Spring 5-7-2011

Diversified Ensemble Classifiers for Highly Imbalanced Data Learning and their Application in Bioinformatics

Zejin Ding
Computer Science Department, Georgia State University

DIVERSIFIED ENSEMBLE CLASSIFIERS FOR HIGHLY IMBALANCED DATA LEARNING AND THEIR APPLICATION IN BIOINFORMATICS

by

ZEJIN DING

Under the Direction of YANQING ZHANG

ABSTRACT

In this dissertation, the problem of learning from highly imbalanced data is studied. Imbalance data learning is of great importance and challenge in many real applications. Dealing with a minority class normally needs new concepts, observations and solutions in order to fully understand the underlying complicated models. We try to systematically review and solve this special learning task in this dissertation.

We propose a new ensemble learning framework—Diversified Ensemble Classifiers for Imbalanced Data Learning (DECIDL), based on the advantages of existing ensemble imbalanced learning strategies. Our framework combines three learning techniques: a) ensemble learning, b) artificial example generation, and c) diversity construction by reversely data re-labeling. As a meta-learner, DECIDL utilizes general supervised learning algorithms as base learners to build an ensemble committee.

We create a standard benchmark data pool, which contains 30 highly skewed sets with diverse characteristics from different domains, in order to facilitate future research on imbalance data learning. We use this benchmark pool to evaluate and compare our DECIDL framework with several ensemble learning methods, namely under-bagging, over-bagging, SMOTE-bagging, and AdaBoost. Extensive experiments suggest that our DECIDL framework is comparable with other methods. The data sets, experiments and results provide a valuable knowledge base for future research on imbalance learning.

We develop a simple but effective artificial example generation method for data balancing. Two new methods DBEG-ensemble and DECIDL-DBEG are then designed to improve the power of imbalance learning. Experiments show that these two methods are comparable to the state-of-the-art methods, e.g., GSVM-RU and SMOTE-bagging.

Furthermore, we investigate learning on imbalanced data from a new angle—active learning. By combining active learning with the DECIDL framework, we show that the newly designed Active-DECIDL method is very effective for imbalance learning, suggesting the DECIDL framework is very robust and flexible.

Lastly, we apply the proposed learning methods to a real-world bioinformatics problem—protein methylation prediction. Extensive computational results show that the DECIDL method does perform very well for the imbalanced data mining task. Importantly, the experimental results have confirmed our new contributions on this particular data learning problem.

INDEX WORDS: Machine learning, Classification, Imbalanced data learning, Diversified ensemble, Active learning, Artificial data generation, Bioinformatics, Protein methylation

DIVERSIFIED ENSEMBLE CLASSIFIERS FOR HIGHLY IMBALANCED DATA LEARNING AND THEIR APPLICATION IN BIOINFORMATICS

by

ZEJIN DING

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2011

DIVERSIFIED ENSEMBLE CLASSIFIERS FOR HIGHLY IMBALANCED DATA LEARNING AND THEIR APPLICATION IN BIOINFORMATICS

by

ZEJIN DING

DEDICATION

To my parents,

To my grandfather, who cherished me since a kid,

To my grandmother, who loved me the most, but I don’t know her name,

To my love.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. Yanqing Zhang, for being my oracle, master, and best friend. This dissertation and related research works would not have been possible without his help. His support, guidance, and encouragement create an excellent research environment for my research and his invaluable ideas always lead me to explore new directions of research territory. We have numerous times of long discussions on different research topics which are turning into valuable treasure to me. I have been very glad to work with Dr. Yanqing Zhang.

I would like to thank Dr. Yi Pan for his great help during my Ph.D. study and valuable insights in my research topic. He can always quickly point out the key component of a new research problem and give great suggestions for improvement. He always encourages graduate students to do better and better. I am also very grateful to have worked with Dr. Yi Pan during the past years.

Special thanks go to Dr. Raj Sunderraman. He has helped me so many times in many different ways throughout my Ph.D. study. Also, many thanks to Dr. Yichuan Zhao for his significant suggestions to improve the research work in this dissertation. His guidance and comments are invaluable and beyond this dissertation. Special thanks to Dr. Yujun George Zheng for providing data and great collaborations in last several years.

Thanks to the continuous supports from the Computer Science Department and the Molecular Basis of Disease (MBD) program at Georgia State University.

Many thanks to former students and friends in the department, Yuchun Tang, Na'El Abu-Halaweh, Qiong Cheng, Yan Chen, Jeff Chen, Amit Sabnis, Irina Astrovskaya, Hailong Hou, etc.

Special thanks to Mrs. Tammie for helping me a lot with all my academic issues, and Shaochieh for his technical support which allows me to conduct experiments without any extra concerns.

Finally, I would like to appreciate my family members for their unlimited support and love. Thanks my parents for supporting my pursuit of Ph.D. degree without any reservations and waiting patiently. Thanks to my younger brother for being my continuous concern for many years, until recently I have realized that he has grown up because I have been in school for so many years. This dissertation is also dedicated to my grandfather, who passed away when I was in a junior school and put his last hope on me to leave my home village and live a better life, to my grandmother, who loved me the most among other grandchildren and passed away during my Ph.D. study abroad, while I did not know her name. Lastly, I thank my girlfriend for her full support and understanding in finishing this dissertation day and night.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC-PR:	Area Under the Curve – Precision and Recall Curve
AUC-ROC:	Area Under the Curve - Receiver Operating Characteristic Curve
CV:	Cross Validation
DECIDL:	Diversified Ensemble Classifiers for Imbalanced Data Learning
DT:	Decision Trees
GA:	Genetic Algorithm
KNN:	K-nearest Neighbor
LDA:	Linear Discriminant Analysis
MCC:	Matthews correlation coefficient
NBC:	Naïve Bayes Classifier
NN/ANN:	Artificial Neural Networks
SMOTE:	Synthetic Minority Over-sampling Technique
SVM:	Support Vector Machines
PRMT:	Protein Arginine Methyltransferases

Chapter 1: INTRODUCTION

In recent decades, data mining and machine learning have been broadly studied and applied in various domains in order to solve many complicated and important real-world problems, such as bio-medical informatics, pattern recognition, fraud detection, nature language processing, medical diagnosis, and so on. Several advantages of such research include that data mining algorithms can help people efficiently analyze raw data, extract hidden patterns, make right decisions, and discover new knowledge.

People learn knowledge from past experiences, and then apply them to solve future problems. Similarly, data mining methods can discover knowledge from known empirical data, and finally use learned knowledge to make decisions based on future unknown data. Thus, many learning algorithms combine the behavior of nature evolving and that of human thinking to mimic the natural learning, leading to very useful predictive systems. The knowledge learned by such algorithms have different formations, such as meaningful rules, applicable mathematical expressions, hidden models, representative instances, interactive networks, etc.

Formally speaking, data information in data mining domains are normally represented by the **attribute–value system**, or called the **information table** [PS81][ZS96][YYZ05]. The data contains many instances (or objects, examples), and every instance is described by several common meaningful features (or factors, attributes), where each feature is represented by a valid nominal or numerical value. Knowledge is implicitly denoted by the features and instances of data, thus learning algorithms need to disclose that knowledge in an understandable and applicable way. Two particular forms of knowledge are generally studied in research communities. The first form is the predictive cause-effect relation: one particular feature of interest (called dependent class), is believed to be dependent with other independent features. Hence, a learning model tries to reconstruct the relations between other independent features (as inputs) and this dependent target class (as outputs). The second from is the inherent structural relation: certain structural or similar characteristics may exist among partial data instances and features; thus a learning method needs to discover the relationship among those instances, e.g., some instances may form a compact cluster.

Accordingly, there are mainly two types of machine learning directions based on the availability of the dependent class. The first type is called supervised learning, where the dependent class is known in advance; the main task for learning is to discover the relation between other features and this target class. On the other hand, the second type is called unsupervised learning: the data does not provide a dependent target class and a learning method needs to discover certain relations among data instances and features. Furthermore, supervised learning also contains two categories of tasks based on the property of target class to be learned. If the class is with discrete values or descriptive labels, then this learning task is called classification, meaning classifying the instances into right classes; otherwise, if the target class is with numerical continuous values, then it is a regression task, where the learner need to output a predicting value for each instance to match the real target value.

The classification task has been the main direction of machine learning and data mining research for many decades due to its extreme importance in real applications. In this dissertation, we focus on the classification problem. In particular, the **binary classification problem**, where the target class is with two discrete labels, is our special interest.

In fact, numerous classification methods have been be proposed by scientists. To name a few here, K-nearest neighbor method (KNN), Naïve Bayes classifier (NBC), linear discriminant analysis (LDA), artificial neural network (NN), decision tree (DT), support vector machines (SVM), boosting, bagging are among the famous and popular ones [Mit97][DHS01]. Many of these advanced learning algorithms and their variations can lead to very high classification accuracies on empirical training data and new testing data. For example, lots of hybrid learning procedures using SVMs are able to classify gene microarray data with accuracies as high as 90 percent or even 100 percent [GWB+02][TZH07]. Newly published machine learning methods keep boosting the learning performance into new high records. Given this high accuracy trend from new algorithms, it's substantially hard for researchers to design any better classification methods. Besides, the impression of high prediction accuracy gives us a feeling that machine learning methods can correctly solve almost any classification tasks, once given the data information; hence there will be no unsolvable classification problems for the machine learning community.

However, this impression is far from the reality. It is broadly known that there still exist lots of tough problems requiring extraordinary efforts to find effective solutions, such as efficiently learning from enormous data, or data with noises and missing values, and others. One of most challenging problems that are still in wide pursuit by numerous researchers is the imbalanced data learning problem. Imbalanced data means the target classes of the data is skew in distributions; that is, there is at least one class of instances which significantly outnumbers other classes. Traditional classification methods, seeking to minimize the overall error rate of the whole training set, do not perform well on imbalanced data, since they generally assume a relatively balanced class distribution, and put too much strength on the majority class. In this dissertation, we are particularly interested in the imbalanced binary classification problem and try to propose effective methods to solve this problem.

1.1 Problem Definition

We first give a relatively formal definition for general binary classification problem, and then introduce the imbalanced binary classification problem.

Binary classification problem. Given an information table in data matrix D, which contains m i.i.d. instances: $(X, Y) = (x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$, where each $x_i = R^n$ $(i = 1, \dots, m)$ is a vector with n feature values , $y_i = \{-1, 1\}$ is the dependant target class, $X = (x_1;\ x_2;\ \dots, x_m)$, and $Y = (y_1; y_2;\ \dots, y_m)$, assume all data instances x_i are mutually different and the class label y_i for each x_i is unique, then a binary classification problem is to find a classifier that explicitly or implicitly represents a decision function $f(X, \Theta)$ such that $Y = f(X, \Theta)$, where Θ is a vector of unknown parameters.

The performance of a classifier needs to be properly measured in order to select a meaningful and useful one for practical use. A classic performance metric is the accuracy or error rate, which computes the difference between outputs of learned decision function $f(X, \Theta)$ and the real class label Y. In general, high accuracy (or low error rate) means better classification models, although other performances are also important, such as the time and space complexities of the learning and predicting process. The performance needs to be evaluated on a separated label-unseen data set $D' = (X', Y')$ to avoid "cheating". Normally, the original known data D for learning a classifier is called the training data, while the separated data D' for evaluating the classifier is called the testing data. There exist several other popular metrics for measuring the general classification performance, such as sensitivity, specifically, F-measure, etc. Details will be given later.

Very similar to the general classification problem, the imbalanced binary classification problem has only one extra special requirement on the data class distribution: one of the two classes (majority class) has many more instances that the other class (minority class). For convenience, we interchangeably use majority class or negative class, minority class or positive class in this dissertation. Although there is no clear threshold to define how imbalanced of the data is being called "imbalanced", it is generally acknowledged in research community that when studying the imbalanced learning problem, the number of negative instances should be significantly larger than that of positive ones.

Thus, we first try to give a formal definition for imbalanced binary classification problem by setting up a relatively standard threshold on imbalance ratio from statistics point of view. First, we define a standard term to measure the imbalance degree in binary classification data. Assume N^+ is number of minority instances, and N^- is for majority ones.

Imbalance ratio is defined as the ratio between the size of majority class and minority class:

$$\beta = N^- : N^+$$

Obviously, β is always bigger than 1. The bigger it is, the more skewed are the data.

Imbalanced binary classification problem. Given a general binary classification problem, if the imbalanced ratio is no less than 19:1, i.e., if the size of minority class is only 5% of entire data size, we call this learning problem the **significantly imbalanced binary classification problem**, or simply, **imbalance learning** in this dissertation.

To our best knowledge, this is the first time in literature that a clear threshold is given to define the imbalance learning problem. Note that: 1) this threshold can be easily extended to imbalanced multiple classification problems; 2) we borrow this "5%" from the significance level in statistical testing; 3) this threshold only limits the scopes of theoretical study on imbalance learning, but experimental studies can still be conducted in less significantly or moderately imbalanced data. To compare different algorithms for imbalance learning, we use less imbalanced data in our experiments as well.

Due to significant data imbalance, the traditional evaluation metrics (accuracy or error rate) for classification performance are not suitable for measuring imbalanced learning results. For example, given a data set with 19:1 imbalance ratio, a learning method can still reach 95% accuracy by simply classifying everything into the majority class. Obviously, such method would be useless in a practical situation. Thus, assuming equally misclassification cost for evaluation is not suitable for imbalanced data classification. New effective performance metrics considering the accuracy on both minority and majority classes are needed for imbalanced data learning. Fortunately, such metrics have already been proposed in literature, such as the F-measure, G-means, AUROC, etc., and we will discuss them in more details in later chapters.

1.2 Challenges

There are several challenges for studying imbalanced classification problem, from both theoretic and experimental point of view.

Challenge 1: the nature of data imbalance is often unclear or different case by case. On one hand, categories in the known data set could be naturally skewed, due to the formation of the problem.

For example, fraud detection, network intrusion or oil-spill detection is naturally forming two imbalanced classes, while the minority class is always of higher interest. However, in other cases, the imbalance may be caused by data collection procedure. The original data distribution of an interested problem may be balanced, but due to practical issues, like time, or storage, or cost, the collected data set is imbalanced. Together, the “intrinsic” and “extrinsic” imbalance needs different learning strategies to build accurate classifiers for future events.

Challenge 2: the data concept of the rare class can be of extreme complexity. Many situations of data complexity can hamper learning on the minority class [HG09]. 1) Between-class imbalance or within-class imbalance. The within-class imbalance occurs when there are several variously sized sub-clusters or sub-concepts within a single class. Such internal imbalance brings another level of complication for general classifiers to learn inner concepts. Together, the concepts formed within- or between-classes will significantly hinder the generalization ability of any learning algorithms. 2) Data or concept overlapping. The knowledge represented in majority and minority class may have some overlapping, thus the resulting data also contains overlapped instances in the data space, which makes the learning to separate them extremely hard. Even a significantly tuned boundary can be drawn between them, such classifiers will be over-fitted and their generalization ability will be jeopardized on future data.

Challenge 3: various data characteristics may bring severe obstacles for learning. 1) High dimensionality. Data sets with high dimensions and small sizes are already very challenging for general classification; with additional imbalance property, it is much more difficult to develop effective learning methods. 2) Data inadequateness (or rare cases). The interested data may be insufficient or extremely imbalanced such that the minority class has only a few instances, which are not enough to represent the minority concept. Consequently, the learned concepts from classifiers may be far from the truth. 3) Noise or missing values. Data noises and missing values have substantial negative impacts for imbalance learning, especially for the minority class. Identifying real knowledge from those rare cases is more difficult.

Challenge 4: no uniform datasets and evaluation metrics have ever been set up widely to measure the performance of imbalance learning classifiers. Broad attentions on imbalanced data classification have only started for about 10 years, yet standard criteria to formalize the questions and evaluate the solutions have not been well established. Current research has been focused on how to develop new algorithms on imbalanced data, while forgetting basic studies on providing fundamental facilities for future research, such as building standard and comprehensive benchmark datasets, or setting up uniform metrics for performance evaluation, etc. Although various preliminary studies have been initialized in recent years, without a clear standard data evaluation platform, the development of imbalance learning will be inefficient.

1.3 Organizations

The rest of this dissertation is organized as follows.

Chapter 2: Literature review. We provide an extensive literature review on imbalance data learning and introduce many popular methods to solve this problem, including randomly under-sampling, synthetic over-sampling, bagging, boosting, cost-sensitive learning and kernel-based learning. Several effective metrics for evaluating classification performance on imbalanced data are also reviewed.

Chapter 3: The DECIDL framework. We present the DECIDL framework in details and discuss several related techniques for implementing the algorithms.

Chapter 4: Experimental studies on DECIDL. A comprehensive public benchmark data pool is created to include 30 highly imbalanced data sets from various domains. We set up the environment and experiment to examine the performance of our DECIDL method, and several other ensemble methods, including under-sampling, over-sampling, SMOTE-bagging and AdaBoost. Comprehensive results are reported and discussed.

Chapter 5: More effective artificial example generation for data balancing. We develop a new distribution-based artificial example generation method and embed it into DECIDL to enhance the classification performance. Additional experiments have been conducted and results have confirmed the effectiveness of this new approach.

Chapter 6: Active example selection for DECIDL. We use active learning to solve the imbalanced data scenarios, and combine it with DECIDL. As active learning is effective in selecting useful examples in training data, it can reduce the examples needed from majority class. The performances of Active-DECIDL on the benchmark data pool have shown a great success in learning on imbalanced data.

Chapter 7: Protein methylation prediction. We apply our DECIDL to solve a real-world learning problem—protein methylation prediction. We introduce the basic knowledge about this problem and discuss several approaches of data representations for protein sequences. Computational results and laboratorial results are presented to confirm our contributions on this topic.

Chapter 8: Conclusions and future work. Lastly, we review the contributions of this dissertation and discuss several future research directions on imbalance learning with DECIDL.

Chapter 2: LITERATURE REVIEW

In this chapter, we briefly review several standard and ensemble classification algorithms that are well developed and popularly used in real applications. Next, we systematically review many existing learning approaches on imbalanced data. Those methods fall into three categories based on their strategies in manipulating the data and conducting the induction. Finally, several typical performance evaluation metrics on skewly distributed data are introduced.

2.1 Standard Classification Algorithms

As we mentioned earlier, machine learning and classification have been studied for several decades, and thousands of learning algorithms were proposed for binary and multiple data classification. Many basic algorithms have been comprehensively and theoretically studied, and become standard and classic machine learning topics, attracting enormous variations and successful applications. Popular standard classification methods include KNN (K-nearest neighbors' algorithm) [CH67], NBC (Naïve Bayes classifier), ANN (Artificial neural network), SVM (Support Vector Machine) [Vap95][Bur98], DT (Decision Tree) [Qui86] [Qui93], LDA (Linear discriminant analysis) [McL04], and more. More theoretical and empirical studies can also be found in [Mit97], [DHS01], [CN06].

2.2 Ensemble Classification

The ensemble method for machine learning is a methodological level of learning strategy. The basic idea of ensemble classification is to combine the strengths of several individual classifiers to achieve higher performance. There are several approaches to choose and create the classifier members, as well as combine them; therefore, many directions can be explored to gain more benefits from ensemble classification. The advantages of classification ensemble can also be explained in statistics with the concept of bias and variance [Die00a] [NIH97] [OM99]. Generally speaking, combining several classifiers together will reduce their variance and improve their generalization ability.

The success of classification ensemble lies in the diversity of ensemble members. If all the component classifiers contain the same learned knowledge, then such integration will shrink to a single classifier without improvement. On the other hand, ensemble with severe diversified classifiers may not produce good performance either; since the true label of each instance is fixed, too many disagreements on it will lead to no agreement, or easily to the wrong one. Therefore, accurately building every individual classifier and then diversely choosing from them is considered as a good strategy for developing ensemble classification methods in practice. While accuracy in individual learner can be easily manipulated in standard classification algorithms, the main direction for ensemble learning is to introduce better diversification.

Several straightforward strategies have been developed for diversification, including: 1) using different subsets of training data (instance subsets or feature subsets) with a single uniform algorithm; 2) using different parameters of a single method on the same training data set; 3) using different learning methods. From the implementation point of view, the construction of classifier committee can be: 1) parallel, where each classifier is independently created; 2) hierarchical or cascade, where constructing new classifiers need information from previous ones, e.g., the training accuracy of current classifier is used to weight the learning focus of next classifier. Three ensemble strategies are frequently studied in literature are bagging, boosting, and random forest [OM99] [Die00a] [Pol06].

Bagging. Bagging, also referred as bootstrap aggregating, creates base learners on many data subsets that are uniformly sampled from the original data, and then uses a linear combination to aggregate them [Bre96]. The combination strategy can be equal-weighted (majority voting) or weighted based on the training performance. Standard bagging uses a “with replacement” method for sampling and on average each sampled subset only contains 63.2% non-duplicated instances of the original data. Obviously, bagging maintains the diversity by generating different sampling subsets, and uses a uniform learner algorithm for all subsets.

Bagging method improves the generalization error by reducing the variance of base classifiers. However, researches have shown that bagging method prefers unstable base learners than stable ones, in order to improve classification performance [Dom97][Dav04][TSK06]. Unstable learners refer those algorithms that have noticeable changes in the learned models even a small perturbation in the training data happens. Thus, bagging is effective when the base classifier is sensitive to data, such as decision trees, or neural networks, etc.

Boosting. Boosting involves incrementally building ensemble members by training each new classifier to emphasize the incorrect learned instances in previous training iterations. Initially, each instance gets equal weight in the learning model. The instances that are not correctly learned will receive more weights of learning in the next round. The classifier will spend more efforts on training those high weighted instances. Iteratively, several base classifiers are sequentially built and a weighted vote is used to combine them for future predictions, where the weight of each base model is proportional to its accuracy on the training set. Boosting method assigns different meaningful weights to instances so as to create diversified learning models. Meanwhile, boosting requires the learning method can handle instances with different weights; if the learner cannot handle, an alternative solution is to use the weights as a selection distribution to randomly draw a new training set from original training data for the base learner.

Boosting methods are able to significantly improve classification performance in many applications, since they can both reduce variations and biases for modeling effectively. One of the most popular boosting methods is the AdaBoost introduced by Freund and Schapire in 1996[FS96]. Given a data set $\{X, Y\}$ and a base learner H_i, the pseudo code of AdaBoost can be described in Table 1.1.

Table 1.1: AdaBoost Algorithm [FS96]

```
Algorithm 1: The AdaBoost.M1 Algorithm
Input:
        $H$ – Base weak learner (e.g., C4.5, NBC, Decision Stump)
        $D$ – $\langle X, Y\rangle$, Training set, $m$ examples $(x_i, y_i)$, $x_i$ is n-dim vector, and $y_i \in Y = \{1,2,\dots,nc\}$
        $max\_iter$ – Maximum number of iterations for boosting
Steps:
  1. Initialize the distribution of weights for all examples as $W_1(x_j) = \frac{1}{m}, for\ x_j \in X$
  2. For $iter = 1: max\_iter$
  3.     Train a base learner with current distribution, $C_i = H(D, W_i)$
  4.     Calculate the error rate of $C_i$, $\varepsilon_i = \sum_{C_i(x_j)\neq y_j} W_i(x_j)$
  5.     If $\varepsilon_i > 1/2$, then set $iter = iter - 1$ and abort this loop
  6.     Set $\alpha_i = \frac{1}{2}\ln(1-\varepsilon_i)/\varepsilon_i$
  7.     Update weights: $D_{i+1}(x_j) = D_i(x_j) * \begin{cases} \exp(-\alpha_i) : & if\, C_i(x_j) = y_j \\ \exp(\alpha_i) : & otherwise \end{cases}$
  8.     Normalize weights, $D_{i+1}(x_j) = \frac{D_{i+1}(x_j)}{\sum_{x_j \in X} D_{i+1}(x_j)}$
Output:
        The final ensemble classifier: $C^*(x) = \arg\max_{y\in Y} \sum_{i:C_i(x)=y} \alpha_i$
```

Using decision trees as base learner in AdaBoost have been practically successful, as seen in [FM99] [Qui96] [Die00b]. However, there are also some drawbacks for boosting methods: boosting method can fail to perform well if there is no sufficient data [Sch99] or the training data contain too much noise [Die00b], which is consistent with boosting theory.

Random Forest. Random forest is a specific ensemble classifier that combines bagging and decision tree learning [Bre01]. A random forest contains many decision tree classifiers where each tree is trained on a randomly sampled feature subset from original data. Suppose there are m instances with n features in the original data set D, and a maximal iteration number $iter$. In each iteration i, randomly select f ($\ll n$) features out of D to create a new data set D_i, then use D_i to build a decision tree tree_i, where each node in the tree is randomly selected from one of the f features. The tree is fully grown without any pruning and the final prediction is the combined result from all trees using a majority voting. To increase more diversity and internal verification, bagging can be used to generate bootstrapped training subsets and validating subsets.

Random forest has shown comparable performance with AdaBoost [Die02] and is robust under noisy data. However, random forests are prone to be overfitting for some data sets, since tree members are fully grown. Besides, it does not perform well when handling large number of irrelevant features since feature subset is randomly chosen for each tree. Nevertheless, random forest is still empirically shown to be a highly accurate classifier in many applications [Ho98][DA06].

2.3 Imbalanced Classification Algorithms

Imbalance data learning have increasingly attracted many researchers for more than a decade. Several special workshops, conferences and issues have been organized successfully drawing grand prosperity of this area. Related important workshops include the Association for the Advancement of Artificial Intelligence) workshop on Learning from Imbalanced Data Sets (AAAI-00) [Jap00], the International Conference on Machine Learning workshop on Learning from Imbalanced Data Sets (ICML-03) [CJK03], and the Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining Explorations (ACM SIGKDD Explorations-04) [CJK04]. Numerous methods have been developed to handle the imbalanced binary and multiple classification problems in literature. To give a comprehensive review on imbalanced classification algorithms, we categorize them into two main classes based on their strengths of handling imbalance. The first kind is the external or data-level methods, where imbalance is handled on the data level by re-balancing or re-weighting the data distribution, such as sampling, boosting, and bagging. The second kind is the internal or algorithm-level methods, where the focus is on the induction or learning phase inside the basic learner (as in the algorithm level), and this category includes cost-sensitive learning [FSZ+99], kernel-based algorithms [TZC+09], recognition-based algorithms [ZD06], etc. Other existing review and categorization on imbalanced data learning can also be found in [JS02] [BPM04] [KKP06] [HG09] [NBP09].

2.3.1 Sampling methods for imbalance learning

Sampling methods are one of the main approaches for imbalance data learning, which try to re-balance the data distribution by adding new instances (over-sampling) or removing existing instances (under-sampling), or their combinations. These methods directly modify the prior distributions of majority class and minority class in the training data set such that traditional classification algorithms can still work well on them. Instances can be added or removed randomly, or intentionally based on certain data characteristics. Three sub-classes of sampling methods exist, including random over- or under-sampling, informed under-sampling, and synthetic over-sampling.

2.3.1.1 Random over-sampling or under-sampling

Random sampling uses the simple randomness to create desired distributions between two classes. Random oversampling will randomly replicate the instances from minority class and add them back to the training set. A **replication percentage** δ can be used to control the amount of duplicated instances; for example, $\delta = 100\%$ means additional N^+ number of replications will be performed resulting in $2 * N^+$ instances for minority class. Therefore, the overall imbalance ratio $\beta = N^-:(1+\delta) * N^+$ can be adjusted by δ accordingly to become a balanced value. On the other hand, random under-sampling randomly selects a subset of instances from majority class and then combines them with the minority class to form a new training data set D', which is then applied with standard classification algorithms. Assume only γ percent of majority instances are selected, the new imbalance ratio is $\beta = \gamma * N^-: N^+$. Obviously, choosing appropriate sampling ratio γ can also lead to balanced data sets.

One important issue for over-sampling is the duplication of minority examples. Since some learning algorithms cannot handle duplicated or weighted instances, over-sampling methods have to be limited to certain weight-awareness learners [KCA03]. Furthermore, a critical observation for random over- or under-sampling is the instability of sampled data sets and subsequent classification models. Therefore, the classification performance on the instable subsets will fluctuate significantly if the learner is sensitive to data. As a result, a practical strategy is to perform several rounds of sampling to create many balanced subsets and classification models, and then use the majority voting to combine them—this procedure is actually the bagging method. Because averaging different classifiers can reduce the variances, the random sampling plus bagging procedure are often used together to solve the imbalance learning in real applications [LWZ06] [PD07].

2.3.1.2 Informed under-sampling

Several advanced strategies have been proposed to help reduce the information loss brought by random under-sampling; examples of these strategies include the ensemble-based under-sampling and clustering-based under-sampling.

Ensemble-based under-sampling: Zhou et al. propose two simple and effective informed under-sampling using ensemble methods: the EasyEnsemble and the BalancedCascade [LWZ06] [LWZ09]. Both methods use the AdaBoost ensemble as the basic learner. The EasyEnsemble is actually using a bagging based under-sampling strategy to create many balanced data subsets and then use ensemble learning models on each subsets; finally it ensemble the entire learned ensemble models together. Obviously, the EasyEnsemble tries to avoid information loss by performing many rounds of samplings. On the other hand, BalancedCascade creates basic learners sequentially, as new learners are built on instances that are filtered by previous learners. Initially, this method builds the first learner L_i on a sampled subset containing partial majority class and the whole minority class; then a new sampled subset from majority class is filtered by L_i such that the correct instances are removed and only incorrect ones are kept; with this refined majority subset and the minority set, a new ensemble learner L_{i+1} is built. Iteratively, more learners are created on filtered sampling data set, and finally all learners are combined together. The BalancedCascade assumes the instances that have been correctly modeled are no longer useful on sub-sequent classifier construction.

The cluster-based sampling utilizes the benefits of data representations with cluster centers [JJ04] [YL09]. First, a simple clustering method, such as *k-means* algorithm, or fuzzy *k-means*, is performed on the majority class and k cluster centers are collected. Then, classification methods are applied to the new data sets containing majority class centers and minority instances. Directly set $k = N^+$ will lead to an even balanced distribution (may not be optimal though). However, k centers should be closely representing the concepts of majority class to lead to better classification modeling. Hence, selecting suitable k is challenging in this method. Clustering algorithms are also used for over-sampling as in the cluster-based over-sampling (CBO) method [JJ04]. CBO first performs *k-means* clustering on both classes with equal number of centers k. When clustering is finished, CBO finds the size of maximum cluster in the majority class, and over-sample every other cluster within the majority class such that they all has the same size. Then, the minority class will also be over-sampled such that new minority class and new majority class has the same cardinality.

2.3.1.3 Synthetic over-sampling

Unlike the random over-sampling that simply replicates minority instances, the synthetic over-sampling methods try to create artificial instances by analyzing the data space of existing minority examples. Ideally, new minority examples should also represent the same minority concepts. One famous synthetic method is the SMOTE method [CHB+02] —synthetic minority oversampling technique, which uses the interpolation technique to create new examples. Specifically, SMOTE uses the following way to generate artificial samples: first, for one x_i in the positive class, find the k nearest neighbors within minority class based on $Euclidian$ distance; next randomly choose one neighbor x_j and a float random number σ in range [0, 1], then the new synthetic example belonging to minority class is:

$$x_{new} = x_i + (x_j - x_i) * \sigma \tag{2.1}$$

Intuitionally, the new example is a point on the line segment joining x_i and one of its randomly selected k-nearest neighbor x_j. By iteratively creating new samples one by one, SMOTE can control the balance ratio of synthetic data and then use general classification methods to build learning models. SMOTE has been shown as one of the most effective over-sampling strategies in several applications [CHB+02]. However, SMOTE also has its own drawbacks, such as over generalization and more computation burden. Several variations of SMOTE have been designed to overcome its side problems, such as the Borderline-SMOTE [HWM05], Adaptive Synthetic Sampling (ADA-SYN)[HBG+08], Kernel-based SMOTE (KSMOTE)[ZG09], and the combination of SMOTE and Tomek link [BPM04].

2.3.2 Bagging methods for imbalance learning

As we discussed in previous sections, bagging has to be involved in random sampling methods to avoid the instability of sampled data sets, especially in the under-sampling situations where lots of majority instances are randomly removed. Obviously, the bagging strategy used in under-sampling for imbalance learning is asymmetric, since only the majority class will be bagged while keeping the minority class untouched. Hence, researchers call it Asymmetric Bagging (AsBagging) in many situations; Li et al. [LML+08] have applied it in the bioinformatics domain with a great success.

