A LINEAR WRAPPER METHOD FOR DETECTION OF ATYPICAL POINTS IN CLASSIFICATION

by

Saeed Hashemi Mohammadabad

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
January 2005

Your file Votre référence
ISBN: 0-494-00952-7
Our file Notre référence
ISBN: 0-494-00952-7

DALHOUSIE UNIVERSITY

To comply with the Canadian Privacy Act the National Library of Canada has requested that the following pages be removed from this copy of the thesis:

Preliminary Pages
- Examiners Signature Page (pii)
- Dalhousie Library Copyright Agreement (piii)

Appendices
- Copyright Releases (if applicable)

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ASR	Atypical Sequential Removing
CART	Classification And Regression Trees
CBR	Case-Based Reasoning
CP curves	Coverage-Performance curves
CV	Cross-Validation
FSS	Feature Subset Selection
KNN	K-Nearest Neighbor classifier
LDA	Linear Discriminant Analysis classifier
NB	Naïve Bayes classifier
PC	Principal Component
PCA	Principal Component Analysis
QB	Quadratic Bayesian classifier
QDA	Quadratic Discriminant Analysis classifier
SSS	Sample Subset Selection
SVM	Support Vector Machines

Abstract

The detection of atypical data in a dataset, using a linear wrapper approach is the focus of this research. Atypical points are considered to be the misclassified points that the proposed algorithm (Atypical Sequential Removing: ASR) finds not useful to the classification task. They may include outliers and/or overlapping samples. The majority of the available atypical detection techniques apply a filter approach in which there is no requirement for the filter to be consistent with the classifier in use. The fastest available wrapper techniques, on the other hand, have a quadratic running time which is prohibitive in practice for sample subset selection. The approach presented in this research is a linear wrapper technique that, instead of using any predetermined criteria, uses only the classifier itself and a performance measure to identify atypical points in the data. As a result, it is expected to be more consistent with the classifier in use. Using a cross validation scheme, ASR manages to give a reliable test performance while identifying and ranking the atypical points in the whole dataset. To ensure that ASR does not remove informative misclassified points, Ada-boost was compared with S-boost (trained with the data without atypicals). The results showed that when a significant portion of misclassified points were removed from the training set, S-boost had a very close performance to Ada-boost. In the comparison between ASR and the Mahalanobis filter method, the results shows that ASR was more accurate in identifying atypical points, it was more consistent with the classifier in use by keeping its performance as high as the classifier with no removal from the training set, and it was able to remove 30% more points than the Mahalanobis filter. However, the assertions in the literature (removing some points from the training can enhance the performance of classifiers) were not confirmed for overall performance under the experimented linear wrapper. Experiments on 20 benchmark datasets and 7 classifiers show promising results and confirm that this linear wrapper method has some advantages and can be used for atypical detection.

I Introduction

"To study the abnormal is the best way of understanding the normal."

— William James (1842-1910)

One of the oldest subjects in data analysis is the study of atypical detection methods (e.g., outliers), their effect on subsequent inference, and the decision on what to do with them (Millar and Hamilton, 1999). In general, outliers are the points that lie reasonably far from the cluster of their claimed class. Comments by Bernoulli, (in 1777, as referenced by Beckman and Cook, 1983, in their outstanding paper on outliers) indicate that discarding unusual cases was common practice more than 200 years ago. There have been long debates in different disciplines such as medicine, statistics, and in machine learning on how to handle abnormal cases and it is still a subject of debate.

Unfortunately, there does not seem to be a standard terminology for the concepts related to atypical points. Authors have often taken liberty in using different terms that match better to their definitions and approaches. While the term outlier is commonly used, novelty data (Schölkopf *et al.*, 2000), ambiguous data (Trappenberg, *et al.*, 1999), irrelevant examples (Blum, 1997), atypical examples (Hashemi and Trappenberg, 2002), inconsistent examples (Gamberger *et al.*, 1996), and other terms have been used to refer to either similar or somewhat different concepts. The best way to understand what different terms mean seems to be through the context in which they have been introduced.

For instance, the terms ambiguous data (Trappenberg, *et al.*, 1999), inconsistent examples (Gamberger *et al.*, 1996), to some extent mislabeled data (Brodley and Friedl, 1999), and overlapping samples all refer to the points that have almost the same values for features but different class labels.

In this research, atypical points are defined from the point of view of the classifier. The general definition of atypical points is as follows.

> *Given an inducer I, and a dataset $\mathcal{D}$ with labeled instances, an* ***atypical subset*** *$\mathcal{D}_{Atp}$, is a subset of $\mathcal{D}$ such that the generalization performance of the induced classifier $C = I(\mathcal{D} - \mathcal{D}_{Atp})$ is maximal.*

Hence, atypical points, as defined in this research, can be two kinds: outliers and overlapping samples. Outliers lie reasonably far from the cluster of their corresponding class in the problem space while overlapping samples are usually close to the boundary of their class and members of more than one class fall in the overlapping region. Outliers, themselves can include noise and the exception points (valid points with feature values far from the rest of their class). The interest of this research is only in the detection of atypical points from the classifier point of view and not in separating different kinds of atypicality.

1.1 The importance of outliers

Generally speaking, research on outliers is important from at least three aspects. In some classification applications like credit card fraud detection, credit approval, insurance claims, network intrusion detection, irregularities in gene expressions, etc., the detection of outliers is considered to be the objective of classification (Ripley, 1996; Han and Kamber, 2001).

The second aspect is to have a reliable method in enhancing the quality of data by removing the outliers from the data. This can increase the confidence factor (CF) and perhaps enhance the performance of the classifier trained on such data. Generally, the farther the location of a point from the center of the cluster of its class, the less confidence we have for the assignment of its class label (more chance of being an outlier).

The last, but not the least important aspect is the possible contribution of outliers in better understanding the domain: they can be noise, they may suggest lack of a discriminative attribute for certain examples in the data, or the birth of a new class in the application domain. Finding examples that do not belong to one of the available classes can lead to the discovery of new classes (Ripley, 1996) or concepts from data in an inductive manner (Jude and McClelland, 1989). This is one of the exciting issues in incremental learning as it provides insights into the evolution of concepts within a particular domain.

1.2 The importance of overlapping samples

The same three aspects are applied to the overlapping samples too. Instances in the overlapping regions are the ones we are less sure about and would like to have a better method of classification for them. There are cases in the real world where more discriminative attributes can be obtained with a higher cost. In medical or automobile diagnostic systems, we can obtain such attributes (e.g. perform medical tests) only when necessary: if the query case falls in the overlapping region (Komorowski and Øhrn, 1999). In fact, Conversational Case-Based Reasoning (CBR) (Aha, *et al.*, 2000) is an example of such systems in machine learning. When presented with a query case, Conversational CBR asks for the value of an attribute (not given in the query) only if the case cannot be answered with a high confidence (an overlapping sample or an outlier case). Locally feature weighting (Wettschereck *et al.*, 1997) and instance weighting (Wilson and Martinez, 2000a), in lazy learners, are also machine learning problems that can benefit from the detection of overlapping samples. Assigning weights can be a time consuming process. In locally weighted learning, to save the time and possibly increase the performance, weights can be fine tuned only for the overlapping samples for which the learner has shown less confidence.

The second aspect (data cleaning), can be performed by removing the overlapping samples that do not contribute to the classification task positively. Indeed, since the number of overlapping samples is often greater than the number of outliers and they are

closer to the boundary of classes, they may have more effect on the decision function then the outliers. Thus, removing some of them may help smooth the decision function too.

For the third aspect (the possible contribution of atypical points in better understanding the domain), overlapping samples may provide more evidence and material than outliers for further studies in the domain area. Overlapping samples violate the basic assumption in the mainstream classification that an instance must belong to only one class. This violation is either because the classifier cannot find a proper rule to separate the classes or it is a problem embedded in the data. Only in the latter case, overlapping samples may imply the discovery of a new class or lack of a discriminative attribute, and perhaps noise, in the data. This is similar to outliers that can be either exceptions (informative) or just noise (not informative). For instance, Figure I-1 shows a simple case of overlapping; a third dimension (a missing attribute) can perhaps separate the two classes. For example, weights of men and women overlap but each instance can belong to only one class, so we need a more discriminative feature. Note that outliers are often far from each other and do not tend to make a cluster in the problem space; besides, the number of overlapping samples is often more than outliers. Therefore, overlapping samples may provide more material for further study in the problematic datasets.

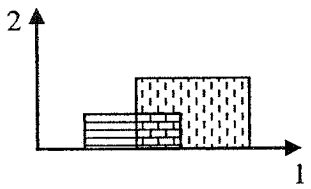

Figure I-1. Overlapping region for two rectangular classes.

1.3 Overview of the thesis

The atypical detection methods developed so far are mainly based on the preprocessing filter approach. In principle, there is no requirement for the filter to be consistent with the classification method (Blum, 1997; Brodley and Friedl, 1999). As a result, it is possible to filter out some examples from data that are not atypical from the classifier algorithm point of view. These points may be harmless to the classifier or even helpful, as they might carry information that could have increased the quality of the classifier if they had been kept in the training set (John, 1995). A solution to this problem can be the use of wrapper methods that, unlike filters, try to keep the atypical detection scheme consistent with the classifier in use (Blum, 1997). While filters use predetermined criteria like distance/similarity values or entropy measures, the wrapper approach can use only the classifier itself and a performance measure to discover atypical points in the data. In Section II, filter and wrapper methods are discussed in more detail.

From the study of methods dealing with atypical points (outlier detection, noise reduction ...) two points can be drawn. First, what to do with atypical points in terms of keeping/removing them in/from the training and test sets has been a decision (dilemma) that researchers have struggled with and have taken different approaches. This is discussed in Subsection 2.4. The second interesting point is that there are assertions in the literature that removing some points from the training set can lead to better classifiers with higher performance (discussed in Subsection 2.3). Not only our experimental results did not support this idea (under the constraints of our linear wrapper system), but also, to the best of the author's knowledge, all the reports of increase in performance due to the removal of atypical points, so far, fall in one of the following 3 categories.

1. The performance increase is insignificant or has happened only in certain datasets.
2. It has been measured in a cleaned test set (hard-to-classify examples were already removed from the test set). For instance, in (Trappenberg and Back, 2000), the reported improved performance is not really generalization accuracy (1 – *classification error*) on the intact test set. But it is the confidence in classification, expressed as accuracy, which has improved for a subset of the test data (not the whole test data). Hence, those legitimate results should not be mistaken with the performance improvement that we generally refer to in machine learning as the generalization accuracy.
3. The performance increase is due to cleaning the noise injected only to the training set (test set was kept clean) as in (Wilson and Martinez, 2000b) and Brodley and Friedl

(1999). This is a popular case that the author thinks needs more consideration and will be explained in Subsection 2.3 too.

In the latter case, it is true that if the test set is also corrupted, measures of performance cannot be interpreted easily (or may not be reliable anymore). But we have to be clear about the difference between the efficiency of a filter and the concept of increase in the generalization performance as a result of atypical removal. It seems that this difference has not been taken seriously in the literature and deserves more attention. Although these two may be expressed both as accuracy, they are two separate measures.

When we inject noise into the training set and show that the performance of the classifier trained on the filtered training set is higher than the classifier trained on the corrupted training set, we may correctly conclude that filter efficiency (expressed as a change in accuracy) was observed. But this has nothing to do with the generalization accuracy of the classifier. In practice, it is hard to imagine if we ever need to inject noise to the data and then look for a way to remove it; but we do like to reduce the noise (or more generally, atypical points) *already embedded* in the data without injecting extra noise into it. If we do filter the original training data, and achieve a better test performance compared to the training on the original training data, then we can rightfully declare that a higher generalization accuracy is obtained. The only case when these two measures are truly equal is when no noise (0% noise level) is added to the training set and a filter is applied to the original data. Please note that since, in machine learning, often the "generalization performance' is referred to as "performance" or "accuracy," it is

misleading to use these words to show the efficiency of a filter without clarifying the subject.

In most reports of filtering in the literature, the case of filtering the original data (0% noise level) is either neglected or paid less attention while it seems to be the most relevant case to show if filtering has any effect on the generalization performance of the classifier. To the best of the author's knowledge, when no noise (0% noise level) was added, on a reasonable variety of datasets and classifiers, there has been no report that the classifiers trained on the filtered training data were significantly better than the classifiers trained on the unfiltered data (in terms of their generalization error). Thus, whether or not cleaning the original training set can make classifiers with higher overall performance (on an intact test set) than the baseline classifier (trained on unclean and uncorrupted data) is still an open question. By overall performance, we mean the performance on a reasonable number of datasets (noisy and regular) and variety of classifiers.

In Section III, coverage-performance curves (as the initial work in this research) to measure the effect of atypical removal and the concept of sample subset selection (SSS) are introduced. The issues associated with these curves and the problems with making a wrapper approach to SSS applicable clarifies what features are preferable in a more desired algorithm.

The main obstacle in using a wrapper method in sample subset selection for atypical detection is that of time complexity. This is a serious problem even in the feature subset

selection (much smaller search space): "For all but the smallest problems, the space of possible feature subsets is too large for brute-force enumeration of all possibilities, and we must resort to heuristic search" (Kohavi and John, 1997). In Section III, there is a detailed discussion on this issue. Note that, in sample subset selection, datasets usually consist of a few hundred to several thousand instances. The size of the search space in an exhaustive search is exponential with $(2^N - 1)$ states where N is the number of instances in a dataset. Even the greedy methods like sequential algorithms (backward elimination or forward selection) with a quadratic running time are still prohibitive in practice. Because each state represents a subset of the training data, a training session is repeated over each subset of the data. The total running time of the search, hence, will be the running time of the subset selection algorithm times the training time complexity of the classifier in use. Even the fastest available wrapper solutions (with quadratic running time) are not practical for most datasets for atypical detection. That is why the approach presented in this research as a linear wrapper method in sample subset selection can be quite beneficial for atypical detection.

Overall, based on the initial efforts made on atypical detection in this research, we would like the proposed algorithm to have the following features.

- To be able to identify and rank the atypical points from the classifier point of view.
- To offer a reliable measure for the generalization power of the classifier after removing atypical points. Identifying the atypicals for the whole dataset is preferable.

- A running time of lower than quadratic in the required number of training sessions (linear preferably).
- To have a cut off point for the removal process (non-monotonic performance function).
- To avoid the classification of imbalanced data with small disjuncts that happens usually in atypical detection.
- To be general enough that the classifiers without a reject-option can also be used.

In Section IV, the definition of atypical points as well as the proposed solution to identify them in a dataset is presented. The methodology and the developed algorithm (Atypical Sequential Removing: ASR) are discussed in detail. The time complexity of the algorithm is also presented. Besides being a linear wrapper technique, the proposed method can identify and rank the atypical points in the whole dataset without sacrificing the validity of the generalization error. This is done by employing a cross validation scheme in response to the dilemma of handling atypical examples within training and/or test sets. The proposed wrapper algorithm is also general enough to be used even with classifiers with no reject option (like the typical support vector machine: SVM).

In Section V, the experimental setup and the experimented datasets are explained. A total of 20 datasets and 7 classifiers were used in this research. The selected performance measure is the prediction accuracy as it is the most commonly used one. However, a single number (accuracy) cannot show if among the points removed by ASR, there are not some points that are not really atypical or if it has left some atypicals in the training

set. To study this effect, a comparison of two ensemble techniques (Ada-boost and S-boost) was performed. Ada-boost is a standard boosting technique that tries to make good use of all misclassified points. S-boost is the same technique but atypical points have been removed from its training set. For instance, if ASR removes some misclassified points that could have been useful to the boosting process, then Ada-boost should perform better than S-boost. The result shows that while a significant portion of misclassified points were removed from the training set, S-boost had a very close performance to Ada-boost indicating that ASR has generally been successful in removing atypical points. Another comparison was designed, in Section V, between ASR and a widely used filter method, the Mahalanobis filter.

In Section VI, the results of these experiments are presented in the form of 3 groups of tables: ASR and baseline; Ada-boost and S-boost; and ASR and the filter method. The discussion of all the obtained results is given in Section VII. For instance, the comparison of ASR and the filter method shows that ASR was more accurate in identifying atypical points, it was more consistent with the classifier in use by keeping its performance as high as that of the baseline (the classifier with no removal from the training set), and it was able to remove 30% more points from the dataset than the Mahalanobis filter. Removing more points, everything else equal, is a positive point for an atypical detection method. The fact that, unlike the filter method, ASR does not degrade because of the removal process is another positive point. The concluding remarks as well as the ideas for future work are presented in Section VIII.

II Atypical detection and instance reduction in the literature

First of all, the goal in discussing the work of other researchers and pointing out the potential for improvement is not, by any means, to discredit their merits. Their work and effort are appreciated; the author sees them all as the steps taken for better understanding the subject.

Atypical data is often a source of concern in any classification process. Atypicality can manifest itself in two different ways: outliers and overlapping samples. Outliers, as an important kind of atypical data, have attracted researchers' attention for a long time. Although, at least to some researchers, recognizing a data point as an outlier is still a subjective issue, there seems to be an agreement on the definition of outliers. Ripley (1996) defines outliers as "examples which did not (or thought not to have) come from the assumed population of examples." Barnett and Lewis (1994) have almost the same definition for outliers: "an observation (or subset of observations) which appear to be inconsistent with the remainder of the set of data." Most definitions of outliers specify that such examples raise the suspicion that they are from a different distribution than the rest of the dataset; in other words, they lie reasonably far from the cluster of their claimed class in the problem space. Outliers themselves can be simply noise or the exceptions that are the true members of their class; for instance, an exceptionally tall camel is still a

camel even though most of its features may be more similar to the giraffe class. Separating exceptions from noise is a key issue (Brodley and Friedl, 1999) and because it can involve domain knowledge some authors leave this decision to the human expert (Guyon, *et al.*, 1996).

Overlapping samples has been discussed in the literature under different names. As mentioned earlier, the terms ambiguous data (Trappenberg, *et al.*, 1999), inconsistent examples (Gamberger *et al.*, 1996), to some extent mislabeled data (Brodley and Friedl, 1999), and overlapping samples (Hashemi, 2002) all refer to the points with almost the same attribute values but different class labels. Overlapping samples present a type of atypicality different from outliers; they do not show any inconsistency with the other members of their own class (Hashemi and Trappenberg, 2002). Hence, outlier detection techniques cannot distinguish them.

In this Section, two main approaches (filter and wrapper methods) to the problem of detecting atypical points are briefly described. Each approach has its own pros and cons. Please note that the distinction between *wrapper* and *filter* methods in feature subset selection (John, *et al.*, 1994), is explained as *open loop* and *closed loop* in statistics, and as *performance* and *preset biases* in feature weighting (Wettschereck *et al.*, 1997). The same distinction can be made in the detection of atypical points. Hence, in this research we choose the terms *wrapper* and *filter* as they are more commonly used.

2.1 The filter method

Numerous methods can be classified as filter methods for outlier or atypical detection. The main characteristic of these methods is that they act separately from the learner (classification or regression module). The filter process is depicted in Figure II-1.

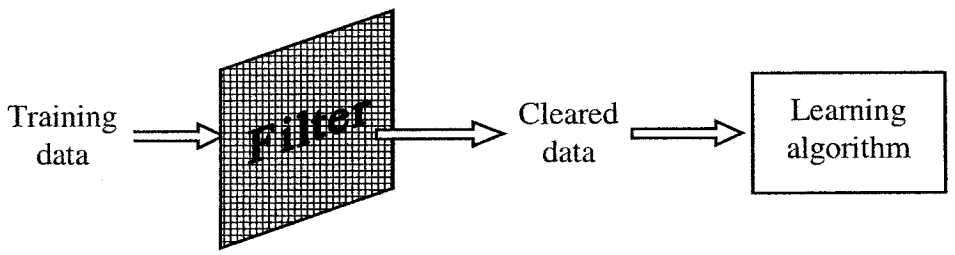

Figure II-1. General procedure to filter the data.

Usually, filters are fast methods that use some predetermined criteria to identify potential outliers. Hence, they can be used as pre-processing techniques in classification or regression. The criteria that define potential outliers can be statistical, distance/similarity-based, or entropy-based.

Statistical techniques assume a probability model for the dataset and try to find potential outliers using some kind of discordancy test. In order to decide if a point is an outlier or not, statistical tests (like Chi-squared) with certain predetermined criteria are usually performed. Any change in the underlying assumptions (e.g., Chi-squared distribution)

and in the value of a parameter (e.g., α, significance level) in these filters may change the outcome of the outlier identification process.

Distance or similarity-based techniques are the most versatile in both classification and regression. Cluster analysis techniques are often used to identify and give ranks to the points that are far from the population. Mahalanobis distance (Ripley, 1996) is a well-known filter that will be explained in detail. Wilson and Martinez (2000b) provided a survey on filter methods of instance reduction techniques for KNN-type algorithms. For instance, edited nearest neighbor rule removes each instance from the training set if it does not agree with the majority of its k nearest neighbors ($k = 3$ usually). Use of unsupervised learning can be seen, for instance, in 1-class SVM for novelty detection (Schölkopf *et al.*, 2000). The results often vary based on what similarity function and what norm has been used (e.g., L_1, L_2, or L_∞).

Entropy-based techniques are also used for outlier detection. One example is robust decision trees proposed by John (1995) in which a pruned tree is used to classify the training data. The misclassified points are removed from the training set and training is repeated till all points in the shrunk training set can be classified correctly by the last pruned tree.

Williams, et al., (2002) took a kind of semi-supervised learning approach (where the training data has already been divided into outliers and non-outliers classes). They applied a replicator neural network (RNN) for outlier detection. RNN uses the features in

the dataset as both input and output nodes of the neural network. After training, RNN is indeed a compressed representation of the dataset. This approach employs multi-layer perceptron neural networks with three hidden layers and the same number of output neurons and input neurons to model the data. A measure of outlier factor of individuals is then developed as the average of reconstruction error for all features of individual data points. For scalability the RNN is trained with a smaller training set and then applied to all of the data to identify outliers. They make a comparison of RNN with two parametric (statistical) methods and a mixture-model clustering. The statistical methods are the Donoho-Stahel estimator and Hadi94. The Donoho-Stahel method is a robust multivariate estimator of location and scatter and Hadi94 is a bulk parametric method that applies Mahalanobis distance. Their 5 statistical test datasets contain only 20 to 85 examples. They also used data mining datasets with much more number of examples. Surprisingly, they considered 239 examples of Wisconsin Breast Cancer dataset (members of Malignant class) as outliers –to accommodate their binary classification without facing imbalanced data problem! This seems to be an awkward approach since the dataset is well-known of being one of the best-behaved datasets and has 683 examples and 2 classes: Benign and Malignant. The members of Malignant class are quite legitimate data points. Their report does not show any comparison of RNN and a binary classifier for the prediction accuracy. Their results show that despite claims to the contrary in the data mining literature, some existing statistical outlier detection methods scale well for large datasets.

Brodley and Friedl (1999) developed an ensemble filtering technique to improve classification accuracy by improving the quality of the training data. Like Wilson and Martinez (2000b), they added class noise (change of class labels of some examples) to the training data only while keeping the test set clean. Their focus was on identifying such mislabeled training points and removing them from the training process. Trappenberg and Back (2000) have used the idea of adding a new class IDK (I Do not Know) to the number of classes, c, and to classify atypical points into the IDK class. In their work, a KNN is used to learn IDK points from the training data and another classifier (e.g., a neural net) is used to classify the test data into $c + 1$ classes. If the majority of K neighboring points (in KNN) do not agree on the class of a point it will be classified as IDK. Their scheme, as a filter method, is explained here.

On training data:

1. IDK detection by a KNN → c+1 classes (IDK class for some of the training data)
2. Train classifier2 (e.g., a neural net) with $c + 1$ classes.

On test data:

1. Use the trained classifier2 (c+1 classes) on the test data.

Despite its advantageous simplicity, this scheme faces two major issues. The first issue is that once some of the points in the test set are assigned to the IDK class, we have no direct way of measuring the test performance of classifier2. In other words, we cannot evaluate the predictive ability of the scheme although it may increase the confidence in

the classification of non-IDK points. The second point is that there may not be enough data points for the IDK class. Hence, to train classifier2 we have to deal with imbalanced data problem.

Unlike the above work, in some systems that use two classifiers, the user cannot change one or both classifiers. Hashemi (2002) introduced a scheme to generate coverage-performance curves using a probabilistic neural network (as a fixed filter) for atypical detection with another classifier (SVM).

