2010

A Selective Sampling Method for Imbalanced Data Learning on Support Vector Machines

Jong Myong Choi
Iowa State University

A selective sampling method for imbalanced data learning on support vector machines

by

Jong Myong Choi

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
John K. Jackman, Major Professor
Sigurdur Olafsson
Douglas D. Gemmill
Dianne H. Cook
Anthony M. Townsend

Iowa State University

Ames, Iowa

2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

It is a pleasure to thank many people who helped me conduct research toward a Ph.D. and write this dissertation. This work would not have been finished without their support and patience.

First, I would like to heartily express the deepest appreciation to my adviser, Dr. John Jackman for his encouraging way to help me whenever I am in a trouble during my graduate school years. His persistent guidance enabled me to complete this work. I am also thankful to my committee members, Dr. Sigurdur Olafsson, Dr. Doug Gemmill, Dr. Anthony Townsend and Dr. Dianne Cook whose valuable and helpful comments improved this work.

I also would like to thank my family members: my parents for educating me with unconditional support to finish my study. Especially, my special thanks goes to my wife, In Suk Lee who always understands and encourages me with her support, patience and endless love throughout my life.

ABSTRACT

The class imbalance problem in classification has been recognized as a significant research problem in recent years and a number of methods have been introduced to improve classification results. Rebalancing class distributions (such as over-sampling or under-sampling of learning datasets) has been popular due to its ease of implementation and relatively good performance. For the Support Vector Machine (SVM) classification algorithm, research efforts have focused on reducing the size of learning sets because of the algorithm's sensitivity to the size of the dataset. In this dissertation, we propose a metaheuristic approach (Genetic Algorithm) for under-sampling of an imbalanced dataset in the context of a SVM classifier. The goal of this approach is to find an optimal learning set from imbalanced datasets without empirical studies that are normally required to find an optimal class distribution. Experimental results using real datasets indicate that this metaheuristic under-sampling performed well in rebalancing class distributions. Furthermore, an iterative sampling methodology was used to produce smaller learning sets by removing redundant instances. It incorporates informative and the representative under-sampling mechanisms to speed up the learning procedure for imbalanced data learning with a SVM. When compared with existing rebalancing methods and the metaheuristic approach to under-sampling, this iterative methodology not only provides good performance but also enables a SVM classifier to learn using very small learning sets for imbalanced data learning. For large-scale imbalanced datasets, this methodology provides an efficient and effective solution for imbalanced data learning with an SVM.

CHAPTER 1 INTRODUCTION

The imbalanced learning problem in data mining has attracted a significant amount of interest from the research community and practitioners because real-world datasets are frequently imbalanced, having a minority class with relatively few instances when compared to the other classes in the dataset. Standard classification algorithms used in supervised learning have difficulties in correctly classifying the minority class. Most of these algorithms assume a balanced distribution of classes and equal misclassification costs for each class. In addition, these algorithms are designed to generalize from sample data and output the simplest hypothesis that best fits the data. This principle is embedded in the inductive bias of many machine learning algorithms including Decision Tree, nearest neighbor, and Support Vector Machine (SVM). Therefore, when they are used on complex imbalanced data sets, these algorithms are inclined to be overwhelmed by the majority class and ignore the minority class causing errors in classification for the minority class. In other words, standard classification algorithms try to minimize the overall classification error rate by producing a biased hypothesis which regards almost all instances as the majority class.

Recent research on the class imbalance problem has included studies on datasets from a wide variety of contexts such as, information retrieval and filtering (Lewis & Catlett, 1994), diagnosis of rare thyroid disease (Murphy & Aha, 1994), text classification (Chawla et al., 2002), credit card fraud detection (Wu & Chang, 2003) and detection of oil spills from satellite images (Kubat et al., 1998). The degree of imbalance varies depending on the context. In intrusion detection, typically less than 10% of the data are actual intrusions. In detection of cancerous cells, less than 1% of cells are actually cancerous.

To illustrate the imbalance problem, consider the "Mammography Data Set", which has been used frequently to study the class imbalance learning problem. This data is a collection of images obtained from a series of mammography exams conducted on a set of distinct patients. Analyzing the images in the two classes, "cancerous" and "noncancerous" patient, it is observed that the number of noncancerous patients greatly exceeds the number of cancerous patients. Indeed, this data set contains 10,923 "Negative" (major class) samples and 260 "positive" (minority class) samples. Ideally, a classifier should classify both classes with almost 100% accuracy. However, classifiers tend to produce severely biased classification with the majority class almost 100% accuracy and conversely the minority class having accuracies of less than 0.5% accuracy. As a result, most cancerous patients are classified as noncancerous (i.e., a Type I error). In the classification of diagnosing patients, such a consequence would be extremely costly because treatment would not be initiated. For these imbalanced scenarios, classifiers should provide much higher accuracy for the minority class without a significant loss in accuracy for the majority class.

New classification methods are needed to address the class imbalance problem in supervised learning. In this research, we propose a new methodology for imbalanced data learning based on the SVM classification algorithm. The remainder of the dissertation is organized as follows. In Chapter 2, we review general approaches for imbalanced data learning and related studies and describe the scope of this research. Chapter 3 briefly describes the SVM classification algorithm and the causes of imbalanced learning with the SVM classifier. This is followed by a discussion of existing methods that have been used for the class imbalance problem based on the SVM algorithm. At the end of this chapter, the approach used in the new methodology is described. In Chapter 4, we address a critical issue in SVM learning, namely,

'*large-scale data'*, and an optimization based under-sampling method using Genetic Algorithm is introduced and classification results are compared with other sampling methods using real datasets. In Chapter 5, the new methodology that solves the class imbalance problem on SVM with relatively small learning sets for SVM classifier is described. Finally we conclude with suggestions on future research directions in Chapter 6.

CHAPTER 2 LITERATURE REVIEW

In general, a class imbalance problem is seen in two situations namely, natural imbalance or rarity of cases (i.e., instances or samples). Underlying reasons for imbalance could be the lack of occurrences in nature for a specific phenomena or possibly insufficient funds or time to collect sufficient data. In recent years, many researchers have studied the class imbalance problem. Weiss (2004) presented an overview of the field of learning from imbalanced datasets. His work particularly focused on the problems with identifying rare objects in data mining by defining two types of rarity: rare classes and rare cases. A rare class contains relatively smaller instances than other classes, while a rare case indicates a small subset of the data (instance) space. Unsupervised learning algorithms such as clustering may help to identify a rare case. More generally, class imbalance is related to rare classes and is associated with classification problems. In his work, Weiss argued that typical evaluation metrics do not adequately describe the value of rarity so that data mining is not likely to handle rare classes and rare cases. Monard et al. (2002) discussed several issues related to learning with skewed class distributions, such as the relationship between cost-sensitive learning and class distributions, and the limitations of accuracy and error rate in measuring the performance of classifiers.

2.1 Handling the Class Imbalance Problem

The various approaches used to deal with the class imbalance problem can be grouped into three categories: (1) changing class distributions (modifying the data itself to rebalance skewed datasets at the data level), (2) adjustment of classifiers (adjust standard classification

algorithms to imbalanced data sets by applying cost or weight for misclassified cases), and (3) ensemble learning methods (using a combination of multiple classifiers with multiple datasets).

2.1.1 Changing class distributions

Changing class distributions is performed at the data level in order to modify class distribution in the training datasets. Since many more instances belong to the majority class than the minority class, class distribution can be balanced by under-sampling the majority class, over-sampling the minority class, combining under-sampling and over-sampling, or some other sampling method. Studies have shown that a balanced data set provides improved classification performance as compared with an imbalanced data set. There have been numerous studies on changing class distribution (Laurikkala, 2001 and Estabrooks et al., 2004). Also, Weiss (2003) investigated the effect of class distribution on decision tree classification by changing class distributions to achieve different ratios and measuring performance using accuracy and Area Under the Curve (AUC). Three basic techniques are used in balancing classes namely, heuristic and non-heuristic under-sampling, heuristic and non-heuristic over-sampling, and advanced sampling. Japkowicz (2000) compared multiple balancing methods and concluded that both under-sampling and over-sampling are very effective methods for dealing with the class imbalance problem.

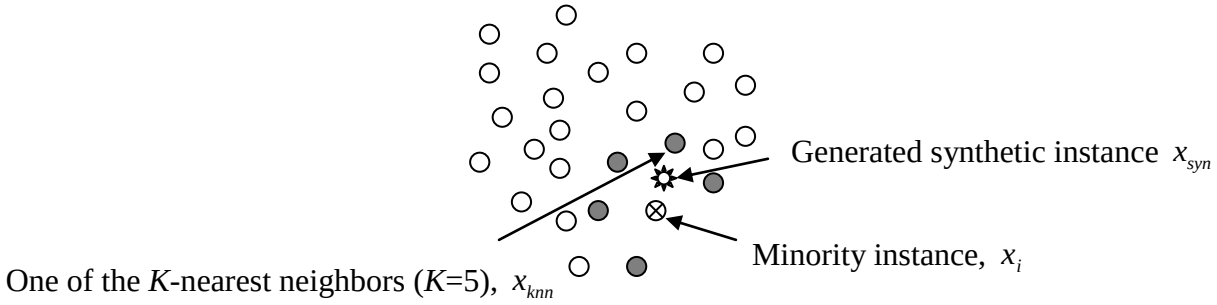
Over-sampling

One simple over-sampling method is random over-sampling. Its mechanism is adding a set *E* of additional instances (i.e., instance duplicates) randomly sampled from the minority class to the original set, *S.* In this way, the number of total instances of the minority classis increased

by *E* and as a result, the class distribution is more balanced. This provides a mechanism for varying the degree of class distribution balance to any desired balance level. Over-sampling does not increase information; instead by replication it raises the weight of the minority samples. The problem with over-sampling is that an over-fitting problem will generally occur, which causes the classification rule to become too specific; even though the accuracy for training set is high, the classification performance for new test datasets will likely be worse. By appending duplicated data to the original data set, some of the data copied becomes too specific and classifiers will produce multiple clauses for the duplicate data (Kubat and Martin, 1997).

To avoid the over-fitting problem in over-sampling, Chawla et al. (2002) suggested a heuristic over-sampling method, called Synthetic Minority Over-sampling Technique (SMOTE), which has worked well in various applications. SMOTE is considered to be one of the state-of-the-art approaches for imbalanced learning. This method generates synthetic data based on the feature space similarities between existing minority instances considering the *K*-nearest neighbors for each minority instance. In order to create a synthetic instance, it finds the *K*-nearest neighbors of each minority instance, randomly selects one of them, and then multiplies the corresponding feature vector difference with a random number between 0 and 1 to produce a new minority instance in the neighborhood. Figure 1 shows an example of the SMOTE procedure. This synthetic over-sampling avoids the over-fitting problem and also causes the decision boundaries for the minority class to move towards the majority class. As a variant of SMOTE, Han et al. (2005) introduced Borderline_SMOTE which only oversamples synthetic instances of the minority class near the decision boundary since those instances are most likely to be misclassified. Results were better when compared to standard SMOTE and random over-sampling using Decision Tree classification. He et al. (2008) introduced a synthetic sampling

method, Adaptive Synthetic Sampling (ADASYN), that uses a density distribution of the minority instance as a criterion to automatically decide the number of synthetic samples generated for each minority instance. ADASYN generates a new instance by calculating the class ratio of the minority and majority instances in the *K*-nearest neighbors of each minority instance. As a result, more synthetic instances are generated for minority class instances that are harder to learn compared to instances that are easier to learn. This approach improved learning with respect to the data distributions on the imbalanced data sets by reducing the bias of class distribution and by adaptively shifting the decision boundary to put more attention on instances difficult to learn.



$$x_{syn} = x_i + (x_{knn} - x_i) \times \alpha$$

α is a random number between 0 and 1

Figure 2.1 Synthetic over-sampling example by SMOTE algorithm

Under-sampling

While over-sampling adds instances to the original data set, under-sampling removes instances from the majority class while keeping all instances of the minority class due to rareness

of information. A simple method for under-sampling the majority class is random under-sampling, a non-heuristic method that balances class distributions by selecting and removing majority instances randomly. Several heuristic under-sampling methods have been proposed from data cleaning in recent years. They are based on either of two different noise model hypotheses: one is that instances near to a decision boundary between two classes are considered noise, while the other considers that instances having more neighbors from different classes are noise.

Since random under-sampling leads to losing potentially useful data, some heuristic under-sampling methods try to remove superfluous instances which will not affect the classification accuracy of the training set. Hart (1968) introduced a training set condensation algorithm, Condensed Nearest Neighbor Rule (CNN), in order to find a consistent subset of a sample set which can correctly classify all of the remaining instances in the training set. The algorithm uses two bins, called *S* and *T*. Initially, the first sample of the training set is placed in *S*, while the remaining samples of the training set are placed in *T*. Then one pass through T is performed. During the scan, whenever a point in *T* is misclassified by using *S* as the training set, it is transferred from *T* to *S*. After classification, process is repeated until no points are transferred from *T* to *S*. The motivation for this heuristic is that misclassified data lies close to the decision boundary. In the same manner, Tomek (1976) proposed an effective method to eliminate data in the overlapping regions. Given two instances *x* and *y* that have a different class label and are separated by a distance $d(x, y)$, the pair (*x*, *y*) is called a *Tomek link* if there is no instance *z* such that $d(x,z) < d(x,y)$ or $d(y,z) < d(x,y)$. Instances participating in *Tomek links* are considered either borderline or noisy. Kubat and Matin (1999) proposed one-sided sampling (OSS) for detecting less relevant instances for learning. This technique is intended to keep all

minority instances since they are rare, (even though some of them can be noisy) and instead prune out only majority instances. Initially it starts with a subset(C) of training set (S) that contains all minority instances, $C \subseteq S$, and using a 1-Nearest Neighbor rule using instances in C, classify the instances in S. Afterwards, all misclassified instances are moved to C, then all majority instances participating in *Tomek links* from C are removed since they are believed to be borderline and/or noisy. Wilson (1972) introduced the Edited Nearest Neighbor Rule (ENN) to remove any instance whose class label is different from the class of at least two of its three nearest neighbors. The idea behind this technique is to remove the instances from the majority class that are near or around the borderline of different classes based on the concept of nearest neighbor (NN) in order to increase classification accuracy of minority instances rather than majority instances.

2.1.2 Adjusting classifiers to imbalanced data sets

Rebalancing the data distribution through either over-sampling or under-sampling has had some success, but the methods are usually computationally expensive. Also, changing class distribution at the data level does not always lead to better classification performance. A classifier is not always influenced by class distributions. Drummond and Holte (2003) observed that over-sampling did not produce effective improvement in performance or there was no change in classification. On the contrary, over-sampling prunes less than under-sampling using the default parameters for the C4.5 algorithm. A modification of the parameter settings of C4.5 improved classification performance and avoided the over-fitting problem during over-sampling. Thus, while sampling methods have tried to balance class distribution by considering the proportions of class instances in the original data distribution, other approaches have been

introduced for imbalanced data learning. One is the cost sensitive method, which uses a cost matrix to penalize misclassification of instances, as shown in Table 2.1. Typically, no costs are applied to the correctly classified cases and the cost of misclassifying minority cases is higher than that of majority cases. The objective of this strategy is to minimize the cost of misclassification. In some applications, cost sensitive techniques have performed better than sampling methods (McCarthy et al., 2005 and Liu et al., 2006).

Actual	Predicted	
	Class *i*	Class *j*
Class *i*	0	c_{ij}
Class *j*	c_{ji}	0

Table 2.1 Cost matrix

MetaCost (Domingos, 1999) is another method related to cost-sensitive learning. It estimates class probabilities using *Bagging* and then re-labels the training instances with their minimum expected classes, and in the end, relearns a model using the modified training set. Based on the weight update rule of AdaBoost (Freund & Schapire, 1997) for misclassified instances at an iterative learning, Fan et al. (1999) proposed a discriminant weight update method for misclassified instances for imbalanced datasets, which is called Adacost. Their approach is to assign larger weights for misclassified instances belonging to the minority class than those belonging to the majority class and as a result, Adacost has performed empirically better in lowering cumulative misclassification costs than AdaBoost.

Some classifiers such as the Naïve Bayes classifier or some Neural Networks use a score to show the degree to which an instance belongs to a class. This type of ranking can be used in alternative classifiers by changing the threshold for an instance belonging to a class (Weiss, 2004). For biasing the discrimination procedure, Barandela et al. (2003) proposed a weighted distance function in classification instead of altering the class distributions in terms of a nearest neighbor (NN) classifier. Supposing that $d_e(\cdot)$ is the Euclidean metric, x_{new} a new instance to classify, x_0 a training sample from class i, n_i the number of instances of class i and m the dimensionality of the input variable, a weighted distance function, $d_w(\cdot)$ is defined as: $d_w(x_{new}, x_0) = (n_i / n)^{1/m} \times d_e(x_{new}, x_0)$. This could assign greater weighting factors to majority instances than minority instances; consequently, producing smaller distances to instances of the minority class than distances to those of the majority class. As a result, the neighbors of the new instances are found among the minority instances, increasing the value of the geometric mean (g-mean). For the SVM classification algorithm, this biasing approach pushes a hyperplane further away from the minority (positive) class for imbalanced datasets. Wu and Chang (2003) proposed a biasing algorithm to change the kernel function. Biasing classification algorithms in SVM using larger penalty constants associated with the minority class made misclassification errors for the minority instances much costlier than errors for majority instances (Veropoulos et al., 1999). Huang at el. (2004) proposed a Biased Minimax Probability Machine (BMPM) to resolve learning for imbalanced datasets. Given the mean and covariance matrices of the majority and minority classes, BMPM formulates an optimization problem to find the decision hyperplane by adjusting the lower bound of the accuracy for the classification of the future data. For example, if the objective function is to maximize the accuracy of classification for the minority class, the

optimization tries to maximize it by setting the lower bound of the classification accuracy for both classes. Achieving the worst case accuracy for the minority can be avoided while maintaining the acceptable accuracy level of the majority class in imbalanced data learning.

One-class learning is an alternative to discrimination where the model is created based on the instances of the target class alone. The main idea is that boundaries between two classes are estimated from data of one class (the target class) so that this approach is not sensitive to the class distribution in the training set. A boundary around the target class is defined in such a way that most of the target objects are included and at the same time the chance of accepting outlier objects is minimized. For instance, Kubat et al. (1998) introduced the SHRINK algorithm following this general principle and applied it to detecting rare oil spills from satellite radar images. The goal was to find the classification rule that best identifies the positive examples (oil spills) using a g-mean measure. Assuming that the negative (majority) instances outnumber positive (minority) instances, the algorithm labeled the mixed regions as positive (minority). This alters the learner's focus: search for *the best positive region*, one with the maximum ratio of positives to negatives. Raskutti and Kowalczyk (2004) studied one-class learning with highly imbalanced dataset learning using a SVM classifier. They showed that one-class learning is useful for extremely imbalanced datasets with a high dimensional noisy feature space.

2.1.2 Ensemble learning methods

Ensemble learning is motivated by the information loss that occurs in under-sampling. In ensemble learning, multiple classifiers are generated by training subsets from the original dataset. In the end, the classifiers are combined in a learning process and the final classification is determined by a voting scheme. Boosting and Bagging (Boostrap aggregating) are the most

successful approaches. Most boosting algorithms use iteratively learning weak classifiers that have been produced by placing different weights on the training instances. In each iteration, boosting increases weights for incorrectly classified instances and decreases weights for correctly classified ones, placing more attention on the incorrectly classified instances for the next iteration. Rare-Boost scales false-positive instances in proportion to how well they are differentiated from true-positive instances and scales false-positive instances in proportion to how well they are distinguished from true-negative instances (Joshi et al., 2001). SMOTEBoost (Chawla et al., 2003) addressed the issue that boosting may produce an over-fitting problem as in over-sampling. Instead of updating weights to change distributions of the training dataset, it adds new instances of the minority class using SMOTE. Chan and Stolfo (2001) proposed another ensemble method conceptually similar to a Bagging approach. They conducted some preliminary experiments to identify a desired class distribution that avoids the class imbalance problem, and then resampled to make multiple training sets based on the desired class distribution. Each training set contained all instances of the minority class and a subset of the majority instances. To use all instances of the majority class, each majority class instance appeared in at least one training set. Finally, the learning algorithm was applied to each training set and a composite learner was created from the classification results of all classifiers.

Recently, two algorithms, EasyEnsemble and BalanceCascade have been introduced (Liu et al., 2006). The strategy of these two methods is to make several training sets by keeping all the minority instances and under-sampling several subsets from the majority class. With replacement in sampling of the majority class, they overcome potential of information loss of the majority class. EasyEnsemble independently samples (with replacement) from the majority class several subsets whose size is equal to the size of the minority class and generates the individual

classifiers for the subsets. In other words, EasyEnsemble generates T balanced training sets. The output of learning the i^{th} training set is an AdaBoost classifier H_i (i = 1,..,T). Then all generated classifiers, $H_{i=1,\ldots,T}$, are combined for the final decision. The BalanceCascade method reduces the size of the majority class iteratively, based on the most recent classifier. This algorithm uses a trained classifier to guide the sampling process for subsequent classifiers. Initially, it samples a balanced training set like EasyEnsemble. After the Adaboost ensemble is trained with the initial balanced training set, all majority instances that have been correctly classified are removed from the majority class. In this manner, the majority training set is reduced after every AdaBoost ensemble, H_i, is trained. This sampling strategy reduces the redundant information of the majority class and explores as much useful information as possible.

Besides the methods already discussed, other approaches have been used to address the class imbalance problem. For example, feature selection was used to select important features for the minority and majority classes separately and then explicitly combine them (Zheng et al. 2004).

2.2 Performance Measures for Imbalanced Data Learning

Performance measures are used to assess the effectiveness of learning methods. In general, accuracy (or error rate) is the most common metric for most classification tasks and is given by

$$Accuracy = \frac{(TP + TN)}{(TP + FN + FP + TN)} \quad . \tag{2.1}$$

For a two class classification problem, classification performance is evaluated by a confusion matrix (contingency table) as seen in Table 2.2.

However, for a skewed class distribution, accuracy is not suitable to evaluate imbalanced data learning because the overall accuracy may be dominated by the classification accuracy of the majority class. For this reason, other metrics have been used, namely, precision, recall, F-measure, geometric mean (g-mean) of the accuracy on the majority class and the minority class, and the maximum sum (MS). These metrics are based on the confusion matrix (see Table 2.2). In this research, positive and negative correspond to the minority and majority class, respectively.

		Prediction	
		Positive	Negative
Real	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

Table 2.2 Confusion matrix for performance evaluation

Intuitively, precision is a measure of how many instances were correctly labeled as positive and is calculated as

$$precision = \frac{TP}{(TP + FP)}. \tag{2.2}$$

Recall is a measure of how many instances of the positive class were labeled correctly and is defined as

$$recall = \frac{TP}{(TP + FN)}. \tag{2.3}$$

Unlike accuracy and error, precision and recall are both less sensitive to changes in data distributions. As an assessment of the accuracy for the positive class, precision is somewhat sensitive to data distributions, while recall is not. Recall gives no insight into how many

instances are incorrectly classified as positive. Similarly, precision does not tell us how many positive instances are incorrectly classified. Nevertheless, when used properly, precision and recall can effectively evaluate classification performance in imbalanced learning scenarios.

The *F-measure* metric combines precision and recall as a measure of the effectiveness of classification in terms of a ratio of the weighted importance on either recall or precision as determined by the β coefficient set by the user and is given by

$$F-measure = \frac{(1+\beta^2)\times recall \times precision}{\beta^2 \times recall + precision} \quad . \tag{2.4}$$

where β is a coefficient to adjust the relative importance of *precision* versus *recall.* As a result, F-measure provides more insight into the functionality of a classifier than the accuracy metric.

Another metric, the *g-mean* evaluates the degree of inductive bias in terms of a ratio of positive accuracy and negative accuracy and is defined as

$$g-mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \quad . \tag{2.5}$$

Maximum sum (*MS*) is used as an evaluation metric that gives equal weight to the classification accuracy of the positive and the negative class and is given by

$$MS = \frac{TP}{TP+FN} + \frac{TN}{TN+FP}. \tag{2.6}$$

Receiver Operating Characteristic (ROC) analysis from signal detection theory is also used as a metric for imbalanced data learning. The area under the ROC curve (AUC) assesses overall classification performance (Bradley, 1997). AUC does not place more emphasis on one class over the other, so it is not biased against the minority class. In addition, Precision-Recall

(PR) curves (Davis & Goadrich, 2006) and Cost curves (Holte & Drummond, 2006) have been used in other to evaluate imbalanced dataset learning of classifiers and also visualize the performance.

2.3 Summary and Research Scope

Although previous methods have in some cases produced satisfactory results for imbalanced data learning, some of them may not be practical to implement or may conflict with a specific classification learning algorithm. Imbalanced data learning methods need to consider performance and interaction with classification algorithms.

In this research, we examine the class imbalance problem focusing on a specific classification algorithm, SVM and a comparison of methodologies from a perspective of *effectiveness* (the ability to accurately classify an unknown dataset) and *efficiency* (the speed of classifying data). Although SVM is more accurate on moderately imbalanced data compared with other standard classifiers, an SVM is also generally prone to generate a classifier that is extremely biased toward the majority class. To cope with the imbalance dataset learning with a SVM, the previously discussed sampling strategies could be used, but would introduce significant degradation in classifier performance. For example, while over-sampling keeps all existing information of a learning dataset and solves the class imbalance problem by adding information of the minority class, processing of learning training datasets could be costly if many instances are over-sampled to handle imbalanced datasets.

