BOOSTING FOR LEARNING FROM IMBALANCED, MULTICLASS DATA SETS

Mohamed Abouelenien

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2013

APPROVED:

Xiaohui Yuan, Major Professor
Kathleen Swigger, Committee Member
Song Fu, Committee Member
Jianguo Liu, Committee Member
Barrett Bryant, Chair of the Department of Computer Science and Engineering
Coastas Tsatsoulis, Dean of the College of Engineering
Mark Wardell, Dean of the Toulouse Graduate School
In many real-world applications, it is common to have uneven number of examples among multiple classes. The data imbalance, however, usually complicates the learning process, especially for the minority classes, and results in deteriorated performance. Boosting methods were proposed to handle the imbalance problem. These methods need elongated training time and require diversity among the classifiers of the ensemble to achieve improved performance. Additionally, extending the boosting method to handle multi-class data sets is not straightforward. Examples of applications that suffer from imbalanced multi-class data can be found in face recognition, where tens of classes exist, and in capsule endoscopy, which suffers massive imbalance between the classes.

This dissertation introduces RegBoost, a new boosting framework to address the imbalanced, multi-class problems. This method applies a weighted stratified sampling technique and incorporates a regularization term that accommodates multi-class data sets and automatically determines the error bound of each base classifier. The regularization parameter penalizes the classifier when it misclassifies instances that were correctly classified in the previous iteration. The parameter additionally reduces the bias towards majority classes.

Experiments are conducted using 12 diverse data sets with moderate to high imbalance ratios. The results demonstrate superior performance of the proposed method compared to several state-of-the-art algorithms for imbalanced, multi-class classification.
problems. More importantly, the sensitivity improvement of the minority classes using RegBoost is accompanied with the improvement of the overall accuracy for all classes. With unpredictability regularization, a diverse group of classifiers are created and the maximum accuracy improvement reaches above 24%. Using stratified undersampling, RegBoost exhibits the best efficiency. The reduction in computational cost is significant reaching above 50%. As the volume of training data increase, the gain of efficiency with the proposed method becomes more significant.
ACKNOWLEDGMENTS

First, I would like to thank God for all his support. I have been blessed with many gifts in my life and I will always be thankful for him. I would like to thank all the individuals that supported me in my Ph.D. journey. I would like to thank and appreciate my advisor, Dr. Xiaohui Yuan, for all his support and advice. I am quite happy with what we have accomplished together. I would also like to thank all my committee members, Dr. Kathy Swigger, Dr. Jianguo Liu, and Dr. Song Fu, for all their advice and help. I would like to especially thank my wife, Hagar, for her patience, exclusive love, and infinite support. I will always love you. I would like to give very special thanks to my baby girl, Judy, for breaking several laptops, spilling liquids and food on my documents, and screaming all night. I love you so much my baby girl and I always will. I would like to thank my parents and sisters for their exclusive support. Their encouragement really supported me in this tough journey.
ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES viii

CHAPTER 1. INTRODUCTION 1
  1.1. Motivation 3
  1.2. Scope of Research 4

CHAPTER 2. RELATED WORK 6
  2.1. AdaBoost Extensions 6
  2.2. Sampling and Learning from Imbalanced Data 8
  2.3. Boosting Methods for Imbalanced Data Sets 13
  2.4. Real-World Scenarios 14

CHAPTER 3. REGULARIZED MULTICLASS BOOSTING 19
  3.1. Weighted Error Analysis 19
  3.2. Boosting Error Bound 21
  3.3. Regularization Parameter 22
  3.4. Multiclass Boosting Method 24

CHAPTER 4. EXPERIMENTAL RESULTS AND DISCUSSION 28
  4.1. Imbalanced Training 28
  4.2. Real-World Data Results 36
  4.3. Regularization Parameter Behavior 42
  4.4. Balanced Training 48

CHAPTER 5. CONCLUSION 55
LIST OF TABLES

4.1 Imbalanced data sets and properties. 29

4.2 The average sensitivity (in percentage %) of minority classes in multi-class problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training. 31

4.3 The average accuracy (in percentage %) in multi-class problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training. 32

4.4 The average sensitivity (in percentage %) of minority classes in binary classification problems. The numbers in parentheses are the standard deviation. 33

4.5 The average accuracy (in percentage %) in binary classification problems. The numbers in parentheses are the standard deviation. 33

4.6 Average number of effective classifiers for imbalanced sets. 35

4.7 Time (in second) used in training ensemble classifiers for imbalanced sets. 36

4.8 Real-world data sets and properties. 37

4.9 The average sensitivity (in percentage %) of minority classes in real-world problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training. 38

4.10 The average accuracy (in percentage %) in real-world problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training. 39

4.11 Average number of effective classifiers for real-world sets. 40

4.12 Time (in second) used in training ensemble classifiers for real-world sets. 40
4.13 Average sensitivity of AdaBoost.M1 and SAMME with stratified sampling and RegBoost. Column “Imp” gives the percentage of improvement using RegBoost. Parentheses indicates negative number. 45

4.14 Average accuracy of AdaBoost.M1 and SAMME with stratified sampling and RegBoost. Column “Imp” gives the percentage of improvement using RegBoost. Parentheses indicates negative number. 46

4.15 Balanced data sets and properties. 49

4.16 Average number of effective classifiers for balanced sets. 53

4.17 Time (in second) used in training ensemble classifiers for balanced sets. 54
LIST OF FIGURES

4.1 Average relative time taken in training for synthetic, UCI, and real-world data sets. The discontinued or incomplete curves indicate failure of generating an ensemble. 41

4.2 Regularization parameter $\delta$ changes along the training iterations. $\delta$ is initialized to one. The solid line depicts the $\delta$ of using decision tree as base classifier and the dash line depicts the $\delta$ of using neural network as base classifier. 43

4.3 Regularization parameter $\delta$ in binary class problems. The solid line depicts the $\delta$ of using decision tree as base classifier and the dash line depicts the $\delta$ of using neural network as base classifier. 44

4.4 Decision boundaries of three consecutive classifiers using (a) stratified sampling with AdaBoost.M1 and (b) RegBoost. Solid line draws the decision boundary for the first classifier; dash line draws the decision boundary for the second classifier; and dotted line draws the decision boundary for the third classifier. 47

4.5 Average error rate of training decision trees for all data sets using AdaBoost.M1 (AB), SAMME (SM), and RegBoost (RB1, RB2, RB3, and RBv). The error rate standard deviation is on the top of the bars. 50

4.6 Average error rate of training neural network for all data sets using AdaBoost.M1 (AB), SAMME (SM), and RegBoost (RB1, RB2, RB3, and RBv). The error rate standard deviation is on the top of the bars. 51
CHAPTER 1

INTRODUCTION

A massive amount of data are generated in real-world applications, which requires processing and categorization. Examples of such applications can be found in the fields of medicine and remote sensing. However, human analysis of such data requires a huge amount of time and effort. The accuracy of human analysis is also subject to concentration lapses and human errors. Additionally, the number of instances within different classes can be uneven, which creates a difficult learning process for finding unbiased decision boundaries. Algorithms were developed to improve efficiency and accuracy of learning from different data. However, training a single model might not achieve an improved performance especially for imbalanced data sets. With the increasing need for improved performance for imbalanced as well as balanced data, ensemble learning was adopted. The improved accuracy that was targeted using an ensemble of classifiers relies on the diversity among the decision boundaries created by these classifiers. However, if this diversity is not achieved, the training process degrades the ensemble into literally a single classifier or a very small number of classifiers such that the spirit of ensemble is lost. Additionally, the ensemble learning process require elongated training time to be able to train multiple classifiers. The uneven instances distribution among different classes in a data set further complicates the learning process. An important example for such data sets is capsule endoscopy. A tiny pill-shaped device is swallowed by a patient who might be suffering from certain diseases in the small intestine. The device has a small camera that captures a video through the digestive tract of the patient, which can take up to eight hours. Capsule endoscopy visualizes bleeding, tumor, and polyps that are hard for other endoscopy methods to visualize. However, a video for one patient has thousands of images to be processed. In this case, the classifiers spend a large amount of time processing a video for just one patient to classify the images into either normal or abnormal classes. These abnormal images are very few in number compared to thousands of normal images that are available. This creates an imbalanced data set where the class
with the diseased images is referred to as the minority class and the other class with normal images is referred to as the majority class. In addition, multi-class classification becomes more common to classify the diseased images into their respective disease types.

Different methods have been suggested to handle the imbalance problem. These techniques, however, focused on binary classifications or alternatively binarizing a multi-class classification. Existing methods to address the imbalance problem can be classified as data-level approaches, where the data is preprocessed using sampling, and algorithmic-level approaches, where algorithms were developed specifically to deal with imbalanced data such as cost sensitive learning [47, 102, 14, 96]. In data-level approaches, undersampling or oversampling is applied in order to balance the minority and majority classes. The oversampling process differs in the way synthetic samples are created. Some techniques created synthetic samples randomly while others created synthetic samples based on density distribution [98] or distance to the decision boundary [45]. Cost sensitive learning methods assign higher costs to the minority class [31]. Hybrid methods, such as boosting, were developed by integrating both sampling and algorithmic approaches to handle the imbalanced data sets. Boosting methods train an ensemble of classifiers in multiple iterations and combine the classifiers into one strong learner. The boosting algorithm has been adapted to address the imbalance problem. Some algorithms either applied sequential oversampling [15] or undersampling [85] within the framework of the boosting algorithm to balance the classes. Other boosting algorithms modified the weight updating rule in order to assign higher weights to misclassified minority instances during each boosting iteration [39]. Most of these methods were developed to address the binary imbalanced classification problems where there exists one minority and one majority class. Recently, boosting methods have been extended to address the more severe case of multi-class imbalance, where several minority and majority classes are present [100, 114]. More details can be found in Chapter 2.

Among the ensemble learning strategies, AdaBoost [33] proved to be most successful in many applications. AdaBoost [33] was originally introduced to sequentially train an ensemble of classifiers on binary data sets in order to achieve an improved accuracy through
an error minimization function. After each training iteration, the boosting method adjusts the weights of the instances by assigning higher weights to misclassified instances and lower weights to correctly classified ones through an exponential loss function. Additionally, the base classifier is assigned a weight based on its performance to express its contribution to the overall decision made by all base classifiers. Based on its success on binary classification applications, AdaBoost was extended to accommodate multi-class data sets. The extensions differ in applying direct transformation or converting the problem into multiple binary classifications.

1.1. Motivation

The conversion to multi-class classification belongs to one of two techniques, the indirect and the direct conversions. The indirect conversion transforms the multi-class into multiple binary classifications using the binarization methods which convert a multi-class classification to one vs. one (OvO) or one vs. all (OvA) classifications such as AdaBoost.M2 [33] and AdaBoost.MH [84]. The direct conversion applies the boosting method to multi-class data sets by extending the exponential loss function of the method. AdaBoost.M1 [33] was introduced to train multi-class data sets in which the error condition for each classifier in the ensemble was too strict compared to random error guessing and in particular with a large number of classes. Stage-wise additive modeling using multi-class exponential (SAMME) loss function [118] extended the loss function to ease the error bound of AdaBoost.M1 by transforming it to that of random guessing of the number of classes in the data set.

Despite the wide success of AdaBoost and its multi-class extensions, these methods suffer several problems in dealing with imbalanced data. First, the performance is deteriorated when the methods learn from multi-class data sets. For instance, an error condition of 0.5 for each weak classifier is far more strict than the error rate of random guessing of multiple classes. With such condition the algorithm is not able to employ the loss function to increase the weights of the misclassified instances. Additionally, an error bound that is equivalent to random guessing can result in unacceptable accuracy with large number of
classes. Second, the uneven number of instances among different classes affects the learning process negatively and introduces additional bias to the majority classes. Third, the algorithm suffers an early termination problem. This problem is associated with the repetition of misclassified instances. The repetition significantly increases the weights of these instances and increases the weighted error of the algorithm above the allowed error bound. The reason for this repetition is attributed to employing stable weak classifiers in the algorithm. Fourth, boosting methods suffer elongated training time using its iterative scheme. Moreover, the binarization methods used for multi-class extensions require extended training time with multiple training iterations and the improvement of accuracy is limited.

1.2. Scope of Research

The goal of this dissertation is to address the aforementioned challenges using a multi-class boosting framework to improve the learning performance of imbalanced as well as balanced data sets. The contribution of this research includes the verification of the following hypothesis and the introduction of an improved multi-class boosting method.

Hypothesis: Employment of intelligent sampling and unpredictability regularization in a boosting framework helps efficient learning from imbalanced, multi-class data sets.

Approach: The proposed multi-class boosting method, referred to as RegBoost, employs a class-based weighted stratified sampling along with a modification to the exponential loss function by adding an automated regularization parameter. The sampling procedure is performed on each class separately based on its instances weight distribution in order to balance the classes. Additionally, the sampling process focuses on instances close to the decision boundaries and, hence, is able to reduce the training time. The regularization parameter accommodates multi-class data sets and automatically adjusts the error bound of each classifier in accordance to its performance in the training process. The parameter additionally penalizes the weight of the classifier when it misclassifies instances that were correctly classified in the previous iteration. This strategy aims at reducing the bias towards majority classes in case of learning from imbalanced sets. Moreover, the proposed method avoids the early termination problem by altering the instances distribution during training and, hence,
avoids repetition of misclassified instances. Based on the destabilization introduced to the learning classifiers, the introduced boosting framework allows the employment of any type of base classifiers.