All these methods (filters) share one characteristic: the atypical detection module is either separate from the classification method or it was designed to work with a certain classifier only. For instance, using a filter designed based on mutual information may work well with a decision trees but removing the filtered points from the training set may be downgrade other classifiers. If we are almost sure that the atypical detection module is a suitable one for the given dataset, like when we know that data follows certain distribution reasonably well, the use of such a technique can be helpful. In reality, however, this is hardly the case; and we often try different classifiers on the same data to choose one. That is why the consistency between the classifier and atypical detection technique becomes important. Use of an entropy-based technique may filter out some points that could have been useful for a neural net or a distance-based technique may remove the points that might have been informative for a decision tree. Users may not know (or may not be able to change) the norm used in their classifier while different norms give different results. Hence, it is worth studying the techniques that detect the

atypical points according to the inductive bias of the classifier in use so that they can be used with almost any classifier.

2.2 The wrapper method

To identify atypicals, unlike filters, a wrapper method does not use any predetermined criterion that may change its outcome. Instead, it uses only the classifier itself and a performance measure to determine atypical points in the data. Hence, it tries to keep the atypical detection scheme consistent with the classifier in use. Figure II-2 shows the interaction between the search algorithm and the learning algorithm (a classification scheme or a regression model) in the wrapper approach.

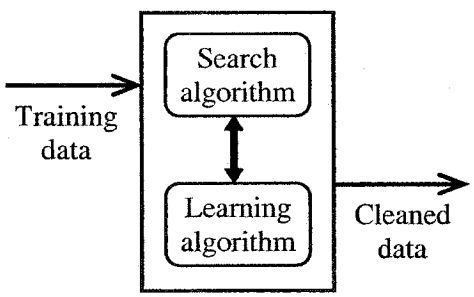

Figure II-2. The procedure in the wrapper approach.

The search (subset generation) algorithm provides the learner with the candidates that are to be evaluated. A candidate (a *state*) is a combination of the examples in which certain points are removed (potential atypicals). The objective is usually to maximize the

performance of the learner. Hence, for a given performance criterion, wrapper methods differ mainly in their strategies in choosing candidate sets.

The wrapper approach is generally far more expensive computationally and that is why it has not been explored in atypical detection yet. With the advance of high-speed computing the process that would take a week 10 years ago, may take only seconds today. This allows us now to study wrapper approaches to atypical detection and investigate the possible advantages of such techniques.

As Kohavi and John (1997) declared, the closest formulation to the wrapper method was the *search of the bias space* approach which dates back to Provost (1992). The work of Skalak (1994) is considered as the first case in using a wrapper approach in instance selection (Kohavi and John, 1997) as opposed to feature selection. To detect prototypes for the KNN algorithm (when $k = 1$), he showed that very few prototypes may suffice sometimes. According to Skalak (1994), a small number of prototypes can achieve comparable or superior predictive accuracy based on two speculations: (1) noise reduction; and (2) overfitting prevention as only a small number of examples are considered as prototypes.

His method starts with *a priori* specification of the number of prototypes (predetermined by the user and usually chosen as a small number in the order of the number of classes) and uses a simplified genetic algorithm approach to search for the best prototypes. When the clusters of classes become a little more sophisticated (such as having disjuncts of a

class or the case of a cluster inside another cluster) his method faces difficulty because the method is suitable only for datasets with widely spaced classes that exhibit a high degree of internal coherence and external isolation (Skalak, 1994). Both issues (predetermined number of points and the assumption of simple and well-defined feature space) prevent his method from being used in atypical detection where we do not now the number of atypical points in advance and datasets are supposed to be problematic.

In a survey of machine learning methods to find irrelevant examples, Blum (1997) categorized Boosting and Windowing as wrapper techniques for instance selection. In windowing, a classifier is trained on a random sample of the training data. Then the classifier tests the rest of the training data and a random subset of its misclassified points is added to the original sample and training is renewed. This process is repeated till all training data can be classified correctly by the classifier. Blum's categorization is more from a sampling view point and certainly not from atypical detection. For instance, for Boosting the paper asserts "by paying more attention to more informative examples (misclassified points in last trials) the classifier can increase the rate of learning." The approach taken in the present study is different from that of Blum's Boosting and Windowing. They are not considered as wrapper methods for atypical detection. This is because Boosting and Windowing at the end of their process, do not identify atypical (outlier or irrelevant) points in a dataset. Indeed they assume that the dataset is rather clean, which is the opposite of what we assume in atypical detection.

2.2.1 Error-reject curves

The concept of error-reject tradeoff curves is rather close to wrapper methods for atypical detection. Chow (1970) introduced this technique for the first time in order to increase the confidence of classifying the test examples. The main idea is that examples close to the decision boundary –the ones we are less sure about– are not classified. They are rejected and might be classified by hand or by another classifier. This limits the classifier to answer only when Confidence/Certainty Factor (CF) is above some pre-specified limit (threshold). So, it is necessary for the classifier to have some kind of approximate posterior probability (reject option) in addition to the hard, unambiguous assignment of labels. By changing the threshold and obtaining new performance measures on test data we can compute an error-reject curve.

Figure II-3 shows the general form of error-reject curves. Performance can be any measure such as accuracy (1 – *error rate*), precision, sensitivity, etc. Reject rate, (1 – *coverage*), is the fraction of the test points that were rejected from the classification and varies between zero and a value for which the performance reaches its highest value. In practice, the curve may not be as smooth as what Figure II-3 depicts, but it is always monotonic.

Although the main purpose in the original design of error-reject curves was not atypical identification (Chow, 1970), it is logically sound and has been used to handle the removal of atypical and ambiguous data (Trappenberg, *et al.*, 1999), and to produce an ordered list

of them ranked by, for instance, the estimate of their posterior probability. Unfortunately, because of its monotonicity, the curve does not give a clue as to when to stop the removal process. Issues with error-reject curves will be discussed later.

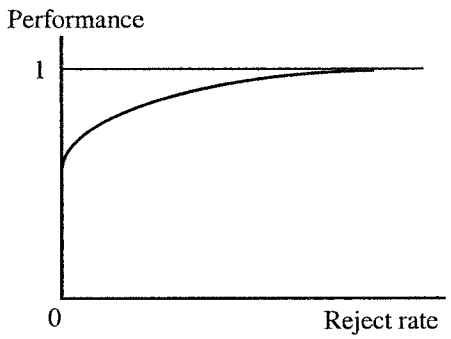

Figure II-3. Error–reject tradeoff curve.

2.3 The quest for performance increase

In classification (unlike regression) a performance increase measured on the training set is not considered to be important by itself as it might be too optimistic (Witten and Frank, 2000). It is, however, the performance on the unseen test data that determines the predictive ability (the generalization power) of a classifier. Hence, in the classification literature, when the words "performance" and "accuracy" are used, they really mean "generalization performance" or "prediction accuracy" unless otherwise is specified.

The race for performance increase seems to be a pivotal idea in most studies somehow related to atypical detection, while there is a point here that has not been addressed well in the literature. Some authors have reported that the removal of some points from the data can enhance the classification performance. In this Section, we examine some of those assertions to see what they have really shown in their reports. Studying their reports carefully, we show that one cannot maintain, based on the reported results, that the removal of some points have led to classifiers with a significantly higher generalization performance overall; for a reasonable number of datasets (noisy and non-noisy) and a variety of classifiers. We will check this point in our experimental results as well.

To the best of the author's knowledge, all the reports of increase in performance, so far, are either indeed insignificant, in a cleaned test set (hard-to-classify examples were removed from the test set), or the cases when some level of noise was injected to the training set. In the latter case, when no noise (0% noise level) was added to the training set, the classifier trained on the filtered training data was either inferior or almost the same as the classifier trained on unfiltered data. Thus, whether or not cleaning the training set can make a classifier with higher performance (on an intact test set) than the baseline classifier (trained on unclean and uncorrupted data) is still an open question. In the following, we examine some of the reported cases from the above 3 categories to clarify the subject.

2.3.1 Cleaning the test set

In the work of Trappenberg and Back (2000), explained before, we noticed that certain points are given the label IDK and on the test data, these points are not considered as misclassified. The authors conclude that the identification of an IDK area and re-classifying the data (both the training and test sets) can lead to a drastic reduction of false predictive classification. Trappenberg, *et al.* (1999) also assert "... This implies that there is a small number of ambiguous data which should be omitted to achieve considerable improved performance." The concept of error-reject curves, as shown before, shows that there is monotonic relationship between the removal of hard-to-classify points and the increase of accuracy. Hashemi and Trappenberg (2002) pursued the same idea and modified error-reject curves into CP curves that gave the bounds (minimum and maximum) on accuracies achievable by a classifier if hard-to-classify points are to be removed from the test set.

It is noticeable that in all these examples, the reported improved performance is not really generalization accuracy = (1 – *classification error*) which is on the intact test set. It is indeed the confidence in classification, expressed as accuracy, which has improved for a subset of the test data (not the whole test data). Hence, those results should not be mistaken with the performance improvement that we generally refer to in machine learning (generalization accuracy).

2.3.2 Injecting noise to the training set

In this case, the test set is kept intact but the training set is altered. In noise reduction techniques, it is customary to add different levels of noise (e.g., 0%, 5%, 10% ...) only to the training data and keep the test set clean (Wilson and Martinez, 2000b). It is true (and understandable) that if the test set is also corrupted, measures of performance cannot be interpreted easily (or may not be reliable anymore). In general, noise can be added to the attribute values (attribute noise) or to the class label (class noise) as in the work of Brodley and Friedl (1999).

The usual scheme in the study of noise reduction techniques is as follows. Some level of noise is added only to the training set; then the filter[1] is applied to the training set and two training sessions are done with the noisy data and the cleaned training data. This leads to two classifiers: classifier *s*, trained with noisy data and classifier *f*, trained with the filtered data. The 3rd classifier (baseline) is sometimes trained on the original training set (no noise, no filter). To show the efficiency of the filtering process, the performance of two classifiers (*s* and *f*) on the intact test set (with no noise added) is measured and compared. If the classifier *f* performs better than the classifier *s*, then they conclude rightfully that the noise was reduced through the filtering process. Different filters and noise levels (e.g., 0%, 5%, 10% ...) can be studied this way.

However, in the case of identifying atypical examples:

---

[1] Filter here was not used as opposed to wrapper; it simply means any instance or noise reduction system.

- We do not add any noise to the training data but we are looking for the atypical points already in the data (0% noise level).
- We do not know, in advance, if atypical points are outliers or not; even if they are assumed to be outliers, we do not know if they are exceptions or noise. Hence, the test set should be kept intact.

The above two issues show that a good comparison can be done between the results of any atypical identification method (such as the one presented in this research) and any noise reduction technique in the case of 0% noise level *only*. It is in 0% noise that a noise filtering system tries to identify the possible noise *already in the training set* (without injecting extra noise) and measures the effect of its removal (from training) on the intact test set. This is basically the same process we plan to perform in our atypical detection system too.

To clarify more about the kinds of training and test sets w.r.t. noise reduction methods, we try to explain the issue in a slightly different way here. Let classifier(x, c) denote the classifier trained on the training data which is explained by two variables: x noise level injected (in %); and the binary variable c showing if cleaning was done on the training data ($c = 1$), otherwise $c = 0$. Hence, the baseline classifier is denoted by classifier(0, 0). To the best of the author's knowledge, in the area of noise reduction, the reports of performance increase (on the intact test data) comes from the comparison of the classifier(x, 0) and the classifier(x, 1) where $x \neq 0$. More interestingly, the highest performance reported in these studies is often the case of baseline, i.e., the classifier(0, 0).

Please, note that when we have an overall performance increase (over a reasonable number of datasets) between the classifier(x, 0) and the classifier(x, 1) when $x \neq 0$, it does not mean that the performance will necessarily increase when $x = 0$ (no noise injection) and c changes from 0 to 1. Indeed, the latter case needs its own experiment. Any expectation such as "removing atypical points from the original training data should increase the test performance because it does when we clean the injected training data" is indeed comparing apples and oranges. Please note that we usually do not know the kind of noise (and its level) already embedded in the data.

The above distinction is very important and may be explained by the following. At the beginning of the process the training set and the test set have almost similar distributions (as a result of partitioning or cross validation over a dataset). When we add noise only to the training set (changing its distribution), the noise reduction process essentially tries to bring the training data back to its initial state. The result is often a new distribution more likely closer to that of the initial one or the test set. Hence we can expect a performance increase going from the classifier(x, 0) to the classifier(x, 1) when $x \neq 0$. But when $x =$ 0% noise is added (unfiltered training and test data have almost the same distribution), the change in the training data, done by instance removal, is more likely to make the filtered training set more different from the test set. As a result, for 0% noise, we may not see the same performance increase anymore. Although this may be the general trend, however, we should note that:

- It does not mean that there cannot be any subtle reduction in the training set that can make the classifier(0, 1) better than the baseline, classifier(0, 0).
- Even identifying the subset of misclassified points that their removal can keep the performance of classifier(0, 1) as high as classifier(0, 0) is still beneficial.

Wilson and Martinez (2000b), in their comprehensive overview of instance selection techniques for exemplar-based learners (KNN-based filters), tested their generalization accuracy against the test set that was not polluted with noise. Once 10% class noise (change of the class labels of 10% of the data) was added only to the training set, some of their noise reduction techniques were able to show better performance (on the intact test set) than the classifier for which training was troubled by noise without filtering. When no noise was added to the training data, *all their tested instance reduction techniques showed generalization accuracies weaker than that of the baseline* (a KNN with 0% noise in the training data and no filtering).

This only supports that cleaning the training set that is already polluted can be a good idea. Interestingly, the accuracy of the baseline without any noise stayed the maximum among all the instance reduction algorithms used in their study (with or without noise).

Almost the same distinction should be made about another interesting work by Brodley and Friedl (1999). They also added class noise to the training set alone and reported that filtering the noise significantly improved the classification accuracy for noise levels up to 30%. Although this statement is true on the cases they experimented, we cannot

generalize it to the case when no noise was injected to the training set. That is, this improvement in accuracy for sure can show the efficiency of the cleaning process but it can hardly support any change in the prediction (generalization) accuracy. The method they introduced for identification errors (filter precision) relies on the case when the number of corrupted instances in data M, is known. Therefore, the given error estimates cannot be used for 0% noise level.

They also found that when no noise was injected, filtering did not make a significant difference even though there was no guarantee that the datasets themselves were noise free. The only dataset which shows that noise filtering made a positive difference was "road segmentation" data. Despite of the efforts to get the dataset, it became known that it is no longer available.

Therefore, in the above examples filter efficiency (expressed as a change in accuracy) was indeed observed. Concerning the generalization performance (for which we have not seen any significant support for a positive change), in practice, it is hard to imagine if we ever need to inject noise to the data and then look for a way to remove it; but we do like to reduce the noise (or more generally, atypical points) already embedded in the data without injecting extra noise to it. Unfortunately, this distinction is not clear in the literature at all, so that one can easily get the impression that performance has been increased significantly by removing some points from the training data. The only case when these two measures (filter efficiency and generalization performance) are truly

equal is when no noise (0% noise level) is added to the training set and the filter is applied to the original data.

2.3.3 Insignificant performance improvement

Similar to the cases of 0% noise in the works of Wilson and Martinez (2000b) and Brodley and Friedl (1999), John (1995) also experienced that Robust-C4.5, specialized in the removal of outliers, degraded performance in some datasets by throwing out points that seemed to be outliers but were indeed valid points. On average, on 21 benchmark datasets, after removing outliers from its training data, Robust-C4.5 performed (accuracy = 84.88%) almost as well as C4.5 (accuracy = 84.42%). Duch, *et al.* (1999), also showed that removing some points from the training set can lead to smaller number of rules but almost the same accuracy for the test set for some benchmark datasets.

The case of Skalak (1994, 1993) is somewhat different. He asserted that "it is possible to maintain or even improve nearest neighbor classification accuracy on out-of-sample data by selecting only a small handful of instances as prototypes." Although he achieved a significant improvement in some of his results, his study is too specific to make a general conclusion. His experiments consist of only one classifier (one-nearest neighbor, which leaves enough room for improvement and may overfit easily) and 4 datasets which exhibit a high degree of internal coherence and external isolation (Skalak, 1994). This means in their datasets, each class has no disjunct cluster (no islands) and all classes are highly separable. He also declares that "such an ideal separation of classes moots the

selection of a prototype, since any instance in an isolated class may give perfect accuracy via a nearest neighbor algorithm." Therefore, although his results inspire more research on the number and quality of training instances, it can hardly be conclusive to be generalized to other datasets or classifiers.

Considering the above discussion, the least we can say is that whether or not cleaning the training set, generally (not a specific dataset or a classifier), can make classifiers with higher performance (on an intact test set) than the baseline classifier (trained on unclean and uncorrupted data) is still an open question.

2.4 Dilemma of handling atypical examples within training and test sets

Some of the algorithms that deal with atypical cases, in general, and outliers in particular, have either kept atypical cases in the training set or discarded them from the test set. However, both these choices have problems associated with them. Keeping them in the training set (as in error-reject curves) can be problematic as these points can be influential and may degrade the performance of the classifier. Concepts like masking and swamping (Barnett and Lewis 1994) from the literature on outliers in regression analysis support this idea indirectly. In classification, Duch, *et al.* (1999), showed that a small number of logical rules that covered the majority (not all) of training examples in some benchmark datasets were enough to obtain high test accuracies. To cover the rest of the examples, the number of rules would explode while accuracy would not increase significantly.

Eliminating atypical cases, on the other hand, from the dataset (both training and test sets as in CP curves and only the test set as in error-reject curves) is done to increase the confidence of prediction and reduce their influence over the decision function. However, this often leads to another problem: loss of vital information, since "one person's noise could be another person's signal" (Han and Kamber, 2001) that is true from both applications and machine learning point of views. While studying astronomical data, in 1777, Daniel Bernoulli faced the same issue and wrote, "I see no way of drawing a dividing line between those that are utterly rejected and those that are wholly retained; it may even happen that the rejected observation is the one that would have supplied the best correction to the others..." (Barnett and Lewis, 1994). In fact, John (1995) experienced the same issue when Robust-C4.5 degraded performance in some datasets by throwing out points that seemed to be outliers but they were indeed perfectly good data belonging to under-represented patterns. Moreover, removing some hard-to-classify points from the test set is problematic to measuring the generalization performance of the classifier; and measures obtained this way are not reliable.

As one can see, deciding on what is to be included or removed from the training and test sets is not a straight forward decision at all. Besides, when reporting performance increase, it seems that we need to be clearer in what we really mean. As a matter of fact, these decisions affect the basic setting for all related experiments and should be decided carefully. That is why different interpretations of performance improvement were explained in the previous Section. As a solution, Trappenberg, *et al.* (1999), introduced a

performance measure which gave different weights to non-classification of the removed points and correctly classified points. Their performance measure is application-dependent, and the user is to choose a scaling function and the value of the required parameter. As a result, their approach does not seem to be straight forward and understandable to all users.

Considering the above problems, the approach in this research (as it will be explained in Section IV) is to stick to 0% noise level and intact test set so that the calculated accuracy is reliable for generalization too. To solve the dilemma, we can search for a subset of misclassified points such that their *removal from training alone* can maintain the classification performance or perhaps enhance it. As for the test set, we can only remove the points that the domain expert has agreed on. Without such permission, *the test set should remain intact.* Since we need to identify the atypical points in the whole dataset (both the training and test subsets), a K-fold cross validation scheme is proposed to keep the training and test subsets separate. The detailed explanation of the proposed wrapper technique for atypical detection is presented in the solution section (Section IV).

III Coverage-performance curves and sample subset selection

In this research, the initial attempts to identify atypical points and to show the effect of their removal led to the development of coverage-performance (CP) curves. In this Section, we explain how CP curves are the modification of error-reject curves and clarify the problems associated with these curves for atypical detection. The expectation for performance improvement as a result of atypical removal is discussed and the dilemma of keeping atypical instances within the training and the test sets are explained.

Sample subset selection is also briefly explained as the basic idea that our wrapper method is built on. Finally, based on the shortcoming of the previous methods, the features we would like a wrapper algorithm for sample subset selection to have is explained as the goal of this research.

3.1 Coverage-performance (CP) curves

In the conventional form of error-reject curves, training is done only once and it is the test set that shrinks gradually. Hashemi and Trappenberg (2002) removed the rejected points from both training and test sets allowing them to retrain the classifier each time a new test removed some points and, thereby, reduce the influence of atypical points on the training

process. They called the obtained curve the coverage-performance (CP) curve. The work of Fumera and Roli (2002) also shows that retraining can enhance the performance in error-reject curves in which the test set is cleaned too. In this Section, the calculation of CP curves for two generated datasets with overlapping samples is presented in more details.

Overlapping (ambiguous) samples, as a kind of atypical data, may cause a serious problem to the classification task. The first generated example is shown in Figure III-1. Figure III-1A shows a training dataset with two attributes and 50 datapoints for each of two classes. The first attribute, x_1, varies uniformly within the interval [-0.8, 0.2] for class 1 and [-.02, 0.8] for class 2. The second attribute, x_2, is also uniformly distributed within [0, 1] and is included only to help to demonstrate the data. Due to the overlapping attribute values in x_1, there is no way to train an algorithm to classify the data points in the region x_1 = [-0.2, 0.2] because data points in this region have equal probability to belong to either of the two classes. Thus, even if an algorithm produces no error in the training set, the upper bound in classification performance on the test set in this example is only about 80%; i.e., 100% in the non-overlapping regions (60% of data), and 50% in the overlapping region (40% of data). That is, 1.0 * 0.6 + 0.5 * 0.4 = 0.8.

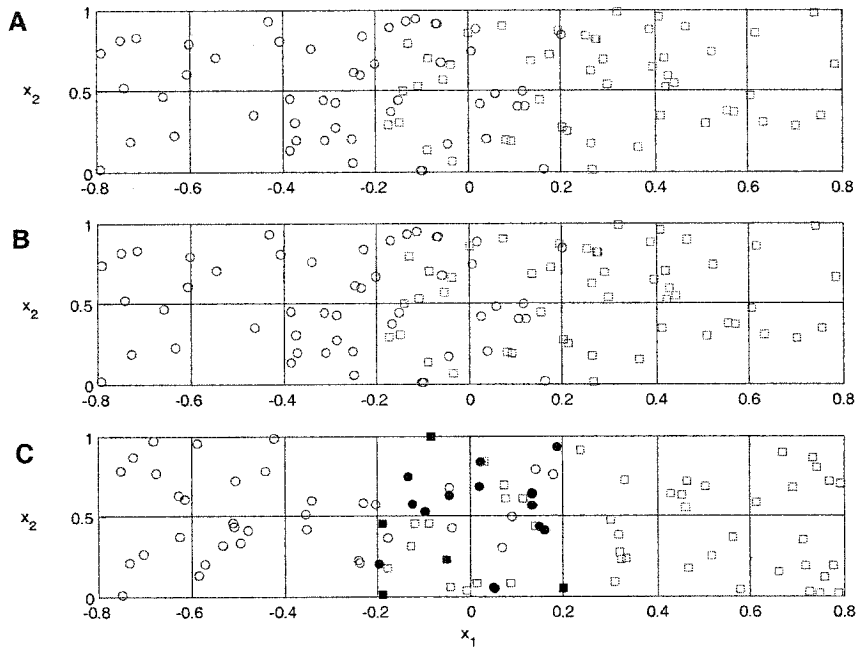

Figure III-1. Example of uniformly distributed overlapping samples. (A): training data; circles are class 1 and squares are class 2. (B): The classification of training data. (C): The classification of test data; false classifications are marked with solid symbols.

Figure III-1B shows the result of training a support vector machine (SVM) applying a Rbf kernel function; all training examples are correctly classified. Figure III-1C shows the result of applying the trained SVM on test data; only 80% of the data were classified correctly.

The second dataset was derived from normally distributed data (Gaussian). For Gaussian data, x_1 has the variance of $\sigma^2 = 1$ and the mean values of -1 and 1 for class 1 and class 2, respectively. The second attribute, x_2, is also uniformly distributed within [0, 1].