Recent work on imbalanced learning with a SVM has focused on improving classification performance (Akbani et al., 2004, Raskutti & Kowalczyk, 2004). However, the

efficiency of imbalanced data learning was not fully considered. In this research, we present a sampling methodology for the problem of class imbalance considering both effectiveness and efficiency in learning with a SVM. In this research, the base assumption is that classification accuracies of both classes are equally important.

CHAPTER 3 CLASS IMBALANCE PROBLEM WITH SUPPORT VECTOR MACHINE LEARNING

3.1 Support Vector Machine (SVM) Classifier

A SVM uses a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. This learning strategy was introduced by Vapnik (1995) and has been widely used in the machine learning community due to its theoretical foundations and practical performance in applications ranging from image retrieval (Tong & Chang, 2001), handwriting recognition (Cortes, 1995) to text classification (Joachims, 1998). In classification tasks, SVM tries to find an efficient way of learning good separating hyperplanes in a high dimensional feature space that maximizes the margin between the two classes.

The simplest model of a SVM starts with a maximal margin classifier. It works only for linearly separable cases in feature space, so it assumes that there is no training error. Generally it may not be applied to separation of many real datasets. If the data are noisy, no separation exists in feature space. Nonetheless the maximal margin classifier provides key characteristics of this kind of learning machine. First, this can be viewed as a convex optimization problem: minimizing a quadratic function under linear inequality constraints. Suppose that we have a training dataset, $\{x_i, y_i\}$ for $i = 1,.....,l,$ where x is a vector in the input space $S = R^N$ and y_i denotes the class label taking either +1 or -1.

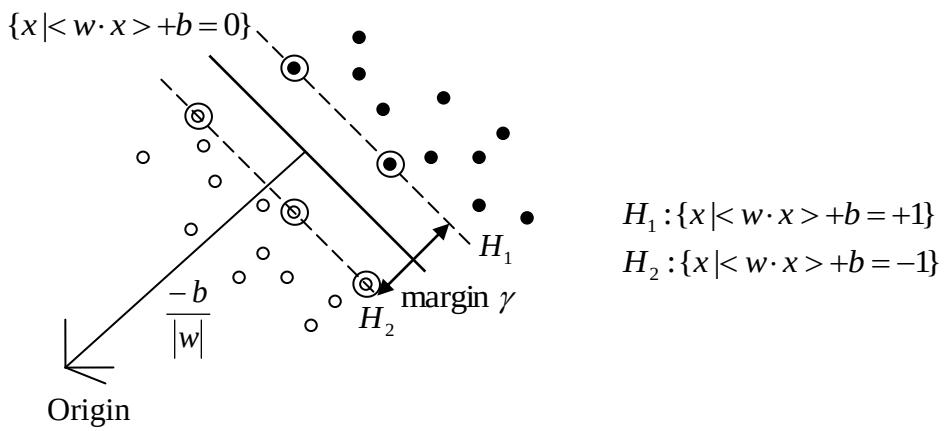


Figure 3.1 Linear separating hyperplanes for the separable case

As shown in Figure 3.1, the points x which lie on the hyperplanes satisfy $< w \cdot x > +b = 0$, where w is normal to the hyperplane, $|b|/\|w\|$ is the perpendicular distance from the hyperplane to the origin, and $\|w\|$ is the Euclidean norm of w. For the linearly separable case, the support vector algorithm looks for the separating hyperplane with the largest margin. Suppose that all the training data meet the following constraints.

$$w \cdot x_i + b \geq +1 \text{ for } y_i = +1 \quad (3.1)$$

$$w \cdot x_i + b \leq -1 \text{ for } y_i = -1 \quad (3.2)$$

The margin γ is defined as distance between two hyperplanes, $H_1 : w \cdot x_i + b = +1$ and $H_2 : w \cdot x_i + b = -1$ with a normal w. Since these two hyperplanes are parallel and have the same normal, the margin distance γ that separates them is given by

$$\gamma = \frac{1}{2}\left(\left\langle \frac{w}{\|w\|^2} \cdot x^+ \right\rangle - \left\langle \frac{w}{\|w\|^2} \cdot x^- \right\rangle\right)$$
$$= \frac{1}{2\|w\|^2}\left(\langle w \cdot x^+ \rangle - \langle w \cdot x^- \rangle\right) \qquad (3.3)$$
$$= \frac{1}{\|w\|^2}.$$

Each instance that falls on one of the two hyperplanes is called a *support vector* (SV). The SVs in Figure 3.1 are circled. Given this geometric relationship, finding the hyperplanes with the maximum margin in feature space can be formulated as a mathematical programming problem. For a linearly separable training data $S = ((x_1, y_1),....,(x_l.y_l))$, the hyperplane producing maximum margin is found by formulating the minimization optimization problem as follows:

$$\begin{aligned} &\text{minimize} && \|w\|^2 / 2 \\ &\text{subject to} && y_i(< w \cdot x_i > + b) \geq 1, \\ & && i = 1,....,l \end{aligned} \qquad (3.4)$$

We now switch to a Lagrangian formulation of the problem with the Lagrange multipliers, $\alpha_i \geq 0$. By doing this, the constraints in Equation (3.4) are replaced by constraints on the Lagrangian multiplier themselves, which are easier to handle. The primal Lagrangian is given by

$$L_P(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i(< w \cdot x_i > + b) - 1], \ \alpha_i \geq 0 \ . \qquad (3.5)$$

Then we must minimize $L_P(w,b,\alpha)$ with respect to *w* and *b*, subject to $\alpha_i \geq 0$. This is a convex quadratic programming problem since the objective function is itself convex, and those

points which satisfy the constraints also form a convex set. This indicates that this problem can be equally solved with its corresponding dual problem, subject to the derivatives of $L_P(w,b,\alpha)$ with respect to *w* and *b*,

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0, \qquad w = \sum_{i=1}^{l} y_i \alpha_i x_i \tag{3.6}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 = \sum_{i=1}^{l} y_i \alpha_i \tag{3.7}$$

also subject to the constraints, $\alpha_i \geq 0$. Since these are equality constraints in the dual formulation, the dual can be substituted (3.6) into (3.5) and results in

$$\begin{aligned} L_D(w,b,\alpha) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i (< w_i \cdot w_i > + b) - 1] \\ &= \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i \cdot x_j > - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i \cdot x_j > + \sum_{i=1}^{l} \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i \cdot x_j > . \end{aligned} \tag{3.8}$$

The primal (L_P) and dual (L_D) come from the same objective function but with different constraints and the solution is found by minimizing L_P or maximizing L_D. Given that we want to maximize L_D with respect to α_i, the optimization problem can be formulated as

$$\begin{aligned} &\text{maximize} && L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i \cdot x_j > \\ &\text{subject to} && \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & && \alpha_i \geq 0, \quad i = 1,....,l . \end{aligned} \tag{3.9}$$

In solving this problem, the positive values for each α_i gives $w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$ which generates the maximal margin hyperplane with margin $\gamma = 1/\|w^*\|^2$. Those points whose α_i is positive are the *SVs* (and are located on H_1 and H_2 in Figure3.1), while other points' α_i are zero.

Since the value of b does not appear in the dual problem, b^* is found making use of the primal constraints. The optimal solutions α^* , (w^*, b^*) must satisfy $\alpha_i^*[y_i(< w^* \cdot x_i > + b^* - 1] = 0, \quad i = 1,...,l$. This is the Karush-Kuhn-Tucker (KKT) complementary optimality condition. With this condition, b can be computed. The hyperplane decision function can then be written as

$$f(x) = sign(\sum_{i=1}^{l} \alpha_i^* y_i < x_i \cdot x_j > + b^*). \tag{3.10}$$

This implies that support vectors are the critical points in the training set and lie closest to the hyperplane producing the maximum margin of two different class labels.

So far we have considered only a separable case of training data. How can we extend these strategies to deal with a non-separable case? This is done by introducing a positive slack variable, ξ_i, $i = 1,....,l$. Constraints become:

$$w \cdot x_i + b \geq +1 - \xi_i \text{ for } y_i = +1 \tag{3.11}$$

$$w \cdot x_i + b \leq -1 + \xi_i \text{ for } y_i = -1 \tag{3.12}$$

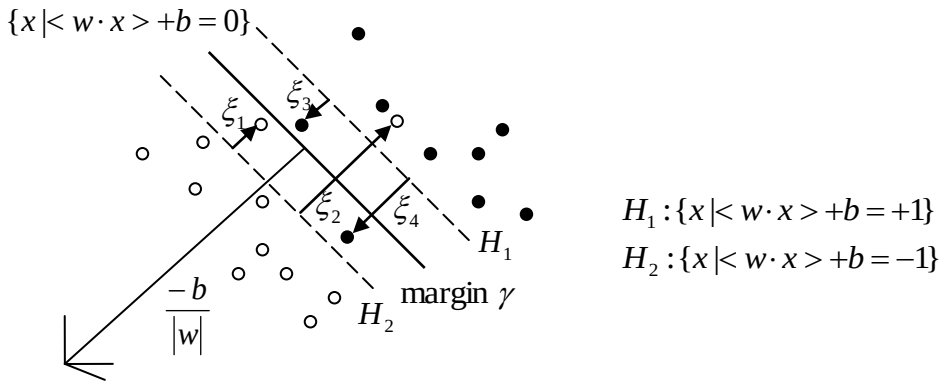


Figure 3.2 Linear separating hyperplanes for the non-separable case

So when an error occurs, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is the upper bound on the number of training errors. Hence a natural way to assign an extra cost for errors is to change the objective function to be minimized from $\|w\|^2 / 2$ (see equation 3.4) to $\|w\|^2 / 2 + C \sum_{i=1}^{l} \xi_i$, where C is a penalty parameter chosen by the user. This is the concept behind *soft-margin SVMs*. Introducing μ_i as the Lagrange multipliers of the ξ_i, the Lagrange (primal) function is:

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i [y_i (< w \cdot x_i > +b) - 1 + \xi_i] - \sum_{i=1}^{l} \mu_i \xi_i \ . \qquad (3.13)$$

For minimizing the Lagrange (primal) problem with respect to *w*, *b*, and ξ_i, setting the respective derivatives to zero, we get equations (3.6) , (3.7) above, and

$$\alpha_i = C - \mu_i, \quad \forall i \ . \qquad (3.14)$$

By substituting the derivatives into the primal problem, the corresponding dual problem is formulated as

$$\begin{aligned} &\text{maximize} && L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i \cdot x_j > \\ &\text{subject to} && \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & && 0 \le \alpha_i \le C, \quad i = 1,....,l . \end{aligned} \tag{3.15}$$

With the derivatives and the KKT condition, equations (3.16)-(3.18), we obtain

$$\alpha_i \{ y_i (x_i \cdot w + b) - 1 + \xi_i \} = 0 \tag{3.16}$$

$$\mu_i \xi_i = 0 \tag{3.17}$$

$$y_i (x_i \cdot w + b) - (1 - \xi_i) \ge 0 \tag{3.18}$$

The solution is given by $w = \sum_{i=1}^{Ns} \alpha_i y_i x_i$, where *Ns* is the number of SVs which non-zero coefficient $\hat{\alpha}_i$. Among the SVs, some lie on the edge of the margin ($\hat{\xi}_i = 0$) that is characterized by $0 < \hat{\alpha}_i < C$ from equations (3.17) and (3.14) and the remaining ($\hat{\xi}_i > 0$) have $\hat{\alpha}_i = C$. Therefore, points on the wrong side of the boundary are SVs and points on the correct side of the boundary are also SVs.

A SVM is a linear classifier, but in most cases, it is practically restrictive. SVM can be easily extended to a nonlinear classifier by mapping the input space into a high dimensional feature space through a kernel function, $K(x_i, x_j)$ which computes the dot product of the data points in the feature space *H*, that is,

$$K(x_i, x_j) = \phi(x_i).\phi(x_j) \quad . \tag{3.19}$$

Functions that satisfy Mercer's theorem (Burges, 1998) can be used as dot products and thus can be used as kernels. Common kernel functions include *Linear:* $K(x_i, x_j) = (x_i \cdot x_j)$, *Polynomial*: $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$ and *Gaussian Radial-based :* $K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2}$.

Thus the nonlinear separating hyperplane can be found formulated as an optimization problem given by

$$\begin{aligned} &\text{maximize} && L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i \cdot x_j) \\ &\text{subject to} && \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & && 0 \le \alpha_i \le C, \quad i = 1,....,l. \end{aligned} \tag{3.20}$$

The corresponding decision function is

$$f(x) = sign(w \cdot z) + b) = sign(\sum_{i=1}^{l} \alpha_i y_i K(x_i \cdot x_j) + b). \tag{3.21}$$

3.2 SVMs and the Skewed Boundary

As noted previously, imbalanced data sets cause a bias in the results of a SVM. Akbani et al. (2004) have summarized three reasons why a skewed boundary occurs in SVM classification for imbalanced data sets: (1) *positive (minority) instances lie further from the ideal boundary compared with negative (majority) instances,* (2) *the weakness of the soft-margin SVMs* and (3) *the imbalanced SV ratio*. For the third reason, according to the KKT conditions in solving the optimization problem in SVM, the values for α_i must satisfy $\sum_{i=1}^{n} \alpha_i y_i = 0$. Since the values for the minority class tend to be much larger than those for the majority class and the number of minority SVs substantially smaller, the nearest neighborhood of a test point is likely to be

dominated by majority SVs. That means that the decision function is more likely to classify a boundary as majority. The second reason, the weakness of the soft-margin SVMs, is an inherent weakness in coping with imbalanced data learning. For separable cases, the imbalance of class distribution rarely influences the performance of SVMs because all the slack variables ξ_i are equal to zero (equations 3.11 and 3.12). Therefore, there is no contradiction between the capacity of the SVMs and the classification error. However, for non-separable cases, soft-margin SVMs should achieve a trade-off between maximizing the margin between two classes and minimizing the classification error. Typically much more majority instances appear in the overlapping area than minority ones. So, the optimal hyperplane will be skewed on the minority class side in order to reduce the overwhelming errors of misclassifying the majority class. If *C* is not very large, SVMs simply predict most of minority instances as majority instances to make the margin as large as possible, making the total misclassification cost as small as possible.

Several methods for SVMs for imbalanced data learning have been studied (Karakoulas & Taylor, 1999; Lin et al., 2002). At the data level, rebalancing approaches such as over-sampling (i.e. SMOTE) and under-sampling have been widely used for SVM to cope with imbalanced datasets. Veropoulos et al. (1999) suggested using different penalties for misclassification of the classes. Amari and Wu (1999) proposed using conformal transformation of the kernel matrix to enlarge the separation between two classes. In the first step, it finds the separating location between two classes through standard SVM learning. In the second step, the primary kernel matrix is conformally scaled to give a wider separation. Separation is controlled by the SVs, so the new kernel matrix is enlarged at the position of SVs.

Another approach is the one-class SVM (Scholkoft and Smola, 2002). This uses only minority instances for learning. Its original application is detecting outliers that differ from most

of the data within a dataset. This determines a hyperplan in feature space that separates most of the data from the origin. It completely ignores information of the majority class: Instead, only using one class, the minority class , it defines a hyperplane that separates most of data belonging to the class from the origin.

3.3 Problems associated with SVM classifier for imbalanced data

When focusing on approaches at the data level (rebalancing the data distribution), there are two significant problems associated with a SVM classifier, namely,

1. *Over-sampling methods significantly increase the dataset size.*
2. *An optimal ratio of class distribution is empirically determined by grid search.*

To address these problems, we propose a new sampling method at the data level for imbalanced data learning. Instead of rebalancing the entire imbalanced dataset, a selective sampling method is proposed that results in a relatively small number of instances. We expect that a small set of representative instances of an imbalanced dataset could determine the desired decision boundary by maintaining the same or achieving even better performance for the class imbalance problem as compared to existing rebalancing methods. The merits of this approach include: (1) skipping empirical search that is necessary in sampling methods such as optimal ratios of class distributions and (2) avoiding producing a large training set that would lead to long training times for a SVM. If this approach performs well as compared to some current methods, it will provide an alternative method to solve the class imbalance problem with the advantage of reducing learning time for a SVM.

3.4 Effectiveness of rebalancing class distribution

For imbalanced and highly overlapped class data, sampling methods such as over-sampling or under-sampling are very effective in terms of the optimization process in a soft-margin SVM. In order to illustrate the effect of sampling methods in rebalancing class distribution, we examine the boundary movement in two common methods, SMOTE over-sampling and Random under-sampling. In this example, a Gaussian kernel function is used for SVM classification.

First, we generated a simple structure of a synthetic dataset showing a typical class imbalance problem, which could be represented in 2-dimensional space as shown in Figure 3.3. The minority class having 40 instances is marked with 'o' and the majority having 400 instances with '•' (a class ratio of 1:10).

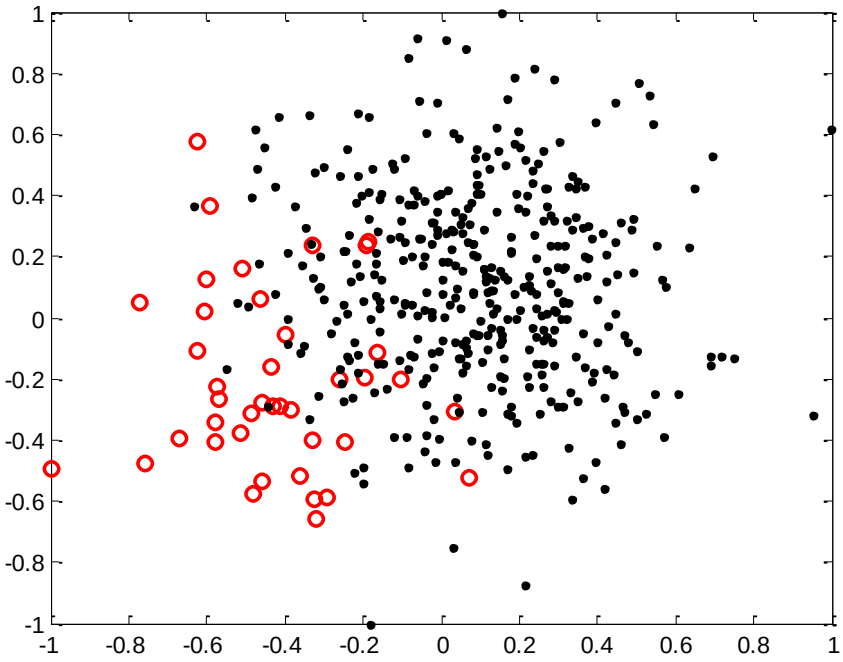
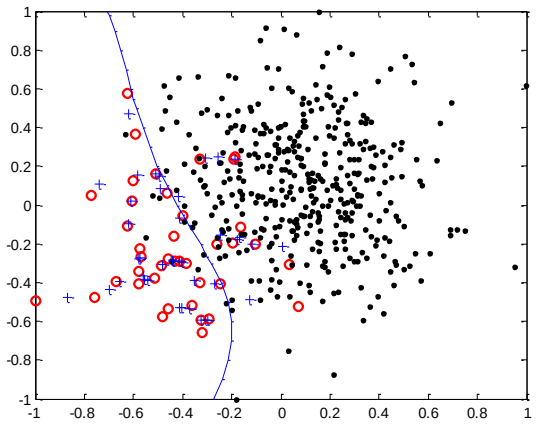


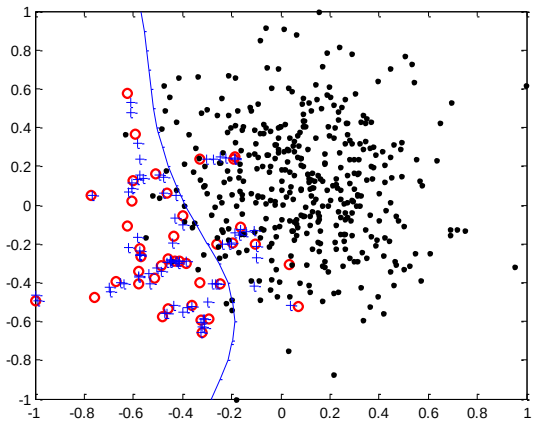
Figure 3.3 Example of class imbalance problem on SVMs

After classification, almost all majority instances were correctly classified, while many minority instances were classified as the majority class. This example illustrates a typical class imbalance problem caused by soft-margin SVM algorithms. In other words, an optimal hyperplane results from the trade-off between maximizing the margin of the minority and the majority class and minimizing misclassification costs in the feature space. To improve the accuracy for the minority class, we need to move the boundary toward the majority class side. To illustrate this, we applied two rebalancing sampling methods, SMOTE and random under-sampling, which will be referred to as SVM-SMOTE and SVM-RU, respectively.

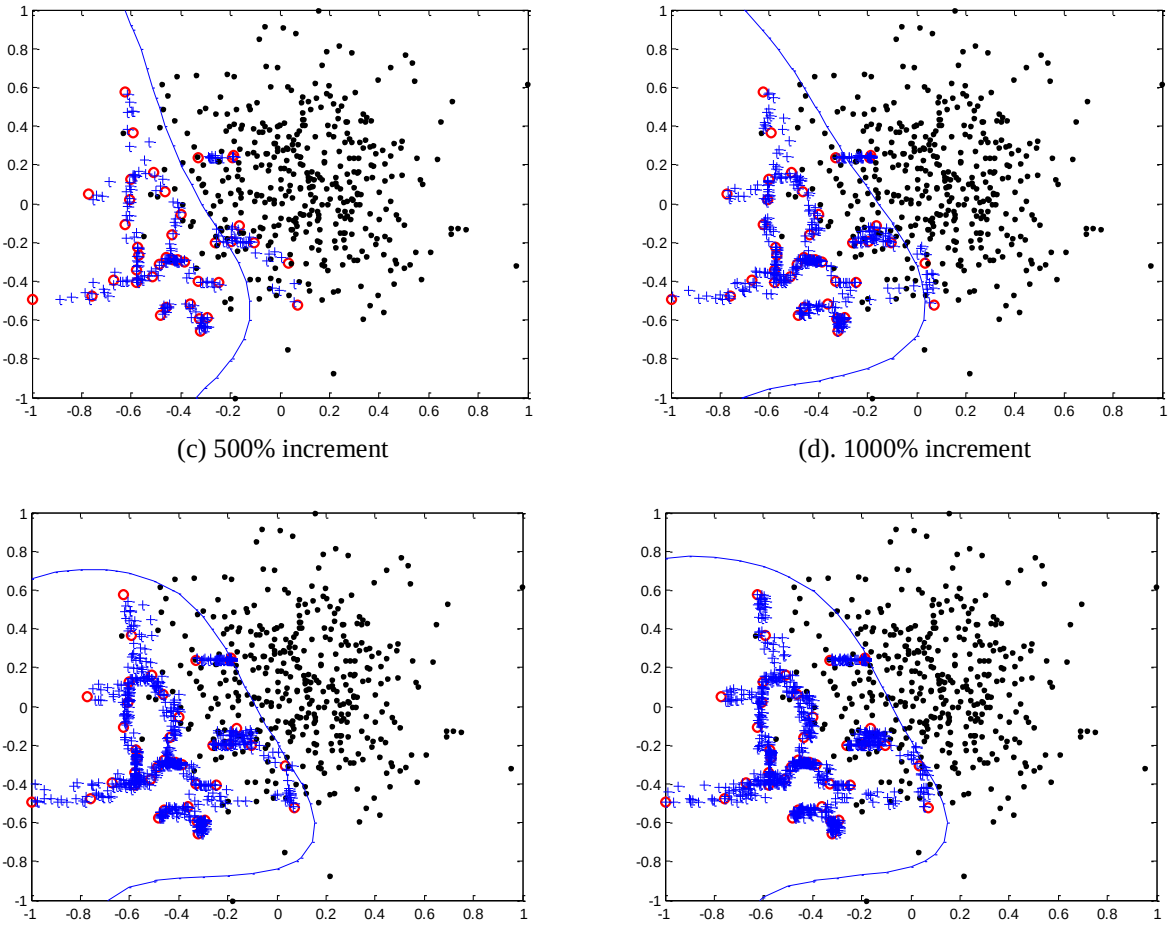
Using SVM-SMOTE, the number of synthetic instances to achieve the desired class balance is unknown and empirical studies must be performed. Minority instances are oversampled gradually with 100%, 300%, 500% and 1000% increases in minority instances. After rebalancing by SMOTE, we observed that the boundary gradually shifted toward the majority class as minority instances are increased as shown in Figure 3.4 (a) to (f).