To fix the limitations of simple bagging, several variations of bagging have also been developed in literature. For example, Li proposed a Bagging Ensemble Variation (BEV) for classifying imbalanced data [Li07]; BEV creates bagging subsets of majority class by equally splitting them into same-size chunks, where each subset has the same number of majority instances with the total number of minority instances. For example, given N^+ minority instances and N^- majority ones, BEV will randomly split $n = (N^-/N^+)$ subsetsD^i_{maj} , where$\cup\, D^i_{maj} = D_{maj}$, and $D^i_{maj} \cap D^j_{maj} = \emptyset$. Thus, BEV will not produce duplicated majority examples and will not miss any majority instances when building classifiers.

Another variation of bagging is proposed by [HKT09], called Roughly Balanced Bagging (RB Bagging). In this method, binomial distribution with parameter $p = 0.5$ is used to draw a sample from minority or majority class. More specifically, in each iteration, either the majority or the minority class is chosen with equal chance, and then one example is uniformly drawn within the chosen class. Using such a strategy could lead to slightly different number of chosen majority instances, but probably all the minority instances due to its small size. The resulting individual bagging sets may have different sizes, but the overall of them are still balanced.

2.3.3 Boosting methods for imbalance learning

Using boosting strategy for imbalance learning has also attracted lots of interest [SKW+07] [Sun07], because the traditional boosting (e.g., AdaBoost) for general classification tasks are very successful. Similar in general classification, the biggest difference between bagging and boosting in imbalance learning is that boosting sequentially builds improved classifiers one previous ones other than independently. Several popular boosting based strategies for imbalance learning include SMOTE-Boost [CLH+03], RUSBoost [SKH+10], and DataBoost-IM [GV04].

SMOTE-Boost. SMOTE-Boost combines the over-sampling method SMOTE and boosting together [CLH+03]. More specifically, A SMOTE over-sampling procedure is performed first in every round of boosting, and then the traditional AdaBoost method is used to update the weight distributions of all examples accordingly. Simply put, adding SMOTE in the boosting procedure forces the algorithm to focus more on difficult examples in the minority class than in the majority class. Chawla et al. [CLH+03] show that SMOTE-Boost works better than SMOTE and AdaBoost in term of classification performance.

RUSBoost. Opposite to SMOTE-Boost, the RUSBoost combines the random under-sampling with boosting strategy. As claimed by Seiffert et al. [SKH+10], SMOTE-Boost is more complex and time-consuming, especially when the imbalance ratio is huge. Therefore, they proposed to use under-sampling to reduce the training time and use boosting to avoid losing information. In each boosting loop, a random under-sampling is performed to select a small portion of majority class, in order to combine with minority class to form a new training set; then a weak learner is trained and weights are updated according to the learning accuracy. Seiffert et al. [SKH+10] conducted a comprehensive investigation with RUSBoost, SMOTE-Boost, AdaBoost, and SMOTE, and they shows that RUSBoost typically performs as well as or better than SMOTEBoost, besides its less modeling time and easy implementation.

DataBoost-IM. DataBoost-IM is also a hybrid method combining synthetic over-sampling and boosting [GV04]. Compare to SMOTE-Boost, DataBoost-IM synthesize new examples for both majority and minority classes, but much more examples for the minority class. Briefly, DataBoost-IM chooses the hard-to-learn examples as templates for synthesiszation. Initially, each example is assigned with an equal weight. In every iteration, the method first identifies the hard-to-learn examples E_{learn} based on their weights; then it generates synthetic data based on the E_{learn} set and also the class distributions; more minority synthetic examples are produced than majority ones such that new training sets are balanced after combing original data and synthetic data; next, the weak learner is applied to this new training set, and error rate and weight distribution are re-calculated accordingly. Guo and Viktor show that DataBoost-IM is quite effective in dealing with learning from imbalanced data [GV04].

2.3.4 Cost-sensitive learning on imbalanced data

From this section, we now study the internal algorithm-level approaches. The first main category is the cost-sensitive learning framework for imbalance data [Elk01]. Cost-sensitive classification assumes different costs (or penalties) when examples are misclassified from one class to another. A cost matrix can be constructed with values representing penalties between every pair of classes. Let $Cost(i, j)$ represent the cost of predicting an instance in class i as class j. With this notation, $Cost(+,-)$ is the cost of predicting a positive (minority) instance as the negative (majority) instance and $Cost(-,+)$ is the cost of the opposite case. In imbalance learning problem, recognizing positive instances is more important than recognizing negative ones. Hence, the cost of misclassifying a positive instance should be much higher than that of misclassifying a negative one, that is, $Cost(+,-) \gg Cost(-,+)$. Meanwhile, correctly classifying a positive or negative instance costs nothing, i.e., $Cost(+,+) = Cost(-,-) = 0$. Consequently, the objective of cost-sensitive learning is to develop a classifier that can minimize the overall costs on the training set. In many cases, the misclassification penalty for examples in class i is equal for all other classes (in multiple classification situations), thus $Cost(i, j)$ can be simply written as $Cost(i)$, i.e., $Cost^{+}$ and $Cost^{-}$ for binary classification.

Many different approaches have been developed to use cost-sensitive information for imbalance learning [KK98] [Mal03] [MZW05] [LZ06] [ZL06] [HG09]. Those cost-sensitive methods are falling into three classes, as suggested in [HG09]. The first category applies the cost matrix to weight the data space accordingly. In other words, the distribution of the training set will be modified using bootstrap sampling based on the misclassification costs. The modified data distributions are biased towards the minority class due to its higher costs. The second category combines the cost-minimizing techniques with the ensemble schemes. Ensemble methods such as the AdaBoost are incorporated with cost coefficients to develop new cost-sensitive meta-classifiers. Both the first and second categories have been justified by rich theoretical foundations, such as the translation theorem and Metacost framework [Dom99]. Meanwhile, the data space weighting and adaptive boosting are often studied together under the Metacost framework by scientists in order to achieve strong classifiers. For example, Sun et al. [SKW+07] [Sun07] proposed three cost-sensitive AdaBoost algorithms, namely the AdaC1, AdaC2, and AdaC3, where the cost vectors are incorporated in different places of the weight-updating formula in traditional AdaBoost. The third category is the direct cost-sensitive learning, where the cost penalties can be embedded in the algorithm design. Such methods are very specific to particular machine learning algorithms (such as DT or SVM or NN), and thus their strategy of utilizing cost matrix is not able to generalize. Examples of this category include cost-sensitive decision trees [Mal03], cost-sensitive neural networks [KK98] [ZL06], and cost-sensitive SVM (SVM-Weight) [CL01].

2.3.5 Kernel-based learning on imbalanced data

We have mentioned the kernel-based learning method—SVMs using as a cost-sensitive approach (SVM-Weight [CL01]) to solve imbalance learning. Here, we continue to explore several other directions of using kernel-based methods for imbalance classification. In imbalanced data set, the hyperplane learned by SVMs can be pushed far away from the ideal separation plane towards the minority class; therefore, solutions to this issue mainly focus on finding "balanced" hyperplanes and support vectors [RK04] [WC04] [TZ06].

Directly modifying the kernel matrix or kernel function to favor minority class is a direct but challenging method. For example, one famous algorithm in this category is the class-boundary alignment (CBA) method proposed by Wu and Chang [WC04].

The granular support vector machines—repetitive under-sampling (GSVM-RU) is a particular interesting methodology for imbalanced data learning [TZ06] [TZC+09]. Simply speaking, GSVM-RU extracts the support vectors from trained SVM models and uses them as the sampled data subsets. Initially, the first SVM is modeled on the original data set; then the SVs for the majority class are collected to form a new "informative" subset in combination with all minority instances; next, those collected SVs are removed from original data set and a new SVM classifier is modeled. Obviously, iteratively removing those majority SVs and creating new SVM models, GSVM-RU creates many subsets containing informative instances. Finally all subsets are used to form an ensemble classifier or the best round of informative subset is used only to build the best individual classifier. Tang et al. [TZC+09] have shown that GSVM-RU performed very well on many highly imbalanced data sets from different domains.

2.4 Performance Evaluation

Comprehensively evaluating the performance of any learning methods is not only important but also necessary. How to fairly compare the classification performance of different learning methods is a nontrivial task. First, a standard dataset covering a variety of domains and containing significantly different data characteristics (such as sample size, feature size, feature values, etc.) should be considered for testing.

Second, since most learning methods require certain known data sets for training the models, the performance must be blindly tested and compared on independent unknown data sets. A cross-validation (CV) methodology is often used when independent testing data are not available. Given a training data set, equally split the set into k folds and then iteratively choose one fold for testing, and the others for training, until all k folds have been used exactly once for testing. k can be pre-specified by user, and is normally chosen to be 5, 7, 10 in literature. When k equals to the total number of examples in the data set, it is also called leave-one-out cross validation (LOOCV).

Thirdly, the metrics for classification performance comparison are very critical for fairness. Specially, imbalance data requires different metrics to standard balance data due to the fact that minority class is often significantly ignored in standard learning methods. We now review several standard assessment metrics used in balanced data learning, and then introduce some advanced metrics used for imbalance learning.

2.4.1 Classic Evaluation Metrics

Traditionally, the most frequently used metrics are accuracy and error rate. But accuracy is not enough when considering the accuracy of positive predictions. Considering a basic two-class classification problem, the difference between real labels and predicted labels of positive and negative classes can be represented in a two by two confusing matrix, as in Figure 2.1.

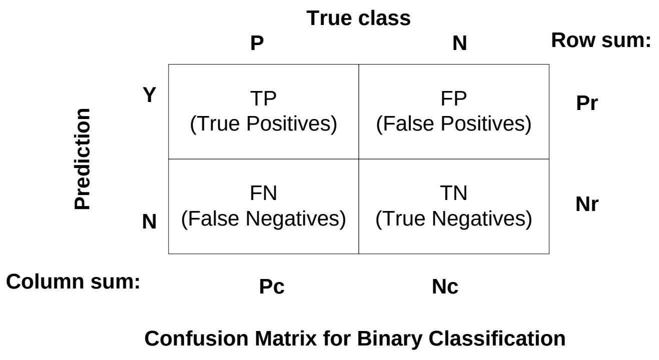


Figure 2.1: Confusion Matrix for Binary Classification

Based on this confusion matrix, several measures can be derived for general classification performance.

$$Accuracy\colon\ ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.2}$$

$$Sensitivity, Recall, True\ Positive\ Rate\colon\ TPR = \frac{TP}{P_c} = \frac{TP}{TP+FN} \tag{2.3}$$

$$Specificity, Ture\ Negative\ Rate\colon TNR = \frac{TN}{N_c} = \frac{TN}{TN+FP} \tag{2.4}$$

$$Pecision, Positive\ Predictive\ Value\colon PPV = \frac{TP}{TP+FP} \tag{2.5}$$

$$Negative\ Predictive\ Value\colon NPV = \frac{TN}{TN+FN} \tag{2.6}$$

Clearly, except the accuracy, all the above metrics mainly consider the performance either on one true class of data or on one predicted class of data, leaving the other part of data ignored. Thus, for imbalanced data learning, new metrics are needed to consider the accuracies on both classes, especially on the minority class.

2.4.2 F-measure, G-Means, and MCC

To fix the “one-side” measurement, several new metrics that combines two or more basic metrics are proposed. For example, the *F-measure* [van79] metric considers both the True Positive Rate (TPR) and the Positive Predictive Value (PPV), which specially focus the learning accuracy on positive class from completeness aspect and efficiency aspect, respectively. The F-measure is written as:

$$F - measure = \frac{2*TPR*PPV}{TPR+PPV} = \frac{2}{\frac{1}{TPR}+\frac{1}{PPV}} \tag{2.7}$$

In other word, F-measure is a harmonic mean between recall and precision.

The *G-Means* criterion considers the performance on both positive class and negative class, and uses the geometric mean to combine them. A high *G-Means* value [KM97] can only be achieved with high prediction accuracy on both positive class and negative class.

$$G - Means = \sqrt{TPR * TNR} \tag{2.8}$$

Lastly, the ***Matthews correlation coefficient*** (MCC) [Mat75] is a strong metric that considers both accuracies and error rates on both classes, since all the four values in the confusion matrix are involved in this formula. A high MCC value means the learner should have high accuracies on positive and negative classes, and also have less misclassification on the two classes. Therefore, MCC can be considered as the best singular assessment metric so far.

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{P_c*N_c*P_r*N_r}} \tag{2.9}$$

2.4.3 AUC-ROC, AUC-PR

So far, all the metrics discussed are based on fixed values of TP, TN, FP, FN, where such values can be easily collected when the class labels and predicted values are both discrete. However, in some other cases, such as the Bayesian network, or some neural network, or some ensemble classifiers, the prediction on testing data are continuous values, and a threshold have to be chosen to discretize them. Shifting the threshold within certain range can produce different groups of TP, TN, FP, FN values. By linking these TP and FP values jointly and plotting them on a 2-D axis, a Receiver Operating Characteristics (ROC) graph [Raw03] [Raw06] is constructed, as in Figure 2.2.

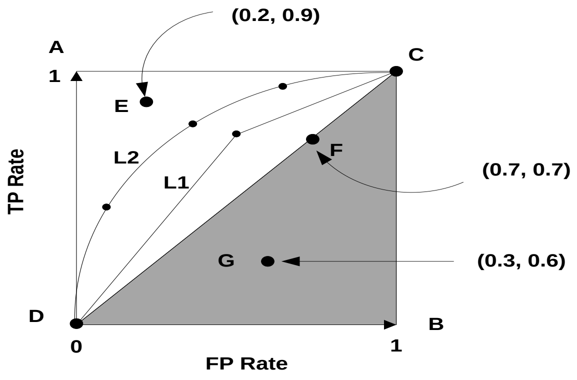


Figure 2.2: AUC-ROC

The idea model should produce a point in Position A—the top left corner, where TPR is 1 and FPR is 0; and the worst model should be the point B at the bottom right corner. Hence, a good classification model should be as close to the top left corner as possible. Meanwhile, a model making random guess will be located on the diagonal, where the TPR and FPR are equal to each other. Note that the point D on the bottom left corner means the classifier predicts every examples as negative, and point C on the top right means all the predictions are positive. The ROC curve is created by connecting all groups of TPR and FPR values and point D and C together. Generally speaking, the ROC graph represents relative trade-offs between gains (true positives) and costs (false positives) over a range of thresholds on a specific classification model.

The closer the ROC curve approaches to the top-left corner, the better the classification performance is. However, directly comparing two or more ROC curves are challenging and impractical, e.g., two curves may be interleaved together and it is hard to claim the better one. Thus, a single numerical value to represent effectiveness of the ROC curve is necessary, which brings the Area under the ROC curves (AUC-ROC). Clearly, the AUC-ROC is ranged from 0 to 1, and the higher it is, the better the classifier.

Although the ROC curve provides a straight visualization for performance evaluation, it also has a particular limitation when it is applied to the highly imbalanced data set. In such cases, the AUC-ROC value may over-estimate the performance of the classifier. For example, if the number of negative examples N_c significantly exceed the number of positive ones P_c, then the FPR rate will not have a noticeable change even the FP values changes drastically, because the denominator (N_c) is very large. Hence, a similar curve using the Precision/Recall values is proposed, the PR-curve. Since both precision and recall are focusing on the classification performance over positive examples, the area under the curve on PR curve (AUC-PR) [DG06] should be more effective on evaluating classification performance on highly imbalanced data.

In this dissertation, we will use several metrics, i.e., F-measure, G-Means, MCC, AUC-ROC and AUC-PR, as performance evaluation measurements.

Chapter 3: THE DECIDL FRAMEWORK

In this chapter, we propose a new meta-learner called Diversified Ensemble Classifiers for Imbalanced Data Learning—the DECIDL ensemble framework. First, we explain the motivation of designing new imbalance learning methods; that is, why we want to start from a new direction—oppositional example synthesis. Since diversity is the key of building successful ensemble classifiers, we then start to design our DECIDL algorithm to build diverse ensemble committees by reversely re-label synthetic data. Pseudo codes are given to show how we approach to the solutions to imbalance learning. Since our method is a meta-learner which is more methodology-oriented than implement-oriented, several closely related concepts and techniques will be discussed to guide the real implementation.

3.1 Motivation

As discussed earlier, the key to develop a successful ensemble method is to build diverse classifiers. In general, two popular directions for diversification are the bagging and boosting methods. In bagging methods, diversification is maintained by creating individual classifiers on different subsets of the training data. While in the boosting algorithms, example distributions are updated iteratively by giving more weights for those previously misclassified examples, and thus diversity means building classifiers on training data with progressively updated distributions. Together, in these two methods, diversity is manipulated by weighting or sampling data distributions on the known training data.

To provide extra diversification to the ensemble construction except from data aspect, the first motivation of our study on imbalance learning is based on the synthetic (or artificial) example generating [CHB+02]. Creating synthetic minority examples has shown to be very successful in imbalanced data learning, such as in the SMOTE [CHB+02] and the DataBoost [GV04]. The benefits for synthesizing new examples are at least two-folds, which both are superior to over-sampling existing examples. First, more diversities can be created by synthesizing new examples than duplicating existing examples, thus better generalization can be achieved potentially. Second, many base learners are unable to process duplicated or weighted examples, such as SVM, or LDA; using over-sampling or boosting methods may limit the applicable scope. However, the challenging point for data synthesis is that new examples may fall into the opposite classes in reality, and deteriorate the generalization performance of learned classifiers.

As a result, we need to evaluate the usefulness of synthesized instances based on existing classifiers. If the new instances can be correctly recognized by current classifiers, then the inclusion of these instances to the original data brings minor or no advantages. In other words, training these instances again will not enlarge the learning power of current models. On the other hand, if the synthetic instances are opposite to predictions of current models, then adding these instances to the original data may help to increase the learning ability of subsequent classifiers. This should be the desired effect for creating synthetic examples. Besides, adding the reversed examples to current classifiers is an explicit action of inducing diversity, as a byproduct effect. However, as discussed above, adding artificial examples (no matter opposite to current models or not) could potentially degrade the generalization power on future data. Furthermore, how to efficiently find the synthetic instances that exactly oppose to predictions of current models seems to be a tough task.

Hence, our second motivation is inspired by the "oppositional re-labeling" technique for general classification tasks in the DECORATE algorithm proposed by Melville [MM04] [Mel05]. In short, the DECORATE creates artificial unlabeled examples and labels them to opposite classes countering to the predictions of existing classifiers; once a new classifier is created on the artificial data and original data, it will be included in the ensemble only if its training accuracy increases. Intuitionally, diversity and accuracy are both maintained in DECORATE.

However, in imbalanced data learning, DECORATE is no longer suitable for three reasons. First, the diversity measure defined in DECORATE for balanced data is no more effective. Clearly, a disagreement on a majority example is much less interesting than that on a minority one, since misclassification on minority examples has higher cost. Second, using the accuracy or error rate as the rejection criterion is not able to distinguish the importance of minority class, and the learned classifiers will be biased towards to the majority class. Third, adding oppositely labeled synthetic examples to majority class may further exaggerate the imbalanced data to more imbalanced. Thus, adding more synthetic instances to majority class is not appropriate.

Hence, based on above motivations, we propose the new DECIDL method for imbalance learning.

3.2 The proposed DECIDL Framework

We propose the Diversified Ensemble Classifier for Imbalanced Data Learning (DECIDL) algorithm, based on the aforementioned motivations on synthetic data generating and reversely example relabeling. The DECIDL algorithm aims to develop an efficient and diverse ensemble committee for imbalanced data in a simple and straightforward manner, as described in Algorithm 2. At first, the ensemble is initialized with a single classifier created by using a base learner on the original data. For imbalance learning, the base learner can be either a general learning algorithm, such as SVM, NN and DT, or a cost-sensitive algorithm, such as SVM-Weight, Cost-NN, and Cost-DT. The training performance is evaluated with a cost-sensitive measurement $Eval$ and noted as E_{best}. Then, artificial unlabeled examples are synthesized one by one based on the feature distributions of original data. More specifically, we first pick an minority example x_i^+ and then find its k nearest neighbors $x_{\{1,\ldots,k\}}^-$ in the majority class; next randomly pick one neighbors x_j^- and a float number σ between 0 and 1, then the new synthetic example is: $x_{new} = x_i^+ + \sigma * (x_i^+ - x_j^-)$. In other words, the artificial examples between majority class and minority class are generated. The number of artificial examples is predefined by giving a ratio Pct to the size of original training set, i.e., $|S| = Pct * |D|$ and S denotes the set of artificial examples. Next, all these examples will be labeled with inverse probabilities to their predictions of current ensemble. For example, for an artificial example s_i, assume its class probabilities predicted by ensemble C^* is $P_{C^*}(y|s_i)$, then the probability of $s_i$being assigned as label y_j is:

$$\hat{P}_{C^*}(y_j|s_i) = \frac{1/P_{C^*}(y_j|s_i)}{\sum_{y=\{1,\dots,nc\}} 1/P_{C^*}(y|s_i)} \tag{3.1}$$

For the binary case, where $y = \{+1, -1\}$, the probability of s_i being assigned to positive and negative classes are:

$$\hat{P}_{C^*}(+|s_i) = \frac{\frac{1}{P_{C^*}(+|s_i)}}{\frac{1}{P_{C^*}(+|s_i)} + \frac{1}{P_{C^*}(-|s_i)}}, \hat{P}_{C^*}(-|s_i) = \frac{\frac{1}{P_{C^*}(-|s_i)}}{\frac{1}{P_{C^*}(+|s_i)} + \frac{1}{P_{C^*}(-|s_i)}} \tag{3.2}$$

Notice that if $\sum_{y=\{1,\dots,nc\}} P_{C^*}(y|s_i) = 1$, then $\hat{P}_{C^*}(+|s_i) = P_{C^*}(-|s_i)$ and $\hat{P}_{C^*}(+|s_i) = P_{C^*}(-|s_i)$. That is, the re-assigned labels are exactly opposite to the predictions of current ensemble. Let assume the set of new labeled artificial examples is $\hat{S}$. Next, we remove examples from $\hat{S}$ that are marked as majority to get a set of minority artificial set $\hat{S}_{mino}$. Then, we combine this set $\hat{S}_{mino}$ with the original training set D together, and use the base learner to build a new classifier C_{i+1}. Again, the classification performance E' of C_{i+1} is evaluated on the original data D. If E' is no more than E_{best}, then this classifier C_{i+1} is permitted to join the ensemble; otherwise, if the performance decrease, then C_{i+1} is rejected. Now, this iteration round is over and a new iteration begins.

In short, every round includes the following important steps: 1) creating artificial unlabeled examples based on training data; 2) predicting their label probabilities with current ensemble; 3) re-assigning new labels to them with inversed predicted probabilities; 4) removing majority-labeled examples and combining the rest with original training data; 5) build a new classifier; 6) and evaluating new classifier for acceptance. This process is repeated until some conditions are reached, such as the committee size mem or the number of iteration $iter$ exceeds certain prefixed numbers.

Once the training process finishes, a classifier ensemble C^* is returned. During the testing process, class with majority votes or maximum probability is applied to classify the unlabeled testing example. For example, assume there are finally mem classifiers in the ensemble C^*, then the probability of unseen example x belonging to class y_i is:

$$P_{C^*}(y_j|x) = \frac{\sum_{C_i \in C^*} P_{C_i}(y_j|x)}{mem} \tag{3.3}$$

and the final class prediction of x is:

$$P_{C^*}(x) = \arg\max_{y_i=\{1,2,\dots,nc\}} P_{C^*}(y_j|x) = \arg\max_{y_i=\{+1,-1\}} P_{C^*}(y_j|x) \tag{3.4}$$

Table 3.1: The DEDICL Algorithm

```
Algorithm 2: The DECIDL algorithm
Inputs:
    $Learner$   – Base learning algorithm (e.g., C4.5, SVM, NN, cost-C4.5, SVM-weight)
    $D$         –$\langle X, Y\rangle$, Training set, $m$ examples$(x_i, y_i)$, $x_i$ is n-dim vector, and $y_i = \{1,2,\dots,nc\}$
    $Cost$      – A $(nc * 1)$ vector, misclassification cost for each class (default:$Cost = I_{nc*1}$)
    $max\_mem$  – Maximum number of member classifiers in the targeted ensemble
    $max\_iter$ – Maximum number of iterations to build an ensemble
    $Eval$      – Evaluation metric for ensemble performance (e.g., Total Cost, or 1-MCC)
    $Pct$       – Percentage (ratio to size of training data D) of synthetic examples to create
Steps:
    Initialization:
    1.  $mem = 1, i = 1$
    2.  $C_i = Learner\ (D, Cost)$
    3.  Initialize ensemble, $C^* = \{C_i\}$
    4.  Computer ensemble performance: $E_{best} = Eval\ (C^*(X),\ Y,\ Cost)$
        Loop:
    5.  While $mem < max\_mem$ and $i < max\_iter$
    6.      Artificially generate data set $S$ with $Pct * |D|$ new training examples that randomly lie be-
            tween majority class and minority class
    7.      Use current ensemble $C^*$ to make probability predictions on $S$, to get $P(S)$
    8.      Label examples in $S$ with probability of class labels inversely to $P(S)$
    9.      Remove the examples in $S$ labeled as majority class to get S'
    10.     $D' = D \cup S'$
    11.     $C' = Learner\ (D',\ Cost)$
    12.     $C^* = C^* \cup \{C'\}$
    13.     Computer new ensemble performance: $E' = Eval\ (C^*(X),\ Y,\ Cost)$
    14.     If $E' \leq E_{best}$ (i.e., performance is better)
    15.         $mem = mem + 1,\ E_{best} = E'$
    16.     Else:
    17.         $C^* = C^* - \{C'\}$
    18.     $i = i + 1$
Outputs:
        Ensemble classifier $C^*$
```

Since the DECIDL is a general ensemble framework so far, there are several important implementing techniques needing further discussions for its real deployments.

3.3 Techniques for Implementing DECIDL

Internal Evaluation Metric. The first task for implementation is to choose a fast evaluation metric that is used inside of this algorithm. The metric should return a single value indicating the performance of ensemble results $C^{*}(X)$ against the training set $< X, Y >$, since the main reason of using such metric is to reject or accept a new classifier. As an internal measurement, efficiency and cost sensitivity is required. Thus, we propose the following two simple metrics for DECIDL implementation. The first one requires cost vector for different classes, while the second one is for situations with no cost information available.

Internal Evaluation **Metrics 1** (with costs available):

$$Eval\ (C^{*}(X), Y, CostMat) = TotalCost(C^{*}(X), Y, CostMat)$$

$$= \sum_{C^{*}(x_j) \neq y_j}^{j=1,\dots,m} Cost(\ y_j) = Cost(+) * FN + Cost(-) * FP \qquad (3.5)$$

Internal Evaluation **Metrics 2** (without cost information):

$$Eval\ (C^{*}(X), Y, Cost) = 1 - MCC(C^{*}(X), Y) = 1 - \frac{TP*TN-FP*FN}{\sqrt{P_C*N_C*P_r*N_r}} \qquad (3.6)$$

Obviously, both metrics can be efficiently calculated when all the parameters are available. Based on the value of internal evaluation metric, a new classifier is accepted if a lower value appears.

Artificial Example Generation. New artificial examples are generated based on the feature values between minority examples and majority examples of the original training data. Examples and features within an ensemble are both considered independently. It is possible that certain features may correlate together and using joint probability distribution may produce more accurate examples. However, such analysis requires more computation costs and large amount of data to estimate relations among features, which are not practical in reality. A similar strategy used in SMOTE is adapted for artificial example generation. However, our assumption is that the data spaces between minority class and majority class are worthy of more investigation and clarification to build a separating model, which is different with the interpolation strategy within minority class used in SMOTE. To generate a new feature value, we first randomly select an minority example x_i^{+}, and find its k nearest neighbors in the majority class $x_{\{1,\dots,k\}}^{-}$. Then, we perform the interpolation between this example x_i^{+} and one of its majority neighbor x_j^{-}, that is the new example is $x_{new} = x_i^{+} + \sigma * (x_i^{+} - x_j^{-})$ where σ is uniformly drawn form range [0, 1]. If the feature is continuous (numeric), this formula can be directly used. However, if the feature is a nominal or discrete value, the new value will be randomly chosen between the two values. Meanwhile, to allow other discrete values of this feature also have certain chances of being chosen, the feature value distributions on the whole training data set and Laplace smoothing technique [Elk01] are used together to guarantee every value still has a non-zero probability to be selected.

Size of Artificial Set in each iteration round. How many artificial examples should be generated in every round is an interesting question. Before giving an exact percentage over the original size, we need to figure out how many artificial examples are truly used in building the individual classifier after removing some of them. In fact, both questions are hard to answer, since it depends on the actual data set and the base learning classifier. Different data may have underlying unknown distributions which make the artificial examples prone to a special class; on the other hand, different base learners also have different tolerance levels of imbalance, thus the exact ratio of new artificial examples varies accordingly.

Interestingly, the size of artificial set can still be retained as a fix number, by keeping generating new samples and removing them if falling in majority class. Thus, an even more interesting question is: does it make any performance difference when maintaining the number of artificial examples before removing majority-labeled ones and after? Hypothetically, such differences should exist, since artificial examples are totally determined by the ensemble, thus more hypothesis hidden inside the classifiers are imposed in the data distributions if maintaining the size of artificial set after removing majority ones.

Reversely Re-labeling Strategy. Our purpose to reversely re-labeling the artificial examples is to bring the maximum diversity to the ensemble. As described in the DECIDL algorithm, an example will be re-labeled based on a distribution opposite to the predictions of current ensemble; for example, there is a high chance of assigning an artificial instance to the least probable class. In contrast, there is a hard way to implement this procedure: 100% of chance to choose the least probable class to label this instance, without a distribution involved. The "soft" relabeling (with inverse probability) will allow the labels of some examples unchanged or to be assigned to other classes, while the "hard" relabeling always label the example to its least likely class predicted by the ensemble. The impacts of the two strategies are: "soft" relabeling permits certain degree of freedom of data randomization, but the "hard" relabeling extensively relies on the classifier judgment. Experiments will be designed to find out which way is better in real applications, and to discover the potential biases of related base classifiers.

Example Removing Strategy. As described in the standard DECIDL algorithm in Algorithm 2, the synthetic examples that are re-labeled with majority class will be removed before building new expanded training sets and classifiers, due to the already outnumbered majority examples. However, we can still keep them in the new training sets and then build learning models, to find if there are different effects between removing and non-removing.

Classifier Rejection. The standard DECIDL method will reject new classifier member if its performance is decreased based on the two internal evaluation metrics. However, rejecting new member means less diversity for the ensemble. On the other hand, if we continue to include new classifiers without any rejection criteria, the diversity of the ensemble may be increased, while the individual performance of new members may drop. How to balance between diversity and performance is hard to answer. Therefore, we will use both rejection strategies—rejection and non-rejection in our experiments to see any effect changes.

Final Performance Metrics. We have previously discussed several metrics for measuring the overall classification performance of learning methods on imbalanced data. Although all these metrics are highly correlated, each metric still has its own strengths and weakness. Choosing an appropriate metric for particular data may depends on the both the data complexity and the actual learning algorithms. To facilitate metric selections for future research and applications, we will use all the five popular metrics, i.e., F-measure, G-means, MCC, AUC-ROC, AUC-PR, to evaluate the classification results of five ensemble learning strategies (including ours, under-bagging, SMOTE-bagging, over-bagging boosting, etc) in our experimental study.

Diversity and Performance Improvement. The relation between the diversity and performance of ensemble classifiers on balanced data sets have been previously studied in several works, such as [KW03] [BWT05]. However, only from very recently, the studies for diversity and performance of ensemble methods on imbalanced data have just started [BWH+05] [CS07] [WY09a] [WCY10]. As we all know, diversity is not the only reason for high classification performance; for example, the base learner has much more impacts on producing high accuracy. Thus, studying the performance improvement from diversity construction should attract more research investigation for imbalance learning, yet such efforts have not started.