3.1.1 Separation scheme and CP curves

Generally speaking, CP curves can be calculated in both filter and wrapper fashions. Hashemi (2002) introduced a scheme to generate CP curves in a filter method. The wrapper scheme to generate CP curves was proposed by Hashemi and Trappenberg (2002). This scheme tries to separate atypical points from both the training and the test sets using an arbitrary classifier with the provision of being able to detect atypical points in its training process. The scheme is based on partitioning the data into the training and test subsets; it is summarized as follows.

On training data:

1. Train classifier 1 using all the training data.
2. Use the information from classifier 1 to divide all points into 2 classes: A (typical) and B (atypical).
3. Train classifier 2 on the training data with new labels (A and B); classifier 2 is atypical detector (separator).
4. Train an additional classifier 3 on only A (typical data) using their original labels.

On test data:

5. Use classifier 2 to remove potential atypical data from the test set (cleaning test data): we get 2 classes $A1$ and $B1$.

6. Use classifier 3 for the classification of $A1$ data. Use original labels to calculate the performance measures. Note that $B1$ (atypical points in the test set) is not classified by classifier 3.

In the second step above, they use some measure to separate the ambiguous data points. This can be done by, for instance, assigning a threshold on posterior probability in probabilistic classifiers, the number of same-class data points found in a KNN algorithm (Trappenberg and Back, 2000), or choosing the bounded support vectors (BSVs) in the case of a SVM (Boser, *et al.*, 1992). The function of classifier 2 is to separate the ambiguous data from the typical ones. In their experiments, they calculate in addition to the performance of classifier 3 (SVM3) in step 6, the performance of classifier 1 (SVM1) in the same step in order to compare the performance of these two classifiers. Note that SVM3 is trained on the clean training data and that both SVM1 and SVM3 are tested on the cleaned test data ($A1$).

The coverage versus performance (CP curve) is calculated to find out how many and which data points to take out from a dataset to have a better classification on the cleaned (typical) data. In general, a CP curve is calculated by first taking out some minimum number of atypical examples in step 2, finishing through step 6, and repeating this process from step 1 to take out some more potentially atypical points from the training set. The reason for such a gradual approach is that (1) atypical points can be influential and training should be repeated for any new subset of training data; and (2) we do not know,

in advance, which points are atypical. Coverage is calculated from the test data as coverage = (number of examples in class $A1$) / (total number of examples in test data).

Every classification algorithm that can somehow distinguish atypical data from regular (typical) data can be used in the above scheme. A SVM classifier was chosen by Hashemi and Trappenberg (2002) with an Rbf kernel function. The involved parameters include C and σ. C determines the tradeoff between minimizing the training error and the model complexity; and parameter σ of the Rbf function defines the nonlinear mapping from input space to some high dimensional feature space. The authors took a fixed σ (obtained initially by parameter optimization) and apply different C values. Each C value gives a different number of bounded support vectors (BSVs) on the training data. BSVs are the most qualified candidates for being atypical data points if their number is chosen properly. This is because they have the largest Lagrange multipliers (Boser, *et al.*, 1992). Note that the number of atypical points is often unknown and that a CP curve can be used to estimate it.

Each time a new C is chosen, they start from step 1 (training with all training data). Thus, points on the CP curve are independent of each other. They found that the resulting coverage by varying C is very sensitive to the C value, leading to clusters with examples around large and small coverage values. To get a sufficient number of examples for intermediary coverage values, they repeated the experiments with different datasets (generated with different random seeds) 100 times.

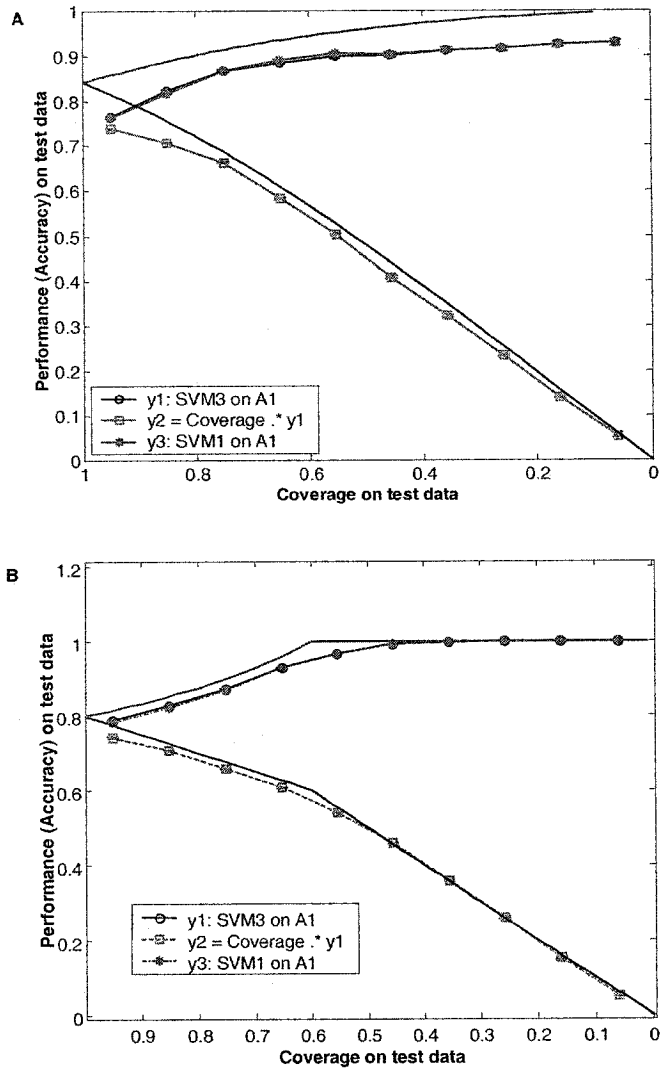

Figure III-2. Coverage performance (CP) curves. Curves without markers are theoretical limits and the ones with markers are experimental results for normally distributed data (A) and uniformly distributed data (B).

In Figure III-2, there are two sets of CP curves representing the results from the datasets with normally distributed (A) and uniformly distributed data (B). In each of these results, the set of monotonically increasing curves represent the performance as measured by the number of correct classifications relative to the number of classified examples (cleaned data). In contrast, the decreasing curves represent the performance as measured by the

number of correct classifications relative to the number of all examples, including non-classified examples. In other words, the non-classified data are simply considered as misclassifications in this performance measure. The curves above thus represent the bounds on any reasonable performance measure. That is, $y1$ and $y3$ (increasing) curves represent the highest accuracies achieved by the classifier (SVM) and $y2$ (decreasing) represent the lowest accuracies SVM can obtain if some % of points are removed from the test data. The solid lines without any marker on them represent the theoretical limits of the performance measures, which can be calculated analytically for these examples considering the known distribution.

Note that, in the above method, the increasing curves do not count the atypical points removed from the test set, in their test performance; also, the decreasing curve considers them as misclassified. In other words, they show the two bounds on the prediction accuracy: the increasing curves are too optimistic and the decreasing curve is a too pessimistic estimate of generalization error.

There is another problem with the above scheme that prevents most classifiers from benefitting from it properly. In step 3, classifier 2 (atypical detector) is trained on a highly imbalanced dataset. For instance, a dataset like Breast-cancer with some 680 points may have less than 15 atypical points. In general, atypical points (especially outliers) may not make a well defined cluster and can be in different locations of the problem space (small disjuncts). Trying to detect atypicals by a classifier trained on such an imbalanced dataset with small disjuncts is known as a problematic approach

(Japkowicz, 2003). Although cost-sensitive classification (not available to all classifiers) has been proposed as a solution to imbalanced data, it is still considered as an open problem (Japkowicz, 2003). Hence, a scheme that relies on such an atypical detector is not advisable.

3.2 Issues with error-reject and CP curves

Error-reject and CP curves can be used with most classifiers. However, if one of the objectives of using these curves is to obtain atypical points, there are certain issues to consider:

1. The evaluation (test) of the classification task becomes problematic because difficult points are gradually removed from the test set. Such a performance is not a reliable measure of generalization error.
2. The curves are always monotonic, not giving any help to the user as to when to stop the removal process. We know that removing the points till the end of the curve (100% performance) is most likely an erroneous idea as these points carry information and discarding them all from the training set may lead to a poor classifier. Note that most of these points are usually in the region where the boundaries of classes are located. But error-reject and CP curves were not designed to answer the question of where to cut off the removal process.

3. If a classifier is not equipped with a reject option (like the typical SVM), computing the curve is troublesome, as was reported in (Hashemi and Trappenberg, 2002).
4. Working on imbalanced data with small disjuncts for atypical detection can be problematic.

3.3 Sample subset selection (SSS)

Detection of a subset of examples (samples) as atypical can be seen as a sample subset selection problem. Feature subset selection (FSS) has been the main area of subset selection so far. Similar to FSS, in which we search for a subset of features with certain qualities (to satisfy an objective function), the term *sample subset selection* (SSS) is used in this research to refer to the problem of finding a subset of examples in a dataset to satisfy the objective function. Although the developed algorithms in FSS are for feature selection, as long as they deal with subset selection, in general, they may be used in SSS as well. Atypical detection can be seen as a sample subset selection problem. The wrapper method presented here is rather similar to feature subset selection proposed by Kohavi and John (1997), except that their wrapper method has a quadratic running time in terms of the number of features. In a survey of feature subset selection methods, Dash and Liu (1997) identified at least three basic parts to any typical FSS method. These parts can work either iteratively or interactively and include:

1. a *subset generation procedure* to generate the next candidate subset,

2. an *evaluation function* to evaluate the quality of the current subset,
3. *stopping criteria* to decide when to stop (e.g. no further increase in evaluation function).

Subset generator (search algorithm) searches the space of feature subsets (sample subset in SSS). Points in this space (different combinations of features or subsets) are called states. There are, generally, three kinds of search algorithms: exhaustive (exponential), sequential algorithms (backward elimination, forward selection, etc.), and randomized algorithms. The evaluation function inputs a state and outputs a numeric evaluation. The search algorithm's goal is to maximize this function.

In a wrapper approach to subset selection, the evaluation function is the performance of the learner and the objective is usually to maximize this performance. In a filter approach to subset selection, the search and evaluation algorithms determine the set of features (or samples) without the classifier being consulted. Filter methods usually use a class separability index as the evaluation function; it measures the degree to which classes are separated and internally cohere.

FSS is basically a search (optimization) problem in a feature set of size n and there are $(2^n - 1)$ states (subsets of one or more features) that can be searched exhaustively to find the optimal feature subset. The number of possible states (features) is usually in the range of 10 to 100; except in full text document collections where it is often much larger. Even for medium-sized n, size of the search space (2^n) is a huge number and an exhaustive search

is prohibitive in practice (Dash and Liu, 1997). As a result, researchers have applied heuristic searches, simplifying assumptions, or even randomized searches (Kohavi and John, 1997; Dash and Liu, 1997; Miller, 1990) as their feature subset generation procedure.

Sample subset selection (SSS) is also a search (optimization) problem in a training set of size N and there are $2^N - 1$ states (subsets of one or more examples) that can be searched exhaustively for the optimal points to be removed. Unlike FSS, in SSS the number of possible states (examples) is usually in the range of a few hundred to several thousand; in the case of information retrieval it is in the range of many thousands. This makes the size of the search space in SSS dramatically huge. Note that an increase of only 10 (in n or N) will result in about 1000 *times* more states than the previous number of states to be searched. That alone can explain why researchers have not tried wrappers on sample subset selection for atypical detection so far, considering that wrapper methods require many retraining sessions that is computationally expensive too. It is indeed like finding a number of needles in a hay stack.

There are techniques like branch-and-bound to skip possibly some states in subset selection (Miller, 1990). For instance, accepting $N/2$ as a minimum size of the training data in inductive learning (as a stopping criterion), the number of states in a branch-and-bound technique is reduced to $(2^{N/2} - 1)$. Still, the remaining number of states is larger than what we can search exhaustively. Considering only the misclassified points as the only candidates for removal also reduces the number of states; yet in most datasets an

exponential search algorithm is far from being a practical option in SSS. For example, a dataset of size 1000 and an error-rate of almost 15% will give at least $N = 150$ misclassified points. The size of the search space (2^{150} = 1.427e+45) is still far from a practical size for a wrapper method.

Generally speaking, other than exponential algorithms, there are two other kinds of algorithms available for subset generation: randomized algorithms (such as GA: genetic algorithms) and heuristic (sequential) algorithms. Skalak (1993) used GA to identify prototypes for one-nearest neighbor algorithm. A simpler GA search strategy (random mutation hill climbing: RMHC) can be found in (Skalak, 1994). The main difference between RMHC and a standard hill climbing (or steepest ascend) search is that in RMHC the length of the binary string representing the chosen subset of data (that is the number of points in the subset) is fixed in advance and mutation is done to find the best evaluated subset. In RMHC, unlike the regular GA approach, different mutations of only one individual (one random initialization of the binary string) are searched for the best evaluated subset. In GA, many individuals are mutated to generate a better final subset (Skalak, 1993). The number of mutations (100) was chosen as a very small number for this type of approach (Skalak, 1994).

Randomized algorithms may randomly ignore some of the atypical points that are important to us (in some applications, identifying atypicals is the objective and more important than the performance). Besides, the ranks of removed points are also an

important piece of information that may not be obtained in randomized search. Therefore, we consider only the sequential algorithms.

The time complexity of a sequential algorithm is $O(m^2)$ (John, *et al.*, 1994) where m is the number of all candidates in the dataset. Thus, when candidates (examples whose removal from training is tried in SSS) are limited to the misclassified points, the overall running time of an inductive classifier used in a wrapper method for sample subset selection will be increased by a multiplicative factor of $O(m^2)$ in the worst case of sequential algorithms. For example, when $m = 150$, we need to retrain the classifier ($150^2 = 22{,}500$) times. If cross validation is performed, the running time will be increased accordingly. To the best of the author's knowledge, this is the fastest available wrapper method. For some classifiers like SVM with nearly a quadratic running time, this can be still prohibitive in practice. Hence, we need to reduce further the time complexity of our SSS process.

Sequential algorithms are greedy algorithms that can be either forward selection or backward elimination. Consider the set of $N = 26$ letters of the alphabet ($a, b, c..., z$) that we would like to find the optimal combination of letters. In forward selection, we start with an empty set and try all 26 letters individually; the one which gives the highest evaluation value (say h) is chosen. Then we try all the 25 2-item sets (h and another letter) and choose the best among them. This process goes on till the stopping criterion is reached. In backward elimination, we start with the full set of letters and each time we remove one letter from the set. The stopping criterion can be, for instance, no more

evaluation value (performance) improvement as we add or remove one more letter. But this imposes the assumption of monotonicity to the objective function that adding more letters can only increase the performance. One can also continue adding or removing letters one by one till all letters are in the subset and choose the subset with the best evaluation value. This can increase the performance as well as the time complexity of the search by a factor of 2 of $O(N^2)$ that is still better than the exhaustive search with $O(2^N - 1)$. When there are a large number of features, instead of removing one at a time, some sequential algorithms, like SBS-SLASH, remove a group of features together (Dash and Liu, 1997; Miller, 1990). This can be a good idea for SSS to reduce the size of the search space.

3.4 The goal (what is needed)

Based on what was mentioned in the previous Section as the requirements to make a wrapper approach to SSS applicable and to address the issues with error-reject and CP curves, we may consider the following six points as the goal for the proposed solution.

1. A time complexity lower than quadratic is preferable. Even the quadratic complexity of sequential algorithms is not good enough for SSS. We would like to remove points in group.
2. The ranks of the removed points are also important; we like to know which points are more atypical than the others.

3. For any given dataset, we would like to identify the set of atypical points for the whole dataset without damaging the validity of the test process (generalization error). This was also a concern with respect to error-reject and CP curves when used for atypical detection (mentioned earlier).

The following three features also come from our discussion on error-reject and CP curves. They are repeated here for completeness.

4. There should be a cut off point for the removal process (non-monotonicity).
5. The methodology of removing atypical points should be so general that even classifiers without a reject option such as the typical SVM can be used.
6. As atypical points are usually in minority in a dataset, it is preferable to avoid classification of imbalanced data with small disjuncts for atypical detection.

IV Solution: Atypical Sequential Removing (ASR)

To address the requirements for atypical detection mentioned in the previous Section, the proposed method for sample subset selection is explained in this Section. Like any subset selection technique, first we need to define what subset of examples we are looking for and then discuss the methodology applied to identify such points in a dataset.

4.1 Definition of atypical points

Atypicality, in this research, is defined from the classifier point of view. The formal definitions of atypical points are given here as follows:

Definition 1

Given an inducer I, and a dataset $\mathcal{D}$ with labeled instances, an ***atypical subset*** *$\mathcal{D}_{Atp}$, is a subset of $\mathcal{D}$ such that the generalization performance of the induced classifier $C = I(\mathcal{D} - \mathcal{D}_{Atp})$ is maximal.*

A problem with the above definition is that it leads to an unmanageable search space: (2^N – 1) states where N is the size of the training set. As it was discussed in Subsection 3.3, a possible shortcut is to consider the set of misclassified points as the only candidates of atypicality. This can be effective because it reduces the number of states dramatically

with minimal loss of information: we have no case against the correctly classified points (they are typical from the classifier view point). Also, for some algorithms such as SVM, removing all within boundary examples (non support vectors) from the training set does not degrade the classifier at all; so, we should not mistaken them with the atypical points. As a result, our definition of atypical points is modified as follows.

Definition 2

Given an inducer I, and a dataset $\mathcal{D}$ with labeled instances, an ***atypical subset*** *$\mathcal{D}_{Atp}$, is a subset of the misclassified set $\mathcal{M}$ such that the generalization performance of the induced classifier $C = I(\mathcal{D} - \mathcal{D}_{Atp})$ is maximal.*

In this definition, the *misclassified set* $\mathcal{M}$ is the set of all the examples misclassified in an iterative process on the training data till the stopping criteria (e.g., state of zero training error) is reached. This set is usually larger than the points misclassified after a single training session; and it helps to capture almost all apprehensive examples to prevent the possible loss of information because of not considering the correctly classified points. Please note that there is no guarantee that the maximum performance will be necessarily higher than the performance of the baseline classifier (with no removal from its training set).

Atypical points usually contain both outliers and overlapping samples. Outliers lie reasonably far from the cluster of their label class in the problem space while overlapping samples are usually close to the boundary of their class and members of more than one

class may fall in the overlapping region. In both cases, to fall in the above definition, atypical points must lie outside of their class boundary (misclassified). There is no attempt in this research to separate outliers from the overlapping samples. The distinction between outliers and genuine exceptions is not performed here either. This is because atypical points are defined here from the classifier point of view.

The term *atypical* and the logic behind the above definition were borrowed from medical practice. In Medicine, when the findings about a patient (case attributes) do not conform to the norm of the disease ($d1$), the case is called atypical. This does not necessarily mean that the case does not belong to the disease $d1$ (so, we keep atypicals in the test set under the class $d1$); but it puts the case in a collection of atypical cases. If a cluster of atypical cases is found -in time- that may lead to the recognition of a new branch or a new kind of the disease ($d2$) by the approval of an authorized body of the domain experts. Only then, cases of $d2$ can be considered as true outliers to $d1$. Examples are abundant: Dementia with Lewy Bodies, Hepatitis C (non-A, non-B), atypical or type 1.5 Diabetes, and hard to diagnose diseases like Lupus and Addison's disease are some examples of the atypical clusters which are given new labels. Severe Acute Respiratory Syndrome (SAR) was also an atypical pneumonia.

More interestingly, an atypical case for physicians in one region may be a very typical case in another. Even different experts in the same region may have their own different views on some subjects. So, atypicality only reflects the view of an expert (particular classifier) who has worked on the cases. By analogy, we must keep atypical detection

and their ranking consistent with the classifier in use. Also, calling cases "outlier" is the task of an authorized body of the domain experts (or perhaps a collection of different classification algorithms for some applications) but finding ranked atypicals can be done from the view point of a single classification algorithm.

4.2 Methodology

After defining atypical points in the context of wrappers, given in the previous Section, we explain the algorithm that tries to identify them in this Section. A modified version of sequential backward elimination was developed to satisfy the time complexity requirement. In this wrapper approach, the classification algorithm itself is used for the evaluation function as in the work of Kohavi and John (1997). A common objective function is to maximize prediction accuracy; however, any other performance measure can be used in this approach.

Misclassification can happen in both training and test sets. But we are only allowed to remove points from the training set because changing the test set damages the performance measure (evaluation) of the classifier. To find the atypical points in the whole dataset (both training and test sets) without sacrificing the evaluation (test) of the classifier, a cross-validation (CV) scheme is proposed. This scheme is presented in Table IV-1 and explained in the following.

4.2.1 Atypical Sequential Removing (ASR) algorithm

ASR is a modified version of sequential backward elimination that tries to reduce the complexity of the standard backward elimination. The input to this algorithm is an arbitrary classifier, a performance measure such as generalization accuracy (as the evaluation function) and the dataset in question. The objective is to identify the set of atypical points in the dataset so that their removal from the training set will benefit or (at least) not damage the test performance of the classifier. The outputs of the algorithm are the set of atypical points in the dataset, average performance measures (e.g., accuracies) with and without removing atypicals, and ranks of atypicals (their frequencies in the K-fold process).

In summary, the presented algorithm identifies the atypical points in the training set in three steps. The first step is to find the misclassification set (the set of all the examples misclassified in an iterative process on the training data till the state of zero training error is reached). The second step is to group and rank the misclassified points. The third step is to identify the atypical points by removing the misclassified groups from the highest rank to the lowest. The atypical points are approximated as the groups that their removal has caused the maximum performance in a validation set. The performance of the classifier trained on the cleaned training set (ASRAcc) is then calculated as well as the performance of a baseline classifier (blineAcc) trained on the whole training set. The comparison of ASRAcc and blineAcc can show how well the detection of atypical points was approximated. According to our discussion on the quest for performance increase

(Subsection 2.3), if ASRAcc is greater than or almost equal to blineAcc, then the approximation can be accepted.

Table IV-1. K-fold CV scheme for ASR algorithm.

```
for i = 1:K,
   trainset: Tr(i) & testset(i) ← Data;                    % divide data into training and test sets
   train classifier1 on Tr(i);                             % classifier1 = baseline
   blineAcc(i) ← test classifier1 on testset(i);           % baseline accuracy
   [coverage, misclpnts, misclNos] ← getMisclfd(trainset);  % get set of Misclassifieds
   misclRanks ← getRank(coverage, misclNos);               % ranking misclassifieds
   [Atypicals{i}, Coverage(i)] ← getAtps(trainset, misclpnts, misclRanks);
                  %  getAtps finds atypicals (not monotonic)
   Trclean(i) ← remove Atypicals{i} from trainset(i);
   train classifier2 on Trclean(i);                        % classifier2 = ASR
   ASRAcc(i) ← test classifier2 on testset(i);             % ASR accuracy
end
blineAcc ← mean(blineAcc);                                 % average over K values
ASRAcc ← mean(ASRAcc);
avgnAtps ← mean(size(Atypicals));
```

Table IV-1 presents a pseudo code of ASR. For each fold of a K-fold cross validation (CV), accuracies with and without removing atypicals are calculated and the set of atypical points –within the training set of the current fold— are identified. In detail, the following is done in each fold.

Data is divided into training (Tr(i) in the pseudo code) and test sets. The first classifier (classifier1 that is our baseline) is trained and tested against the test set. All training

sessions are done in a standard form (with parameter optimization for overfitting prevention, if necessary). Note that the test set (testset) has nothing to do with any of the three used functions: **getMisclfd**, **getRank**, and **getAtps**.

Extracting misclassified set (**getMisclfd** function):

The purpose of this function is to gather all misclassified points (misclpnts) and to record the order of their misclassification that is, the time (order) of their rejection to be used for ranking in **getRank** function. Training and removal of misclassified cases from the training set is repeated till we reach the stopping criteria. The behavior of the **getMisclfd** function is monotonic, like an error-reject curve, but it updates its training after each removal. Figure IV-1 shows this process. The first removal points are the ones that were misclassified in the first training. Training for the 2nd time is done on the points correctly classified in the 1st training. The 2nd removal points are the ones that were misclassified in the 2nd training. This process repeats till we reach the stopping criteria.

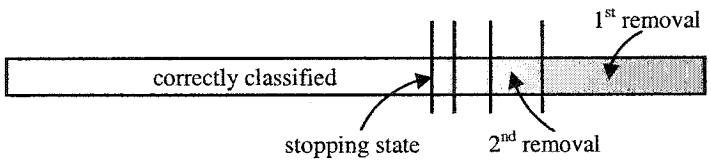

Figure IV-1. Extracting the set of misclassified examples.

