	0	0	0.35	0.65	0.67	0.69	0.71	0.73	0.75	0.77	0.78	0.78	0.78	0.82	0.84	0.85	0.86	0.88	0.87
1.5	0	0.26	0.65	0.68	0.69	0.73	0.76	0.78	0.8	0.84	0.88	0.92	0.94	0.95	0.97	0.98	0.98	0.99	0.99
1	0	0.31	0.62	0.68	0.71	0.78	0.84	0.91	0.95	0.97	0.99	0.99	0.99	0.99	1	1	1	1	1
0.5	0	0	0.38	0.58	0.76	0.88	0.97	0.99	0.99	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0.38	0.63	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1
–0.5	0	0	0	0.05	0.36	0.55	1	1	1	1	1	1	1	1	1	1	1	1	1
–1	0	0	0	0	0.16	0.41	1	1	1	1	1	1	1	1	1	1	1	1	1
–1.5	0	0	0	0	0	0.25	1	1	1	1	1	1	1	1	1	1	1	1	1
–2	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
–2.5	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
–3	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

a.2 *Total accuracy* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
5.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76
5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.76	0.76
4.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.83	0.85	0.85	0.85	0.85	0.85	0.85
4	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.83	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
3.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.83	0.85	0.85	0.86	0.85	0.86	0.86	0.86	0.86	0.86	0.86
3	0.75	0.75	0.75	0.75	0.75	0.76	0.84	0.85	0.85	0.86	0.86	0.87	0.87	0.87	0.87	0.88	0.88	0.88	0.88
2.5	0.75	0.75	0.75	0.76	0.83	0.85	0.86	0.86	0.87	0.87	0.87	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88
2	0.75	0.75	0.79	0.86	0.86	0.86	0.87	0.87	0.88	0.89	0.89	0.89	0.89	0.91	0.92	0.92	0.93	0.94	0.93
1.5	0.75	0.78	0.86	0.86	0.86	0.87	0.89	0.9	0.91	0.93	0.94	0.96	0.97	0.97	0.98	0.99	0.99	0.99	0.99
1	0.75	0.79	0.85	0.87	0.88	0.91	0.93	0.96	0.97	0.98	0.99	1	1	1	1	1	1	1	1
0.5	0.75	0.75	0.79	0.84	0.9	0.95	0.99	1	1	1	1	1	1	1	1	1	1	1	1
0	0.75	0.75	0.75	0.79	0.85	0.94	1	1	1	1	1	1	1	1	1	1	1	1	1
−0.5	0.75	0.75	0.75	0.76	0.79	0.83	1	1	1	1	1	1	1	1	1	1	1	1	1
−1	0.75	0.75	0.75	0.75	0.76	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0.75	0.75	0.75	0.75	0.75	0.77	1	1	1	1	1	1	1	1	1	1	1	1	1
−2	0.75	0.75	0.75	0.75	0.75	0.75	1	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0.75	0.75	0.75	0.75	0.75	0.75	1	1	1	1	1	1	1	1	1	1	1	1	1
−3	0.75	0.75	0.75	0.75	0.75	0.75	1	1	1	1	1	1	1	1	1	1	1	1	1

b.1 *G-means* values on testing data (Parkinson)

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.09	0.09	0.09
4.5	0	0	0	0	0	0	0	0	0	0	0	0.06	0.51	0.66	0.66	0.69	0.68	0.68	0.68
4	0	0	0	0	0	0	0	0	0	0.06	0.51	0.66	0.68	0.68	0.68	0.68	0.69	0.69	0.69
3.5	0	0	0	0	0	0	0	0.06	0.51	0.66	0.66	0.68	0.68	0.68	0.69	0.69	0.71	0.71	0.71
3	0	0	0	0	0	0	0.53	0.66	0.68	0.69	0.7	0.7	0.7	0.71	0.71	0.74	0.74	0.74	0.75
2.5	0	0	0	0	0.53	0.66	0.67	0.7	0.7	0.7	0.72	0.74	0.75	0.74	0.74	0.74	0.75	0.75	0.75
2	0	0	0.29	0.65	0.68	0.7	0.7	0.73	0.74	0.74	0.76	0.78	0.79	0.76	0.76	0.77	0.77	0.78	0.76
1.5	0	0.06	0.63	0.68	0.69	0.7	0.73	0.76	0.77	0.77	0.77	0.79	0.8	0.84	0.83	0.88	0.83	0.85	0.87
1	0	0.22	0.55	0.66	0.68	0.75	0.78	0.8	0.79	0.81	0.84	0.86	0.87	0.91	0.91	0.91	0.91	0.92	0.92
0.5	0	0	0.24	0.47	0.6	0.68	0.78	0.84	0.89	0.89	0.89	0.88	0.9	0.9	0.9	0.9	0.9	0.9	0.9
0	0	0	0	0.13	0.25	0.47	0.58	0.67	0.7	0.73	0.71	0.73	0.76	0.76	0.77	0.77	0.8	0.8	0.8
−0.5	0	0	0	0	0.07	0.13	0.31	0.31	0.33	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
−1	0	0	0	0	0	0	0.07	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
−1.5	0	0	0	0	0	0	0	0	0	0	0.2	0	0.2	0.26	0.26	0.26	0.26	0.26	0.26
−2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
−2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
−3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b.2 *Total accuracy* values on testing data (Parkinson)