The rest of this dissertation is organized as follows: Chapter 2 provides a literature survey of boosting methods and the imbalanced learning problem. Chapter 3 identifies the early termination problem, derives the regularization parameter, and introduces the multi-class boosting framework. Chapter 4 discusses the experimental results. Finally, Chapter 5 concludes this dissertation.
2.1. AdaBoost Extensions

Boosting methods and in particular AdaBoost\cite{33} has proven to be successful in improving performance of learning from binary data sets in multiple applications. AdaBoost alters the weights of different instances in order to change the data distribution by assigning higher weights to misclassified instances and lower weights to correctly classified ones. The process aims at improving the overall accuracy by minimizing a convex loss function. The improvement achieved using boosting methods was analyzed by Friedman et al. \cite{35} where a maximum Bernoulli likelihood was used to view boosting as an additive modeling approximation on the logistic scale. A discussion of an opposing view is provided in \cite{71} debating whether AdaBoost minimizes a surrogate loss function or the misclassification error. Reviews of selected boosting and ensemble methods can be found in \cite{82, 34, 83, 72, 12}. Boosting methods were extended to learn from multi-class data sets. These methods can be divided into indirect and direct multi-class extensions.

Indirect multi-class boosting extensions were proposed to deal with multi-class data sets. AdaBoost.M2 \cite{33} and AdaBoost.MH \cite{84} were introduced as indirect multi-class extensions to AdaBoost. They converted the multi-class problem into multiple binary classification problems. A hamming loss function was introduced to represent an average error rate for the weak hypothesis over all the binary predictions. The class label is added as an additional feature for the instances. The goal was to reduce the hamming loss. Jun and Ghosh \cite{51} proposed a multi-class boosting method that converts the multi-class problem to multiple problems using a hierarchy that depends on the closeness of classes to each other. Saberian and Vasconcelos \cite{80} developed two multi-class boosting algorithms which differ in the weak classifiers they support. One of them is based on coordinate descent while the other is based on gradient descent. Their framework originated from multi-dimensional codewords and predictors. Eibl and Pfeiffer \cite{28} proposed two boosting methods for multi-class classification.
Compared to AdaBoost.M2, the two algorithms minimize the confidence rated error instead of the pseudo-loss and converge in a faster manner. Yet, the methodology is not designed to minimize the training error. Li [61] introduced a multi-class boosting method with a modification of the loss function. The method adaptively chooses a base class and derives algorithms for the remaining classes.

Direct multi-class extensions were proposed to efficiently deal with the multi-class classification problem. However, finding the optimal direct conversion is not straightforward. AdaBoost.M1 [33] was introduced to directly extend AdaBoost to multi-class classification. The extension enforced a fixed error bound of 0.5 for each weak classifier for any number of classes. Mukherjee and Schapire [73] used a drifting games framework to develop an adaptive boosting method. Huang et al. [48] introduced gentle adaptive multi-class boosting learning (GAMBLE) which extends the binary gentle AdaBoost algorithm [35] to multi-class classification. The method used direct transformation from binary to multi-class classification by fitting regression functions to the data, employing active learning-based sampling, and utilizing a multi-class loss function. The methodology allows the classes to be likely imbalanced. Abouelenien and Yuan [1] developed a direct multi-class boosting extension that ensures diversity between different classes using instances weight analysis. Zhu et al. [118] proposed stagewise additive modeling using multiclass exponential loss function, SAMME. For $C$ number of classes, the algorithm assumes a maximum error bound of $(C - 1)/C$ for each weak classifier which corresponds to the error rate of random guessing of $C$ classes. On learning from a binary class data set, SAMME reduces to AdaBoost. Abouelenien and Yuan [3] introduced the idea of combining undersampling with boosting at each iteration according to the data distribution. The algorithm employed a constant error parameter in the range of 1 to $C - 1$ with different undersampling sizes to improve the performance of learning from multi-class imbalanced sets. Kalai and Servedio [52] additionally modified the boosting algorithm to avoid noisy data. Kanamori et al. [53] proposed a truncation of part of the loss function to avoid the effect of outliers on the boosting method.
2.1.1. Review of AdaBoost and SAMME

This section reviews three of the most successful boosting methods for binary and multi-class classifications, namely, AdaBoost, AdaBoost.M1, and SAMME. The boosting methods combine several weak classifiers into a stronger learner that is more accurate in predicting the correct labels of unseen instances. At each iteration, the weak classifier is evaluated using the training set and the weighted error is calculated. Each weak classifier is assigned a weight based on its evaluation to express its strength in the overall decision. The weight is then used by the exponential loss function to alter the weight distribution in order to assign higher weights for misclassified instances and lower weights for correctly classified instances. After training all iterations, a strong classifier is built using weighted majority voting. The difference between the methods is found in the exponential loss function which reflects in the error bounds of each method as seen in Algorithm 1.

Assume having a training set \((x_1, y_1), \ldots, (x_N, y_N)\) where the instances \(x_i \in \mathcal{X}\), the class labels \(y_i \in \mathcal{Y} = \{1, \ldots, C\}\), and \(N\) is the total number of instances. The training data is assumed to be distributed from an unknown probability distribution.

\(E\) is equal to \(\frac{C-1}{C}\) for SAMME and 0.5 for AdaBoost and AdaBoost.M1. \(\gamma\) is equal to \((C-1)\) in case of SAMME and 1 in case of AdaBoost and AdaBoost.M1. Function \(1_{y_i \neq f_t(x_i)}\) is an indicator function that returns 1 if true and 0 otherwise. The exponential loss function reduces to

\[
e^{(\alpha_t 1_{y_i \neq f_t(x_i)})} = \begin{cases} < 1 & \text{if } y_i = f_t(x_i); \\ > 1 & \text{if } y_i \neq f_t(x_i). \end{cases}
\]

2.2. Sampling and Learning from Imbalanced Data

Sampling has been widely used in machine learning algorithms. To estimate the sample size, most of the methods, however, use prior information about the data population. This is the main issue, that prior information must be obtainable, which is not always the case. Lenth [58] proposed practical guidelines to effectively specify a sample size without such information available. Adcock [5] provided a review of sample size determination where methods were categorized as either frequentist or Bayesian. The difference relies in dealing
Algorithm 1 Boosting Algorithms

1: Initialize data distribution weights \( w_0(i) = 1/N, i = 1, \ldots, N \).

2: for \( t = 1, \ldots, T \) : do

3: Train a weak classifier \( f_t \) according to the weight distribution.

\[ \epsilon_t = \sum_{i=1}^{N} w_{t-1}(i) 1[y_i \neq f_t(x_i)] \]

4: if \( \epsilon_t > \epsilon \) then

5: then return \( \alpha_t = 0 \)

6: else

7: Set a weak learner weight \( \alpha_t \)

\[ \alpha_t = \frac{1}{2} \log\left(1 - \frac{\epsilon_t}{\epsilon_t}ight) + \frac{1}{2} \log(\gamma) \]

8: Update the data distribution weights

\[ w_t \leftarrow w_{t-1} \cdot e^{(\alpha_t 1[y_i \neq f_t(x_i)])} \]

9: Normalize \( w_t \) to convert to a probability distribution

10: end if

11: end for

12: Combine weak learners \( f_t \) into a strong classifier \( F(x) \)

\[ F(x) = \arg \max_y \sum_{t=1}^{T} \alpha_t f_t(x) \]

with uncertainty in prior information. Boonyanuntha and Zeephongseku [10] discussed the relation between the predictive power of classifiers and choosing a sample size. Nguyen et al. [77] proposed methodologies to achieve a trade-off between accuracy and sample size under interval uncertainty. The importance of determining the sample size also arises in the medical field. Maxwell et al. [70] discussed factors when planning a sample size for statistical power and accuracy in parameter estimation. Eng [29] provided some methods in the calculation
of the required sample size. Abouelenien and Yuan [4] studied the undersampling sizes to associate with boosting.

Imbalanced data sets are defined based on the ratio of the sizes of different classes for a given data set [57]. Existing methods to improve the imbalance problem can be divided into algorithmic-level approaches and data level approaches. Algorithmic-level approaches proposed to handle the binary imbalance problem improved classification algorithms in order to relocate the decision boundary to achieve higher accuracy for the minority class. Chen et al. [103] modified support vector machines (SVM) by iteratively pruning the support vectors of the majority class. Liu et al. [62] developed a class confidence proportion decision tree using a class confidence metric to improve the sensitivity. Yang et al. [110] employed penalty regularization and margin compensation as two regularization factors to improve the minority class performance. Tang et al. [90] developed cost sensitive learning method based on SVM. Williams et al. [104] used remote sensing data sets for mine classification to apply the infinitely imbalanced logistic regression. Diamantini and Potena [24] used bayes vector quantizer (BVQ) algorithm [25] which is a learning algorithm that uses a labeled vector quantizer to define the nearest neighbor decision rule. BVQ adapts this rule to the optimum Bayes decision rule to improve the minority performance. Su and Hsiao [88] used Mahalanobis-Taguchi System [89] to evaluate its robustness for the imbalance problem. Wu and Chang[105] introduced kernel-boundary-alignment algorithm by modifying the kernel matrix to eliminate the bias towards the majority class. Wu and Amari[107] adopted conformal transformation of kernel functions that use a data-dependent way of extracting the information of the support vectors. Cieslak and Chawla[20] proposed Hellinger distance as the splitting criterion for decision tree for class imbalance. Nguyen et al. [76] provided a review and analysis of pattern classification tasks with binary imbalanced data sets. They combined supervised and unsupervised learning using clustering to train feed forward neural network to achieve improvement for the minority class.

Data-level approaches proposed to handle the binary imbalance problem relied on resampling the data to balance the two classes. Kubat and Matwin [57] removed a set of
majority instances that were close to the boundary, redundant, and noisy using one-sided selection. Chawla developed SMOTE [15], a technique that creates synthetic minority instances to balance the minority and majority classes. Borderline-SMOTE [45] extended the SMOTE idea to intelligently oversample the minority instances close to the decision boundary. Lerner et al. [59] raised the classification accuracy for a small database of fluorescence in situ hybridization signals to determine genetic abnormalities by oversampling minority instances using monolithic strategy. Li et al. [60] employed oversampling and undersampling in combination to improve classification of medical data sets. The majority class was undersampled using the Gaussian type fuzzy membership function and alpha cut and the minority class was oversampled to create virtual instances using the mega trend diffusion membership function. Wasikowski and Chen [101] used feature selection to reduce dimensionality of imbalanced data sets as well as to gain higher accuracy for the minority class. Drown et al. [26] developed a genetic algorithm-based data sampling method to improve the software quality modeling and address the imbalance problem in high-assurance systems. Yen and Lee [111] introduced a cluster-based undersampling approach using back-propagation neural network and investigated the effect of undersampling on the class distribution. Batuwita and Palade [8] proposed a method that resamples data by first selecting the most informative instances determined by the hyperplane of an SVM classifier on the original data. Second, these instances are used to resample the data and balance the minority and majority classes. Barua et al. [7] developed majority weighted minority oversampling technique (MWMOTE), to efficiently handle imbalanced data by specifying weights for selected hard-to-classify minority instances based on the Euclidean distance from the nearest majority instance. A clustering method is then employed to generate synthetic minority instances based on their respective weights. Garcia et al. [38] tried evolutionary algorithms to propose a method that handles the imbalance by storing objects in the Euclidean space. New instances are classified by measuring their distances to the closest generalized exemplar. The algorithm selects the best generalized exemplars for optimization. Piras and Giacinto [78] utilized the nearest neighbor paradigm to artificially create instances in the feature space of the minority class
in agreement with its local distribution for the image retrieval domain. Wang et al. [98] introduced an oversampling method based on data density. The technique adaptively creates different number of new instances around each minority instance based on its difficulty. Yu et al. [112] proposed an undersampling algorithm that eliminates noise using feature selection and applies ant colony optimization algorithm to exclude majority instances that are less informative in order to balance the classes. Zhou [117] investigated the effect of different oversampling and undersampling methods for bankruptcy prediction. The experiments concluded that the proper sampling method depends on the number of instances in the minority class. Cao et al. [13] introduced a structure-preserving oversampling method combined with interpolation-based oversampling to create synthetic minority instances using multivariate Gaussian distribution. The method expands the minority instances in the empty area of the instance space by keeping the covariance structure of the minority class and creating protective variances in the Eigen dimensions. Lopez et al. [64] conducted a comparative study of the data-level approaches such as SMOTE with algorithmic-level approaches such as cost-sensitive models that assign higher weights to minority instances. Additionally, the study compared a hybrid approach that combines both methods. The comparison concluded that both techniques are equivalent in effectiveness and that the hybrid approach has improved performance in some cases.