(a) 100% increment



(b) 200% increment



(c) 500% increment

(d). 1000% increment

(e) 1700% increment

(f) 2000% increment

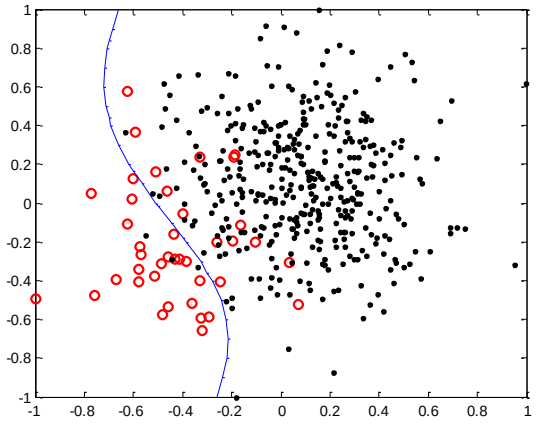
circle(○): minority instances, dot(•): majority instances, cross(+): synthetic instances by SMOTE

Figure 3.4 Boundary movements by SMOTE algorithm

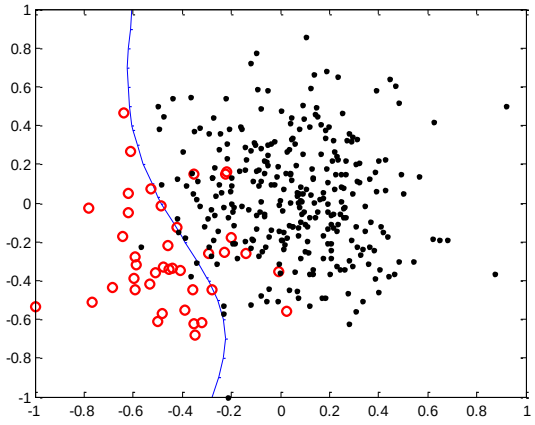
Though SVM-SMOTE shifts the decision boundary, it comes at a penalty of increasing the size of the dataset as mentioned in the previous section. Assume that Np is the number of the positive (minority) instances and Nn the number of the negative (majority) instances, typically SVM takes $O((Np+Nn)^3)$ time for learning in the worst case (Burges, 1998). For imbalanced data learning, SVM-SMOTE will take $O((Np\times(1+R_{smote})+Nn)^3)$ where R_{smote} is an optimal ratio

of size of increasing instances. Here R_{smote} is determined empirically. What is worse, over-sampling also increases instances in the complex region between classes. By generating instances near or in overlapping areas which might be misclassified, classification is more difficult. In solving the optimization problem in the SVM algorithm, many cases can violate KKT conditions. As a result, this will requires much more time to convergence of optimization in SVM algorithms in spite of its good performance. So, if a dataset is extremely imbalanced and overlapped, over-sampling through the SMOTE algorithm would not be efficient. In regard to the problems introduced by an over-sampling approach, an under-sampling method is preferred to over-sampling methods and is commonly used.

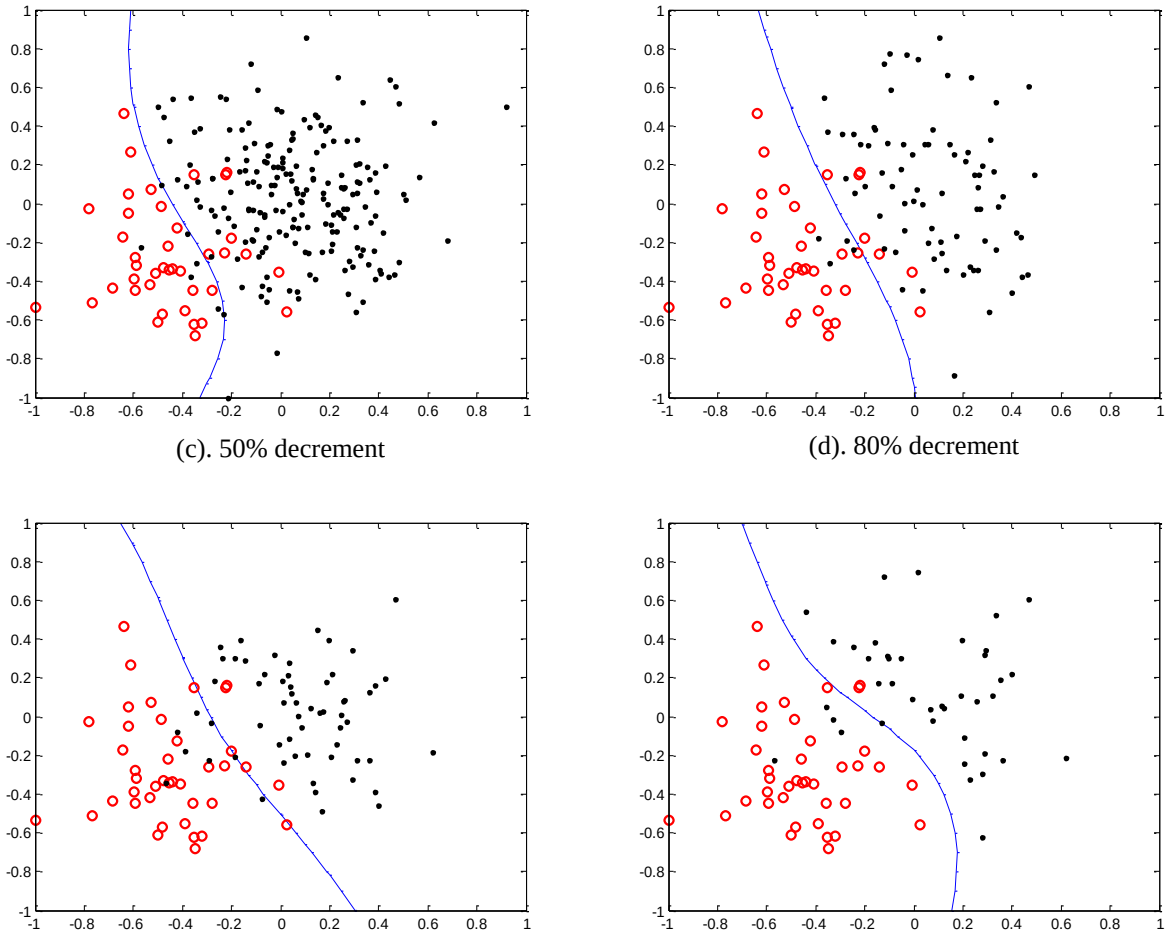
SVM-RU uses random under-sampling to rebalance the class distribution. Instead of increasing minority instances, decreasing the number of the majority instances will make an optimal hyperplane in terms of the trade-off between maximizing the margin and minimizing misclassification costs. Figure 3.5 below shows the boundary movement as the number of the majority instances removed randomly is increased.



(a) No under-sampling



(b). 25% decrement



(c). 50% decrement

(d). 80% decrement

(e). 85% decrement

(f). 90% decrement

Figure 3.5 Boundary movements by random under-sampling

Similar with SVM-SMOTE, random under-sampling causes a shift in the decision boundary towards the majority class. Considering time complexity in learning the dataset, SVM-RU takes $O((Np + Nn \times R_u)^3)$ which is quicker than SVM with the original training set since $Nn \times R_u$ is approximately equal to Nn . Because majority instances have been randomly eliminated, it may not be easy to determine an optimal size of training set for rebalancing. Also

similar to SVM-SMOTE, the optimal desired class distribution for imbalanced data learning is unknown and needs to be determined empirically

3.5 Hypotheses

Given the nature of imbalanced datasets as previously discussed, two hypotheses were formulated to address *effectiveness* and e*fficiency* issues in imbalanced data learning for SVMs.

Hypothesis 1

> *A relatively small number of instances from an imbalanced training set are needed to obtain good performance in solving the class imbalance problem using a SVM.*

Hypothesis 2

> *A smaller subset within the set of support vectors can be found that produces a better boundary using a SVM.*

Hypothesis 1 is related to the premise that the boundary between the major and minor classes is strongly influenced by a relatively small number of instances. The inference is that a potential down-sizing of learning sets will lead to significant improvements in classifier performance. Hypothesis 2 is based on the expectation that there is a smaller group of SVs that will provide a near optimal classification of the classes and improve the efficiency of learning. If such a set of SVs exist, then a metaheuristic-based sampling approach could be used to select the vectors.

CHAPTER 4 SELECTIVE SAMPLING USING A GENETIC ALGORITHM

This chapter presents a selective sampling method based on a Genetic Algorithm for imbalanced data learning with a SVM. Instead of rebalancing new training data distributions for learning, we investigate how this selective sampling method performs on class imbalance problems. Instance selection is a form of under-sampling in which specific instances are selected based on some criteria. We use a Genetic Algorithm to perform selective sampling of the majority instances and retain all minority instances because they are assumed to be informative.

4.1 SVMs for Large-Scale Datasets

Training a SVM involves solving a constrained quadratic programming (QP) problem, requiring a large memory allocation and resulting in long training times for large-scale data. Two issues in using SVMs with large scale data are the generation of the kernel matrix and data overlapping. With a dataset size of N, an $N \times N$ kernel matrix is generated. Furthermore, storing the entire kernel matrix in memory is problematic due to computer memory limitations. The second issue is the increase in classification difficulty when many class data are overlapped.

Due to large-scale data problem associated with SVMs, many studies have examined methods for reducing training samples to achieve relatively fast SVM classification. For example, Shin and Cho (2003) proposed a selection method for selecting patterns from training data near the decision boundary based on the neighborhood properties. Zhang and King proposed a β-skeleton algorithm to identify SVs. Almeida et al. (2000) employed K-means clustering to select patterns from a training set. Lee and Mangasarian (2001) selected a subset of training instances using random sampling to reduce training sample size. Huang and Lee (2004) showed

that uniform random sampling is an optimal robust selection scheme in terms of several statistical criteria. Han et al. (2008) proposed a novel pre-extracting method for SVM classification using a non-parametric *k*-NN rule called relative neighborhood graph (RNG). Given that SVs are critical points in learning classification boundaries for a SVM, these methods extract significant samples which are likely to become SVs prior to the learning process. Some variant SVM methods include modifications of the SVM algorithm. For example, some decomposition methods break the large QP problem into a series of smaller QP sub-problems (Osuan et al., 1996) and Sequential Minimal Optimization (SMO) by Platt (1999). SVM^{light} is an implementation of an SVM learner for large scale tasks that uses the decomposition idea of Osuan et al.

To address the difficulties in learning a large-scale dataset with a SVM, methods which significantly reduce the size of the training set will have better computational performance (i.e., more efficient classification). Some form of instance selection in which instances are selectively sampled is consistent with the concept of efficiency.

4.2 Genetic Algorithm for Under-sampling of the Majority Class

The Genetic Algorithm (GA) metaheuristic was introduced by Holland (1973) as an evolutionary algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, natural selection and recombination (or crossover) (Goldberg, 1989). In diverse application areas, GAs have been used extensively to solve combinatorial optimization problems to find a good solution even though it may not be optimal. In data mining, GA can improve the behavior of a predictive model for classification performance (accuracy) and interpretability through selective sampling of training sets.

Briefly reviewing general GAs, the main characteristics of a GA include:

- **A population of individuals** The population in the k^{th} generation, P^k, contains a set of individuals. Each individual represents a possible solution and has a unique configuration of parameters called a chromosome, *C*.
- **Fitness function** An individual is tested and assigned a level of fitness that indicates the degree to which it is a good solution. Fitness levels are used to decide which individuals should be selected to produce the next generation of individuals. The better the fitness of an individual, the more likely it is to be selected to be a parent for the next generation.
- **GA operators** The next generation of individuals is produced using two genetic operators after selecting the parents. The first operator, recombination, exchanges a part of the parents' genes (i.e., subsets of the parent members). The second operator, mutation, changes a small subset of an individual gene in a random fashion. The combination of these operators with parent selection and a fitness function enables the GA to evolve towards a better solution.
- **Termination of the GA** The stopping condition is based on determining if the quality of the members of the population is satisfactory or if a pre-determined number of iterations have been completed.

In order to use GA to select majority instances, we first define the solution representation (i.e., chromosome) as a set of binary digits (0: if an instance is not selected and 1: if an instance is selected). Given that selective sampling is performed only for the majority class, only instances of the majority class are represented in a chromosome,

The binary chromosome can be represented as a vector of genes $C = [g_1, g_2, g_3, \cdots\cdots, g_N]$, where *N* is the number of the majority instances, g_i is 0 (not selected) or 1 (selected). GA starts with an initial random population of *M* chromosomes, $P^0 = \{C_1^0, C_2^0, C_3^0, \cdots\cdots, C_M^0\}$. The superscript indicates the k^{th} iteration for the GA method.

Starting with an initial population of solutions, P^0, which is randomly generated, the GA operations are applied to improve the population. For chromosome selection from P^k, we employ a "roulette wheel" mechanism to probabilistically select a C_i^k based on some measure of its performance. A real-valued interval, *Sum*, is determined as the sum of fitness values of all individual chromosomes in the current population. Then each chromosome is mapped one-to-one into contiguous intervals in the range [0, *Sum*]. The length of each chromosome interval corresponds to the fitness value of the associated chromosome. In order to select a chromosome, a uniform random number is generated in the interval [0, *Sum*] and the C_i^k whose segment spans the random number is selected. This selection method gives larger weight to chromosomes whose fitness values are large relative to other chromosomes, so they have a higher probability of being chosen. This selection process is repeated until the desired number of individuals has been selected.

Genetic operators manipulate the g_is of a chromosome directly, using the assumption that certain g_is , on average, produce fitter individuals. The crossover operator is used to exchange certain g_is between individuals. There are different ways to perform the crossover operation. The simplest one is one-point crossover that randomly picks an *i* and switches the g_i between a pair of chromosomes. Multi-point crossover picks *m* crossover points $\{\chi_1, \ldots, \chi_m\}$

randomly with no duplicates such that $\chi_i \in \{1,2,.....,l-1\}$, where l is the length of the chromosome. The values at g_{χ_i} are exchanged between the two parents to produce two new offspring chromosomes. In our method, we use a single point crossover. As another natural evolution, mutation is a random process where a small subset of genes are replaced by another to produce a new genetic structure. In GA, mutation is randomly applied with low probability, typically in the range 0.001 and 0.01, and modifies elements in the chromosomes.

The fitness function used in this study was Kubat's (1999) g-mean value for the original training set after training the SVM classifier with the instances specified in the chromosome for the majority class instances along with all of the minority class instances. The geometric mean (g-mean) was used as a fitness function in GA because it simultaneously considers the accuracy of the minority and majority classes. The fitness function, $f(svm(IS), T_0)$ is defined as

$$\max f(svm(IS), T_0) = \arg max\ (A_s^+ \cdot A_s^-)^{1/2}$$

where,

T_0 = original training set including all instances

IS = instance set (all minor instances and selected majority instances by GA-based Instance Selection)

A_s^+ = accuracy of the minority class in T_0 after learning a new training set (IS)

A_s^- = accuracy of the majority class in T_0 after learning a new training set (IS)

A common practice is to terminate the GA operations after a predetermined number of generations. Here we set up a maximum number of generations to stop the GA. The procedure is described in Figure 4.1.

T_0	Original training dataset
C_i^k	Binary chromosome representation of the majority instances
M_k	Number of individual chromosomes in population *k*
MAXGEN	Maximum number of generations
$sel.IS_i^k$	Instance set that is combined with minority instances and a subset of the majority instance that is represented by C_i^k
selection	Selection method in selection operator in GA

Initialize: *k*=0;

Generate an initial population, $P^0 = \{C_1^0, C_2^0, C_3^0, \ldots\ldots, C_{M_k}^0\}$

while *k* < *MAXGEN* **do**

for *i* ← 1 to M_k

$fitness_i^k \leftarrow f(svm(sel.IS_i^k), T_0)$

endfor

/***** new chromosomes produced by GA operators*/

$P^{k+1} = P^k$

$C_i^k = select('selection', P^k);$

$(C_i^{k+1}, C_j^{k+1}) = crossover(C_i^k, C_j^k);$

$C_i^{k+1} = mutate(C_i^{k+1});$

k = *k*+1;

endwhile

return max $(sel.IS_i^k, f(svm(sel.IS_i^k), T_0))$ /* select an instance set that maximizes the fitness function after terminating GA

}

Figure 4.1 GA-based Instance Selection from the majority instances

4.3 Experiments

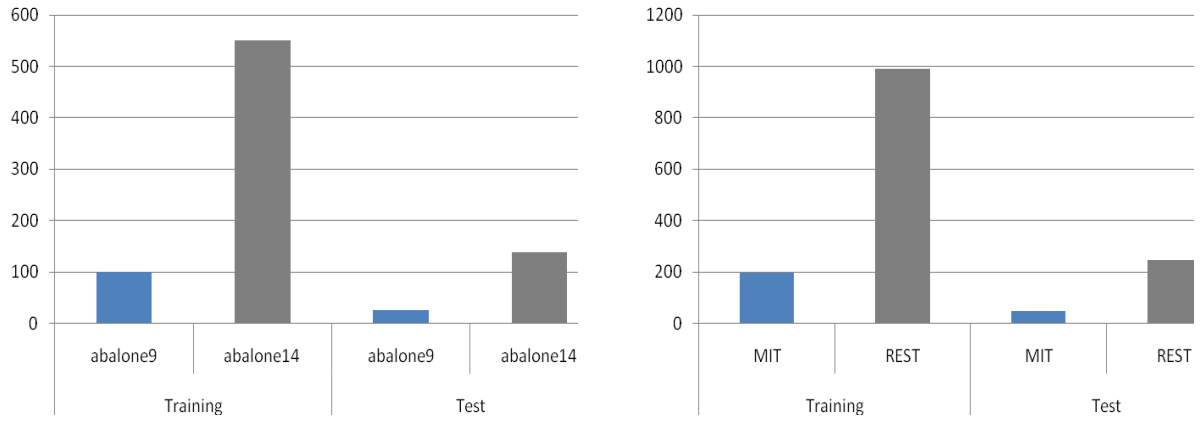
This section describes the empirical study of instance selection on SVMs using our GA method as compared to existing methods. For this experiment, we used five real datasets, *abalone, blood transfusion, pima, wine, and yeast* datasets from the UCI data mining repository (Table 4.1). The class imbalance problem was limited to a two class classification problem in each case. For those datasets having multiple classes, two overlapping classes were selected (Appendix A). All steps of data pre-processing and model development were carried out in MATLAB R2008b and the algorithms were implemented using the SVM-KM toolbox (Canu et al., 2005).

4.3.1 Experimental Design

First, before using SVM, scaling of attribute values was necessary for the datasets to avoid attributes in greater numeric ranges from dominating those in smaller numeric ranges. All attributes were scaled within a range of [-1, 1]. Using a Gaussian radial-based kernel function, $k(x,x^{'}) = \exp(-\left\|x-x^{'}\right\|^2 \big/ 2\sigma^2)$, we set parameters C and σ such that the best performance was obtained for the overall classification accuracy for each dataset (see Appendix B). This was done using five-fold cross validation through different combinations of C and σ, ($\{1, 10, 100, 200, 500, 1000\} \times \{0.1, 0.2, 0.5, 1, 2, 3\}$). Each dataset was then divided into 2 sets: training set (80%) and test set (20%) according to the original class distribution (see Figure 4.2). After using our GA method to find the boundary with the training set, the boundary was evaluated using independent test datasets.

Table 4.1 Descriptions of experimental datasets

Datasets	Data description	Number of Instances	Number of Attributes
Abalone	Predicts the number of rings through a microscope for measuring the age of abalone. Originally, this dataset contained 4177 instances and 29 predicted rings (abalone1~29). For our experiment, abalone 9 and abalone 14 are used for classification (abalone9 vs abalone14).	815 (abalone9:126, abalone14:689	7 numeric
Blood transfusion	Classifies whether a person donated blood (Donate vs Non-donate)	748 (Donate:178 Non-datae:570)	4 numeric
Pima	Predicts when a patient shows signs of diabetes according to World Health Organization criteria (Positive vs Negative)	768 (positive:268, negative:500)	8 numeric
Wine (red wine)	Predicts wine quality (score between 0 and 10) based on physicochemical tests. For the two class problem, those with a score greater than 5 were rated High quality and less than 5 were rated Low quality.	1599 (High:217, Low:1382)	11 numeric
Yeast	Predicts localization site of protein. Originally it is grouped with 10 classes. For this experiment, we take MIT (mitochondrial) as the minority class the rest of classes as the majority class for two class classification problem (MIT vs REST).	1484 (MIT:244, REST:1240)	8 numeric



(a) abalone dataset

(b) yeast dataset

Figure 4.2 Class distributions of experimental datasets (*abalone* and *yeast*)

Conducting SVM classification with a Gaussian radial based kernel function without selective sampling, we observed a typical class imbalance problem from those experimental datasets. This can be found that the g-mean values are consistently low, as seen in Table 4.2. Details on the classification results can be found in Appendix C.

Table 4.2 G-mean values of the experimental datasets on SVM (training and test sets)

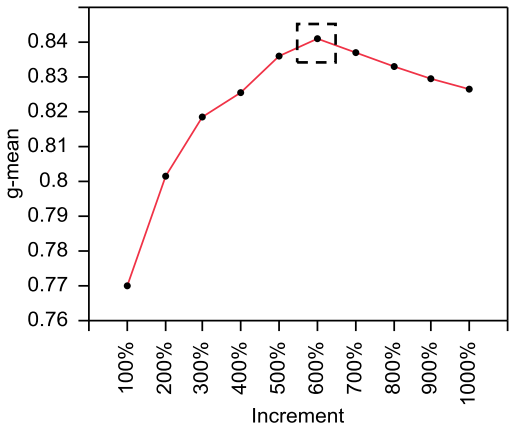
Datasets	*g-mean*	
	Training Dataset	Test Dataset
abalone	0.65	0.48
blood transfusion	0.49	0.44
pima	0.71	0.67
wine	0.84	0.68
yeast	0.69	0.65

4.3.2 Approaches for imbalanced data learning

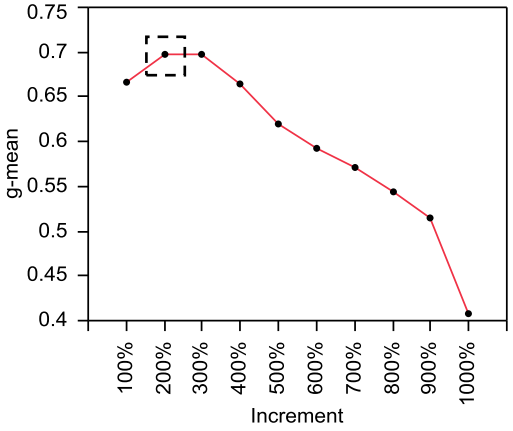
We evaluated our GA-based Instance Selection method by comparing its results with those obtained using SVM-SMOTE, SVM-RU, and Biased penalty methods.

Method 1. SVM-SMOTE

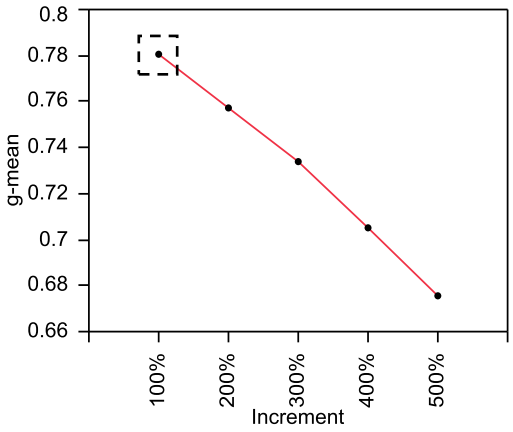
For the SMOTE algorithm, *K* was set to 5 for the *K-NN* calculation. The number of synthetic instances for the minority class was increased linearly. Learning was performed using 20 independent synthetically enhanced datasets and the maximum average g-mean was calculated to identify the best synthetic sample size. For example, if the maximum average g-mean of the original training set appears when 200% of new synthetic instances are added into the training dataset, an increment of 200% is selected. While increasing minority instances gradually, we observed g-mean values of the original training sets for each experimental dataset. Figure 4.3 shows g-mean values of the training sets as instances are added by SMOTE.



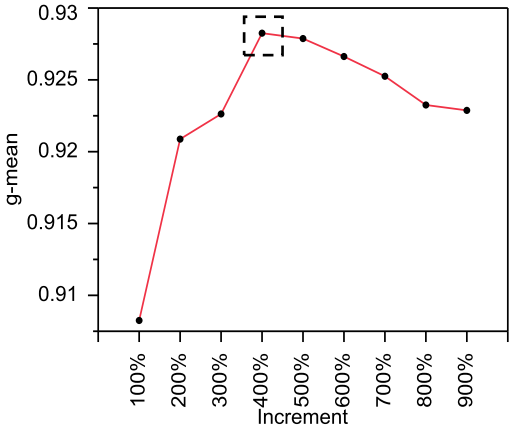
(a) abalone dataset



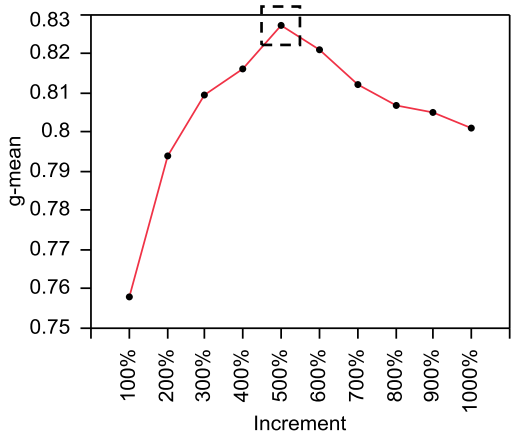
(b) blood transfusion dataset



(c) pima dataset



(d) wine dataset



(e) yeast dataset

Figure 4.3 Average g-mean values of the training dataset in terms of increase of synthetic minority instances by SMOTE

Based on the results, adding instances near the original minority instances causes SVM to incur more misclassification costs. In order to reduce misclassification cases, the hyperplane is shifted to the majority class side. This shift can be seen as the g-mean value changes. From SVM-SMOTE, the maximum g-means were found for increments of 600% for *abalone*, 200% for *blood transfusion*, 100% for *pima*, 400% for *wine*, and 500% for *yeast* . These are marked with a dash-square as shown in Figure 4.3.