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
5.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.76	0.76
4.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.82	0.86	0.86	0.86	0.84	0.84	0.84
4	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.82	0.86	0.86	0.85	0.84	0.84	0.84	0.84	0.84
3.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.82	0.86	0.86	0.85	0.85	0.85	0.84	0.84	0.85	0.85	0.85
3	0.75	0.75	0.75	0.75	0.75	0.75	0.83	0.86	0.86	0.86	0.87	0.86	0.86	0.86	0.86	0.87	0.87	0.87	0.87
2.5	0.75	0.75	0.75	0.75	0.83	0.86	0.86	0.86	0.86	0.85	0.86	0.86	0.86	0.86	0.85	0.86	0.86	0.86	0.86
2	0.75	0.75	0.78	0.86	0.86	0.86	0.85	0.86	0.87	0.87	0.88	0.89	0.89	0.88	0.88	0.88	0.87	0.86	0.86
1.5	0.75	0.76	0.85	0.86	0.87	0.86	0.87	0.88	0.89	0.89	0.88	0.89	0.89	0.91	0.9	0.92	0.88	0.89	0.89
1	0.75	0.77	0.83	0.86	0.87	0.89	0.9	0.9	0.89	0.89	0.9	0.91	0.92	0.93	0.94	0.94	0.94	0.94	0.94
0.5	0.75	0.75	0.78	0.81	0.85	0.87	0.9	0.92	0.94	0.93	0.93	0.93	0.94	0.94	0.94	0.94	0.94	0.94	0.94
0	0.75	0.75	0.75	0.76	0.78	0.81	0.84	0.87	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.9	0.9	0.9
−0.5	0.75	0.75	0.75	0.75	0.76	0.76	0.78	0.78	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79
−1	0.75	0.75	0.75	0.75	0.75	0.75	0.76	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77
−1.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.77	0.75	0.77	0.77	0.77	0.77	0.77	0.77	0.77
−2	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
−2.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
−3	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75

2. Glass7

a.1 *G-means* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0.04	0.31	0.8	0.88	0.89	0.91	0.92
4	0	0	0	0	0	0	0	0	0	0	0.08	0.48	0.85	0.88	0.89	0.9	0.9	0.9	0.9
3.5	0	0	0	0	0	0	0	0	0.08	0.56	0.86	0.88	0.89	0.89	0.89	0.9	0.9	0.9	0.9
3	0	0	0	0	0	0	0.08	0.56	0.83	0.9	0.9	0.91	0.9	0.92	0.93	0.93	0.94	0.94	0.94
2.5	0	0	0	0	0.08	0.55	0.83	0.9	0.91	0.91	0.91	0.92	0.93	0.94	0.94	0.95	0.94	0.95	0.95
2	0	0	0	0.42	0.82	0.89	0.91	0.91	0.93	0.93	0.93	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
1.5	0	0	0.73	0.88	0.9	0.92	0.93	0.93	0.96	0.96	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
0.5	0	0	0	0	0	0	0.24	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0.61	1	1	1	1	1	1	1	1	1	1	1	1
−0.5	0	0	0	0	0.08	0.4	1	1	1	1	1	1	1	1	1	1	1	1	1
−1	0	0	0	0.14	0.66	0.8	0.99	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0	0	0.39	0.8	0.85	0.87	0.98	0.98	0.99	1	1	1	1	1	1	1	1	1	1
−2	0	0.54	0.85	0.85	0.87	0.91	0.96	0.98	0.98	0.98	0.99	1	1	1	1	1	1	1	1
−2.5	0	0.73	0.85	0.88	0.9	0.94	0.95	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	1
−3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

a.2 *Total accuracy* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
4.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.88	0.95	0.96	0.96	0.97	0.96
4	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.89	0.96	0.96	0.96	0.96	0.96	0.96	0.96
3.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.91	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97
3	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.91	0.95	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98
2.5	0.86	0.86	0.86	0.86	0.87	0.91	0.95	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98
2	0.86	0.86	0.86	0.89	0.95	0.97	0.97	0.97	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1.5	0.86	0.86	0.94	0.97	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1	0.86	0.94	0.96	0.97	0.97	0.98	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1
0.5	0.86	0.91	0.96	0.96	0.97	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1	1
0	0.86	0.86	0.89	0.95	0.96	0.97	1	1	1	1	1	1	1	1	1	1	1	1	1
−0.5	0.86	0.86	0.86	0.87	0.92	0.95	1	1	1	1	1	1	1	1	1	1	1	1	1
−1	0.86	0.86	0.86	0.86	0.87	0.89	1	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0.86	0.86	0.86	0.86	0.86	0.86	0.91	1	1	1	1	1	1	1	1	1	1	1	1
−2	0.86	0.86	0.86	0.86	0.86	0.86	0.87	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	1	1	1	1	1	1	1	1	1	1	1	1
−3	0.86	0.86	0.86	0.86	0.86	0.86	0.86	1	1	1	1	1	1	1	1	1	1	1	1

b.1 *G-means* values on testing data (Glass7)

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0.21	0.78	0.87	0.88	0.88	0.9
4	0	0	0	0	0	0	0	0	0	0	0	0.21	0.83	0.87	0.88	0.9	0.9	0.9	0.9
3.5	0	0	0	0	0	0	0	0	0	0.44	0.82	0.86	0.88	0.88	0.9	0.92	0.92	0.92	0.92
3	0	0	0	0	0	0	0	0.44	0.8	0.84	0.9	0.9	0.9	0.9	0.92	0.92	0.92	0.92	0.92
2.5	0	0	0	0	0	0.44	0.8	0.85	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.91	0.91
2	0	0	0	0.29	0.8	0.86	0.88	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.92	0.91	0.91	0.91	0.91
1.5	0	0	0.6	0.85	0.87	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
1	0	0.6	0.83	0.85	0.85	0.87	0.87	0.89	0.91	0.91	0.91	0.91	0.91	0.91	0.93	0.93	0.93	0.93	0.93
0.5	0	0.32	0.81	0.85	0.85	0.85	0.85	0.85	0.85	0.87	0.87	0.87	0.85	0.87	0.85	0.85	0.85	0.85	0.85
0	0	0	0	0.7	0.78	0.8	0.83	0.83	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
−0.5	0	0	0	0	0.09	0.63	0.74	0.76	0.74	0.76	0.78	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
−1	0	0	0	0	0	0	0	0.17	0.29	0.25	0.29	0.45	0.45	0.45	0.45	0.49	0.52	0.52	0.55
−1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
−2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
−2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
−3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b.2 *Total accuracy* values on testing data (Glass7)