Chapter 4: EXPERIMENTAL STUDIES ON DECIDL

In this chapter, we will verify the performance of our proposed method—the DECIDL framework—on various highly skewed representative data sets. Comparisons will be made among several popular imbalance learning strategies, ranging from sampling methods, ensemble methods, and kernel-based methods. More specifically, the under-bagging (undersampling + bagging), over-bagging (oversampling + bagging), SMOTE, AdaBoost, SVM-weight, and Random Forest will be used for experiments and comparisons. For these ensemble methods, we choose several commonly used base learners to build effective ensembles separately in the experiments. Results have shown that our proposed DECIDL framework are comparable and superior to most advanced imbalanced learning methods on averaged results on 30 data sets.

4.1 Base-learner and algorithm selection

Our first interest of experiment design is to choose an appropriate base learner for ensemble. As we all know, ensemble methods, especially the AdaBoost algorithm, normally work better if the base learner is weak, e.g., the accuracy of an individual model only needs to be better than random guessing. Frequently used base learners in literature include the decision stump (single-feature split) [IL92], naïve Bayes classifier, decision tree C4.5, CART, and even NN. However, recent studies have also started to incorporate strong learning algorithms such as kernel methods [KPJ+02] [YLJ+03] [Zhu08] and even AdaBoost (two layers of ensemble) [LWZ09] into ensemble systems, with certain tuning strategies to avoid over-fitting. Thus, in this study, we also try to use diverse levels of learning models as base learners; particularly, the four base learners, decision stump, C4.5, SVM, and NN, are considered in the experiments.

Furthermore, five diverse ensemble frameworks, namely, under-bagging, over-bagging, SMOTE, AdaBoost, and our DECIDL will be used for experimental comparisons. Notice that each ensemble method has its special strategy to build the ensemble committee. Therefore, the total number of ensemble learning algorithms is 5*4=20. Besides, two individual approaches, the random forest and SVM-weight, are also considered for experiments. Thus, together we will at least compare 22 different algorithms, without mentioning various parameters used in each algorithm. An overview of experimental comparison framework is shown in later section, as in Figure 4.1 and Figure 4.2.

4.2 Dataset Selection

Our focus is the binary classification task, thus all the data sets in this benchmark pool are required to meaningfully represent practical classification problems, plus satisfying the imbalanced data distributions. More specifically, all the related problems are predictive tasks with multivariate hidden relations between independent features and dependent labels; the data sets naturally (or at least presumably) form multivariate data types which can extensively and meaningfully represent the targeted problems. Regression problems, or clustering problems, or other problems with sequential or time-series data properties, are not within the scope of this dissertation. However, special problems from bioinformatics (such as the protein structure prediction) are still within the consideration, since those problems are generally considered as classification tasks after proper data transformation (e.g., protein structures are believed to be determined only by their neighboring amino acids). With algorithms being selected for experiments, we now propose several criteria to select a standard and comprehensive benchmark dataset from public machine learning repositories for testing imbalance learning algorithms.

The first criterion to consider is the **imbalance ratio β**. Although we have defined that for (significantly) imbalanced data, the ratio should be no less than 19:1, in the actual experimental settings, some non-significant imbalanced data should also be tested, in order to find how the learning algorithms behave under variety of imbalance degrees. Thus, we suggest the benchmark data sets contain different imbalance levels, i.e., the imbalance ratio from 10:1 to 1000:1 or more. More specifically, we recommend the following imbalance ratios included in the benchmark data pool, β =10:1, 20:1, 50:1, 100:1, as considered in our selection. Note that this special criterion is exclusive in differentiating general data learning benchmark set and our imbalanced data learning benchmark set.

The dimensions of data are also very important for learning, as considered as the second criterion. **Number of instances and features** should vary dramatically from few of them to plenty of them. Generally, the fewer instances the data contain, the easier the learning algorithm gets over-fitting; while the more the data, the longer it takes to train a learning model. From feature aspect, fewer features mean less data complexities, thus easier to train a model; while more features mean more data complexities, and harder to reveal the true knowledge behind the data. Thus, with various sizes of instances and features, the learning ability and classification performance of different algorithms will be extensively challenged. Considering the data acquisition cost and problem complexity in many real applications, we recommend that the total number of instances in the benchmark datasets changes from hundreds to hundreds of thousands, and the feature quality varies from less than ten to thousands, which are both very realistic in practical problems.

Feature value properties. The format of feature values also has strong impacts in building classification models. Two main formats are generally used in the representation of information data: one is the numeric continuous value, and the other is nominal categorical value. Numeric values are mutually computable, while nominal values normally represent descriptive information, thus mathematics computation on them is meaningless. From algorithm aspect, general classification algorithms are either in favor of continuous data or discrete data. Therefore, data pre-processing procedure, such as discretization or binarization, is frequently involved in preparing data for particular learning algorithms. Consequently, to broadly test general imbalanced learning techniques, we suggest the features of benchmark set contains only continuous values, only nominal values, and mixed values of the two kinds.

Data domains. Data and problems from various domains will naturally bring different levels of data complexities and difficulties for learning. The relations between independent features and the target class vary dramatically from one domain to another domain. For example, a linear relation may exist between the education level and salary level for general people, while the pathogenesis of cancer may depends on the statuses of thousands of genes. Meanwhile, testing classification performance on various kinds of data resources is beneficial for future researchers to select appropriate learning algorithms for problems in their own domains. For example, SVM is frequently used for classifying microarray data only since 1999 [BGL+99] while it was invented around 1995 in [Vap95]. Hence, we suggest that the benchmark data should contain classification problems from diverse application domains, such as business, medical diagnosis, biological analysis, pattern recognition area, natural life, environment, etc.

Notice that all the above criteria are trying to bring different configurations of internal data complexities, in order to differentiate the learning ability of various algorithms. No doubtfully, there are other advanced measurements to introduce extra data complexities, such as analyzing the statistical relations among features or discovering cluster relations among examples. However, those extra criteria require much more computations, and sometimes are hard to hierarchically list their relations, as compare to above criteria.

We search the publicly available machine learning repositories (the UCI, and KDD), and data frequently mentioned in publications at IEEE, ACM, SpringLink, and Science Direct. Finally, 30 representative data sets are chosen to construct the standard benchmark pool for imbalance learning. In summary, the benchmark data are from 9 different domains of our society, example size varies from few hundreds to hundreds of thousands, feature size changes from less than ten to tens of hundreds, and imbalance ratio changes from about 9:1 to 130:1. The characteristics of those datasets are carefully summarized in Table 4.1.

Table 4.1: Benchmark Data Characteristics

(Feature types: N: nominal, C: continuous, B: binary.)

Name	Description & Reference	Imb. Ratio	#Examples	#Features	Feature Types	Others
Ecoli	UCI, target:=imU	8.6:1	336	7	7C	Life
Optical Digits	UCI, target: class=8	9.1:1	5,620	64	64C	Computer
SatImage	UCI, satellite, target:=4	9.3:1	6,435	36	36C	Physical
Pen Digits	UCI, target: class=5	9.4:1	10,992	16	16C	Computer
Abalone_7	UCI, target: Ring=7	9.7:1	4,177	8	7C, 1N	Life
Sick Euthyroid	UCI, target: sick euthyroid	9.8:1	3,163	25	7C, 18N	Life
Spectrometer	UCI, target: LRS class>=44	11:1	531	93	93C	Physical
Balance	UCI, target:=balance	12:1	625	4	4N	Social
Car_Eval_34	UCI, target:=good, v good	12:1	1,728	6	6N	Business
ISOLET	UCI, spoken letters, target: letter=A|B	12:1	7,797	617	617C	Computer
US Crime	UCI, crimes, target: freq>0.65	12:1	1,994	122	122C	Social
Yeast_ML8	LIBSVM, multiple labels, target 8	13:1	2,417	103	103C	Life
Scene	LIBSVM, target > one label	13:1	2,407	294	294C	Nature
Libras Move	UCI, target: class=1	14:1	360	90	90C	Physical
Thyroid Sick	UCI, target: class=sick	15:1	3,772	28	7C, 21N	Life
Coil_2000	KDD, CoIL, target: minority	16:1	9,822	85	85C	Social
Arrhythmia	UCI, target: class=06	17:1	452	279	206C, 73N	Biology
Solar Flare M0	UCI, target: M-class>0	19:1	1,389	10	10N	Nature
OIL	UCI, target: minority	22:1	937	49	49C	Environment
Car_Eval_4	UCI, target: class=vgood	26:1	1,728	6	6N	Business
Wine Quality	UCI, wine, target: score<=4	26:1	4,898	11	11C	Business
Letter Img	UCI, target: class=Z	26:1	20,000	16	16C	Computer
Yeast _ME2	UCI, target: class=ME2	28:1	1,484	8	8C	Life
Webpage	LIBSVM, w7a, target: minority	33:1	49,749	300	300B	Web
Ozone Level	UCI, ozone, data	34:1	2,536	72	72C	Environment
Mammography	UCI, target: minority	42:1	11,183	6	6C	Life
Reuters-21578	KDD, text, target: ship	52:1	7,674	17387	17387B	Web
Forest Cover_5	KDD, target class: Aspen	60:1	581,012	54	10C, 44N	Nature
Protein homo.	KDD CUP 2004, minority	111:1	145,751	74	74C	Biology
Abalone_19	UCI, target: Ring=19	130:1	4,177	8	7C, 1N	Life

4.3 Experimental Setup

We develop a simple straightforward comparison framework to perform various algorithms on the proposed diversified benchmark data set, as shown in Figure 4.1. First, five different ensemble meta frameworks for imbalanced data learning are included in the experiments, namely, the under-bagging, over-bagging, SMOTE-bagging, AdaBoost, and DECIDL. Next, four different base classifiers, decision stump, C4.5, SVM, and NN are used as internal weak learners for the meta-ensembles. Note that to save the computational burden, single layer perceptron NNs and linear SVMs are used in real experiments. For ensemble algorithms, different sizes of ensemble committees will be used to find potential influences. Furthermore, we also try two popular direct imbalance learning algorithms, the cost-sensitive SVM (SVM-weight) and the random forest, to show more comparisons. After applying these 22 algorithms on 30 sets to the benchmark data, we compare their imbalance-oriented evaluation metrics, such as the F-measure, G-means, MCC, AUC-ROC, and AUC-PR. Note that each learner has its own parameters; for example, the SVM can select different kernel functions costs. Therefore, trying all combinations of these parameters and algorithms can be computationally lengthy and infeasible, and the default or best recommended values for these parameters are used in the actual experiments.

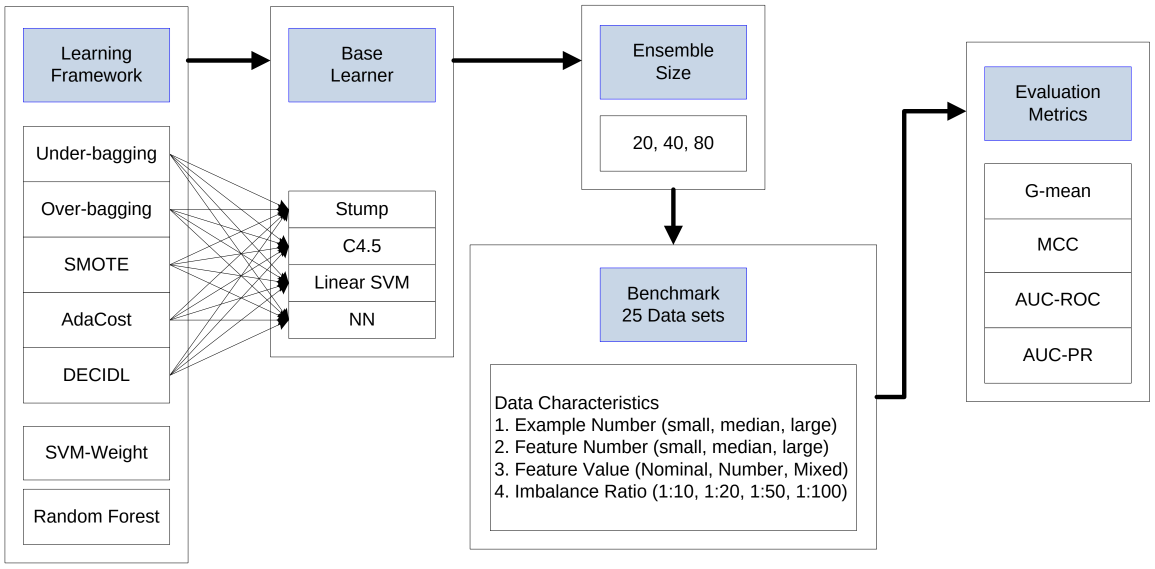
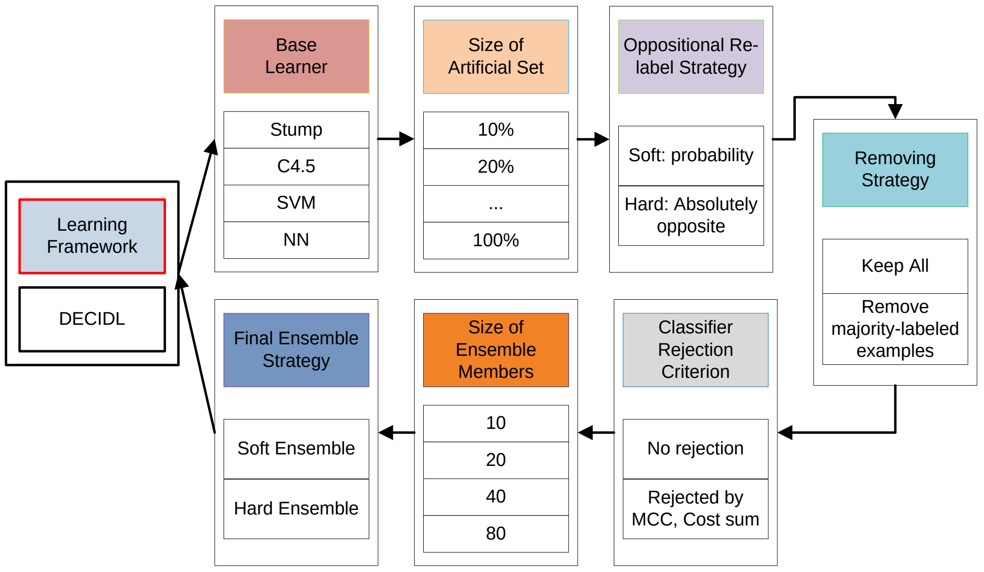


Figure 4.1: Global Experimental Comparison Framework



Experimental Setups for DECIDE

Figure 4.2: The workflow of DECIDL procedure and parameter settings

In addition, to comprehensively test our DECIDL ensemble system, we will extensively test various settings for several important internal steps of the procedure. An expanded experimental setup for the DECIDL method is depicted in Figure 4.2. DECIDL consists of several steps of creating new ensemble classifiers, thus each step requires different parameters to adjust the properties of artificial data set and new classifiers. As can be seen in Figure 4.2, there are 7 basic steps to finish a DECIDL ensemble construction, and every step has its own special tuning strategies. The potential choices for every step are listed in the figure clearly. Hence, the combinations of the whole procedure are quite large, and all individual learning models are extraordinarily diversified. To save computational cost, we only run several critical steps under different parameter settings on partial benchmark data sets to show the trends. Furthermore, when comparing with other ensemble methods, the original default settings of DECIDL will be used for experiments and result comparisons. Default settings for DECIDL are: using uniform distribution for artificial sample generating, 50% size of original data set for artificial set, hard reversely re-labeling strategy, removing majority-labeled examples, rejection by MCC criterion, 15 as the maximum ensemble size and a final hard ensemble.

The performance of each learning algorithm was evaluated using 10 complete runs of 5-fold cross-validation. In each 5-fold cross-validation, each data set is randomly split into 5 equal-size segments and results are averaged over 10 full trials. For each trial, one segment is set aside for testing, while the remaining data is available for training.

4.4 Experimental Results

4.4.1 Overall Comparisons with Different Ensemble Frameworks

First, Table 4.2 and 4.3 show the detailed classification performance of DECIDL on the 30 data sets. We can easily see that DECIDL produce some good performance on several data sets, such as Sick Euthyroid, Spectrometer, Arrhythmia, and Reuters-21578, but also it gives bad predictions on other data sets, such as Balance, Yeast_ML8, Coil_2000, Webpage, and Forest Cover_5. This trend is similar for other ensemble frameworks; for the concision of the dissertation, we do not list all performance of other ensemble frameworks here. This phenomenon strongly suggests the data sets we used are very diverse and complicated. It is hardly possible to design an algorithm that works well for all kinds of problems.

Table 4.4 shows average performance of all 22 algorithms on all 30 data sets with all five evaluation metrics. Each value in the table represents the mean metric values over 30 data sets with 10 times of 5-fold cross validations. As previously mentioned, the parameters used for our DECIDL framework in this comparison are default settings: artificial example generating—Gaussian distribution, size of artificial set—50% of original data set, oppositional re-label strategy—hard, removing strategy—removing majority, classifier rejection—rejected by MCC, ensemble size—15, ensemble strategy—hard. For other four ensemble frameworks, parameters comparable to those in DECIDL are used to show fair final comparisons; for example, the ensemble size is also set as 15, and the oversampling or under-sampling ratio is also set as 0.5.

Figure 4.3-4.7 compared the results under different evaluations metrics. We group the algorithms by the base learner types so that comparisons between different ensemble methods can be easily made. A first observation on these results is that there is no best ensemble and base learner combination in terms of different imbalance classification performance. Every kind of ensemble and base learner only outperforms others on some of five metrics and no uniformly better ones. This implication is consistent with general opinions in research on supervised classification. However, all the five ensemble strategies have significantly improved the classification performance from the base learners on all the five evaluation metrics. This result suggests that ensemble strategies are much more effective than individual learning methods on imbalanced data learning, which is also a generally accepted conclusion in machine learning communities.

Table 4.2: Classification Performance (F-Measure and G-Means) of DECIDL on 30 Data Sets

(DS: decision stump; DT: decision tree; L-SVM: Linear SVM; PNN: perceptron NN)

	F-Measure				G-Means			
DECIDL	**DS**	**DT**	**L-SVM**	**PNN**	**DS**	**DT**	**L-SVM**	**PNN**
Ecoli	0.448	0.581	0.649	0.539	0.831	0.841	0.887	0.817
Optical Digits	0.440	0.910	0.745	0.571	0.664	0.949	0.923	0.858
SatImage	0.424	0.622	0.203	0.208	0.735	0.794	0.707	0.572
Pen Digits	0.475	0.964	0.717	0.541	0.699	0.980	0.895	0.819
Abalone_7	0.309	0.367	0.308	0.284	0.698	0.578	0.696	0.738
Sick Euthyroid	0.630	0.856	0.456	0.570	0.911	0.935	0.906	0.909
Spectrometer	0.496	0.801	0.839	0.496	0.632	0.929	0.942	0.680
Balance	0.000	0.000	0.000	0.011	0.202	0.565	0.670	0.032
Car_Eval_34	0.303	0.930	0.733	0.844	0.692	0.984	0.965	0.911
ISOLET	0.223	0.781	0.627	0.619	0.890	0.928	0.836	0.657
US Crime	0.476	0.509	0.547	0.468	0.698	0.800	0.789	0.836
Yeast_ML8	0.021	0.087	0.000	0.113	0.406	0.543	0.504	0.338
Scene	0.207	0.201	0.268	0.233	0.500	0.441	0.649	0.505
Libras Move	0.405	0.819	0.827	0.628	0.707	0.918	0.892	0.894
Thyroid Sick	0.428	0.837	0.320	0.290	0.882	0.920	0.805	0.759
Coil_2000	0.049	0.115	0.070	0.194	0.612	0.628	0.678	0.470
Arrhythmia	0.668	0.656	0.334	0.223	0.950	0.848	0.647	0.481
Solar Flare M0	0.242	0.070	0.206	0.229	0.646	0.644	0.611	0.595
OIL	0.454	0.583	0.479	0.035	0.789	0.692	0.773	0.404
Car_Eval_4	0.204	0.925	0.677	0.783	0.760	0.991	0.974	0.925
Wine Quality	0.187	0.225	0.110	0.022	0.649	0.407	0.666	0.069
Letter Img	0.472	0.808	0.658	0.514	0.816	0.939	0.839	0.863
Yeast _ME2	0.392	0.337	0.197	0.248	0.785	0.537	0.847	0.692
Webpage	0.077	0.546	0.629	0.520	0.550	0.723	0.837	0.861
Ozone Level	0.208	0.180	0.267	0.151	0.589	0.475	0.819	0.701
Mammography	0.211	0.516	0.369	0.341	0.768	0.767	0.816	0.826
Reuters-21578	0.696	0.759	0.682	0.657	0.857	0.938	0.960	0.976
Forest Cover_5	0.074	0.277	0.112	0.203	0.702	0.414	0.835	0.659
Protein homo.	0.527	0.576	0.574	0.154	0.762	0.793	0.884	0.817
Abalone_19	0.019	0.004	0.018	0.020	0.701	0.645	0.701	0.505
Average	**0.325**	**0.528**	**0.421**	**0.357**	**0.703**	**0.752**	**0.798**	**0.672**

Table 4.3: Classification Performance (MCC and AUC-ROC) of DECIDL on 30 Data Sets

(DS: decision stump; DT: decision tree; L-SVM: Linear SVM; PNN: perceptron NN)

DECIDL	MCC				AUC-ROC			
	DS	DT	L-SVM	PNN	DS	DT	L-SVM	PNN
Ecoli	0.445	0.550	0.626	0.515	0.913	0.942	0.924	0.913
Optical Digits	0.294	0.900	0.726	0.646	0.786	0.991	0.961	0.956
SatImage	0.295	0.580	0.264	0.148	0.766	0.920	0.730	0.690
Pen Digits	0.305	0.961	0.694	0.513	0.750	0.997	0.939	0.905
Abalone_7	0.249	0.308	0.247	0.304	0.760	0.769	0.792	0.775
Sick Euthyroid	0.622	0.843	0.427	0.626	0.920	0.955	0.862	0.712
Spectrometer	0.471	0.789	0.828	0.492	0.761	0.974	0.970	0.856
Balance	0.000	0.000	0.000	-0.026	0.497	0.450	0.500	0.390
Car_Eval_34	0.254	0.926	0.736	0.854	0.708	0.996	0.992	0.988
ISOLET	0.052	0.763	0.631	0.622	0.790	0.976	0.977	0.949
US Crime	0.449	0.472	0.516	0.497	0.889	0.887	0.882	0.903
Yeast_ML8	0.012	0.034	0.075	0.057	0.512	0.567	0.500	0.591
Scene	0.149	0.134	0.203	0.186	0.669	0.666	0.686	0.738
Libras Move	0.392	0.814	0.826	0.624	0.853	0.974	0.940	0.944
Thyroid Sick	0.607	0.828	0.337	0.311	0.940	0.965	0.869	0.881
Coil_2000	0.035	0.082	0.067	0.143	0.649	0.629	0.649	0.647
Arrhythmia	0.681	0.648	0.303	0.179	0.966	0.961	0.849	0.659
Solar Flare M0	0.216	0.089	0.175	0.206	0.659	0.683	0.754	0.755
OIL	0.449	0.593	0.468	0.005	0.895	0.929	0.884	0.623
Car_Eval_4	0.000	0.925	0.700	0.808	0.846	0.999	0.995	0.992
Wine Quality	0.197	0.210	0.143	0.018	0.741	0.776	0.701	0.563
Letter Img	0.473	0.806	0.654	0.533	0.870	0.991	0.960	0.962
Yeast _ME2	0.401	0.327	0.207	0.266	0.894	0.899	0.849	0.872
Webpage	0.105	0.536	0.624	0.536	0.728	0.914	0.939	0.947
Ozone Level	0.200	0.156	0.317	0.109	0.785	0.785	0.865	0.793
Mammography	0.253	0.512	0.399	0.359	0.831	0.896	0.854	0.820
Reuters-21578	0.695	0.764	0.704	0.688	0.910	0.988	0.988	0.994
Forest Cover_5	0.065	0.305	0.124	0.155	0.801	0.756	0.783	0.795
Protein homo.	0.540	0.582	0.600	0.226	0.868	0.940	0.912	0.923
Abalone_19	0.016	0.002	0.020	0.029	0.656	0.695	0.743	0.695
Average	**0.297**	**0.515**	**0.421**	**0.354**	**0.787**	**0.862**	**0.842**	**0.808**

Table 4.4: Overall Average Classification Performance of the 22 Algorithms on 30 Data Sets

Ensemble	Base Learner	Metric Values				
		F-mea.	**G-means**	**MCC**	**AUCROC**	**AUC-PR**
DECIDL	Stump	0.325	0.703	0.297	0.787	0.292
UnderBagging	Stump	0.294	**0.749**	0.292	0.802	0.260
OverBagging	Stump	0.301	0.730	0.294	0.760	0.199
SMOTE	Stump	0.294	0.716	0.294	0.751	0.180
AdaBoost	Stump	**0.378**	0.472	**0.383**	**0.861**	**0.447**
Stump		0.181	0.265	0.190	0.575	0.157
DECIDL	DT	**0.528**	0.752	**0.515**	0.862	**0.534**
UnderBagging	DT	0.396	**0.828**	0.409	**0.885**	0.465
OverBagging	DT	0.520	0.688	0.498	0.819	0.477
SMOTE	DT	0.519	0.713	0.507	0.779	0.369
AdaBoost	DT	0.523	0.616	0.515	0.843	0.528
DT		0.468	0.562	0.462	0.722	0.394
DECIDL	Linear SVM	0.421	0.798	0.421	0.842	0.437
UnderBagging	Linear SVM	0.368	**0.800**	0.373	0.845	0.350
OverBagging	Linear SVM	0.388	0.795	**0.431**	0.816	0.324
SMOTE	Linear SVM	**0.424**	0.799	0.423	**0.866**	**0.453**
AdaBoost	Linear SVM	0.366	0.463	0.355	0.817	0.399
Linear SVM		0.278	0.320	0.281	0.155	0.029
DECIDL	Perceptron NN	0.357	0.672	0.354	**0.808**	**0.370**
UnderBagging	Perceptron NN	0.341	**0.695**	0.335	0.823	0.356
OverBagging	Perceptron NN	**0.383**	0.675	**0.374**	0.803	0.350
SMOTE	Perceptron NN	0.360	0.651	0.355	0.686	0.215
AdaBoost	Perceptron NN	0.325	0.374	0.323	0.790	0.365
Perceptron NN		0.167	0.209	0.184	0.654	0.267
Linear SVM-Weight		0.367	0.429	0.374	0.870	0.475
Random Forest		0.487	0.567	0.493	0.826	0.521

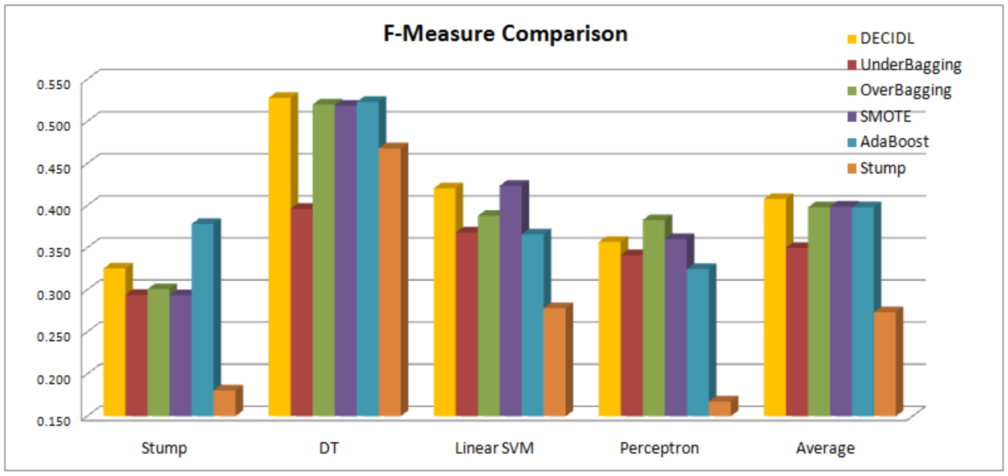


Figure 4.3: F-Measure Comparison of Average Performance on 22 Algorithms on 30 Data Sets

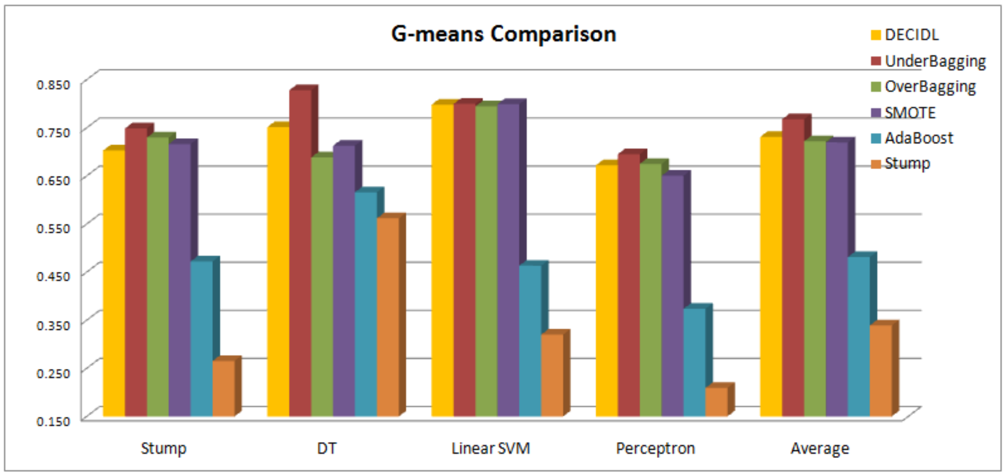


Figure 4.4: G-means Comparison of Average Performance on 22 Algorithms on 30 Data Sets

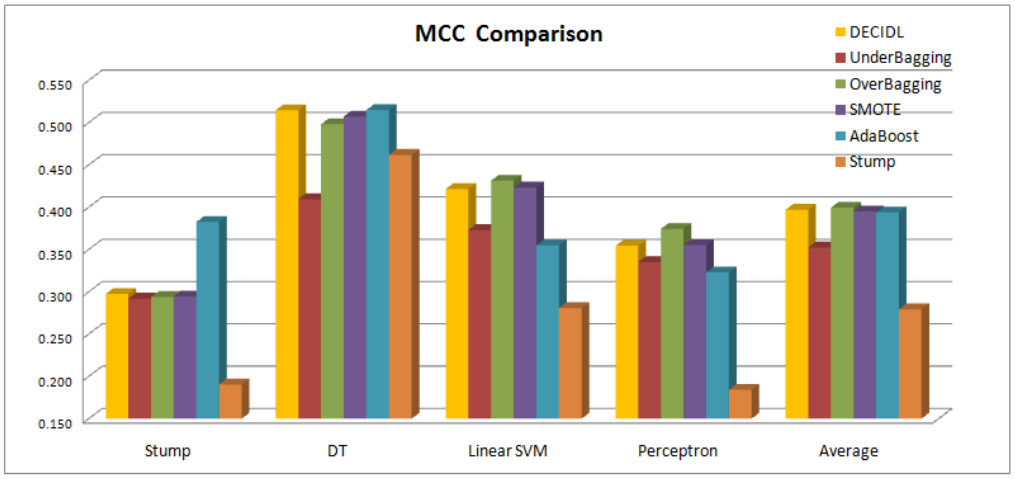


Figure 4.5: MCC Comparison of Average Performance on 22 Algorithms on 30 Data Sets

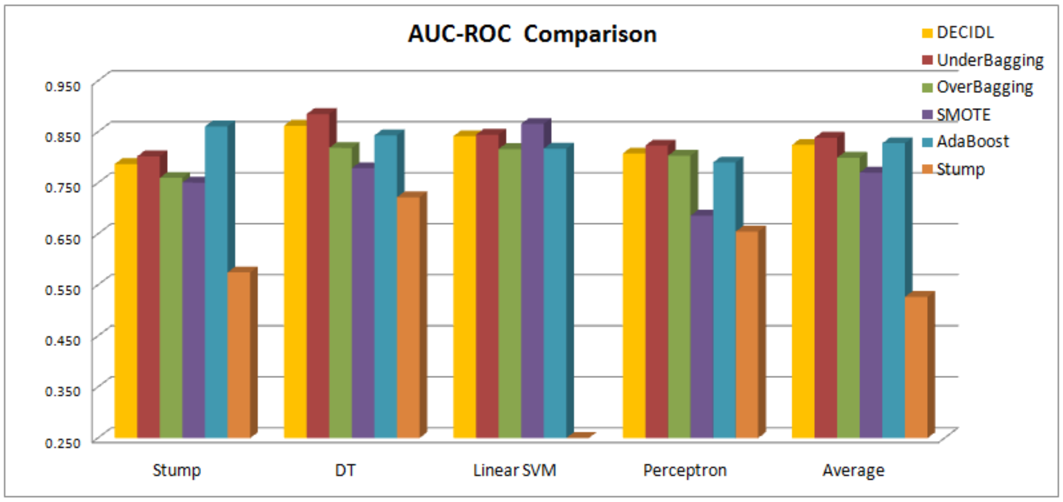


Figure 4.6: AUC-ROC Comparison of Average Performance on 22 Algorithms on 30 Data Sets

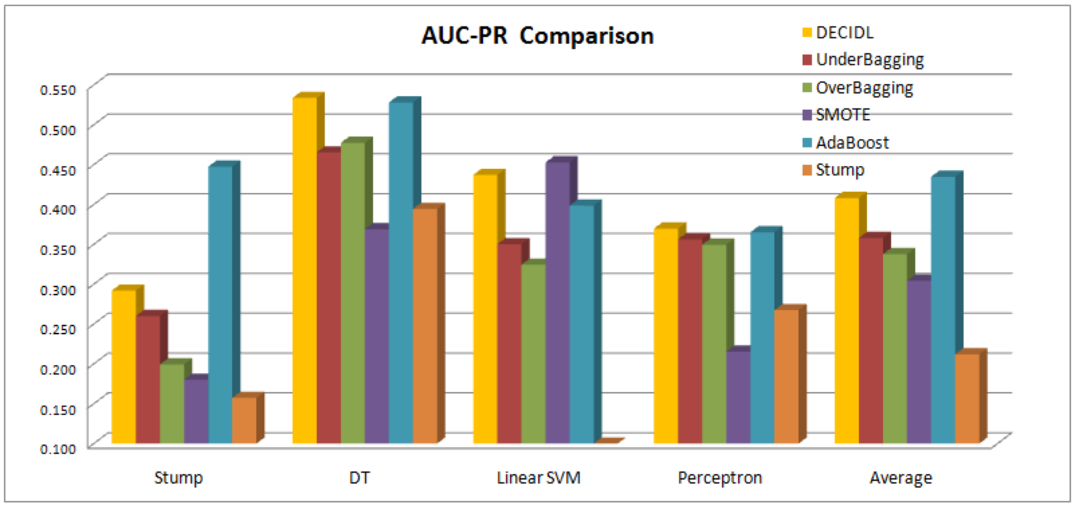


Figure 4.7: AUC-PR Comparison of Average Performance on 22 Algorithms on 30 Data Sets

From Figure 4.3-4.7, a close observation on the averaged results of different base learners shows that the DEICDL ensemble produce more stable and comparable performance over the five evaluation metrics. More specifically, the DECIDL ranks the first in the averaged F-measure comparison in Figure 4.3; it is the second best among six in averaged G-means comparison in Figure 4.4; it also stays in the second positions when comparing the averaged MCC and AUC-PR performance in Figure 4.5 and 4.7; finally, the DECIDL only drop to the third positions when coming to the mean AUC-ROC values based on Figure 4.6. In total, the average rankings of DECIDL will still be in the leading position among the five ensemble strategies, as will be shown later.