Method 2. SVM-RU

We also conducted random under-sampling of the majority instances. In the same way as SVM-SMOTE, learning with under-sampled training sets was repeated 20 times for each size of the reduced training set. Then we chose the reduced training set that produced the maximum g-mean value for the original training set. Consequently, as we anticipated, the randomness of under-sampling did not produce a consistent result as compared with SMOTE.

Method 3. Biased Penalty method (Distinct C parameters)

One way to alleviate imbalanced data learning is to assign a smaller misclassification cost to the majority class while assign a larger cost to the minority class (Veropoulos et al., 1999).

Supposing that m_1 and m_2 denote the size of class1 (minority) and class2 (majority) respectively, we want to

$$\text{minimize } \|w\|^2 / 2 + C_1 \sum_{i=1}^{m1} \xi_i + C_2 \sum_{j=1}^{m2} \xi_j \tag{4.1}$$

where C_1 and C_2 are defined respectively as

$$\begin{aligned} C_1 &= \frac{m_2}{m_1 + m_2} \times C, \\ C_2 &= \frac{m_1}{m_1 + m_2} \times C. \end{aligned} \tag{4.2}$$

Assigning distinct misclassification penalties emphasizes the minority instances rather than the majority instances. Adjusting different penalties for misclassified instances according to class distribution was not considered in this comparison.

Method 4. GA-based Instance Selection

As described earlier in this chapter, we used the GA-based Instance Selection (GA-IS) for the majority class. Similar to SVM-SMOTE, this method was repeated 20 times. The settings for the GA method included a fixed population size of 40 and an upper limit of 200 generations for termination.

4.4 Experimental Results and Discussion

The average g-mean values for the original training set using the four different methods are shown in Table4.3. From Analyis of Variance (ANOVA), differences were observed for the methods (see Appendix D). Comparison of mean differences for all pairs of methods showed that SVM-SMOTE and GA-IS slightly outperformed SVM-RU and the biased Penalty methods. In the case of the *wine* dataset, SVM-SMOTE and GA-IS did not show a significant mean difference.

Table 4.3 Average G-mean of Training sets obtained from 4 different methods

	G-mean of Training set				
	SVM with the original training set	SVM-SMOTE	SVM-RU	Biased Penalty C	GA-IS
abalone	0.6471	**0.8390**	0.8129	0.8185	0.8173
blood	0.4929	0.6973	0.6899	0.6900	**0.7072**
pima	0.7115	0.7717	0.7662	0.7454	**0.7914**
wine	0.8377	**0.9285**	0.8827	0.8942	**0.9257**
yeast	0.6929	**0.8251**	0.8055	0.8015	0.8089

In summary, the two rebalancing approaches, particularly SVM-SMOTE, are effective methods to move a decision boundary for highly overlapped cases as illustrated in the experimental datasets. However, since optimal value for R_{smote} (the number of synthetic instances) is unknown, it should be determined using a grid search, which means gradually increasing or decreasing the training set size to find an optimal rebalanced training dataset. While SMOTE produced good results, the increase in instances can cause a large increase in the computation time for a SVM learning process and at the same time require a large amount of memory capacity. GA-IS provides equivalent accuracy performance with a significant reduction in learning time and demands for memory.

In reducing the training dataset size and avoiding a grid search for determining an optimal class distribution, GA-IS performs well. However, when the number of the majority instances is large, it requires additional time for the selection process. Therefore, we need to modify the method to reduce the instance set and produce much smaller training datasets, This is addressed in the next chapter.

CHAPTER 5 SMALLER LEARNING SETS for IMBALANCED DATA LEARNING with SVMs

This chapter introduces a new methodology that produces a smaller training set for imbalanced data learning with a SVM. Since the size of training set on SVM is a critical issue, previous research has focused on finding representative training sets for SVM classification. In this chapter, we describe a new methodology to solve the class imbalance problem with SVMs by producing small training sets that have good performance in terms of efficiency and effectiveness.

5.1 A new method to reduce learning time

The time complexity of a SVM is considered to be $O(M^2)$ where M is the number of training instances (Shin and Cho, 2003). Effective heuristic methods used to reduce SVM learning time divide the original quadratic program into a series of small problems (Platt, 1999). Using decomposition methods, time complexity of $I \cdot O(Mq + q^3)$ can be achieved where I is the number of iterations and q is the number of working sets, but the dilemma is that I is proportional to M. Even with these methods, large learning times are typical for large datasets. Therefore, many studies have tried to reduce training samples for fast learning on SVM as explained in Chapter 4. These approaches try to detect instances that could be defined as SVs in SVM classification, focusing primarily on instances close to the overlapping region of the classes. Imbalance data learning introduces the additional problem of finding a hyperplane that results in high G-mean values.

We propose a new approach at the data level for the class imbalance problem to significantly reduce the size of the training dataset. This approach is based on the premise that

redundant instances do not harm correct classifications while they increase classification costs (i.e., computation time or memory usage). Without considering redundant instances, we select *influential* instances (i.e., instances having major influence on the boundary) that affect the classification accuracies of two classes.

In sampling instances, particularly for imbalanced data learning, it is difficult to detect influential instances that possibly influence the determination of decision boundaries because of the mapping of instances from input space to a high-dimensional space through kernel functions. Although we may observe some instances around or in an overlapping region in the input space (real space), the instance locations are unknown in the high-dimensional space (kernel or feature space). Considering the nature of the SVM algorithm in defining a decision boundary, we focus our attention on SVs to effectively move the decision boundary. SVs lie closest to a hyperplane and determine an optimal separation between two classes in kernel space, while non-support vectors do not affect the decision boundary.

Our expectation is that if we can define a hyperplane with relatively few instances from the training set, then we can avoid the high learning time cost and the usual empirical studies used to rebalance datasets while ensuring balanced accuracies for the minority and majority classes. Our approach consists of two stages namely, (1) Rough elimination of instances that are SVs for the majority class in kernel space, and (2) Selection of majority instances which are SVs that maximize the accuracy of classification using a Metaheuristic Algorithm. In the first stage, we collect SVs from iterative removal of instances from the majority class that are SVs. This procedure attempts to under-sample the majority SVs, moving the decision boundary toward the majority class. Furthermore, it narrows the space for instance selection in the second stage. In the second stage, instance selection is performed on the set of SVs for the majority class subsets

using a Genetic Algorithm. The goal in this stage is to find instances that improve overall accuracy of both classes.

5.1.1 Stage1. Rough elimination of support vectors of the majority class in kernel space

The primary question of which instances and how many instances should be eliminated in terms of an under-sampling approach is addressed at this stage. To select instances of the majority class for removal, a SVM learner is used for classification and its corresponding SVs are found. Given that SVs are located closest to a decision boundary, removing SVs from the majority class should be more effective in moving the boundary towards the majority class than under-sampling of instances farther from the boundary. By iteratively classifying and removing SVs, the boundary will gradually shift towards the majority class.

Stage 1 is a preprocessing step for Stage 2 in that the resultant set of SVs are used in Stage 2 to achieve near optimal instance selection. Instead of searching the whole training dataset to select instances that maximize the G-mean of the original training set, we produce a significantly smaller subset of the training set to start instance selection. This improves the performance of instance selection and reduces processing costs in Stage 2.

In general, rebalancing learning sets means over-sampling minority instances, under-sampling majority instances with the whole initial learning sets. Knowing that a SVM learner defines an optimal hyperplane with only SVs, we would avoid over-sampling or under-sampling the whole learning sets, instead, approach to a certain subset that possibly contains learning set for imbalanced data learning, then find an optimal learning set from it.

In experiments with real datasets, results of initial studies on the datasets showed that the G-mean value as a function of iteration number in Stage 1 was well behaved. The value

increased to a maximum and then decreased, which is found that too much elimination of them does not result in a good decision boundary for imbalanced data sets with a SVM, since Stage 1 conducts the elimination of SVs of the majority instances. Therefore, in order to avoid searching for learning sets at Stage 2, the top three performing subsets of SVs were selected for Stage 2. Figure 5.1 summarizes the procedure for Stage1.

Notation for Stage 1

T_0	set of all instances in the original training set.
T_i	set of instances used for training at i^{th} iteration such that $\|T_i\| < \|T_0\|$
$svm(T_i)$	SVM trained with T_i
SV_i^+	set of positive (minority) class instances that are SVs in the at i^{th} iteration
SV_i^-	set of negative (majority) class instances that are SVs in the i^{th} iteration
SV_i	subset of SVs for the i^{th} iteration, $\{SV_i^+, SV_i^-\}$
$\bar{g}(SV_i, T_0)$	G-mean for T_0 using $SV_i SV_i$
$\widehat{SV}$	set of SV_i producing the highest G-mean values for $T_0 T_0$

find $\widehat{SV}$ **(** T_0 **) {**

Initialize:

$i \leftarrow 0$

$Stop \leftarrow False$

While not *Stop*

$i \leftarrow i+1$

If $i > 1$ **then**

$T_i \leftarrow T_{i-1} - SV_{i-1}^-$ /* remove SV_i^- from T_{i-1}.*/

end if

$\{SV_i^+, SV_i^-\} \leftarrow svm(T_i)$ /* collect SVs from T_i */

$\bar{g}(svm(T_i), T_0)$ /* G-mean of T_0 based on classification with T_i */

$SV_i \leftarrow \{SV_i^+, SV_i^-\}$

If $|\widehat{SV}| < 3$ **then**

Add SV_i to $\widehat{SV}$

else

If $\bar{g}(SV_i, T_0) > \min\{\bar{g}(SV_j, T_0) | SV_j \in \widehat{SV}\}$ **then**

Replace SV_j with SV_i

else

$Stop \leftarrow True$

end if

end if

end while

return $\widehat{SV}$

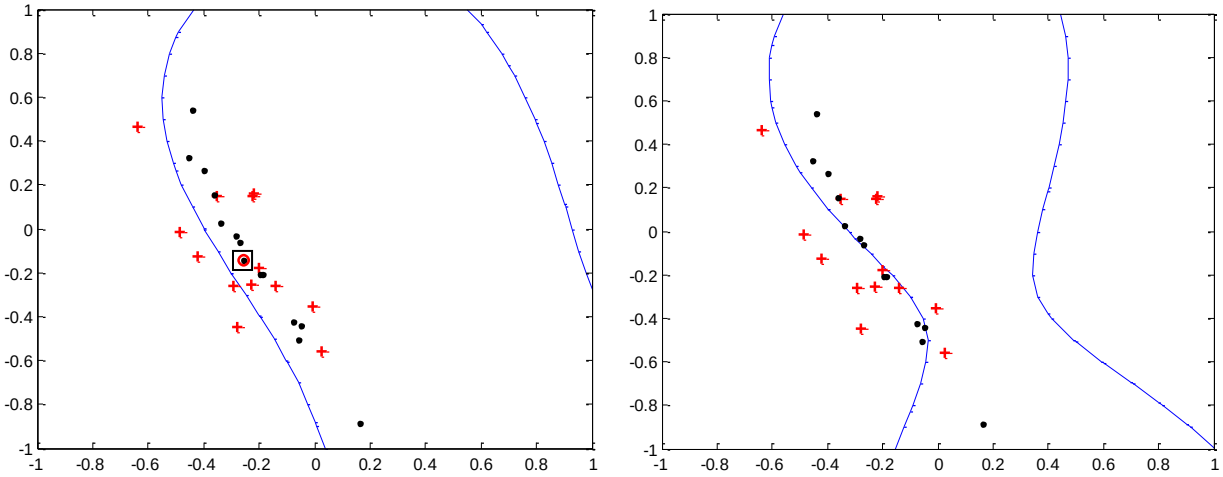
}

Figure 5.1 Stage 1 Algorithm

5.1.2 Stage2. Selection of majority instance support vectors

In Stage 2, we want to select majority instances from the reduced instance training set found in Stage 1 that produce a near optimal classification result. Given a $SV_i \in \widehat{SV}$ from Stage 1 where $SV_i = \{x_1, \ldots, x_N\}$, calculating the inner product of instances according to a kernel function in a SVM results in a $N \times N$ kernel matrix. When we sample a small subset of instances from SV_i, we obtain a different and smaller kernel matrix which may cause an unexpected shift as shown in Figure 5.2.

This example illustrates the sensitivity of the decision boundary to one SV instance when it is removed from a set of SVs collected in Stage 1. On the left of Figure 5.2 are the decision boundaries defined by the entire set of SVs. On the right side are the new boundaries after arbitrarily removing one instance of a majority support vector (the square in (a) indicates the instance removed).



(a) decision boundary with all support vectors

(b) decision boundary after removing one majority support vector instance

Figure 5.2 Boundary sensitivity to removing one SV instance

Since this set is relatively small, removing an instance causes a large boundary shift. In the case of a relatively small SV instance set of size *n*, we could enumerate all possible decision boundaries ($2^n - 1$) and choose the best one. However, when dealing with large datasets, this is not practical. Instead of enumeration, a metaheuristic instance sampling approach is proposed. We use a GA-based SV selection (GA-SS) in Stage 2. GA-SS uses the same fitness function described in Chapter 4 that was used in the GA-IS method.

The criterion for removing instances from each $SV_i \in \widetilde{SV}$ is the effect on the G-mean for the original training dataset, T_0. Using GA-SS, we want to find SV^* such that

$$SV^* = \arg\max\{g(svm(SV_j), T_0) | SV_j \subset SV_i, \forall SV_i \in \widetilde{SV}\}$$

The GA-SS algorithm is described in Figure 5.3

GASS(SV_i,T_0){

T_0	The entire training set
SV_i	Candidate SV set from $\widetilde{SV}$ produced in Stage 1
SV_i^+	Set of positive (minority) *SVs* in SV_i
SV_i^-	Set of negative (majority) *SVs* in SV_i
C_i^k	Binary chromosome representation of SV_i^- in population *k*
M_k	Number of individual chromosomes in population *k*
MAXGEN	Maximum number of generations
$sel.SS_i^k$	Instance set that is combined with SV_i^+ and a set of SV_i^- represented by C_i^k
$fitness_i^k$	Fitness function, G-mean of T_0
selection	Selection method in selection operator in GA

Initialize: *k*=0;

Generate an initial population, $p^0 = \{C_1^0, C_2^0, C_3^0, \dots, C_{M_k}^0\}$

while *k* < *MAXGEN*

for *i* ← 1 to M_k

$fitness_i^k \leftarrow f(svm(sel.SS_i^k), T_0)$

endfor

/***** new chromosomes produced by GA operators*/

$P^{k+1} = P^k$

$C_i^k = select('selection', P^k);$

$(C_i^{k+1}, C_j^k) = crossover(C_i^k, C_j^k);$

$C_i^{k+1} = mutate(C_i^{k+1});$

k = *k*+1;

endwhile

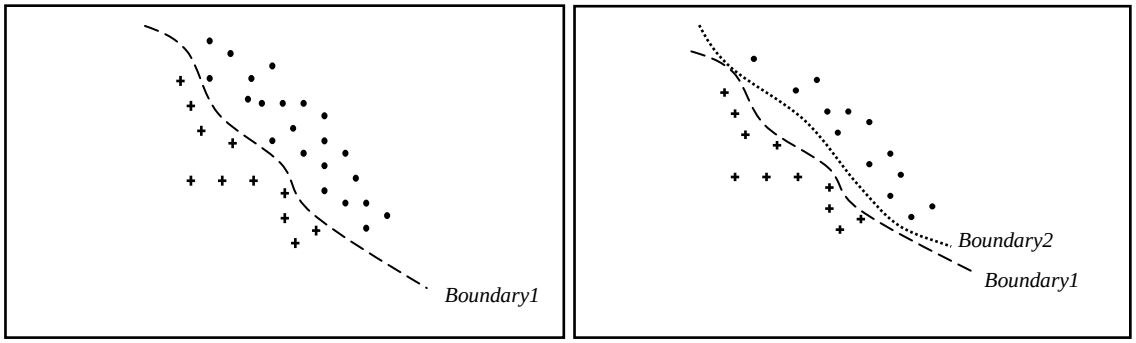
return $max(\ sel.SS_i^k,\ g(svm(sel.SS_i^k), T_0))$ /* select an instance set of support vectors that maximize the fitness function at the final generation*/

}

Figure 5.3 GA-SS Algorithm

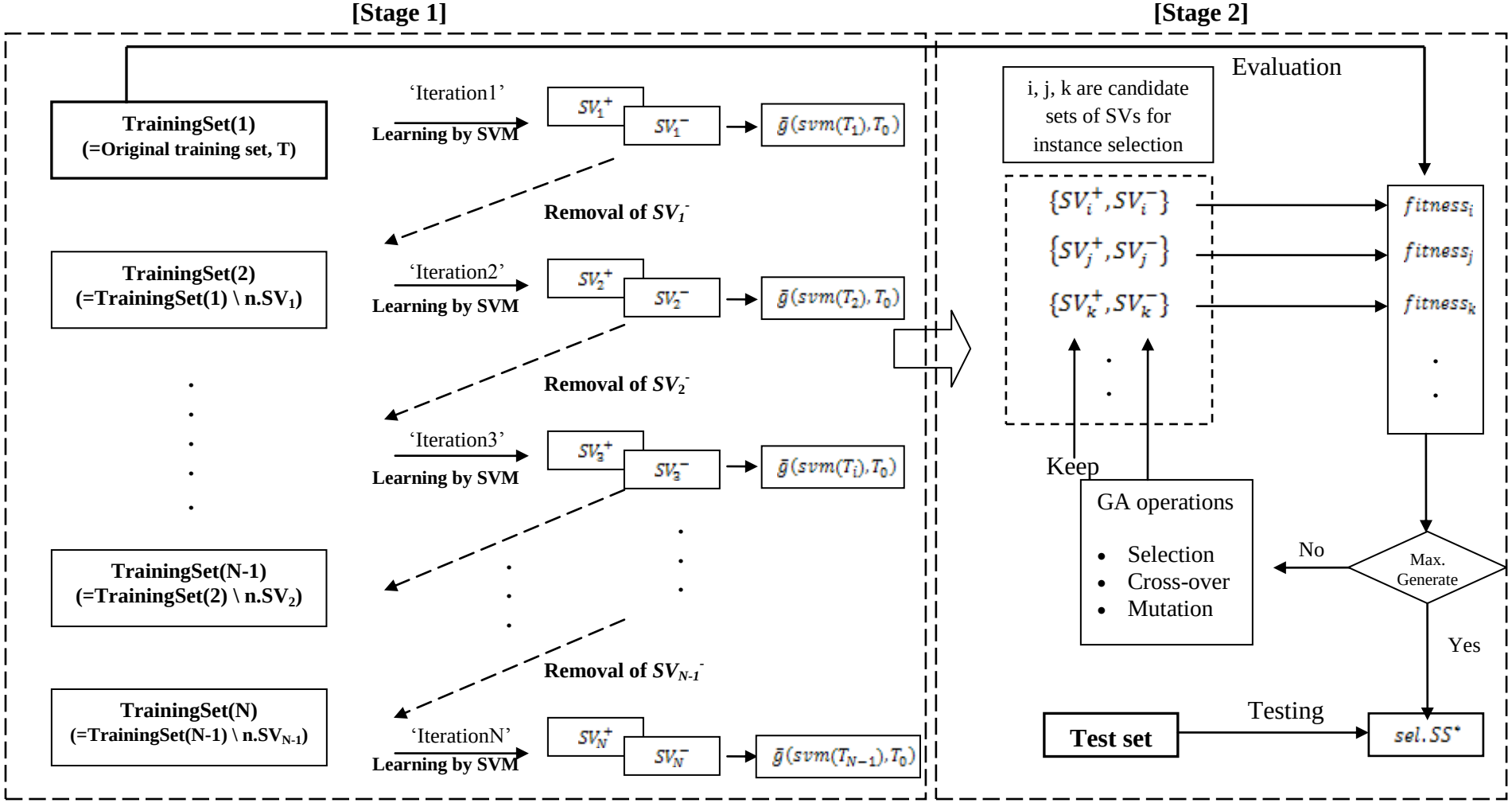
Given a decision boundary (*'Boundary1'*) by SVs at Stage 1 after eliminating SVs of the majority instances, as seen in Figure 5.4(a), then in Stage 2, we select only majority SVs based on GA-based selection method for a better boundary ('*Boundary2*') as seen in (b).

The overall procedure for Stages 1 and 2 is described in Figure 5.5.



(a) Boundary in Stage1 (b). Boundary determined in Stage 2

Figure 5.4 Boundary movement through selecting instances in Stage2



SV_i^+: SVs of the positive (minority) class SV_i^-: SVs of the negative (majority) class. T_0: Original training set

$fitness_j$:fitness function (G-mean of the original training set, T_0)

$sel.SS^*$: Final Instance set that maximizes the G-mean value of the original training set (T_0) through GA-based instance selection

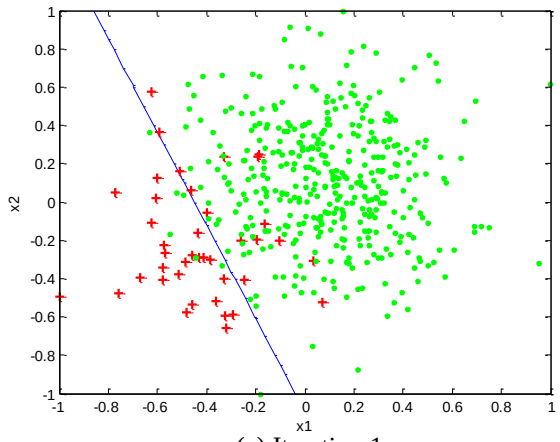
Figure 5.5 Overall procedure of our approach for imbalanced training datasets

5.2 Demonstration of GA-SS

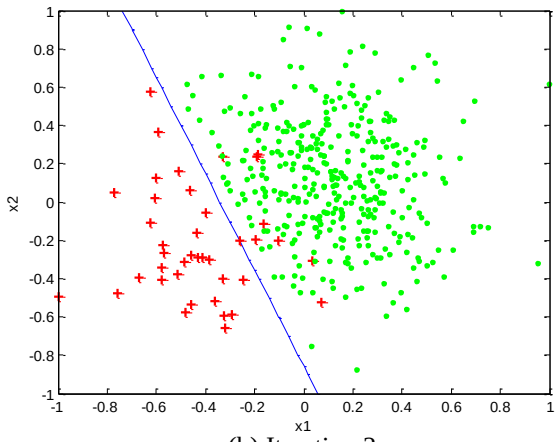
In general, classification performance varies according to which kernel function is used within the SVM. In our research, a Gaussian radial-based function was used due to its outstanding classification performance in most applications. However, in order to demonstrate boundary shifting in our methodology, we also make use of a linear kernel function in this section. Here, the proposed method is demonstrated using a simple artificial binary-class problem that has been used in previous chapters.

5.2.1 Linear kernel function case

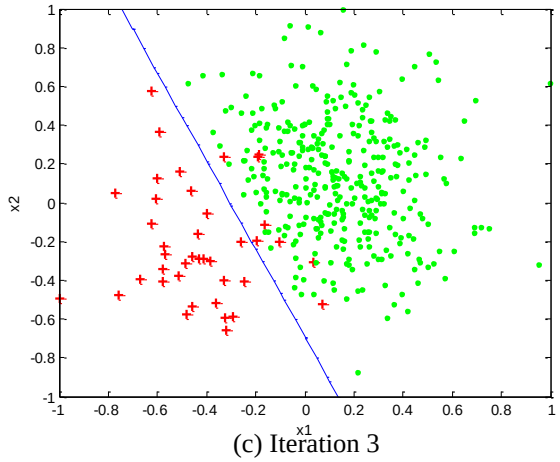
Figure 5.6 shows the boundary shifting from iterations 1 to 4 in Stage 1. SVs from the majority class (•) are iteratively removed after classification with the linear kernel function shifting the boundary for the next iteration toward the majority class.



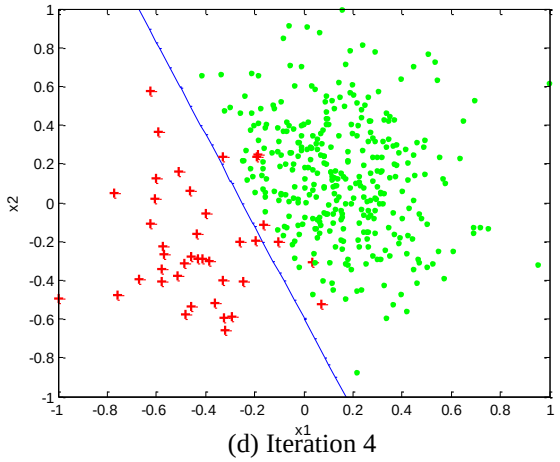
(a) Iteration 1



(b) Iteration 2



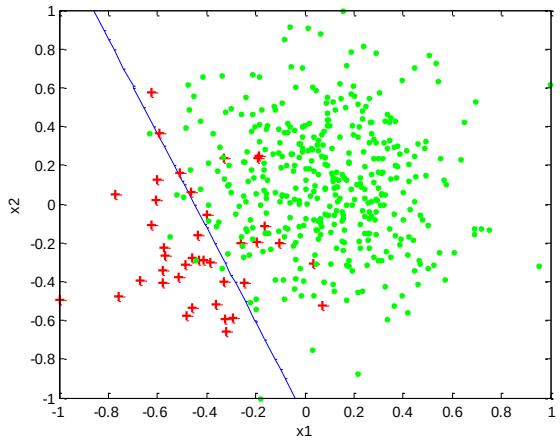
(c) Iteration 3



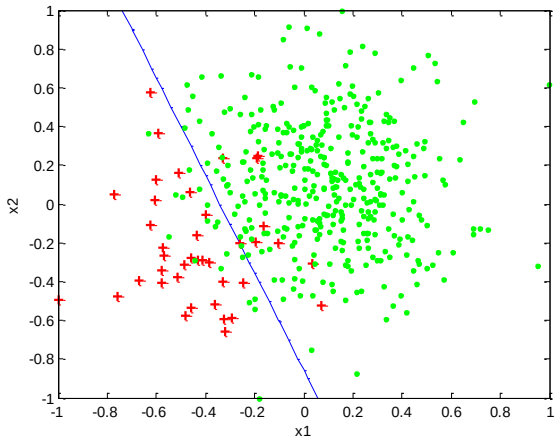
(d) Iteration 4

Figure 5.6 Decision boundaries at each iteration through selecting instances from SVs of the majority class (linear kernel function)

The boundary shift by eliminating SVs of the majority class affects the G-mean for the original training dataset. Figure 5.7 presents the boundary shift through two sequential iteratrions.