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
4.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.94	0.96	0.96	0.96	0.96
4	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.95	0.96	0.96	0.96	0.96	0.96	0.96
3.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.89	0.95	0.96	0.96	0.96	0.96	0.97	0.96	0.96	0.96
3	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.89	0.94	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
2.5	0.86	0.86	0.86	0.86	0.86	0.89	0.95	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97
2	0.86	0.86	0.86	0.88	0.95	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
1.5	0.86	0.86	0.93	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
1	0.86	0.93	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98
0.5	0.86	0.89	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.96	0.97	0.96	0.96	0.96	0.96	0.96
0	0.86	0.86	0.86	0.93	0.95	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
−0.5	0.86	0.86	0.86	0.86	0.87	0.92	0.94	0.94	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
−1	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.9	0.9	0.9	0.91
−1.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
−2	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
−2.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
−3	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86

3. ImgSegment1

a.1 *G-means* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.08	0.19	0.23	0.4
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0.06	0.21	0.51	0.72	0.82	0.9
4.5	0	0	0	0	0	0	0	0	0	0	0	0.06	0.25	0.55	0.74	0.84	0.92	0.94	0.96
4	0	0	0	0	0	0	0	0	0	0.08	0.26	0.56	0.74	0.84	0.91	0.94	0.96	0.96	0.96
3.5	0	0	0	0	0	0	0	0.06	0.26	0.56	0.75	0.85	0.91	0.95	0.96	0.97	0.97	0.97	0.97
3	0	0	0	0	0	0.06	0.29	0.59	0.77	0.88	0.94	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99
2.5	0	0	0	0	0.32	0.64	0.81	0.92	0.96	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2	0	0	0.37	0.71	0.87	0.94	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1.5	0.42	0.77	0.9	0.95	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1	0.89	0.93	0.96	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.5	0.93	0.96	0.97	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1
0	0.89	0.95	0.96	0.96	0.97	0.98	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1
−0.5	0.75	0.86	0.93	0.95	0.95	0.97	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1
−1	0.23	0.57	0.75	0.85	0.91	0.95	0.96	0.99	1	1	1	1	1	1	1	1	1	1	1
−1.5	0	0.06	0.37	0.55	0.72	0.84	0.95	1	1	1	1	1	1	1	1	1	1	1	1
−2	0	0	0	0.1	0.37	0.61	0.83	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0	0	0	0	0	0.41	0.63	1	1	1	1	1	1	1	1	1	1	1	1
−3	0	0	0	0	0	0.4	0.45	1	1	1	1	1	1	1	1	1	1	1	1

a.2 *Total accuracy* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87
5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.89	0.93	0.95	0.97
4.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.98	0.98	0.99
4	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.98	0.98	0.98	0.99	0.99
3.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.98	0.98	0.99	0.99	0.99	0.99	0.99
3	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.91	0.94	0.97	0.98	0.99	0.99	0.99	0.99	1	1	1	1
2.5	0.86	0.86	0.86	0.86	0.87	0.92	0.95	0.98	0.99	0.99	1	1	1	1	1	1	1	1	1
2	0.86	0.86	0.88	0.93	0.96	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1	1
1.5	0.88	0.94	0.97	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0.97	0.98	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0.5	0.98	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0.97	0.99	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1
−0.5	0.94	0.96	0.98	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1
−1	0.86	0.9	0.94	0.96	0.98	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0.86	0.86	0.88	0.9	0.93	0.96	0.98	1	1	1	1	1	1	1	1	1	1	1	1
−2	0.86	0.86	0.86	0.86	0.88	0.91	0.96	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0.86	0.86	0.86	0.86	0.86	0.88	0.91	1	1	1	1	1	1	1	1	1	1	1	1
−3	0.86	0.86	0.86	0.86	0.86	0.88	0.89	1	1	1	1	1	1	1	1	1	1	1	1

b.1 *G-means* values on testing data (ImgSegment1)