Moreover, we notice that the five evaluation metrics are not consistent in measuring classification performance even on the same algorithm. This suggests that each individual metric has its own strength and weakness, as also can be explained directly from their definitions. For example, the F-Measure emphasizes the rate of true positive predictions and the accuracy of positive predictions; while the G-means combines the rate of true positive and true negative predictions. Therefore, choosing an appropriate evaluation metric depends on the practical imbalanced learning task, data set and intention.

If the precision and recall are the main concerns of learning results, then the F-measure or AUC-PR might be better choices; if the accuracies on positive and negative classes are both important, then G-means and AUC-ROC are top considerations. Besides, if no specific desire on the accuracy of positive or negative class, then MCC will give a generally balanced classification result on the overall performance. Based on these results, the DECIDL seems to be slightly better and more stable from an overall view. A deep investigation is needed to compare these five ensemble methods. Therefore, to further compare the performance of five ensemble strategies, we rank these metric values within the same group of metric and base learner, i.e., from 1 to 6 for Decision Stump and related ensemble methods with the F-measure metric. Then, the average ranks of different ensemble strategies over the four base learners are summarized separately with respect to the 5 evaluation metrics, as shown in Table 4.3. Clearly, we found that on average our proposed DECIDL method ranks the first among the 5 ensemble systems. This result has strongly proven that our artificial example generation and the oppositional re-labeling strategy are quite effective on the imbalanced data learning tasks. Meanwhile, this also suggests that the diversified ensemble construction with reversely re-labeling does improve the imbalanced classification performance dramatically for ensemble learning. Interestingly, the second and third best ensemble strategies are the under-bagging and over-bagging. The possible reason for such result might be that both under-bagging and over-bagging are directly manipulating the class distribution of the data. However, the under-bagging performance is more fluctuated across the five evaluation metrics than that of over-bagging; this is also explainable since under-bagging randomly drops examples in different bags while the over-bagging always keeps all examples, except duplicating certain examples. The AdaBoost is slightly lower than over-bagging ensemble in the average ranking; however, AdaBoost has consistent better performances over four evaluation metrics except the G-means. Astonishingly, the SMOTE method ranks at the last position, which means this ensemble strategy are not effective enough on highly imbalanced data, comparing to other methods. However, all these ensemble methods have substantially higher rankings than the individual learning algorithms.

Table 4.5: Overall Performance Lead and Ranking of 5 Ensemble Strategies over 4 Base Learners

Ranking	**F-measure**	**G-means**	**MCC**	**AUROC**	**AUC-PR**	**Average**
DECIDL	2	3	2.5	2.5	1.5	**2.3**
UnderBagging	4.5	1	4.75	1.5	3.5	**3.05**
OverBagging	2.5	3	2.5	4	4	**3.2**
SMOTE	3	3	2.5	4	4.5	**3.4**
AdaBoost	3.25	5	3	3	2	**3.25**
NONE	5.75	6	5.75	6	5.5	**5.8**

4.4.2 Influence of Size of Artificial Set to Performance

To evaluate the influence of size of artificial example set to the performance of our DECIDL ensemble classification, we set different size ratios for experiments. This ratio is a ratio value between the size of artificial set and the training set; for example, a default ratio 0.5 (50%) means that, if the size of training set is 100, 50 artificially synthesized instances will be generated and added back to the original training data set for building new ensemble members. The ratios are set with 0.1, 0.3, 0.5, 0.7 and 0.9. Other parameters are set with default values in our DECIDL. Table 4.10-4.11 and Figure 4.8-4.9 show the influence of size of artificial set with respect to the five imbalanced evaluation metrics, averaged from 4 base learners on the Ecoli and Sick-Euthyroid data. From these results, we notice that a small size or a large size for artificial set are both not the best, ratios between 0.3 and 0.7 seem to be a better range for all the five evaluation metrics. This underlying reason might be that a small artificial set is not enough to re-balance the data distribution so that learner will focus more on the minority class; on the other hand, a large artificial set may also hamper improving the performance due to two reasons: first, learning on large artificial set may be hard to generalize the knowledge to the original set, thus the performance will not be improved in the later iterations of ensemble building; second, too large extra set may also force the learner to be over-tuned for the extra knowledge, instead of the original data set. Therefore, a moderate size of artificial set may be the best choice for our DECIDL ensemble framework.

Table 4.6: Influence of Size of Artificial Set to Classification Performance on Ecoli Data

Ecoli Data	Size Ratio				
	0.1	0.3	0.5	0.7	0.9
F-measure	0.429	0.504	0.481	0.456	0.443
G-Means	0.599	0.671	0.648	0.624	0.611
MCC	0.412	0.466	0.440	0.417	0.402
AUC-ROC	0.774	0.791	0.797	0.776	0.766
AUC-PR	0.415	0.419	0.418	0.380	0.355

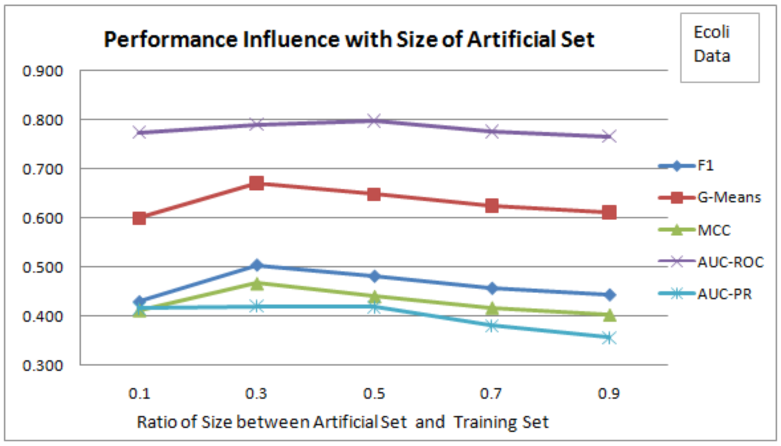


Figure 4.8: Plot for Influence of Size of Artificial Set to Classification Performance on Ecoli Data

Table 4.7: Influence of Size of Artificial Set to Performance on Sick-Euthyroid Data

Sick_Euthyroid	Size Ratio				
	0.1	0.3	0.5	0.7	0.9
F-measure	0.430	0.462	0.468	0.493	0.451
G-Means	0.551	0.593	0.597	0.686	0.662
MCC	0.405	0.426	0.432	0.448	0.407
AUC-ROC	0.810	0.801	0.853	0.845	0.845
AUC-PR	0.475	0.470	0.516	0.497	0.485

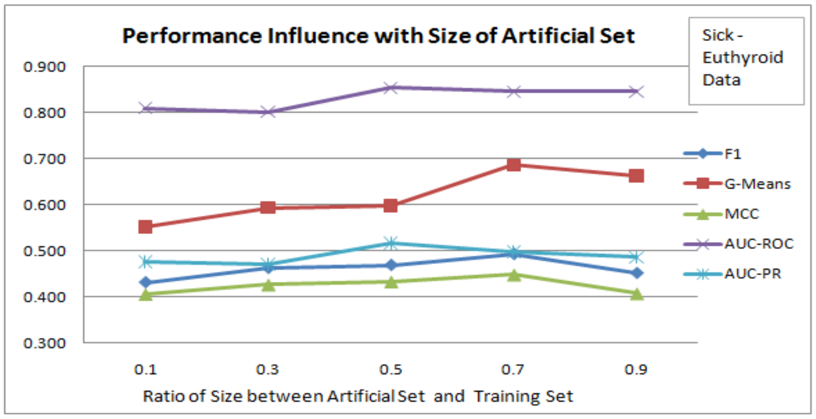


Figure 4.9: Influence of Size of Artificial Set to Performance on Sick-Euthyroid Data

4.4.3 Importance of Rejection Criterion to Performance

In our DECIDL framework, a new classifier member will be rejected if its internal performance on the training data set is not improved. However, this rejection criterion can be ignored to simply add new member no matter whether the performance improves or not. The effects of the two strategies are: rejecting new member will bring less diversity, but stronger classifiers; without rejection can introduce more diversified members, but may also allow low accurate classifiers to join the committee. Therefore, both strategies have their own advantages and disadvantages. Here, we conduct experiments with the two choices in our DECIDL framework on two data sets. Table 4.12 and Figure 4.10 show their averaged performance on the 4 base learners. From these results, we notice the two strategies almost have relatively similar performance on the two data sets. However, the AUC-ROC and AUC-PR are always higher for no rejection situation. This means that including more diversified classifiers do have positive influence for the probability prediction of ensemble. On the other hand, F-Measure, G-means and MCC performances fluctuated from one data to another. This further suggests that building and evaluating ensemble classifiers on imbalanced data are very complicated.

Table 4.8: Effect of Rejection Strategy to Performance on Ecoli and Sick-Euthyroid Data

Data	Strategy	F-Measure	G-means	MCC	AUC-ROC	AUC-PR
Ecoli	No-Reject	0.480	0.705	0.435	0.879	0.537
Ecoli	Reject	0.487	0.640	0.451	0.784	0.399
Sick-Euthyroid	No-Reject	0.476	0.608	0.439	0.831	0.491
Sick-Euthyroid	Reject	0.542	0.652	0.509	0.780	0.455

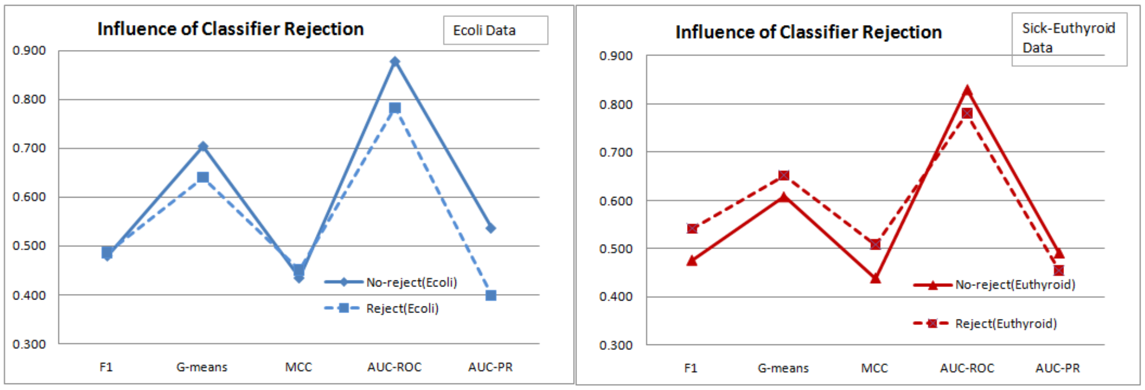


Figure 4.10: Effect of Rejection Strategy to Performance on Ecoli and Sick-Euthyroid Data

4.4.4 Influence of Ensemble Size to Performance

The size of ensemble committee is also believed to be well associated with performance of ensemble learning. Therefore, we test our DECIDL method under different sizes of ensemble. Since a rejection step may change the real size of final ensemble, we use the no-reject option to make sure that the original setting of ensemble size is the final size of ensemble prediction. A range of ensemble size is chosen between 3 and 100, which is very reasonable in most current ensemble studies. Other options are still kept as the default values of DECIDL framework when conducting this experiment. The 4 base learners are again used separately and their averaged evaluation performance are reported. Table 4.13 and Figure 4.11 show the actual average results with above settings on the Ecoli and Sick-Euthyroid data. Clearly, these results show that the five evaluation metrics are very stable one the ensemble size becomes relatively sufficient, more than 10. However, if the size is under 10, the performance could fluctuate significantly. This effect has been broadly found in existing ensemble learning methods. Moreover, the result means that the default setting for ensemble size (as 15) in our DECIDL method is enough.

Table 4.9: Effect of Ensemble Sizes to Classification Performance on Ecoli Data

Ecoli Data					
EM Size	F-measure	G-means	MCC	AUC-ROC	AUC-PR
3	0.544	0.770	0.518	0.890	0.558
6	0.497	0.829	0.480	0.917	0.616
10	0.538	0.847	0.521	0.930	0.638
20	0.559	0.852	0.541	0.933	0.655
30	0.563	0.853	0.546	0.935	0.658
40	0.566	0.853	0.550	0.934	0.660
50	0.566	0.846	0.549	0.936	0.654
60	0.558	0.847	0.541	0.938	0.663
70	0.558	0.844	0.542	0.936	0.653
80	0.569	0.851	0.552	0.937	0.662

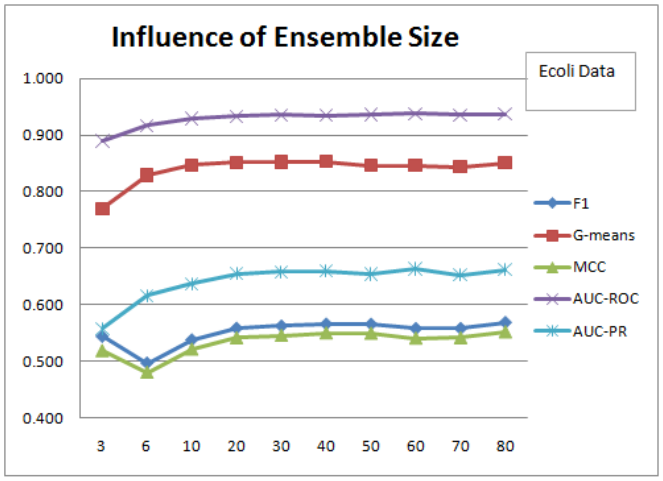


Figure 4.11: Plot for Effect of Ensemble Sizes to Classification Performance on Ecoli Data

Table 4.10: Effect of Ensemble Sizes to Classification Performance on Ecoli Data

Sick-Euthyroid					
EM Size	F-measure	G-means	MCC	AUC-ROC	AUC-PR
3	0.656	0.914	0.645	0.919	0.491
6	0.626	0.910	0.618	0.924	0.542
10	0.623	0.912	0.617	0.922	0.510
20	0.617	0.913	0.613	0.929	0.514
30	0.629	0.906	0.619	0.917	0.520
40	0.623	0.913	0.617	0.927	0.517
50	0.625	0.909	0.616	0.925	0.527
60	0.629	0.912	0.621	0.922	0.522
70	0.625	0.909	0.616	0.920	0.526
80	0.632	0.908	0.622	0.919	0.512

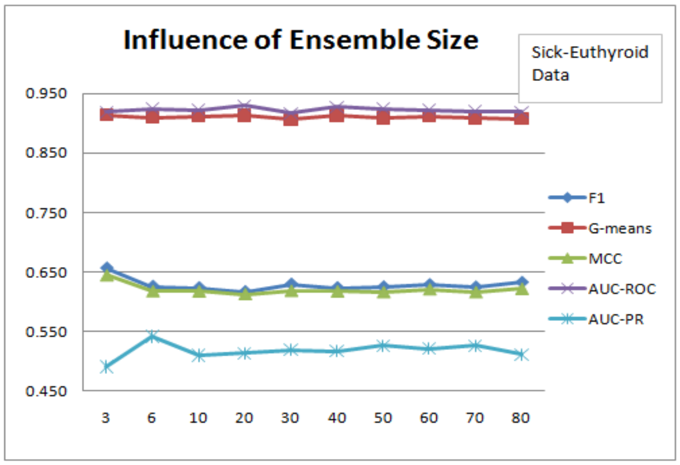


Figure 4.12: Effect of Ensemble Sizes to Classification Performance on Sick-Euthyroid Data

4.5 Discussion

In this chapter, extensive experiments have been conducted to 1) compare imbalanced classification performance between various ensemble strategies, 2) verify the effectiveness of our proposed DECIDL ensemble framework under different internal conditions.

Based on the above results and discussions, we can draw the following general conclusions and summaries:

--There is no single best ensemble imbalanced learning algorithm for variety of data domains, in terms of classification performance. Although the average rank of our DECIDL ensemble is on the top among the five ensemble frameworks, for each individual imbalanced learning data set, every algorithm shows very different classification performance.

--The five imbalanced evaluation metrics, namely, F-measure, G-means, MCC, AUC-ROC, AUC-PR, are not consistent in evaluating the performance of a classifier. Thus, it is hard to say which evaluation metric is the best in real applications. Choosing a suitable metric for a particular imbalanced learning task may depend on the target of interest when deploying such prediction system.

--The proposed DECIDL framework has shown relatively superior ability in learning on imbalanced data, compared with other four ensemble systems; however, several steps of parameter settings are still required to be investigated before applying to real-world problems, in order to fully discover the performance of DECIDL learning.

Chapter 5: MORE EFFECTIVE ARTIFICIAL EXAMPLE GENERATION FOR DATA BALANCING

In this chapter, we focus on a particular step in our DECIDL framework—the artificial example generation process, as this is an interesting research topic and several practical strategies have been proposed. We develop another new effective data synthesizing strategy, called distribution-based example generation (DBEG), in order to improve our DECIDL performance. To measure the effectiveness of the DBEG method and the DBEG embedded DECIDL method, we compare them with two popular and start-of-art imbalanced learning strategies—SMOTE [CHB+02] and GSVM-RU [TZC+09], and show their detailed learning performances on our benchmark data pool.

5.1 Introduction

We have reported extensive classification results from experiments performed on a large benchmark data pool with several popular ensemble imbalanced learning strategies in previous chapters. A brief conclusion on these results is that DECIDL framework is comparable to other ensemble learning methods on highly imbalanced data. Meanwhile, as there are several strategy-choosing and parameter-tuning steps involved in a concrete DECIDL procedure, it provides lots of flexibility and potentials to learn on various kinds of imbalanced data from different domains.

Hence, in this chapter, we want to examine the potentials of DECIDL framework by specially focusing on the artificial example generation step. The default data synthesis strategy is to create new examples among two existing data points, where one of them is from negative class and the other is from positive class. Detailed description can be found in Section 3.2. The idea behind this strategy is that the examples in the intersection of positive and negative data spaces are more interesting, as discriminating data objects there could help separating two classes in space.

Here, we consider this data generation problem from a different angle. After studying several existing synthesis methods in literature, such as SMOTE [CHB+02], Borderline-SMOTE [HWM05], we also propose a simple and more effective method to generate artificial examples—called the distribution-based example generation (DBEG). To test this strategy in real learning tasks, we develop a new ensemble learning algorithm based on this data synthesis strategy—the distribution-based example generation for ensemble learning (DBEG-Ensemble). We also incorporate this strategy inside our DECIDL framework—the DECIDL-DBEG. We then apply these two methods on the benchmark data pool to verify their performance.

More importantly, the two newly developed methods are compared with two popular and start-of-art imbalanced learning methods, the SMOTE-Bagging, and GSVM-RU. GSVM-RU is the most effective classification algorithms that even proposed in literature for highly imbalanced data [TZC+09]. Comprehensive experimental results will be listed to compare their performance together, and conclusions will be drawn to describe the effectiveness of DBEG in imbalanced learning tasks.

5.2 Related works on artificial example synthesis

As described in section 2.3.1.3, the first and one of most popular example synthesizing strategies is the SMOTE method, which inspires the creation of DECIDL framework in this dissertation. SMOTE creates new minority examples by interpolating new data points between an existing positive data point x_i and one of its Euclidian positive-class neighbors. A simple figure to show the result of SMTOE data synthesis and DECIDL synthesis is depicted in Figure 5.1. Here, we continue to introduce several other existing example generation strategies. One of the drawbacks of SMOTE is that it creates same number of synthetic examples for every original minority example, hence increasing the occurrence of overlapping between classes in dense data spaces. To remove such drawbacks, some improved methods are proposed based on SMOTE, such as ADASYN (Adaptive Synthetic Sample) [HBG+08], Borderline-SMOTE [HWM05] and DataBoost-IM [GV04].

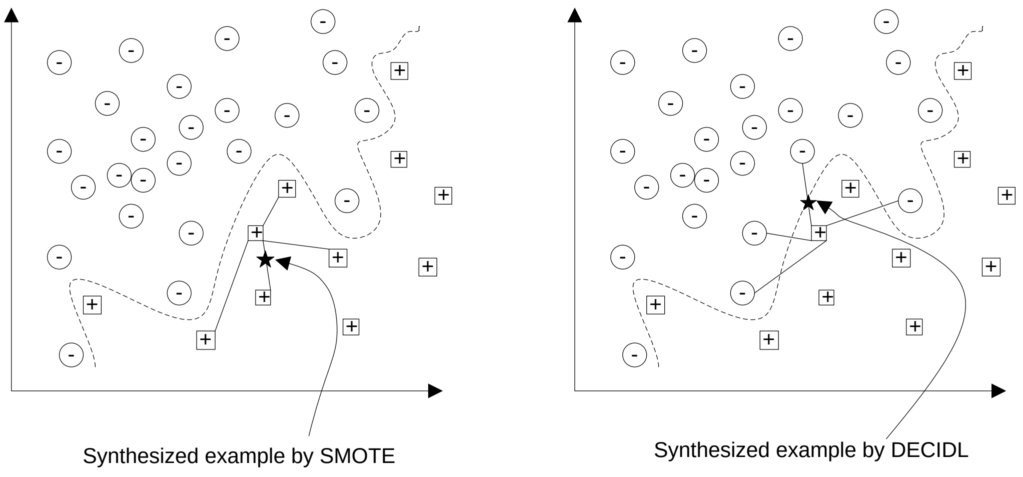


Figure 5.1: New example generation based on SMOTE and DECIDL

ADASYN [HBG+08]: the idea of ADASYN is to adaptively decide the number of synthetic examples to be generated for each minority example by using a density distribution function. The density distribution Γ_i measures the percentage of majority-class nearest neighbors for each minority example $x_i \in D_{\min}$, given a pre-defined neighbor number K:

$$\Gamma_{x_i} = \frac{\Delta_i / K}{Z}, x_i \in D_{min} \tag{5.1}$$

where $\Delta_i$is the number of majority-class neighbors of x_i within its K nearest neighbors, and Z is normalization term such that Γ_{x_i} is a distribution function ($\sum \Gamma_{x_i} = 1$). Then the exact number of synthesized examples for each x_i is:

$$N'_{x_i} = \Gamma_{x_i} * (N^- - N^+) * \beta \tag{5.2}$$

where β is a parameter to specify the desired balance level after the synthesizing process. Obviously, each x_i will not have equal number of artificial examples N'_{x_i}, as it depends on their neighbored example distribution. The more surrounded by majority-class data a minority example is, the more synthetic examples it needs to be generated. Oppositely, if a minority example is mixed within the same-class neighboring points, no synthetic examples are needed for itself.

Borderline-SMOTE [HWM05]: Borderline-SMOTE selects minority examples that have more majority-class neighbors than minority-class neighbors as seed points to synthesize new examples. To be more specific, let x_i is an original example from minority class D_{min}, and first determine the set of nearest K neighbors $D_{x_i}^{nb}$for each x_i, and then choose those $x_i$has more than half neighbors in the majority class, which means that $x_i$satisfy:

$$K/2 \leq \left|D_{x_i}^{nb} \cap D_{maj}\right| < K, \tag{5.3}$$

to create new artificial data points with the standard SMOTE procedure, i.e., interpolating between two examples. And the set of these minority examples x_i is called "Danger" set, as they represent the borderline minority examples which are highly likely to be misclassified later. Hence, Borderline-SMOTE is able to create new artificial minority examples in the neighboring areas of border. Notice that if the neighbors of a minority example x_i all belong to majority class, such x_i will not be chosen as seed, as Borderline-SMOTE consider it as noisy minority data.

In summary, Borderline-SMOTE only chooses those minority examples "closer" to the border of two classes to generate synthetic instances.

DataBoost-IM [GV04]: this method also introduces a simple data synthesis strategy, in combination with algorithm (AdaBoost.M1) to achieve high prediction accuracy on imbalanced data. First, DataBoost-IM uses the weighted distributions in boosting procedure to represent the difficulty of learning for each example x_i in the whole data set D. Rank all x_i in decreasing order the current classifier C_j and only top $\mathrm{N} * error(C_j)$ examples (set E) are selected as seeds for data synthesis. Set E consist of two subsets: E_{min} and E_{maj}, representing examples in minority and majority classes, respectively. Then the augmenting ratio of new synthesized examples is related to the size of these two subsets. More concretely, the ratio for generating new majority examples is $M_{maj} = \min(\frac{N^+}{N^-}, |E_{maj}|)$, and the ratio for new minority examples is $M_{min} = min(M_{maj} * \frac{N^+}{N^-}, |E_{min}|)$.Next, the numbers of synthesized majority and minority examples are $|E_{smin}| = M_{min} * N^{+}$ and $|E_{smaj}| = M_{maj} * N^{-}$, respectively, where $E_{smin}$and E_{smaj} represent two synthesized example sets.

All the above computing steps are only used to determine how many artificial examples need to be generated. Finally, DataBoost-IM uses the value distributions of features to actually create new examples for two classes. For nominal feature, a new synthetic value is chosen to reflect the value distribution contained in the original training data set, with respect to the particular class. In other words, the occurrences of different feature values are used as weights to synthesize a value. For continuous feature, the range and standard deviation of feature value in the original training data set is collected, and then new synthesized feature values will have same mean values and standard deviations. Obviously, the Gaussian distribution is assumed for such features.

5.3 Distribution-based artificial example generation (DBEG)

From the summary on artificial example synthesis in previous section, we notice that the existing synthesis strategies only contain two kinds: either using interpolating between two examples or assuming even or normal distributions on feature values to create new data examples. Besides, most methods require lots of computations to focus on choosing appropriate minority examples as seeds, instead of designing more effective methods to create the actual values for new synthesized examples.

Due to the drawbacks of above methods, here we propose a simple and fast distribution-based example generation strategy. First, we assume that each example in the original training data set is i.i.d. distributed, which is a standard assumption for data in many algorithm design. The occurrences of feature values are naturally representing the underlying distribution of a particular feature for a specific class. Obviously, such distributions are not necessarily either uniform, or Gaussian, and may be very complicated. Hence, making further assumptions on them are not appropriate to create synthesized values for new examples. In other words, the originality of the distribution of those feature values should be preserved, instead of assumed and simulated, within the new synthesized example set. As only minority class need extra synthesized examples, so only the feature distributions in minority class are investigated.

Naturally, the distribution of feature value occurrences can be used as guidance to synthesize new examples. To maintain the same distribution of every feature among artificial data set, we can randomly draw new feature values based on such distribution. High frequent values should have high chances of to be drawn, and vice versa. To make the computational process simple, we also assume that features are independent to each other, and hence a sequential process can be used to generate one feature after another. This assumption is also very standard during many machine learning algorithms.

Obviously, collecting the occurrences of values in nominal features is easy and making more sense. But for continuous features, purely counting these numerical values to synthesize new examples could make the combined data set so specific and hence leads to poor generalization ability. We use an alternative way to represent the distribution of numeric values and introduce extra variances at the same time. This idea is to duplicate all the values of this feature in minority class for T times and add additional Gaussian noise on them. Then, whenever a new feature value is needed to synthesize new example, simply equally pick one from them. Through this way, the distributions of numeric feature values are still maintained and certain degree of variations are included.

We propose the following steps to create the feature values $\{x_i^1, x_i^2, \ldots, x_i^k\}$ of a new artificial example x_i to be added to minority class:

1) If feature F^j is nominal, then collect all the unique values of this feature in the minority training set D^+ and their percentages of occurrences. The jth feature of x_i^j is then randomly chosen to reflect these percentages. For example, let assume in a rare disease data set, there are 55 patients including 35 males and 20 females and other 769 healthy people. Obviously the gender feature is one interesting attribute and has two distinct values: “Male” or

"Female". Then, a new gender value is randomly drawn with 35/55=63.6% of being male and 20/55=36.4% of being female.

2) If feature F^j is continuous, and assume f_{min}^j is the feature vector of the minority training set D^+, then duplicate this vector for T times to get $Dup_T(f_{min}^j) = \{f_{min}^j; f_{min}^j; \dots; f_{min}^j\}$. Next add additional Gaussian distributed noise value v_i with standard deviation δ_i to each feature value in $Dup_T(f_{min}^j)$. In need of synthesizing this feature for a new example, just uniformly choose on value in this feature vector $Dup_T(f_{min}^j)$.

In fact, the method of synthesizing new examples for continuous features is also applicable for nominal features, if ignoring the noise addition part; hence in the real implementation, we simply use the step 2 for all features, but do not add noises for nominal features and their values.

One may argue that the above strategy (Step 2) is not identical to creating new values based on feature distribution directly, but simply an approximate method: choosing new values from a distribution-based feature value pool without replacement. However, this procedure is simple and requires much lower computation cost and space. Besides, this approximate method can also provide good artificial data and enhances classification performance. Figure 5.2 shows the original data distribution of a particular numeric feature, distribution of SMOTE-synthesized data, and distribution of DBEG-synthesized data. Clearly, we can see that DBEG is more accurate to draw similar distributions of original data set.