(a). Iteration 1



(b). Iteration 2

Figure 5.7 Mapping decision boundaries at Iteration 1 and 2 on the original training set (Linear kernel function case)

The shift in the decision boundary increased the G-mean value for the original training set from 0.7836 to 0.8798 by achieving greater classification accuracy for the minority class.

For Stage2 the population size was set to 40 individual chromosomes and 50 generations was the stopping point for the Genetic Algorithm. Figure 5.8 below shows a new decision boundary that are defined by learning the selected data sets and maxmize the G-mean values of the original imbalanced data set after completing Stage 2 with 4 candidate subsets. Three linear lines in Figure 5.8 represents the decision boundaries that are determined by the original skewed dataset, a reduced dataset from Stage1, and selected learning dataset (marked with circle) through Stage2 respectively.

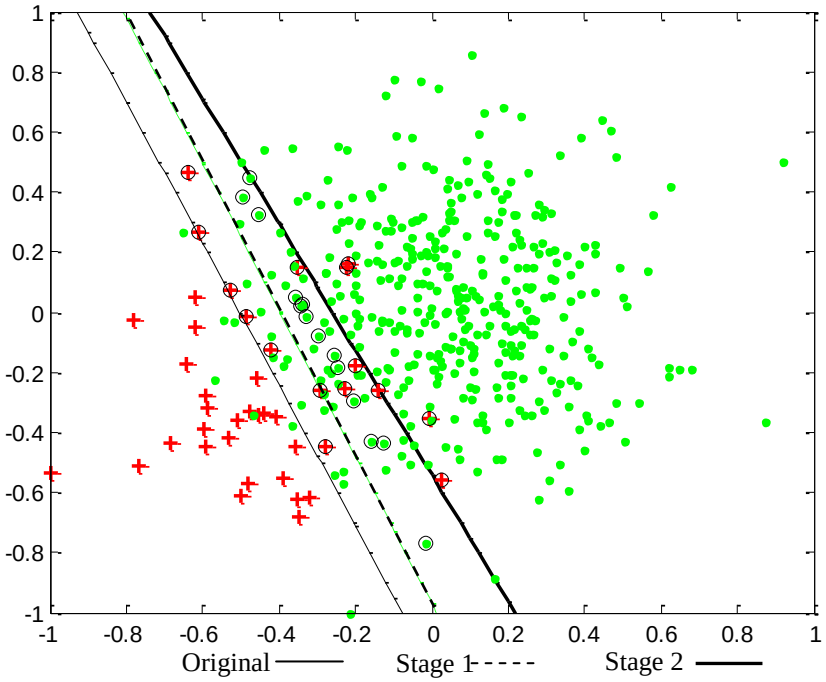


Figure 5.8 A decision boundary that produces the maximum G-mean for the original training set using GA-SS
(Linear kernel function)

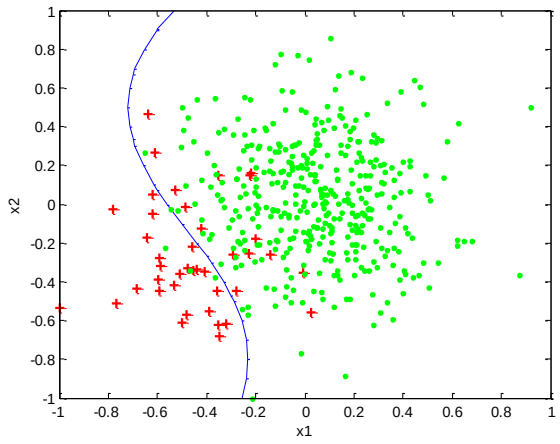
Based on our proposed methodology, a good decision boundary was generated seen in the change in the G-mean value from 0.7836 to 0.8811 (about a 13 % improvement). Furthermore,

this boundary was obtained with a relatively small training set, 29 instances (15 minority instances and 14 majority instances).

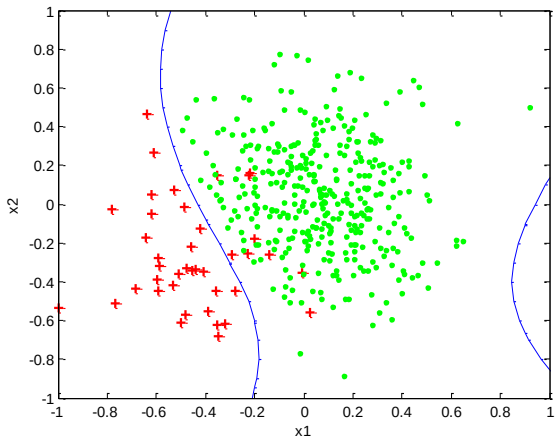
5.2.2 Gaussian radial-based kernel function case

Due to consistently good performance of the radial-based kernel function in most applications, a Gaussian radial-based kernel was used in our research. To demonstrate the effects of a Gaussian radial-based kernel function (C=100, $\sigma = 1$) on the boundary shift, we performed 4 iterations in Stage 1.

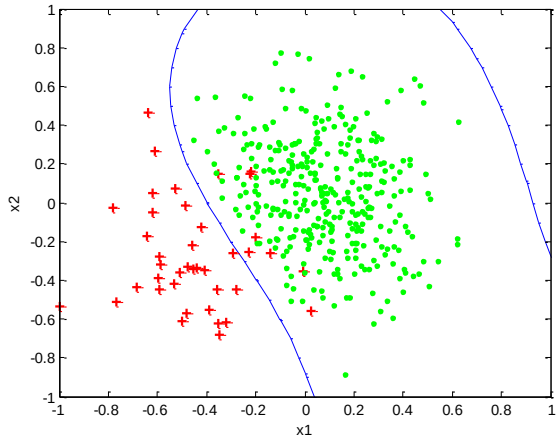
As seen in Figure 5.9, the elimination of SVs of the major class made the classification boundary gradually shift towards the majority class.



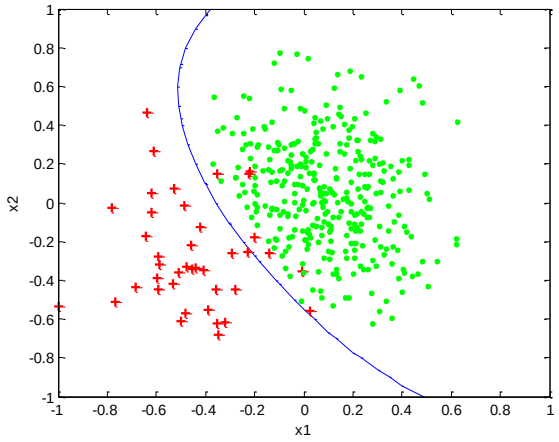
(a) Iteration 1



(b) Iteration 2



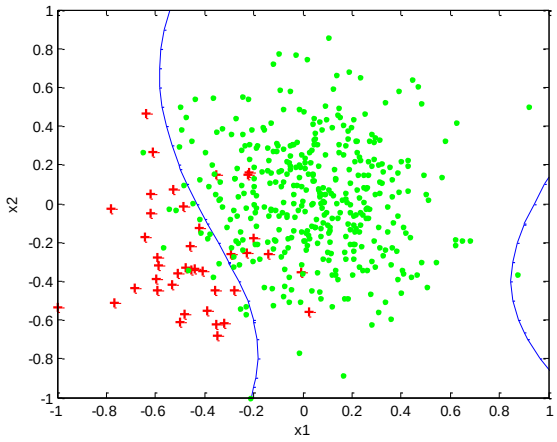
(c) Iteration 3



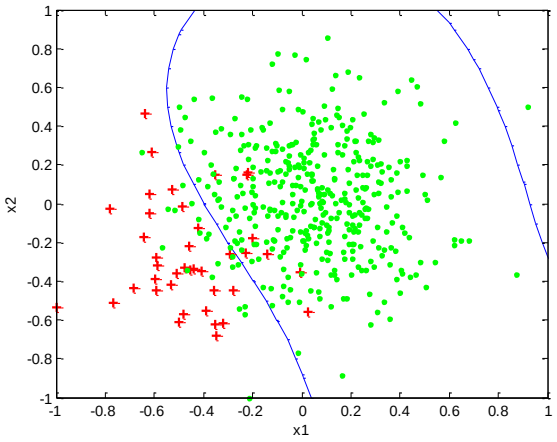
(d) Iteration 4

Figure 5.9 Decision boundaries for Stage 1 iterations
(Gaussian Radial-based kernel function)

Figure 5.10 shows the mapping of decision boundaries for iterations 2 and 3 on the original training set. The results indicate that removing SVs of the majority class shifted the decision boundary and increased the G-mean value for the original training set.



(a) Iteration 2



(b) Iteration 3

Figure 5.10 Mapping decision boundaries for Iterations 2 and 3 on the original training dataset
(Gaussian Radial-based kernel function)

After GA-IS in Stage 2, a good set of instances were found in SV_2. The resulting decision boundary is displayed in Figure 5.11. The G-mean value has improved from 0.7876 to

0.8923with 32 instances (minority: 16 and majority 16). The three curves in Figure 5.11 show the decision boundaries for the original skewed dataset, SV_2 a reduced dataset from Stage 1, and SV^* selected learning dataset (marked with circles) from Stage 2.

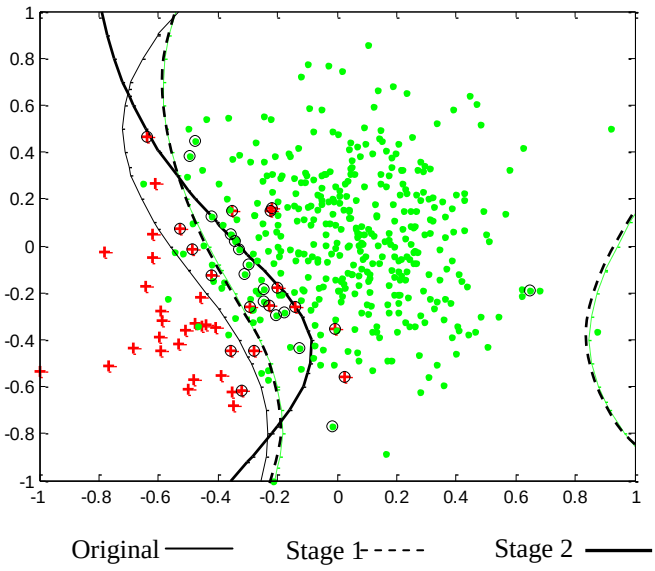
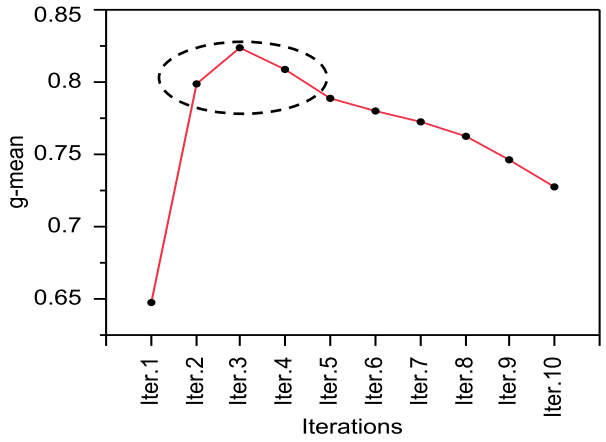


Figure 5.11 Decision boundary that produces the maximum G-mean of the original training set through GA-SS
(Gaussian Radial based kernel function)

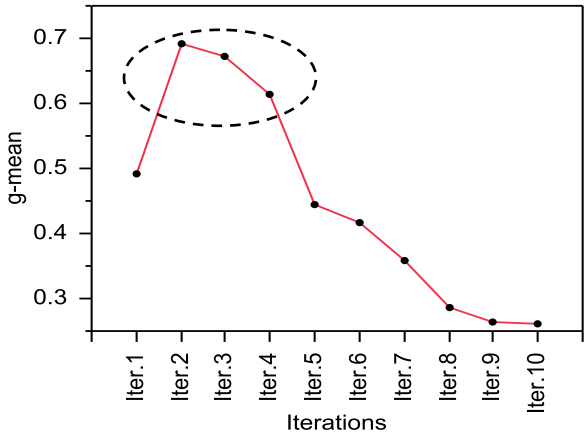
Through the experiments with a simple artificial dataset applying both linear and a Gaussian radial-based kernel function to the SVM classifier, we can observe the effects of GA-SS on the boundary and significant increases in the G-mean value for the original training set.

5.3 Experiments with real datasets

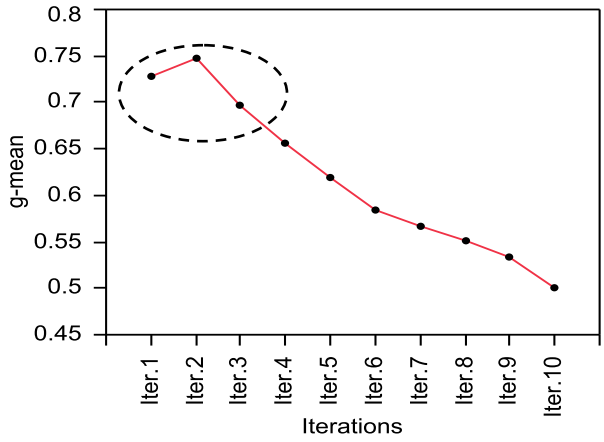
Using the datasets discussed in 4.3.1, Stage 1 of the method was performed for 10 iterations on each of the datasets. Figure 5.12 presents trends for the G-mean of the original training set after Stage 1 was completed.



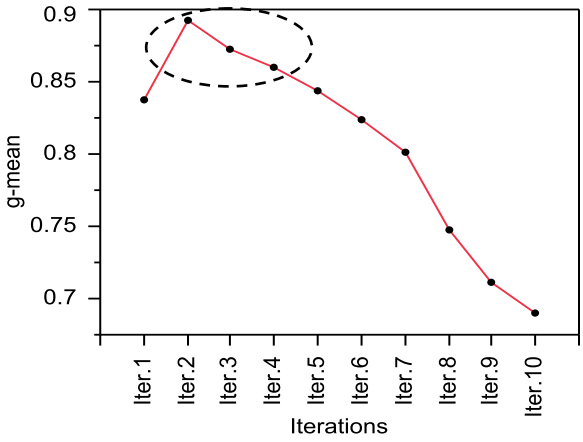
(a) abalone training dataset



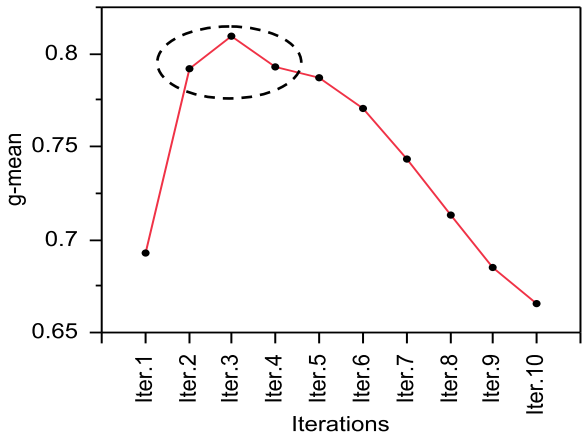
(b) blood training dataset



(c) pima training dataset



(d) wine training dataset

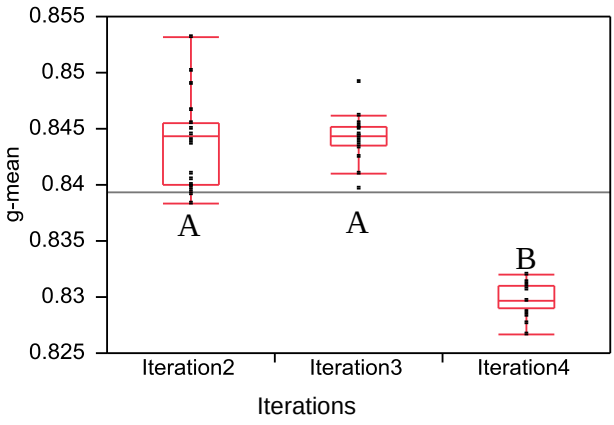


(e) yeast training dataset

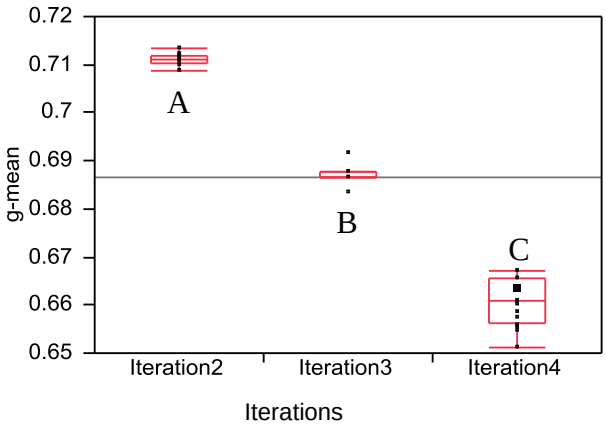
Figure 5.12 Trend of G-mean values of the original training set on reduction of the majority instances in Stage1

As seen in Figure 5.12, removing SVs increases G-mean values up to a point at which the trend reverses because too many instances have been removed for the majority class. For Stage 1 we found that the maximum G-mean values were obtained at iteration 3 for *abalone*, iteration 2 for *blood transfusion*, iteration 2 for *pima*, iteration 2 for *wine*, and iteration 3 for *yeast* dataset. The dashed oval is used to indicate the sets of SVs producing the 3 highest G-mean values. Since we do not guarantee that the set of SVs that showed the highest G-mean in Stage 1 will produce a training set with the highest G-mean in Stage 2, all three sets of SVs are processed separately in Stage 2.

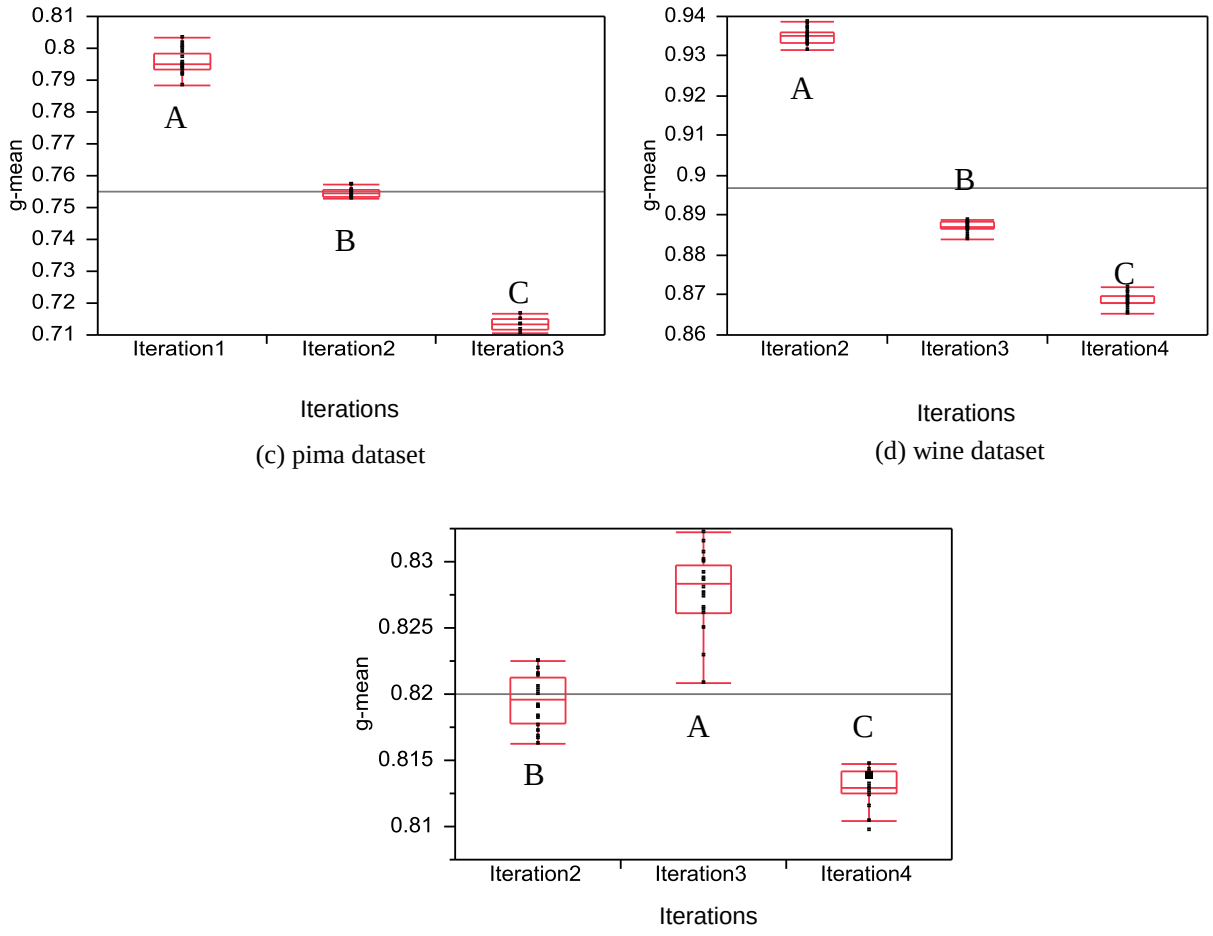
Given the nature of GA-based instance selection, GA-SS was conducted 20 times for each $SV_i \in \widetilde{SV}$. Figure 5.13 below displays box plots of the G-mean of the training set of 20 runs corresponding to the datasets from iterations in Stage1.



(a) abalone dataset



(b) blood transfusion dataset



(c) pima dataset

(d) wine dataset

(e) yeast dataset

Figure 5.13 Box plots of G-mean of the training set after Stage 2 for $SV_i \in \widetilde{SV}$
(A, B and C represents groups associated with comparision of mean difference in a pair.)

Results of one-way ANOVA indicated significant differences in G-mean values (see Appendix E). After ANOVA, we also observed which iterations are different from each other using pairwise comparison with a post-hoc test (Tukey HSD). Significant differences were found and are represented by Groups A, B and C in Figure 5.13. In the case of *abalone* dataset, Iteration 2 and Iteration 3 did not show significant mean difference each other (Group A), but since one of our objectives is to have a smaller training set for classification, we prefer to take the instance set (97 instances) obtained from iteration 3 rather than iteration 2 (137 instances).

Table 5.1 shows the improvement of G-mean values and the size of the training set from Stage 1 to Stage 2. For example, in the *wine* dataset, the G-mean increased by about 5% (0.8925→0.9349) after Stage 2, while the size of the training set was reduced by 84% (1054→164).

Table 5.1 Improvement of G-mean and training set reduction after Stage2

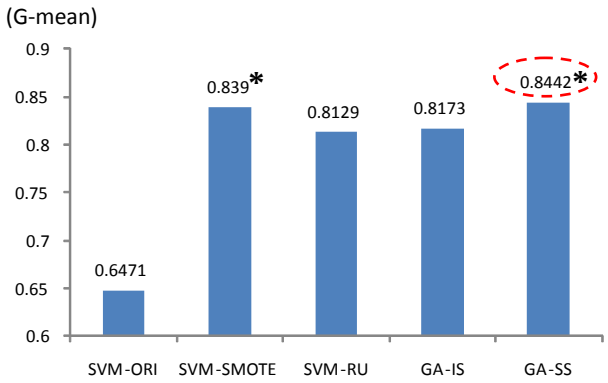
	G-mean of the original training set		Size of training set	
	Stage1	Stage2 (average)	Stage1	Stage2
abalone	0.8237	0.8442	502	84 (83%)
blood	0.6917	0.7111	387	137 (65%)
pima	0.7284	0.7957	457	120 (74%)
wine	0.8925	0.9349	1054	164 (84%)
yeast	0.8093	0.8278	911	145 (84%)

% in Stage2 indicates a percentage of data reduction based on size of training set in Stage 1

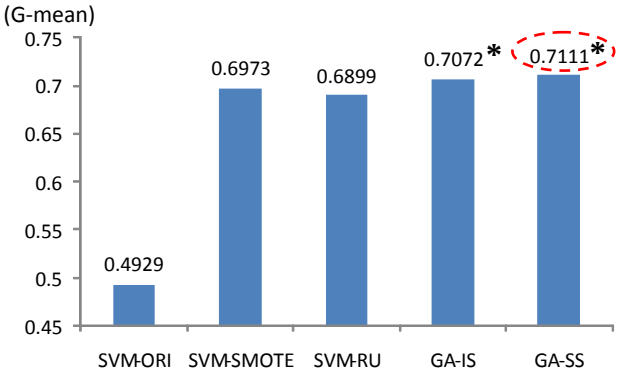
5.4 Summary and Discussions

In this chapter, we have introduced a new metaheuristic approach to imbalanced data learning with SVMs. The basis of this approach is to iteratively remove majority instances from the original traning set and then select SVs from the remaining majority instances that maximize the G-mean value using a Genetic Algorithm. In that SVs are a smaller subset of the learning set which is located outside of classes, this set of SVs does not follow the original data distribution of the class. Thus, arbitrarily removing even one of SVs could result in a totally different decision compared with a previous one, as seen in Figure 5.2. At this point, instanc selection based on Genetic Algorithm in Stage2 could find an optimal training set for imbalanced data learning, further determined a relatively small training set for SVM classifier.