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.05	0.18	0.22	0.38
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.19	0.51	0.71	0.83	0.9
4.5	0	0	0	0	0	0	0	0	0	0	0	0.05	0.24	0.52	0.74	0.83	0.91	0.95	0.96
4	0	0	0	0	0	0	0	0	0	0.05	0.24	0.54	0.74	0.83	0.91	0.95	0.96	0.96	0.96
3.5	0	0	0	0	0	0	0	0.05	0.24	0.55	0.74	0.84	0.91	0.95	0.96	0.97	0.96	0.97	0.97
3	0	0	0	0	0	0.02	0.27	0.57	0.77	0.88	0.94	0.97	0.97	0.98	0.99	0.99	0.99	0.99	0.99
2.5	0	0	0	0	0.31	0.62	0.81	0.92	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2	0	0	0.35	0.7	0.87	0.94	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1.5	0.42	0.77	0.9	0.95	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1	0.9	0.93	0.96	0.97	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.5	0.93	0.96	0.96	0.96	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0	0.88	0.94	0.96	0.96	0.97	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
–0.5	0.72	0.84	0.91	0.93	0.93	0.95	0.96	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
–1	0.21	0.52	0.69	0.81	0.85	0.89	0.89	0.92	0.94	0.95	0.95	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97
–1.5	0	0	0.27	0.45	0.59	0.69	0.75	0.79	0.78	0.81	0.82	0.83	0.84	0.85	0.85	0.86	0.86	0.86	0.86
–2	0	0	0	0	0.15	0.31	0.48	0.55	0.6	0.57	0.61	0.62	0.63	0.64	0.64	0.64	0.64	0.64	0.64
–2.5	0	0	0	0	0	0	0.26	0.41	0.41	0.42	0.41	0.42	0.43	0.43	0.43	0.43	0.44	0.44	0.44
–3	0	0	0	0	0	0	0.06	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4

b.2 *Total accuracy* values on testing data (ImgSegment1)

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
5.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87
5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.89	0.93	0.95	0.97
4.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.97	0.98	0.99
4	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.98	0.98	0.99	0.99	0.99
3.5	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.96	0.97	0.99	0.99	0.99	0.99	0.99	0.99
3	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.9	0.94	0.97	0.98	0.99	0.99	0.99	0.99	0.99	1	1	1
2.5	0.86	0.86	0.86	0.86	0.87	0.91	0.95	0.98	0.99	0.99	0.99	1	1	1	1	1	1	1	1
2	0.86	0.86	0.87	0.93	0.96	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1	1
1.5	0.88	0.94	0.97	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1
1	0.97	0.98	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1
0.5	0.98	0.99	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1	1
–0.5	0.93	0.96	0.98	0.98	0.98	0.99	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	1
–1	0.86	0.9	0.93	0.95	0.96	0.97	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
–1.5	0.86	0.86	0.87	0.89	0.91	0.93	0.94	0.95	0.94	0.95	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
–2	0.86	0.86	0.86	0.86	0.86	0.87	0.89	0.9	0.91	0.9	0.91	0.91	0.91	0.92	0.92	0.92	0.92	0.92	0.92
–2.5	0.86	0.86	0.86	0.86	0.86	0.86	0.87	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88
–3	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88

4. Balance2

a.1 *G-means* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.03	0.06	0.1	0.21	0.25
0.5	0	0	0	0	0	0	0	0	0	0.03	0.06	0.1	0.16	0.24	0.28	0.27	0.28	0.32	0.32
0	0	0	0	0	0	0	0.03	0.03	0.08	0.24	0.3	0.41	0.47	0.52	0.59	0.64	0.65	0.67	0.68
−0.5	0	0	0	0	0	0	0.03	0.2	0.59	0.87	0.99	1	1	1	1	1	1	1	1
−1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
−2	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
−3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

a.2 *Total accuracy* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
5.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
4.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
4	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
3.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
3	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
2.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
2	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
1.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
1	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.93	0.93
0.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.93	0.93	0.93	0.93	0.93	0.93
0	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.93	0.93	0.94	0.94	0.94	0.95	0.95	0.96	0.96	0.96
−0.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.93	0.95	0.98	1	1	1	1	1	1	1	1	1
−1	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1	1	1	1	1	1	1	1	1	1
−1.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1	1	1	1	1	1	1	1	1	1
−2	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1	1	1	1	1	1	1	1	1	1
−2.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1	1	1	1	1	1	1	1	1	1
−3	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1	1	1	1	1	1	1	1	1	1

b.1 *G-means* values on testing data (Balance2)

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.06	0.06	0.06
0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b.2 *Total accuracy* values on testing data (Balance2)

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
5.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
4.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
4	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
3.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
3	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
2.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
2	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
1.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
1	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
0.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.91	0.91	0.91	0.91
0	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.91	0.91	0.91	0.89	0.89	0.89	0.88	0.88	0.88
−0.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.91	0.91	0.9	0.88	0.86	0.84	0.82	0.8	0.79	0.78	0.78	0.78
−1	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.91
−1.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
−2	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
−2.5	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
−3	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92

5. Car3

a.1 *G-means* values on training data

Exp.	λ=2	1.5	1	0.5	0	−0.5	−1	−1.5	−2	−2.5	−3	−3.5	−4	−4.5	−5	−5.5	−6	−6.5	−7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.13	0.23	0.23	0.23
2	0	0	0	0	0	0	0	0	0	0	0	0.07	0.21	0.25	0.26	0.28	0.34	0.36	0.37
1.5	0	0	0	0	0	0	0	0	0.19	0.25	0.31	0.41	0.45	0.48	0.51	0.53	0.55	0.56	0.57
1	0	0	0	0	0	0.14	0.28	0.43	0.49	0.54	0.58	0.6	0.68	0.74	0.8	0.85	0.88	0.89	0.9
0.5	0	0	0	0.09	0.33	0.51	0.59	0.72	0.81	0.88	0.93	0.96	0.98	0.98	0.99	1	1	1	1
0	0	0	0	0.14	0.42	0.71	0.87	0.97	1	1	1	1	1	1	1	1	1	1	1
−0.5	0	0	0	0	0	0.2	0.69	0.99	1	1	1	1	1	1	1	1	1	1	1
−1	0	0	0	0	0	0	0	0.03	1	1	1	1	1	1	1	1	1	1	1
−1.5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
−2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
−2.5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
−3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