Recently, attention was driven to the more severe case of multi-class imbalance by manipulating different classifiers. Murphey et al. [74] proposed a pattern classification algorithm using neural network to learn from multi-class imbalanced data sets to improve the accuracy of minority classes compared to the OvO and OvA classification methods. Bae et al. [6] proposed a mix-ratio sampling procedure to handle the multi-class imbalance problem. The procedure considers minority classes that can have higher accuracy once oversampled. The method employed SVM to determine varying oversampling sizes to different minority classes. Ghanem et al. [42] developed a Multi-IM method to address the imbalance problem. The method extends probabilistic relational technique, PRMs-IM [41], by embedding the balancing concept of the method to apply to multi-class data sets. Navarro et al. [30]
proposed a dynamic oversampling procedure in a memetic algorithm that utilizes radial basis functions neural network. The procedure resamples the data in two stages. First, the minority instances are oversampled to partially balance the classes. Then the memetic algorithm is applied to oversample the data and generate new patterns for the least sensitivity class. Tong et al. [91] proposed an analytical procedure to specify the resampling scheme by utilizing response surface and design of experiments methods. Areibi and Tempelman [22] proposed a dynamic sampling framework that automatically tunes the training set distribution by combining sampling techniques such as SMOTE, random undersampling, and random oversampling methods. Jeatrakul and Wong [49] developed one-against-all with data balancing (OAA-DB) that combines undersampling technique using complementary neural network with the oversampling technique SMOTE using the OVA approach. Fernandez et al. [31] compared the classification performance of multi-class imbalanced data sets using binarization techniques such as OVO and OVA to the performance of preprocessing the instances and to the performance of cost sensitive learning with ad hoc approaches. The comparison concluded that cost-sensitive learning is more robust and has improved performance.

2.3. Boosting Methods for Imbalanced Data Sets

Hybrid methods, such as boosting, have been adapted to handle binary imbalanced data sets. Boosting methods combine algorithmic and data-level approaches to handle the imbalance problem. Karakoulas and Shawe-Taylor [54] proposed AdaUBoost where the weights updating rule and the loss function were modified to assign higher weights to minority instances within a boosting framework. Chawla et al. [15] integrated SMOTE with the boosting procedure which iteratively trains the balanced data after adding the randomly created synthetic instances. Chen et al. [17] introduced RAMOBoost, an oversampling procedure for the minority instances. The technique ranks the minority instances at each training iteration based on a sampling probability distribution. Seiffert et al. [85] alleviated class imbalance by using a weight updating rule with an undersampling strategy in their method RUSBoost. Wang and Japkowicz [97] used asymmetric misclassification cost with SVM for minority classification improvement. Geiler et al. [39] developed DataBoost-IM
where they employed an adaptive sampling ensemble classifier to learn from imbalanced data sets. E-Adsampling [39] algorithm created synthetic samples and updated the instances weights during training iterations to improve the accuracy of imbalanced data sets. The difference between both methods is that DataBoost-IM leads to the creation of a large number of minority synthetic instances for imbalanced data sets, while E-Adsampling could lead to the loss of the originally misclassified instances. Dubout and Fleuret [27] proposed algorithms that adaptively sample both instances and features during each iteration to reduce the computational cost. Liu et al. [63] incorporated an ensemble of SVMs with oversampling and undersampling techniques combined to improve the accuracy of the minority class with highly skewed data. Saadi et al. [79] developed a classifier that applies multiple SVM classifiers to ensembles of the data to balance the minority and majority classes for highly imbalanced data sets in the biomedical field. He et al. [46] proposed ADASYN, a method that utilizes the weights of minority instances to generate more synthetic samples towards harder-to-classify minority instances compared to easier instances. Yuan and Ma [113] proposed using a sampling-reweighing strategy to tune AdaBoost towards a specific performance by initially oversampling the data followed by adjusting the weights of the classifiers using genetic algorithms.

Recently, advanced boosting methods have been proposed to deal with multi-class imbalanced data sets. Wang and Yao [100] studied the effect of multi-minority and multi-majority classes on the learning process. They concluded that multi-majority classes pose increased harm to the learning process. The study additionally applied AdaBoost.Nc [99] on multi-class imbalanced data sets by using a negative correlation learning algorithm that utilizes an ambiguity term to add explicit diversity. Yuan and Abouelenien [114] proposed a boosting framework using weighted sampling to achieve improved performance for the minority classes using multi-class imbalanced face recognition.

2.4. Real-World Scenarios

Real-World data is vital in determining the validity and efficacy of a learning algorithm. An improved performance of these algorithms on such data is considered as an
indicator of a successful learning process. This section reviews related work for two important Real-World applications, face recognition and capsule endoscopy (CE). Face recognition is an example of data sets with large number of classes which negatively affects the performance. CE is an example of the massive data sets that suffer from a great imbalance ratio.

2.4.1. Face Recognition

Methods were developed to improve the accuracy and robustness of automated face recognition. Eigenfaces [92] is a state-of-the-art algorithm that presents one of the earliest attempts to improve face recognition. The method employs principal component analysis (PCA) to project the training data on a lower dimensional space followed by a nearest neighbor classification scheme. Linear discriminant analysis, LDA, was also used in Fisherfaces [9] to recognize face images in a supervised manner. LDA finds linear combinations of features that can separate between classes efficiently. The algorithm maximizes the between-class scatter matrix while minimizing the within-class scatter matrix where each class contains images of one subject. This is achieved by maximizing the ratio which is referred to as Fisher criterion. Sharkas and Abouelenien [86] provided a comparative study on Eigenfaces, Fisherfaces, and independent component analysis for face recognition. A review of advances in the field of face recognition is provided in [56]. Many methods were developed based on the Eigenfaces and Fisherfaces to improve the multi-class face recognition problem.

One important aspect of face recognition data sets is the large number of classes. Face images are required to be recognized as belonging to a subject out of tens or hundreds of other subjects. Boosting methods have been developed to improve the performance of face recognition. Gao et al [37] trained Gabor feature classifier using random subspace for face recognition. Their method generated several subspaces randomly from the original Gabor feature space to reduce the dimensionality on the FERET database. To achieve this, the multi-class problem was converted to multiple binary classifications using AdaBoost framework. Guo and Zhang [44] trained AdaBoost to recognize faces with constrained majority voting strategy for the multi-class recognition problem. James and Annadurai developed a weakness analysis theory in a boosting framework to overcome the traditional boosting
limitations in face recognition. Jones and Viola [50] extracted local features for face recognition using rectangular filters which compare regions of the images at different orientations, and scales. Using the selected features, they used a face similarity function to train AdaBoost. Lu and Plataniotis [65] introduced a two-stage hierarchical boosting framework for face recognition on large-scale databases. The method used LDA-based clustering with a novel separability criterion to divide the training set into smaller clusters.

With the unavailability of images for some subjects (classes) in real-world scenarios, face data sets become imbalanced and suffered low sample size per class. Wu and Zhou [106] introduced a modification to Eigenfaces method called \((PC)^2A\) which applies PCA to the original face image combined with its vertical and horizontal projections in a one-image-per-class scenario. Gao et al. [109] decomposed the face image into a smooth general appearance image and difference image using singular value decomposition to generate the between-class scatter matrix and the within-class scatter matrix which cannot be created from a single image per class. Chen et al. [18] partitioned each face image to a set of non-overlapping sub images with the same dimensionality to have several images in each class and, thereby, be able to apply LDA. Martinez [68] proposed a new distance approach which automatically assigns weights to the image pixels based on the similarity of areas of muscular activity of test images to the only one training image per class. Singh et al. [87] extracted phase textural features using 2D log polar Gabor transform in a dynamic neural network architecture to increase the accuracy of disguised face images and single gallery images. Xu and Yang [108] generated additional images by mean filtering for each class by developing a feature extraction method called Local Graph Embedding Discriminant Analysis. This algorithm maximizes the class separability while preserving local neighborhood relationships of the images. Ng and Chen [75] calculated an expression invariant similarity vector for single image per class by using the similarities between an input face and training faces of similar poses and expressions in the PCA space. This similarity vector is used for comparing and recognizing face images with other poses and expressions. Other face recognition approaches were introduced to combine dimensionality reduction and image feature extraction methods [16, 115, 19]. The
orthogonal subspaces were used to generate additional images so that each class has more representative instances. Similarly, [23] employed image filters, e.g., Gabor filters, to extract frequency related features space and synthesize subband images for the training process. Masip et al. [69] introduced a face recognition boosted online learning algorithm where new classes were incrementally added to the previously trained classes. Salehi et al. [81] combined boosting with regularized LDA, which is robust to the LDA small sample size problem, using a nearest center classifier for classification. Ma et al. [66] converted the face recognition multi-class problem to a two-class problem of intra- and extra-class representation using boosting methods. They trained local binary pattern feature-based classifiers and used a cascade framework to overcome the massive imbalance between the intra- and extra-classes.

2.4.2. Capsule Endoscopy

CE is an example of the massive data sets that require improved algorithms whether to achieve segmentation of different organs or to detect abnormality in the digestive tract. The segmentation process is conducted among thousands of images to classify the images relative to different body organs. The abnormality detection is required to detect few abnormal images among the thousands of images which creates a huge imbalance.

Most of the approaches developed to classify CE images relied on extracting discriminatory features from the images. Histogram of texture and color were proposed in [55]. They were integrated in an adaptive neurofuzzy framework to diagnose images. Magoulas et al. [93] proposed a double stage unsupervised k-windows clustering for classification of CE images. Coimbra et al. [21] detected abnormality in CE images using MPEG-7 visual descriptors. Haar features were selected in [36] inside a cascaded AdaBoost framework to detect intestinal lumen. Giritharan et al. [43] applied incremental SVM to learn from new data in the segmentation process of CE. Vilario et al. [95] proposed a sequential design using SVM classifiers with analysis of color, textural, and blob features to detect intestinal contraction in CE images. Ravesteijn et al. [94] presented a computer-aided detection (CAD) system of polyps in computed tomography colonography. The system has two stages, candidate detection and supervised classification by logistic regression. Zhao et al. [116] proposed
fusing multiple statistical classifiers using texture, edge, and color features using a CAD system. Abouelenien and Yuan [2] proposed an automated system that employs sampling within a boosting framework to categorize CE images and to detect abnormality.
CHAPTER 3

REGULARIZED MULTICLASS BOOSTING

This chapter describes the motivation behind this work by explaining the early termination problem followed by deriving the regularization parameter. This is followed by discussing the boosting error bounds and introducing the proposed multi-class boosting method.

3.1. Weighted Error Analysis

This section starts with the motivation behind this method using a weighted error analysis. The improvement achieved using AdaBoost and its multi-class extensions relies on the variation in the decision boundaries provided by individual base classifiers, i.e. diversity of classifiers. However, the spirit of ensemble learning is lost if these classifiers result in creating the same decision boundary. In fact, the boosting algorithm terminates as early as the second iteration if the first two weak classifiers create the same decision boundary. Once this occurs, the weights of the repeatedly misclassified instances increase significantly and push the weighted error above the allowed bound. Hence, the algorithm terminates and converts to a single classifier that requires longer training time. This problem is more prevalent when training stable base classifiers. The algorithms presented in [33, 118] are followed. The analysis can then be easily transformed to AdaBoost, AdaBoost.M1 and SAMME algorithms. The analysis proves that once the same group of instances is repeatedly misclassified, the algorithm terminates regardless of the base classifier accuracy.