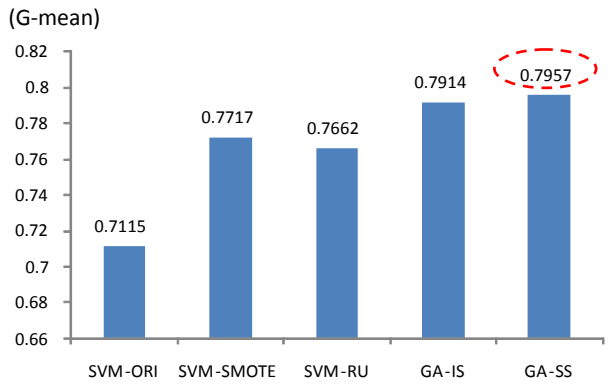
A comparison of the performances of all 5 different approaches based on the G-mean of the original training set showed that GA-SS had the best performance in the 5 experimental datasets (see Figure 5.14).



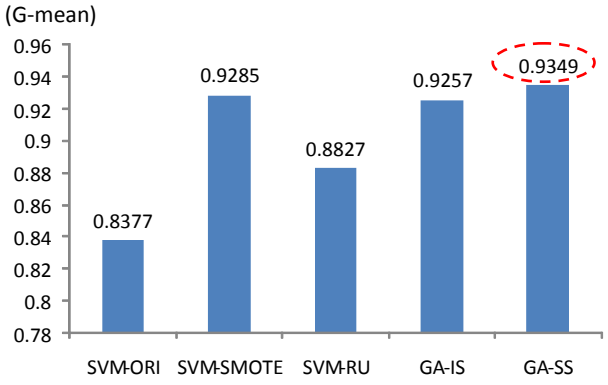
(a) abalone dataset



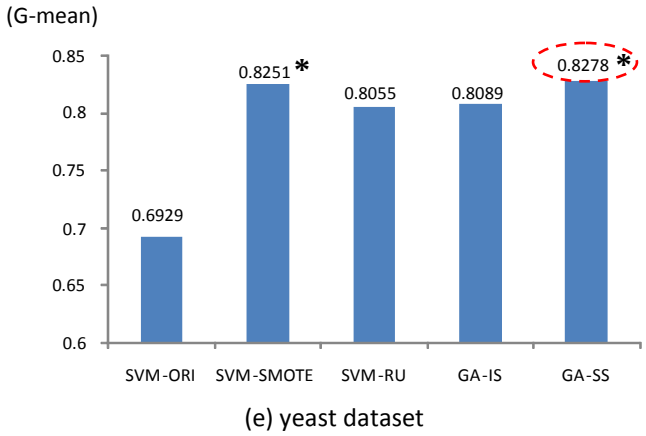
(b) blood transfusion dataset



(c) pima dataset



(d) wine dataset



SVM-ORI means learning on SVM with the original training set

Figure 5.14 Comparison of G-mean values for the training sets (average) for 5 different methods

Analysis of the results showed that differences between GA-SS and other methods were not statistically significant in some cases. This included SVM-SMOTE for the abalone and yeast datasets and GA-IS in the blood transfusion dataset. In other words, the GA-SS approach performance was comparable in these cases (which are marked with a ***** in Figure 5.14).

GA-SS approach produces smaller training sets as compared with SVM-SMOTE and GA-IS. Figure 5.15 shows the sizes of the training sets for SVM-SMOTE, GA-IS and GA-SS which had high classification accuracy. As expected, smaller sets use less memory and have shorter learning times, which can be seen in the results of measuring the learning time of training sets (see Figure 5.16).

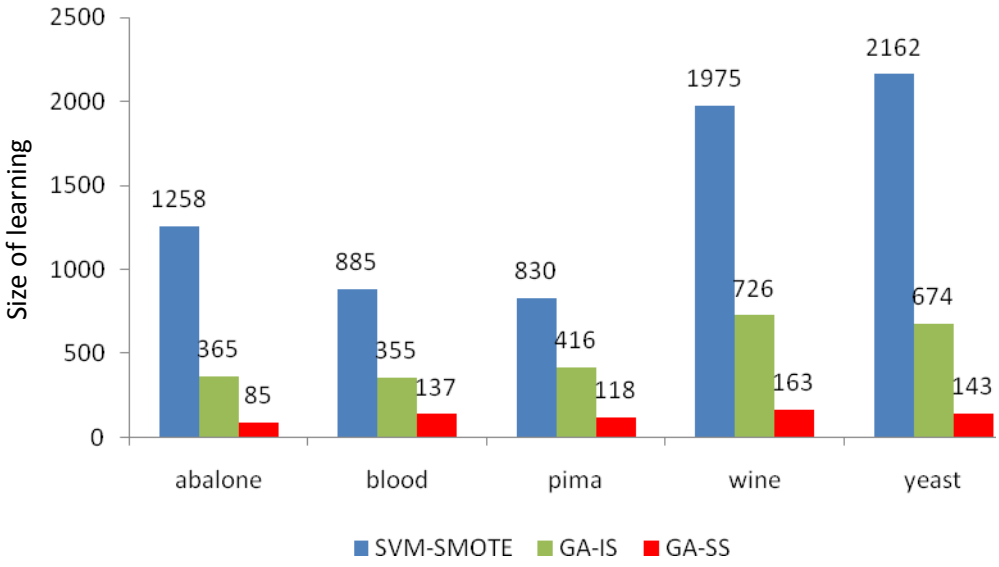


Figure 5.15 Size of training sets

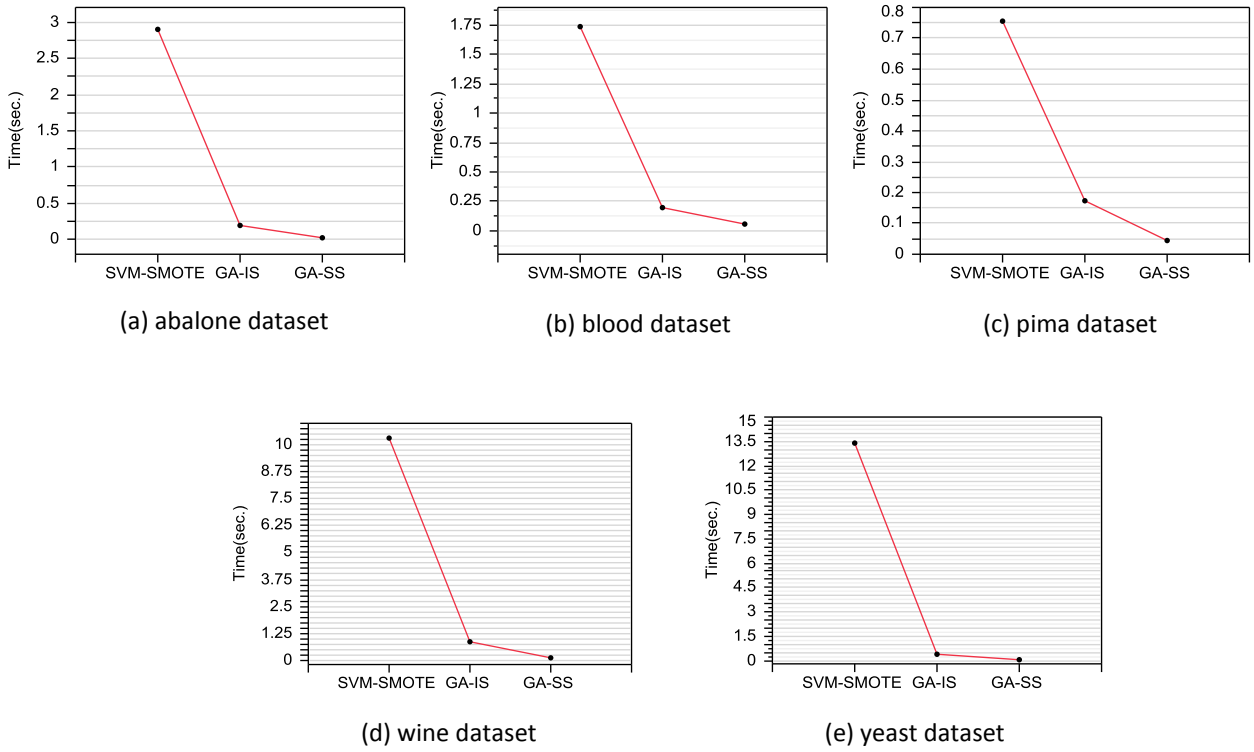


Figure 5.16 Comparison of learning time

Comparing learning time for SVM-SMOTE and GA-SS, it was observed that GA-SS showed outstanding performance. For example, for the *yeast* dataset, GA-SS (0.06 seconds) is about 207 times faster than that from SVM-SMOTE (13.36 seconds).

In order to evaluate decision boundaries after training, we determined the G-mean values after classification of independent test datasets which were separated from the each UCI dataset according to its original class distribution as depicted in Figure 4.5 in Chapter 4. Table 5.2 shows the average G-mean values of 20 runs.

Table 5.2 Average G-mean of Test sets in terms of 5 different methods in 20 runs

	G-mean of Test set					
	SVM with test sets	SVM-SMOTE	SVM-RU	Biased Penalty C	GA-IS	GA-SS
abalone	0.4791	0.7824	0.7882	0.7644	0.7127	**0.8296**
blood	0.4433	0.6819	0.6771	0.6791	0.6685	**0.7150**
pima	0.6733	**0.7663**	0.7583	0.7542	0.757	0.7524
wine	0.6784	0.8061	0.8144	0.8185	0.7683	**0.8447**
yeast	0.6467	0.7982	0.794	0.8022	0.739	**0.8112**

We found that the decision boudnaries made by GA-SS approach had the highest accuracy in 4 of 5 test datasets. In addition, GA-SS produces much smaller training sets as compared with the SMOTE sampling approach.

CHAPTER 6 CONCLUSIONS

Supervised learning of imbalanced datasets continues to be an important research issue in the data mining and machine learning communities. A direct method to solve the imbalance problem is artificially balancing the class distributions. In this dissertation, a new methodology has been proposed for imbalanced data learning that simultaneously considers the aspects of effectiveness and efficiency for a SVM classifier. The research problems considered in this dissertation are (1) the existing rebalancing approaches require an empirical process to find an optimal class distribution by grid search and (2) in spite of its good performance, the over-sampling algorithm, SMOTE has computational difficulties when used with a SVM classifier.

The major contributions of this dissertation include the following items.

1. *Metaheuristic under-sampling*

Random under-sampling is a common under-sampling approach for rebalancing the dataset to obtain a better class distribution. Although random under-sampling of the majority data pushes the learned boundary toward the majority class, this may not bring a stable class distribution for imbalanced data learning due to its randomness. On the other hand, over-sampling approach based on SMOTE algorithm gives us a promising class distribution for imbalanced data learning on SVM. However, increasing the minority instances makes training sets significantly larger, which increases the classifier's computational load for optimization within the SVM classification algorithm. Consequently, use of these two rebalancing approaches can be problematic. These aspects motivated us to develop an alternative method that does not require the empirical steps of looking for an optimal class ratio in finding reduced training sets.

First assuming that all minority instances are informative, Genetic Algorithm based under-sampling of the majority instances is applied through formulating an optimization problem. Instead of grid search for finding a desired class ratio for imbalanced data learning, the Genetic Algorithm approach determined a near optimal training set for the SVM classifier.

2. *Informative and representative near optimal training sets*

As mentioned earlier this chapter, in terms of effectiveness and *e*fficiency in dealing with imbalanced datasets using a SVM classifier, we need to find training sets to solve this problem as well as take weakness of the learning algorithm, large-scale problem with a SVM, into consideration in determining training sets. The two hypotheses presented in Chapter 3 are the basic premises that are the foundation of the GA-SS method.

Hypothesis 1: *A relatively small number of instances from an imbalanced training set are needed to obtain good performance in solving the class imbalance problem using a SVM.*

Hypothesis 2: *A smaller subset within the set of SVs can be found that produces a better boundary using a SVM.*

Even though GA-IS can be used as an alternative approach for imbalanced data learning without rebalancing the class distribution by grid search, search time increases exponentially with the number of majority instances in order to reach at an optimal class distribution. To address this problem, we proposed a two stage methodology, GA-SS, which had good performance and resulted in a much smaller training set. This approach will be quite useful when the number of the majority instances is large. The idea is based on the well-known fact about SVMs that only SVs are necessary and other redundant samples (non-SVs) can be removed

without affecting classification. This fact allows us to explore the possibility of selecting small training sets from instances that have been named sets of SVs through learning gradually reduced training sets. Investigating GA-IS from the majority instances, this tends to search and select them as learning instances even though most of them do not affect classification results. In other words, since binary chromosomes contained all majority instances as possible solutions, after terminating GA, only those genes that are SVs affected the decision boundary. As a result, searching the whole space of the majority class in Genetic Algorithm will require a longer time to find an optimal training set. As compared with SMOTE algorithm by grid search in Chapter 4 (Table4.2), GA-IS had good performance in 3 (blood transfusion, pima, and wine) of the 5 UCI experimental datasets, while performances by SMOTE algorithm approach were better in 2 datasets (abalone and yeast).

To make the training set as small as possible for a SVM, we proposed a new method that selects only influential instances that affect decision boundaries. The motivation of this approach is that by making the search space smaller for selecting instances, we expect to find a better near optimal training set and at the same time, have a much smaller training set. This was achieved by iteratively using the SVM classifier to obtain good candidate subsets (sets of SVs) which were then removed. Given that the initial imbalanced class distribution causes an imbalanced ratio of SVs, we suppose that all SVs of the minority class are informative as assumed in GA-IS method. Instead we select and remove instances from SVs in the majority class.

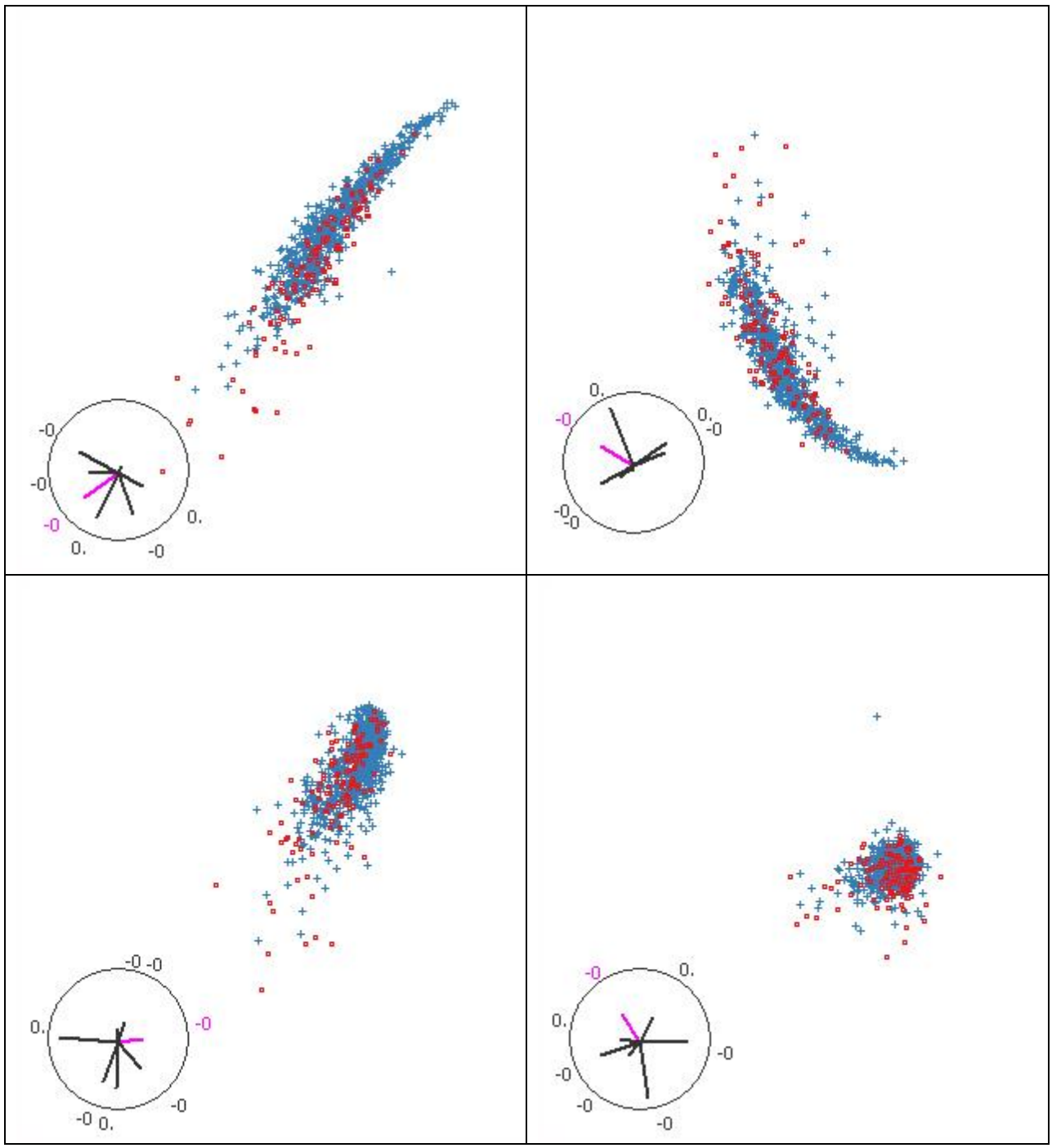
Since we collect just a small portion of instances which are SVs from the training sets, eliminating even an instance from a SV set could cause a major shift in the decision boundary. This is another reason why a metaheuristic approach was used to find near optimal training sets instead of enumerating all possible decision boundaries. Compared with 2 sampling approaches

(SMOTE and random under-sampling) and GA-IS, GA-SS showed overall better performances (G-mean of the training sets and test sets) in all experimental datasets. In addition, GA-SS produced relatively small optimal training sets.

Although our methodology produces smaller training sets, the final training sets through Genetic Algorithm based instance selection still contain dummy instances which are non-support vectors. Future research should consider how to find better training sets by detecting dummy instances. Some combinations of heuristic and non-heuristic instance selection approaches should be studied for SVM classifiers. The G-mean metric was used in the fitness function of the Genetic Algorithm and as a measure of inductive bias for imbalanced data learning in this dissertation. Given that class labels for learning data are classified by a hyperplane between two different classes and predicted values of data represent distance from an optimal hyperplane, future work should consider using the distance and predicted class label as part of a more robust fitness function in GA-based.

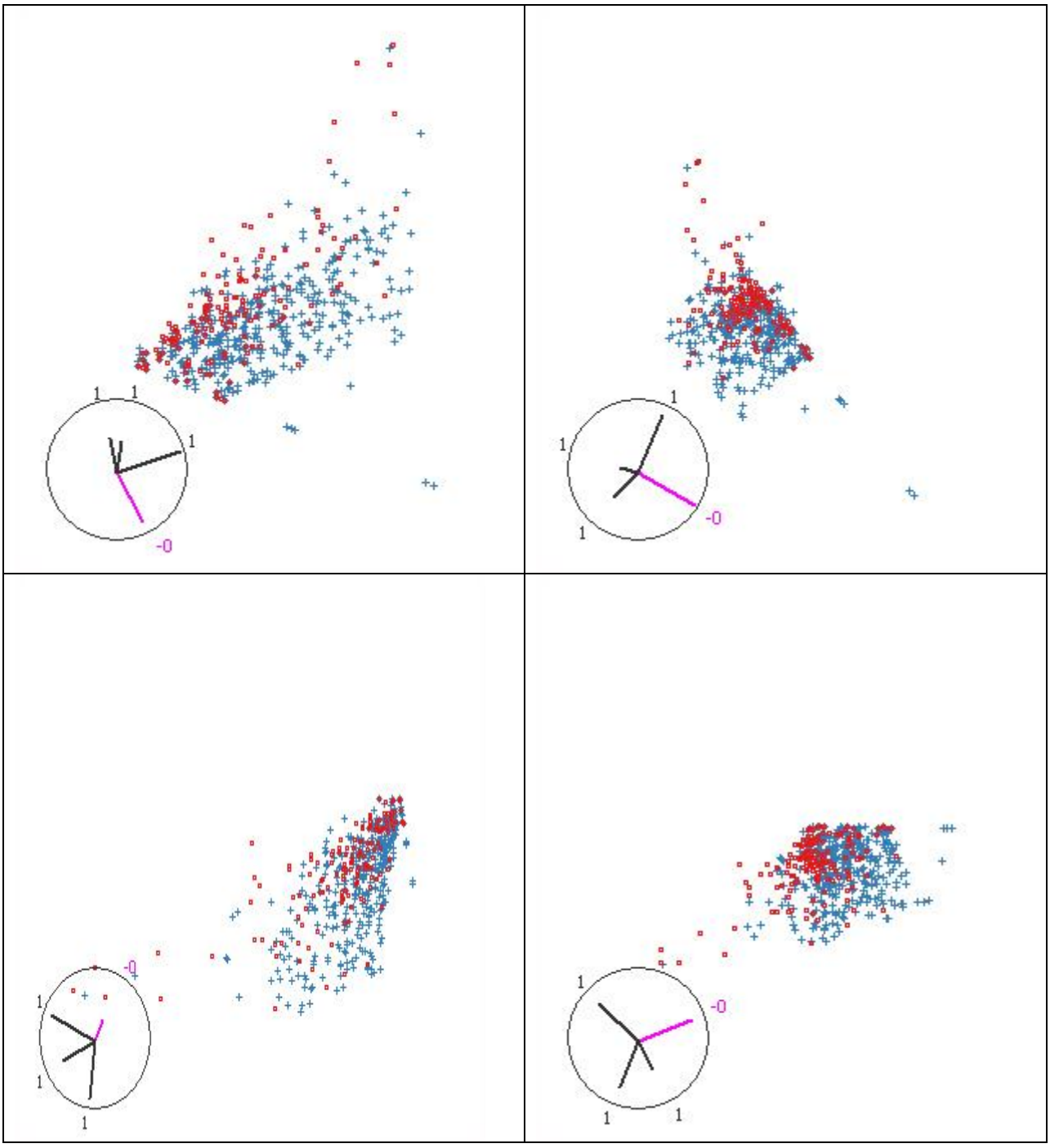
APPENDIX A. EXPERIMENTAL IMBALANCED TRAINING SETS

1. abalone dataset



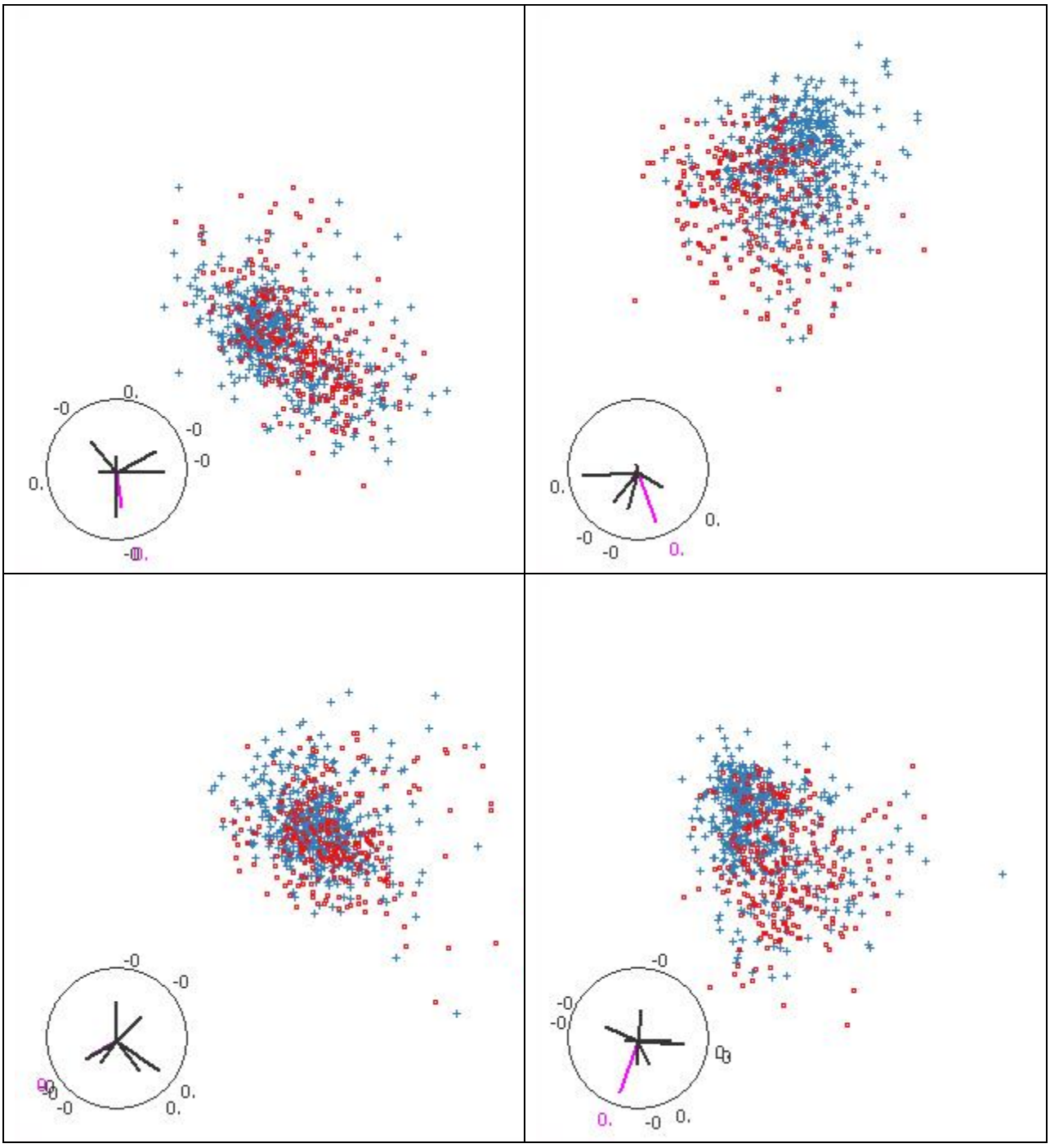
APPENDIX A. EXPERIMENTAL IMBALANCED TRAINING SETS (CONTINUED)