a.2 *Total accuracy* values on training data

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
5.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
4.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
4	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
3.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
3	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
2.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
2	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
1.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97
1	0.96	0.96	1	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.99	0.99	0.99	0.99
0.5	0.96	0.96	1	0.96	0.96	0.97	0.97	0.98	0.98	0.99	0.99	1	1	1	1	1	1	1	1
0	0.96	0.96	1	0.96	0.97	0.98	0.99	1	1	1	1	1	1	1	1	1	1	1	1
–0.5	0.96	0.96	1	0.96	0.96	0.96	0.98	1	1	1	1	1	1	1	1	1	1	1	1
–1	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	1	1	1	1	1	1	1	1	1	1	1
–1.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	1	1	1	1	1	1	1	1	1	1	1
–2	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	1	1	1	1	1	1	1	1	1	1	1
–2.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	1	1	1	1	1	1	1	1	1	1	1
–3	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	1	1	1	1	1	1	1	1	1	1	1

b.1 *G-means* values on testing data

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.13	0.15	0.15
2	0	0	0	0	0	0	0	0	0	0	0	0	0.13	0.13	0.2	0.25	0.25	0.28	0.28
1.5	0	0	0	0	0	0	0	0	0.11	0.18	0.24	0.31	0.37	0.43	0.49	0.49	0.5	0.51	0.51
1	0	0	0	0	0	0.05	0.24	0.34	0.42	0.48	0.5	0.51	0.58	0.62	0.65	0.72	0.73	0.73	0.74
0.5	0	0	0	0.06	0.31	0.43	0.49	0.59	0.67	0.68	0.75	0.79	0.83	0.86	0.89	0.9	0.91	0.91	0.91
0	0	0	0	0.06	0.33	0.5	0.66	0.71	0.75	0.82	0.85	0.87	0.88	0.91	0.91	0.91	0.91	0.91	0.91
–0.5	0	0	0	0	0	0	0.22	0.33	0.51	0.6	0.65	0.67	0.73	0.78	0.78	0.81	0.81	0.82	0.85
–1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
–3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b.2 *Total accuracy* values on testing data

Exp.	λ=2	1.5	1	0.5	0	–0.5	–1	–1.5	–2	–2.5	–3	–3.5	–4	–4.5	–5	–5.5	–6	–6.5	–7
σ=6	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
5.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
4.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
4	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
3.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
3	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
2.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
2	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
1.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.96	0.96	0.96
1	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98
0.5	0.96	0.96	1	0.96	0.96	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
0	0.96	0.96	1	0.96	0.96	0.97	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
–0.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99
–1	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
–1.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
–2	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
–2.5	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
–3	0.96	0.96	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96

APPENDIX C

THE INFLUENCE OF PARAMETER TO CLASSIFICATION PERFORMANCE OF NTR-LR

1. Magic data set

λ Exp.	*Total Accuracy*		*G-means*	
	Training Data	Testing Data	Training Data	Testing Data
7	0.778128	0.778181	0.683088	0.683148
6.5	0.781165	0.780915	0.696242	0.696057
6	0.783636	0.783386	0.705629	0.705017
5.5	0.787605	0.787224	0.717168	0.716377
5	0.788775	0.789064	0.721262	0.721346
4.5	0.789235	0.789274	0.723532	0.723742
4	0.790142	0.790221	0.725728	0.72608
3.5	0.790457	0.790431	0.726872	0.726935
3	0.790786	0.790641	0.727796	0.727605
2.5	0.790786	0.791009	0.728064	0.728435
2	0.79097	0.791009	0.728417	0.728614
1.5	0.791115	0.791062	0.728745	0.728765
1	0.791246	0.791115	0.729006	0.7288
0.5	0.791207	0.79122	0.729011	0.729046
0	0.79122	0.791167	0.729064	0.729013
-0.5	0.79122	0.79122	0.729094	0.729106
-1	0.791259	0.79122	0.729148	0.729106
-1.5	0.791246	0.79122	0.72914	0.729106
-2	0.791233	0.79122	0.729131	0.729106
-2.5	0.79122	0.79122	0.729123	0.729106
-3	0.791246	0.79122	0.729155	0.729106
-3.5	0.791246	0.79122	0.729155	0.729106
-4	0.791259	0.79122	0.729178	0.729106
-4.5	0.791259	0.79122	0.729178	0.729106
-5	0.791259	0.79122	0.729178	0.729106
-5.5	0.791259	0.79122	0.729178	0.729106
-6	0.791259	0.79122	0.729178	0.729106
-6.5	0.791259	0.79122	0.729178	0.729106
-7	0.791259	0.79122	0.729178	0.729106

2. Shuttle2to7 data set

λ Exp.	*Total Accuracy*		*G-means*	
	Training Data	**Testing Data**	**Training Data**	**Testing Data**
7	0.924302	0.924224	0.819632	0.819522
6.5	0.932444	0.932414	0.845438	0.845388
6	0.943004	0.942966	0.87624	0.876314
5.5	0.947651	0.947586	0.889765	0.889662
5	0.952522	0.952638	0.902796	0.903074
4.5	0.955289	0.955379	0.911054	0.911491
4	0.95806	0.957897	0.918668	0.918326
3.5	0.960185	0.960259	0.924243	0.924524
3	0.961539	0.961534	0.927925	0.928053
2.5	0.963099	0.962948	0.931796	0.931552
2	0.964491	0.964397	0.935255	0.935153
1.5	0.965582	0.96531	0.937831	0.937177
1	0.966784	0.966793	0.940485	0.9404
0.5	0.967974	0.967948	0.942975	0.943
0	0.968513	0.968466	0.943998	0.943979
-0.5	0.968629	0.968569	0.944574	0.944674
-1	0.96919	0.969293	0.945959	0.946406
-1.5	0.969534	0.969483	0.946579	0.946525
-2	0.970892	0.970707	0.949637	0.949344
-2.5	0.971526	0.971414	0.951174	0.951057
-3	0.971966	0.971741	0.952169	0.951819
-3.5	0.971996	0.971724	0.952234	0.951742
-4	0.971767	0.971586	0.951665	0.951347
-4.5	0.971578	0.971483	0.951112	0.951033
-5	0.971526	0.971431	0.950893	0.95075
-5.5	0.971466	0.971379	0.950723	0.950625
-6	0.971392	0.971397	0.95053	0.950667
-6.5	0.971362	0.971362	0.950449	0.950583
-7	0.971332	0.97131	0.950376	0.950458