After the data set is trained in the first iteration, assume that $m$ instances were misclassified, $m \in \{1, \ldots, N\}$. The weighted error is then calculated according to $\epsilon_1 = \frac{m}{N}$. After training the second iteration, assume the same $m$ instances are repeatedly misclassified. Accordingly, the weights for the misclassified instances are

$$w_1(i) = \frac{\epsilon_1^{\alpha_1}}{N\tilde{W}_1}$$
and for the instances that are correctly classified to

$$w_1(i) = \frac{e^{-\alpha_1}}{NW_1}$$

$W_1$ represents a normalization factor, $W_1 = \sum_1^N w_1 = \frac{m}{N} e^{\alpha_1} + \frac{N-m}{N} e^{-\alpha_1}$. $\alpha$ represents the weight assigned to individual weak classifiers and is calculated according to their evaluation on the training set.

$$\alpha_1 = \frac{1}{2} \log\left(\frac{1-\epsilon_1}{\epsilon_1}\right) + \frac{1}{2} \log(\gamma) \quad (2)$$

$\gamma$ is a constant equal to 1 for AdaBoost and AdaBoost.M1, and to $C - 1$ for SAMME algorithm. The weighted error is then updated to

$$\epsilon_2 = \frac{\frac{m}{N} e^{\alpha_1} + e^{-\alpha_1} - \frac{m}{N} e^{-\alpha_1}}{e^{2\alpha_1} + \frac{N}{m} - 1} \quad (3)$$

After substitution with the values of $\epsilon_1$ and $\alpha_1$,

$$\epsilon_2 = \frac{(\frac{N}{m} - 1)(\gamma)}{(\frac{N}{m} - 1)(\gamma) + (\frac{N}{m} - 1)} \quad (4)$$

The weighted error is simplified to

$$\epsilon_2 = 1 - \frac{1}{\gamma + 1} \quad (5)$$

After substitution with the $\gamma$ value,

$$\epsilon_2 = \begin{cases} 0.5 & \text{AdaBoost} \\ 0.5 & \text{AdaBoost.M1} \\ 1 - \frac{1}{C} & \text{SAMME} \end{cases} \quad (6)$$

The analysis shows that for AdaBoost and its multi-class extensions, the weighted error reaches its maximum bound once the same misclassified instances are repeatedly misclassified. This results in termination of the algorithms and conversion to a single training process.
3.2. Boosting Error Bound

In AdaBoost.M1 [33], the weak classifier is assigned a weight following training, based on its evaluation on the training set. The weight function is given by

\[ \alpha_t = \frac{1}{2} \log\left( \frac{1 - \epsilon_t}{\epsilon_t} \right). \]

Where \( \epsilon_t \) is the weighted error of training iteration \( t \) calculated as the summation of the weights of misclassified instances. This weight presents the strength of the contribution of the corresponding weak classifier to the overall decision made on unseen data. The weighted error is the controlling factor of \( \alpha \). In order to avoid a negative weight, the following condition must hold

\[ 1 - \epsilon_t > \epsilon_t \]

Hence, the weighted error bound is

\[ \epsilon_t < 0.5 \]

Following SAMME algorithm [118], the boosting exponential loss function is extended by adding a constant term to the weight function. This term is presented by the logarithm of the constant \( C - 1 \), which depends on the number of classes in the data set.

\[ \alpha_t = \frac{1}{2} \log\left( \frac{1 - \epsilon_t}{\epsilon_t} \right) + \frac{1}{2} \log(C - 1). \]

Accordingly, the condition to avoid a negative weight is modified to

\[ (1 - \epsilon_t)(C - 1) > \epsilon_t \]

Hence, the weighted error bound becomes

\[ \epsilon_t < \frac{C - 1}{C} \]

AdaBoost.M1 accounts for the deterioration presented in the weighted error while SAMME adds a constant to the weight function that sets the error bound of each weak classifier to that of random guessing of \( C \) classes. A possibility exists that some instances are misclassified after being correctly classified by the previous weak classifier. In the case of imbalanced learning, a weak classifier can then correctly classify most majority instances
on the expense of minority instances that were correctly classified in previous iterations and, hence, the decision boundary is not adjusted correctly.

3.3. Regularization Parameter

To address the aforementioned considerations and accommodate multi-class data sets, this research introduces a regularization parameter that is added to the convex loss function via the classifier weight function. The parameter penalizes the classifier if it misclassifies instances that were correctly classified in previous iterations. The penalty is determined by the weights of these instances. This parameter varies during each iteration based on the evaluation of the weak classifiers.

After each training iteration, each weak classifier is evaluated and the weight function is calculated according to Equation (8). The regularization parameter $\delta$ is initialized to 1. The weight function is equivalent to that of SAMME in the first iteration.

\[
\alpha_t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) + \frac{1}{2} \log(\delta_t(C - 1)).
\]  

The exponential loss function adjusts the instances weights to increase weights of misclassified instances and reduce weights of correctly classified ones as

\[
w_t = \begin{cases} 
  w_{t-1}(i)e^{-\alpha_t} & \text{for correctly classified instances} \\
  w_{t-1}(i)e^{\alpha_t} & \text{for misclassified instances.}
\end{cases}
\]

The weights are then normalized. In the next iteration, i.e., $t \rightarrow t-1$ and $t-1 \rightarrow t-2$, a weak classifier is trained and evaluated. The misclassified instances are determined and the weighted error is calculated as

\[
\epsilon_t = W_c + W_m.
\]

$W_c$ is the weighed error summation for the $m_1$ instances that are misclassified by the current weak classifier and were correctly classified by the previous one. Such instances are named
second-round-misclassified instances:

\[
W_c = \sum_{i \in m_1} w_{t-1}(i) = \sum_{i \in m_1} w_{t-2}(i)(\delta_{t-1}(C - 1)(1 - \epsilon_{t-1}))^{\frac{1}{2}}. \tag{11}
\]

\(W_m\) is the weighed error summation for the \(m_2\) instances that are misclassified by the current weak classifier and were also misclassified by the previous one:

\[
W_m = \sum_{i \in m_2} w_{t-1}(i) = \sum_{i \in m_2} w_{t-2}(i)(\delta_{t-1}(C - 1)(1 - \epsilon_{t-1}))^{\frac{1}{2}}. \tag{12}
\]

The goal of the iterative boosting method is to adjust the decision boundary to correctly classify hard-to-classify instances. However, this adjustment is not supposed to misclassify instances that were correctly classified by earlier classifiers. The parameter \(\delta_t\) adjusts the classifier’s weight to penalize the one that misclassifies second-round-misclassified instances. It measures the difference in the weighted error if all currently misclassified instances were also misclassified by the previous iteration. In this case, to derive an expression for \(\delta_t\), the assumed weighted error is equivalent to:

\[
\epsilon_t' = \sum_{i \in (m_1 + m_2)} w_{t-2}(i) \left(\frac{\delta_{t-1}(C - 1)(1 - \epsilon_{t-1})}{\epsilon_{t-1}}\right)^{\frac{1}{2}}. \tag{13}
\]

The instances weights are re-normalized accordingly. The actual weighted error is equivalent to the assumed weighted error \(\epsilon_t'\) multiplied by the parameter \(\delta_t^{\frac{1}{2}}\). The parameter provides a measurement for the deterioration achieved from the second-round-misclassified instances. The value of the parameter is proportional to these instances weights:

\[
\epsilon_t = \epsilon_t' \delta_t^{\frac{1}{2}}. \tag{14}
\]

Since \(\epsilon_t \leq \epsilon_t'\), the parameter is less than or equal to 1, i.e., \(\delta \leq 1\).
By substituting Equation (14) in Equation (13), the regularization parameter $\delta_t$ can be represented as

$$
\delta_t = \frac{\epsilon_t^2 \epsilon_{t-1}}{(\sum_{i=1}^{m_1+m_2} w_{t-2}(i))^2(1 - \epsilon_{t-1})\delta_{t-1}(C - 1)}.
$$

The weight function is adjusted by integrating the logarithm of $\delta_t$, which is always less than or equal to 0. Hence, this regularization term penalizes the classifiers that result in second-round-misclassification as shown in Equation (9). $\delta_t$ varies during each iteration based on the performance of the two consecutive weak classifiers. Accordingly, the error bound is adjusted at each iteration based on the performance of the classifiers.

The weighted error must satisfy

$$(1 - \epsilon_t)\delta_t (C - 1) > \epsilon_t$$

Hence, the new error bound for weak classifier $t$ becomes

$$
\epsilon_t < \frac{1}{1 + \delta_t^{-1}(C - 1)^{-1}}
$$

The new condition relaxes the termination criteria of the training process compared to AdaBoost.M1 and is not as loose as that of SAMME. This methodology targets a smooth accommodation for multi-class imbalanced data sets with any number of classes. Additionally, in multi-class imbalanced learning, it suppresses the bias towards the majority classes by avoiding adjustment of the decision boundaries in directions that favor majority instances.

### 3.4. Multiclass Boosting Method

This section presents the regularized multiclass boosting method referred to as RegBoost. After initializing the weights of all instances with $\frac{1}{N}$ and initializing the regularization parameter with 1, the data set is undersampled using weighted stratified sampling at each training iteration. Stratified sampling divides the instances into groups known as strata followed by applying random sampling to each stratum. At each training iteration, the proposed algorithm applies weighted class-based stratified sampling to each class (stratum)
separately. The weights of the instances represent their probability of selection. The selection process deals separately with each class to ensure that all the classes are balanced and represented in the sampled data. However, RegBoost does not lose valuable information by downsampling the majority classes. Different instances can be selected at different iterations according to their weight distribution. This procedure reduces the training time and tends to select highly weighted instances and those close to the decision boundaries of different classes. In the case of balanced training, the downsampling size of each class is pre-specified and all classes are downsampled equally to the new size.

In the case of imbalanced training, the classes of the data set are downsampled according to the size of the smallest minority class. Let \(|c_i|\) denote the size of class \(c_i\). The smallest minority class is \(|c^*| = \min_{i \in \{1, 2, \ldots, C\}} (c_i)\). For each class in the data set \(R\), stratified sampling selects a subset of instances, denoted with \(r_i\), from each class \(c_i\) such that number of selected instances from a majority class equals the size of the smallest minority class, i.e., \(|r_i| = |c^*|\). The balanced training set, \(S_t\), becomes a collection of sampled majority classes and minority classes:

\[
S_t = \bigcup_i r_i \subset R.
\]

Algorithm 2 presents RegBoost. \(R = \{(x_1, y_1), \ldots, (x_N, y_N)\}\) represents the training set, where the instances are denoted by \(x_i \in \mathcal{X}\) and the class labels are denoted by \(y_i \in \mathcal{Y} = \{1, \ldots, C\}\). \(C\) presents the total number of minority and majority classes, \(N\) presents the total number of instances, and \(T\) presents the total number of iterations. For balanced training, \("r"\) is the number of instances per class after undersampling to form a total sample size of \(S_t = r \times C\). Function \(\mathbb{1}_0[\cdot]\) is an indicator function that returns 1 if true and 0 if otherwise in order to calculate the weighted error; and function \(\mathbb{1}_{-1}[\cdot]\) denotes an indicator function that returns 1 if true and \(-1\) if otherwise in order to increase or decrease the weights of the instances.

With a balanced training set \(S_t\), a classifier \(f_t\) is trained that minimizes the weighted error of \(S_t\). Since data distribution plays a key role in both error minimization (as shown in
Algorithm 2 RegBoost

1: Initialize samples weights $w_0(i) = 1/N$ and $\delta_1 = 1$

2: for $t = 1, \ldots, T$ do

3: Select a subset $S_t \subset R$, according to Equation (17) for Imbalanced sets, and according to $S_t = r \times C$ for balanced sets.

4: Train a weak classifier $f_t$ with $S_t$

\begin{equation}
    f_t = \arg \min_{f_t \in F} \sum_{i=1}^{N} w_{t-1}(i) \mathbb{1}_y[y_i \neq f_t(x_i)]
\end{equation}

5: if $t > 1$ then

6: Calculate $\delta_t$ according to Equation (15).

7: end if

8: if $\epsilon_t > \frac{1}{1 + \delta_t^{-1}(C-1)}$ then

9: return $\alpha_t = 0$

10: else

11: Assign weight $\alpha_t$ for the weak classifier

\begin{equation}
    \alpha_t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) + \frac{1}{2} \log(\delta_t(C - 1))
\end{equation}

12: Modify samples distribution to,

$$w_t \leftarrow w_{t-1}e^{\alpha_t \mathbb{1}_y[y_i \neq f_t(x_i)]}$$

13: Normalize $w_t$

$$w_t \leftarrow \frac{w_t}{\sum_{i=1}^{N} w_t(i)}$$

14: end if

15: end for

16: Integrate weak classifiers $f_t$ into $F(x)$

$$F(x) = \arg \max_y \sum_{t=1}^{T} \alpha_t f_t(x)$$
Eq. (18)) and data sampling, a complete view is needed of the classifier’s empirical performance. Therefore, despite that $f_t$ is trained with $S_t$, which is a subset of $R$, the classifier is evaluated using the entire data set such that the weight of each instance is updated, i.e.,

$$
\epsilon_t = \sum_{i=1}^{N} w_{t-1}(i) \mathbb{1}[y_i \neq f_t(x_i)].
$$

Except in the first iteration, the regularization parameter $\delta$ is computed according to the errors of two consecutive classifiers, which is used in both calculation of error bound and classifier weight.

The modified exponential loss function is adjusted according to the parameter to assign higher weights to misclassified instances and lower weights to correctly classified instances. Once all the weak classifiers are trained, the overall decision is made using weighted majority voting.
CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

This chapter conducts extensive experiments using synthetic and UCI data sets. It also evaluates the proposed method using real-world imagery face recognition and capsule endoscopy data sets. The experiments are conducted using imbalanced and balanced scenarios to demonstrate the efficacy of the method.

4.1. Imbalanced Training

4.1.1. Experimental Setup

The imbalanced experiments used both synthetic and UCI data sets. Two multimodal Gaussian data sets, namely SYN1 and SYN2, are generated by randomly varying the means and variances, which changes the overlaps among classes. Classes in SYN2 are closer to each other on average compared to that of the classes in SYN1. Both synthetic data sets are two dimensional. Four other data sets include, from UCI repository [32], Image Segmentation (Image), Letter Recognition (Letter), Pen-Based Recognition of Handwritten Digits (Pen), and Statlog Landsat Satellite (Statlog).

The experiments also used two binary-class data sets which are the popular synthetic data sets Circle-Disk (CD) and Banana. Circle-Disk is formed of a disk (one class) surrounded by a circle (the other class), and Banana consists of two Banana-shaped classes protruding into each other. Binary cases are included to demonstrate the efficacy of the methods, because it essentially is a special case of the multi-class classification. Additional details of the data sets are summarized in Table 4.1.