The stopping criteria are either zero training error, the removal of all members of a class, or removing 50% of the size of initial training set. Please note, zero training error is not reached at a single training session and must not be mistaken with overfitting. In fact, by removing misclassified points from the training set the chance of overfitting due to outliers becomes even less.

Ranking misclassified points (**getRank** function):

Misclassified points are ranked according to the order (time) of their misclassification. The purpose of this function is to rank the misclassified groups so that they can be searched in by a computationally light search (**getAtps** function, linear in our case) in the next step. Ranking atypical points is not done here. Vectors coverage, and misclNos are of the same size; coverage is the proportion of number of points left in training after every removal to the initial number of points in the training set; and elements of misclNos show how many points in misclpnts are associated with any entry in coverage. Since more than one point may be misclassified in a single test, a group of points can have the same misclassification rank. For instance, if it takes 3 trials to reach the stopping state in **getMisclfd**, the first group of misclassified points is given rank 3, the second rank 2, and the third rank 1. Assuming there are 100 points in the training set initially, the inputs: coverage =[1 .96 .93 .92] and misclNos =[0 4 3 1] will generate the output: misclRanks =[3 3 3 3 2 2 2 1] that shows the ranks of all points misclassified in **getMisclfd** function . The higher the rank number, the more likely the points are atypical.

Finding atypicals (**getAtps** function):

Not all misclassified groups within the list are to be removed, as some may carry useful information to the classification task. In order to find out which group(s) to be removed from the training set, we perform backward elimination from the highest to the lowest ranked groups using a 2^{nd} level k-fold CV on the training set. This validation process starts by removing the group with the highest rank and measuring the performance using the 2^{nd} level k-fold CV on the training set. Then the 1^{st} and the 2^{nd} highest rank groups are removed and performance is measured. This process continues till all misclassified groups are removed from the training of the 2^{nd} level CV. The maximum performance in this sequence determines which groups (subset of the training set) to be removed from training. This CV acts as subset evaluation, that is the evaluation of the objective function (maximum performance). It is part of the search method and has nothing to do with the outer CV loop, which is for the purpose of test (generalization power). The output of this function (Atypicals{i}) are the points that their removal has caused the best performance inside **getAtps**. We hope to see a non-monotonic behavior from **getAtps**. Since groups are removed in sequence, the algorithm is called Atypical Sequential Removing (ASR).

Finally, construct a clean training set, Trclean, train classifier2 on it, and test on the test set. Note that classifier1 and classifier2 are of the same kind and only their training data is different. Since the atypical points are the best to be removed from the training set, we denote Trclean = (training set – atypicals). A single training of classifier2 is finally done on Trclean and it is tested against the unseen test data. The obtained accuracy is called

ASRAcc(i), which is then averaged over its K values. The same averaging is done for baseline accuracy.

In the ASR algorithm, subset generation is done in two functions (**getMisclfd** and **getRank**). The result of these two functions is the identification of misclassified points that are in groups with a specified rank for each group. Thus, subset generation is more elaborate than a traditional sequential algorithm, however, the overall time complexity is significantly less than quadratic. The complexity of **getMisclfd** is $O(ntrains \cdot tc)$, where *ntrains* is the number of times training is done in **getMisclfd** and *tc* is the time complexity of the classifier in use. The average of *ntrains* for the different datasets and the classifiers used in this research is about 5. Hence, subset generation is still a linear process but it turns our combinatorial problem into a linear search. Since we know which points are grouped together and also the ranks of the groups, now we can perform our subset evaluation as a linear search to find atypical points. Subset evaluation is done mainly in **getAtps** function. Evaluation function (check of performance) is also used within **getMisclfd** function for subset generation. The stopping criteria are used in **getMisclfd** function (either zero training error, the removal of all members of a class, or 50% of the size of initial training set). However, in **getAtps** there is no stop and we perform an exhaustive search by trying the removal of all groups sequentially considering their ranks.

In K-fold CV (unlike holdout and bootstrap methods) all examples of a dataset have equal chance of being in the training and test sets. Hence, the possible change in the final rank

of atypical points, due to the order of data, is less than what may be obtained using holdout or bootstrap methods.

4.2.1.1 Time complexity of ASR and the applied heuristics

The overall time complexity of ASR can be computed for two cases. The first case is when we do not use any inner loop cross validation (CV) in **getAtps** function. Since the complexity of **getAtps** would be the same as that of **getMisclfd**, we have:

$$O(K\,(2\,.\,ntrains\,.\,tc)),$$

for a $K = 10$ folds (outer loop) and an average of $ntrains = 5$ (this is the average for the datasets and classifiers used in this research), we have $O(100\,.\,tc)$. The second case is when we use a k-fold CV within **getAtps** function. In that case, we have:

$$O(K\,((k+1)\,.\,ntrains\,.\,tc)),$$

for a $K = 10$ folds of the outer loop, an average of $ntrains = 5$, and $k = 5$ for the CV inside **getAtps** function, we have $O(300\,.\,tc)$. Therefore the complexity of ASR is still a linear factor of the time complexity of the classifier in use (tc). This feature of ASR makes this research viable. As explained in Subsection 3.3, the running time of the fastest available wrapper method (a conventional sequential algorithm) with an average error rate of 15%

is at least[2] $O(22{,}500 \cdot tc)$. Hence, roughly speaking, ASR is at least 22500 / 300 = 75 times faster than the fastest available wrapper method for the average of datasets and classifiers used in this research.

To make the search for atypical points computationally feasible, we have applied three heuristics in ASR. The first heuristic is the focus on the set of misclassified points only, as explained in **getMisclfd**. Since we collect all the misclassified points till the state of zero training error, this heuristic seems to be very reasonable. The second heuristic is in the method of ranking misclassified points: a group of points misclassified together have the same rank. As a result, in **getAtps**, points with the same rank are removed at the same time. This makes ASR less precise because the problem is indeed combinatorial optimization and removing a group of points that were misclassified at the same time can affect the quality of the results. The third heuristic happens in **getAtps**. In backward elimination, once some points are removed we cannot bring any of them back into the next subsets. This also causes ASR not to be precise. More time consuming techniques like sequential replacement algorithms (Miller, 1990) can be used to modify the latter problem.

---

[2] We used "at least" because with 15% error-rate and |dataset| = 1000, we have |misclassified set| >150 (since the iterative process of collecting misclassified points continues till zero training error is reached); beside, cross validation was not considered for the sequential wrapper.

4.2.1.2 ASR with the check option

It is not necessary to remove some points each time that one fold of the ASR scheme runs. Some classification algorithms, like local learners (KNN, CBR ...), may not be good at finding the examples that have a negative influence on the overall performance. Hence, it is beneficial to have an option to check if removing atypicals is possibly useful. In Table IV-2, a simple *check option* is presented that tries to prevent a classification algorithm from performing lower than its baseline. The ASR algorithm with the check option (Table IV-2) follows exactly the same procedure as the regular ASR (Table IV-1) except for the part added for the check option.

The training set for the baseline classifier has no removal. In the check option, we compare the training accuracy of the baseline classifier, TrAccbline, with TrAccASR, which is the accuracy of classifier2 (ASR classifier) on the whole training set of the current fold (Tr(i)). Note that, generally, ASR is trained on a smaller set than the baseline but they are tested on the same dataset. If (TrAccASR + ε) >= TrAccbline, then we can accept the removal process. Otherwise, calculation will be as for the baseline (no removal). The tolerance value, ε, can be used for numerical and randomness inaccuracies (1e – 6, for instance).

Table IV-2. K-fold CV scheme for ASR algorithm with the check option.

```
for i = 1:K,
   trainset: Tr(i) & testset(i) ← Data;                    % divide data into training and test sets
   train classifier1 on Tr(i);                             % classifier1 = baseline
   blineAcc(i) ← test classifier1 on testset(i);           % baseline accuracy
   [coverage, misclpnts, misclNos] ← getMisclfd(trainset);    % get set of Misclassifieds
   misclRanks ← getRank(coverage, misclNos);               % ranking misclassifieds
   [Atypicals{i}, Coverage(i)] ← getAtps(trainset, misclpnts, misclRanks);
                  %  getAtps finds atypicals (not monotonic)
   Trclean(i) ← remove Atypicals{i} from trainset;
   train classifier2 on Trclean(i);                        % classifier2 = ASR
   TrAccASR ← test classifier2 on Tr(i);                   % ASR accuracy on training set
   TrAccbline ← test classifier1 on Tr(i);                 % baseline accuracy on training set
   If TrAccASR >= TrAccbline,                              % check option
     ASRAcc(i) ← test classifier2 on testset(i);           % ASR accuracy
   else
     ASRAcc(i) ← blineAcc(i);  Atypicals{i} = [ ];         % no removal
   end
end
blineAcc ← mean(blineAcc);                                 % average over K values
ASRAcc ← mean(ASRAcc);
avgnAtps ← mean(size(Atypicals));
```

V Experimental setup

Seven classifiers were used in this research. They include:

1. SVM (support vector machine with Rbf kernel function and equipped with parameter optimization), (Vapnik 1995, 1998; Burges 1998); the Matlab interface (OSU-SVM) to the LibSVM implementation (Chang and Lin, 2002) was used. In LibSVM multi-classification is done by 1-vs.-1 (pair wise) method. The optimization of 2 parameters was done by the in-home implementation added to OSU-SVM Matlab interface.
2. KNN (that optimizes for its *k*);
3. CART (classification and regression trees; its Matlab implementation), (Breiman, et al., 1984);
4. NB (Naïve Bayes);
5. QB (quadratic Bayesian);
6. LDA (linear discriminant analysis), (Krzanowski, 1998); and
7. QDA (quadratic discriminant analysis).

Each classifier was tested on twenty datasets with different sizes; with some exceptions such as QB and NB that had a problem with the Dermatology dataset. For the datasets with size <= 1000 examples, repeated stratified 10-fold CV was used. The algorithm in Table IV-1 was run for $K = 10$ (10-fold CV). $K = 10$ was recommended by Kohavi (1995) as the best empirical number for accuracy estimation and model selection. This

was also repeated 5 times to reduce the randomness in the results. Between each of 5 iterations, the data was shuffled randomly. When the dataset has size > 1000, $K = 2$ was chosen and only 1 iteration was done to save time; in large datasets usually the estimate of variance of the performance is not high and there is no need for many cross-validations (Kohavi and John, 1997). $K = 2$ is still more reliable than simple partitioning as it reduces the effect of uneven representation in the training and test sets (Witten and Frank, 2000; and Kohavi, 1995). Indeed, for the same reason 10-fold CV is the standard method of evaluation practically (Witten and Frank, 2000).

We are mainly interested in investigating the behavior of the ASR algorithm without the check option. Once the regular ASR generates satisfactory results for a classifier, there is no need to experiment with the ASR with the check option because they are basically the same. In the case of a significant degradation in the performance of a classifier (from its baseline to its ASR results), one may be interested in experimenting with the ASR with the check option. A thorough comparison of the results with and without the check option is a subject for future studies.

5.1 Ada-Boost and S-Boost

Although the performance measure (accuracy for its popularity) can show how ASR performs compared to the baseline case which has no removal from the training set, we do not know if the removed points (atypicals) carry some useful information to the

classifier in use. One reasonable way to test this issue is to train 2 boosting ensembles: one with the training set without any removal and the other with the training set after removal of atypicals, and then to compare their results.

In boosting, the ensemble is the collection of base classifiers put together as a series that tries to focus more on the misclassified points by giving them more chance to be learned by the next classifiers in the series (Schapire, *et al.*, 1998). In bagging, however, the ensemble works as parallel; the training data is chosen randomly with replacement and any example has an equal chance to be chosen (Breiman, 1996). Since in ASR we deal with selected misclassified points, boosting (not bagging) is a natural choice to investigate if there is useful information in the removed points. Disagreement among base classifiers is the pivotal idea in all ensemble techniques (no gain if base classifiers are identical). This has been shown analytically in (Krogh and Vedelsby, 1995) and empirically by Opitz and Shavlik (1996). A major source of disagreement is the misclassified data; note that disagreement on an example implies that at least one of two classifiers has misclassified the example. In other words, disagreement is always associated with misclassification. That is why boosting focuses on misclassified points.

When Ada-boost is applied to the Subset of data without atypicals, we refer to this as S-boost. So, Ada-boost and S-boost follow exactly the same procedure and use the same test set but different training data. We are interested in observing the difference in the performance of these two.

Hence, if we have removed a significant number of misclassified points from the training data, boosting may not perform as well unless the removed points are not useful to the classification algorithm (atypicals). On the other hand, unlike bagging, boosting ensembles are rather sensitive to noise (Optiz and Maclin, 1999). Although their performance is generally better than bagging, when the dataset is problematic they may degrade substantially. This can be seen from the result of this research as well. Therefore, when we have removed atypical points from the training set, boosting may not degrade as much showing that some of the removed points were indeed atypicals. If it does, or if it degrades even more, then there are still atypicals in the training set and/or there are useful typical points in the removed set (which causes more degradation).

Both boosting cases were also treated as the baseline and ASR in terms of K-fold CV and number of iterations. So, we have 50 ensembles at the end for each of two boosting cases (when |dataset| <=1000) for which the results were averaged at the end. Different authors have recommended different ensemble sizes (number of bootstrap resamples) as the highest number practically necessary (Schapire, *et al.*, 1998 and Breiman, 1998). The highest value (25) from among the suggested ones: 10, 15, and 25 (Opitz and Maclin, 1999) was chosen for our experiments to make sure that Ada-boost has sufficiently reduced test set error and nearly asymptoted to a plateau.

5.2 Comparing ASR with a filter method

ASR, as a wrapper method, can be compared with a well-known filter method to see if our wrapper method can perform as expected. We employed a widely used filter method (Mahalanobis distance measure) to extract outliers from the datasets. The comparison can be done on the quality of the cleaned training data. That is, when the clean data is used for training, we compare the quality of the classifier trained on the data cleaned by the filter with the classifier trained on the data cleaned by our wrapper method (ASR). Also, the more points a method (filter or ASR) can remove the cleaner the training data if their corresponding performance is almost the same.

5.2.1 Mahalanobis distance, a filter for outlier detection

Mahalanobis distance (Ripley, 1996) is one of the widely used measures for outlier detection. In general, Mahalanobis distance is a measure of distance between two points in the space defined by two or more correlated variables. Each example in a multidimensional input space can be plotted as a point. Also, one can plot a point representing the means for all attribute values of a certain class. This "mean point" in the multidimensional space is also called the *centroid* of the class. *Mahalanobis distance* (*Md*) is the distance of an example from the centroid in the multidimensional space, defined by the correlated attributes. *Md* can be written as follows.

$$Md_c(x) = [(x - m_c)^T S_c^{-1} (x - m_c)]^{1/2},$$

where Md_c (scalar, dimensionless) is the Mahalanobis distance of the point $\boldsymbol{x}$ (feature vector) from class c. Often we use squared Md, skipping the sqrt operation. $\boldsymbol{S_c}$ represents the within-class covariance matrix, T stands for transpose, and $\boldsymbol{m_c}$ is the vector of the means of the attribute values for class c. If attributes are uncorrelated, it is the same as the simple Euclidean distance ($\boldsymbol{S_c}$ will become a diagonal matrix with elements representing the variance of each attribute).

One of the main reasons Mahalanobis distance is used is that it is sensitive to the correlations between attributes in the training data. When the attributes are correlated, the axes in the input space can be thought of as being non-orthogonal; that is, they would not be positioned perpendicular to each other. In addition, since Mahalanobis distance is measured in terms of standard deviations from the mean of the training examples ($\boldsymbol{m_c}$), the distance values give a statistical measure of how well the query feature vector ($\boldsymbol{x}$) matches (or does not match) the original examples of class c. Hence, one way to detect outliers is to calculate Md of a set of points from themselves and sort them out (Barnett and Lewis, 1994). In classification, it can be more accurate to calculate Md values for each class separately (a covariance matrix for each class as opposed to a pooled covariance matrix for all the training data) and then sort out the Md values. The outliers detected in this research, as the result of using Md (filter method), follow this method. If the number of examples in a class is small, then calculating a pooled covariance matrix can be more appropriate because calculating a separate covariance matrix may become instable numerically (singularity problem) and may show high variance in estimation.

Since $Md \geq 0$, the cutting point to separate outliers from other points can be determined by Chi squared distribution. The Chi squared distribution is skewed to the right. The two parameters of Chi squared distribution are [*p, df*]. For outlier detection, it is conventional to use $p = (1 - \alpha)$, $\alpha = 0.01$, and *df* = degree of freedom = number of attributes in examples.

Mahalanobis distance is also the basic idea behind discriminant analysis. Once we have the centroid vectors for the classes, we can calculate the *Md* between a query point and all the centroid vectors. The smallest *Md* can determine the estimate of the class label for the query. In Linear Discriminant Analysis (LDA), a pooled covariance matrix for all training data is used while in Quadratic Discriminant Analysis (QDA), a separate covariance matrix is calculated for each class. They give the optimal Bayes rules when the within class data is normally distributed.

5.3 Description of datasets

The 20 datasets used in the study are summarized in Table V-1. They vary across size (number of examples in the dataset) and kinds of attributes (continuous, discrete or both). Among the 20 datasets used, 3 are artificial (generated). All datasets, except for GenNor and GenUni, are chosen from the UCI data repository (Blake and Merz, 1998) from which their documentation can be obtained. The emphasis in choosing datasets was on well-behaved datasets (high accuracies can be achieved) and problematic ones as well as

different sizes. The sizes of the datasets are given with their name to distinguish which version of the dataset was used here. It also helps to compare the coverage values, in the tables of the following Sections, without referring back to this table.

Table V-1. Summary of the datasets used in this research. The names with italic font are generated datasets. The numbers in the parentheses are the number of instances.

Name (size)	Classes	Attributes	
		continuous	discrete
AbaloneAll3(4177)	3	7	1
Balance-scale(625)	3	4	0
BCancerW(683)	2	9	0
Chess(3196)	2	0	36
Cmc(1473)	3	9	0
Crx1(653)	2	6	9
Dermatology(358)	6	33	1
DiabetesPima(768)	2	8	0
Ecoli(332)	5	6	0
GenNor(300)	2	2	0
GenUni(300)	2	2	0
Haberman(306)	2	2	0
Hcleve(296)	2	6	7
Iris(150)	3	4	0
Ionosphere(351)	2	33	0
LiverBupa(345)	2	6	0
SegmentationAll(2310)	7	18	0
Splice1(3190)	3	0	60
Thyroid(7200)	3	6	15
Vehicle(846)	4	18	0

AbaloneAll3 is a modified version of the Abalone dataset. The class label is the age of a kind of sea shell (Abalone). According to the domain expert samples are highly overlapped and the dataset lacks necessary attributes to estimate the age of abalone. Originally there were 29 highly imbalanced classes. The modified version groups the data

into 3 classes: $C(1{:}8)$ into class 1 (1407 examples), $C(9,10)$ into class 2 (1323 examples), and $C(11{:}29)$ into class 3 (1447 examples).

The Balance-scale dataset is a nonlinear classification problem. It has previously been used in cognitive psychology experiments. The class label can be found by calculating the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced (3^{rd} class).

In the Chess (king-rook-vs-king-pawn) dataset, the target is to predict the outcome of the game (the class label) given input attributes, as a set of possible moves of black and white pieces.

Cmc (Contraceptive Method of Choice) is a subset of a socio-economic survey. The dataset is highly undeterministic with possible overlaps between classes.

The Crx1 (credit card approval) dataset is the same as crx dataset without 37 rows (examples) with missing values that were omitted.

In the Ecoli dataset (Protein Localization Sites), 4 examples with missing values were removed. Also, 2 last classes were removed because each had only 2 instances out of 336. The instances of the class with 5 instances (omL) were given to its closest class (om). As a result, 8 classes were reduced to 5 classes.

The GenNor dataset was derived from Normally distributed data (Gaussian) with two classes and two attributes. The first attribute has the variance of $\sigma^2 = 1$ and the mean values of -1 and 1 for class 1 and class 2, respectively. The second attribute is uniformly distributed within [0, 1] and has no effect on class membership.

The GenUni dataset was derived from uniformly distributed data with two classes and two attributes. The first attribute varies uniformly within the interval [-0.8, 0.2] for class 1 and [-.02, 0.8] for class 2. The second attribute is also uniformly distributed within [0, 1] and has no effect on class membership.

Haberman's Survival dataset is from a study conducted between 1958 and 1970 on the survival of patients who had undergone surgery for breast cancer. This dataset is another good example of problematic datasets as it lacks the necessary attributes (as in abalone) to predict the class value. The two attributes used here are the patient's age and the number of positive auxiliary nodes detected.

Iris and BCancerW (breast cancer Wisconsin) are two well-known and well-behaved datasets. Hcleve (heart Cleavland) and Diabetes Pima are also well-known datasets. In Hcleve, 7 instances with missing values were removed.

The LiverBupa (Liver disorder from BUPA Medical Research Ltd.) dataset is another problematic dataset. The highest accuracy obtained on LiverBupa, using mixture of experts, is about 70%.

In the SegmentationAll dataset, the training and test sets of Image Segmentation data were combined to make a single dataset. In this dataset, instances were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a class for every pixel. Each instance is a 3x3 region. Each of 7 classes (brickface, sky, foliage, cement, window, path, and grass) has 330 instances.

Vehicle is from a study to see if 3-D objects (4 types of vehicles) can be recognized from 2-D images. So, there is no or very little noise but not enough attributes. The accuracy obtained, using 10-fold cross validation and Ada-Boosting on a back propagation neural network, was about 80.5% (Maclin and Opitz, 1997).

VI Experimental results

There are three groups of tables in this Section. The first group (Table VI-2 to Table VI-8) compares the performance of baseline (bline Acc) with that of our wrapper method (ASR). The second group of tables (Table VI-10 to Table VI-16) compares Ada-boost (Abst) with S-boost (Sbst) and the baseline (bline). All values are in percent and rounded off to one decimal point, except for *nAtp* and the last column (*ntrains*) in group 1 tables. The third group of tables (Table VI-18 to Table VI-24) shows the comparison of the result of a filter method with that of ASR.

In Subsection 6.1 Tables of group 1, also given are: p-values from McNemar's test; average Coverage; average number of atypicals (*nAtp*) in the training set of size: 9/10 of data for the datasets with size ≤ 1000, and 1/2 of data otherwise; the ratio of *nAtp* to the average number of misclassifieds (*nAtp/nMiscl*); and the average of number of times training was repeated in the getMisclfd function (*ntrains*). All averages, except for p-values, are over K * number of iterations. *Coverage* is the proportion of points left in the training set after removing atypicals.

McNemar's test (Ripley, 1996) is a nonparametric test of proportion. It is designed for the test of binary outcomes (correct and incorrect classifications) and uses a binomial distribution. In McNemar's test, when the performance of two classifiers are compared, concordant pairs (the instance for which two classifiers have the same answers) are

discarded and only discordant pairs are considered. In discordant pairs, correct classifications for classifier *A* are exactly incorrect classifications for classifier *B*. When the number of elements in the discordant pairs vector is large (>100, in the implementation of this research), a parametric approximation like Normal approximation, is used to get around the numerical problem of calculating very large factorials.

For McNemar's test, a p-value is calculated for each iteration and averaging is done over the number of iterations. The shown results are for 1-tailed test. For an individual item, let μ be the probability of classifier *A* classifies correctly given that only one of *A* or *B* can classify correctly. Therefore, the hypotheses are: H_o (Null hypothesis): "population mean, $\mu = 1/2$", i.e., two classifiers are the same. H_a (alternative): "$\mu > 1/2$", e.g., (number of correct classifications in classifier *A*) > (number of correct classification for classifier *B*); or *A* is better than *B*; in the following tables, this is shown as McN, Pv(*A*>*B*).