3. LetterImg26 data set

λ Exp.	*Total Accuracy*		*G-means*	
	Training Data	**Testing Data**	**Training Data**	**Testing Data**
7	0.9633	0.9633	0	0
6.5	0.96735	0.9673	0.331837	0.32834
6	0.96955	0.96955	0.412628	0.411954
5.5	0.970913	0.9709	0.45944	0.456928
5	0.9731	0.97295	0.53021	0.525145
4.5	0.975375	0.9751	0.596795	0.590972
4	0.97795	0.9777	0.666579	0.661034
3.5	0.9808	0.98055	0.736323	0.731471
3	0.982038	0.9818	0.769248	0.765828
2.5	0.982625	0.98235	0.784859	0.77972
2	0.982963	0.98255	0.795065	0.79078
1.5	0.983125	0.9828	0.802176	0.799254
1	0.983213	0.983	0.806941	0.805982
0.5	0.98325	0.983	0.810639	0.809256
0	0.983363	0.98315	0.813738	0.813368
-0.5	0.9834	0.98315	0.815576	0.814996
-1	0.983425	0.9831	0.816394	0.815786
-1.5	0.983425	0.9831	0.817205	0.815786
-2	0.98345	0.9833	0.817618	0.819102
-2.5	0.98345	0.9833	0.817618	0.819102
-3	0.983463	0.98335	0.817825	0.819922
-3.5	0.983463	0.98335	0.817825	0.819922
-4	0.983463	0.98335	0.817825	0.819922
-4.5	0.983488	0.98335	0.818241	0.819922
-5	0.983488	0.98335	0.818241	0.819922
-5.5	0.983488	0.98335	0.818241	0.819922
-6	0.983488	0.98335	0.818241	0.819922
-6.5	0.983488	0.98335	0.818241	0.819922
-7	0.983488	0.98335	0.818241	0.819922

APPENDIX D

MATLAB CODE OF PROPOSED NTR-KLR ALGORITHM

```
x=xtr;
n=size(x,1);

y=ytr;

xval=xts;
n1=size(xval,1);

yval=yts;

%rbf
sigma2=sigma*sigma;
gamma=(1/(2*sigma2));
XXh = sum(x.^2,2)*ones(1,n);
ktr = XXh+XXh'-2*x*x';
ktr = exp(-gamma*ktr);

% kernel matrix testing
%rbf
XXh1 = sum(x.^2,2)*ones(1,n1);
    XXh2 = sum(xval.^2,2)*ones(1,n);
    kts = exp(-gamma*kts);
    kts = kts';

% adding bias term
% training
[n,n]=size(ktr);
ktr=[ktr ones(n,1)];
[n,m]=size(ktr);
ktrp=[ktr' zeros(m,1)];
[m,m]=size(ktrp);

[n1,n]=size(kts);
kts=[kts ones(n1,1)];
[n1,m]=size(kts);

ktry=zeros(n,m);
for j=1:m;
    for i=1:n;
        ktry(i,j)=ktr(i,j)*y(i);
    end;
end;

Id=eye(m);Id(m,m)=0;
alpha=zeros(m,1);

h=ones(1,n)*log(1+exp(-ktry*alpha))+(lambda/2))*alpha;

for iterKLR = 1:30
  old_alpha = alpha;
  old_h = h;
```

```
   % s1 is n by 1
  % s1 = 1-sigma;
  s1 = 1./(1+exp(ktry*alpha));
  v = s1.*(1-s1);

  ktryv=zeros(n,m);
for j=1:m;
    for i=1:n;
        ktryv(i,j)=ktry(i,j)*v(i);
    end;
end;
    g = -ktry'*s1+ lambda*(Id*ktrp)*alpha;
    H = ktryv'*ktry + lambda*(Id*ktrp);

    [s,itercg,rtr] = cg_klr(H, g);
    alpha = alpha + s;

  if nargout > 1
    run.alpha(:,iterKLR) = alpha;
  end

    etr=abs(h - old_h)/abs(h);

  if etr < 2.5
    break
  end
end


function [s,itercg,rtr] = cg_klr(H, g)

b = -g;
m = length(b);
maxiter = 1000;
errtol = 0.005;
s = zeros(m,1);
r = b;
rtr = r'*r;
d = r;

itercg = 0;
while rtr > errtol  &&  itercg < maxiter
    itercg = itercg+1;

    Hd = H*d;
    a = rtr / (d'*Hd);
    s = s + a * d;

    old_rtr = rtr;
    r = r - a * Hd;
    bta = rtr / old_rtr;
    d = r + bta * d;

end;
```