How to choose the duplicate factor T is an interesting problem. Obviously, the larger it is, the more similar this approximate procedure is to pure distribution-based creation. However, a large T also requires more space and computational burden. In our experiments, we use a range for T: $T \in [20, 40]$.

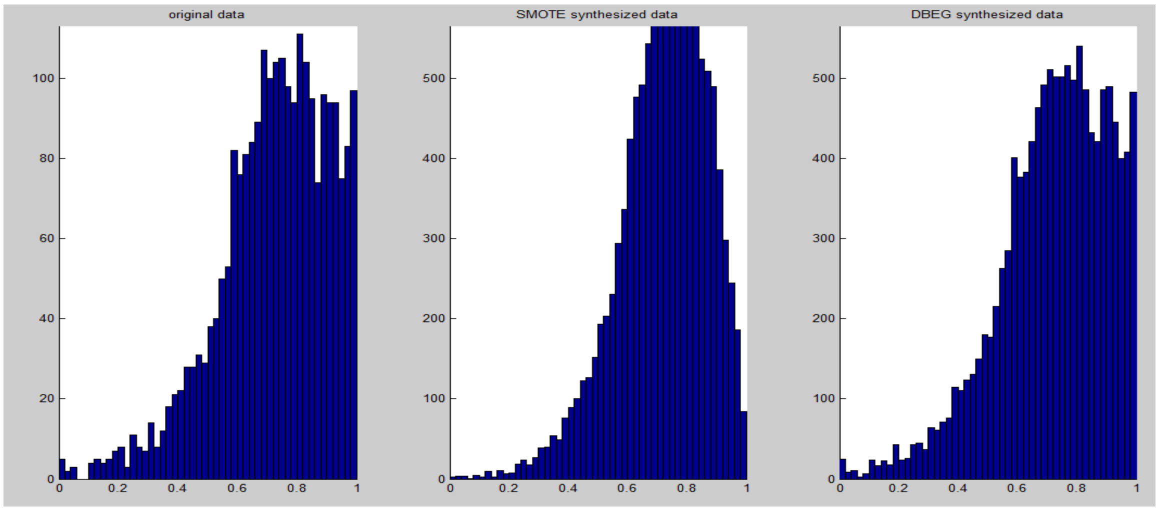


Figure 5.2: Distribution comparison among original data, SMOTE-synthesized data, and DBEG-Synthesized Data

Table 5.1: Distribution-based example generation (DBEG)

Algorithm 5.1: The DBEG Algorithm

Input:

```
    $D - \langle X, Y \rangle$, training set with $D^+$ is the minority class and $D^-$ is the majority class;
    $T$, duplication factor, default 40;
    $\beta_{syn}$ , desired balance ratio after synthesis;
    $\delta$, standard deviation for noise level, default 0.1.
```

Steps:

```
1. Let $N^+ = |D^+|$, $N^+ = |D^-|$, and the number of examples to be synthesized is:
            $N_{syn}^+ = N^- * \beta_{syn} - N^+$
2. Initialize an empty matrix $D_{syn}^+$ with size $N_{syn}^+ * n$, where
3. For feature $i = 1{:}n$
4.     Duplicate the $ith$ vector$f_{min}^j$ of $D^+$for $T$ times to get $Dup_T(f_{min}^j)$
5.     If $f_{min}^j$is a continuous feature:
6.         Add additional Gaussian noises (0, $\delta$) to $Dup_T(f_{min}^j)$
7.     Uniformly pick $N_{syn}^+$ values from $Dup_T(f_{min}^j)$ without replacement
8.     Put the vector of selected values into $ith$ column of $D_{syn}^+$
```

Output:

```
    $D_{syn}^+$
```

In order to determine the total number of synthesized examples, we again introduce a balance ratio β_{syn} as the stopping criterion for synthesis process. β_{syn} will be the targeted ratio between the size of minority class and majority class, hence it is same as Pct parameter in our DECIDL framework. The total number of minority examples is:

$$N_{syn}^{+} = N^{-} * \beta_{syn} - N^{+}. \tag{5.4}$$

In summary, the distribution-based example generation procedure is proposed in Table 5.1.

5.4 DBEG-Ensemble and DECIDL-DBEG

In this section, we will utilize the DBEG procedure to develop new imbalance learning algorithms. As a data generation step, DBEG has to be used with general learning methods together to show it effectiveness. Furthermore, DBEG involves a stochastic procedure to create new values for artificial examples, thus the resulting synthesized minority set might be unstable. Hence, an ensemble step is needed to a classification committee on several data sets after several rounds of using DBEG data balancing algorithm. Under the ensemble strategy, a fundamental and general machine learning algorithm, or called base learner, will be used to build each concrete classifier. To be more specific, we will use support vector machines (SVMs) as the base learner, instead of using several different ones as in previous chapters. Hence, we can specifically concentrate on the data generation steps.

Our first DBEG based ensemble algorithm for imbalanced learning, called DBEG-Ensemble, is proposed as follows. Given an imbalanced data set D with binary classes and an ensemble committee size mem, we will employ the DBEG procedure to create an synthesized data D_{syn} and union it to D to have a balanced training data $D_{bal} = D \cup D_{syn}$ Then a regular base learner is used to build a classifier C_i on it. Iteratively execute this procedure mem times to have mem classifiers as a committee. Then the final ensemble classifier C^* is achieved by using majority voting on this committee. To be more precise, the following pseudo code describes the DBEG-Ensemble methods.

Table 5.2: The DBEG-Ensemble algorithm

Algorithm 5.2: The DBEG-Ensemble algorithm

```
Inputs:
    $Learner$    – Base learning algorithm (e.g., C4.5, SVM, NN)
    $D$          –$\langle X, Y \rangle$, Training set, $m$ examples$(x_i, y_i)$, $x_i$ is n-dim vector, and $y_i = \{1,2,\ldots,nc\}$
    $max\_mem$   – Maximum number of member classifiers in the targeted ensemble
    $\beta_{syn}$ – Desired balance ratio after synthesis
Steps:
    Initialization:
    1. $i = 1$
    2. While $i < max\_mem$
    3.     Apply DBEG algorithm(5.1) to generate set $D_{syn}$ with $D^+$ and $\beta_{syn}$ ; use default $T$ and $\delta$
    4.     Append the synthesized data on original data to get $D_{bal} = D \cup D_{syn}$
    5.     Use $Learner$ on the overall training data $D_{bal}$ to create classifier $C_i$
    6.     $i = i + 1$
Outputs:
        Ensemble classifier $C^* = majority\_vote\{C_1, C_2, \ldots, C_{mem}\}$
```

Similarly, we can embed the DBEG procedure into our proposed DECIDL method to have another version of DECIDL, the DECIDL-DBEG. However, in the DECIDL framework, the data synthesis is not for a particular class, i.e., either for minority class or majority class. It requires creating extra data from overall data sets and then using classifiers to predict their labels, and then further refining them with performance improvement if they are included. Therefore, the DBEG for DECIDL framework will use the whole training data set as input data for feature distribution collection, instead of only the minority class examples. Meanwhile, the number of synthesizing examples is pre-determined in the parameter of DECIDL, Pct. Other than this, the similar procedure is applied into DECIDL. The detailed pseudo code procedure is described in Table 5.3.

As two imbalanced learning algorithms have been properly developed to use the DBEG data generation step, we now proceed to introduce other two start-of-art and popular imbalanced learning algorithms in order to compare their performance later.

Table 5.3: The DECIDL-DBEG algorithm

```
Algorithm 5.3: The DECIDL-DBEG algorithm
Inputs:
    $Learner$ – Base learning algorithm (e.g., C4.5, SVM, NN, cost-C4.5, SVM-weight)
    $D$ –$\langle X, Y\rangle$, Training set, $m$ examples$(x_i, y_i)$, $x_i$ is n-dim vector, and $y_i = \{1,2,\dots,nc\}$
    $Cost$ – A $(nc * 1)$ vector, misclassification cost for each class (default:$Cost = I_{nc*1}$)
    $max\_mem$ – Maximum number of member classifiers in the targeted ensemble
    $max\_iter$ – Maximum number of iterations to build an ensemble
    $Eval$ – Evaluation metric for ensemble performance (e.g., Total Cost, or 1-MCC)
    $Pct$ – Percentage (ratio to size of training data D) of synthetic examples to create
Steps:
    Initialization:
    1.  $mem = 1, i = 1$
    2.  $C_i = Learner\ (D, Cost)$
    3.  Initialize ensemble, $C^* = \{C_i\}$
    4.  Computer ensemble performance: $E_{best} = Eval\ (C^*(X),\ Y,\ Cost)$
        Loop:
    5.  While $mem < max\_mem$ and $i < max\_iter$
    6.      Use DBEG algorithm to generate artificial data set $S$ :
                $S = DBEG(D, Pct * |D|)$, note that $\beta_{syn}$ is replaced with exact synthesizing number
    7.      Use current ensemble $C^*$ to make probability predictions on $S$, to get $P(S)$
    8.      Label examples in $S$ with probability of class labels inversely to $P(S)$
    9.      Remove the examples in $S$ labeled as majority class to get S'
    10.     $D' = D \cup S'$
    11.     $C' = Learner\ (D',\ Cost)$
    12.     $C^* = C^* \cup \{C'\}$
    13.     Computer new ensemble performance: $E' = Eval\ (C^*(X),\ Y,\ Cost)$
    14.     If $E' \le E_{best}$ (i.e., performance is better)
    15.         $mem = mem + 1,\ E_{best} = E'$
    16.     Else:
    17.         $C^* = C^* - \{C'\}$
    18.     $i = i + 1$
Outputs:
        Ensemble classifier $C^*$
```

5.5 Other methods for comparison

In this section, we will briefly re-introduce a start-of-art learning algorithm for highly imbalanced data, the GSVM-RU (Granular Support Vector Machine-Repetitive Under-sampling) [TZC+09], and another most popular imbalanced learning strategy, the SMOTE, because later we will compare the classification performance of these two methods with our DBEG-Ensemble method and the DECIDL-DBEG method on the benchmark data pool.

As simply described in Section 2.3.5, GSVM-RU is a kernel-based learning strategy [TZC+09]. Simply speaking, GSVM-RU examines the support vectors from trained SVM models and removes these negative support vectors recursively from the training data sets. Hence, the classification boundary will be moved towards the majority class, hence leaving more feature spaces for minority class, in order to easily catching more minority examples in future testing datasets. This repetitive model building and SV removing process will continue until the classification performance on a valid data set decreases. Finally all the removed negative vectors in conjunction with positive examples are used to train the final classifier. An alternative way is to use the model built in the final round of removing step as the best individual classifier. Tang et al. [TZC+09] have shown that GSVM-RU performed very well on many highly imbalanced data sets from different domains.

On the other hand, SMOTE is simply a data generating algorithm, providing over-sampling strategy to balance data. It can be followed with any general machine learning algorithms to build the final classifier. As described in section 2.3.1.3, SMOTE tries to create interpolating points between one minority example and one of its nearest same-class neighbors, in order to enlarge the minority class. Each minority example is chosen once and its nearest K positive neighbors are identified. Then, an interpolation is performed between this example and one randomly-picked neighbor; the coefficient of interpolation is a random number that is evenly distributed in range [0,1]. Obviously, SMOTE also involves stochastic values in its synthesis step, and the resulting data set and classification models will not be stable. A bagging ensemble procedure can be used to provide stability and remove variance. The SMOTE-Bagging method used in previous chapter is iteratively using SMOTE to create balanced training data sets and build several classifiers to form an ensemble committee for classification. As SMOTE is a very popular data synthesis algorithm, we select SMOTE-Bagging as one method to compare the performance with our proposed method: the DBEG-Ensemble and the DECIDL-DBEG.

All together, we will compare the following four imbalanced algorithms on the benchmark data pool: the DBEG-Ensemble, DECIDL-DBEG, GSVM-RU, and SMOTE-Bagging.

5.6 Experimental setup and results

Out of the above four methods, three of them are meta-learners, thus a base learner has to be provided to implement the real algorithms; the other one GSVM-RU uses SVM as classification learner. Therefore, we will also use SVM as base learner for other three methods, in order to produce more fair performance comparisons.

The 30 imbalanced data sets in the proposed benchmark data pool are again used to test their binary classification performance. As to the performance evaluation metrics, we will use two informative metrics: F-measure and MCC.

For the internal parameters used in each method, we will use the default or suggested values and perform certain amount of optimization to achieve high classification accuracy. For example, the balanced ratio after data synthesizing is about 50%; the ensemble size of classification committee is 15. Meanwhile, the kernel type and parameters of base SVM learner are set same for all four methods for each individual benchmark data set. Through this way, the resulting performances are fair enough to compare.

We use a 5-fold cross-validation on each data set to the performance of four methods. Meanwhile, 5 individual runs are performed to have an averaged final classification metric value. The detailed classification performances of the four methods on all 30 benchmark data sets are listed in Table 5.4.

Table 5.4: Performance Comparison among three meta- learning algorithms (based on optimized SVM) and GSVM-RU on benchmark data pool (averaged on 5 runs of 5-folds cross-validation)

	F-Measure				MCC			
Data Set	**DECIDL -DBEG**	**DBEG**	**SMOTE**	**GSVM**	**DECIDL -DBEG**	**DBEG**	**SMOTE**	**GSVM**
Ecoli	0.609	0.596	0.608	0.606	0.576	0.581	0.593	0.585
Optical_Digits	0.822	0.741	0.699	0.715	0.805	0.722	0.686	0.692
SatImage	0.358	0.409	0.424	0.400	0.358	0.411	0.423	0.390
Pen_Digits	0.775	0.819	0.816	0.769	0.763	0.807	0.805	0.764
Abalone_7	0.310	0.356	0.346	0.323	0.239	0.320	0.311	0.313
Sick_Euthyroid	0.751	0.647	0.667	0.543	0.727	0.634	0.651	0.538
Spectrometer	0.750	0.803	0.770	0.819	0.738	0.793	0.760	0.809
Balance	0.067	0.077	0.058	0.048	0.004	-0.009	-0.123	-0.027
Car_Eval_34	0.784	0.749	0.722	0.825	0.784	0.752	0.727	0.825
ISOLET	0.776	0.784	0.758	0.823	0.757	0.767	0.739	0.815
US_Crime	0.449	0.453	0.455	0.481	0.448	0.457	0.458	0.478
Yeast_ML8	0.160	0.027	0.028	0.158	0.091	0.041	0.024	0.090
Scene	0.210	0.249	0.251	0.230	0.138	0.192	0.191	0.199
Libras_Move	0.832	0.721	0.688	0.760	0.834	0.716	0.676	0.753
Thyroid_Sick	0.635	0.547	0.549	0.560	0.618	0.557	0.556	0.562
Coil_2000	0.117	0.069	0.064	0.151	0.029	0.048	0.046	0.112
Arrhythmia	0.206	0.234	0.284	0.439	0.170	0.191	0.243	0.418
Solar_Flare_M0	0.146	0.176	0.193	0.134	0.130	0.162	0.184	0.101
OIL	0.301	0.315	0.356	0.435	0.333	0.286	0.329	0.415
Car_Eval_4	0.936	0.979	0.969	0.822	0.934	0.979	0.968	0.826
Wine_Quality_4	0.269	0.073	0.077	0.127	0.242	0.023	0.040	0.126
Letter_Img	0.844	0.905	0.895	0.858	0.849	0.902	0.894	0.856
Yeast_UCI_ME2	0.137	0.312	0.281	0.226	0.156	0.352	0.324	0.272
Web_page	0.582	0.462	0.507	0.396	0.594	0.452	0.505	0.393
Ozone_Level	0.111	0.240	0.233	0.290	0.102	0.281	0.290	0.322
Mammography	0.530	0.368	0.392	0.303	0.554	0.420	0.437	0.395
Reuters-21578	0.647	0.758	0.804	0.701	0.655	0.763	0.806	0.704
Forest_Cover_5	0.285	0.165	0.252	0.284	0.278	0.193	0.249	0.277
Protein_homo	0.675	0.410	0.384	0.696	0.679	0.444	0.433	0.713
Abalone_19	0.082	0.039	0.053	0.036	0.151	0.087	0.118	0.054
Average	**0.472**	**0.449**	**0.453**	**0.465**	**0.458**	**0.444**	**0.445**	**0.459**
Win/Loss to GSVM-RU	**15**	**15**	**15**		**13**	**14**	**12**	

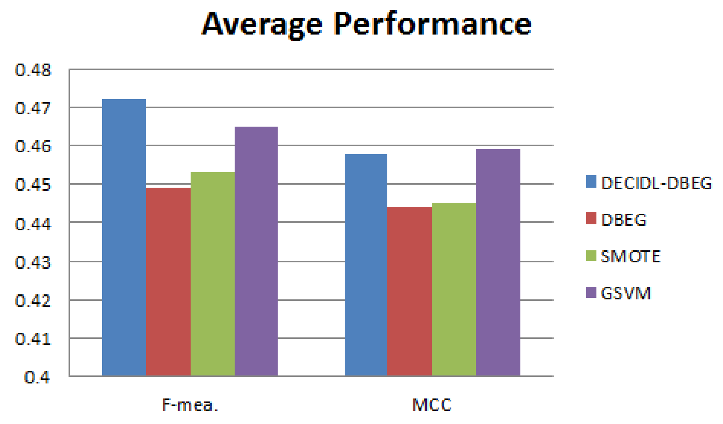


Figure 5.3: Average Performance Comparison of the four methods on benchmark data pool

5.7 Discussion and Conclusion

Table 5.4 shows the detailed F-measure and MCC performance of all four learning methods over each of the 30 benchmark data sets, and Figure 5.3 shows their averaged performance. First, in Table 5.4, it is noted that all four methods are relatively performing equally well on the whole benchmark data. On some data sets, the F-measure and MCC are quite high, e.g., Optical_Digits, Car_Eval_4, Libras_Move, Letter_Img, suggesting these classification problems are easy to solve and the four learning models are fitting the underlying class distributions. However, on some other data sets, e.g., Balance, Coil_2000, Yeast_ML8, Ozone_Level, the performance are quite low for all four learning methods. Such results suggest that the underlying models for distinguishing two classes are difficult to learn or to generalize; and all the four learning methods do not fit such models very well.

Table 5.4 also shows that our DECIDL-DBEG learning method has the highest averaged F-measure performance across the whole data pool, surpassing the SMOTE-Bagging and GSVM-RU methods. This result proves that using DBEG synthesis procedure does help our DECIDL framework achieve high classification performance. For the MCC performance, our DECIDL-DBEG method ranks the second and is close to the top one GSVM-RU method. Meanwhile, as GSVM-RU is currently the best developed method in literature, we can conclude that DECIDL framework is also one of most effective ensemble models for imbalanced learning in our study.

We have also noticed that both DECIDL-DBEG and GSVM-RU are relatively better than SMOTE-Bagging and DBEG-Ensemble, which suggests that basic ensemble strategies on balanced synthesized data are not superior to other refined ensemble methods.

The most interesting founding from the above result comparisons is that the average performance of our new developed DBEG-Ensemble is only slightly lower than that of all other sophisticated methods. This great discovery means that a simple effective data synthesizing strategy could also lead to high classification performance for imbalanced data. Using complicated learning strategies may not significantly enhance the learning power on a particular imbalance problem.

Hence, our conclusion on this example generation topic is that DBEG can be used as a fast and effective data balancing algorithm in the initial stage of solving an imbalance classification problem. To further study the potentials of DBEG synthesis, we propose the following directions in future study. First, as features are examined as independent to each other in sequentially generating feature values, we may consider certain interactions between two close-related attributes by calculating their correlations. A pair of values can be generated together if two features have high correlations. Second, as DBEG is able to generate new examples, we can apply the same process to output few examples by using the majority examples and a reversed balance ratio as input parameters (i.e., DBEG-Undersampling). In other words, under-sampling is used in each individual feature to generate new majority examples to replace original majority class. Last but not least, we can combine the original DBEG and DBEG-Undersampling together to balance data sets, and then use regular machine learning methods to test their classification performance.

Chapter 6: ACTIVE EXAMPLE SELECTION FOR DECIDL

In this chapter, we continue to solve the imbalance data learning problem, by investigating a new direction—active learning. Active learning is one subfield of supervised machine learning in which the learning algorithm can dynamically select interested unlabeled data examples and query their labels from domain experts or users, in order to incrementally build classification models. Active learning allows algorithm to choose what data it will learn from, and hence such algorithms will perform equally or better but with less training data. Active learning is originally trying to solve the problem that acquiring labeled data examples are costly or time-consuming in some contexts, such as speech recognition annotation, document categorization, etc. It has attracted special focuses from many researchers and become a particular research topic in recent years. Meanwhile, its related learning algorithms have also been developed and applied in several real-world applications, such as speech recognition, document classification, and mutant identification for protein functions, etc.

As active learning is able to choose appropriate data examples to learn from, we will use this advantage to select informative data examples to form a balanced training data set, and then build effective ensemble learning committee. This chapter describes the background of active learning and related works, and a new Active-DECIDL method is then proposed to combine the strength of active learning and DECIDL to tackle the problem of highly imbalanced data learning. Extensive experiments are conducted to examine the performance of this new Active-DECIDL method; comparative results are reported to draw a positive conclusion on this method.

6.1 Background and Related Works in Active Learning

Active learning is also referred to “query learning” or “optimal experimental design” in the statistics context [Set09]. The main focus of active learning in supervised or semi-supervised learning is its ability to select certain interested unlabeled data points for further learning. Supposedly there exists an oracle that is able to tag any data examples, such as a human annotator, or domain expert. Active learning will collect some interested unlabeled examples based on its selecting strategy and sent them to the oracle for querying their class labels. Once these data are labeled, the learning algorithm can build an improved prediction model and, at the same time, active learning will select another round of unlabeled data for labeling. This process is a typical pool-based earning cycle which is the main category of active learning [LG94]. The unlabeled example selecting and labeling step is a query process, and hence active learning is also called query learning.

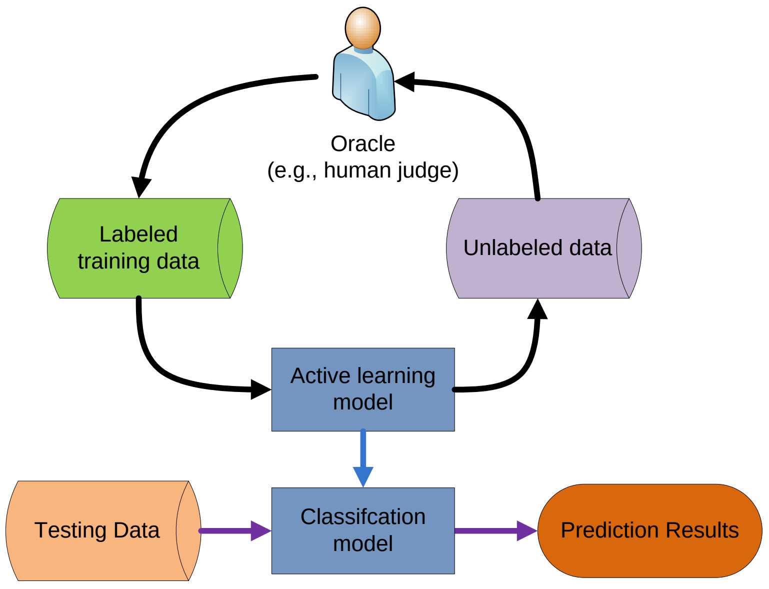


Figure 6.1: Pool-based Active Learning Cycle [Set09] and Classification Step

Obviously, active learning is quite different to traditional passive supervised learning. In traditional learning models, data examples are all having pre-determined class labels and no further relabeling process is involved during the training step. In other words, no further additional data information or knowledge is needed once the learning algorithm proceeds. In contrary, active learning requires continuous interactions between the learning procedure and the “oracle” for data selection and labeling. It can start from a small size of training data, and then it dynamically seeks promising and useful examples for continuously learning. Although its querying process needs interactive efforts from the oracle, the initial learning cost is cheap and prompt. This advantage is significantly important in many contexts where data acquisition or data labeling is prohibited or expensive. For example, in speech recognition problem, labeling record audio files at the word level takes 10 times longer than the actual length of audio; hence it is not likely to prepare large size of training data for audio categorization.

Not only does active learning have the benefits of requiring less training data in the initial stage of a learning process, but also it could improve the classification performance after choosing appropriate training examples to learn. For example, randomly selecting a subset of data examples is a popular way to build classification models if acquiring more examples is costly; however, the consequence of this strategy is that the resulting prediction model will be very unstable. On the other hand, using active learning to identify critical examples residing in the margin area of different classes will be much useful to build predicting models. This is another important reason that many researchers trying to achieve: using less or at most equal number of training examples to create better classification models.

Many researches on active learning have been done in recent years and lots of related works have been published in literature [Ton01] [EHB+07] [Ols08] [Set08]. To give a general overview of those works, we divide the process of active learning into two steps and discuss them in detail in the next two paragraphs.

The first step is how to find or create the unlabeled data examples. As active learning needs to query unlabeled data for annotating, the first task is to generate such data points. There are mainly three ways of identifying unlabeled data [Set09]: membership query synthesis, stream-based selective sampling, and pool-based sampling. Membership query synthesis assumes the learner can query the labels of any unlabeled instances in the input space; these instances may be artificially generated, rather than coming from some underlying natural distributions. The benefit of query synthesis is obvious: it can query any desired point in the input space; hence, it is very flexible. However, such arbitrarily synthesized examples may be not following the underlying data distribution, and are unreasonable for annotation in real applications. For example, Lang and Baum [LB92] use active learning for recognizing images of hand-written characters, but the queries generated by the learning algorithms are invalid symbols. The second way–stream-based selective sampling—considers the actual distributions of existing training data. This approach first performs sampling an unlabeled instance from the data distributions, and applies the learning algorithm to determine whether or not to query its label. As each unlabeled instance is sequentially sampled and queried or discarded, it is also called sequential active learning. This method is guaranteed that queries are reasonable, as they are sampled from the real underlying distribution. The strategy of whether or not query the chosen instance's label will be further discussed next. Several publications have used this approach to solve real-world problems [Kir02] [Yu05]. The third way of active learning is the pool-based active learning, as shown in Figure 7.2. The rationale of such approach is that in most scenarios, the unlabeled data are abundant while only labeled data are limited. Hence, the active learning strategy is trying to estimate the usefulness of every unlabeled instance based on the information on labeled set. The unlabeled data pool is static or non-changing (although this is not necessary), and queries can be chosen from such pool based on certain evaluating metrics. This approach is largely studied in literature as many real-world problems match this scenario; examples can be found in text classification [TK00], image classification and retrieval [TC01], video classification [YYH03], cancer diagnosis [Liu04].

The second step is how to judge and select the most appropriate or useful unlabeled examples for "oracle" labeling. In other words, what will be the ideal metrics to evaluate the usefulness of an unlabeled example, such that the performance of predicting model will be largely improved once this unlabeled example is annotated? Many frameworks have been proposed to develop an effective query selecting strategy and we will briefly name them a few here.

Uncertainty reduction. The simplest method of choosing an unlabeled instance for querying is to choose the most uncertain instance determined by current learning algorithm on some labeled data.

This approach is particularly convenient for probabilistic learning models, as their predictions can be directly used to compute uncertainty. Assume for an unlabeled data point x_i and its prediction probabilities by current learner L are $P_L(y_j|x_i)$, where y_j $(1 \leq j \leq c)$ represents total c different classes, then the chosen unlabeled instance is the one with least predicting confident: $x^* = \arg\min_x 1 - P_L(\hat{y}|x_i)$, where $\hat{y} = \mathrm{argmax_y} P_L(y_j|x_i)$, and x^* represents the chosen unlabeled example. Obviously, this strategy chooses the least confident instance with respect to its most probable class. Hence, it ignores the uncertainties of unlabeled instances on other remaining classes. To correct this bias, an improved selecting criterion is proposed to use the uncertain margins between the two most probable classes: $x^* = \arg\min_x P_L(\hat{y}_1|x_i - P_L(\hat{y}_2|x_i)$, where $\hat{y}_1$ and $\hat{y}_2$are the first and second most probable class labels predicted by the current learner. A more general selection for uncertainty reduction is to use the entropy as the uncertain measure [Set09]. The most uncertain data example is the same as the most informative example based on information theory. The selection criterion of entropy-based approach is $x^* = \arg\max_x - \sum_{1 \leq j \leq c} P_L(y_j|x_i) * \log P_L(y_j|x_i)$. In summary, the uncertainty reduction method tries to identify the most ambiguous unlabeled instances for annotation in order to reduce the uncertainty of training data for future learning.

Query by committee [SOS92]. This method involves using a committee of learning models to vote the labels of unlabeled data and identifies the one with most disagreement as the most useful querying example. The fundamental idea is to minimize the version space, which is the set of models or hypothesis that are consistent with current labeled training data [Set09]. The goal of active learning is to minimize the number of instances while maximize the precision of version space. Two issues need to be resolved to implement the QBC algorithm: a) how to build a committee of different models that represent different sub-spaces of the version space, and b) how to measure the degree of disagreement among committee members. This first issue can be solved with ensemble methods, such as boosting or bagging algorithms. Constructing learning models on subsets of training data or applying different learning algorithms and parameters on these data are the most popular ways to create classification committees. Notice that the DECORATE [Mel05] and our DECIDL are also ensemble-based committee forming methods with explicitly encouraging diversity among model members. The second issue is how to measure the degree of disagreement within the committee. Several effective metrics have been proposed in literature to address this issue, such as the vote entropy, Kullback-Leibler (KL) divergence, and Jensen-Shannon divergence, etc [Mel05]. The vote entropy uses information theory to evaluate the disagreement:

$$x_{VE}^{*} = \arg\max_{x} - \sum_{1\le j\le C} \frac{V(y_j|x_i)}{C} * \log \frac{V(y_j|x_i)}{C} \tag{6.1}$$

where $V(y_j|x_i)$ denotes the number of members which assign unlabeled instance x_i to label y_j and C is the total number of classes. Hence, this metric tries to find the instance with most disagreements, e.g., if each member disagrees with each other on x_i, then the vote entropy is maximum; on the contrary, it reaches 0 if every member reach a consensus. KL divergence is also an information-theoretic metric which measures the difference between two probability distributions. On the other hand, JS divergence is an extension of KL divergence, working on multiple distributions. JL divergence is the averaged KL divergence of each distribution with respect to the mean distribution of the whole set.

Expected error reduction. Expected error reduction tries to find the unlabeled data which can be able to maximally reduce the future generalization error if its label is known. The procedure of this approach is like the following: first a loss function is defined to estimate the expected generalization error, e.g., 0/1-loss or log-loss; then select one instance x_i in the unlabeled data set and label it; next include this single labeled instance in the original labeled set D and build a new learning model L; test the classification performance of L on an separate validating data sets to have an error rate e_{xi}. Iteratively select every unlabeled instance and repeat the process to get all of the error rates. The best query is the unlabeled example which has the lowest error rate. Obviously, to get all the error rates, the label of every unlabeled example needs to be known during the computational procedure, although the final output of the result only requires the labels of those chosen unlabeled data. In theory, the error rate may not be the only metric to evaluate the improvement of classification performance; other generic measurement such as precision, recall, F-measure, or AUC-ROC can also be used here for particular interest. This approach has been successfully used with various machine learning models, including naive Bayes [RM01], nearest neighbor [LMR04], logistic regression [GG07], and support vector machines [EHB+07]. An obvious effect of expected error reduction is that its computational cost is very expensive, as it builds the same number of models as the number of unlabeled examples in order to select only one optimal instance. For some machine learning algorithms will have already required lots of computations (e.g., SVM), this approach is extremely inefficient. Hence, this framework and its similar extension—expected model change framework [SCR08]—only consider simple and fast learning models.

Other frameworks such as variance reduction, density-weighted QBC methods for query selection have also been studied in literature; interested readers may refer them to [Coh94] [Set08] [Set09].