Note that p-values do not provide us with a simple Yes or No answer; they provide a sense of the strength of the evidence against the null hypothesis. A p-value is basically a measure of how much evidence we have against the null hypothesis; the lower the p-value, the stronger the evidence. A low p-value for the statistical test points to the rejection of the null hypothesis because it indicates how unlikely it is that a test statistic as extreme as or more extreme than the one given by the experiment will be observed from this population if the null hypothesis is true. For instance, McN, Pv(*A*>*B*) = 15%, means that if the population means were equal as hypothesized (under the null), there is a 15%

chance that a more extreme test statistic would be observed (*A* better than *B*) using our experiments on this population. If you agree that this is enough evidence to reject the null hypothesis, you conclude that there is significant evidence to support the alternative hypothesis (or *A* is better than *B*). The researcher (or reader) decides what significance level to use –that is, what cut off point will decide significance (domain dependent). The most commonly used level of significance is $\alpha = 0.05$. When the significance level is set at 0.05, any test resulting in a p-value under 0.05 would be statistically significant. Therefore, you would reject the null hypothesis in favor of the alternative hypothesis.

When the p-value is too high, it means that we cannot reject the null hypothesis (not enough evidence). When the performance of two classifiers are the same (discordant pairs vector is empty), NaN is generated.

6.1 Tables of group 1: ASR and baselines

To make the descriptions of the column headers easily accessible, the descriptions for the Tables of group 1 are given in Table VI-1.

Table VI-1. The descriptions of column headers for tables of group 1.

No	Column header	Description
1	**dataset(size)**	The name of dataset and its size
2	**bline Acc**	The average accuracy of the baseline (no removal), (%)
3	**ASR Acc**	The average accuracy of the ASR classifier (with removal), (%)
4	**McN, Pv(bl>ASR)**	McNemar's p-value, if baseline is better than ASR, (%)
5	**McN, Pv(ASR>bl)**	McNemar's p-value, if ASR is better than baseline, (%)
6	**avg Coverage**	The average coverage, (%) of points left in the training set
7	**avg nAtp**	The average number of atypical points in the training set
8	**nAtp/nMiscl**	(%) of points in the misclassified set that were found as atypical
9	**avg ntrains**	The average no. of times training was repeated in getMisclfd

Table VI-2. Comparison of baseline and ASR for SVM.

SVM		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	66.0	65.5	2.7	97.3	69.3	641.5	97.3	5.5
Balance-scale(625)	97.9	98.0	62.7	48.0	99.7	1.6	55.6	1.6
BCancerW(683)	96.9	96.9	66.0	56.8	98.0	12.2	69.5	2.8
Chess(3196)	99.0	98.9	31.3	93.8	99.9	2.0	100.0	2.5
Cmc(1473)	53.4	51.8	1.2	98.8	61.2	286.5	100.0	5.5
Crx1(653)	85.3	85.3	52.5	51.2	91.6	49.5	64.5	3.1
Dermatology(358)	97.4	97.0	48.5	71.7	99.2	2.7	39.1	3.1
DiabetesPima(768)	77.1	76.8	45.9	57.5	79.7	140.8	91.6	4.4
Ecoli(332)	87.9	89.1	75.5	36.9	91.1	26.8	86.3	3.6
GenNor(300)	85.4	86.0	67.3	46.1	87.8	32.9	87.3	3.4
GenUni(300)	79.8	80.7	65.7	45.0	82.1	48.3	88.1	3.3
Haberman(306)	74.2	74.9	64.5	46.8	65.4	95.4	90.7	3.2
Hcleve(296)	83.4	83.9	66.0	42.2	87.8	32.7	83.7	3.8
Iris(150)	96.1	95.6	56.7	65.6	98.4	2.2	62.7	2.5
Ionosphere(351)	94.8	94.5	48.3	66.1	99.1	2.9	38.6	3.0
LiverBupa(345)	70.3	70.6	57.2	45.8	82.1	55.6	98.4	3.6
SegmentationAll(2310)	95.1	94.7	8.7	95.2	97.3	31.0	87.8	4.5
Splice1(3190)	96.0	95.7	12.6	92.4	96.9	49.5	90.0	3.5
Thyroid(7200)	97.3	97.5	92.0	12.1	99.3	26.0	94.6	3.5
Vehicle(846)	83.8	83.4	40.1	60.9	96.6	25.7	84.8	3.7
Average:	**85.9**	**85.8**	**48.3**	**61.5**	**89.1**	**78.3**	**80.5**	**3.5**

Table VI-3. Comparison of baseline and ASR for KNN.

KNN		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	63.1	62.4	9.0	91.0	65.9	713.5	85.4	10.0
Balance-scale(625)	90.3	85.3	0.0	100.0	94.1	33.2	73.1	2.0
BCancerW(683)	97.0	96.7	42.4	69.0	97.1	17.9	85.6	3.6
Chess(3196)	90.8	90.8	NaN	NaN	100.0	0.0	0.0	9.5
Cmc(1473)	49.9	47.0	0.4	99.6	53.8	341.0	83.7	14.0
Crx1(653)	79.1	77.9	30.4	70.0	95.4	26.9	18.9	6.8
Dermatology(358)	95.5	95.5	61.1	57.1	97.6	7.7	50.0	4.0
DiabetesPima(768)	75.1	74.8	46.6	55.1	75.3	171.2	84.4	8.5
Ecoli(332)	87.5	87.8	56.9	56.5	91.1	26.7	69.5	4.5
GenNor(300)	84.9	85.7	66.2	45.1	85.9	38.0	92.3	4.0
GenUni(300)	77.5	76.3	37.6	66.0	78.9	57.0	90.4	4.5
Haberman(306)	75.8	74.8	39.4	68.5	64.5	97.9	84.2	4.3
Hcleve(296)	82.0	82.4	61.5	49.1	84.9	40.4	85.9	4.0
Iris(150)	95.3	94.7	51.2	71.1	97.5	3.3	55.0	2.6
Ionosphere(351)	84.5	84.6	54.5	54.0	93.8	19.8	35.4	3.7
LiverBupa(345)	63.4	63.9	54.6	47.4	67.2	102.0	78.2	7.3
SegmentationAll(2310)	92.8	95.2	99.9	0.0	99.8	2.0	1.9	8.5
Splice1(3190)	81.4	74.3	0.0	99.9	99.8	2.5	0.4	18.0
Thyroid(7200)	94.7	94.7	NaN	NaN	100.0	0.0	0.0	5.5
Vehicle(846)	72.5	68.9	4.1	95.9	82.2	136.2	48.5	10.8
Average:	**81.7**	**80.7**	**39.8**	**66.4**	**86.2**	**91.9**	**56.1**	**6.8**

Table VI-4. Comparison of baseline and ASR for CART.

CART		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	62.7	62.8	52.3	47.7	67.9	670.0	100.0	2.5
Balance-scale(625)	79.3	79.7	99.9	25.0	96.9	17.3	98.9	1.3
BCancerW(683)	94.5	95.1	75.3	32.8	96.8	19.7	93.8	2.5
Chess(3196)	99.1	99.1	25.0	99.9	99.3	10.5	100.0	2.5
Cmc(1473)	54.1	54.1	50.0	61.0	56.1	324.0	100.0	2.5
Crx1(653)	86.4	86.2	49.9	55.0	87.3	74.5	96.8	2.1
Dermatology(358)	94.5	94.9	65.8	54.4	97.6	7.7	96.9	2.2
DiabetesPima(768)	74.9	75.7	69.5	33.6	78.6	148.1	99.2	2.7
Ecoli(332)	79.9	82.6	82.0	22.9	87.9	36.4	89.6	3.0
GenNor(300)	82.8	81.6	40.5	72.7	87.5	33.9	92.5	2.3
GenUni(300)	80.7	78.6	32.2	73.0	85.5	39.0	92.8	2.7
Haberman(306)	72.3	71.5	45.0	68.1	80.7	53.2	100.0	1.7
Hcleve(296)	78.3	78.7	60.9	47.3	84.8	40.5	96.5	2.3
Iris(150)	93.6	93.6	56.5	65.7	97.1	3.9	77.1	2.4
Ionosphere(351)	89.4	88.6	42.6	69.6	92.4	24.0	91.0	2.2
LiverBupa(345)	67.5	67.4	50.6	51.6	78.3	67.4	87.2	2.5
SegmentationAll(2310)	94.0	94.4	67.6	32.4	97.9	23.8	74.6	3.2
Splice1(3190)	94.4	92.9	0.0	100.0	97.4	41.5	50.0	2.5
Thyroid(7200)	99.4	99.5	97.1	9.0	99.7	9.5	100.0	2.0
Vehicle(846)	68.7	69.2	61.6	38.4	79.5	157.0	89.3	4.9
Average:	**82.3**	**82.3**	**56.2**	**53.0**	**87.5**	**90.1**	**91.3**	**2.5**

Table VI-5. Comparison of baseline and ASR for Quadratic Baysian.

QB		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	61.4	61.9	82.9	17.1	60.2	831.0	92.6	9.0
Balance-scale(625)	91.9	91.9	NaN	NaN	100.0	0.0	0.0	2.0
BCancerW(683)	95.2	95.0	47.1	65.7	97.8	13.5	31.7	4.6
Chess(3196)	66.7	66.7	NaN	NaN	100.0	0.0	0.0	6.5
Cmc(1473)	52.2	52.0	27.1	84.6	99.0	7.5	1.6	4.5
Crx1(653)	80.6	80.3	19.8	96.4	98.8	6.8	5.1	5.9
DiabetesPima(768)	74.1	73.8	49.0	52.6	74.8	174.3	93.4	5.6
Ecoli(332)	57.5	57.0	1.1	99.9	99.6	1.1	2.1	2.0
GenNor(300)	85.3	86.0	69.6	39.4	86.7	35.9	89.7	2.9
GenUni(300)	79.3	80.0	68.9	42.3	82.4	47.6	85.5	2.8
Haberman(306)	75.2	74.1	36.4	73.1	65.9	94.1	77.6	7.4
Hcleve(296)	82.6	81.5	19.6	87.5	95.0	13.4	28.9	4.2
Iris(150)	97.2	97.2	63.7	63.0	98.9	1.5	53.3	2.1
Ionosphere(351)	90.8	90.8	56.5	53.8	98.0	6.4	44.0	2.0
LiverBupa(345)	60.7	59.6	38.5	61.5	99.3	2.1	1.4	6.2
SegmentationAll(2310)	62.3	62.3	NaN	NaN	100.0	0.0	0.0	1.5
Splice1(3190)	84.8	84.7	50.0	99.9	99.9	1.0	1.3	2.5
Thyroid(7200)	95.5	96.8	99.9	0.0	94.3	203.5	98.6	5.5
Vehicle(846)	85.3	84.6	32.5	69.1	90.3	73.9	94.5	5.4
Average:	**77.8**	**77.7**	**47.7**	**62.9**	**91.6**	**79.7**	**42.2**	**4.3**

Table VI-6. Comparison of baseline and ASR for Naive Bayes.

NB		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	57.8	57.5	13.3	90.6	78.0	459.0	50.0	5.0
Balance-scale(625)	91.6	91.6	NaN	NaN	100.0	0.0	0.0	1.0
BCancerW(683)	96.3	96.3	NaN	NaN	100.0	0.0	0.0	4.0
Chess(3196)	66.7	66.7	NaN	NaN	100.0	0.0	0.0	6.5
Cmc(1473)	47.1	46.7	25.6	83.8	97.4	19.5	3.8	3.5
Crx1(653)	80.2	80.2	NaN	NaN	100.0	0.0	0.0	7.1
DiabetesPima(768)	75.7	75.7	47.2	58.3	96.6	23.5	11.7	11.0
Ecoli(332)	63.2	63.2	NaN	NaN	100.0	0.0	0.0	1.2
GenNor(300)	85.3	85.3	58.1	75.4	90.3	26.3	65.4	2.6
GenUni(300)	79.9	80.7	69.1	40.6	80.0	54.1	97.2	3.2
Haberman(306)	74.6	74.3	48.6	62.9	64.5	98.0	80.3	8.3
Hcleve(296)	83.0	83.2	49.7	66.0	98.2	4.8	9.8	4.1
Iris(150)	95.2	95.2	60.9	60.9	98.1	2.6	45.7	2.4
Ionosphere(351)	80.6	80.5	43.3	85.0	92.8	22.8	35.9	2.5
LiverBupa(345)	54.7	53.7	46.7	62.6	91.9	25.1	15.6	4.3
SegmentationAll(2310)	72.6	72.5	31.3	93.8	98.3	19.5	5.2	2.5
Splice1(3190)	84.8	84.7	50.0	99.9	99.9	1.0	1.3	2.5
Thyroid(7200)	95.3	96.8	99.9	0.0	94.5	197.0	92.1	7.0
Vehicle(846)	45.4	48.7	92.9	7.1	44.8	421.8	92.9	6.2
Average:	**75.3**	**75.4**	**52.6**	**63.4**	**90.8**	**72.4**	**31.9**	**4.5**

Table VI-7. Comparison of baseline and ASR for Quadratic Discriminant Analysis.

QDA		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	61.4	61.4	100.0	100.0	100.0	0.0	NaN	16
Balance-scale(625)	91.6	91.8	57.7	54.7	95.7	24.4	65.3	7.1
BCancerW(683)	90.3	94.7	99.9	0.2	63.1	227.2	52.9	8.16
Chess(3196)	88.7	88.7	100.0	100.0	100.0	0.0	NaN	15.5
Cmc(1473)	49.4	49.4	57.0	57.0	97.0	22.5	5.3	4
Crx1(653)	80.6	80.7	51.5	51.1	83.8	95.5	76.2	8.22
DiabetesPima(768)	66.8	66.8	100.0	100.0	100.0	0.0	NaN	10.96
Ecoli(332)	85.5	85.9	63.2	47.4	89.2	32.4	83.3	5
GenNor(300)	85.2	85.7	67.6	42.2	86.8	35.6	99.6	2.84
GenUni(300)	79.7	78.9	45.0	68.9	84.4	42.1	99.2	3.06
Haberman(306)	71.8	73.1	72.5	38.1	87.0	35.8	34.8	9.18
Hcleve(296)	77.7	76.6	38.2	69.1	83.1	45.1	77.5	8.8
Iris(150)	97.5	97.5	70.5	72.5	99.3	0.9	64.5	3.32
Ionosphere(351)	61.1	61.1	100.0	100.0	100.0	0.0	NaN	4.06
LiverBupa(345)	69.1	68.6	47.6	68.4	98.1	5.8	9.2	6.3
SegmentationAll(2310)	89.1	89.2	100.0	50.0	99.8	2.0	2.1	9
Thyroid(7200)	91.5	91.5	100.0	100.0	100.0	0.0	NaN	2
Vehicle(846)	84.3	83.9	37.9	64.0	91.3	66.4	83.1	7.4
Average:	**79.0**	**79.2**	**72.7**	**65.8**	**92.2**	**35.3**	**57.9**	**7.3**

Table VI-8. Comparison of baseline and ASR for Linear Discriminant Analysis.

LDA		ASR						
	bline	ASR	McN, Pv	McN, Pv	avg	avg	nAtp/	avg
dataset(size)	Acc	Acc	(bl>ASR)	(ASR>bl)	Coverage	nAtp	nMiscl	ntrains
AbaloneAll3(4177)	61.8	61.5	21.6	78.4	81.8	380.0	78.7	7
Balance-scale(625)	70.2	70.2	100.0	100.0	100.0	0.0	NaN	3.12
BCancerW(683)	96.3	96.2	90.0	96.3	99.9	0.9	95.6	3.32
Chess(3196)	93.4	93.4	100.0	100.0	100.0	0.0	NaN	9.5
Cmc(1473)	48.9	48.8	50.0	66.1	98.9	8.0	3.6	8.5
Crx1(653)	84.0	86.1	88.0	12.4	81.6	108.6	98.1	3.46
Dermatology(358)	96.7	96.4	53.6	78.8	99.3	2.3	94.7	2.66
DiabetesPima(768)	76.3	76.3	51.6	54.1	87.0	90.0	95.3	5.82
Ecoli(332)	88.6	88.6	100.0	100.0	100.0	0.0	NaN	5.82
GenNor(300)	85.3	85.7	63.8	47.8	86.1	37.6	98.9	2.9
GenUni(300)	79.3	79.9	67.9	40.6	81.1	51.0	99.6	2.68
Haberman(306)	74.1	75.2	68.4	39.6	55.0	124.2	96.9	5.4
Hcleve(296)	78.0	79.9	68.8	34.8	80.3	52.5	82.2	6.48
Iris(150)	98.0	98.3	88.8	72.5	99.7	0.4	100.0	2
Ionosphere(351)	86.0	86.4	61.7	48.0	90.6	29.9	95.1	3.74
LiverBupa(345)	63.4	62.6	39.9	62.0	90.0	31.1	31.6	5.18
SegmentationAll(2310)	91.7	90.8	0.0	100.0	95.5	52.5	53.2	4.5
Thyroid(7200)	63.1	69.2	100.0	0.0	80.4	705.5	49.6	8.5
Vehicle(846)	77.9	77.0	29.2	72.5	86.5	103.5	89.5	5.98
Average:	**79.6**	**80.1**	**65.4**	**63.4**	**89.1**	**93.6**	**78.9**	**5.1**

6.2 Tables of group 2: Ada-Boost vs. S-boost

The datasets are divided into the ones for which blineAcc >= AbstAcc (i.e., Ada-boost has not helped in classification accuracy) and the ones for which blineAcc < AbstAcc (i.e., Ada-boost has helped in classification accuracy). This was only done to get a sense of how each dataset was perceived by the classifier. That is, if the baseline classifier has learned the dataset enough (the upper portion of Tables) or if there was still information in the misclassified points that could be learned by a boosting process that focuses on misclassified examples (the lower portion of Tables). To make the descriptions of the column headers easily accessible, the descriptions for the Tables of group 2 are given in Table VI-9.

Table VI-9. The descriptions of column headers for tables of group 2.

No	Column header	Description
1	**dataset(size)**	The name of dataset and its size
2	**bline Acc**	The average accuracy of the baseline (no removal), (%)
3	**Abst Acc**	The average accuracy of Ada-boost, (%)
4	**McN, Pv(Abst>bl)**	McNemar's p-value, if Ada-boost is better than baseline, (%)
5	**Sbst Acc**	The average accuracy of S-boost (no atypical), (%)
6	**McN, Pv(Abst>Sbst)**	McNemar's p-value, if Ada-boost is better than S-boost, (%)
7	**McN, Pv(Sbst>Abst)**	McNemar's p-value, if S-boost is better than Ada-boost, (%)

Table VI-10. Comparison of Ada-boost and S-boost for SVM.

SVM		Ada-boost Vs S-boost				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
AbaloneAll3(4177)	66.0	65.7	79.5	65.2	15.7	84.3
Balance-scale(625)	97.9	97.9	64.1	98.0	71.5	52.3
BCancerW(683)	96.9	96.0	93.1	96.7	82.9	26.7
Chess(3196)	99.0	98.8	97.5	98.7	33.9	79.8
Cmc(1473)	53.4	52.7	87.7	51.9	15.2	84.8
Crx1(653)	85.3	84.1	79.0	85.4	68.8	33.0
Dermatology(358)	97.4	96.3	87.0	96.1	49.1	65.9
DiabetesPima(768)	77.1	76.5	71.1	76.4	50.1	51.8
Ecoli(332)	87.9	86.1	92.1	88.1	82.5	24.0
GenNor(300)	85.4	84.5	81.5	85.7	73.1	35.7
GenUni(300)	79.8	79.6	55.6	79.9	57.0	49.8
Haberman(306)	74.2	74.2	53.0	74.9	59.8	47.9
Hcleve(296)	83.4	81.9	81.3	83.2	69.5	37.8
Iris(150)	96.1	96.1	70.0	95.9	62.0	63.7
Ionosphere(351)	94.8	93.8	84.7	95.2	82.6	27.6
LiverBupa(345)	70.3	68.6	76.0	69.4	56.6	45.5
Splice1(3190)	96.0	93.3	99.9	95.5	99.9	0.0
Thyroid(7200)	97.3	97.1	95.6	97.5	98.9	1.1
Vehicle(846)	83.8	82.6	84.8	83.7	73.7	26.7
sub Average:	**85.4**	**84.5**	**80.7**	**85.1**	**63.3**	**44.1**
SegmentationAll(2310)	95.1	96.3	0.1	95.7	8.3	94.7
sub Average:	**95.1**	**96.3**	**0.1**	**95.7**	**8.3**	**94.7**
Total Average:	**85.9**	**85.1**	**76.7**	**85.7**	**60.5**	**46.7**

Table VI-11. Comparison of Ada-boost and S-boost for KNN.

KNN		Ada-boost Vs S-boost				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
AbaloneAll3(4177)	63.1	60.0	99.9	61.7	99.4	0.6
Balance-scale(625)	90.3	82.5	99.9	77.5	0.8	99.3
BCancerW(683)	97.0	93.8	99.9	96.0	97.6	4.1
Cmc(1473)	49.9	48.3	90.7	46.8	10.9	89.1
Crx1(653)	79.1	77.1	85.7	78.1	66.5	34.7
Dermatology(358)	95.5	91.3	99.9	92.1	67.6	43.0
DiabetesPima(768)	75.1	71.6	96.5	74.4	90.3	9.7
Ecoli(332)	87.5	78.4	99.9	85.9	99.2	1.3
GenNor(300)	84.9	78.4	99.4	85.7	98.4	2.6
GenUni(300)	77.5	75.4	80.8	76.1	56.2	45.0
Haberman(306)	75.8	73.3	85.3	74.0	63.3	44.5
Hcleve(296)	82.0	76.1	96.7	79.5	76.2	26.1
Iris(150)	95.3	91.7	98.9	93.7	83.3	30.2
LiverBupa(345)	63.4	60.9	81.3	62.1	63.3	36.7
Thyroid(7200)	94.7	92.5	99.9	92.5	NaN	NaN
Vehicle(846)	72.5	71.0	83.9	67.9	10.0	90.0
sub Average:	**80.2**	**76.4**	**93.7**	**77.8**	**65.5**	**37.1**
Chess(3196)	90.8	94.1	0.0	94.1	NaN	NaN
Ionosphere(351)	84.5	84.9	47.4	85.8	66.8	42.6
SegmentationAll(2310)	92.8	93.9	1.3	94.8	99.9	0.1
Splice1(3190)	81.4	82.6	5.1	73.3	0.0	99.9
sub Average:	**87.4**	**88.9**	**13.5**	**87.0**	**55.6**	**47.5**
Total Average:	**81.7**	**78.9**	**77.6**	**79.6**	**63.9**	**38.9**

Table VI-12. Comparison of Ada-boost and S-boost for CART.

CART		Ada-boost Vs S-boost				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
AbaloneAll3(4177)	62.7	60.6	99.7	62.9	99.9	0.1
Balance-scale(625)	79.3	77.2	93.1	77.3	68.5	50.0
Cmc(1473)	54.1	50.8	99.0	54.1	99.1	0.9
Crx1(653)	86.4	86.1	61.7	86.7	62.7	38.2
DiabetesPima(768)	74.9	73.6	77.9	75.8	86.4	13.6
GenNor(300)	82.8	82.0	71.5	82.3	58.5	50.2
GenUni(300)	80.7	79.4	71.0	78.7	43.5	60.2
Haberman(306)	72.3	70.1	80.7	70.0	57.1	54.8
Splice1(3190)	94.4	94.1	76.8	94.0	36.9	63.1
sub Average:	**76.4**	**74.9**	**81.3**	**75.8**	**68.1**	**36.8**
BCancerW(683)	94.5	96.6	2.5	96.1	38.3	71.9
Chess(3196)	99.1	99.2	50.0	99.1	41.9	72.9
Dermatology(358)	94.5	97.2	3.2	96.0	27.3	83.8
Ecoli(332)	79.9	86.6	0.5	83.2	13.3	90.8
Hcleve(296)	78.3	79.2	44.0	78.9	51.5	55.1
Iris(150)	93.6	94.7	42.5	93.7	44.9	74.2
Ionosphere(351)	89.4	93.6	0.8	89.9	4.5	97.7
LiverBupa(345)	67.5	68.9	34.1	68.1	41.8	59.8
SegmentationAll(2310)	94.0	98.1	0.0	97.4	26.2	77.1
Thyroid(7200)	99.4	99.5	9.2	99.5	50.0	69.6
Vehicle(846)	68.7	76.9	0.0	72.1	2.5	97.5
sub Average:	**87.2**	**90.0**	**17.0**	**88.5**	**31.1**	**77.3**
Total Average:	**82.3**	**83.2**	**45.9**	**82.8**	**47.7**	**59.1**

Table VI-13. Comparison of Ada-boost and S-boost for Quadratic Bayesian.