APPENDIX E

MATLAB CODE OF PROPOSED NTR-LR ALGORITHM

```
x=xtr;
y=ytr;
xval=xts;
yval=yts;

% adding bias term
[n,d]=size(x);
x=[x ones(n,1)];
[n,p]=size(x);

[n1,d]=size(xval);
xval=[xval ones(n1,1)];
[n1,p]=size(xval);

xy=zeros(n,p);
for j=1:p;
    for i=1:n;
        xy(i,j)=x(i,j)*y(i);
    end;
end;

Id=eye(p);Id(p,p)=0;
beta=zeros(p,1);
h=ones(1,n)*log(1+exp(-xy*beta;

for iterRLR = 1:30
  old_beta = beta;
  old_h = h;
  % s1 = 1-sigma
  % s1 is n by 1
  s1 = 1./(1+exp(xy*beta));
  v = s1.*(1-s1);

   xyv=zeros(n,p);
for j=1:p;
    for i=1:n;
        xyv(i,j)=xy(i,j)*v(i);
    end;
end;

   g = -xy'*s1+ lambda*Id*beta;
    H = (xyv'*xy) + lambda*Id;
        [s,itercg,rtr] = cg_reglr(H, g);
    beta = beta + s;

h=ones(1,n)*log(1+exp(-xy*beta))+ lambda*beta'*Id*beta;
  if nargout > 1
    run.beta(:,iterRLR) = beta;
  end
        etr=abs(h - old_h)/abs(h);
if etr < 1e-2
     break
```

```
  end
end
% prediction with training data
p1tr=1./(1+exp(-x*beta));

% labelling of training data
class=sign(p1tr-(1/2));

% prediction with testing data
p1val=1./(1+exp(-xval*beta));

% labelling of testing data
classval=sign(p1val-(1/2));


function [s,itercg,rtr] = cg_reglr(H,g)

b = -g;
p = length(b);
maxiter = 200;
errtol = 0.005;
r = b;
rtr = r'*r;
d = r;

itercg = 0;
while rtr > errtol  &&  itercg < maxiter
    itercg = itercg+1;

    Hd = H*d;
    s = s + a * d;

    old_rtr = rtr;
    r = r - a * Hd;
    rtr = r'*r;
    bta = rtr / old_rtr;
    d = r + bta * d;

end;
```

APPENDIX F

MATLAB CODE OF PROPOSED AB-WKLR ALGORITHM

```
x=xtr;
n=size(x,1);

y=ytr;

Smin=lnsigma;
Smax=lnsigma+du;

%parameter
sigmap=exp(Smax:step:Smin);
nS=size(sigmap,2);

%initialization
we = zeros(n,1);
we(1:n) = 1/n; % weight initialization
PSI = [];
Alpha =[];
H = [];
SIGMA=[];

iterRBF=0;
    for S=1:nS;
        sigma=sigmap(S);

%rbf train
sigma2=sigma*sigma;
gamma=(1/(2*sigma2));
XXh = sum(x.^2,2)*ones(1,n);
ktr = exp(-gamma*sqe);
[n,n]=size(ktr);

% adding bias term
% training
ktr=[ktr ones(n,1)];
[n,m]=size(ktr);

    for t=1:30;
    %fprintf('new training data...\n');
    [alpha,iterKLR,iterCG] = adaweight_trklr(ktr,y,lambda,we);
    f = ktr*alpha;
    p1tr=1./(1+exp(-f));
    ht=sign(p1tr-0.5);
    n_ierr = sum(i_err);
    E = (we'*i_err);%weighted error of ht
    if E == 0
      %E = 1e-20;
        break
    end
    if E > 0.5
```

```
            break
        end

        iterRBF=iterRBF+1;

        psi = 0.5*log((1-E)/E);

        Alpha = [Alpha alpha];
        SIGMA=[SIGMA sigma];
        H = [H ht];
        PSI = [PSI psi];

        for i = 1:n
            %we(i) = we(i)*exp(-PSI(:,t)*(y(i).*h(i)));

            m(i)=y(i).*f(i);

            if m(i) <= 0
            g(i)=exp(PSI(:,t))+ 1*(1-exp(PSI(:,t)))*(1-exp(y(i).*f(i)));
            else
            g(i)=exp(-PSI(:,t))+ 1*(1-exp(-PSI(:,t)))*exp(-(y(i).*f(i)));
            end
            we(i) = we(i)*g(i);

        end

        we = (we/sum(we));%A.*i_err;
      end
        end

iterSi=iterRBF/nS;
Ftr = zeros(n,1);
for t=1:T
ftr= PSI(:,t)*H(:,t);
Ftr=Ftr+ftr;
class=sign(Ftr);
end

%validation
%data test

xval=xts;
n1=size(xval,1);

yval=yts;

HTS=[];
Fts=zeros(n1,1);



for t = 1:T
    %rbf test
    sigma2=SIGMA(:,t)*SIGMA(:,t);
    gamma=(1/(2*sigma2));
    XXh1 = sum(x.^2,2)*ones(1,n1);
    XXh2 = sum(xval.^2,2)*ones(1,n);
```

```
    kts = exp(-gamma*sqe1);
    kts = kts';
    [n1,n]=size(kts);

   % adding bias term
   % testing
   kts=[kts ones(n1,1)];
   [n1,m]=size(kts);

    f1 = kts*Alpha(:,t);
    p1ts=1./(1+exp(-f1));
    hts=sign(p1ts-0.5);
    HTS=[HTS hts];
    fts= PSI(:,t)*HTS(:,t);
    Fts=Fts+fts;
    classval = sign(Fts);
end


function [alpha,iterKLR,itercg] = adaweight_trklr(ktr,y,lambda,we)

[n,m]=size(ktr);
ktrp=[ktr' zeros(m,1)];
[m,m]=size(ktrp);

ktry=zeros(n,m);
for j=1:m;
    for i=1:n;
        ktry(i,j)=ktr(i,j)*y(i);
    end;
end;

we=we/mean(we);
ktryw=zeros(n,m);
for j=1:m;
    for i=1:n;
        ktryw(i,j)=ktry(i,j)*we(i);
    end;
end;

Id=eye(m);Id(m,m)=0;
alpha=zeros(m,1);

h=(we'.*ones(1,n))*log(1+exp(ktry*alpha))+(lambda/2)*alpha'*(Id*ktrp)*
alpha;

for iterKLR = 1:30
  old_alpha = alpha;
  % s1 is n by 1
  % s1 = 1-sigma;
  s1 = 1./(1+exp(ktry*alpha));
  v = s1.*(1-s1);

ktrywv=zeros(n,m);
for j=1:m;
    for i=1:n;
        ktrywv(i,j)=ktryw(i,j)*v(i);
    end;
end;
```

```
    g = -ktryw'*s1+ lambda*(Id*ktrp)*alpha;
    H = ktrywv'*ktry + lambda*(Id*ktrp);

    [s,itercg,rtr] = cg_klr(H, g);

h=(we'.*ones(1,n))*log(1+exp(ktry*alpha))+(lambda/2)*alpha'*(Id*ktrp)*
alpha;

  if nargout > 1
   run.alpha(:,iterKLR) = alpha;
  end

    etr=abs(h - old_h)/abs(h);

 if etr < 2.5
    break
  end
end
```