2. blood transfusion dataset



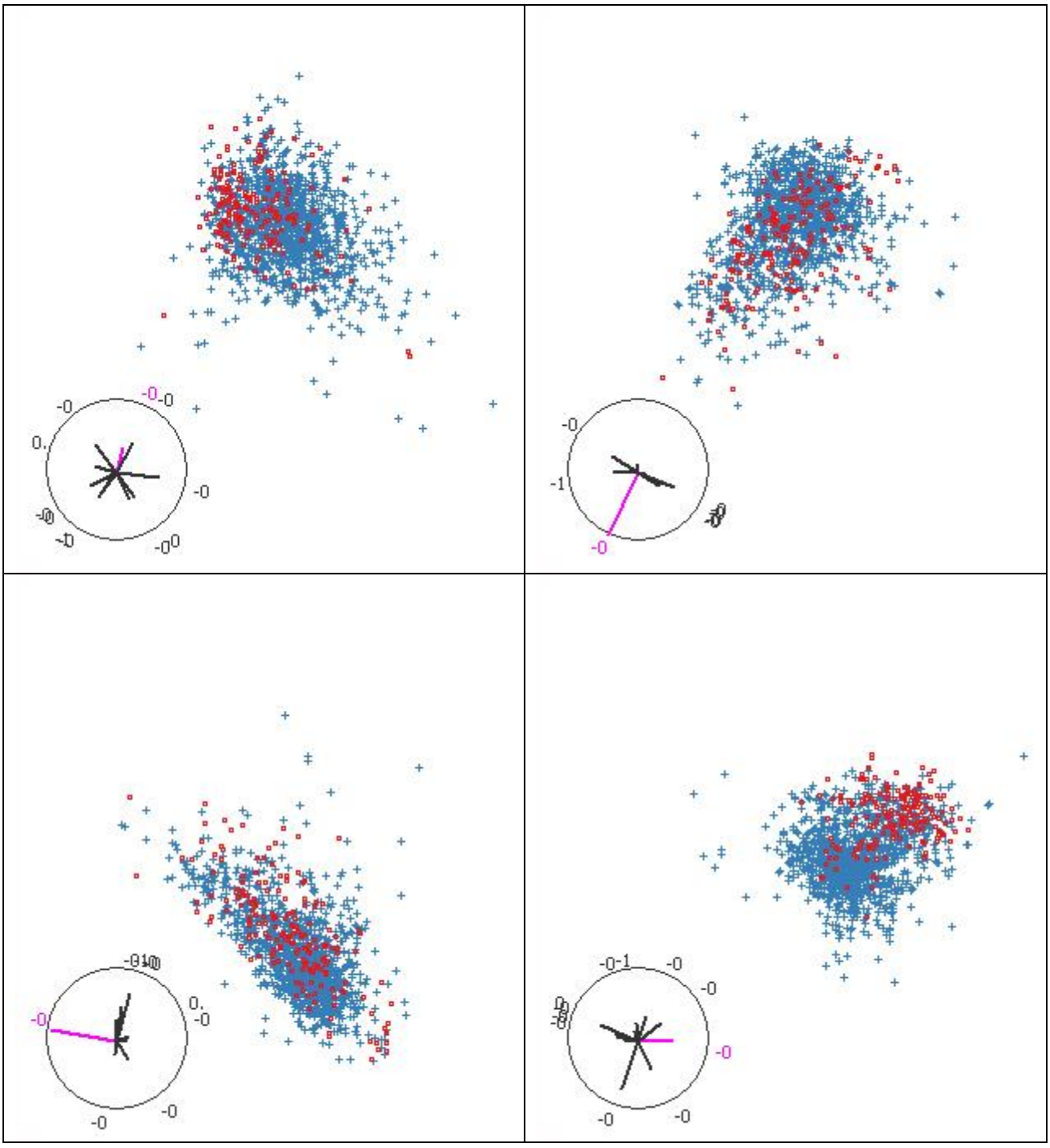
APPENDIX A. EXPERIMENTAL IMBALANCED TRAINING SETS (CONTINUED)

3. Pima dataset



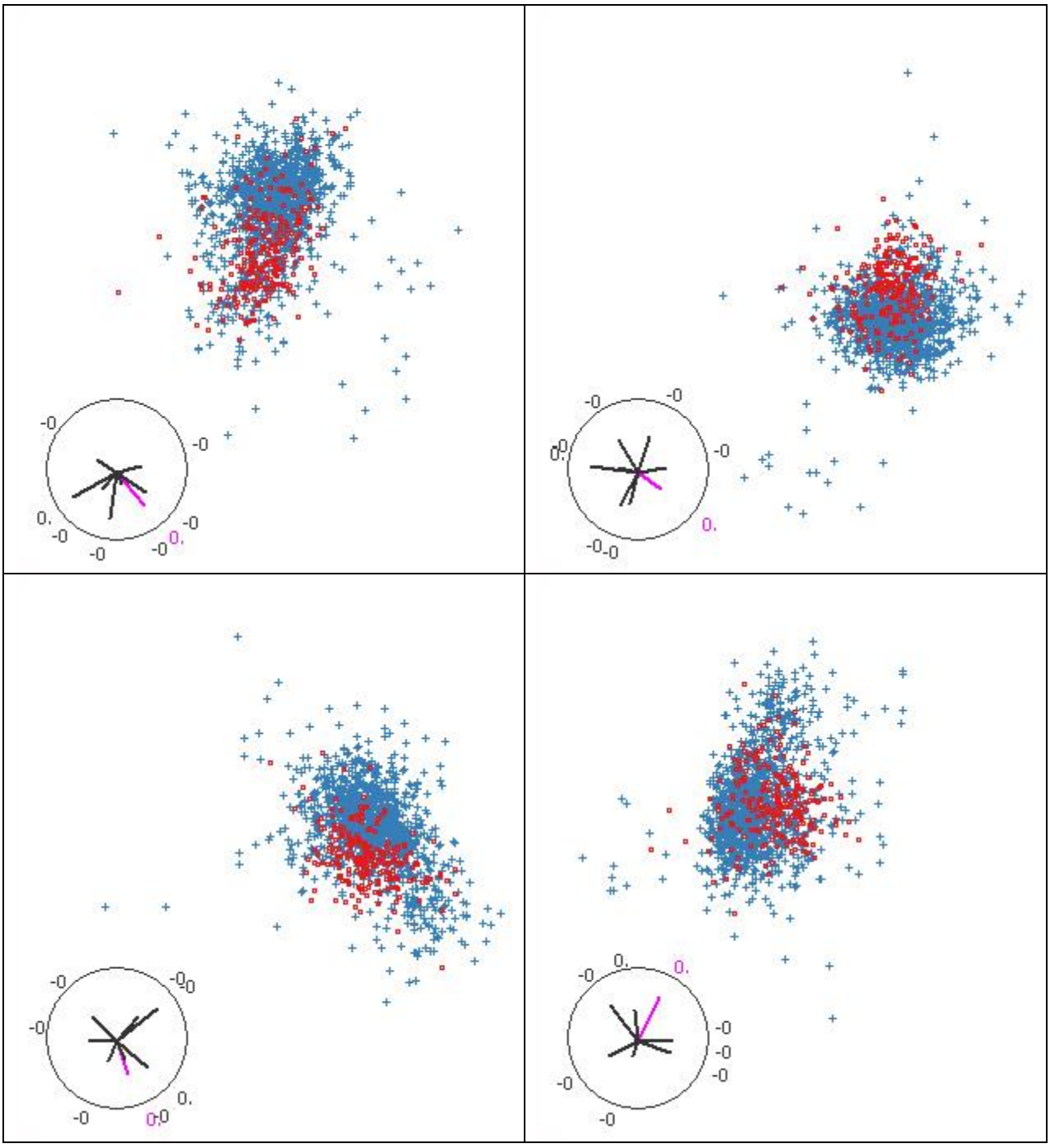
APPENDIX A. EXPERIMENTAL IMBALANCED TRAINING SETS (CONTINUED)

4. wine dataset



APPENDIX A. EXPERIMENTAL IMBALANCED TRAINING SETS (CONTINUED)

5. yeast dataset



APPENDIX B. PARAMETER C AND σ SETTING THROUGH 5-FOLD CROSS VALIDATION

1. *abalone* dataset

(Classification accuracy)

C	σ					
	0.1	0.2	0.5	**1**	2	3
1	0.8503	0.8552	0.8626	0.8515	0.8454	0.8454
10	0.8246	0.8540	0.8663	0.8687	0.8638	0.8528
100	0.8172	0.8209	0.8589	**0.8798**	0.8699	0.8748
200	0.8172	0.8099	0.8577	0.8748	0.8736	0.8736
500	0.8172	0.8111	0.8528	0.8724	0.8773	0.8712
1000	0.8172	0.8135	0.8454	0.8651	0.8761	0.8748

2. *blood transfusion* dataset

(Classification accuracy)

C	σ					
	0.1	0.2	**0.5**	1	2	3
1	0.7754	0.7901	0.7687	0.7674	0.7620	0.7620
10	0.7272	0.7754	0.7874	0.7687	0.7647	0.7661
100	0.6992	0.7380	0.7928	0.7781	0.7741	0.7674
200	0.6992	0.7259	**0.7941**	0.7861	0.7701	0.7701
500	0.7072	0.7286	0.7888	0.7874	0.7701	0.7741
1000	0.7059	0.7152	0.7821	0.7848	0.7714	0.7727

3. *pima* dataset

(Classification accuracy)

C	σ					
	0.1	0.2	0.5	1	2	**3**
1	0.6497	0.6796	0.7668	0.7734	0.7682	0.7669
10	0.6497	0.6771	0.7330	0.7512	0.7760	0.7669
100	0.6497	0.6771	0.7044	0.7447	0.7538	**0.7695**
200	0.6497	0.6771	0.7044	0.7382	0.7486	0.7617
500	0.6497	0.6771	0.7017	0.7278	0.7474	0.7525
1000	0.6497	0.6771	0.7017	0.7161	0.7382	0.7591

APPENDIX B. PARAMETER C AND σ SETTING THROUGH 5-FOLD CLOSS VALIDATION (CONTINUED)

4. *wine* dataset

(Classification accuracy)

C	σ					
	0.1	0.2	0.5	**1**	2	3
1	0.8718	0.8993	0.8855	0.8999	0.8643	0.8643
10	0.8780	0.9006	0.8899	0.9031	0.8724	0.8637
100	0.8849	0.8999	0.8887	**0.9031**	0.8812	0.8818
200	0.8868	0.8999	0.8856	0.9031	0.8793	0.8812
500	0.8824	0.8999	0.8874	0.9031	0.8768	0.8812
1000	0.8780	0.8999	0.8862	0.9031	0.8799	0.8830

5. *yeast* dataset

(Classification accuracy)

C	σ					
	0.1	0.2	0.5	**1**	2	3
1	0.8726	0.8868	0.8881	0.8821	0.8390	0.8356
10	0.8538	0.8720	0.8895	0.8828	0.8841	0.8780
100	0.8491	0.8437	0.8861	**0.8908**	0.8787	0.8841
200	0.8497	0.8416	0.8780	0.8895	0.8807	0.8801
500	0.8497	0.8234	0.8679	0.8868	0.8814	0.8767
1000	0.8497	0.8268	0.8605	0.8888	0.8794	0.8814

APPENDIX C. INITIAL CLASSIFICATION RESULTS on SVM LEARNING WITH THE ORIGINAL TRAINING AND TEST DATASETS

1. *abalone* dataset

Training		Predicted	
		abalone9	abalone14
Actual	abalone9	43	58
	abalone14	9	542
	overall accuracy=585/652=0.8972		
	accuracy of abalone9=43/101=0.4257		
	accuracy of abalone14=542/551=0.9837		
	g-mean=0.6471		

Test		Predicted	
		abalone9	abalone14
Actual	abalone9	6	19
	abalone14	6	132
	overall accuracy=138/163=0.8466		
	accuracy of abalone9=6/25=0.2400		
	accuracy of abalone14=132/138=0.9565		
	g-mean=0.4791		

2. *blood transfusion* dataset

Training		Predicted	
		Donate	Non-donate
Actual	Donate	36	107
	Non-donate	16	440
	overall accuracy=476/599=0.7947		
	accuracy of Donate=36/143=0.2517		
	accuracy of Non-donate=440/456=0.9649		
	g-mean=0.4929		

Test		Predicted	
		Donate	Non-donate
Actual	Donate	7	28
	Non-donate	2	112
	overall accuracy=119/149=0.7987		
	accuracy of Donate=7/35=0.2000		
	accuracy of Non-donate=112/114=0.9825		
	g-mean=0.4433		

3. *blood transfusion* dataset

Training		Predicted	
		positive	negative
Actual	positive	118	97
	negative	31	369
	overall accuracy=487/615=0.7919		
	accuracy of Positive=118/215=0.5488		
	accuracy of Negative=369/400=0.9225		
	g-mean=0.7115		

Test		Predicted	
		positive	negative
Actual	positive	27	26
	negative	11	87
	overall accuracy=116/153=0.7582		
	accuracy of positive=27/53=0.5094		
	accuracy of negative=89/100=0.8900		
	g-mean=0.6733		

APPENDIX C. INITIAL CLASSIFICATION RESULTS on SVM LEARNING WITH THE ORIGINAL TRAINING AND TEST DATASETS

(CONTINUED)

4. *wine* dataset

Training		Predicted	
		High	Low
Actual	High	124	50
	Low	17	1088
	overall accuracy=1212/1279=0.9476		
	accuracy of High=124/174=0.7126		
	accuracy of Low=1088/1105=0.9846		
	g-mean=0.8377		

Test		Predicted	
		High	Low
Actual	High	21	22
	Low	16	261
	overall accuracy=282/320=0.8813		
	accuracy of High=21/43=0.4884		
	accuracy of Low=262/277=0.9422		
	g-mean=0.6784		

5. *yeast* dataset

Training		Predicted	
		MIT	REST
Actual	MIT	36	107
	REST	16	440
	overall accuracy=1063/1187=0.8955		
	accuracy of MIT=96/195=0.4923		
	accuracy of REST=967/992=0.9748		
	g-mean=0.6927		

Test		Predicted	
		MIT	REST
Actual	MIT	21	28
	REST	6	242
	overall accuracy=263/297=0.8855		
	accuracy of MIT=21/49=0.4286		
	accuracy of REST=242/248=0.9758		
	g-mean=0.6467		

APPENDIX D. MEAN DIFFERENCE BEWTEEN *g-mean* OF *the training set* IN TERMS OF THREE APPROACHES (SVM-SMOTE, SVM-RU and GA-IS)

1. abalone dataset

ANOVA table (α=0.05)

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
methods	2	0.0078	0.0039	69.8840	< .0001*
Error	57	0.0032	0.0001		
C. Total	59	0.0110			

Comparison of mean difference of the g-mean (Tukey's HSD Post-hoc test)

Level	-Level	Difference	Std. Err. Diff.	Lower CL	Upper CL	p-value
SVM-SMOTE	SVM-RU	0.0261	0.0024	0.0204	0.0318	< .0001*
SVM-SMOTE	GA-IS	0.0218	0.0024	0.0161	0.0275	< .0001*
GA-IS	SVM-RU	0.0044	0.0024	-0.0013	0.0101	0.1609

Level	Groups		Mean
SVM-SMOTE	A		0.8390
GA-IS		B	0.8173
SVM-RU		B	0.8129

2. blood transfusion dataset

ANOVA table (α=0.05)

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
methods	2	0.0030	0.0015	25.2081	< .0001*
Error	57	0.0034	0.0001		
C. Total	59	0.0064			

Comparison of mean difference of the g-mean (Tukey's HSD Post-hoc test)

Level	-Level	Difference	Std. Err. Diff.	Lower CL	Upper CL	p-value
SVM-SMOTE	SVM-RU	0.0173	0.0024	0.0114	0.0232	< .0001*
SVM-SMOTE	GA-IS	0.0099	0.0024	0.0040	0.0158	0.0005*
GA-IS	SVM-RU	0.0074	0.0024	0.0015	0.0133	0.01*

Level	Groups			Mean
GA-IS	A			0.7072
SVM-SMOTE		B		0.6973
SVM-RU			C	0.6899

APPENDIX D. MEAN DIFFERENCE BEWTEEN *g-mean* OF *the training set* IN TERMS OF THREE APPROACHES (SVM-SMOTE, SVM-RU and GA-IS) (CONTINUED)

3. Pima dataset

ANOVA table (α=0.05)

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
methods	2	0.0085	0.0043	234.0065	< .0001*
Error	57	0.0010	0.0000		
C. Total	59	0.0096			

Comparison of mean difference of the g-mean (Tukey's HSD Post-hoc test)

Level	-Level	Difference	Std. Err. Diff.	Lower CL	Upper CL	p-value
SVM-SMOTE	SVM-RU	0.0286	0.0014	0.0253	0.0318	< .0001*
SVM-SMOTE	GA-IS	0.0197	0.0014	0.0164	0.0229	< .0001*
GA-IS	SVM-RU	0.0089	0.0014	0.0056	0.0121	< .0001*

Level	Groups			Mean
GA-IS	A			0.7914
SVM-SMOTE		B		0.7717
SVM-RU			C	0.7628

4. wine dataset

ANOVA table (α=0.05)

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
methods	2	0.0264	0.0132	514.0868	< .0001*
Error	57	0.0015	0.0000		
C. Total	59	0.0279			

Comparison of mean difference of the g-mean (Tukey's HSD Post-hoc test)

Level	-Level	Difference	Std. Err. Diff.	Lower CL	Upper CL	p-value
SVM-SMOTE	SVM-RU	0.0458	0.0016	0.0419	0.0497	< .0001*
SVM-SMOTE	GA-IS	0.0431	0.0016	0.0392	0.0469	<.0001*
GA-IS	SVM-RU	0.0027	0.0016	-0.0011	0.0066	0.2093

Level	Groups			Mean
GA-IS	A			0.9257
SVM-SMOTE	A			0.9285
SVM-RU			B	0.8827

APPENDIX D. MEAN DIFFERENCE BEWTEEN *g-mean* OF *the training set* IN TERMS OF THREE APPROACHES (SVM-SMOTE, SVM-RU and GA-IS) (CONTINUED)

5. yeast dataset

ANOVA table (α=0.05)

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
methods	2	0.0044	0.0022	99.6993	< .0001*
Error	57	0.0013	0.0000		
C. Total	59	0.0057			

Comparison of mean difference of the g-mean (Tukey's HSD Post-hoc test)

Level	-Level	Difference	Std. Err. Diff.	Lower CL	Upper CL	p-value
SVM-SMOTE	SVM-RU	0.0196	0.0015	0.016	0.0232	< .0001*
SVM-SMOTE	GA-IS	0.0162	0.0015	0.0126	0.0198	< .0001*
GA-IS	SVM-RU	0.0034	0.0015	-0.0002	0.007	0.0648

Level	Groups	Mean
SVM-SMOTE	A	0.8251
GA-IS	B	0.8089
SVM-RU	B	0.8055

APPENDIX E. MEAN DIFFERENCE BEWTEEN *g-mean* OF *the training set* CORRESPONDING TO ITERATIONS CHOSEN FOR INSTANCE SELECTION

ONE-WAY ANOVA RESULTS (α=0.05)

1. abalone dataset

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
Iterations	2	0.0026	0.0013	176.9726	< 0.0001*
Error	57	0.0004	0.0000		
C. Total	59	0.0031			

2. blood transfusion dataset

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
Iterations	2	0.0156	0.0078	70.6904	< 0.0001*
Error	57	0.0063	0.0001		
C. Total	59	0.0220			

3. Pima dataset

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
Iterations	2	0.0673	0.0337	5295.5310	< 0.0001*
Error	57	0.0004	0.0000		
C. Total	59	0.0677			

4. wine dataset

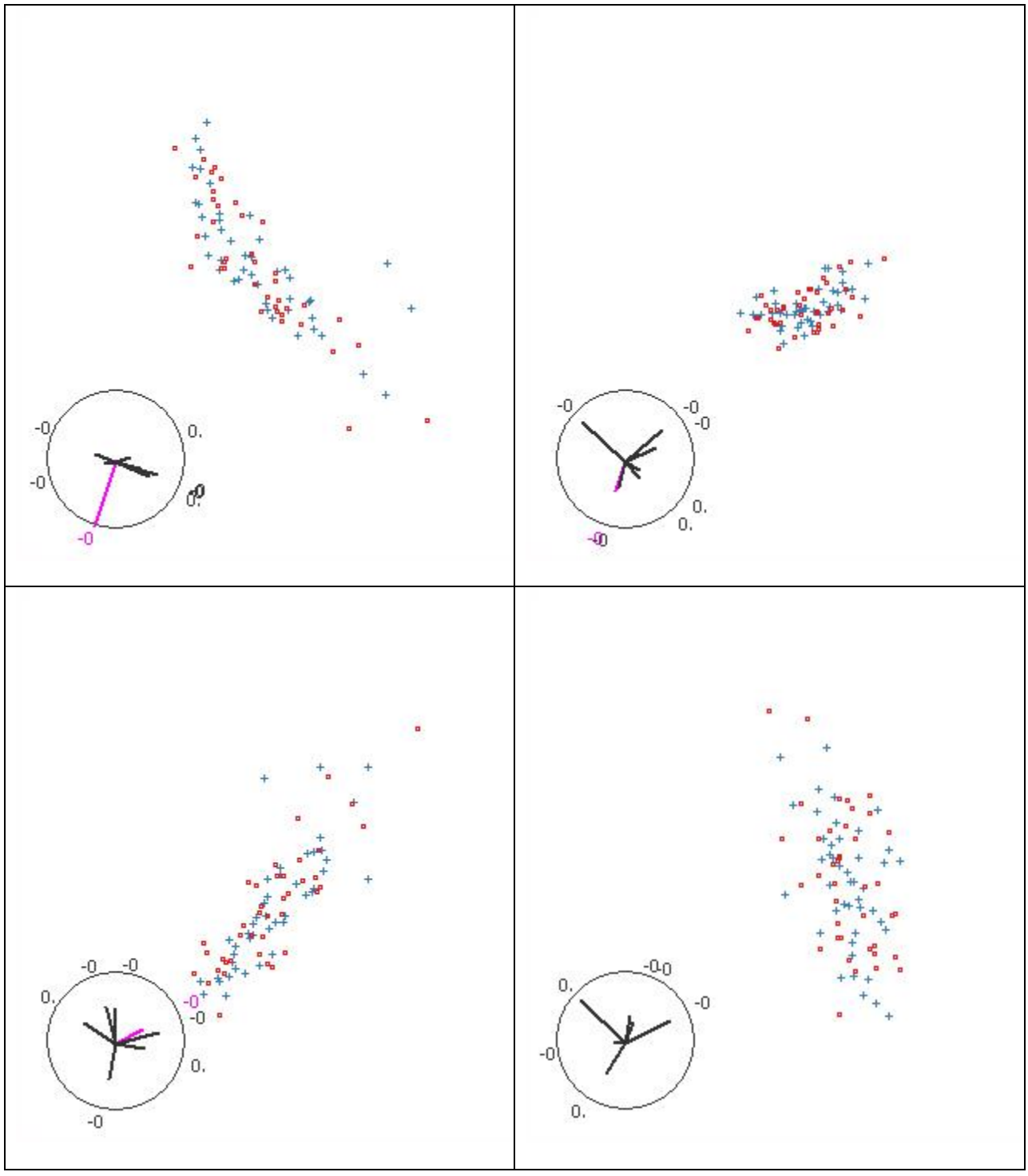
Source	DF	Sum of Squares	Mean Square	F ratio	p-value
Iterations	2	0.0470	0.0235	8692.3748	< 0.0001*
Error	57	0.0002	0.0000		
C. Total	59	0.0472			

5. yeast dataset

Source	DF	Sum of Squares	Mean Square	F ratio	p-value
Iterations	2	0.0022	0.0011	250.8151	< 0.0001*
Error	57	0.0003	0.0000		
C. Total	59	0.0025			