QB		Ada-boost Vs S-boost				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
BCancerW(683)	95.2	95.1	55.5	95.1	50.3	58.2
Cmc(1473)	52.2	52.0	77.4	51.6	31.3	76.8
GenNor(300)	85.3	85.3	62.5	85.4	57.5	55.9
Haberman(306)	75.2	74.7	77.9	75.3	67.8	44.4
Hcleve(296)	82.6	82.6	56.8	82.1	34.8	78.0
Iris(150)	97.2	96.7	90.0	96.5	59.1	62.4
Ionosphere(351)	90.8	89.9	91.1	89.6	51.1	66.8
Vehicle(846)	85.3	82.5	99.6	85.1	90.1	9.9
sub Average:	**83.0**	**82.3**	**76.4**	**82.6**	**55.3**	**56.6**
AbaloneAll3(4177)	61.4	61.9	2.8	61.9	45.4	54.6
Balance-scale(625)	91.9	94.0	0.4	94.0	NaN	NaN
Chess(3196)	66.7	94.7	0.0	94.7	NaN	NaN
Crx1(653)	80.6	83.3	2.0	83.2	66.0	67.6
DiabetesPima(768)	74.1	74.8	30.4	74.7	50.6	51.1
Ecoli(332)	57.5	81.9	0.0	81.7	6.3	99.9
GenUni(300)	79.3	79.5	57.8	80.1	67.2	41.7
LiverBupa(345)	60.7	68.6	0.2	68.1	45.5	59.4
SegmentationAll(2310)	62.3	75.8	0.0	75.8	NaN	NaN
Splice1(3190)	84.8	94.0	0.0	93.6	6.3	96.7
Thyroid(7200)	95.5	95.8	0.0	96.1	94.8	5.2
sub Average:	**74.1**	**82.2**	**8.5**	**82.2**	**47.8**	**59.5**
Total Average:	**77.8**	**82.3**	**37.1**	**82.3**	**51.5**	**58.0**

Table VI-14. Comparison of Ada-boost and S-boost for Naive Bayes.

NB		Ada-boost Vs S-boost				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
AbaloneAll3(4177)	57.8	57.6	84.7	57.2	3.4	98.1
BCancerW(683)	96.3	96.2	67.2	96.2	NaN	NaN
GenNor(300)	85.3	85.2	71.9	85.3	72.2	59.7
GenUni(300)	79.9	79.7	75.0	80.7	70.0	35.9
sub Average:	**79.8**	**79.7**	**74.7**	**79.8**	**48.5**	**64.6**
Balance-scale(625)	91.6	92.0	38.2	92.0	NaN	NaN
Chess(3196)	66.7	94.7	0.0	94.7	NaN	NaN
Cmc(1473)	47.1	48.2	8.5	47.8	25.7	74.3
Crx1(653)	80.2	83.3	0.4	83.3	NaN	NaN
DiabetesPima(768)	75.7	76.6	19.2	77.0	68.3	34.4
Ecoli(332)	63.2	82.8	0.0	82.8	NaN	NaN
Haberman(306)	74.6	75.1	39.2	75.1	57.5	50.5
Hcleve(296)	83.0	83.2	54.6	82.7	36.0	74.7
Iris(150)	95.2	95.9	48.8	95.2	47.4	72.9
Ionosphere(351)	80.6	80.7	50.0	80.6	58.7	81.3
LiverBupa(345)	54.7	63.8	1.0	60.8	10.9	90.8
SegmentationAll(2310)	72.6	75.4	0.0	75.8	92.2	12.3
Splice1(3190)	84.8	94.0	0.0	93.6	6.3	96.7
Thyroid(7200)	95.3	95.4	11.9	95.9	100.0	0.0
Vehicle(846)	45.4	50.2	0.1	50.3	49.4	50.6
sub Average:	**74.0**	**79.4**	**18.1**	**79.2**	**50.2**	**58.0**
Total Average:	**75.3**	**79.5**	**30.0**	**79.3**	**49.9**	**59.4**

Table VI-15. Comparison of Ada-boost and S-boost for QDA.

QDA		Ada-boost Vs S-Boosting				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
Cmc(1473)	49.4	49.3	61.0	49.7	75.6	32.2
Crx1(653)	80.6	79.2	75.8	82.5	94.8	5.3
Ecoli(332)	85.5	84.9	79.5	85.9	70.7	38.2
GenNor(300)	85.2	84.9	88.8	84.9	57.4	55.9
GenUni(300)	79.7	79.6	77.5	78.7	40.8	71.5
Hcleve(296)	77.7	75.0	85.3	76.7	68.6	38.1
Iris(150)	97.5	97.5	70.0	97.7	81.6	65.6
LiverBupa(345)	69.1	69.1	56.8	69.5	64.4	46.5
Vehicle(846)	84.3	82.7	85.5	84.0	78.6	21.6
sub Average:	**78.8**	**78.0**	**75.6**	**78.8**	**70.3**	**41.7**
AbaloneAll3(4177)	61.4	61.6	29.5	61.6	100.0	100.0
Balance-scale(625)	91.6	93.5	3.7	92.4	25.2	87.7
BCancerW(683)	90.3	95.4	0.0	95.6	58.6	52.5
Chess(3196)	88.7	99.0	0.0	99.0	100.0	100.0
DiabetesPima(768)	66.8	72.4	0.0	72.4	100.0	100.0
Haberman(306)	71.8	74.7	3.9	75.2	63.1	50.6
Ionosphere(351)	61.1	85.1	0.0	85.1	100.0	100.0
SegmentationAll(2310)	89.1	93.6	0.0	93.2	12.1	92.8
Thyroid(7200)	91.5	94.6	0.0	94.6	100.0	100.0
sub Average:	**79.1**	**85.5**	**4.1**	**85.5**	**73.2**	**87.1**
Total Average:	**79.0**	**81.8**	**39.8**	**82.1**	**71.7**	**64.4**

Table VI-16. Comparison of Ada-boost and S-boost for LDA.

LDA		Ada-boost Vs S-Boosting				
	bline	Abst	McN, Pv	Sbst	McN, Pv	McN, Pv
dataset(size)	Acc	Acc	(Abst>bl)	Acc	(Abst>Sbst)	(Sbst>Abst)
Dermatology(358)	96.7	96.5	67.4	96.1	63.2	79.5
DiabetesPima(768)	76.3	75.7	77.3	76.7	77.2	24.6
GenNor(300)	85.3	85.1	81.3	85.7	66.8	43.2
Haberman(306)	74.1	73.9	71.7	74.8	66.7	41.1
Iris(150)	98.0	97.2	97.5	97.5	88.8	72.5
sub Average:	**86.1**	**85.7**	**79.0**	**86.2**	**72.5**	**52.2**
AbaloneAll3(4177)	61.8	63.4	0.0	63.6	74.9	25.1
Balance-scale(625)	70.2	86.0	0.0	86.0	100.0	100.0
BCancerW(683)	96.3	96.6	38.9	96.6	85.0	91.3
Chess(3196)	93.4	96.2	0.0	96.2	100.0	100.0
Cmc(1473)	48.9	50.6	1.0	50.0	16.1	89.3
Crx1(653)	84.0	85.2	22.9	86.1	68.3	32.1
Ecoli(332)	88.6	89.7	24.2	89.7	100.0	100.0
GenUni(300)	79.3	79.5	62.5	80.1	66.2	47.2
Hcleve(296)	78.0	82.5	8.8	81.3	39.2	68.7
Ionosphere(351)	86.0	87.9	17.4	86.9	37.9	70.6
LiverBupa(345)	63.4	64.7	23.8	65.5	57.5	47.6
SegmentationAll(2310)	91.7	92.3	5.7	91.3	0.6	99.7
Thyroid(7200)	63.1	85.1	0.0	85.8	97.0	3.0
Vehicle(846)	77.9	78.3	36.0	78.2	49.0	54.9
sub Average:	**77.3**	**81.3**	**17.2**	**81.2**	**63.7**	**66.4**
Total Average:	**79.6**	**82.4**	**33.5**	**82.5**	**66.0**	**62.7**

6.3 Tables of group 3: ASR and Mahalanobis filter

The Tables in this Section show the result of our comparison between the ASR wrapper and the Mahalanobis filter to detect and remove atypical points. The first column shows the dataset name and its size. The next three columns are the average accuracies obtained for the baseline, the ASR, and the Mahalanobis filter (averaged over five times of 10-fold cross validation). Columns 5 and 6 show the McNemar's p-values for the comparison of ASR and the filter method. The average number of points identified as atypical is given in column seven (*nAtp*). The average number of outliers removed by the filter is in the

eighth column (*nOut*). The average number of atypicals and the number of outliers are in the training set (9/10 of data for the datasets with size $\leq$ 1000, and 1/2 of data otherwise). Note that *nAtp* and *nOut* are averaged over (the number of iterations times the number of folds) numbers.

It is noticeable that some datasets have problem with Mahalanobis distance in calculating the inverse of covariance matrix. Hence, the average values (last rows of the tables) of baseline and ASR may be different from what Tables of group 1 and 2 show. In particular, for Dermatology dataset the inverse of covariance matrix could not be calculated. For this dataset, the baseline of NB, QB, and QDA could not be calculated either. Splice1 dataset showed the same problem with LDA and QDA. Also the blineAcc of SVM in Table VI-18 is a little different from those in Table VI-2; this is because of a slight change in the range of C and σ parameters. To make the descriptions of the column headers easily accessible, the descriptions for the Tables of group 3 are given in Table VI-17.

Table VI-17. The descriptions of column headers for tables of group 3.

No	Column header	Description
1	**dataset(size)**	The name of dataset and its size
2	**bline Acc**	The average accuracy of the baseline (no removal), (%)
3	**ASR Acc**	The average accuracy of the ASR (with atypical removal), (%)
4	**Filt Acc**	The average accuracy of (Filt), the classifier trained after removing the outliers detected by Mahal. filter, (%)
5	**Pv% ASR>Filt**	McNemar's p-value, if ASR is better than Filt classifier, (%)
6	**Pv% Filt>ASR**	McNemar's p-value, if Filt classifier is better than ASR, (%)
7	**ASR nAtp**	The average number of atypical points in the training set (ASR)
8	**Filt nOut**	The average number of outliers in the training set (Mahal. filter)

Table VI-18. Comparing ASR and Mahalanobis filter for SVM.

SVM	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	63.9	64.4	65.3	97.8	2.2	594.0	99.0
Balance-scale(625)	97.4	97.7	97.4	48.6	59.9	1.6	3.6
BCancerW(683)	97.0	97.0	96.6	38.7	73.7	7.2	36.4
Chess(3196)	99.0	99.0	96.5	0.0	100.0	3.0	152.5
Cmc(1473)	53.4	51.5	53.3	99.3	0.7	304.0	29.0
Crx1(653)	86.2	85.7	85.7	54.6	47.8	47.9	83.0
DiabetesPima(768)	77.3	76.9	77.1	50.8	51.0	150.0	42.2
Ecoli(332)	88.3	87.6	88.8	75.3	33.9	25.4	13.8
GenNor(300)	85.7	85.6	85.7	61.8	48.1	35.7	0.0
GenUni(300)	79.1	79.5	79.7	51.5	55.5	48.7	0.0
Haberman(306)	74.5	74.0	74.7	65.5	43.0	101.5	7.2
Hcleve(296)	84.0	84.6	83.3	46.8	62.7	31.2	20.2
Iris(150)	95.2	95.1	95.9	76.6	45.5	2.0	0.6
Ionosphere(351)	94.1	93.6	93.6	53.2	58.8	2.2	56.8
LiverBupa(345)	69.2	69.8	68.2	34.1	68.2	51.9	18.7
SegmentationAll(2310)	95.6	95.6	93.2	0.0	100.0	0.0	102.5
Splice1(3190)	96.0	96.0	93.8	0.0	100.0	14.0	147.0
Thyroid(7200)	97.5	97.7	96.3	0.0	100.0	20.0	477.5
Vehicle(846)	83.9	83.4	82.8	37.7	64.0	35.2	20.5
Averages:	**85.1**	**85.0**	**84.6**	**47.0**	**58.7**	**77.7**	**69.0**

Table VI-19. Comparing ASR and Mahalanobis filter for KNN.

KNN	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	63.1	62.4	63.0	87.2	12.8	713.5	99.0
Balance-scale(625)	90.3	87.9	90.5	92.1	10.3	50.6	3.6
BCancerW(683)	97.0	96.6	97.1	77.0	35.4	14.6	36.6
Chess(3196)	90.8	90.8	90.5	26.0	74.0	0.0	152.5
Cmc(1473)	49.9	47.0	50.0	99.7	0.3	341.0	29.0
Crx1(653)	79.1	78.4	79.3	66.5	34.4	27.7	79.4
DiabetesPima(768)	75.1	74.8	74.5	43.3	57.8	171.2	42.2
Ecoli(332)	87.5	87.4	87.7	65.0	47.7	28.9	13.9
GenNor(300)	84.9	85.5	85.0	50.8	60.8	37.9	0.0
GenUni(300)	77.5	76.5	77.3	59.4	43.7	56.8	0.0
Haberman(306)	75.8	74.9	76.4	72.4	37.2	96.9	7.3
Hcleve(296)	82.0	83.1	82.6	50.3	59.8	40.7	20.5
Iris(150)	95.3	94.9	95.1	58.0	62.4	3.5	0.6
Ionosphere(351)	84.5	83.5	86.4	90.2	14.0	16.4	56.9
LiverBupa(345)	63.4	63.9	63.5	49.2	52.6	102.0	18.9
SegmentationAll(2310)	92.8	95.2	92.0	0.0	100.0	2.0	102.5
Splice1(3190)	81.4	78.2	77.6	19.4	80.6	2.5	147.0
Thyroid(7200)	94.7	94.7	92.4	0.0	100.0	0.0	477.5
Vehicle(846)	72.5	69.9	72.0	85.1	14.9	140.0	20.7
Averages:	**80.9**	**80.3**	**80.7**	**57.5**	**47.3**	**97.2**	**68.8**

Table VI-20. Comparing ASR and Mahalanobis filter for CART.

CART	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	62.1	62.9	62.9	55.0	45.0	646.0	105.5
Balance-scale(625)	79.5	80.8	79.6	27.9	73.7	86.1	3.6
BCancerW(683)	94.5	94.9	94.0	31.3	76.6	20.0	36.6
Chess(3196)	98.2	98.2	98.3	80.4	28.4	14.0	144.0
Cmc(1473)	54.1	54.1	52.4	4.4	95.6	324.0	29.0
Crx1(653)	86.5	86.2	86.6	58.8	46.1	75.0	37.0
DiabetesPima(768)	74.9	74.4	74.5	51.7	49.9	147.9	42.2
Ecoli(332)	79.9	81.8	80.9	39.9	68.1	36.0	13.9
GenNor(300)	82.8	81.9	83.1	67.3	44.0	34.6	0.0
GenUni(300)	80.7	78.9	80.6	70.8	35.7	40.2	0.0
Haberman(306)	72.3	71.7	72.3	64.5	44.7	75.4	7.3
Hcleve(296)	77.5	78.9	76.8	38.2	67.4	40.3	4.3
Iris(150)	93.6	94.9	93.9	44.1	75.9	4.0	0.6
Ionosphere(351)	89.4	89.2	90.1	71.3	37.1	21.7	57.0
LiverBupa(345)	67.5	66.9	68.0	66.3	34.8	77.6	18.7
SegmentationAll(2310)	93.4	94.8	91.9	0.0	100.0	34.5	102.5
Splice1(3190)	94.4	94.1	94.4	99.4	2.1	80.5	28.5
Thyroid(7200)	99.4	99.4	99.2	0.0	100.0	15.0	477.5
Vehicle(846)	68.7	69.0	65.6	8.3	91.7	147.9	20.7
Averages:	**81.6**	**81.7**	**81.3**	**46.3**	**58.8**	**101.1**	**59.4**

Table VI-21. Comparing ASR and Mahalanobis filter for QB.

QB	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	61.4	61.9	62.3	81.0	19.0	831.0	105.5
Balance-scale(625)	91.9	91.9	57.3	0.0	100.0	0.0	3.6
BCancerW(683)	95.2	95.1	92.4	2.7	98.4	13.0	36.4
Chess(3196)	66.7	66.7	66.8	76.5	33.3	0.0	144.0
Cmc(1473)	52.2	52.0	33.8	0.0	100.0	7.5	29.0
Crx1(653)	80.6	80.6	81.8	73.5	28.9	0.0	37.3
DiabetesPima(768)	74.1	73.8	73.9	51.5	50.7	174.3	42.2
Ecoli(332)	58.0	58.0	67.6	98.9	1.1	0.0	13.8
GenNor(300)	85.3	86.5	85.3	32.9	75.6	35.9	0.0
GenUni(300)	79.3	79.3	79.5	50.9	57.2	51.8	0.0
Haberman(306)	75.2	74.6	75.4	69.0	38.9	98.1	7.2
Hcleve(296)	82.6	80.6	81.9	73.9	33.9	17.5	4.4
Iris(150)	97.2	97.5	97.2	57.1	73.7	1.3	0.6
Ionosphere(351)	90.6	91.2	83.2	0.2	99.9	6.1	56.8
LiverBupa(345)	60.7	59.6	67.1	95.9	4.1	2.1	18.7
SegmentationAll(2310)	62.5	62.5	15.7	0.0	100.0	0.0	102.5
Splice1(3190)	84.8	84.7	92.8	100.0	0.0	1.0	28.5
Vehicle(846)	85.3	84.6	84.3	42.6	59.6	73.9	20.5
Averages:	**76.9**	**76.7**	**72.1**	**50.4**	**54.1**	**73.0**	**36.2**

Table VI-22. Comparing ASR and Mahalanobis filter for NB.

NB	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	57.8	57.5	57.8	88.1	11.9	459.0	105.5
Balance-scale(625)	91.6	91.5	91.4	61.3	69.5	7.1	3.6
BCancerW(683)	96.3	96.3	93.6	1.8	99.0	3.9	36.4
Chess(3196)	66.7	66.7	66.8	76.5	33.3	0.0	144.0
Cmc(1473)	47.1	46.7	33.5	0.0	100.0	19.5	29.0
Crx1(653)	80.2	80.2	81.3	71.3	31.6	0.0	37.3
DiabetesPima(768)	75.7	75.7	74.4	25.0	75.2	3.5	42.2
Ecoli(332)	63.2	63.2	82.8	100.0	0.0	0.0	13.8
GenNor(300)	85.3	84.9	85.5	70.2	46.7	19.2	0.0
GenUni(300)	79.9	80.7	79.8	40.0	69.8	54.1	0.0
Haberman(306)	74.6	74.3	74.9	66.9	43.2	98.0	7.2
Hcleve(296)	83.0	82.8	81.4	42.3	66.8	8.2	4.4
Iris(150)	95.2	95.2	95.3	69.8	66.1	1.8	0.6
Ionosphere(351)	80.6	81.0	83.0	86.7	18.9	11.5	56.8
LiverBupa(345)	54.7	54.2	58.4	86.3	14.0	12.4	18.7
SegmentationAll(2310)	72.6	72.5	65.8	0.0	100.0	19.5	102.5
Splice1(3190)	84.8	84.7	92.8	100.0	0.0	1.0	28.5
Vehicle(846)	45.4	45.7	48.7	93.7	6.4	38.5	20.5
Averages:	**74.1**	**74.1**	**74.9**	**60.0**	**47.4**	**42.1**	**36.2**

Table VI-23. Comparing ASR and Mahalanobis filter for QDA.

QDA	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	61.4	61.4	59.8	0.4	99.6	0.0	99.0
Balance-scale(625)	91.6	91.8	92.0	62.1	49.3	24.4	3.6
BCancerW(683)	90.3	94.7	83.3	0.0	100.0	227.2	36.4
Chess(3196)	88.7	88.7	74.9	0.0	100.0	0.0	152.5
Cmc(1473)	49.4	49.4	47.6	2.5	97.5	22.5	29.0
Crx1(653)	80.6	80.7	77.8	16.3	83.9	95.5	83.0
DiabetesPima(768)	66.8	66.8	66.0	32.3	69.4	0.0	42.2
Ecoli(332)	85.5	85.9	85.9	56.9	55.2	32.4	13.8
GenNor(300)	85.2	85.7	85.3	44.0	65.2	35.6	0.0
GenUni(300)	79.7	80.3	79.6	47.4	63.1	38.0	0.0
Haberman(306)	71.8	73.1	73.5	59.2	45.3	35.8	7.2
Hcleve(296)	77.7	76.6	76.2	49.9	56.2	45.1	20.2
Iris(150)	97.5	97.3	97.3	62.2	67.8	0.7	0.6
Ionosphere(351)	61.1	61.1	55.4	2.0	98.5	0.0	56.8
LiverBupa(345)	69.1	68.6	68.2	49.0	52.4	5.8	18.7
SegmentationAll(2310)	89.1	89.2	89.2	50.0	50.0	2.0	102.5
Thyroid(7200)	91.5	91.5	86.4	0.0	100.0	0.0	477.5
Vehicle(846)	84.3	83.9	84.0	55.8	46.4	66.4	20.5
Averages:	**79.0**	**79.3**	**76.8**	**32.8**	**72.2**	**35.1**	**64.6**

Table VI-24. Comparing ASR and Mahalanobis filter for LDA.

LDA	bline	ASR	Filt	Pv %	Pv %	ASR	Filt
Dataset(size)	Acc	Acc	Acc	ASR>Filt	Filt>ASR	nAtp	nOut
AbaloneAll3(4177)	61.8	61.5	62.9	99.1	0.9	380.0	99.0
Balance-scale(625)	70.2	70.2	68.8	28.3	76.1	0.0	3.6
BCancerW(683)	96.3	96.2	96.0	45.5	73.1	0.9	36.4
Chess(3196)	93.4	93.4	92.3	0.2	99.8	0.0	152.5
Cmc(1473)	48.9	48.8	49.7	94.8	8.4	8.0	29.0
Crx1(653)	84.0	86.3	74.8	0.0	100.0	102.5	83.0
DiabetesPima(768)	76.3	75.8	75.6	49.0	53.6	100.0	42.2
Ecoli(332)	88.6	88.6	88.6	54.7	57.9	0.0	13.8
GenNor(300)	85.3	86.4	85.6	41.3	70.6	36.8	0.0
GenUni(300)	79.3	80.2	79.4	36.8	68.9	46.6	0.0
Haberman(306)	74.1	74.1	74.3	55.4	53.2	121.0	7.2
Hcleve(296)	78.0	80.6	80.3	49.6	57.5	52.7	20.2
Iris(150)	98.0	98.0	98.0	70.3	71.9	0.4	0.6
Ionosphere(351)	86.0	86.4	85.9	46.2	63.7	29.9	56.8
LiverBupa(345)	63.4	62.6	66.3	78.2	21.8	31.1	18.7
SegmentationAll(2310)	91.7	90.8	90.5	23.8	82.9	52.5	102.5
Thyroid(7200)	63.1	69.2	86.1	100.0	0.0	705.5	477.5
Vehicle(846)	77.9	77.0	78.8	81.7	18.3	103.5	20.5
Averages:	**78.7**	**79.2**	**79.7**	**53.0**	**54.4**	**98.4**	**64.6**

VII Discussion of the results

In this Section, the obtained results from the experiments are discussed. We divide the discussion into different parts like the way the experimental results were presented. The first part, the discussion on the results of ASR and the baseline is presented and followed by the discussion of the reject option results. In the next part, the results of experiment on Ada-boost and S-boost are discussed. The last part is to discuss the results obtained from the comparison between ASR and the Mahalanobis filter.

7.1 ASR and baseline

In Figure VII-1, the overall average accuracies of baseline and ASR for all datasets and all classifiers (Table VI-2 to Table VI-8) are shown. Classifiers in Figure VII-1 are ordered according to their performance measures.

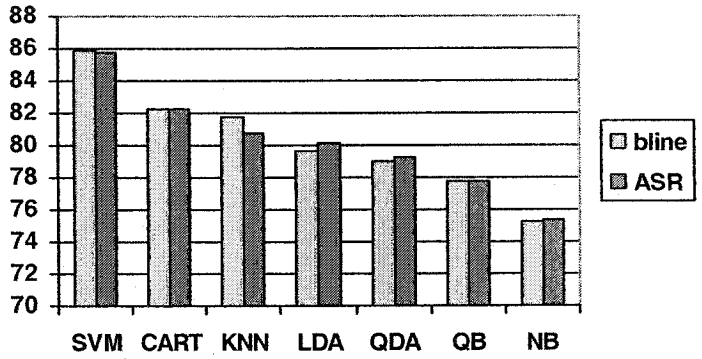

Figure VII-1. Average accuracies of baseline and ASR over all datasets.