APPENDIX G

MATLAB CODE OF PROPOSED AB-WLR ALGORITHM

```
x=xtr;

% adding bias term
[n,d]=size(x);
x=[x ones(n,1)];
[n,p]=size(x);

y=ytr;

Lmin=lnlambda;
Lmax=lnlambda+du;

%parameter
lambdap=exp(Lmax:step:Lmin);
nL=size(lambdap,2);

%initialization
we = zeros(n,1);
we(1:n) = 1/n; % weight initialization
PSI = [];
BETA =[];
H = [];
LAMBDA=[];

iterWB=0;
for l=1:nL;
        lambda=lambdap(l);

    for t=1:30;
    %fprintf('new training data...\n');
    [beta,iterRLR,itercg,rtr] = adaweight_trreglr(x,y,lambda,we);

    f = x*beta;
    ht=sign(p1tr-0.5);
    i_err = (ht~=y);
    n_ierr = sum(i_err);

    E = (we'*i_err);%weighted error of ht
    if E == 0
    %   E = 1e-20;
    break
    end

    if E > 0.5
         break
    end
```

```
    iterWB=iterWB+1;

    psi = 0.5*log((1-E)/E);

    BETA = [BETA beta];
    H = [H ht];
    PSI = [PSI psi];
    LAMBDA=[LAMBDA lambda];

    for i = 1:n
        %we(i) = we(i)*exp(-PSI(:,t)*(y(i).*h(i)));

        m(i)=y(i).*f(i);

        if m(i) <= 0
        g(i)=exp(PSI(:,t))+ 1*(1-exp(PSI(:,t)))*(1-exp(y(i).*f(i)));
        else
        g(i)=exp(-PSI(:,t))+ 1*(1-exp(-PSI(:,t)))*exp(-(y(i).*f(i)));
        end

        we(i) = we(i)*g(i);
    end

    we = we/sum(we);

    end
end

iterLi=iterWB/nL;
T = size(PSI,2);

for t=1:T
ftr= PSI(:,t)*H(:,t);
Ftr=Ftr+ftr;
class=sign(Ftr);
end

%validation
%data test
xval=xts;

% adding bias term
[n1,d]=size(xval);
xval=[xval ones(n1,1)];
[n1,p]=size(xval);

yval=yts;

HTS=[];
Fts=zeros(n1,1);

for t = 1:T
    f1 = xval*BETA(:,t);
    p1ts=1./(1+exp(-f1));
    hts=sign(p1ts-0.5);
    HTS=[HTS hts];
    fts= PSI(:,t)*HTS(:,t);
```

```
    Fts=Fts+fts;
    classval = sign(Fts);
end



function [beta,iterRLR,itercg,rtr]=adaweight_trreglr(x,y,lambda,we)

[n,p]=size(x);

xy=zeros(n,p);
for j=1:p;
    for i=1:n;
        xy(i,j)=x(i,j)*y(i);
    end;
end;

we=we/mean(we);
for j=1:p;
    for i=1:n;
        xyw(i,j)=xy(i,j)*we(i);
    end;
end;

Id=eye(p);Id(p,p)=0;
h=(we'.*ones(1,n))*log(1+exp(-xy*beta))+ lambda*beta'*Id*beta;

for iterRLR = 1:30
  old_beta = beta;
  old_h = h;
  % s1 = 1-sigma
  % s1 is n by 1
  s1 = 1./(1+exp(xy*beta));
  v = s1.*(1-s1);

  xywv=zeros(n,p);
for j=1:p;
    for i=1:n;
        xywv(i,j)=xyw(i,j)*v(i);
    end;
end;

   g = -xyw'*s1+ lambda*Id*beta;
    H = (xywv'*xy) + lambda*Id;

    [s,itercg,rtr]=cg_reglr(H, g);
    beta = beta + s;

h=(we'.*ones(1,n))*log(1+exp(-xy*beta))+ lambda*beta'*Id*beta;


  if nargout > 1
    run.beta(:,iterRLR) = beta;
  end

    etr=abs(h - old_h)/abs(h);

 if etr < 1e-2
```

```
      break
    end
  end
```