Without loss of generality, each data set was divided into a minority group and a majority group. The size of classes in each group was the same except for the Image data set which had an odd number of classes. In the experiments, two-fold cross validation was conducted and classes in both groups were then switched. That is, each class served as the minority class and the majority class.
Table 4.1. Imbalanced data sets and properties.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>No. of classes</th>
<th>Set size</th>
<th>No. of features</th>
<th>Min. size</th>
<th>Maj. size</th>
<th>Test size</th>
<th>Imb. ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN1</td>
<td>50</td>
<td>5000</td>
<td>2</td>
<td>5</td>
<td>50</td>
<td>50</td>
<td>1:10</td>
</tr>
<tr>
<td>SYN2</td>
<td>50</td>
<td>5000</td>
<td>2</td>
<td>5</td>
<td>50</td>
<td>50</td>
<td>1:10</td>
</tr>
<tr>
<td>Image</td>
<td>7</td>
<td>2310</td>
<td>18</td>
<td>15</td>
<td>165</td>
<td>165</td>
<td>1:11</td>
</tr>
<tr>
<td>Letter</td>
<td>26</td>
<td>18200</td>
<td>16</td>
<td>50</td>
<td>350</td>
<td>350</td>
<td>1:7</td>
</tr>
<tr>
<td>Pen</td>
<td>10</td>
<td>10540</td>
<td>16</td>
<td>31</td>
<td>527</td>
<td>527</td>
<td>1:17</td>
</tr>
<tr>
<td>Statlog</td>
<td>6</td>
<td>3744</td>
<td>36</td>
<td>24</td>
<td>312</td>
<td>312</td>
<td>1:13</td>
</tr>
<tr>
<td>CD</td>
<td>2</td>
<td>4000</td>
<td>2</td>
<td>50</td>
<td>1000</td>
<td>1000</td>
<td>1:20</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>4000</td>
<td>2</td>
<td>50</td>
<td>1000</td>
<td>1000</td>
<td>1:20</td>
</tr>
</tbody>
</table>

In constructing ensembles, both decision trees and feed forward back-propagation neural network were used as base classifiers. Early pruning [11] was employed in the decision tree implementation to avoid over fitting. Each ensemble was trained with 100 iterations, i.e., 100 classifiers were created. The results of the proposed method were compared against the state-of-the-art algorithms for multi-class, imbalanced classification including AdaBoost.M1, SAMME, RUSBoost, and SMOTEBoost. To extend RUSBoost for multi-class problems, AdaBoost.M1 strategy was followed. Preliminary experiments of SMOTEBoost using AdaBoost.M1 and AdaBoost.M2 for the multi-class problem showed that results with AdaBoost.M1 had higher sensitivity, accuracy, and efficiency, and hence, AdaBoost.M1 was adopted in the multi-class extension for RUSBoost and SMOTEBoost.

4.1.2. Classification of the Minority Classes

Analysis of classification performance of the minority classes in a multi-class problem is complicated because the impact of imbalance to the discriminant among classes is usually heterogeneous. For instance, if a large margin exists between two classes, the impact of imbalance is much less than the case where two classes are heavily overlapped. In addition,
it is not trivial to characterize the class layout especially for high-dimensional scenarios. Hence, the statistical performance of classification using average sensitivity $\bar{\epsilon}$ and accuracy $\bar{\rho}$ is reported as follows:

\begin{align}
\bar{\epsilon} &= \sum_i \frac{\sum_{k \in c_i} 1[y_k = F(x_k)]}{|c_i|} \\
\bar{\rho} &= \sum_j \frac{\sum_{k \in c_j} 1[y_k = F(x_k)]}{|c_j|}
\end{align}

where $|c_l|$ denotes the size of class $l$, $c_i$ denotes the minority classes, and $c_j$ denotes any class in the data set. Again, $1_{[\cdot]}$ is an indicator function that gives a one if the condition is true and a zero if otherwise.

Table 4.2 summarizes the average sensitivity and Table 4.3 summarizes the average accuracy. The standard deviation is reported in parentheses. The best values achieved are highlighted with bold font. In general, the sampling-based methods consistently exhibit greater performance in both sensitivity and overall accuracy. The average improvement is two- to three-fold with respect to both metrics. However, there are two cases in which RUSBoost was unsuccessful in creating an ensemble. The cause of these failures is rooted in the random undersampling scheme and is also a result of joint factors including problem complexity and base learner. Further discussion will be given in the next section.

RegBoost achieves the best minority class sensitivity in most cases and is the second best in the rest of the cases. Importantly, despite that a superiorly large average sensitivity has been achieved by RegBoost, its overall accuracy remains only merely competitive, not outstanding. Using decision trees as base learner, RegBoost yielded the best sensitivity and accuracy for four cases; using neural networks, it resulted in the best performance in terms of both metrics for two cases. In addition, the standard deviations of RegBoost are small, which demonstrate the consistency of its performance. The underlined results in Table 4.2 and Table 4.3 highlight the best performance among all learner and method combinations. It is evidential that RegBoost presents the superior sensitivity (5 out of 6 cases) and still achieves competitive accuracy (2 out of 6).
Table 4.2. The average sensitivity (in percentage %) of minority classes in multi-class problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sensitivity</td>
<td>Ada</td>
<td>SAMME</td>
<td>RUS</td>
<td>SMT</td>
</tr>
<tr>
<td>SYN1</td>
<td>20.6 (5.0)</td>
<td>20.3 (4.9)</td>
<td>53.94 (11.3)</td>
<td>55.6 (5.8)</td>
<td><strong>58.3 (12.7)</strong></td>
</tr>
<tr>
<td>SYN2</td>
<td>13.2 (4.8)</td>
<td>13.1 (4.8)</td>
<td>14.9 (12.8)</td>
<td>39.7 (2.0)</td>
<td><strong>43.1 (6.1)</strong></td>
</tr>
<tr>
<td>Image</td>
<td>38.1 (7.4)</td>
<td>38.1 (7.0)</td>
<td>78.1 (3.9)</td>
<td>71.3 (3.9)</td>
<td><strong>82.9 (3.7)</strong></td>
</tr>
<tr>
<td>Letter</td>
<td>18.9 (7.5)</td>
<td>19.0 (7.5)</td>
<td>–</td>
<td>35.4 (4.1)</td>
<td><strong>59.5 (5.8)</strong></td>
</tr>
<tr>
<td>Pen</td>
<td>43.4 (8.8)</td>
<td>43.1 (8.3)</td>
<td>81.0 (3.1)</td>
<td>74.7 (4.7)</td>
<td><strong>86.6 (3.1)</strong></td>
</tr>
<tr>
<td>Statlog</td>
<td>45.8 (9.4)</td>
<td>46.0 (9.5)</td>
<td>72.2 (10.9)</td>
<td>64.2 (12.8)</td>
<td><strong>73.5 (14.9)</strong></td>
</tr>
<tr>
<td></td>
<td>Neural networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN1</td>
<td>1.7 (3.4)</td>
<td>0.9 (1.7)</td>
<td>55.0 (9.9)</td>
<td>63.5 (9.9)</td>
<td><strong>63.6 (3.2)</strong></td>
</tr>
<tr>
<td>SYN2</td>
<td>1.04 (1.4)</td>
<td>0.9 (1.1)</td>
<td>–</td>
<td>12.3 (22.0)</td>
<td><strong>45.6 (2.7)</strong></td>
</tr>
<tr>
<td>Image</td>
<td>80.8 (6.9)</td>
<td>80.7 (5.7)</td>
<td><strong>91.9 (2.4)</strong></td>
<td>88.4 (3.6)</td>
<td>88.5 (2.3)</td>
</tr>
<tr>
<td>Letter</td>
<td>2.4 (1.8)</td>
<td>3.4 (3.8)</td>
<td>61.1 (3.0)</td>
<td>63.8 (2.4)</td>
<td><strong>67.7 (2.1)</strong></td>
</tr>
<tr>
<td>Pen</td>
<td>68.2 (11.2)</td>
<td>76.9 (1.4)</td>
<td>93.0 (2.8)</td>
<td>89.0 (6.0)</td>
<td><strong>94.2 (2.0)</strong></td>
</tr>
<tr>
<td>Statlog</td>
<td>61.6 (10.9)</td>
<td>58.8 (9.3)</td>
<td>75.1 (14.7)</td>
<td>72.5 (12.2)</td>
<td><strong>75.3 (14.2)</strong></td>
</tr>
</tbody>
</table>

It is interesting to note that using neural networks as the base learner AdaBoost.M1 and SAMME resulted in extremely low sensitivity but relatively larger standard deviation. This is because there are a small number of cases that yield very low performance but the majority is relatively higher. That is the performance is skewed and the median is greater than the mean.

Although the focus is on the multi-class problem, some experiments were conducted with binary classification problems. The results are reported in Table 4.4 for the average
Table 4.3. The average accuracy (in percentage %) in multi-class problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>SYN1</td>
<td>47.8 (3.4)</td>
</tr>
<tr>
<td>SYN2</td>
<td>37.5 (2.8)</td>
</tr>
<tr>
<td>Image</td>
<td>61.9 (8.4)</td>
</tr>
<tr>
<td>Letter</td>
<td>41.2 (3.5)</td>
</tr>
<tr>
<td>Pen</td>
<td>66.3 (3.6)</td>
</tr>
<tr>
<td>Statlog</td>
<td>63.8 (2.8)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN1</td>
</tr>
<tr>
<td>SYN2</td>
</tr>
<tr>
<td>Image</td>
</tr>
<tr>
<td>Letter</td>
</tr>
<tr>
<td>Pen</td>
</tr>
<tr>
<td>Statlog</td>
</tr>
</tbody>
</table>

Sensitivity and in Table 4.5 for the average accuracy. With binary problems, both RUSBoost and SMOTEBoost achieved much greater performance in contrast to AdaBoost.M1 and SAMME. SMOTEBoost yields the best sensitivity and accuracy in both synthetic data sets. It is appropriate to say that with small to moderate amounts of training data, SMOTEBoost is very competitive. However, in one case, which is Banana with decision trees; the significant improvement in sensitivity achieved by SMOTEBoost falls on the expense of the accuracy of
Table 4.4. The average sensitivity (in percentage %) of minority classes in binary classification problems. The numbers in parentheses are the standard deviation.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
</tr>
<tr>
<td>Sensitivity</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>2.3 (4.7)</td>
</tr>
<tr>
<td>Banana</td>
<td>16.9 (13.1)</td>
</tr>
<tr>
<td>Neural networks</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>53.0 (19.1)</td>
</tr>
<tr>
<td>Banana</td>
<td>51.6 (5.7)</td>
</tr>
</tbody>
</table>

The best accuracy here is achieved by RegBoost; therefore, RegBoost is a competitive and consistent method for dealing with binary classification problems.

Table 4.5. The average accuracy (in percentage %) in binary classification problems. The numbers in parentheses are the standard deviation.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>51.1 (2.2)</td>
</tr>
<tr>
<td>Banana</td>
<td>58.4 (6.5)</td>
</tr>
<tr>
<td>Neural networks</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>75.6 (9.2)</td>
</tr>
<tr>
<td>Banana</td>
<td>75.6 (2.7)</td>
</tr>
</tbody>
</table>
4.1.3. Effective Classifiers in Ensemble

An ensemble relies on the diversity of its classifiers to model the data distribution closely. To deal with multi-class problems, the proposed method, as well as many other state-of-the-art methods, introduces a lower error bound to the learning process. This ensures more training iterations to be conducted but also allows classifiers to have very low weights. When the weight of a classifier is close to zero, it essentially has little contribution to the final decision despite the time for training, as well as the time taken to process a new instance. The number of effective classifiers (NEC) is hence an important property of an ensemble.

Table 4.6 summarizes the average number of effective classifiers in the ensemble out of 100 training iterations. Recall that there is no value reported for the sensitivity or accuracy of some experiment cases. NEC values explain this. For example, cross reference Tables 4.3 and 4.6, and it is easy to see that when NEC is zero, the ensemble yields no results. When NEC is one, that is, when one classifier has a non-zero weight, the ensemble reduces to single classifier and its performance is very low in most cases.

On the other hand, large NEC values could be an indicator of over fitting as well, especially for the undersampling based methods. For instance, when learning from CD data set, RUSBoost resulted in an average of 22.75 effective classifiers when using neural network; however, its sensitivity and accuracy are not proportional to its NEC. The possible cause of this inconsistency is that random undersampling has the great potential of missing key instances when are selected just a small number of samples. Clearly this random process diversifies the classifier, yet it generates a misrepresentation of the underlying model. The proposed method, on the other hand, acts consistently in all cases. The large number of effective classifiers helps RegBoost perform either to the very best or very competitively.

4.1.4. Training Efficiency

The computational cost is a major consideration in the employment of ensemble learning algorithms. The iterative training scheme usually takes longer than does any single classifier. The programs are implemented with MATLAB in a 64-bit Windows system with
Table 4.6. Average number of effective classifiers for imbalanced sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
<td>SAMME</td>
</tr>
<tr>
<td>SYN1</td>
<td>1.75</td>
<td>1</td>
</tr>
<tr>
<td>SYN2</td>
<td>2.75</td>
<td>1.25</td>
</tr>
<tr>
<td>Image</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Letter</td>
<td>1.75</td>
<td>2.25</td>
</tr>
<tr>
<td>Pen</td>
<td>2.25</td>
<td>2</td>
</tr>
<tr>
<td>Statlog</td>
<td>1.75</td>
<td>1</td>
</tr>
<tr>
<td>CD</td>
<td>1.25</td>
<td>1.5</td>
</tr>
<tr>
<td>Banana</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Intel Core 2 Duo processor at 3GHz. The system has 4GB memory and 6.6GB virtual memory.