The accuracies of ASR match very well with that of the baseline for all classifiers except for KNN. This indicates that the ASR scheme has been successful in keeping the overall performance level of these classifiers (averaged over all applied datasets). Keeping the performance level almost the same as that of the baseline was the highest achievement reported in the previous studies as well. In certain instances, however, ASR has performed better than the baseline. Table VII-1 shows some of the datasets for which ASR has been successful in particular (based on the results from Table VI-2 to Table VI-8). In particular, the Thyroid data shows significant improvement under 4 classifiers (out of 7). This further reduces the chance that such a behavior may be random and reinforces that there may be a structure in the data that the removal of atypicals has changed it towards the better performance. Unfortunately, all these datasets have more than 3 attributes; so, plotting them is not possible. For instance, Thyroid data has 21 attributes and 3 classes.

Table VII-1. Some of the datasets for which ASR has performed better than the baseline.

Classifier	dataset	baseline accuracy	ASR accuracy	Pv % (ASR>baseline)
SVM	Ecoli(332)	87.9	89.1	36.9
SVM	Thyroid(7200)	97.3	97.5	12.1
KNN	SegmentationAll(2310)	92.8	95.2	0.0
CART	Ecoli(332)	79.9	82.6	22.9
QB	Thyroid(7200)	95.5	96.8	0.0
NB	Thyroid(7200)	95.3	96.8	0.0
QDA	BCancerW(683)	90.3	94.7	0.2
LDA	Crx1(653)	84.0	86.1	12.4
LDA	Thyroid(7200)	63.1	69.2	0.0

Note that for the majority of these datasets, the improvement in the test accuracy is statistically significant (see the last column of Table VII-1). Also, for large datasets like Thyroid, even a small change in accuracy can save many points from misclassification. However, we are not presently interested in focusing on such results because what really matters is the overall performance of ASR (average over all datasets).

The p-values from Table VI-2 to Table VI-8 are shown in Figure VII-2. The p-values are far from showing something statistically significant. In other words, considering the results of this research, the null hypothesis that two classifiers (baseline and ASR) are the same, cannot be rejected.

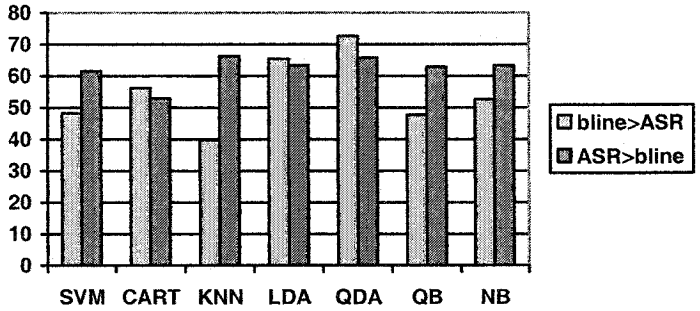

Figure VII-2. Average p-values over all datasets.

As it was mentioned in Subsection 2.3 there have been assertions of performance increase, in the literature, as a result of atypical removal. The above results shows that it was not the case for average performance over all datasets under the constraints of our wrapper method (a linear search as explained in Subsection 4.2.1). This is also in line with what we discussed about those assertions (filter efficiency is not always the same as generalization performance). Therefore, the best result achieved so far by removing outliers, noise, or atypicals is to keep the overall (average) performance on the intact test set almost the same as that of the baseline with no removal. The advantage of such a process is the identification of atypical points that was discussed in Subsections 1.1 and 1.2.

For KNN, however, the result is somewhat different compared to the others: the average of all baseline accuracies for KNN = 81.7 and that of ASR = 80.7; p-values also show a slight trend in favor of the baseline which is far from being statistically significant: Pv(bline>ASR) = 40% which is far from α–level of significance = 5% or even 10%. It

may be because KNN is a local learner and outliers may have less impact on the decision of local learners (they also gain less from removing outliers) and since the wrapper approach makes its evaluation based on the performance of the classifier itself, it cannot make a proper decision about potential outliers.

It is interesting to look at the average of *ntrains* for KNN, the last column in Table VI-3. Recall that *ntrains* is the number of times training was repeated to reach the stopping criteria e.g., zero training error in the getMisclfd function. The average of *ntrains* for KNN is 6.8 which is the 2nd highest among all classifiers (after *ntrains* of QDA: 7.3) and far from the average of the other 5 classifiers: (3.5 + 2.5 + 4.3 + 4.5 + 5.1) / 5 = 3.98. Hence, QDA and KNN have had a hard time to reach their zero training error state. The average of *ntrains* for all 7 classifiers and all datasets is about 5.0. Also, the average of Coverage values for KNN (86.2) is the minimum of that of all classifiers. Recall that Coverage is the proportion of points left in the training set after removing atypicals. That is, KNN has chosen more points, as atypical, than any other classifier.

Note that *nAtps* for KNN (91.9) is the maximum among all other classifiers although LDA seemingly shows a slightly higher number (93.6). This is because LDA (and QDA) could not complete their calculation for the Splice dataset. If we exclude the values of the Splice data, for a fair comparison with LDA, the KNN results would be: average Coverage = (86.2 * 20 – 99.8) / 19 = 85.5%; and *nAtps* = (91.9 * 20 – 2.5) / 19 = 96.6. If we perform the same calculation (excluding Splice dataset) for other classifiers, we obtain average Coverage values for 19 datasets: 88.69, 86.98, 91.16, 90.32, and 92.2% for SVM,

CART, QB, NB, and QDA respectively. The averages for 19 datasets for *nAtps* are 79.8, 92.66, 84, 76.1, and 35.3 for SVM, CART, QB, NB, and QDA respectively. This shows that KNN has taken the most points as atypicals followed by LDA. These facts support the idea that the drop of accuracy for KNN is due to its learning strategy (instance-based learning). Nevertheless, using ASR even KNN has not fallen far behind its baseline (only 1 percent in accuracy which is important practically but not statistically significant). As we will see in Subsection 7.2 this is fixed by using the check option for KNN.

It is noticeable that the drop of KNN was caused mainly by 3 datasets: Balance-scale, Splice1, and Vehicle; and there is one dataset (SegmentationAll) for which ASR has performed better than the baseline. To study what kind of points the ASR algorithm removes, some detailed work, done on this case, is presented here. The baseline accuracy (with no removal) for KNN was 92.8% and the accuracy of KNN with ASR was 95.2%, and Pv(ASR >bline) = 0%. ASR performs 1 iteration of 2-fold CV on SegmentationAll data (its size is 2310). This 2-fold CV generates two curves from the output of getAtps function (each for 1 fold of the outer CV loop) which are shown in Figure VII-3. The lower curve is for the first fold and no atypical was found in the training set (half of the data); the upper curve corresponds to the second fold in which 4 points are introduced as atypicals.

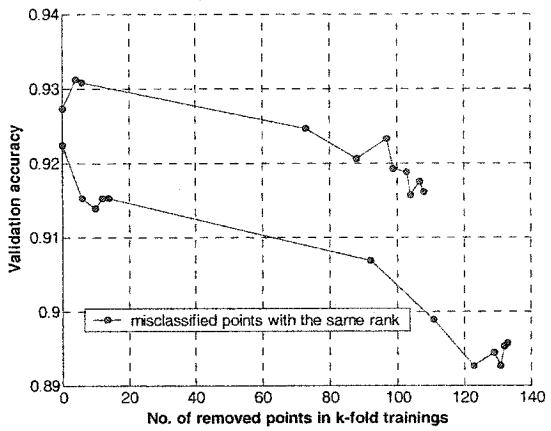

Figure VII-3. Results of getAtps function on SegmentationAll data using KNN.

It is noticeable that the validation accuracy (within getAtps function) is slightly more than 93% but the test accuracy averaged over two folds is increased to 95.2 by removing those 4 points in ASR. Interestingly, the Mahalanobis filter has taken out an average of 102.5 points (205 points for the whole dataset) which has decreased KNN accuracy from 92.8 to 92%. Further investigation showed that the 4 atypical points were *not* among the 205 points found by the Mahalanobis filter. The 4 atypical points were indeed the duplications of two points: two points from class 2 (Cement) and two points from class 7 (Window). We needed a closer look at where the atypical points were located in the problem space to study why they have been chosen as atypical by ASR and why they have such effect on the classification performance.

A 3-D plot of the raw RGB values of all class 2 and class 7 members is shown in Figure VII-4. RGB values are, more likely, the most discriminative 3 features among the 18 in

the image file. The duplicate atypical points are shown as a large circle (class 2) and a large square (class 7). Figure VII-4 shows that class 2 atypical point is in a location with not many members of its own class; it is positioned outside of the two islands formed by the members of its class (one large island to its up and left side and a small island to its down and right side). This idea is supported by Figure VII-7 (left) and Figure VII-8 as well. Class 7 atypical point is at the end of the tail of its class distribution. Hence, considering them as atypical (by ASR) is justified by the position of these points in the 3-D plot.

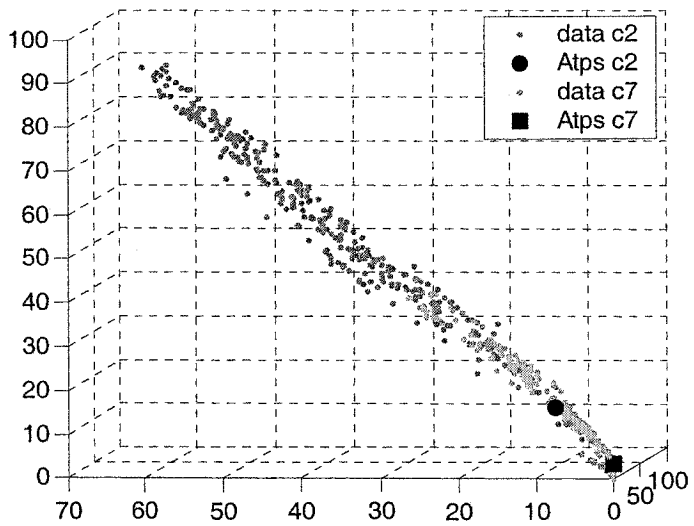

Figure VII-4. Raw RGB values for members of class 2 and 7 in SegmentationAll.

A 3-D plot of the RGB values of all class 2, class 7, and other 5 classes' members is shown in Figure VII-5. The mass of points from other 5 classes surrounding two atypical points suggests that there can be plenty of points that two atypical points may have influence in their classification.

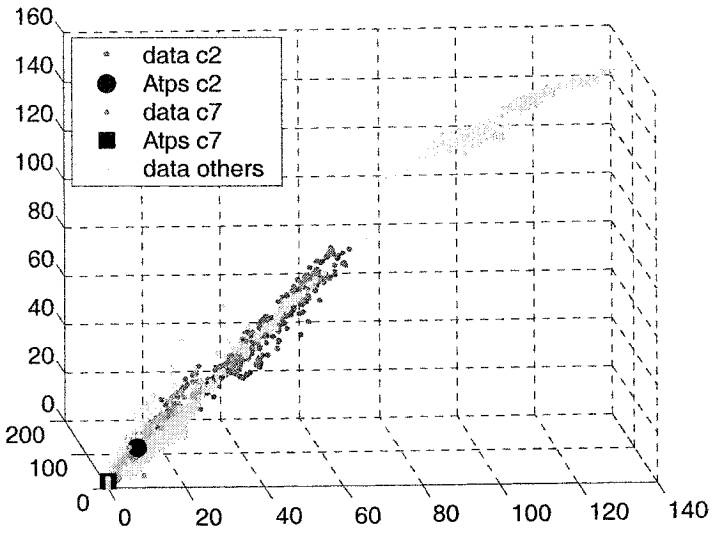

Figure VII-5. Raw RGB values for members of class 2, 7, and other classes in SegmentationAll.

Since SegmentationAll data has 18 attributes (and some of them revealed to be redundant), PCA was applied to the dataset to plot them. Figure VII-6 shows the histogram of the first principal component of all classes (all the dataset). The two atypical points identified by ASR are plotted as white dots at the first PC = 1.617 for class 2 and the first PC = 2.617 for class 7. This plot shows how many other points (from other classes) are close to the atypicals and, hence, may be mistaken by the presence of the atypical points. SegmentationAll has the same number of examples (330) for all 7 classes. The fact that both these points are duplicated can have a significant effect in misleading the classification of the close by points especially when K = 3 (in KNN) is used. As Figure VII-6 shows, both atypical cases seem to have enough close by points from classes other than their own to influence their classification. This supports what we saw in Figure VII-5.

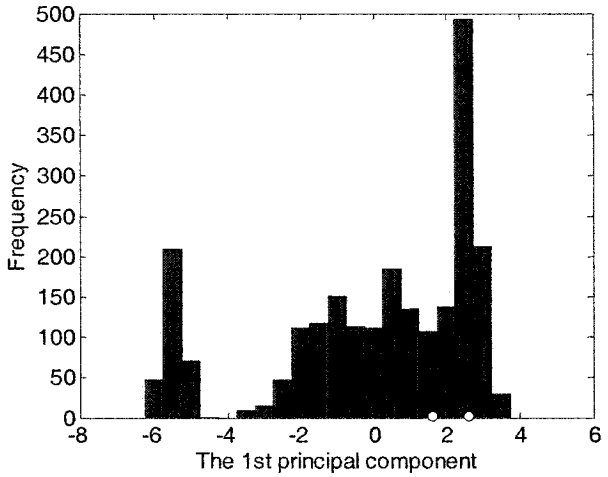

Figure VII-6. The 1st PC in SegmentationAll data.

Figure VII-7 shows the first PCs only on the classes of atypical points separately. The two atypical points are shown as white markers at the first PC = 1.617 for class 2 and the first PC = 2.617 for class 7.

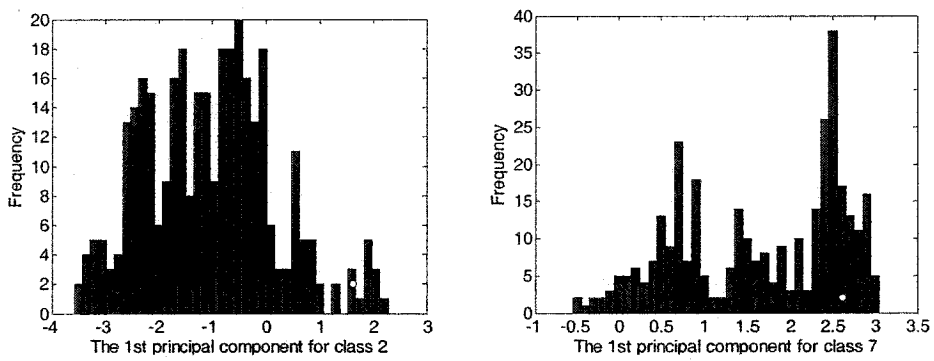

Figure VII-7. The 1st PC for class 2 (left) and class 7 (right) in SegmentationAll data.

Figure VII-7 shows that whether the class of atypical points was underrepresented close to the position of those two points. While this can be the case for class 2, the class 7 atypical shows the opposite. This may suggest that perhaps being an outlier is less important than being duplicate points that may be influential when they are close to many points of other classes. In this case, even if the class of the atypical point is not underrepresented, it can have a misleading effect on other classes.

Figure VII-8 shows the plot of the first PC against the second PC for all 7 classes. The plot was zoomed in where the atypical point of class 2 can be seen clearly (shown as a bold circle in the center of the plot). As Figure VII-8 shows, some members of class 1 and 7 may be misclassified by the two atypical points of class 2. We know that there is no guarantee that the situation would be the same in the actual case (18 dimensional problem space) but this may be a good approximation.

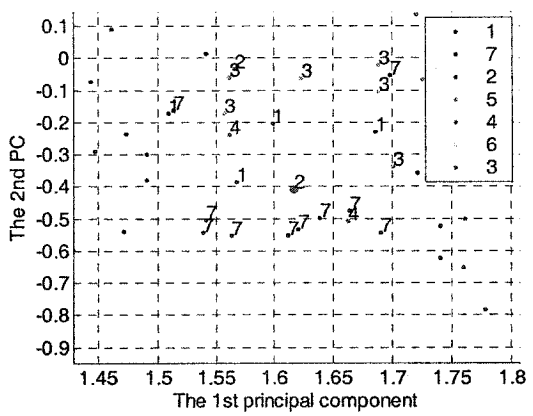

Figure VII-8. The 1st and 2nd PCs zoomed in where class 2 atypical is located.

Figure VII-9 shows the plot of the first PC against the second PC for all 7 classes, zoomed in where the atypical point of class 7 can be seen clearly (shown as a bold circle in the center of the plot). As Figure VII-9 shows, some members of class 1 and 3 may be misclassified by the presence of two atypical points of class 7.

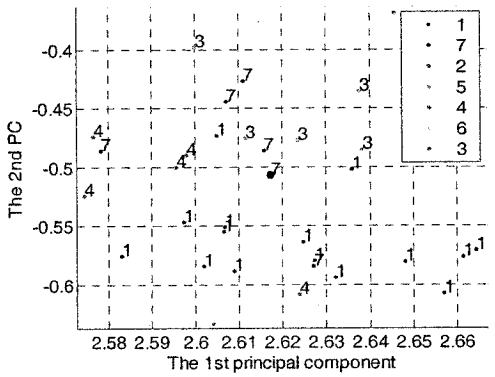

Figure VII-9. The 1st and 2nd PCs zoomed in where class 7 atypical is located.

Overall, the case of SegmentationAll data shows an example of the kind of points ASR considers as atypical. This example shows that an atypical point can be at the tail of the distribution of its class. If there are enough examples of other classes and enough examples of its own class close to the point, then the point is an overlapping sample and may be influential (the case of class 7). In other words, if its negative effect on other classes overcomes the positive effect on its own class, then it is negatively influential[3]. If the area close to the point is underrepresented by the members of its own class, then it can

[3] The misleading effect is not limited to the case when some members of other classes (in the neighborhood) are classified as class 7, but dropping their true class from the majority is enough for misclassification.

be an outlier; but it can be influential if there are enough examples of other classes close to the point (the case of class 2). For a classifier like KNN, duplicate points have obviously more chance of being influential.

The ASR wrapper approach may not be a preferred method of atypical detection for instance-based learners, like KNN, as the overall accuracy dropped by 1%. But KNN provides an easier way of interpreting the plots and that is why we focused more on this case. A survey of different instance reduction methods (somewhat related to atypical detection) for instance-based learners is given in (Wilson and Martinez, 2000b).

Considering the average of all coverage values for SVM (89.1), the ASR accuracy is achieved when removing almost 10% of the points in the training sets. In some datasets like AbaloneAll3 and Cmc that are considered to be the most problematic ones, the coverage value is lower (69.3 and 61.2) indicating that ASR has needed to take more points to reach its highest performance. In the case of AbaloneAll3, the removal is done till 97 percent of all misclassified points and in Cmc it goes all the way to 100 percent of misclassified points.

In the Chess data the ratio of *nAtp/nMiscl* is also 100 percent but this is when there are only 2 points to remove (all examples carry useful information). Considering the size of the Chess data (3196), it is not disappointing. Chess data is deterministic, in the sense that there is no uncertainty on the class memberships. For classifiers other than SVM, no points were taken from Chess data; except for CART that has taken 10.5 points (decimal

values in the number of points are due to averaging). The removal of zero points has happened in the other 5 classifiers while their baseline accuracy is not very high (KNN, 90.8; QB, 66.7; NB, 66.7; QDA, 88.7; and LDA, 93.4).

The average values of ratio of *nAtp/nMiscl* (shown in Figure VII-10) are: 80.5, 91.3, 56.1, 78.9, 57.9, 42.2, and 31.9% for SVM, CART, KNN, LDA, QDA, QB, and NB, respectively. This clearly shows that getAtps function is not monotonic, unlike getMisclfd or error-reject curves. If more misclassified points were added to the set of atypicals, we would possibly see a drop in the test accuracy of ASR.

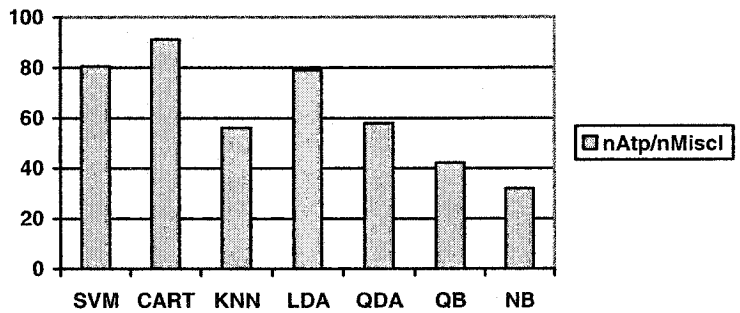

Figure VII-10. Average ratios of nAtp/nMiscl over all datasets.

The point that getAtps function shows a non-monotonic behavior (unlike error-reject and CP curves) guarantees a cut-off point in removal process before it starts damaging the validation performance on the training set. Figure VII-11 illustrates this point better on

the DiabetesPima dataset, using KNN classifier. Every curve on Figure VII-11 is the result of getAtps function on the training set of one fold of a 10-fold CV (i.e. 90% of the data in the outer CV loop). On average, ASR has identified 171 points as atypical (84% of all misclassified points). The DiabetesPima dataset is a problematic dataset with a baseline accuracy of 75.1%; the average accuracy for ASR was 74.8% for KNN.

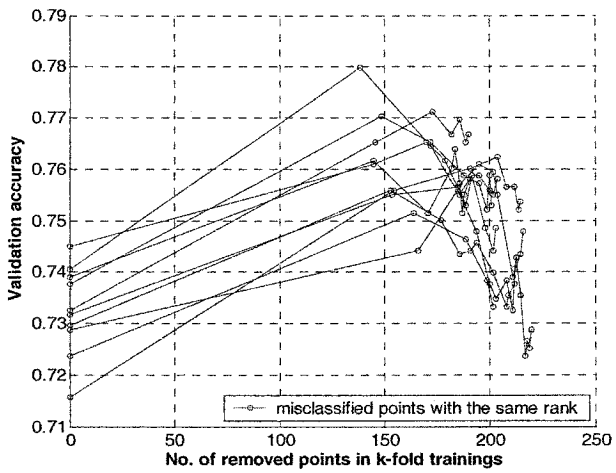

Figure VII-11. Non-monotonic behavior of getAtps function on DiabetesPima dataset, using KNN.

Note that removing all the points in the misclassified set from the training set is not necessarily a good idea as it may degrade the classifier. Also, there may be cases that ASR does not need to remove any point at all. Figure VII-12 shows these two concepts on the Chess data and the Crx1 (credit card approval) data using KNN classifier. A 2-fold CV was applied on the Chess data and a 10-fold CV for Crx1.

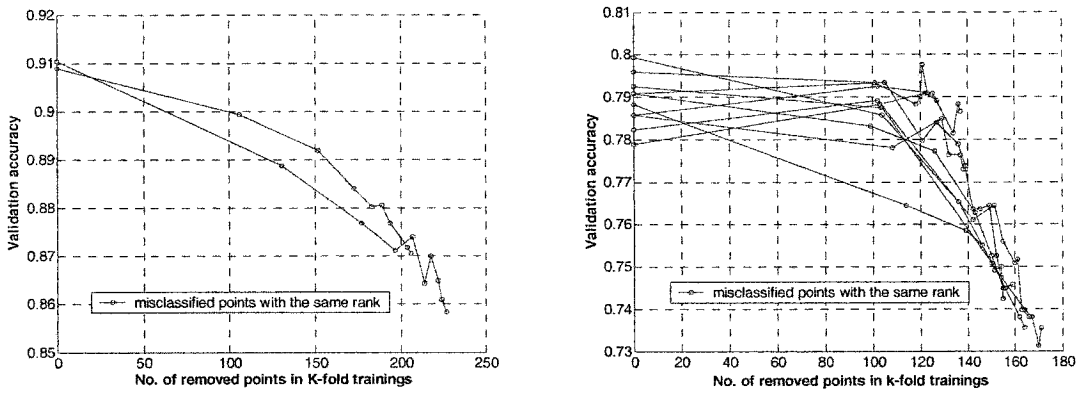

Figure VII-12. Results of getAtps function on the Chess and Crx1 datasets (left to right), using KNN.