6.1.1 Active Learning for imbalanced data

The earliest works on active learning to address imbalance data learning problems started only a few years ago, around 2007. Zhu and Hovy applied active learning for word sense disambiguation (WSD) with approaches to solve the class imbalance problem [ZH07]. They claimed they are the first work of studying active learning with data resampling (over- and under-sampling) to add imbalance issue on WSD. Their contributions in this paper are three-folds. The first developed an over-sampling strategy, called BootOS, to create extra minority examples. For a given minority instance x_i, a new bootstrap example x_{Bi} is the averaged summation of x_i and its all kth nearest neighbors on all dimensions, e.g., $x_{Bi} = \frac{1}{k+1}\sum_{l=0}^{k} x_{i,l}$. As each example's $kt\mathrm{h}$ nearest neighbors are unchanged, so each example can only create a single additional new instance. They developed an active-learning-with-resampling algorithm by embedding the aforementioned resampling strategy into a standard active learning framework. The resampling step is performed after new labeled examples are added to training data set. This algorithm is their second main work of the paper. Their last contribution introduces a combined stopping criterion to stop active learning procedure. Two conditions are used to form this stopping criterion: a) max-confidence, which limits the predicted uncertainty (entropy) of each selected unlabeled example to a very small number and b) min-error, which measures the accuracy of predicted labels comparing to the real labels annotated by the “oracle” on the selected unlabeled examples. Their experiments on 38 random chosen ambiguous nouns show that their developed methods are better than random sampling, uncertain sampling, under- and over-sampling based on accuracy and recall comparisons.

Another very earlier work on active learning especially for class imbalance problem is studied by Seyda et al. [EHB+07] in 2007. They propose an effective unlabeled example selection strategy to address the class imbalance in text categorization. They use SVM to create learning models and use the separating hyper-plane as the evaluation criterion to estimate the usefulness of unlabeled examples. More precisely, they believe that the data points within the margin area are less imbalanced than the entire data distributions; hence unlabeled instances that are closer to the SVM hyper-plane are more informative than others. The distance of a data point between each unlabeled example to the hyper-plane built on existing training data set is used as selection criterion. Although similar works have been done before this paper in [AKJ04], the main contribution here is the “59 trick” proposed by Seyda et al. [EHB+07]. They proved that “the active learner will pick one instance (with 95% probability) that is among the top 5% closest instances to the hyperplane, by randomly sampling only 59 instances regardless of the training set size”. In other words, their method do not need to check the entire unlabeled data set, but only a small pool of 59 examples, and it is guaranteed at 95% confidence that the closest example to the hyperplane in this pool will also be in the top 5% closet examples to that hyperplane for all unlabeled data set [EHB+07]. Without looking over the whole unlabeled set, their method yields to a very efficient learning system which can work on very large datasets, such text categorization. They use an online SVM and an early stopping criterion to perform experiments on several real-world data sets, including data sets from Reuters-21578 text categorization data, CiteSeer, and UCI. They show that their method ranked the first on the average PRBEP performance on 18 data sets, comparing to under-sampling, SMOTE, and simple cost-sensitive learning.

The most recent work of applying active learning for imbalanced data classification is done by Oh. et al. [LOZ09] [OLZ11]. Their initial study is to develop an active example selection (AES) method by using the expected error reduction framework [LOZ09]. They first formulize a measure of usefulness for data examples with a posterior likelihood and information theory analysis, and define the usefulness of an example to be the sum of squared errors between desired output and actual output of the trained classifier. The class imbalance is resolved through selecting procedure of useful examples [LOZ09]. An incremental naive Bayes classifier is used as the base learner for this iteratively AES learning, and their initial results have proved the effectiveness. However, as AES starts to build initial classifier from a portion of randomly sampled training data set, slight changes may lead to the instability of output model. They then propose an ensemble AES (EASE) method to improve the generalized classification performance[LOZ09]. More precisely, the original training data set are split into equal size of sub-components. For each component, an AES procedure is performed to create a learning model. As each resulting model is built from different portion of the original data, Oh et al. claim that this method are better and more efficient than traditional bagging [OLZ11]. The final decision of these models is made by a weighted voting policy: the weighted summation of prediction probabilities from all models and weight derived from each model's training performance. After using an incremental naive Bayes method as the base learner again, Oh et al. show that this new EASE method also perform very well on several less imbalanced biomedical data sets, with less training data used [OLZ11].

6.2 Active-DECIDL

In this section, we will incorporate the active learning strategy to the DECIDL framework in order to solve the class imbalance problem. As active learning can start with a small data set to build classifiers initially and select useful examples in the later iterations, this property is naturally beneficial for tackling imbalanced data set. Although the data sets in our study are highly imbalanced, the useful examples among them may be balanced or less imbalanced. Naturally, applying active learning to identify the useful examples is possible to create balanced training subsets. However, in our imbalanced classification problem, there are no unlabeled data instances as all the examples are labeled. One way of fixing this issue is to consider partial data as unlabeled, and retrieve their labels when necessary. Therefore, we have to slightly change the standard active learning procedure to adapt it for imbalanced data.

The adapted active learning procedure is as follows. First, perform under-sampling the majority class to get a subset D_{-}^{1} and add it to minority class to form a balanced initial training data set $D^1 = D_+ \cup D_{-}^{1}$, and then build an initial classifier with a chosen base learner $C_1(D^1)$. Next, apply an effectiveness measure ee on the rest of majority examples $D_{-}^{\Delta} = D_{-} - D_{-}^{1}$ and identify the most useful examples D_{-}^{*}. Then add these useful examples back to D^1 to have another round of data set $D^2 = D^1 \cup D_{-}^{*}$ and build a new classifier $C_2(D^2)$, and at the same time remove D_{-}^{*} from D_{-}^{Δ} . These steps repeat several iterations until certain conditions are reached. As all data examples are labeled, no "oracle" is needed for annotation. That is, there is no additional cost to query all the real labels for all the unselected majority examples. Hence, we can use the difference between the predicted and its real label as the criteria for evaluating the effectiveness ee of an instance. In other words, the majority examples that can be accurately predicted by current model are considered no useful, while those incorrectly predicted examples are much more useful. Note that as the initial under-sampling on the majority class may introduce certain instability for the final output, a bagging or boosting has to be involved to generalize the classification performance of the described procedure, meaning an ensemble adaptive active learning procedure (EAAL).

Clearly, this ensemble adaptive active learning procedure also encourages the diversity of classification models in two aspects. First, it incrementally includes new training examples to build different learning models. Second, the new examples are those most contradicted examples to current classifier, and hence the new classifier built on the combined data set will produce certain variations to previous classifiers.

To combine the strengths and advantages of above active learning strategy and our DECIDL framework, we now embed this EAAL procedure into our DECIDL framework to develop a new meta-learning strategy: Active-DECIDL. As EAAL uses under-sampling to shrink the large majority class and DE-CIDL creates artificial examples to enlarge the minority class, the two methods compensate each other to become a perfect combination for solving imbalance learning problem.

Note that the active learning steps are embedded in our DECIDL procedure, thus the stopping criterion is shared with DECIDL, when the committee members reach to a maximum pre-settings. Also note that in the above algorithm, each time we select top $IncNum = |D_+| + |S'^i| - |D_-^i|$ majority examples to form the active set $D_-^{\Delta i}$, instead of selecting one or two instances in a traditional way. The reason is that after adding this active set to the current training set, the total number of positive examples $|D_+| + |S'|$ is equal to total number of negative examples $\left|D_-^i\right| + IncNum$. Hence, in each iteration the total learning data set D' is balanced.

Table 6.1: The Active-DECIDL algorithm

```
Algorithm 6.1: The Active-DECIDL algorithm
Inputs:
    $Learner$   – Base learning algorithm (e.g., C4.5, SVM, NN, cost-C4.5, SVM-weight)
    $D$         –$\langle X, Y\rangle$, Training set, $m$ examples$(x_i, y_i)$, $x_i$ is n-dim vector, and $y_i = \{1,2,\dots,nc\}$
    $Cost$      – A $(nc * 1)$ vector, misclassification cost for each class (default:$Cost\ =\ I_{nc*1}$)
    $max\_mem$  – Maximum number of member classifiers in the targeted ensemble
    $max\_iter$ – Maximum number of iterations to build an ensemble
    $Eval$      – Evaluation metric for ensemble performance (e.g., Total Cost, or 1-MCC)
    $Pct$       – Percentage (ratio to size of training data D) of synthetic examples to create
Steps:
    Initialization:
    1.  $mem\ =\ 1, i\ =\ 1$
    2.  Under-sample the majority class $D_-$ to create a subset $D_-^i$, and remaining set $D_-^\Delta$
    3.  Build a classifier model $C_i\ =\ Learner\ (D', Cost)$, where $D' = D_+ \cup D_-^i$
    4.  Initialize ensemble, $C^*\ =\ \{C_i\}$
    5.  Computer ensemble performance: $E_{best}\ =\ Eval\ (C^*(X),\ Y,\ Cost)$
        Loop:
    6.  While $mem < max\_mem$ and $i\ <\ max\_iter$
    7.      Reset current training data set: $D'\ =\ D_+\ \cup\ D_-^{i-1}$
    8.      Artificially generate data set $S$ with $Pct * |D|$ new training examples
    9.      Use current ensemble $C^*$ to make predictions on $S^i$ , to get $Prob(S^i)$
    10.     Label examples in $S$ with probability of class labels inversely to $Prob(S)$
    11.     Remove the majority-labeled examples in $S^i$ get  $S'^i$
    12.     Use current ensemble $C^*$ to make predictions on $D_-^\Delta$, to get $Prob(D_-^\Delta)$
    13.     Calculate effectiveness $ee(x_j^\Delta) = |Prob(x_j^\Delta) - \mathrm{Y}(x_j^\Delta)|^2$for each majority instance $x_j^\Delta$ in $D_-^\Delta$,
            and choose top $IncNum = |D_+| + |S'| - |D_-^i|$ examples to form active set $D_-^{\Delta i}$
    14.     Update current negative set $D_-^i = D_-^i \cup D_-^{\Delta i}$; update remaining negative set $D_-^\Delta = D_-^\Delta - D_-^{\Delta i}$
    15.     Updated current total training data set to be: $D'\ =\ D_+\ \cup\ S'^i \cup D_-^i$
    16.     $C'\ =\ Learner\ (D',\ Cost)\ ,\ C^* =\ C^* \cup \{\ C'\}$
    17.     Computer new ensemble performance: $E' =\ Eval\ (C^*(X),\ Y,\ Cost)$
    18.     If $E' \le E_{best}$ (i.e., performance is better)
    19.         $mem = mem + 1,\ E_{best} = E'$
    20.     Else:
    21.         $\mathrm{C}^*\ =\ \mathrm{C}^* -\ \{\mathrm{C}'\}$
    22.     $i\ =\ i + 1$
Outputs:
    Ensemble classifier $C^*$
```

6.3 Experimental Settings and Results

To evaluate the performance of Active-DECIDL, we run the experiments on the 30 data sets in the benchmark pool. We compare the classification performance of Active-DECIDL with DECIDL, randomly under-bagging and randomly over- bagging; similar to previous experiments, the F-measure and MCC are reported for comparisons. Meanwhile, the percentages of queried majority data instances are also reported to show the effectiveness of active learning; we defined such percentage as the data query ratio. SVM is used as the base learner and all three methods have equal number of committee member. Therefore, the active learning procedure stops at the same time when the size of committee member reaches to a predefined number.

The parameter settings of Active-DECIDL are set to be the same as DECIDL, in order to have a fair comparison. More precisely, the size of artificial set, the number of committee members, and the evaluation metric are all same for Active-DECIDL and DECIDL. The parameters inside the base learner (SVM) are optimized internally by an internal training-testing step. The final reported performance of each method was averaged over ten runs of 5-fold cross-validation.

Table 6.2: Classification performance and data query ratio of Active-DECIDL on benchmark data

Data	F-mea.	G-means	MCC	AUROC	AUCPR	Data Query Ratio
Ecoli	0.633	0.858	0.605	0.942	0.610	12.3%
Optical_Digits	0.921	0.968	0.913	0.996	0.971	8.2%
SatImage	0.558	0.821	0.518	0.923	0.582	27.5%
Pen_Digits	0.909	0.965	0.900	0.997	0.970	15.5%
Abalone_7	0.381	0.741	0.342	0.850	0.319	57.4%
Sick_Euthyroid	0.715	0.906	0.695	0.950	0.747	18.6%
Spectrometer	0.840	0.933	0.831	0.954	0.831	25.8%
Balance	0.063	0.222	-0.017	0.416	0.072	53.7%
Car_Eval_34	0.927	0.987	0.923	0.999	0.964	24.2%
ISOLET	0.838	0.955	0.829	0.990	0.923	16.5%
US_Crime	0.540	0.804	0.510	0.922	0.566	42.4%
Yeast_ML8	0.126	0.467	0.019	0.531	0.090	84.5%
Scene	0.252	0.534	0.207	0.771	0.228	79.5%
Libras_Move	0.717	0.902	0.717	0.979	0.804	29.3%
Thyroid_Sick	0.662	0.904	0.656	0.946	0.642	38.6%
Coil_2000	0.163	0.472	0.116	0.671	0.130	90.1%
Arrhythmia	0.204	0.682	0.197	0.752	0.181	46.2%
Solar_Flare_M0	0.197	0.480	0.160	0.738	0.167	80.9%
OIL	0.354	0.773	0.360	0.917	0.385	50.8%
Car_Eval_4	0.868	0.992	0.873	0.999	0.933	44.7%
Wine_Quality_4	0.151	0.704	0.170	0.778	0.216	19.6%
Letter_Img	0.777	0.825	0.781	0.992	0.866	67.8%
Yeast_UCI_ME2	0.372	0.773	0.383	0.888	0.311	83.0%
Web_page	0.452	0.784	0.457	0.925	0.464	55.1%
Ozone_Level	0.192	0.487	0.188	0.772	0.180	91.3%
Mammography	0.490	0.837	0.504	0.921	0.540	83.1%
Reuters-21578	0.582	0.887	0.600	0.984	0.619	54.9%
Forest_Cover_5	0.093	0.743	0.150	0.831	0.079	80.0%
Protein_homo.	0.137	0.745	0.199	0.787	0.307	90.7%
Abalone_19	0.035	0.484	0.041	0.666	0.022	90.4%
Average	**0.472**	**0.755**	**0.461**	**0.860**	**0.491**	**52.1%**

Table 6.3: Performance Comparison among Active-DECIDL, DECIDL, under-bagging and over-bagging

Performance	F-measure				MCC			
Data Set	**Active-DECIDL**	**DECIDL**	**Over-bagg.**	**Under-bagg.**	**Active-DECIDL**	**DECIDL**	**Over-bagg.**	**Under-bagg.**
Ecoli	0.633	0.609	0.567	0.517	0.605	0.576	0.552	0.515
Optical_Digits	0.921	0.822	0.735	0.696	0.913	0.805	0.734	0.682
SatImage	0.558	0.358	0.000	0.276	0.518	0.358	0.272	0.256
Pen_Digits	0.909	0.775	0.000	0.629	0.900	0.763	0.702	0.616
Abalone_7	0.381	0.310	0.000	0.334	0.342	0.239	0.337	0.309
Sick_Euthyroid	0.715	0.751	0.605	0.584	0.695	0.727	0.607	0.581
Spectrometer	0.840	0.750	0.811	0.636	0.831	0.738	0.796	0.637
Balance	0.063	0.000	0.059	0.088	-0.017	0.000	-0.184	-0.159
Car_Eval_34	0.927	0.784	0.913	0.706	0.923	0.784	0.907	0.713
ISOLET	0.838	0.776	0.769	0.671	0.829	0.757	0.771	0.675
US_Crime	0.540	0.449	0.465	0.445	0.510	0.448	0.446	0.451
Yeast_ML8	0.126	0.157	0.159	0.173	0.019	0.060	0.074	0.098
Scene	0.252	0.210	0.236	0.241	0.207	0.138	0.180	0.208
Libras_Move	0.717	0.832	0.678	0.674	0.717	0.834	0.685	0.676
Thyroid_Sick	0.662	0.635	0.515	0.481	0.656	0.618	0.528	0.504
Coil_2000	0.163	0.117	0.189	0.177	0.116	0.029	0.153	0.140
Arrhythmia	0.204	0.206	0.261	0.154	0.197	0.170	0.214	0.125
Solar_Flare_M0	0.197	0.146	0.170	0.190	0.160	0.130	0.158	0.199
OIL	0.354	0.301	0.378	0.292	0.360	0.333	0.412	0.333
Car_Eval_4	0.868	0.936	0.812	0.752	0.873	0.934	0.823	0.768
Wine_Quality_4	0.151	0.269	0.171	0.177	0.170	0.242	0.198	0.175
Letter_Img	0.777	0.844	0.541	0.486	0.781	0.849	0.548	0.534
Yeast_UCI_ME2	0.372	0.137	0.284	0.294	0.383	0.156	0.334	0.341
Web_page	0.452	0.582	0.501	0.306	0.457	0.594	0.485	0.370
Ozone_Level	0.192	0.111	0.237	0.173	0.188	0.102	0.281	0.235
Mammography	0.490	0.530	0.264	0.217	0.504	0.554	0.347	0.285
Reuters-21578	0.582	0.647	0.749	0.433	0.600	0.655	0.802	0.491
Forest_Cover_5	0.093	0.285	0.102	0.091	0.150	0.276	0.199	0.154
Protein_homo.	0.137	0.675	0.420	0.126	0.199	0.679	0.470	0.223
Abalone_19	0.035	0.082	0.049	0.024	0.041	0.151	0.105	0.056
Average	**0.472**	**0.469**	**0.388**	**0.368**	**0.461**	**0.457**	**0.431**	**0.373**
Win/Loss to Under-bagg.	**26**	**22**	**21**		**21**	**21**	**24**	

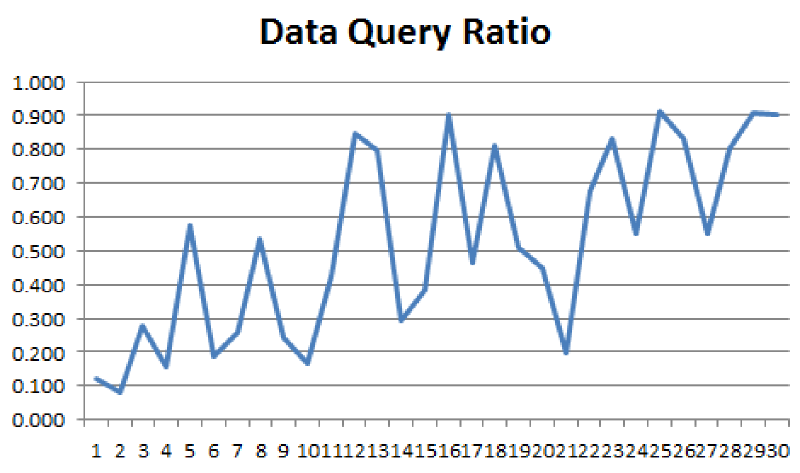


Figure 6.2: Data query ratio of majority class for the 30 benchmark data

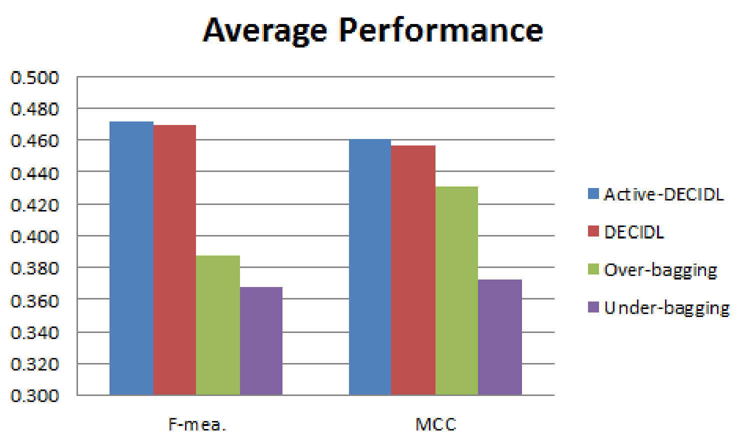


Figure 6.3: Average Performance Comparison of the four methods on benchmark data pool

6.4 Result Discussion and Conclusion

Table 6.2 shows the detailed F-measure, MCC, G-means, AUCROC, AUCPR and data query ratio of Active-DECIDL on the 30 benchmark data. In general, the Active-DECIDL works well on most of the data sets, although some of them are still difficult to tackle, e.g., Balance, Yeast_ML8, Coil_2000, Wine_Quality_4, Forest_Cover_5, Abalone_19. The possible reason is that underlying distributions of minority class may be very complicated, and selecting informative majority instances do not alleviate the difficulty of creating effective classification models.

However, it is worth to notice that the data query ratios on these data sets are generally low, ranging from less than 8% to 90%, which means in most cases, the active learning could significantly reduce the number of examples needed to build efficient classification models. This result successfully proves that active learning is very effective in reducing labeled examples in machine learning process, while still keeping the similar prediction performance.

Result in Table 6.3 and Figure 6.3 compare the classification performance between Active-DECIDL, DECIDL, under-sampling and over-sampling. Clearly, from those results, we can see that Active-DECIDL is generally superior to other three methods in terms of performance both on F-measure and MCC. The phenomenon that Active-DECIDL is slightly better than DECIDL strongly suggests that active learning is able to help a learning algorithm to reach the same level of classification performance while using much less data examples for training. This effect completes our original idea of using active learning to help solving class imbalance problem. The developed Active-DECIDL method has confirmed our initial hypothesis: the combination of active learning and DECIDL are fitting to each other for handling imbalance data learning problem.

Chapter 7: PROTEIN METHYLATION PREDICTION

In this chapter, we will study the problem of protein methylation prediction and apply the DECIDL framework to solve it. A brief introduction will outline the methylation prediction problem and its current research statuses and challenges. The biological knowledge about protein methylation is presented to provide further detailed information about the target problem—a real-world imbalanced data learning in bioinformatics. Many researchers have already provided their own classification methods and servers to predict methylated positions in protein sequence. Here, we apply our DECIDL framework that is specially designed for imbalanced learning to solve this problem more effectively.

7.1 Introduction

As we all know, cell is the fundamental functional unit of all kinds of lives in our world, and protein is an essential component of all kinds of cells. Proteins are playing many critical roles in defining particular functions and structures of cells. They can also help cells communicating with outside environment and transport useful substances for cells to survive, grow and reproduce.

Protein methylation is one important type of post-translational modifications of proteins. It was discovered more than 40 years ago [PK67], and offers greatly functional diversity to the protein sequence. It typically happens on arginine or lysine amino acid in a protein sequence [Wal05] [BR05]. The η-nitrogens of an arginine residue in a protein can be monomethylated or dimethylated, with either both methyl groups on single terminal nitrogen (asymmetric dimethylated arginine) or on either nitrogens (symmetric dimethylated arginine) by protein arginine methyltransferases (PRMTs) [BR05]. The ε-amino group of lysine can be methylated in mono-, di-, and tri-methylated states by protein lysine methyltransferases. Currently, protein methylation has been intensely studied in the histones—the main protein components of chromatin because of its important role in epigenetic regulation of gene functions.

Although protein methylation is an important modification in biology and chemistry, the molecular mechanism underlying the methylation is still poorly understood. In particular, the information of the substrate specificity of different PRMT members is largely incomplete. A genome-wide of searching of methylated substrates is highly needed to unravel many unknown functions of PRMTs in biological processes and cellar components. Therefore, accurately predicting these methylation sites will greatly help researchers setting smart experiments to find more interesting sites in unknown sequences. Currently, there exist some algorithms published in literature to handle this methylation prediction problem [CXH+06] [SLC+09] [SXT+09]. Many of them used the SVM method to train classifiers on known protein sequences and then made predictions on unknown sequences. If there are enough identified methylation sites for SVM training, then it's easy to build reliable and robust classifiers for further prediction. However, in most cases, the number of methylated sites is far less than un-methylated ones, and thus the dataset for SVM training is highly imbalanced. Directly using SVM to build classifiers will result in highly skewed hyperplanes, which give the same prediction for any other sequences without truly differentiating them. Therefore, it's important to design algorithms to solve the problems in imbalanced and small datasets.

7.2 Background

7.2.1 Protein Structure

Proteins are normally made of one or more polypeptides and fold to particular 3D structures based on the amino acid sequence in those polypeptides. A polypeptide is a linear polymer chain that is consisting of many amino acids and bonded by the peptide bonds between the carboxyl and amino groups of neighboring amino acid. The amino acid, or called residue in a sequence, is encoded by gene code, and there are on 20 kinds of them in total. However, a sequence of these amino acids can produce countless number of combinations, and thus there are more than millions of meaningful and functional proteins. A typical protein contains hundreds or thousands of amino acid residues. In essence, it is the primary amino acid sequence that determines the 3D structure and functionality of a protein. Proteins have four levels of structures in general. The first level is its primary structure, which is the sequence of 20 kinds of amino acids. The second level is protein secondary structure, which has 3 main forms of shapes called helix, strand, and coil. Each shape is formed by dozens or hundreds of amino acids. The third level is the tertiary structure, which is also the 3D folding structure of protein. It is mainly the spatial assembly of helices (helix), sheets (strand) and the pattern of connections between them (coil). The fourth level is for those proteins with more than one polypeptide chain. The combinations of those tertiary structures make up the quaternary structure. More detailed knowledge for protein structure can be found in [PR04].

7.2.2 Protein Methylation

The genetic material present in the nucleus of eukaryotic cells is tightly packaged into chromatin, which functions as a structural and dynamic scaffold in the regulation of various nuclear processes, including transcription, DNA replication and repair, mitosis and apoptosis [HM05] [Tch05]. From the past decade of work, epigenetics has been established as a critical topic of research in current biology dealing with heritable changes in a gene function that does not entail changes in the nucleotide sequence of the gene [Hol87][Bir02] [WM01] [CL05]. Epigenetic factors, in particular DNA methylation and histone modifications, significantly contribute to chromatin remodeling and function [RJ00]. This rapidly evolving field offers exciting new opportunities for investigating the molecular and cellular mechanisms underlying various poorly understood biological phenomena such as dosage compensation [BBK+05] and genomic imprinting [KS05], as well as providing new approaches to the diagnosis and treatment of complex clinical disorders such as cancer and cardiovascular diseases [YJ06].

Among different chromatin modifications, the histone arginine methylation is catalyzed by protein arginine methyltransferases (PRMTs) that transfer the methyl group from S-adenosyl-L-methionine

(AdoMet, SAM) to the guanidino group of arginines in histone or non-histone protein substrates, resulting in mono and di-methylarginine residues in substrate proteins (Figure 7.1).

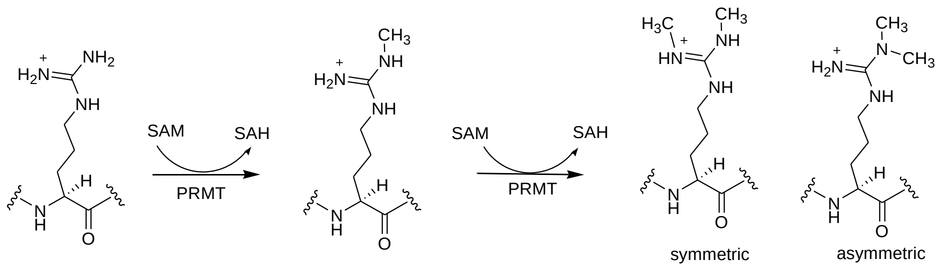


Figure 7.1: Protein arginine methylation catalyzed by PRMTs

Eleven PRMT family members have been identified at the protein and genomic levels in human tissues or cells and categorized into two major types, type I and type II, according to substrate and product specificity [BR05] [LTS+05].Type I enzymes (PRMT1, 3, 4, 6 and 8) catalyze the transfer of the methyl group from S-adenosyl-L-methionine (SAM, AdoMet) to the guanidino nitrogen atoms of arginine residue to produce ω-NG monomethylarginines (MMA, L-NMMA) and ω-NG,NG-asymmetric dimethylarginines (ADMA) (for a review, see a ref [BR05]). Type II enzymes (PRMT 5, 7 and 9) catalyze the formation of MMA and ω-NG,N'G-symmetric dimethylarginines (SDMA). Of note, the enzymatic activity of PRMT2, 10 and 11 remains uncharacterized. PRMT-catalyzed arginine methylation has been shown to be involved in many biological processes including signal transduction, gene transcriptional regulation, RNA transport, RNA splicing, and embryonic development [BR05].

Arginine methylation has a clear impact on the ability of the PRMT substrates to perform their biological functions. Although it does not alter the overall charge on an arginine residue, the addition of methyl groups increases steric hindrance and removes amino hydrogens that might be involved in hydrogen bonding. Therefore, methylation could serve to modulate intra- or intermolecular interactions of target proteins. This may be suggestive of the role of protein arginine methylation as a mark of signaling transduction in cells. However, a challenging problem in the study of protein arginine methylation is understanding and exploration of the substrate specificity of PRMTs. Thus far, only limited numbers of substrates of PRMTs (~50-70 confirmed reports in total for all the PRMT members) have been identified and experimentally verified. In particular, no apparent methylation consensus sequence has been determined. PRMT1 (like PRMT3) mediates methylation typically within the archetypal Arg–Gly- or Arg–Gly–Gly sequences [BSA06]. This overly simplified rule cannot account for all the existing PRMT substrates. Recent studies have shown that the substrate specificity of PRMT1 goes far broader than the typical RGG paradigm and suggest that many of the cellular functions of PRMT1 may not yet have been explored [WZZ+08]. A better computer-aided analysis of the existing PRMT substrates will yield new insight into the diversity of PRMT substrates, and identify new PRMT targets and related biological pathways.

7.3 Related Works in Protein Methylation Prediction

Four current protein methylation prediction servers are (1) the Memo [CXH+06], (2) the AutoMotif [PTW+05], the MASA [SLC+09], and the BPB-PPMS [SXT+09].

Memo: In [CXH+06], Chen et al. collected verified methylation sites from annotations in SWISS-PROT database. They also searched the PubMed with keyword “Methylation Arginine” and then manually collected confirmed methylations. After combining data from two resources together, BlastClust is used to remove homologous proteins. Finally, 250 methylated arginines were used as the positive samples for SVM modeling, and all other non-methylated arginines from the same proteins were used as negative samples. RBF kernel function was used and the average cross-validation accuracy is 86.70% for Arginine prediction. However, the collected data in Memo didn’t differentiate protein substrates methylated by different PRMTs. Therefore, even it makes predictions on unknown protein sequences, but it provides little information about which PRMT family causes this methylation. Such information is critical for users to verify the prediction. Meanwhile, mixing methylations by different protein families together for training SVM will also potentially overestimate the number of methylated Arginines when predicting.

AutoMotif Server[PTW+05]: Positive instances which are 9-amino acid long sequence fragments have been confirmed with one type of post-translational modifications (PTMs) from SWISS-PROT database [PTW+05]. The negative instances are constructed randomly by choosing fragments without any type of PTMs. They collected all type of PTMs, and each type of PTM has been trained and modeled separately. Thus, user can choose one specific modification to predict. However, similar to the Memo, this server doesn't provide further information about which PRMT is responsible for the methylation. Noticeably, their Leave-one-out precision and recall performance for Omega-N-methylated arginine are both 0, due to highly imbalanced data (62 positives, and 2044 negatives). In fact, we have tried several known sequences, such as Histone 2 & 3, but it produces either incomplete predictions or nothing.

MSAS Server: Most recently, Shien et al. [SLC+09] combines more structural information of the sequences to identify methylation sites. They first collect methylated lysine, arginine and other residues from Memo and newest SWISS-PROT database. Then each methylated fragment was encoded with their amino acid characteristics, predicted secondary structures (with PSIPRED), and predicted area accessible ability (with RVP-Net). They also use SVM as the learning classifiers. Since more methylated residues have been collected and more information has been used for representation, their method produces the highest prediction accuracy compared with other existing methods. The sensitivity they got is 82.1% and the specificity is 87.4%. Similar to previous methods, their server also only provide a general methylation prediction, and the related PRMTs are not identified. After balancing the positives and negatives with ratio 1:1 (246 samples), they report the average cross validation with 82.1% for sensitivity and 87.4% for specificity. However, the MSAS server only provides the general methylation prediction without identifying related PRMTs.