Table 4.7 summarizes the average time used for training ensemble classifiers. The second column lists the data volume in the training phase, which is the product of the number of instances used for training and the number of features per instance. The data sets are put in descending order according to the training data volume. The shortest time is highlighted in bold font.

Due to the fact that RUSBoost, SMOTEBest, and the proposed method require a data resampling step, intuitively they take a longer time to complete training. This is true when the problem is relatively simple; both Adaboost.M1 and SAMME took much less time to process the synthetic data sets using decision trees as the base classifier. However, as the problem complexity grows, especially as the training data volume increases, the advantage of the undersampling based methods reveals that RUSBoost and RegBoost took the least amount of time to process all four UCI data sets. RUSBoost failed to achieve an effective ensemble for Letter data set. Hence the time used to generate RUSBoost ensemble is omitted.
SMOTEBost, on the other hand, suffers from processing the additional synthetic instances; the extended time cost is significant with high training data volume.

Table 4.7. Time (in second) used in training ensemble classifiers for imbalanced sets.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Training Volume</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ada SAMME RUS SMT Reg</td>
<td>Ada SAMME RUS SMT Reg</td>
</tr>
<tr>
<td>Banana</td>
<td>2100</td>
<td>1.42 <strong>1.41</strong> 1.7 2.6 2.5</td>
<td>859 854 510 910 <strong>426</strong></td>
</tr>
<tr>
<td>CD</td>
<td>2100</td>
<td><strong>1.21</strong> 1.22 1.8 2.5 2.7</td>
<td>1029 1030 <strong>424</strong> 1072 475</td>
</tr>
<tr>
<td>SYN2</td>
<td>2750</td>
<td>50.6 50.6 <strong>41.6</strong> 88.7 47.4</td>
<td>3027 3159 – 4856 <strong>1511</strong></td>
</tr>
<tr>
<td>SYN1</td>
<td>2750</td>
<td>38.4 <strong>38.3</strong> 38.7 50.2 41.8</td>
<td>3558 3595 1809 5136 <strong>1679</strong></td>
</tr>
<tr>
<td>Image</td>
<td>11340</td>
<td>6.7 6.5 <strong>5.3</strong> 13.7 6.3</td>
<td>1400 1032 <strong>708</strong> 1494 785</td>
</tr>
<tr>
<td>Statlog</td>
<td>36288</td>
<td>12.3 12.3 <strong>5.1</strong> 28.9 7.2</td>
<td>1171 1196 <strong>619</strong> 1514 746</td>
</tr>
<tr>
<td>Pen</td>
<td>44640</td>
<td>31.8 30.3 <strong>12.4</strong> 76.1 16.4</td>
<td>3142 2318 1283 4033 <strong>993</strong></td>
</tr>
<tr>
<td>Letter</td>
<td>83200</td>
<td>213 221 – 409 <strong>108</strong></td>
<td>6587 6701 2108 11373 <strong>1718</strong></td>
</tr>
</tbody>
</table>

Using neural network as the base classifier requires longer training time to train an ensemble. It is evident that the proposed method is very predictable in its superior efficiency among all methods. The time used by RUSBoost and RegBoost is close, although RegBoost yielded the shortest average time for 5 cases among the 8 data sets. RUSBoost again failed to generate effective ensemble for SYN2 data set. Compared to AdaBoost.M1 and SAMME methods, the proposed method reduces the training time by more than 50%.

4.2. Real-World Data Results

4.2.1. Experimental Setup

These experiments used real-world imagery data sets. Face recognition data sets are an example of sets with a large number of subjects, i.e., classes. As the number of subjects increases, the multi-class learning process becomes more difficult. Capsule Endoscopy (CE) data set is an example of data sets with a massive number of images. While thousands of
these images stand normal, a significantly low number of images present the diseased cells. This creates a large imbalance ratio between the normal and abnormal images. These types of sets have a high-dimensionality.

Face recognition data sets include two popular databases (AT&T and AR [67]). Images in the AR data set were converted into gray scale and manually cropped and aligned. The cropped image size is $120 \times 110$. The images of all data sets were preprocessed using principal component analysis for dimensionality reduction before the training. Test images were projected on the Eigen-space before testing. The CE data set was collected from eight different patients. Additional details of the data sets are summarized in Table 4.8.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>No. of classes</th>
<th>No. of images</th>
<th>Min. size</th>
<th>Maj. size</th>
<th>Test size</th>
<th>Imb. ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>40</td>
<td>400</td>
<td>10304</td>
<td>2</td>
<td>9</td>
<td>1:4.5</td>
</tr>
<tr>
<td>AR</td>
<td>50</td>
<td>550</td>
<td>13200</td>
<td>2</td>
<td>10</td>
<td>1:5</td>
</tr>
<tr>
<td>CE</td>
<td>2</td>
<td>4200</td>
<td>36864</td>
<td>25</td>
<td>2075</td>
<td>25-2075 1:83</td>
</tr>
</tbody>
</table>

Each of the face recognition data sets was divided into a minority group and a majority group. The size of classes in each group is the same. Limited by the number of instances in each class in the AT&T and AR data sets, leave-one-out cross validation was used. Classes in both groups were switched. That is again, each class served as the minority class and the majority class. The two classes of the CE data were not switched owing to the data set being naturally imbalanced and the minority class (diseased images) having very few instances compared to the majority class (normal images).

These real-world data experiments trained both decision trees and feed forward back-propagation neural network as base classifiers with the same previous settings. Each ensemble was trained with 100 iterations and the results were compared against the state-of-the-art boosting algorithms.
4.2.2. Classification of the Minority Classes

Table 4.9 summarizes the average sensitivity and Table 4.10 summarizes the average accuracy. The standard deviation is reported in parentheses. The best values achieved are highlighted with bold font. As with synthetic and UCI data sets, RegBoost consistently exhibits greater performance in both sensitivity and overall accuracy compared to AdaBoost.M1 and SAMME. The improvement is significant with face recognition using neural networks. RegBoost achieved the highest sensitivity while SMOTEBoost achieved the highest accuracy for the AT&T and AR data sets. RUSBoost was unsuccessful in creating an ensemble for any of the face recognition data sets. The cause of these failures is rooted in the random undersampling scheme especially with the low number of images per class for these sets.

Table 4.9. The average sensitivity (in percentage %) of minority classes in real-world problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
</tr>
<tr>
<td>Sensitivity</td>
<td></td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>18.7 (8.1)</td>
</tr>
<tr>
<td>AR</td>
<td>20.0 (9.6)</td>
</tr>
<tr>
<td>CE</td>
<td>8.0 (11.3)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Neural networks</td>
<td></td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>1.2 (2.9)</td>
</tr>
<tr>
<td>AR</td>
<td>1.0 (2.0)</td>
</tr>
<tr>
<td>CE</td>
<td>16.0 (16.9)</td>
</tr>
</tbody>
</table>

Using neural networks as the base learner AdaBoost.M1 and SAMME resulted in extremely low sensitivity but relatively larger standard deviation due to the low performance
Table 4.10. The average accuracy (in percentage %) in real-world problems. The numbers in parentheses are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training.

| Data sets | Decision trees | Data | Accuracy |          |          |          |
|-----------|----------------|------|----------|----------|----------|
|           |                | Ada  | 35.6 (8.2) | 35.5 (8.4) | -        | 40 (8.6) | 35.8 (9.0) |
|           |                | SAMME|          |          |          |          |
|           |                | RUS  |          |          |          |          |
|           |                | SMT  |          |          |          |          |
|           |                | Reg  |          |          |          |          |
| AT&T      |                |      | 40 (8.6)  |          |          |          |
| AR        |                |      | 28.5 (9.5) | 28.5 (9.5) | -        | 35.5 (10.6) | 29.0 (8.9) |
| CE        |                |      | 53.6 (5.7) | 53.6 (5.9) | 68.9 (3.8) | -        | 72.7 (7.5) |

| Data sets | Neural networks | Data | Accuracy |          |          |          |
|-----------|----------------|------|----------|----------|----------|
|           |                | Ada  | 11.8 (15.2) | 40.2 (6.2) | -        | 56.2 (14.9) | 48.7 (11.0) |
|           |                | SAMME|          |          |          |          |
|           |                | RUS  |          |          |          |          |
|           |                | SMT  |          |          |          |          |
|           |                | Reg  |          |          |          |          |
| AT&T      |                |      | 56.2 (14.9) |          |          |          |
| AR        |                |      | 8.0 (5.1)  | 19.0 (6.6) | -        | 2.0 (5.1)  | 22.0 (6.7) |
| CE        |                |      | 57.8 (8.4) | 57.9 (8.2) | 73.5 (2.2) | -        | 73.8 (1.2) |

of some folds and the increased performance of others, i.e., the median is greater than the mean. With large data and dimensionality such as in CE data set, SMOTEBoost failed training due to the large number of synthetic instances created. In handling large data sets, RegBoost exhibited both robustness and the best performance. Although RUSBoost returned the highest sensitivity using neural network, RegBoost’s result was extremely close and with a smaller standard deviation. It is plausible to say that RegBoost has the great advantage in handling large data sets.

4.2.3. Effective Classifiers in Ensemble

To measure the diversity of the ensemble, the number of effective classifiers (NEC) is reported as an important property. Clearly, RegBoost has the highest NEC of all methods, which is consistent with its improved performance for the face recognition data sets. RUSBoost possesses the highest NEC for CE set. However, RUSBoost did not achieve the
highest sensitivity and accuracy for this data set in all cases. RegBoost exhibited robustness and consistency in the relationship between NEC and its improved performance.

Table 4.11. Average number of effective classifiers for real-world sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada SAMME RUS SMT Reg</td>
<td>Ada SAMME RUS SMT Reg</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>1.4 1.4 0 1.35 54.65</td>
<td>1.6 37.8 0 3.8 79.85</td>
</tr>
<tr>
<td>AR</td>
<td>1.68 1.72 0 1.9 66</td>
<td>1 59.5 0 1 73.8</td>
</tr>
<tr>
<td>CE</td>
<td>10.5 10 45.5 – 14</td>
<td>2.5 2 13.5 – 7.5</td>
</tr>
</tbody>
</table>

4.2.4. Training Efficiency

Table 4.12. Time (in second) used in training ensemble classifiers for real-world sets.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Training Volume</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada SAMME RUS SMT Reg</td>
<td>Ada SAMME RUS SMT Reg</td>
<td></td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>2266880</td>
<td>27.2 27.1 – 551</td>
<td>26.1 917 936 – 1597 727</td>
</tr>
<tr>
<td>AR</td>
<td>3960000</td>
<td>62.2 62.8 – 1059</td>
<td>40.6 1198 1144 – 2421 832</td>
</tr>
<tr>
<td>CE</td>
<td>77414400</td>
<td>1012 1012 29.2 –</td>
<td>27.6 10892 10902 314 – 293</td>
</tr>
</tbody>
</table>

Table 4.12 summarizes the average time needed for training ensemble classifiers for the real-world sets. Clearly, the problem complexity is larger with these real-world data sets due to high-dimensionality and the large number of images. In all cases, RegBoost achieved superior efficiency. RUSBoost failed to achieve an effective ensemble for AT&T and AR data sets. SMOTEBest, on the other hand, suffered from processing the additional synthetic instances; the extended time cost would be significant with high training data volume. In particular, when training with CE data, SMOTEBest was unsuccessful at completing due to the extremely large amount of data generated for training. As shown in Table 4.8, the imbalance ratio of CE is 1:83. Therefore, the SMOTEBest method has to handle almost 80
times more data than any other method, which causes it to fail, given the limited amount of memory in the computer system.

(a) using decision tree as base classifier

(b) using neural network as base classifier

Figure 4.1. Average relative time taken in training for synthetic, UCI, and real-world data sets. The discontinued or incomplete curves indicate failure of generating an ensemble.

To gain an idea of the improvement in efficiency among all the evaluated data sets, Fig. 4.1 illustrates scatter plots of the relative average time used in training with respect to the training data volume for all synthetic, UCI, and real-world data sets. The plots show the ratio of time used by each method with respect to the minimum time in each case. To reveal the difference, the vertical axis is in a logarithmic scale. Although the time is probably
affected by the complexity of the problem and hence curves are not increasing functions, the relative computational cost among methods is clearly depicted where RegBoost exhibits the best efficiency among all cases especially when dealing with large amounts of data as shown in the plot. Despite the absolute amount of time used with different base classifiers, this trend remains consistent.

4.3. Regularization Parameter Behavior

In the proposed method, the regularization parameter $\delta$ plays a vital role in the algorithm’s capability and performance. This section studies its properties as well as its behavior, with respect to the performance to gain a deep insight into this strategy. The performance of the parameter is evaluated on all synthetic, UCI, and real-world data sets.