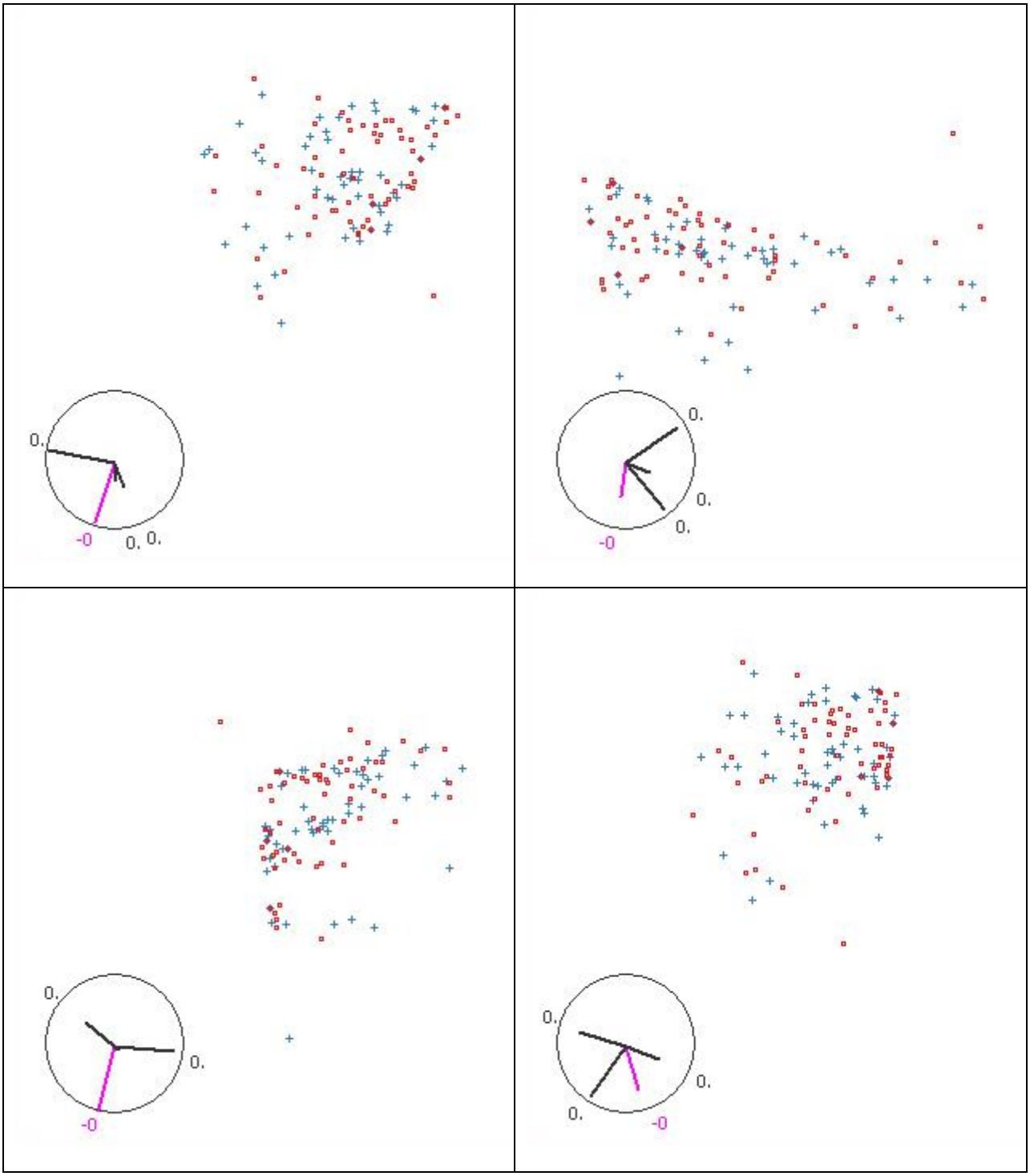
APPENDIX F. SELECTED INSTANCES FOR LEARNING FROM GENETIC ALGORITHM BASED INSTACE SELECTION APPROACH

1. abalone dataset



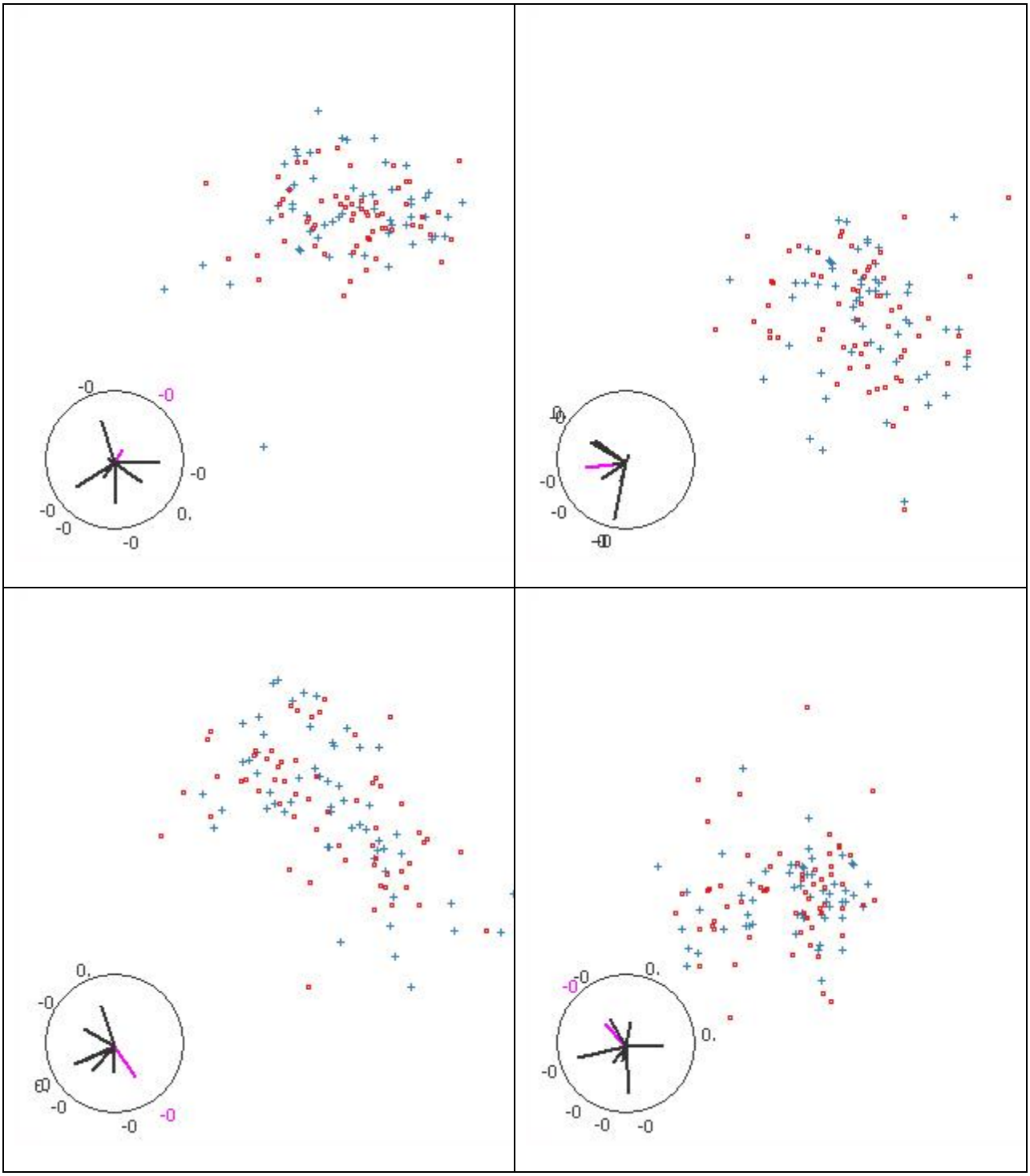
APPENDIX F. SELECTED INSTANCES FOR LEARNING FROM GENETIC ALGORITHM BASED INSTACE SELECTION APPROACH (CONTINUED)

2. blood transfusion dataset



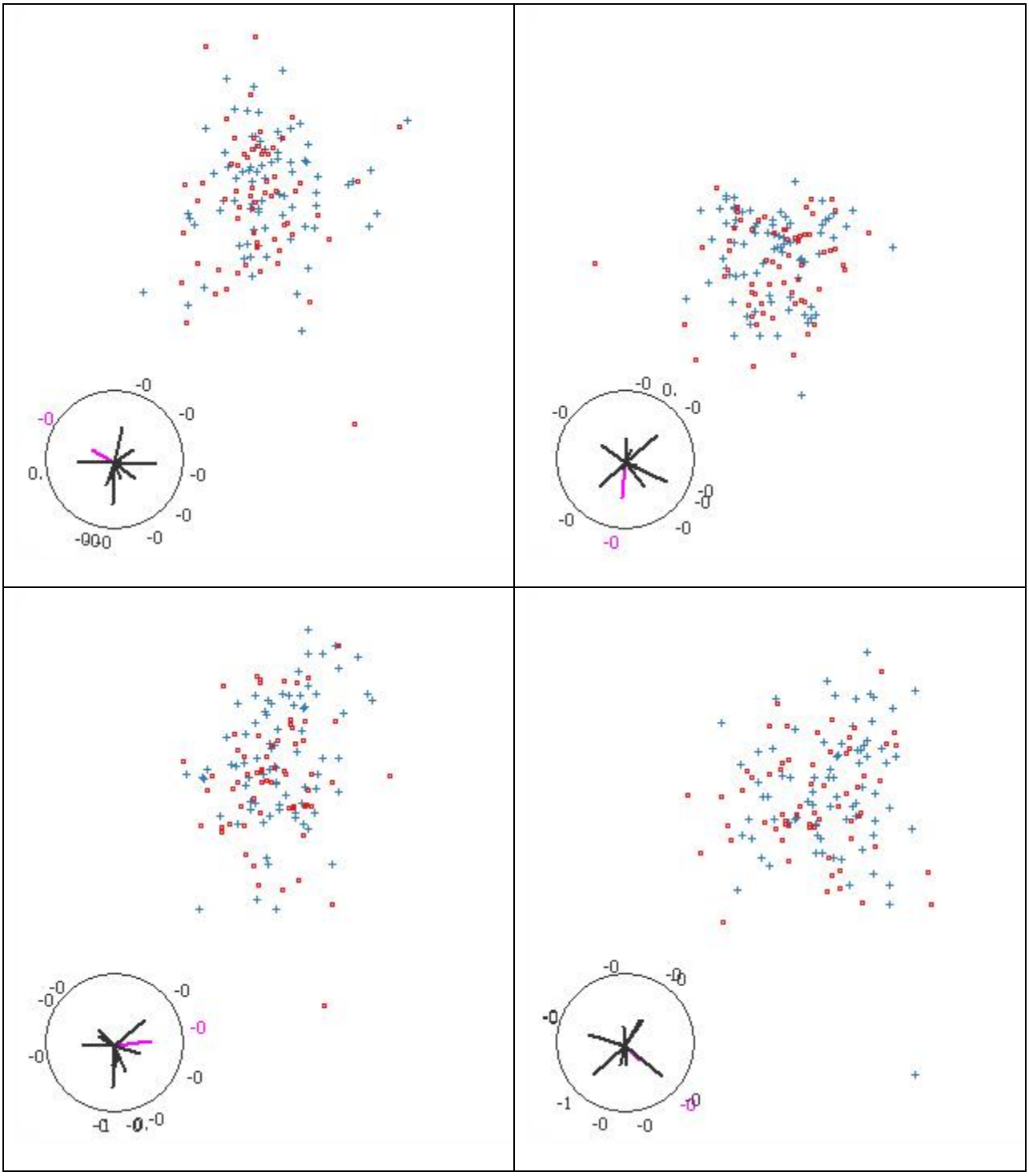
APPENDIX F. SELECTED INSTANCES FOR LEARNING FROM GENETIC ALGORITHM BASED INSTACE SELECTION APPROACH (CONTINUED)

3. Pima dataset



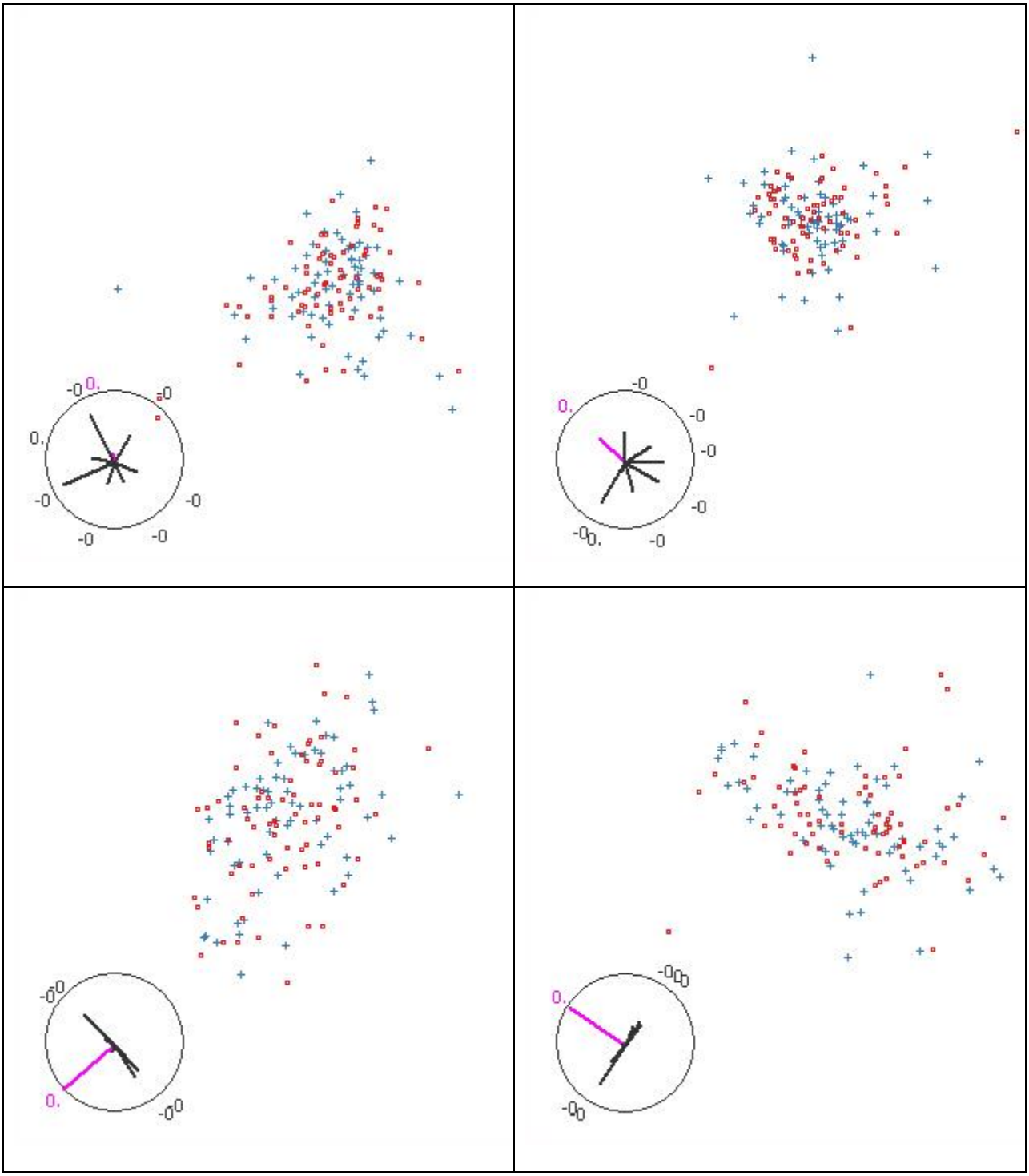
APPENDIX F. SELECTED INSTANCES FOR LEARNING FROM GENETIC ALGORITHM BASED INSTACE SELECTION APPROACH (CONTINUED)

4. wine dataset



APPENDIX F. SELECTED INSTANCES FOR LEARNING FROM GENETIC ALGORITHM BASED INSTACE SELECTION APPROACH (CONTINUED)

5. yeast dataset



BIBLIOGRAPHY

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying Support Vector Machines to Imbalanced Datasets. *Proceedings of ECML'04*, pp. 39-50.

Almeida, M., Braga, A., & Braga, J.(2000). SVM-KM: speeding Sams learning with a priori cluster selection and *k*-means. In Proceedings of the 6th Brazilian Symposium on Neural Networks, 162-167.

Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12, 783-789.

Barandela, R., Sanchez, J.S., Garcia, V., & Rangel, E.(2003). Strategies for learning in class imbalance problem. *Pattern Recognition 36,* 849-851.

Bradley (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), pp. 1145-1159.

Breiman, L., Friedman, J., Olshen, R., & Stone, C.(1984). Classification and regression trees, Wadsworth & Brooks Publisher, Monterey, CA.

Burges, C.J.(1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery2*, pp.121-167.

Canu S., Grandvalet Y., & Rakotomamonjy A. (2005). SVM and Kernel methods matlab toolbox. *Preception Systems et Information*, INSA de Rouen, Rouen, France.

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

Chawla, N., Lazarevic, A., Hall, L., & Bowyer, K.(2003). SMOTEBoost: Improving prediction of the minority class in boosting. *In Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases*

Chan, P. & Stolfo, S.(2001). Toward scalable learning with non-uniform class and cost distribution: a case study in credit card fraud detection. *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining.*

Cortes, C.(1995). Prediction of Generalisation Ability in Learning Machines. PhD thesis, Department of Computer Science, University of Rochester.

Cristianini, N., Kandola, J., Elisseeff, A, & Shawe-Taylor, J.(2002). On kernel target alignment. *Advances in Neural Information Processing Systems 14.* Cambridge, MA: MIT Press.

Domingos, P.(1999). MetaCost: A general method for making classifiers cost-sensitive. *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining,* ACM Press.

Drummond, C. & Holte, R. (2003). C4.5, Class Imbalance and Cost Sensitivity: Why Under-sampling beats Over-sampling. *ICML 2003 Workshop on Learning from Imbalanced Data Sets II*, Washington, DC.

Estabrooks, A., Jo, T. & Japkowicz, N. (2004). A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence*. Vol. 20, pp 18-36.

Fan, W. Stolfo, S., Zhang, J., & Chan, P.(1999). AdaCost: misclassification cost-sensitive boosting. *In Proceedings of the Sixteenth International Conference on Maching Learning.*

Freund, Y. & Schapire, E.(1997). A decision-theoretic generation of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55(1*) pp. 119-139.

Han, H., Wang, W., & Mao, B.(2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *In Proceedings of the International Conference on Intelligent Computing 2005, Part I*, LNCS.

Han, D., Han, C., Yang, Y., Liu, Y. & Mao, W. (2008). Pre-extracting Method for SVM Classification Based on the Non-parametric k-NN Rule. *In the Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)*, Tampa, Florida, USA

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory, IT-14*, pp. 515-516.

He, H., Bai, Y., Garcia, E.A., & Li, S. (2008). ADASYN: Adaptive Synthetic sampling approach for imbalanced learning. *Proceedings of International Joint Conference on Neural Networks*, pp. 1322-1328.

Holland, J.(1973). Genetic Algorithms and the Optimal Allocation of Trials. *In SIAM Journal on Computing, 2(2),* pp. 88-105.

Holte, R. & Drummond, C. (2006). Cost Curves: An Improved Method for Visualizing Classifier Performance. *Machine Learning, vol. 65, no 1*, pp. 95-130.

Huang, S. & Lee, Y.(2004). Reduced support vector machines: a statistical theory. Technical report, Institute of Statistical Science, Academia Sinica, Taiwan.

Japkowicz, N.(2000). The Class Imbalance Problem: Significance and Strategies. *In Proceedings of the 2000 International Conference on Artificial Intelligence: Special Track on Inductive Learning*, Las Vegas, Nevada.

Japkowicz, N. (2000). Learning from Imbalanced Data Sets: a Comparison of Various Strategies. *AAAI Workshop on Learning from Imbalanced Data Sets, AAAI Press,* Menlo Park, CA,

Japkowicz, N.(2001). Supervised versus unsupervised binary-learning by feedforward neural networks, *Machine Learning* **42** (1–2), 97–122

Joachims, T.(1998). Text Categorization with SVM: Learning with Many Relevant Features. *Proceedings of ECML-98, 10th European Conference on Machine Learning.*

Joshi, M., Kumur, V., & Agarwal, R.(2001). Evaluating boosting algorithms to classify rare cases: comparison and improvements. *In Proceedings of the First IEEE International Conference on Data Mining.*

Karakoulas, G., & Taylor, J. (1999). Optimizing classifiers for imbalanced training sets. *In Advances in Neural Information Processing Systems.*

Kotsiantis, S. & Pintelas, P. (2003). Mixture of Expert Agents for Handling Imbalanced Data Sets. *Annals of Mathematics, Computing & TeleInformatics,* Vol 1.No 1 pp. 46-55.

Kubat, M. & Martin, S.(1997). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. *Proceedings of the 14th International Conference on Machine Learning.*

Kubat, M., Holte, R., & Matwin, S.(1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning, 30,* 195-215

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. *Proceedings of AI in Medicine in Europe: Artificial Intelligence Medicine,* pp. 63-66.

Lee, Y. & Mangasarian, O.(2001). RSVM: Reduced Support Vector Machines. In *Proceedings of the 1st SIAM International Conference on Data Mining.*

Lewis, D & Catlett, J.(1994). Heterogeneous uncertainty sampling for supervised learning. In Cohen, W. W. & Hirsh, H. (Eds.), *Proceedings of ICML-94, 11th International Conference on Machine Learning,* (pp. 148-156)., San Francisco. Morgan Kaufmann.

Liu, X., Wu, J., & Zhou, Z. (2006). Exploratory under-sampling for class imbalance learning. *Proceedings of the 6th IEEE International Conference on Data Mining*. pp. 965-969.

Liu, X. & Zhou, Z. (2006). The influence of class imbalance on cost-sensitive learning: An empirical study. *Proceedings of International Conference on Data Mining.*pp. 970-974.

Liu, Y., Lee, Y., & Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning*, 46, 191-202.

Liu, Y., An, A., & Huang, X.(2006). Boosting prediction accuracy on imbalanced datasets with svm ensembles. *In the Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Singapore, 107-118.

McCarthy, K., Zabar, B., & Weiss, G. (2005). Does Cost-sensitive learning Beat Sampling for Classifying rare classes? *Proceedings of International Workshop Utility-based Data Mining*, pp. 69-77.

Manevitz, L. M. & Yousef, M.(2002). One-class SVMs for Document Classification. *Journal of Machine Learning Research, (2)*, 139-154.

Monard, M. & Batista, G. (2002). Learning with Skewed Class Distribution. *Advance in Logic, Artificial Intelligence and Robotics*, Sao Paulo, SP, IOS Press pp. 173-180.

Murphy, P. & Aha, D.(1994). UCI repository of machine learning databases. University of California-Irvine, Department of Information and Computer Science. http://www1.ics.uci.edu/mlearn/MLRepository.html.

Osuna, E., Freund, R., & Girosi, R.(1996). Support vector machines: training and applications. A.I. Memo AIM – 1602, MIT A. I. Lab.

Platt, J. (1999). Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In Advances in Kernel Methods - Support Vector Learning, Schoelkopf, B., Burges, C., and Smola, A. Eds.,185-208.

Scholkopf, B & Smola, A. (2002). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. *MIT Press*, Cambridge, MA.

Shin, H. & Cho. S.(2003). Fast pattern selection for support vector classifiers. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Lecture Notes in Artificial Intelligence (LNAI 2637), 376-387.

Quinlan, J.(1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

Raskutti, B. & Kowalczyk, A. (2004). Extreme Re-balancing for SVMs: a case study. *SIGKDD Explorations*, 6(1), pp. 60-69.

Tomek, I. (1976). Two Modifications of CNN. *IEEE Transactions on Systems Man and Communications SMC-6*, pp. 760-772.

Tong, S. Chang, E. (2001). Support Vector Machine Active Learning for Image Retrieval. *Proceedings fo ACM International Conference on Multimedia*, 107-118.

Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer-Verlag, NY

Veropoulos, K., Cristianini, N., Campbell, C. (1999). Controlling the Sensitivity of Support Vector Machines. *In Proceeding of IJCAI'99*.

Wang, Benjamin. & Japkowicz, N. (2008). Boosting Support Vector Machines for Imbalanced Data Sets. *ISMIS 2008*, 38-47.

Wilson, D. (1972). Asymptotic Properties of Nearest Neighbor Rules using Edited Data. *IEEE Transactions on Systems, Man and Cybernetics, Vol. 2*, pp 408-420.

Weiss, G. (2003). The Effect of Small Disjuncts and Class Distribution on Decision Tree Learning, *Ph.D. Dissertation, Department of Computer Science, Rutgers University*, New Brunswick, New Jersey.

Weiss, G. (2004). Mining with rarity: A unifying framework. *SIGKDD Explorations 6 (1). pp 7-19.*

Wu, G. & Chang, E. (2003). Class-Boundary Alignment for Imbalanced Dataset Learning. *In ICML 2003 Workshop on Learning from Imbalanced Data Sets II,* Washington, DC.

Zhang, J. & Mani, I.(2003). KNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction. *In ICML 2003 Workshop on Learning from Imbalanced Data Sets II,* Washington, DC.

Zhang, W. & Kin, I. (2002). Locating support vectors via β-skeleton technique. *In Proceedings of the International Conference on Neural Information Processing (ICONIP)*, 1423-1427

Zheng, Z., Wu, X., Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *SIGKDD Explorations, 6*(1).