BPB-PPMS Server: Most recently, Shao et al. [SXT+09] also develop a new computational model for methylation identification. They too collect methylated residues from the SWISS-PROT database (version 56) by search keywords. Then they compute the posterior probability of each amino acid in positive dataset and negative dataset, respectively, which is called position-specific bi-profile. Through Bayes formula, bi-profile is used to predict the methylation status given any future unknown sequences. The final dataset for cross validation is balanced with ratio 1:3 between positives and negatives; the final classification performance on arginine prediction is 74.71% for sensitivity and 94.32% for specificity. Obviously, their method only uses the basic residue frequency information for learning, which underestimates the importance of other features, such as evolutionary profile, structure information, etc.

Summaries of features of the four Web servers are listed in Table 7.1.

Table 7.1: Features of the Four Current Prediction Web Servers

Features	***Memo***	***AutoMotif***	***MASA***	***BPB-PPMS***
Model for Classification	SVM	SVM	SVM	SVM
Sequence Representation	Bin (Binary)	Bin, Blosum, Frequency	Bin, 2D Structure, Solvent Acc.	Frequency
Data Under-sampling / Ratio (negative vs. positive)	Yes 1:1	No 46:1	Yes 5:1	Yes 3:1
Learning and Predicting on different PRMT families	NO	NO	NO	NO

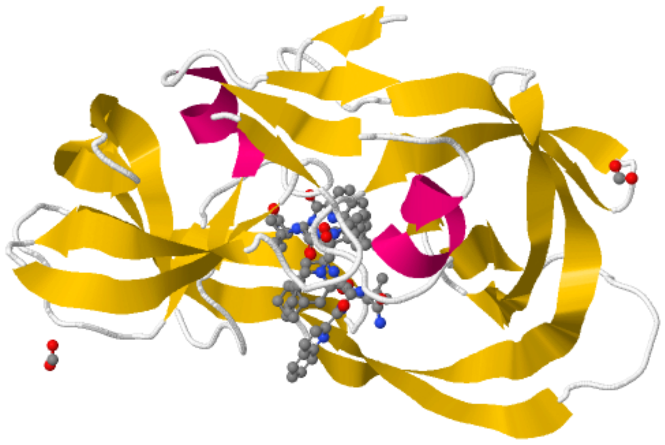
7.4 Our Proposed Algorithms

7.4.1 Representation of Protein Sequence

As described before, protein methylation is one kind of modifications after proteins are translated from gene codes to their primary amino acid sequences. Hence, the main information we can use to predict such methylation are their polypeptides, i.e., their amino acid sequences. Table 7.2 lists all the 20 kinds of amino acids and their properties and Figure 7.2 shows a typical protein sequence and its structures [KWT+07].

Table 7.2: 20 Amino Acids and their Abbreviations

Amino Acid	3-L Abbr.	1-L Abbr.	Polarity	Amino Acid	3-L Abbr.	1-L Abbr.	Polarity
Alanine	Ala	A	nonpolar	Leucine	Leu	L	nonpolar
Arginine	Arg	R	polar	Lysine	Lys	K	polar
Asparagine	Asn	N	polar	Methionine	Met	M	nonpolar
Aspartic acid	Asp	D	polar	Phenylalanine	Phe	F	nonpolar
Cysteine	Cys	C	nonpolar	Proline	Pro	P	nonpolar
Glutamic acid	Glu	E	polar	Serine	Ser	S	polar
Glutamine	Gln	Q	polar	Threonine	Thr	T	polar
Glycine	Gly	G	nonpolar	Tryptophan	Trp	W	nonpolar
Histidine	His	H	polar	Tyrosine	Tyr	Y	polar
Isoleucine	Ile	I	nonpolar	Valine	Val	V	nonpolar



Fasta format of 3OXC
>3OXC:A|PDBID|CHAIN|SEQUENCE
PQITLWKRPLVTIKIGGQLKEALLDTGADDTVI
EEMSLPGRWKPKMIGGIGGFIKVRQYDQIIIEI
AGHKAIGTVLVGPTPVNIIGRNLLTQIGATLNF

Figure 7.2: Protein (PDB ID: 3OXC [KWT+07]) Structure and Sequence

Clearly, the protein sequences are consisting of long string combinations of 20 kinds of amino acids, and they cannot be directly used as numbers for the computations in our learning algorithms. Therefore, the first step is to convert these sequence information into integers or float numbers in a reasonable manner, and then identify the independent features and dependent target class. After this step, we can fit these values into learning algorithms for classification.

Before introducing any advanced sequence representation techniques, let first describe a simple idea of converting these amino acids into integers. As there is limited number of amino acids, it is similar to a nominal feature for each position in a protein sequence. Hence, a traditional way for amino acid is using a 20-bits binary value to represent each amino acid. First, we rank all amino acids based on their 1-letter name alphabetically (as shown in Table 7.1). For example, Alanine is the first; Arginine is the second, and so on. Then, the 20-bits representation will have a 1 in its ith position where i is its ranking number. All other positions will have zeros. Hence, the 20-bits value for each amino acid has only 1 one and 19 zeros. We call this representation method—orthogonal code. A more concrete example is described in Table 7.3.

Obviously, these binary representations can be considered as the independent features for further classification, because the primary sequences are the fundamental information for any advanced protein structures and functions. The next step is to choose a predicting target property. Depending on the interested question, it is generally a sophisticated protein characteristic of the sequence. For example, secondary structure, tertiary structure, surface solvent ability, and a post-translational modification are the most popular properties to be predicted based on amino acid sequences. It is generally assumed that an individual residue and its flanking amino acids are the main causes to determine the interested property on this residue; hence a subsequence around a center residue is used to represent the independent features for the target prediction of this residue. Assume that there are n residues on the left and n on the right, then there are total $w = 2n + 1$ residues together in determining the property of the residue in the center. Normally, w is chosen based on a particular problem, ranging from 7 to 21. The status of the centering residue is represented with +1 or -1, in order to fit to classification algorithms. By sequentially sliding the representation range to the next subsequence, we can collect more and more predicting examples and their features, resulting in a training data set. Such sliding technique is usually called *sliding window*.

Table 7.3: Orthogonal representation of 20 amino acids and a short sequence

Amino Acid	Rank	1-L Abbr.	Representation	Example
Alanine	1	A	$\underbrace{1\,0\,0\,0\,0 \ldots 0\,0\,0}_{20\ bits}$	Sequence : PQIT R LWKR Converted to: 00000 00000 00001 00000 00000 01000 00000 00000 00000 00001 00000 00000 00000 00000 00000 01000 01000 00000 00000 00000 00000 00000 10000 00000 00000 00000 00000 00100 00000 00000 01000 00000 01000 00000 00000 00000
Arginine	2	R	$\underbrace{0\,1\,0\,0\,0 \ldots 0\,0\,0}_{20\ bits}$	
Asparagine	3	N	$\underbrace{0\,0\,1\,0\,0 \ldots 0\,0\,0}_{20\ bits}$	
…	…	…	…	
Tyrosine	19	Y	$\underbrace{0\,0\,0\,0\,0 \ldots 0\,1\,0}_{20\ bits}$	
Valine	20	V	$\underbrace{0\,0\,0\,0\,0 \ldots 0\,0\,1}_{20\ bits}$	

7.4.2 Feature Extraction from Protein Sequences

Similar to other methods, we also assume that methylated Arginines are largely determined by its neighboring residues in primary sequence structures. That is, the characteristics of residues centered with an Arginine will be explored as deep and diversified as possible to find the potential relations. Each status of an Arginine (being methylated or non-methylated) is represented by a vector of features extracted from its primary neighboring residues. These features include physicochemical properties (PhCh), the multiple sequence alignment profiles (the PSSM), secondary structure (SS), and solvent accessible area (SAS). Thus, our first step is to extract these features for each Arginine in the PRMTs.

To better understand the underlying mechanism of protein Arginine methylation, enough information has to be collected for further machine learning task. So far, most studies of identifying methylation sites only use the primary structure information: the properties of amino acids and the evolutionary profiles. However, it is well known that the protein secondary and tertiary structures have significant impact on many PTMs. Therefore, structure information should also be used to differentiate the methylation status. In this paper, the secondary structure and solvent accessible area information are extracted along with physicochemical properties and PSSM to represent the characteristics of neighboring residues. We believe it's the first time that such comprehensive information has been explored for protein methylation identification.

The physicochemical (PhCh) properties of each amino acid include the status of being polar, charged, aromatic, small, tiny, hydrophilic/hydrophobic, and aliphatic [Tay86]. Each property is represented with a binary bit, so 8 bits are used for PhCh feature.

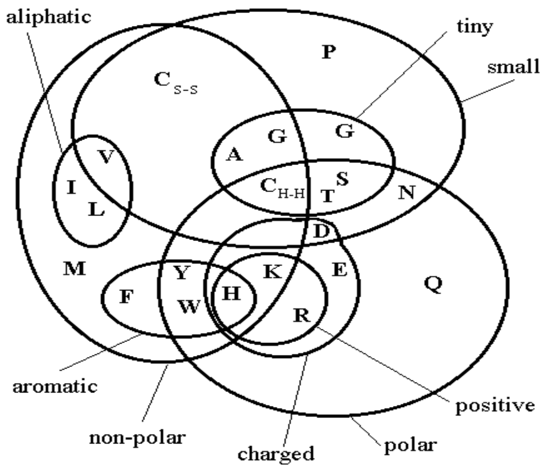


Figure 7.3: The physico-chemical relationship of the 20 amino acids [Tay86]

The evolutionary profiles of each protein sequence are generated by PSI-BLAST [AMS+97] against the NCBI non-redundant database with three iterations and a cutoff E-value 10-3. The position specific scoring matrix (PSSM) produced by PSI-BLAST is used as the evolutionary information; its value will be scaled in the range [0, 1].

Not all PRMT sequences have been discovered with 3D structures so far, thus the predicted structure information is used for feature extraction. For secondary structures of each sequence, the PSI-PRED server [BMM+05] can provide reliable three-status (helix, coil, and strand) prediction for each residue of a query sequence. The real values of three kinds of secondary structures are directly incorporated in the feature list.

For solvent accessible area, we also use the predicted information, which is obtained from RVP-net [AGS03]. The RVP-net can output real-valued predictions of accessible surface area for each amino acid; hence, it should bring more information for further classification than binary or discrete predictions.

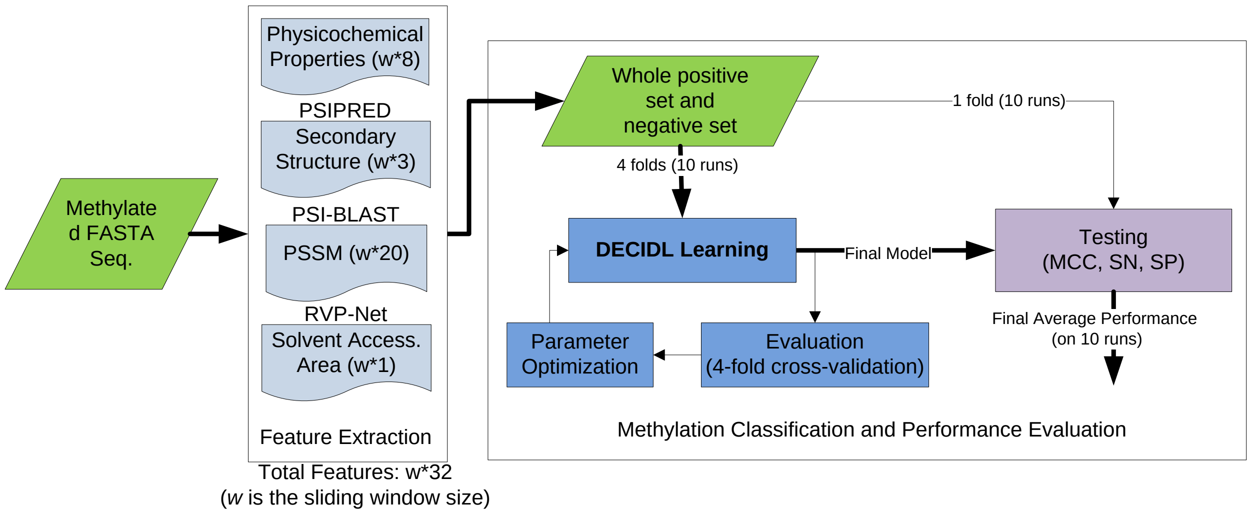

Methylated FASTA Seq.
Physicochemical Properties (w*8)
PSIPRED
Secondary Structure (w*3)
PSI-BLAST
PSSM (w*20)
RVP-Net
Solvent Access. Area (w*1)
Feature Extraction
Total Features: w*32
(w is the sliding window size)
Whole positive set and negative set
1 fold (10 runs)
4 folds (10 runs)
DECIDL Learning
Final Model
Testing (MCC, SN, SP)
Parameter Optimization
Evaluation (4-fold cross-validation)
Final Average Performance (on 10 runs)
Methylation Classification and Performance Evaluation

Figure 7.4: Data flow of imbalanced data learning on Methylated Protein Sequences

Hence, for each Arginine, the feature number is $w * 32$, where w is the total sliding window size. w is set with odd number between 7 and 21 in or computational experiments.

7.4.3 DECIDL on Methylation Data

As arginine methylation is a rare modification among all arginine in protein sequences, the resulting methylation data set is quite imbalanced. From the data sets used in the four methylation prediction server, we know that such imbalance ratio can reach to as high as 10:1, i.e., 216 methylated (positive) arginines and 1980 non-methylated (negative) arginines are collected in BPB-PPMS server[SXT+09]. Hence, this arginine methylation prediction problem is a perfect real-application test for our DECIDL framework and its variations. We will directly apply our DECIDL algorithm on the methylation data set. The detailed data collection step for methylated protein sequences will be discussed in next section.

Notice that, as summarized in section 7.3, most current methods used under-sampling to balance the data set and then apply SVM for model building. Meanwhile, their reported classification performances are totally based on the under-sampled balanced data set, not on the original data set.

7.5 Methylated Protein Sequence Collection

The methylated protein sequences for arginine in our following experiment are collected from MeMo server [CXH+06]. MeMo generated such data set in the following way: they first examined the methylation residues from the SWISS-PROT database (ver. 48) [BBA+03] and only selected those experimentally verified methylated arginine sites. In other words, potential methylated residues with keywords “By similarity”, “Potential” or “Probable” are not considered. They also manually search PubMed for keywords “methylation arginine” and found more than 1700 articles. The methylation information found by these two methods are combined together to produce 273 methylated arginine positions. To remove the effect of homologous proteins, they use BLASTCLUST with a 30% identity threshold to cluster similar protein sequences and then remove one of the cases if two sequences have methylated arginines at same positions after alignment. They finally collected 250 positive methylated arginines from 91 proteins. All the non-methylated arginines in these 91 proteins are considered as negative cases, resulting in 2700 negative examples. Hence, the data set in our experiment has 250 positive examples, and 2700 negative examples, meaning the imbalance ratio is about 11:1.

7.6 Computational and Laboratorial Results

In this section, we will show the computational and laboratorial results on this methylation data set with several imbalanced learning strategies. Four ensemble methods used in previous chapters are selected for comparisons: DECIDL, GSVM-RU, SMOTE-bagging, and DBEG-ensemble. We use two levels of 5-folds cross validation to report their averaged performance metrics over 10 independent runs. The two levels of cross validation are conducted as follows: the original methylated data set (250 positives and 2700 negatives) are randomly divided into 5 equal folds, then one fold is used as independent testing set and the rest 4 folds are used as training set. Within these 4 folds, 3 of them are used for building prediction models, and the left 1 fold is set to validate the performance. After each of the 4 folds is treated as validation fold, their averaged performance is used to select the best modeling parameters and a final prediction model is built on all 4 folds data with the best parameters. The process of cross validation, parameter optimization, and evaluations is described in figure 7.5.

7.6.1 Computational Results

Table 7.4 and Figure 7.5 to 7.10 show the F-Measure, G-means, and MCC performance of the four imbalanced learning methods with window size ranging from 7 to 21.

From these results, we can see that our DECIDL method can produce high and stable performance with different sizes of window. All the classification metrics, i.e., F-measure, G-means, MCC, and AUC-ROC, decrease very dramatically for all other three learning methods (GSVM-RU, SMOTE-bagging, DBEG-ensemble), except our DECIDL method.

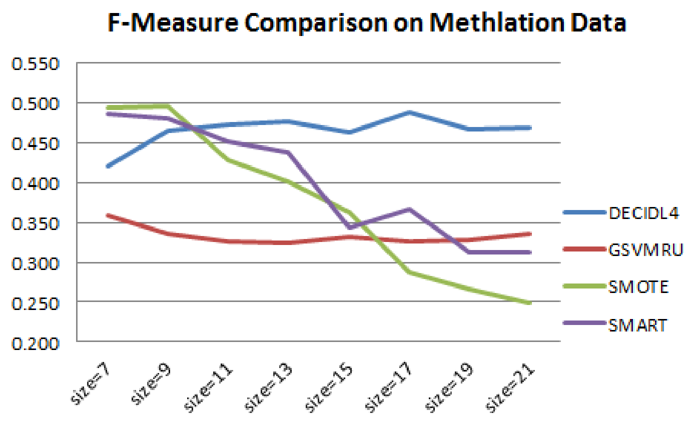


Figure 7.5: F-Measure Performance Comparison with Different Window Sizes

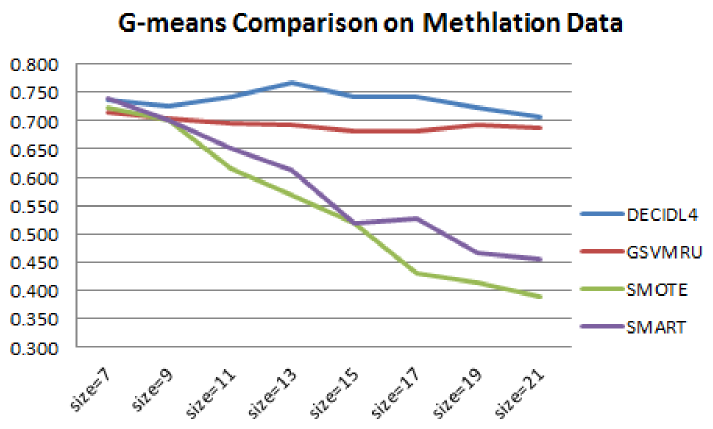


Figure 7.6: G-means Performance Comparison with Different Window Sizes

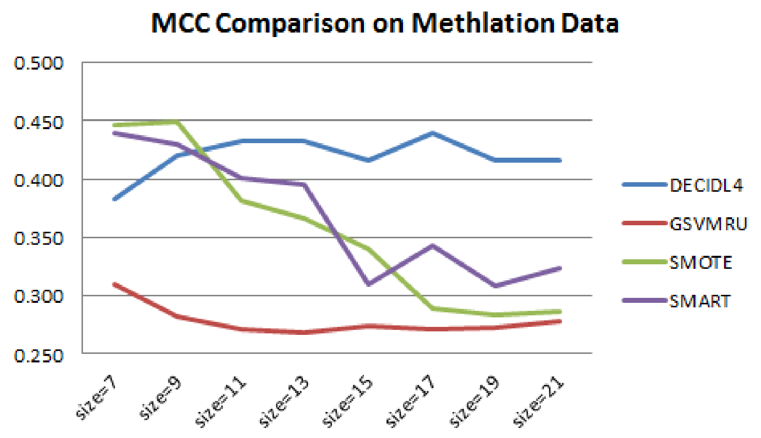


Figure 7.7: MCC Performance Comparison with Different Window Sizes

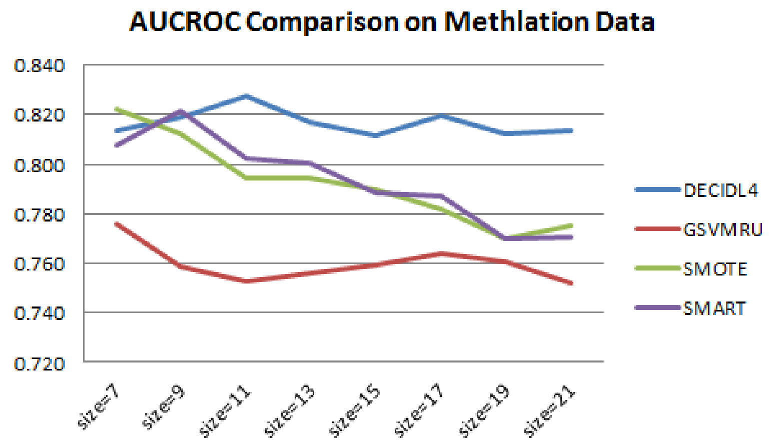


Figure 7.8: AUC-ROC Performance Comparison with Different Window Sizes

Table 7.4: Classification Performance on Arginine Methylation Data Set

Window Size	Metric	F-Mea.	G-means	MCC	AUROC	AUCPR	Sen.	Spec.
size=07	DECIDL	0.421	0.736	0.382	0.814	0.382	0.642	0.864
	GSVM-RU	0.359	0.715	0.310	0.776	0.319	0.631	0.819
	SMOTE-Bagging	0.495	0.723	0.446	0.822	0.401	0.563	0.933
	DBEG-Ensemble	0.487	0.740	0.440	0.808	0.390	0.598	0.920
size=09	DECIDL	0.464	0.727	0.420	0.819	0.407	0.595	0.903
	GSVM-RU	0.336	0.704	0.283	0.759	0.287	0.628	0.796
	SMOTE-Bagging	0.497	0.702	0.449	0.812	0.406	0.524	0.944
	DBEG-Ensemble	0.480	0.702	0.430	0.822	0.420	0.530	0.936
size=11	DECIDL	0.472	0.743	0.432	0.827	0.404	0.622	0.906
	GSVM-RU	0.326	0.694	0.271	0.752	0.280	0.620	0.789
	SMOTE-Bagging	0.429	0.616	0.382	0.794	0.387	0.402	0.955
	DBEG-Ensemble	0.452	0.652	0.401	0.802	0.395	0.454	0.948
size=13	DECIDL	0.477	0.766	0.433	0.817	0.377	0.659	0.894
	GSVM-RU	0.323	0.692	0.269	0.756	0.283	0.616	0.790
	SMOTE-Bagging	0.401	0.569	0.367	0.794	0.387	0.341	0.967
	DBEG-Ensemble	0.437	0.612	0.396	0.800	0.409	0.391	0.962
size=15	DECIDL	0.463	0.741	0.416	0.811	0.381	0.611	0.906
	GSVM-RU	0.332	0.682	0.274	0.759	0.283	0.578	0.818
	SMOTE-Bagging	0.363	0.520	0.341	0.790	0.400	0.285	0.975
	DBEG-Ensemble	0.343	0.518	0.309	0.789	0.370	0.282	0.968
size=17	DECIDL	0.488	0.741	0.439	0.820	0.418	0.603	0.916
	GSVM-RU	0.327	0.683	0.271	0.764	0.292	0.588	0.811
	SMOTE-Bagging	0.288	0.430	0.290	0.782	0.362	0.203	0.984
	DBEG-Ensemble	0.367	0.527	0.342	0.787	0.390	0.289	0.973
size=19	DECIDL	0.466	0.723	0.416	0.812	0.382	0.574	0.917
	GSVM-RU	0.329	0.693	0.273	0.761	0.289	0.608	0.799
	SMOTE-Bagging	0.267	0.413	0.284	0.770	0.362	0.176	0.987
	DBEG-Ensemble	0.313	0.465	0.308	0.770	0.363	0.225	0.980
size=21	DECIDL	0.469	0.705	0.416	0.814	0.411	0.540	0.927
	GSVM-RU	0.336	0.687	0.279	0.752	0.305	0.584	0.819
	SMOTE-Bagging	0.249	0.390	0.286	0.775	0.380	0.158	0.991
	DBEG-Ensemble	0.313	0.456	0.323	0.771	0.381	0.215	0.986

7.6.2 Laboratorial Results

We have applied the under-sampling ensemble methods on three particular sub-classes of PRMT families and made predictions on several unknown protein sequences, and then submitted the predictions for a chemistry laboratory in our university for actual verification. Detailed experiments and results can be found in [DZZ09][DFZ+10]. Here, will briefly describe the whole process and laboratorial results.

We were interested in predicting the methylated arginines catalyzed by special types of PRMTs (PRMT1, PRMT4, and PRMT5), due to the special interests in the chemistry lab in our university. Thus, we manually examine the publications on PubMed and collect different groups of substrates based on their PRMT types. As a result, 28 confirmed methylated arginines and 523 non-methylated ones are found within 18 methylated PRMT1, PRMT4 and PRMT5 substrates. Obviously, the two groups are highly imbalanced. Then, we encode the 28 methylated fragments into vectors as positive examples, and the 523 fragments as negative examples. Each amino acid is represented with BLOSUM62 and PhCh properties, total in 28 bits. Thus, the total number of features for each fragment is 28 * length of the fragment size. Different fragment sizes are used to generate different datasets and our algorithms are executed on each of them, to find the best fragment size.

We applied the granular computing ideas into the simple imbalanced learning process and developed an ensemble under-sampling algorithm to build predicting committee for the three kinds of PRMTs. SVM is used as the base learner. The result of this under-sampling strategies with a leave-one-out cross-validation are shown in Table 7.5.

Table 7.5: Under-sampling Classification Performance on Three PRMT Data Sets

Datasets (pos : neg)	Sen.	Spec.	Accu.	G-means	AUC-ROC
PRMT1 (15:209)	0.733	0.852	0.793	0.790	0.818
PRMT4 (8:196)	0.625	0.842	0.733	0.725	0.735
PRMT5 (5:118)	0.800	0.924	0.862	0.860	0.903
PRMT 1, 4, 5 (28:523)	0.714	0.861	0.788	0.784	0.833

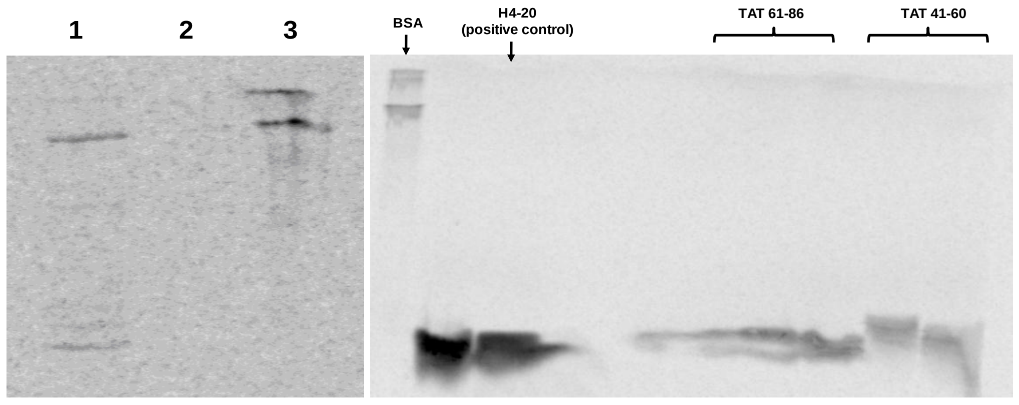
After optimizing parameters for the SVM learner, we then trained classification models on the whole datasets, and make predictions on further unknown sequences, e.g. Tip60 and HIV-1 Tat, as show in Table 7.9.

Table 7.6: Probability Predictions on Two Unknown Protein Substrates

Substrate	PRMT1	PRMT4	PRMT5
tip60	379R 29.8%	186R 5.5% 239R 5.5%	220R 0.9% 186R 0.2%
hiv1_tat	78R 31.7% 57R 11.4%		49R 9.1% 57R 1.8%

We then sent these predictions for real biological experiments. Based on the prediction in Table II, it is quite likely that PRMT1 is able to methylate Tip60 at R379 position and HIV-1 Tat at R57 and R78 positions. To validate the prediction and to investigate potential functions of PRMT1 in virus infection, we expressed the Tip60 recombinant protein using a pET15b vector and tested the methylation by PRMT1 using radioisotope-labeled assay. Clearly, Tip60 is methylated by PRMT1 (in Figure 2). Further, we synthesized two peptides derived from HIV-1 Tat protein that contain the two predicted methyl arginines. They are 41-60 (KALGISYGRKKRRQRR**R**AHQ) and 61-86 (NSQTH-QASLSKQPTSQP**R**GDPTGPKE). Phosphor-imaging analysis showed clearly that both of the two peptides were methylated by PRMT1.

These experimental results strongly substantiate the efficiency of our granular decision fusion model, and also uncovered a new function of PRMT1 in the pathway of HIV infection.



(a) Tip60 methylated by PRMT1 (b) HIV-1 Tat methylated by PRMT1

Figure 7.9: Enzymatic assay of potential methylation substrates of PRMT1.

In Figure 7.9, the left graph clearly shows that Tip60 is methylated by PRMT1. Lane 1 is PRMT1/Tip60; lane 2 is negative control; and lane 3 is 14C-BSA standard. The right graph shows the methylation of Tat by PRMT1. The peptide H4-20 is used as a positive substrate control.

7.7 Conclusion

We have successfully applied a simple imbalanced learning strategy on the methylation dataset. The biological experiments have shown that our predictions do match certain methylation residues; hence the process and costs of identifying methylated arginines can be significantly reduced with our prediction algorithms.

Out next step is to apply our DECIDL algorithms on these three PRMT families to further enhance the prediction accuracies and submit another round of predictions on several new proteins that are currently interested in the chemistry lab. At the same time, we will also plan to build a predication sever that provide predictions for user-input sequences for 11 kinds of PRMTs.

Chapter 8: CONCLUSION AND FUTURE WORK

8.1 Conclusion

In this dissertation, we tried to solve the imbalance data learning problem by providing an effective ensemble framework and a series of related methods from different problem-solving angles.

We started from giving a formal definition to highly imbalanced binary classification problem and discussing several challenging aspects on this topic. A clear threshold for imbalance ratio is given based on statistical analysis to separate imbalanced learning problems and general learning problems. We believe that the new definition will be broadly accepted by researchers in machine learning and data mining communities for further study of imbalanced data classification. Meanwhile, we have also provided a comprehensive literature review on imbalance learning.

This dissertation introduced the DECIDL ensemble system, which is a robust and effective method that directly utilizes diversity to guide the construction of ensemble classification for highly imbalanced data learning. The fundamental idea of DECIDL is to synthesize artificial examples among the data spaces of different classes and reversely re-label these examples oppositely to current ensemble predictions. Due to the natural characteristics of imbalance data, several detailed strategies have involved and over-tuned to build a diversified ensemble committee, which overall distinguish our works with current imbalanced learning algorithms. Our method can be considered as the ideal combination of two general classification methods—SMOTE and DECORATE—for imbalance learning, but more powerful and generalized, due to the various tuning steps in the method procedure. As a meta learner, our DECIDL algorithm can be broadly used in any imbalance learning tasks to build high accurate prediction classifiers. Extensive experiments have preliminary shown that DECIDL system is comparable with many ensemble methods, such as under-bagging, over-bagging, SMOTE-bagging, etc.

We created a comprehensive standard imbalanced data benchmark pool to facilitate any future research on imbalance data learning, as well as test the performance of our DECIDL framework. Data sets in this benchmark are from various application domains, consisting of a large range of example sizes and feature sizes, with diverse feature forms, and a variety of imbalance ratios. Meanwhile, we systematically developed extensive experiments to globally reveal the learning performance of several popular imbalanced classification algorithms, such as under-sampling, over-sampling, SMOTE-bagging, AdaBoost and our DECIDL. To the best of our knowledge, such extensive experiment comparisons with many ensemble methods have never been conducted on highly imbalanced data sets before. Therefore, the experiments and results in this dissertation can also be used as a useful reference for future research on highly imbalanced data classification problems.

We further analyzed the artificial data synthesis step and developed a new strategy for data balancing, after studying the advantages and weakness of existing data synthesizing methods. The new method—DBEG—is able to create new minority examples based on the original data distributions in a very efficient way. With new minority generated by DBEG, we conducted experiments to show that DBEG-Ensemble and DECIDL-DBEG are both very effective, in comparison with the state-of-the-art imbalanced learning techniques, SMOTE-Bagging and GSVM-RU.

We have utilized active learning to help solving imbalanced learning problem. Active learning is able to identify useful unlabeled examples and incrementally build classification models. With active learning selecting only useful majority class instances, and DECIDL creating additional synthesized minority examples, the combination of the two methods produce great data manipulation power and also learning ability to classify highly imbalanced data. Extensive examples have proven that the joint performance is very promising on benchmark data sets.