In general, an increase of $\delta$ between two consecutive training iterations indicates that the number of second-round-misclassified instances decreases. However, the amplitude of the change of $\delta$ is also determined by the instances weights. That is, an “easy” instance that has been correctly classified by several previous consecutive classifiers and is labeled incorrectly makes a small increment to the $\delta$ compared to those borderline instances that are repeatedly classified correctly and incorrectly. The weight associated to such an easy instance is clearly less than the weight of a borderline instance, which in turn limits that easy instance’s contribution to the change of $\delta$. In addition, the employment of different training instances injects variations into the regularization parameter. It is hence anticipated that $\delta$ fluctuates during the training process, which is evident from Fig. 4.2.

Fig. 4.2 illustrates the changes in the regularization parameter with respect to the training iteration. The horizontal axis yields the iteration number (up to 100) and the vertical axis shows the $\delta$ values in the range of $(0, 1]$. The solid line and dash line depict $\delta$ with decision tree and neural network as base learner, respectively. Despite some disparities in fluctuation between two base learners, the overall behaviors mostly coincide.

Recall that $\delta$ is initialized with one in the first training iteration. Hence, its value tends to drop sharply at the very beginning because there are surely many instances that are
Figure 4.2. Regularization parameter $\delta$ changes along the training iterations. $\delta$ is initialized to one. The solid line depicts the $\delta$ of using decision tree as base classifier and the dash line depicts the $\delta$ of using neural network as base classifier.
misclassified. The weight of these classifiers, i.e., $\alpha$, then relies mostly on the performance-based evaluation term $\log(1 - \epsilon_t)/\epsilon_t$ as shown in Eq. (8). As training continues, $\delta$ tends to increase, which allows further contribution from the multi-class regularization term $\log(C - 1)$ to the classifier weight and shows that the method was successful in reducing the number of second-round-misclassified instances.

Since the proposed method was successfully applied to binary class problems, it is interesting to see how $\delta$ modifies the learning behavior. Fig. 4.3 illustrates the changes of the regularization parameter with respect to the training iteration in three binary class problems. In contrast to the multi-class cases shown in Fig. 4.2, the regularization parameter depicts no consistent increasing trend but becomes rather steady in a range. This is mostly because $\delta$ exclusively dominates the multi-class regularization term. As $\delta \leq 1$, the classifier weight $\alpha$ is slightly suppressed, which makes RegBoost perform closely to RUSBoost in binary classification problems, as shown in Table 4.4.

**Figure 4.3.** Regularization parameter $\delta$ in binary class problems. The solid line depicts the $\delta$ of using decision tree as base classifier and the dash line depicts the $\delta$ of using neural network as base classifier.
To study the effect of the regularization term on performance improvement, stratified sampling was incorporated into AdaBoost.M1 and SAMME. Hence, the difference between these methods and RegBoost mostly results from the regularization term. Table 4.13 and Table 4.14 list the average performance of the three methods with two base learners. The column “Imp” gives the improvement of performance in percentage, computed by comparing the best results from AdaBoost.M1 and SAMME (with stratified sampling).

Table 4.13. Average sensitivity of AdaBoost.M1 and SAMME with stratified sampling and RegBoost. Column “Imp” gives the percentage of improvement using RegBoost. Parentheses indicates negative number.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada SAM Reg Imp</td>
<td>Ada SAM Reg Imp</td>
</tr>
<tr>
<td>Sensitivity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN1 57.1 60.1 58.3 (3)</td>
<td>53.2 63.1 63.6 0.8</td>
<td></td>
</tr>
<tr>
<td>SYN2 18.2 42.1 43.1 2.4</td>
<td>– 45.1 45.6 1.1</td>
<td></td>
</tr>
<tr>
<td>Image 79.8 82.0 82.9 1.1</td>
<td>88.1 89.0 88.5 (0.6)</td>
<td></td>
</tr>
<tr>
<td>Letter – 57.8 59.5 2.9</td>
<td>60.3 67.0 67.7 1</td>
<td></td>
</tr>
<tr>
<td>Pen 84.0 86.5 86.6 0.1</td>
<td>93.2 94.0 94.2 0.2</td>
<td></td>
</tr>
<tr>
<td>Statlog 71.5 71.6 73.5 2.7</td>
<td>67.8 64.1 75.3 11.1</td>
<td></td>
</tr>
<tr>
<td>AT&amp;T – 36.2 40.0 10.5</td>
<td>– 58.7 60.0 2.2</td>
<td></td>
</tr>
<tr>
<td>AR – 45.0 46.0 2.2</td>
<td>– 24.0 29.0 20.8</td>
<td></td>
</tr>
<tr>
<td>CD 46.7 46.6 54.5 16.7</td>
<td>67.3 67.8 66.3 (2.2)</td>
<td></td>
</tr>
<tr>
<td>Banana 51.1 51.6 53.9 4.5</td>
<td>69.2 69.4 69.6 0.3</td>
<td></td>
</tr>
<tr>
<td>CE 40.0 39.3 50.0 25</td>
<td>80.0 80.0 82.0 2.5</td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 4.13, the average improvements of RegBoost in sensitivity with decision tree and neural network as base learners are 5.92% and 3.38%, respectively. More importantly, such improvements are not a result of sacrificing the performance of the majority classes. This is evidential from the improvement in accuracy, as shown in Table 4.14. While
the sensitivity of the minority classes improves, the overall accuracy also increases. The average improvements of RegBoost in accuracy with decision tree and neural network as base learners are 3.94% and 1.91%, respectively. The underlined results are the best performance among all ensemble and learner combinations. Out of 11 test cases, RegBoost results in 9 best sensitivities and 9 best accuracies. Even in the very few cases where RegBoost gives a lower performance, the difference is mostly less than one percent. Clearly, the regularization parameter brings a positive impact to the performance of multi-class ensemble.


<table>
<thead>
<tr>
<th>Data sets</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada</td>
<td>SAM</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN1</td>
<td>61.5</td>
<td>57.1</td>
</tr>
<tr>
<td>SYN2</td>
<td>20.2</td>
<td>34.4</td>
</tr>
<tr>
<td>Image</td>
<td>82.3</td>
<td>83.2</td>
</tr>
<tr>
<td>Letter</td>
<td>–</td>
<td>54.3</td>
</tr>
<tr>
<td>Pen</td>
<td>87.3</td>
<td>87.9</td>
</tr>
<tr>
<td>Statlog</td>
<td>77.1</td>
<td>74.8</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>–</td>
<td>28.7</td>
</tr>
<tr>
<td>AR</td>
<td>–</td>
<td>27.5</td>
</tr>
<tr>
<td>CD</td>
<td>68.2</td>
<td>68.1</td>
</tr>
<tr>
<td>Banana</td>
<td>74.2</td>
<td>74.7</td>
</tr>
<tr>
<td>CE</td>
<td>69.6</td>
<td>69.8</td>
</tr>
</tbody>
</table>

Cross-referencing with Table 4.2 and Table 4.3 and the real-world data sets results in Table 4.9 and Table 4.10, it can be seen that stratified sampling also improves the performance of AdaBoost.M1 and SAMME and in some cases the improvement is great. Note
that AdaBoost.M1 produces no results in some cases. Clearly, integrating stratified sampling adds no help to AdaBoost.M1. However, the innate constraint is the stringent error bound that halts the learning process from creating effective classifiers.

![Decision boundaries of three consecutive classifiers using (a) stratified sampling with AdaBoost.M1 and (b) RegBoost. Solid line draws the decision boundary for the first classifier; dash line draws the decision boundary for the second classifier; and dotted line draws the decision boundary for the third classifier.](image)

**Figure 4.4.** Decision boundaries of three consecutive classifiers using (a) stratified sampling with AdaBoost.M1 and (b) RegBoost. Solid line draws the decision boundary for the first classifier; dash line draws the decision boundary for the second classifier; and dotted line draws the decision boundary for the third classifier.
Fig. 4.4 depicts the decision boundaries of three intermediate classifiers using CD data set \(^1\) and decision tree as base learner. Fig. 4.4(a) shows the results of AdaBoost.M1 with stratified sampling and Fig. 4.4(b) shows the results of RegBoost. The minority and majority instances are visualized with circles and asterisks, respectively. Using a subset of instances for training, AdaBoost.M1 started with a good model that missed only one minority instance. However, the dominating majority instances quickly migrated the following classifiers in order to minimize the count of misclassifications by sacrificing the much lower number of minority instances.

In contrast, RegBoost demonstrated great robustness using the regularization parameter. As classifiers were developed, each of them attended part of the data distribution and the aggregation of them clearly shows the non-linear margin between classes. The bias of the overwhelmingly large number of majority instances was greatly suppressed.

4.4. Balanced Training

4.4.1. Experimental setup

The experiments used the same data sets trained in the imbalanced experiments in addition to the Yale [40] face database. The Yale set was not chosen for the imbalanced experiments due to its complexity which results in a performance close to that of random guessing. For the balanced experiments, the CE data set included three classes which segmented the images into stomach, small intestine, and large intestine. The modified setup for the experiments is shown in Table 4.15. Two-fold cross validation was conducted for all data sets except for AT&T and AR sets where leave-one-out cross validation was conducted due to the low number of images per class. For imagery sets, only two sampling sizes in addition to a varying sampling size were used due to the lower number of instances per class for some classes and the large training volume for others. Data sets were trained using AdaBoost.M1, SAMME, and RegBoost for 100 iterations for all experiments. Decision trees with early pruning and feed forward back-propagation neural network were used as the base learners.\(^1\)

\(^1\)The choice of CD data set is simply to minimize the complexity of visualization and focus on the propagation of the learning process.
Different sampling sizes were used for RegBoost including a variable scheme that increases the sampling size every 25 iterations as shown in Table 4.15.

**Table 4.15.** Balanced data sets and properties.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>No. of classes</th>
<th>No. of instances per class</th>
<th>No. of features</th>
<th>Size per class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN1</td>
<td>50</td>
<td>100</td>
<td>2</td>
<td>10, 25, 40, 10, 20, 30, 40</td>
</tr>
<tr>
<td>SYN2</td>
<td>50</td>
<td>100</td>
<td>2</td>
<td>10, 25, 40, 10, 20, 30, 40</td>
</tr>
<tr>
<td>Image</td>
<td>7</td>
<td>660</td>
<td>18</td>
<td>50, 100, 150, 30, 60, 90, 120</td>
</tr>
<tr>
<td>Letter</td>
<td>26</td>
<td>1400</td>
<td>16</td>
<td>100, 200, 300, 75,150,225,300</td>
</tr>
<tr>
<td>Pen</td>
<td>10</td>
<td>1054</td>
<td>16</td>
<td>150, 300, 450, 150,250,350,450</td>
</tr>
<tr>
<td>Statlog</td>
<td>6</td>
<td>626</td>
<td>36</td>
<td>100, 200, 300, 75,150,225,300</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>40</td>
<td>10</td>
<td>10304</td>
<td>5, 8, 2, 4, 6, 8</td>
</tr>
<tr>
<td>AR</td>
<td>50</td>
<td>11</td>
<td>13200</td>
<td>5, 8, 2, 4, 6, 8</td>
</tr>
<tr>
<td>Yale</td>
<td>38</td>
<td>64</td>
<td>896</td>
<td>20, 28, 10, 16, 22, 28</td>
</tr>
<tr>
<td>CD</td>
<td>2</td>
<td>2000</td>
<td>2</td>
<td>250, 500, 750, 200,400,600,800</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>2000</td>
<td>2</td>
<td>250, 500, 750, 200,400,600,800</td>
</tr>
<tr>
<td>CE</td>
<td>3</td>
<td>1400</td>
<td>36864</td>
<td>50, 150, 50,125,200,275</td>
</tr>
</tbody>
</table>

4.4.2. Classification Error Rate

In dealing with balanced data sets, the experiments report the statistical performance of classification using the average error rate (AER). Figure 4.5 and Figure 4.6 show the AER of the twelve data sets using decision trees and neural network, respectively. The standard deviation is shown on the top of the AER bars. In general, RegBoost with different sampling sizes consistently exhibits greater performance compared to AdaBoost.M1 and SAMME. Among all learner and method combinations, RegBoost presents a superior performance with the lowest error rate for the multi-class data sets (9 out of 10). As for the boosting base classifiers, decision trees showed a significant improvement, whereas neural network
Figure 4.5. Average error rate of training decision trees for all data sets using AdaBoost.M1 (AB), SAMME (SM), and RegBoost (RB1, RB2, RB3, and RBv). The error rate standard deviation is on the top of the bars.

turned in results merely equivalent to other methods, with indeed an improvement, but not a significant one.

When the decision trees classifiers were trained in Figure 4.5, the standard deviations of RegBoost were small, which demonstrate the consistency of its performance. For instance,
Figure 4.6. Average error rate of training neural network for all data sets using AdaBoost.M1 (AB), SAMME (SM), and RegBoost (RB1, RB2, RB3, and RBv). The error rate standard deviation is on the top of the bars.

the AER of RegBoost is reduced by 77% compared to other methods as seen with the Pen data set and by 57% as seen with AT&T data set. Additionally, RegBoost achieves the best performance for both binary sets.
When neural network classifiers were trained in Figure 4.6, the performance of all methods were close, with a tiny improvement using RegBoost. The performance of AdaBoost.M1 and SAMME is improved using neural network due to lower stability compared to decision trees which results in more diversity among the ensemble and in reduction of the early termination problem. In some cases, training the lowest sampling size for RegBoost results in increased AER compared to other sampling sizes such as SYN1, SYN2, and Letter sets. The lowest and the variable sampling sizes have a deteriorated performance with the face recognition sets due to the low sample size per class for these specific sets. On the contrary, the lowest and variable sampling sizes achieve better performance with the two binary data sets CD and Banana. The medium and highest sampling sizes have close and consistent performance for all multi-class sets.