Our final work is applying our proposed methods on a real-world application—protein arginine methylation prediction, which is a very hot research topic in bioinformatics. After conducting initial experiments in both computational and laboratorial directions, we found that the methods developed for imbalance learning do have great advantages in solving real-world problems.

8.2 Future Work

Although we have conducted studies on DECIDL in both experiment and application aspect, we still have several other directions to explore the scopes and potentials of DECIDL. Some close related directions, but not limited to, are listed here.

More experiments on DECIDL-DBEG and Active-DECIDL. In Chapter 5, we compared DECIDL-DBEG with DECIDL, SMOTE-Bagging, GSVM-RU; however, our main interest is to compare DBEG and other data synthesizing methods, e.g., ADASYN, Borderline-SMOTE, and DataBoost-IM. Hence, more experiments need to be performed to compare their effectiveness. Similar to Active-DECIDL, we also need to conduct more experiments to verify the efficiency of example query, e.g., measuring the classification performance when fixing the data query ratio.

Cost-sensitive base learner. In our current experiments with DECIDL learning, the four traditional base learners are Decision Stump, Decision Trees, Linear SVM, and Perceptron NN, which are not cost-sensitive learners. Therefore, to further evaluate effectiveness of DECIDL framework, we will use several cost-sensitive learners (such as the SVM-weight, cost-sensitive DTs, and cost-sensitive NNs) to build the ensemble committee. The following experiments will be conducted:

1) Running DECIDL based on cost-sensitive learners under different parameters;
2) Comparing the performance of individual cost-sensitive learners and the combination of cost-sensitive learner based DECIDL;
3) Showing the difference between DECIDL with cost-sensitive learners and traditional learners.

Boosting Classifiers inside DECIDL. Inside of our DECIDL procedure, every classifier member is individually created and a final hard or soft strategy is used to ensemble all members. Apparently, there is no boosting strategy involved, and all the examples are equally weighted. However, the previous experiments (Figure 4.3-4.7) showed that the AdaBoost could also produce very good performance on imbalanced data when a weak base learner is used. Hence, an immediate new idea will be to boosting methods (i.e., AdaBoost method) into our DECIDL framework. Since the boosting methods can utilize the pure base learners (AdaBoost) and cost-sensitive learners (AdaCost), we can have at least two alternatives to boost the DECIDL procedure. Potential future works with Boosted DECIDL would be:

1) Combining the AdaBoost strategy with our DECIDL framework and checking performance changes;
2) Combining the AdaCost boosting strategy to the DECIDL framework to find potential performance improvements;
3) Comparing the standard DECIDL, Boosted DECIDL (with AdaBoost), Cost-sensitive Boosted DECIDL (with AdaCost).

Imperfect data. The data used in our experiments are almost complete and perfect. However, in many practical applications, the collected data have plenty of missing values, or significantly noises among features and examples, or strongly correlated examples, etc. Therefore, using imperfect data sets to test the DECIDL ensemble framework is very important for real data mining applications. More specifically, we plan to test it with the following conditions:

1) Add extra normal distributed noises to original data values with various levels, e.g., 5%-30%;
2) Remove some features from some data examples at different percentages;
3) Add additional noisy features and/or noisy examples to original data sets.

We hope after giving further investigation on these directions, our DECIDL framework can be survived with strong performance on both experimental data real applications. More and more effective significantly imbalanced data learning algorithms can be developed based on DECIDL framework.

APPENDICES

Appendix A: PUBLICATIONS RELATED TO THIS RESEARCH

JOURNALS

• Z. J. Ding, Y. Feng, Y. G. Zheng, and Y.-Q. Zhang, Protein Methylation Prediction Using Granular Decision Fusion Methods, International Journal of Soft Computing and Bioinformatics, vol. 1, no. 1, pp. 19-27, 2010.

• Z. J. Ding, and Y.-Q. Zhang, Data Shuffling and Statistical Analysis on Microarray Data for Gene Selection-A Comparative Study on Filtering Methods, International Journal of Functional Informatics and Personalised Medicine (IJFIPM), vol. 3, no. 3, pp. 183-203, 2011.

CONFERENCE PAPERS

• Z. J. Ding and Y.-Q. Zhang, An Effective Filtering Gene Selection Method for Microarray Data via Shuffling and Statistical Analysis. ACM International Conference on Bioinformatics and Computational Biology (ACM-BCB 2010), Niagara Falls, Aug. 2-4, 2010. (Travel Grant Award)

• Z. J. Ding and Y.-Q. Zhang, Additive Noise Analysis on Microarray Data via SVM Classification. 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, May 2-5, 2010 Montreal, Canada.

• Z. J. Ding, Y.-Q. Zhang and Y. G. Zheng, Feature Selection and Granular SVM Classification for Protein Arginine Methylation Identification, Proc. of IEEE -SMC2009, San Antonio, Oct. 11-14, 2009.

• Z. J. Ding, Y.-Q. Zhang, Nan Xie and Y. G. Zheng, Identifying New Methylation Arginine via Granular Decision Fusion with SVM Modeling, Proc. of 2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing, pp. 237-241, Shanghai, Aug. 3-5, 2009.

• Z. J. Ding, Y. Feng, Y. G. Zheng and Y.-Q. Zhang, Granular Decision Fusion Systems for Effective Protein Methylation Predication, Proc. of IEEE CIBCB 2008, Sept. 15-17, Sun Valley, Idaho. (Travel Grant Award)

• Z. Ding, J. Yu, Y-Q. Zhang, A New Improved K-means Algorithm with Penalized Term. The 2007 IEEE Int'l Conf. on Granular Computing: Silicon Valley, CA, Nov. 2-4, 2007.

REFERENCES

[AGS03] S. Ahmad, M. M. Gromiha, and A. Sarai, “RVP-net: online prediction of real valued accessible surface area of proteins from single sequences,” *Bioinformatics*, vol. 19, no. 14, pp. 1849-1851, 2003.

[AKJ04] R. Akbani, S. Kwek, and N. Japkowicz, “Applying support vector machines to imbalanced datasets,” *Proc. 15th Euro. Conf. Machine Learning*, Pisa, Italy, 2004, pp.39-50.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schäffer, et al., “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389-3402, 1997.

[BBA+03] B. Boeckmann, A. Bairoch, and R. Apweiler, et al., “The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003”, *Nucleic Acids Research*, vol. 31, pp. 365-370, 2003.

[BBK+05] M. P. Bhadra, U. Bhadra, J. Kundu, et al., “Gene expression analysis of the function of the male-specific lethal complex in Drosophila,” *Genetics*, vol. 169, no.4, pp. 2061-2074, 2005.

[BGL+99] M. Brown, W. Grundy, D. Lin, et al., “Support vector machine classification of microarray gene expression data,” Dept. of Computer Science, Univ. of California, Santa Cruz, CA, Rep. UCSC-CRL-99-09, June 12, 1999.

[Bir02] A. Bird, “DNA methylation patterns and epigenetic memory,” *Genes and Development*, vol. 16, no. 1, pp. 6-21, 2002.

[BMM+05] K. Bryson, L. J. McGuffin, and R. L. Marsden, “Protein structure prediction servers at University College London,” *Nucleic Acids Research*, vol. 33, pp. W36-38, 2005.

[BPM04] G. E. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets*, vol. 5, no. 1, pp. 20-29, 2004.

[BR05] M. T. Bedford and S. Richard, “Arginine methylation: An emerging regulator of protein function,” *Molecular Cell*, vol. 18, pp. 263-272, 2005.

[Bre01] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[Bre96] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[BSA06] F. Blanchet, B. T. Schurter, and O. Acuto, “Protein arginine methylation in lymphocyte signaling,” *Current Opinion in Immunology*, no. 18, pp. 321-328, 2006.

[Bur98] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 121-167, 1998.

[BWH+05] G. Brown, J. Wyatt, R. Harris, et al., “Diversity creation methods: a survey and categorization,” *Information Fusion*, vol. 6, no. 1, pp. 5-20, 2005.

[BWT05] G. Brown, J. L. Wyatt, and P. Tiňo, “Managing diversity in regression ensembles,” *J. Machine Learning Research*, vol. 6, pp. 1621-1650, 2005.

[CC08] D. A. Cieslak and N. V. Chawla, “Learning decision trees for unbalanced data,” *Proc. 19th Euro. Conf. Machine Learning (ECML-2008)*, Antwerp, Belgium, 2008, pp. 241-256.

[CH67] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21-27, Jan. 1967.

[CHB+02] N. V. Chawla, L. O. Hall, K. W. Bowyer et al., “SMOTE: synthetic minority oversampling technique,” *J. Artificial Intell. Research*, vol. 16, pp. 321-357, 2002.

[CJK03] N. V. Chawla, N. Japkowicz, and A. Kolcz, *Proc. 20th Int. Conf. Machine Learning Workshop learning from imbalanced data sets II*, 2003.

[CJK04] N.V. Chawla, N. Japkowicz, and A. Kolcz, “Editorial: special issue on learning from imbalanced data sets,” *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 1-6, 2004.

[CL01] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001. Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[CL05] P. Cheung and P. Lau, “Epigenetic regulation by histone methylation and histone variants,” *Molecular Endocrinology*, vol. 19, no. 3, pp. 563-573, 2005.

[CLH+03] N. V. Chawla, A. Lazarevic, L. O. Hall, et al., “SMOTEBoost: improving prediction of the minority class in boosting,” *Proc. 7th Euro. Conf. Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, Croatia, 2003, pp. 107-119.

[CN06] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” *Proc. 23rd Int. Conf. Machine Learning*, Pittsburgh, PA, 2006, vol. 148, 161-168.

[Coh94] D. Cohn, “Neural network exploration using optimal experiment design,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 6, pp. 679-686, 1994.

[CS07] N. V. Chawla and J. Sylvester, “Exploiting diversity in ensembles: improving performance on unbalanced datasets,” *Proc. 7th Int. Workshop Multiple Classifier Systems (MCS)*, Prague, Czech Republic, 2007, pp. 397-406.

[CXH+06] H. Chen, Y. Xue, N. Huang, et al., “MeMo: a web tool for prediction of protein methylation modifications,” *Nucleic Acids Research*, vol. 34, pp. 249-253, 2006.

[DA06] R. Díaz-Uriarte and S. A. de Andrés, “Gene selection and classification of microarray data using random forest,” *BMC Bioinformatics*, vol. 7, no. 3, Jan. 2006. doi: 10.1186/1471-2105-7-3.

[Dav04] Ian Davidson, “An ensemble technique for stable learners with performance bounds,” *Proc. 19th Nat. Conf. Artificial Intell. (AAAI-2004)*, San Jose, CA, 2004, pp. 330-335.

[DFZ+08] Z. J. Ding, Y. Feng, Y. G. Zheng, et al., “Granular decision fusion systems for effective protein methylation predication,” *Proc. 2008 IEEE Symp. Computational Intelligence in Bioinformatics and Computational Biology*, Sun Valley, ID, 2008, pp. 214-218.

[DFZ+10] Z. J. Ding, Y. Feng, Y. G. Zheng, et al., "Protein methylation prediction using granular decision fusion methods," *Int. J. Soft Computing and Bioinformatics*, vol. 1, no.1, pp. 19-27, 2010.

[DG06] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," *Proc. 23rd Int. Conf. Machine learning*, Pittsburgh, PA, 2006, pp. 233-240.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification (2nd Ed.). New York: Wiley, 2001.

[Die00a] T. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139-157, 2000.

[Die00b] T. G. Dietterich, "Ensemble methods in machine learning," *Proc. 1st. Int. Workshop Multiple Classifier Systems (Springer LNCS-1857)*, Cagliari, Italy, 2000, pp. 1-15.

[Die02] T. G. Dietterich, "Ensemble learning," in The Handbook of Brain Theory and Neural Networks, M.A. Arbib, Ed. (2nd ed.), Cambridge, MA: The MIT Press, 2002, pp. 405-408.

[Dom97] P. Domingos, "Why does bagging work? A Bayesian account and its implications," *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining*, Newport Beach, CA, 1997, pp. 155-158.

[Dom99] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," *Proc. Int. Conf. Knowledge Discovery and Data Mining*, San Diego, CA, 1999, pp. 155-164.

[DZW+07] S. A. Danziger, J. Zeng, Y. Wang, et al., "Choosing where to look next in a mutation sequence space: Active Learning of informative p53 cancer rescue mutants," *Bioinformatics*, vol. 23, no. 13, pp. 104-114, 2007.

[DZX+09] Z. Ding, Y.-Q. Zhang, Nan Xie, et al. "Identifying new methylation arginine via granular decision fusion with SVM modeling," *Proc. 2009 Int. Joint Conf. Bioinformatics, Systems Biology and Intelligent Computing*, Shanghai, China, 2009, pp. 237-241.

[DZZ09] Z. Ding, Y.-Q. Zhang and Y. G. Zheng, “Feature selection and granular SVM classification for protein arginine methylation identification,” *Proc. 2009 IEEE Int. Conf. Syst., Man, and Cybern.*, San Antonio, TX, 2009, pp. 2979-2983.

[EHB+07] S. Ertekin, J. Huang, L. Bottou, et al., “Learning on the border: Active learning in imbalanced data classification,” *Proc. ACM 16th Conf. Information and Knowledge Management (CIKM 2007)*, Lisboa, Portugal, 2007, pp. 127-136.

[EHG07] S. Ertekin, J. Huang, and C. L. Giles, “Active learning for class imbalance problem,” *Proc. 30th Int. Conf. Research and Development in Information Retrieval (ACM SIGIR 2007)*, Amsterdam, Netherland, 2007, pp. 823-824.

[EHG07] S. Ertekin, J. Huang, and C. L. Giles, “Active learning for class imbalance problem,” *Proc. 30th Int. Conf. Research and Development in Information Retrieval (ACM SIGIR 2007)*, Amsterdam, Netherlands, 2007, pp. 823-824.

[Elk01] C. Elkan, “The foundations of cost-sensitive learning,” *Proc. 17th Int. Joint Conf. Artificial Intelligence*, Seattle, WA, 2001, pp. 973-978.

[FM99] Y. Freund and L. Mason, “The alternating decision tree algorithm,” *Proc. 16th Int. Conf. Machine Learning*, Bled, Slovenia, 1999, pp. 124-133.

[FS96] Y. Freund and R.E. Schapire, “Experiments with a new boosting algorithm,” *Proc. 13th Int. Conf. Machine Learning*, Bari, Italy, 1996, pp. 148-156.

[FSZ+99] W. Fan, S. J. Stolfo, J. Zhang, et al., “AdaCost: misclassification cost-sensitive boosting,” *Proc. Int. Conf. Machine Learning*, Bled, Slovenia, June, 1999, pp. 97-105.

[GG07] Y. Guo and R. Greiner, “Optimistic active learning using mutual information,” *Proc. Int. Joint Conf. on Artificial Intelligence (IJ-CAI)*, Hyderabad, India, 2007, pp. 823-829.

[GV04] H. Guo and H.L. Viktor, “Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach,” *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 30-39, 2004.

[GWB+02] I. Guyon, J. Weston, S. Barnhill, et al., “Gene selection for cancer classification using support vector machines,” *Machine Learning*, vol. 46, no. 1-3, pp. 389-422, 2002.

[HBG+08] H. He, Y. Bai, E.A. Garcia, et al., “ADASYN: adaptive synthetic sampling approach for imbalanced learning,” *Proc. Int. J. Conf. Neural Networks*, Hong Kong, China, 2008, pp. 1322-1328.

[HG09] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. pp. 91263-1284, Sep., 2009.

[HKT09] S. Hido, H. Kashima, and Y. Takahashi, “Roughly balanced bagging for imbalanced data,” *Statistical Analysis and Data Mining*, vol. 2, no. 5-6, pp. 412-426, 2009.

[HM05] M. A. Holbert and R. Marmorstein, “Structure and activity of enzymes that remove histone modifications,” *Current Opinion in Structural Biology*, vol. 15, no. 6, pp. 673-680, 2005.

[Ho98] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832-844, Aug., 1998.

[Hol87] R. Holliday, “The inheritance of epigenetic defects,” *Science*, vol. 238, no. 4824, pp. 163-170, 1987.

[HWM05] H. Han, W.Y. Wang, and B. H. Mao, “Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning,” *Proc. Int. Conf. Intelligent Computing*, Hefei, China, 2005, pp. 878-887.

[IL92] W. Iba and P. Langley, “Induction of one-level decision trees,” *Proc. 9th Int. Conf. Machine Learning, Scotland*, UK, 1992, pp. 233-240.

[Jap00] N. Japkowicz, Proc. Association for the Advancement of Artificial Intelligence (AAAI) 2000 workshop on learning from imbalanced data sets. AAAI Tech. Report WS-00-05, 2000.

[JJ04] T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 40-49, 2004.

[JS02] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *J. Intelligent Data Analysis*, vol. 6, no.5, pp. 429-450, Nov., 2002.

[KCA03] A. Kolez, A. Chowdhury, and J. Alspector, “Data duplication: An imbalance problem?,” *Proc. 20th Int. Conf. Machine Learning Workshop on Learning from Imbalanced Datasets*, Washington DC, 2003.

[KK98] M.Z. Kukar and I. Kononenko, “Cost-sensitive learning with neural networks,” *Proc. Euro. Conf. Artificial Intelligence*, Brighton, UK, 1998, pp. 445-449.

[KKP06] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, “Handling imbalanced datasets: A review,” *GESTS Int. Trans. Comput. Sci. and Eng.*, vol. 30, no. 1, pp. 25-36, 2006.

[KM97] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: One-sided selection,” *Proc. 14th Int. Conf. Machine Learning*, Nashville, TN, 1997, pp. 179-186.

[KPJ+02] H.-C. Kim, S. Pang, H.-M. Je, et al., “Pattern classification using support vector machine ensemble,” *Proc. IEEE 16th Int. Conf. Pattern Recognition (ICPR-2002)*, Quebec, Canada, 2002, vol. 2, pp. 160-163.

[Kri02] V. Krishnamurthy, “Algorithms for optimal scheduling and management of hidden Markov model sensors,” *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1382-1397, 2002.

[KS05] Y. Kato and H. Sasaki, “Imprinting and looping: epigenetic marks control interactions between regulatory elements,” *Bioessays.*, vol. 27, no. 1, pp. 1-4, 2005.

[KW03] L. Kuncheva and C. Whitaker, “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy,” *Machine* Learning, vol. 51, pp. 181-207, 2003.

[KWT+07] A.Y. Kovalevsky, Y.-F. Wang, Y. Tie, et al., “Atomic resolution crystal structures of HIV-1 protease and mutants V82A and I84V with saquinavir,” *Proteins*, vol. 67, pp. 232-242, 2007.

[LB92] K. Lang and E. Baum, “Query learning can work poorly when a human oracle is used,” *Proc. IEEE Int. Joint Conf. Neural Networks*, 1992, pp. 335-340.

[LG94] D. Lewis and W. Gale, “A sequential algorithm for training text classifiers,” *Proc. 17th Annual Int. ACM Conf. Research and Development in Information Retrieval (ACM-SIGIR)*, Dublin, Ireland, 1994, pp. 3-12.

[Li07] C. Li, “Classifying imbalanced data using a bagging ensemble variation (BEV),” *Proc. 45th Annu. ACM Southeast Regional Conf.*, Winston-Salem, NC, 2007, pp. 203-208.

[Liu04] Y. Liu, “Learning with support vector machine applied to gene expression data for cancer classification,” *J. Chemical Information and Computer Sciences*, vol. 44, pp.1936-1941, 2004.

[LML+08] G.-Z. Li, H.-H. Meng, W.-C. Lu, et al., “Asymmetric bagging and feature selection for activities prediction of drug molecules,” *BMC Bioinformatics*, vol. 9(Suppl. 6), S7, 2008. doi:10.1186/1471-2105-9-S6-S7.

[LMR04] M. Lindenbaum, S. Markovitch, and D. Rusakov, “Selective sampling for nearest neighbor classifiers,” *Machine Learning*, vol. 54, no.2, pp. 125-152, 2004.

[LOZ09] M. S. Lee, S. Oh, and B.-T. Zhang, “Ensemble learning based on active example selection for solving imbalanced data problem in biomedical data,” *Proc. 2009 IEEE Int. Conf. Bioinformatics and Biomedicine (BIBM 2009)*, Washington, DC, 2009, pp. 350-355.

[LTS+05] D. Y. Lee, C. Teyssier, B D. Strahl, et al., “Role of protein methylation in regulation of transcription,” *Endocrine Reviews*, vol. 26, no.2, pp. 147-170, 2005.

[LWZ06] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory under-sampling for class-imbalance learning,” *Proc. IEEE 6th Int. Conf. Data Mining*, Hong Kong, China, 2006, pp. 965 -969.

[LWZ09] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory under-sampling for class-imbalance learning," *IEEE Trans. Syst., Man, And Cybern.-Part(B): Cybern.*, vol. 39, no. 2, pp. 539-550, Apr., 2009.

[LZ06] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," *Proc. IEEE 6th Int. Conf. Data Mining*, Hong Kong, China, 2006, pp. 970-974.

[Mal03] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," *Proc. Int. Conf. Machine Learning (ICML) 2003 Workshop on Learning from Imbalanced Datasets*, Washington, DC, 2003.

[Mat75] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta*, vol. 405, pp. 442-451, 1975.

[McL04] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Hoboken, NJ: Wiley Interscience, 2004.

[Mel05] P. Melville, "Creating diverse ensemble classifiers to reduce supervision," Ph.D. Thesis, Dept. Comput. Sci., Univ. Texas, Austin, TX, Nov. 2005.

[Mit97] T. Mitchell. *Machine Learning*, New York: McGraw Hill, 1997.

[MM04] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *J. Information Fusion (Special Issue on Diversity in Multiple Classifier Systems)*, vol. 6, no. 1, pp. 99-111, 2004.

[MZW05] K. McCarthy, B. Zabar, and G.M. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes," *Proc. 1st Int. Workshop Utility-Based Data Mining*, Chicago, IL, 2005, pp. 69-77.

[NBP09] G. H. Nguyen, A. Bouzerdoum and S. L. Phung, "Learning pattern classification tasks with imbalanced data sets," in *Pattern Recognition*, P.-Y. Yin Ed. Rijeka, Croatia: InTech, 2009, pp. 193-208.

[NIH97] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Network: Computation in Neural Systems*, vol. 8, no. 3, pp. 283-296, 1997.

[Ols08] F. Olsson, "Bootstrapping named entity annotation by means of active machine learning," Ph.D. dissertation, Dept. Swedish Language, Univ. Gothenburg, Kista, Sweden, 2008.

[OLZ11] S. Oh, M. S. Lee, and B.-T. Zhang, "Ensemble learning with active example selection for imbalanced biomedical data classification," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 8, no. 2, pp. 316-325, 2011.

[OM99] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artificial Intell. Research*, vol. 11, pp. 169-198, 1999.

[PD07] P. Panov and S. Džeroski, "Combining bagging and random subspaces to create better ensembles," *Advances in Intelligent Data Analysis VII* (Springer, LNCS 4723), pp. 118-129, 2007.

[PK67] W. K. Paik and S. Kim, "Enzymatic methylation of protein pabp1 identified as an arginine fractions from calf thymus nuclei," *Biochemical and Biophysical Research Commun.*, vol. 29, pp. 14-20, 1967.

[Pol06] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Syst. Mag.*, vol. 6, pp. 21–45, 2006.

[PR04] G. A. Petsko, D. Ringe. *Protein structure and function (Primers in Biology)*, Sunderland, MA: New Science Press, 2004.

[PS81] Z. Pawlak and N. Shan, "Information systems: Theoretical foundations," *Information Systems*, vol. 6, no. 3, pp. 205-218, 1981.

[PTW+05] D. Plewczynski, A. Tkacz., L. Wyrwicz, et al., "AutoMotif server: prediction of single residue post-translational modifications in proteins," *Bioinformatics*, vol. 21, no. 10, pp. 2525-2527, 2005.

[Qui86] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.

[Qui93] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[Qui96] J. R. Quinlan, “Bagging, boosting, and C4.5,” *Proc. 13th Nat. Conf. Artificial Intelligence*, Portland, OR, 1996, pp. 725-730.

[RJ00] K. D. Robertson and P. A. Jones, “DNA methylation: past, present and future directions,” *Carcinogenesis*, vol. 21, no. 3, pp. 461-467, 2000.

[RK04] B. Raskutti and A. Kowalczyk, “Extreme re-balancing for SVMs: A case study,” *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 60-69, 2004.

[RM01] N. Roy and A. McCallum, “Toward optimal active learning through sampling estimation of error reduction,” *Proc. 18th Int. Conf. Machine Learning*, Williamstown, MA, 2001, pp. 441-448.

[Sch99] R. E. Schapire, “A brief introduction to boosting,” *Proc. 16th Int. Joint Conf. Artificial Intell.*, Orlando, FL, 1999, vol. 2, pp. 1401-1406.

[SCR08] B. Settles, M. Craven, and S. Ray, “Multiple-instance active learning,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 20, pp. 1289-1296, 2008.

[Set08] B. Settles, “Curious machines: Active learning with structured instances,” Ph.D. dissertation, Dept. Comput. Sci., Univ. of Wisconsin–Madison, Madison, Wisconsin, 2008.

[Set09] B. Settles, “Active learning literature survey,” Comput. Sci. Dept., Univ. of Wisconsin–Madison, Madison, Comput. Sci. Tech. Rep. 1648, 2009.

[SH07] C. T. Su and Y. H. Hsiao, “An Evaluation of the robustness of MTS for imbalanced data,” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 10, pp. 1321-1332, Oct., 2007.

[SKH+10] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, et al., “RUSBoost: A hybrid approach to alleviating class imbalance,” *IEEE Transactions Syst., Man, And Cybern. -Part A: Syst. and Humans*, vol. 40, no. 1, pp. 185-197, Jan., 2010.

[SKW+07] Y. Sun, M.S. Kamel, A.K.C. Wong, et al., “Cost-sensitive boosting for classification of imbalanced data,” *Pattern Recognition*, vol. 40, no. 12, pp. 3358-3378, 2007.

[SLC+09] D. M. Shien, T. Y. Lee, W. C. Chang, et al., "Incorporating structural characteristics for identification of protein methylation sites," *J. Computational Chemistry*, vol. 30, no. 9, pp. 1532-1543, 2009.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," *Proc. ACM 5th Annu. Workshop on Computational Learning Theory*, Pittsburgh, PA, 1992, pp. 287-294.

[Sun07] Y. Sun, "Cost-sensitive boosting for classification of imbalanced data," Ph.D. Dissertation, Dept. Electrical and Comput. Eng., Univ. of Waterloo, Ontario, Canada, 2007.

[SXT+09] J. Shao, D. Xu, S.-N. Tsai, et al., "Computational identification of protein methylation sites through bi-profile bayes feature extraction," *PLoS ONE*, vol. 4, no. 3, pp. e4920, 2009. doi:10.1371/journal.pone.0004920

[Tay86] W. R. Taylor, "The Classification of amino acid conservation," *J. Theoretical Biology*, vol. 119, pp. 205-218, 1986.

[TC01] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," *Proc. 9th ACM Int. Conf. Multimedia*, Ontario, Canada, 2001, pp. 107-118.

[Tch05] N. A. Tchurikov, "Molecular mechanisms of epigenetics," *Biochemistry*, vol. 70, no. 4, pp. 406-423, 2005.

[TK00] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Proc. 17th Int. Conf. Machine Learning*, Stanford, CA, 2000, pp. 999-1006.

[Ton01] S. Tong, "Active learning: Theory and applications," Ph.D. dissertation, Dept. of Comput. Sci., Stanford Univ., Stanford, CA, 2001.

[TSK06] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, MA: Pearson Addison Wesley, 2006.

[TZ06] Y. Tang and Y.Q. Zhang, "Granular SVM with repetitive undersampling for highly imbalanced protein homology prediction," Proc. *2006 IEEE Int. Conf. Granular Computing*, Atlanta, GA, 2006, pp. 457-460.

[TZC+09] Y.C. Tang, Y.-Q. Zhang, N. V. Chawla, et al., "SVMs modeling for highly imbalanced classification," *IEEE Trans. Syst., Man, and Cybern. - Part B*, vol. 39, no. 1, pp. 281 - 288, Feb. 2009.

[TZH07] Y. Tang, Y.-Q. Zhang, and Z. Huang, "Development of two-stage SVM-RFE gene selection strategy for microarray expression data analysis," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 365-381, July, 2007.

[van79] C. J. van Rijsbergen, *Information Retrieval* (2nd ed.). Newton, MA: Butterworth-Heinemann, 1979.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[Wal05] C. Walsh, *Posttranslational modification of proteins: expanding nature's inventory*. Greenwood Village, CO: Roberts & Company Publishers, 2005.

[WC04] G. Wu and E. Y. Chang, "Aligning boundary in kernel space for learning imbalanced data set," *Proc. 4th IEEE Int. Conf. Data Mining*, Brighton, UK, 2004, pp. 265-272.

[WCY10] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," *2010 Int. Joint Conf. Neural Networks*, Barcelona, Spain, 2010, pp. 1-8.

[Web00] G. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, no.2, pp. 159-196, 2000.

[WM01] C. Wu and J. R. Morris, "Genes, genetics, and epigenetics: a correspondence," *Science*, vol. 293, no. 5532, pp. 1103-1105, 2001.

[WY09a] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," *Proc. IEEE Symp. Comput. Intell. and Data Mining (CIDM-2009)*, Nashville, TN, 2009, pp. 324-331.

[WZZ+08] W. L. Wooderchak, T. Zang, Z. S. Zhou, et al., "Substrate profiling of PRMT1 reveals amino acid sequences that extend beyond the 'RGG' paradigm," *Biochemistry*, vol. 47, pp. 9456-9466, 2008.

[YJ06] C. B. Yoo and P. A. Jones, “Epigenetic therapy of cancer: past, present and future,” *Nature Reviews Drug Discovery*, vol. 5, no. 1, pp. 37-50, 2006.

[YL09] S.-J. Yen and Y.-S. Lee, “Cluster-based under-sampling approaches for imbalanced data distributions,” *An Int. J. Expert Systems with Applications*, vol. 36, no. 3, pp. 5718-5727, 2009.

[YLJ+03] R. Yan, Y. Liu, R. Jin, et al., “On Predicting rare classes with SVM ensembles in scene classification,” *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP 2003)*, Hong Kong, China, 2003, pp. 21-24.

[Yu05] H. Yu, “SVM selective sampling for ranking with application to data retrieval,” *Proc. 11th ACM Int. Conf. Knowledge Discovery and Data Mining (ACM-SIGKDD)*, Chicago, IL, 2005, pp. 354-363.

[YYH03] R. Yan, J. Yang, and A. Hauptmann, “Automatically labeling video data using multi-class active learning,” *Proc. 9th IEEE Int. Conf. Computer Vision (ICCV 2003)*, Nice, France, 2003, pp. 516-523.

[YYZ05] J.T. Yao, Y.Y. Yao, and Y. Zhao, “Foundations of classification”, in T. Y. Lin, S. Ohsuga, C. J. Liau, et al. (Eds.), *Foundations and Novel Approaches in Data Mining*, Berlin: Springer, 2005, pp. 75-97.

[ZD06] L. Zhuang and H. Dai, “Parameter estimation of one-class SVM on imbalance text classification,” *Lecture Notes in Artificial Intelligence (LNAI-4013)*, vol. 4013, pp. 538-549, 2006.

[ZG09] Z.-Q. Zeng and J. Gao, “Improving SVM classification with imbalance data set,” *Proc. 16th Int. Conf. Neural Information Processing (ICoNIP 2009)*, Bangkok, Thailand, 2009, pp. 389-398.

[ZH07] J. Zhu and E. H. Hovy, “Active learning for word sense disambiguation with methods for addressing the class imbalance problem,” *Proc. 2007 Joint Conf. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic, 2007, pp. 783-790.

[Zhu08] M. Zhu, “Kernels and ensembles,” American Statistician, vol. 62, no.2, pp. 97-109, 2008.

[ZL06] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63 – 77, 2006.

[ZS96] W. Ziarko and N. Shan, “A method for computing all maximally general rules in attribute-value systems,” *Computational Intell.*, vol. 12, no. 2, pp. 223-234, 1996.