4.4.3. Effective Classifiers in Ensemble

To measure the diversity of the classifiers in order to insure that the data distribution is modeled closely, the number of effective weak classifiers (NEC), i.e., classifiers with non-zero alphas, was measured. Table 4.16 summarizes the number of effective classifiers in the ensemble out of 100 training iterations using decision trees and neural networks for the balanced learning experiments.

The table indicates that RegBoost has the highest NEC using different learners for the majority of the multi-class data sets. It also has the highest NEC for binary data sets using decision trees. Using decision trees, all RegBoost sampling sizes consistently achieve higher NEC than AdaBoost.M1 and SAMME. Using neural networks, NEC of AdaBoost.M1 and SAMME improved significantly. Additionally, SAMME achieved larger NEC than AdaBoost.M1 due to the strict error bound of the latter. It can be concluded that NEC is directly related to the AER achieved by these methods. The increase in NEC for RegBoost is reflected in its improved performance measured by AER. The improvement margin achieved by RegBoost compared to AdaBoost.M1 and SAMME is reduced using neural networks. This can be attributed to the effect of the early termination problem and repetition.
of misclassified instances which, when using neural networks, is reduced for AdaBoost.M1 and SAMME.

4.4.4. Training Efficiency

With the iterative training scheme, the computational cost is a major consideration in the employment of ensemble learning. In particular, the balanced experiments had larger training sizes compared to the imbalanced ones. Table 4.17 summarizes the average time used for training ensemble classifiers for the balanced experiments. The shortest training time is highlighted in bold font. RegBoost took a slightly longer time in training for relatively simple problems. In particular, using decision trees for the binary sets and SYN1, RegBoost took longer due to the data resampling step.

Table 4.16. Average number of effective classifiers for balanced sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada  SAMME RB1 RB2 RB3 RBv</td>
<td>Ada  SAMME RB1 RB2 RB3 RBv</td>
</tr>
<tr>
<td>SY1</td>
<td>2.0 1.5</td>
<td><strong>84.5</strong> 54.5 39.5 54</td>
</tr>
<tr>
<td>SYN2</td>
<td>2.0 2.0</td>
<td><strong>77.0</strong> 51.5 61 63.5</td>
</tr>
<tr>
<td>Image</td>
<td>2.0 1.0</td>
<td>61.5 44.5 58</td>
</tr>
<tr>
<td>Letter</td>
<td>2.5 4.0</td>
<td><strong>78.0</strong> 59 50.5</td>
</tr>
<tr>
<td>Pen</td>
<td>1.5 1.5</td>
<td>38</td>
</tr>
<tr>
<td>Statlog</td>
<td>2.5 2.5</td>
<td><strong>85.0</strong> 66.5 48</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>1.3 2.3</td>
<td>59.7 56.4</td>
</tr>
<tr>
<td>AR</td>
<td>1.2 1.8</td>
<td>46.6 47.2</td>
</tr>
<tr>
<td>Yale</td>
<td>3.0 2.0</td>
<td>72.5 44.5</td>
</tr>
<tr>
<td>CD</td>
<td>1.5 2.0</td>
<td>5.5 6.5</td>
</tr>
<tr>
<td>Banana</td>
<td>1.5 2.0</td>
<td>13 6 8</td>
</tr>
<tr>
<td>CE</td>
<td>2.5 1.5</td>
<td><strong>95.5</strong> 55</td>
</tr>
</tbody>
</table>

53
Table 4.17. Time (in second) used in training ensemble classifiers for balanced sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Decision trees</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ada  SAMME RB1 RB2 RB3 RBv</td>
<td>Ada  SAMME RB1 RB2 RB3 RBv</td>
</tr>
<tr>
<td>Banana</td>
<td>1.9  1.9</td>
<td>6.7  11.7</td>
</tr>
<tr>
<td>CD</td>
<td>1.6  1.6</td>
<td>6.2  11.0</td>
</tr>
<tr>
<td>SYN2</td>
<td>105  105</td>
<td>61.8  97.8</td>
</tr>
<tr>
<td>SYN1</td>
<td>7.2  7.0</td>
<td>52.7  73.1</td>
</tr>
<tr>
<td>Image</td>
<td>13.1 13.2</td>
<td>11.1  16.9</td>
</tr>
<tr>
<td>Statlog</td>
<td>25.1 25.4</td>
<td>19.4  31.3</td>
</tr>
<tr>
<td>Pen</td>
<td>62.4 62.2</td>
<td>51.6  80.9</td>
</tr>
<tr>
<td>Letter</td>
<td>378  378</td>
<td>219  327</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>59.5 61.9</td>
<td>51.3  70.1</td>
</tr>
<tr>
<td>AR</td>
<td>150  151</td>
<td>101  140</td>
</tr>
<tr>
<td>Yale</td>
<td>684  713</td>
<td>451  698</td>
</tr>
<tr>
<td>CE</td>
<td>2360 2493</td>
<td>37.8  144</td>
</tr>
</tbody>
</table>

However, when the data complexity increases and when neural network is used as the base learner, RegBoost consistently takes the least amount of time to process other data sets. The least and the variable sampling sizes interchangeably achieve the highest efficiency for the majority of the data sets. The improvement is significant with the imagery high-dimensional data sets. The reduction in the training time using RegBoost is up to 98% with CE data set trained by decision trees. The proposed method, therefore, is predictable in its superior efficiency among all methods for the balanced data sets.
A massive amount of data is generated daily in different applications. Commonly this data has an uneven distribution of examples among multiple classes. This complicates the learning process especially for the minority classes. Existing methods handle the imbalance problem mostly by using binarization techniques that require an elongated training time. Ensemble learning and, in particular, boosting methods were developed to target an improved accuracy for binary data sets; however, the extension from binary to multi-class classification is not straightforward. This extension is divided into indirect and direct conversions. In the indirect conversion, the multi-class classification problem that converts into multiple binary problems. The direct conversion adjusts the exponential loss function to accommodate multi-class data sets. However, boosting methods suffer several problems. First, the methods exhibit a deteriorated performance in learning from imbalanced sets and, in particular, from multi-class data sets. Second, the boosting algorithms assign a strict weighted error bound for the base learners in some cases such as AdaBoost.M1 and assign a loose bound in other cases such as SAMME. Third, the performance suffers from the early termination problem that is caused by repetition of misclassified instances which limits the types of base classifiers to be trained based on their stability. Fourth, the methods additionally require an elongated training time that linearly increases with the number of weak classifiers employed.

This dissertation proposes a multi-class boosting-based method, namely RegBoost, to handle the imbalanced, multi-class learning problems. This method uses stratified under-sampling to recover the balance among classes, and addresses the unpredictability of base learners with regularization. The sampling procedure randomly selects instances for the majority classes based on their data distribution, and the regularization modifies the loss function to penalize the classifiers with second-round-misclassified instances. This regularization parameter also adjusts the error bound in accordance to a classifier’s performance. Moreover, the parameter combined with stratified sampling improves the training efficiency,
avoids early termination, and results in a bound that is not too strict or too loose to smoothly accommodate multi-class data sets. Additionally, the proposed method allows for integrating any type of weak classifier having different degrees of stability.

This research investigated imbalanced and then balanced learning. For imbalanced learning, experiments are conducted using 11 diverse synthetic, UCI, and real-world data sets with moderate to high imbalance ratios. The experiments aim to evaluate the capability and stability of the proposed method. The results demonstrate superior performance of the proposed method compared to several state-of-the-art algorithms for imbalanced, multi-class classification problems. RegBoost achieved the highest sensitivity for the minority classes in most cases (the best sensitivity in 7 out of 8 multi-class cases). More importantly, sensitivity improvement of the minority classes is accompanied with the improvement of the overall accuracy for all classes. The standard deviation of the results illustrates the consistency of the proposed method. Note that although the proposed method uses an undersampling strategy, the ensemble stability is retained. Despite the failure of training by other methods, RegBoost successfully devises ensembles in all cases. In addition to multi-class problems, the experiments with binary-class problems also reveal the applicability of RegBoost and that its performance is highly competitive.

For balanced learning, experiments were conducted using 12 diverse synthetic, UCI, and real-world data sets. RegBoost demonstrated superior performance compared to other multi-class boosting methods. Evidently RegBoost achieved the lowest error rate in most cases (9 out of 10 multi-class sets and 2 out of 2 binary sets). The standard deviation further illustrates the consistency of RegBoost for balanced data sets.

Another unique fact arising from the experiments is that looking at the number of effective classifiers resulted from the training of the ensemble. The experiments show that RegBoost yielded the largest number of effective classifiers in most cases, which indicates the diversity of the base classifiers in the ensemble. This is true for both the imbalanced and balanced learning processes. RegBoost was consistent in achieving the largest number of effective weak classifiers, which was directly reflected in its higher performance. In
undersampling-based methods, the number of effective classifiers is also an indicator of overfitting, which could be caused by random undersampling missing important examples when a fairly small number of examples are needed to balance the training data set. Moreover, the large number of effective weak classifiers indicates the capability of the proposed method to employ any type of base learners and to avoid the early termination problem related to repetition of misclassified instances. This problem was prevalent in learning from the balanced sets using decision trees.

A major concern of applying the boosting method to a large data set is the computational cost. With the step of forming the training set, RegBoost, as well as RUSBoost and SMOTEBoost, requires extra time. However, the time used to select a subset of examples is outweighed by the training time. It turns out that the sampling process allows RegBoost to achieve the highest efficiency. The reduction in computational cost is significant and in some cases reaches more than 50%. As the volume of training data increases, efficiency enhancement with the proposed method gains in significance.

The regularization parameter played a vital role in the algorithm’s capability of handling multi-class data sets and in improving performance. The regularization penalizes the weights of the base classifiers when they exhibit second-round-misclassified instances especially those with increased weights near the borderline. With different training examples, the regularization term fluctuates in the training process. As training continues, the regularization term enlarges asymptotically towards one. By integrating stratified sampling with AdaBoost.M1 and SAMME, the performance impact from regularization only was examined. The improvements with different base learners differ and the average accuracy enhancement is on the order of 1.91% to 3.94% with the maximum being 24.7%.

Despite the superior performance of the boosting method, there are things to consider. In particular, two parameters need to be specified for the algorithm, which are the sampling size and the number of iterations. Balanced training experimented different sampling sizes; however, it could not be concluded whether larger or smaller sampling sizes delivered a better performance. For imbalanced training, the sizes of the least minority class could
be too small which would create a small training size that deteriorates the performance. Additionally, this small training size limits the capability of the regularization parameter in reducing second-round-misclassified instances as was the case with the capsule endoscopy data set.

The plan for future work is to study the preferred sampling size for imbalanced and balanced training and whether the sampling size is related to a certain domain. The study can set specific rules for the undersampling sizes when the least minority class is too small. Moreover, the relationship between the number of iterations and the sampling size will be studied. The study will investigate whether using a smaller training size can achieve improved performance with a larger number of training iterations or whether the increased number of iterations results in over fitting.
BIBLIOGRAPHY


[13] Hong Cao, Xiao-Li Li, Yew-Kwong Woon, and See-Kiong Ng, *Integrated oversampling for imbalanced time series classification*, IEEE Transactions on Knowledge and Data Engineering (2013). 12


A. Frank and A. Asuncion, *UCI machine learning repository*, 2010. 28


[47] Haibo He and Edwardo A. Garcia, *Learning from imbalanced data*, IEEE Transactions on Knowledge and Data Engineering 21 (2009), no. 9. 2


[56] S. Kong, J. Heo, B. Abidi, J. Paik, and M. Abidi, *Recent advances in visual and infrared face recognition - a review*, Journal of Computer Vision and Image Understanding 97 (2005), no. 1, 103–135. 15


65


[88] Chao-Ton Su and Yu-Hsiang Hsiao, *An evaluation of the robustness of mts for imbalanced data*, IEEE Transactions on Knowledge and Data Engineering 19 (2007), no. 10. 10


[98] Senzhang Wang, Zhoujun Li, Wenhan Chao, and Qinghua Cao, *Applying adaptive over-sampling technique based on data density and cost-sensitive svm to imbalanced learning*, The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–8. 2, 12

[99] Shuo Wang, Huanhuan Chen, and Xin Yao, *Negative correlation learning for classification ensembles*, The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–8. 14


[105] Gang Wu and Edward Y. Chang, *Kba: kernel boundary alignment considering imbalanced data distribution*, IEEE Transactions on Knowledge and Data Engineering 17 (2005), 786–795. 10


[113] Bo Yuan and Xiaoli Ma, *Sampling + reweighting: Boosting the performance of AdaBoost on imbalanced datasets*, The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–6. 14