The accuracy of the baseline KNN on the Chess data is 90.8% which is the same as the ASR (no removal). The Chess data has 3198 examples. Removing all the points in the misclassified set will cause a 5% drop in the validation accuracy in the Chess data and almost 6% in the Crx1 data. This further shows that getAtps function is not monotonically increasing. The slight possible differences in the results presented in the tables and in these figures are because for figures only 1 iteration was done while for tables 5 iterations were performed for most datasets.

Another interesting point is that the average number of atypical points (shown in Figure VII-13) is not very different for 6 classifiers (excluding QDA). This is while the average number of misclassified points varies significantly (more than twice as much). QDA has behaved very differently: among all classifiers, it has the maximum *ntrains* = 7.3 and the minimum of *nAtps* and *nMiscl* (35.3 and 61.0 respectively). This unexpected behavior of QDA needs to be studied in the future. Generally speaking, the weaker a classifier, the more number of misclassified points it has picked up (see Figure VII-1 and Figure

VII-13), yet, ASR has not been misled by the large numbers of misclassified points (as in KNN, QB, and NB). It has chosen a portion that seems to be about reasonable for all classifiers.

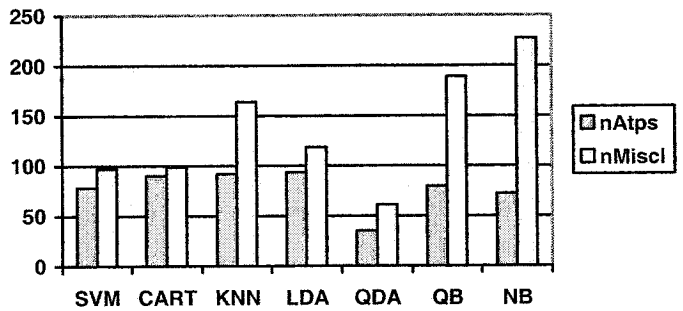

Figure VII-13. Average numbers of atypical and misclassified points over all datasets.

At the end of a K-fold CV, ASR gives the list of atypical points obtained from the removed points in all folds. This list can be sorted according to the frequency of these points in all K-folds. Note that ASR can make this ordered list of atypical points that covers the whole of the dataset (both training and test parts) without sacrificing the legitimacy of test (evaluation) results. That is why CV is necessary in the ASR algorithm.

7.2 Check option results

ASR was able to perform as well as the baseline classifier, except for the case of KNN that showed an average drop of 1% for all 20 datasets. ASR with the check option, presented in Table IV-2, was introduced as a possible way out. The results obtained, using the check option with the KNN classifier, are given in Table VII-2 and show that the check option has solved performance degradation of KNN. The lower average accuracies of ASR compared to that of baseline is not seen anymore (baseline total average is 81.7%, see Table VI-3; and that of ASR is now 81.6%).

Another feature of the check option is that it is guaranteed to produce the number of atypical points smaller than or equal to that of the regular ASR scheme. The smaller case happens when the points prepared for removal, in some folds, become disqualified because their ASR classifier cannot pass the check. In other words, the behavior of ASR with the check option (in terms of its performance, p-values, and the number of removed points) is *always* between that of ASR and the baseline classifier. As a result, we did not pay much attention to the small differences between ASR with and without the check option which can be a subject of future studies.

ASR with the check option was also tested on other classifiers and similar results were obtained: fewer atypical points; and performance measures (accuracies) close to the baseline classifier. The calculations of p-values, coverage, s-boost, etc. were not done, however, for the ASR with the check option. For classifiers other than KNN, since

accuracies of the ASR for these classifiers are already close enough to the baseline, we do not need to repeat experiments for the ASR with the check option. So, all the results presented in this research are from regular ASR unless otherwise specified (as in Table VII-2).

Table VII-2. The accuracy of ASR after adding the check option for KNN.

KNN	bline	ASR
dataset(size)	Acc	Acc
AbaloneAll3(4177)	63.1	63.0
Balance-scale(625)	90.3	90.3
BCancerW(683)	97.0	96.7
Chess(3196)	90.8	91.5
Cmc(1473)	49.9	49.6
Crx1(653)	79.1	79.3
Dermatology(358)	95.5	95.9
DiabetesPima(768)	75.1	74.8
Ecoli(332)	87.5	88.0
GenNor(300)	84.9	84.7
GenUni(300)	77.5	77.8
Haberman(306)	75.8	74.5
Hcleve(296)	82.0	82.2
Iris(150)	95.3	94.9
Ionosphere(351)	84.5	85.1
LiverBupa(345)	63.4	62.9
SegmentationAll(2310)	92.8	92.9
Splice1(3190)	81.4	81.4
Thyroid(7200)	94.7	94.7
Vehicle(846)	72.5	71.8
Average:	**81.7**	**81.6**

7.3 Ada-Boost vs. S-boost

The Tables in the second group (Table VI-10 to Table VI-16) focus on the behavior of Ada-boost algorithm on two different training sets (with and without atypicals). As a reminder, when Ada-boost is applied to the Subset of training data without atypicals, we refer to this as S-boost (Sbst in tables); and if the whole training data was used, as Ada-boost (Abst in tables).

Note that the presence of misclassified points is an essential element for the boosting process to increase the performance of the classifier. When the majority of misclassified points were removed from the training set, the boosting process has little chance to make a difference. Hence, S-boost should perform worse than Ada-boost unless the removed misclassified points were not of much use to the classification task. Figure VII-14 shows the average accuracies of the baseline classifier, Ada-boost and S-boost ensembles over all datasets.

The interesting observation here is that although a significant number of misclassified points are not present in the training set of S-boost (see Figure VII-10 and Figure VII-13), it has performed slightly better than Ada-boost in SVM, KNN, and QDA; almost the same in LDA, QB, and NB and slightly worse in CART (see Figure VII-14). The corresponding p-values, in Table VI-10 to Table VI-16 do not contradict this observation. This further shows that the points taken out by ASR did not carry useful information as

far as the performance measure and the classifier in use are concerned. This statement is nothing but our definition of "atypical points," i.e., a sign of success for ASR.

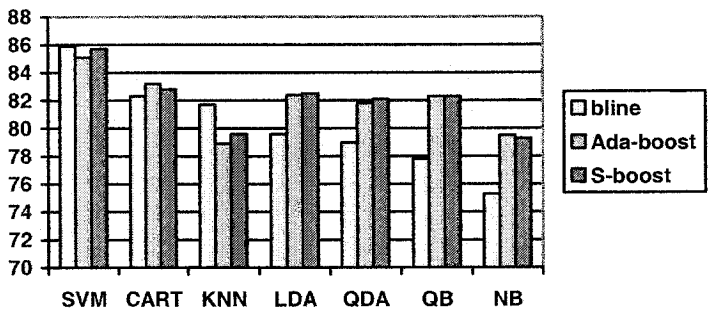

Figure VII-14. Average accuracies of baseline classifier, Ada-boost and S-boost ensembles over all datasets.

For SVM, Ada-boost has hardly any positive effect. Only on one dataset, SegmentationAll, Ada-boost does a better job than the baseline. One explanation for why boosting works is that the combined classifier (the ensemble) increases the margin of classification compared to the base classifiers (Schapire, *et al.*, 1998); and since SVM has already maximized the margin, there is no room for enhancement. But pushing SVM to focus on the misclassified points has had a small cost, while a benefit was expected because choosing a kernel function and SVM parameters were not optimized globally. This cost is less when most of the misclassified points are not present in the training data.

For KNN, overall, S-boost seems to be better than Ada-boost but neither one is better than the baseline. The reason (or part of it) can be that, as explained earlier, KNN is a local learner and is not expected to be good at finding the points that can influence its performance globally. As a result, the struggle of boosting on learning misclassified points better does not lead to a positive outcome. The same argument can explain, maybe partially, why S-boost has performed somewhat better than Ada-boost.

In addition, in the datasets for which Ada-boost was stronger than the baseline (the 2nd sub Average in Table VI-11), S-boost is weaker than Ada-boost. This means that perhaps there has been some useful information for KNN in the points that ASR has taken away from training. Also, the 1st sub Average of S-boost accuracy in Table VI-11 is considerably smaller than the baseline. This means that maybe there have been still problematic points (atypicals) for KNN that were not removed by ASR. In other words, ASR has achieved something in terms of picking some atypicals (S-boost is somewhat better than Ada-boost overall), but it has not done a perfect job.

In the case of CART, both S-boost and Ada-boost have done slightly better than the baseline. However, the same conclusion as in KNN can be drawn. Since the 1st sub Average of S-boost accuracy in Table VI-12 is smaller than that of baseline but larger than that of Ada-boost, then ASR has been successful in removing some atypicals but it has not found them all. Also, since the 2nd sub Average of S-boost accuracy in Table VI-12 is smaller than that of Ada-boost but larger than that of baseline, then ASR has

been successful in not taking too many useful points from within the misclassified points but perhaps some of those points were removed by ASR.

For both QB and NB, S-boost and Ada-boost are almost the same and they have done a better job than their baselines. Since the performance of these two boosting ensembles is the same, the removed points by ASR were not contributing to the classification task; that is, they were rightfully removed.

For QDA, S-boost has done a better job than Ada-boost and both have shown better results than the baseline. Since the 1st sub Average of S-boost accuracy in Table VI-15 is the same as that of baseline and larger than that of Ada-boost, then ASR has been successful in removing some atypicals but we do not know if it has found them all. The p-values also show a considerable difference in favor of S-boost. Pv(Abst > Sbst) = 70.3 while Pv(Abst > Sbst) = 41.7. Moreover, since the 2nd sub Average of S-boost accuracy in Table VI-15 is the same as that of Ada-boost but larger than that of baseline, then ASR has been successful in not taking useful points from within the misclassified sets (there is no evidence that some of those points were removed by ASR). This is inline with the finding that QDA had the minimum *nAtps*. The case of LDA, is almost the same as that of QDA.

Generally speaking, the results show that ASR has the ability to recognize and remove some of the atypical points from training while keeping its test accuracy on the whole dataset rather as high as the baseline classifier. This is, by itself, a significant

achievement especially considering the generality and features of the proposed technique (no need for a reject option; no need for extra criteria and performance measures; finding atypical points for the whole dataset while keeping the test set intact; consistent with the classifier in use, etc.).

Although all the points removed by ASR are from the set of misclassified points, there seems to be some useful points removed by ASR and also not all atypical points were removed by ASR. The main reasons for this imperfect result can be: 1. ASR gathers the collection of misclassified points in groups (points with the same rank) because they were misclassified together. There may be better ways to group misclassified points together. If a classifier has a kind of reject option, this can be easily modified but to keep the generality of the algorithm, we did not consider that option. 2. In getAtps we perform a backward elimination. Once some points are removed we cannot bring them back into the next subsets. This is problematic in combinatorial optimization problems like sample subset selection. However, better solutions like sequential replacement algorithms are more complicated and computationally more expensive. For the first study of subset selection for samples (rather than features), the results seems to be promising. In the future research the above two issues can be pursued.

7.4 Comparing ASR and filtering

Tables of group 3 (Table VI-18 to Table VI-24) give the results of the comparison between when the points removed from the training set are identified by ASR and when

removing outliers, identified by the Mahalanobis filter, from the training set. For the outlier detection, the Mahalanobis distance was used with Chi-squared distribution and $\alpha = 0.01$, significance level.

Figure VII-15 shows the average accuracies of the baseline, ASR, and the Mahalanobis filter method over all applied datasets. The averages of the baseline and ASR may differ slightly from that of Tables of group 1 and 2 because the Mahalanobis distance could not be calculated for Dermatology dataset; inverse of the covariance matrix is sometimes problematic.

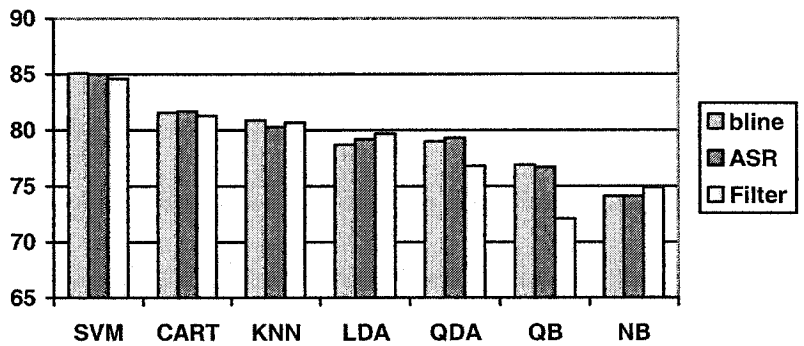

Figure VII-15. Average accuracies (over all datasets) of the baseline, ASR, and the filter method.

As Figure VII-15 shows, ASR can match the baseline accuracy somewhat better than the Mahalanobis filter. There was 0.6% drop in KNN for ASR that could have been fixed by the KNN with the check option (we did not use the check option results in this Section).

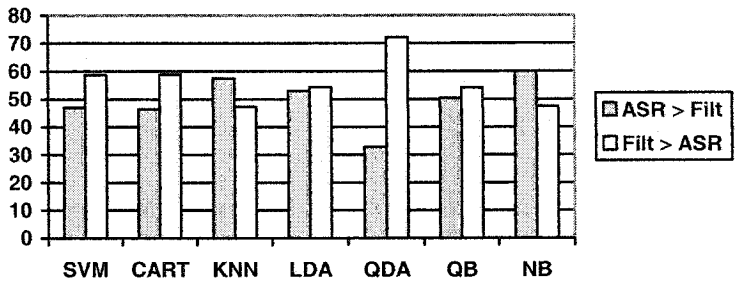

Figure VII-16. The p-values comparing the performance of ASR with the filter method.

In Figure VII-16, the p-values comparing the performances of ASR with that of the Mahalanobis filter method are shown. Please note that the lower the p-value, the more evidence we have to reject the null hypothesis in favor of the alternative hypothesis. There are two alternative hypotheses here for two 1-tailed tests: ASR > Filt (ASR has performed better than the filter) and Filt > ASR (the filter has performed better than ASR). The most evidence in Figure VII-16, is for QDA classifier showing that ASR has performed considerably better than the filter. It is interesting to notice that accuracy measure is not always consistent with p-values. For instance, QB has shown the most decrease in accuracy for the filter but the p-values consider the QDA case more significant.

More detailed information is given in Table VII-3.

Table VII-3. Comparing average values for ASR and the Mahalanobis filter.

classifier	bline Acc	ASR Acc	Filter Acc	nAtps	nOutliers
SVM	85.1	85.0	84.6	77.7	69.0
KNN	80.9	80.3	80.7	97.2	68.8
CART	81.6	81.7	81.3	101.1	59.4
QB	76.9	76.7	72.1	73.0	36.2
NB	74.1	74.1	74.9	42.1	36.2
QDA	79.0	79.3	76.8	35.1	64.6
LDA	78.7	79.2	79.7	98.4	64.6
average:	**79.5**	**79.5**	**78.6**	**74.9**	**57.0**

Table VII-3 summarizes the result by giving the average accuracies and number of removed points in all datasets. Columns 2 to 4 are the average accuracies obtained for the baseline, the ASR, and the Mahalanobis filter (averaged e.g., over five times of 10-fold cross validation). The average number of points identified as atypical is given in the 5^{th} column (*nAtps*). The average number of outliers removed by the filter is in the 6^{th} column (*nOutliers*). The average number of atypicals and outliers are for the training sets (that is 9/10 of data for the datasets with size $\leq$ 1000, and 1/2 of data otherwise) and averaged over the number of iterations times the number of folds (for a dataset) and then over all datasets. There is a change in the number of points removed by the filter (*nOutliers*) although it should have been the same for all classifiers. This is because some classifiers were not stable and did not finish the process for some datasets. The most instable cases were QDA and LDA that had problem (calculating the inverse of the covariance matrix)

with Dermatology and splice1 datasets. The Mahalanobis filter could not finish with the Dermatology data either.

The filter method performed well in NB and LDA classifiers. In LDA, perhaps the filter takes the advantage of having almost the same inductive bias as the classifier and shows 1% increase in performance: 78.7, 79.2, and 79.7 for the baseline, ASR, and the filter method, respectively. The average results, however, show that the removal of points, identified by the filter, from the training set has caused almost 1% drop in the test accuracy. While ASR has removed more points from the training set (74.9 vs. 57, that is almost 30% more), its test accuracy is the same as the baseline. It is worth mentioning that 1% may not be taken as a significant change but considering that 5 times iteration of 10-fold CV was done on most of the 20 datasets (to reduce the randomness in the results), this drop can be considered as meaningful. The drop of accuracy of the filter method is mainly due to QDA and QB classifiers.

It is interesting to note that the QDA is the classifier one expects to benefit most from the removal of points identified by the Mahalanobis filter because they share a common inductive bias (Mahalanobis distance) and follow basically the same calculation. Nevertheless, the average performance of QDA has dropped from 79% to 76.8%. ASR has been successful in controlling this drop by not letting more points to be removed from the training set. Surprisingly, QDA is the only case for which ASR has removed even fewer points than the filter: the average *nAtps* is 35.1 and the average *nOutliers* is 64.6 for QDA; that is, ASR has removed 90% less points in this case. In other words, in the case

of QDA, the filter method and ASR are both consistent with the classifier in terms of their inductive biases but ASR has an extra advantage of stopping the removal process whenever it may start damaging the performance of the classifier. The number of removal by the filter is controlled by a *predetermined* criterion (α = 0.01 significance level). That seems to be the reason why ASR did not allow performance degradation.

In the case of QB, the filter shows an average drop of almost 5% in performance (from 76.9% to 72.1%). In this case, ASR has removed almost *twice* as many as the filter (*nAtps* = 73 vs. *nOutliers* = 36.2) without sacrificing the performance of classification. This shows that removing more atypical points does not necessarily mean compromising the performance as long as the points that have a positive contribution to the classification task are not removed. The problem with the filter approach is that neither the number of removal points nor which points to be removed is decided based on the consultation with the learner (classifier). Even if the classifier and the filter share the same inductive bias, the performance is still in danger because the number of removal may not be what the classifier can afford to give up (the case of QDA).

Figure VII-17 shows, as bar lines, how many points each method (ASR and the filter) has removed from the training sets averaged over all applied datasets. As mentioned earlier, ASR has removed almost 30% more points than the Mahalanobis filter on average.

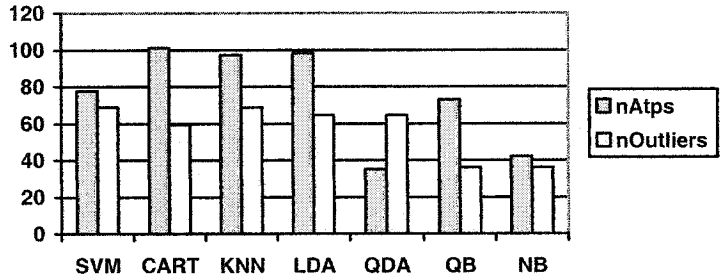

Figure VII-17. Number of atypicals identified by ASR and number of outliers chosen by the filter.

It is worth mentioning that although ASR is able to identify more atypical points, it seems that ASR chooses these points more selectively than a filter method. For instance, the Chess dataset has 3196 examples (points), it is quite deterministic: it has no noise and every example is a unique valid scenario. Yet, the Mahalanobis filter has chosen on average almost 150 points as outliers from the training set (about 300 points in the whole dataset). The baseline accuracy for the Chess data varies from 66.7% (for QB and NB), 88.7% (for QDA) to almost 99% (for SVM). So there were plenty of misclassified points for ASR to choose from. The number of atypical points (*nAtp*) ASR has identified were 3 (for SVM): which were among the outliers as well, 14 (for CART): only 10 of them were among the 300 outliers, and 0 (for all other classifiers).

Another interesting case was the case of SegmentationAll data with the KNN classifier. ASR identified only 2 points as atypicals on average (4 points in the whole dataset). Removing these points from the training set caused KNN to increase the baseline accuracy from 92.8% to 95.2%. The filter chose 102.5 points as outliers on average (205

points in the whole dataset). Removing the outliers from the training set caused KNN to decrease the baseline accuracy from 92.8% to 92%. Surprisingly, none of the 4 atypical points were among the 205 points identified as outliers by the filter. A detailed investigation on these 2 duplicated atypical points (given in Subsection 7.1) showed that one of them is located between the two islands (smaller clusters) formed by the members of its own class (see Figure VII-18). The reason why the Mahalanobis filter does not recognize it as an outlier is that it considers all the members of a class as one group (sample). If they form different islands in the problem space, the points outside of the islands may not be identified as outliers (by the Mahalanobis filter) when they are not in the tail of the group distribution.

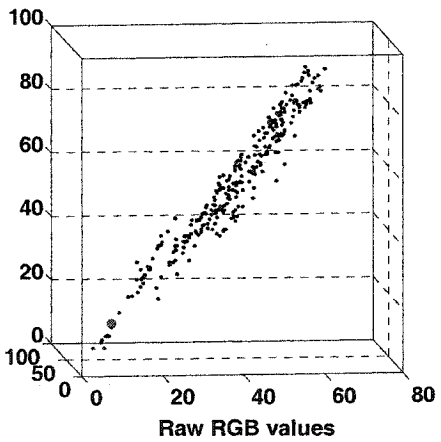

Figure VII-18. Class 2 members and atypical points (bold circle) for SegmentationAll data and KNN.

Although such a clear behavior can only be seen in some cases, generally, ASR has removed almost 30% more points than the filter and still produced 1.0% higher

generalization accuracy than the filter. If both ASR and the filter had removed the same number of points, the one with the higher performance would be considered more accurate and more specific as to which points should be removed. Also, if both ASR and the filter had the same performance, the one that had removed more points would be considered more efficient and also consistent with the classifier in use. This is because by removing more points it was more in risk of removing some informative points and dropping the performance but it did not happen. Therefore, ASR is more consistent with the classifier in use; it is more accurate and more specific in finding the atypical examples. In other words, ASR knows better than the filter: how many points and which ones to remove.

Considering the above findings, the following can be said about the comparison of the Mahalanobis filter and ASR method. ASR knew better which points to remove and had a better control on the number of points to be removed before damaging the performance of the classifier. On average, ASR has identified 30% more atypical points than the Mahalanobis filter. The points removed by ASR were also more consistent with the classifier in use; as a result, ASR has produced 1% higher performance than the Mahalanobis filter. All the points removed by the ASR are guaranteed to be either outliers or from the conflict area of the problem space where more than one class claims for the region (overlapping region). This guarantee comes from the fact that atypical points are a subset of the misclassified set (by definition) but there is no such requirement for the filtered outliers. Note that removing more points, by itself (everything else equal), is more preferable for the possible goals of atypical detection (cleaning the data from

noise, making a smoother decision function to act against overfitting, finding the possible instances of a new class, etc.).

The Mahalanobis filter, on the other hand, is very fast (like many other filters) and ASR is considerably slower because it is a wrapper method. The elapsed time for SVM was from 2.5 mins for Iris data to almost 50 mins for Thyroid data and exceptionally 140 mins for Vehicle data; the average elapsed time was about 30 mins. This was on a PC with 2.4 GHz CPU and 512 Megs of RAM and included all the processes involved (baseline, ASR, Ada-boost, S-boost, etc.). The average time for CART was almost as that of SVM. Other classifiers were faster than SVM with an average of about 20 mins. This is while the code was not optimized for speed and there is plenty of room for that.

VIII Conclusion and future work

The main goals of this research were to develop a wrapper method for atypical detection with a running time that allows sample subset selection to be practical and remove some of the obstacles faced by the previous efforts. While the fastest available wrapper (sequential) algorithm is quadratic, the presented algorithm (ASR) has a linear time. Our analysis showed that ASR is at least 75 times faster than a standard sequential wrapper averaged over the 20 tested datasets and 7 classifiers. A wrapper method is more consistent with the classifier in use than a filter method. This consistency can be seen in the way a wrapper technique works and the results of the comparisons between the two methods. The filter approach applies one definition (predetermined criterion) to all domains while the wrapper method finds atypical points by consulting the classifier in use. Overall, considering the fact that ASR is only a linear approximation to an exponential (combinatorial) problem, the obtained results seems to be satisfactory.

8.1 Conclusion

The experimental results obtained in this research did not support the assertions that removal of atypical points can increase the generalization performance of classifiers averaged over all datasets. Our closer look at the reports of performance increase as a result of atypical removal also showed that the reported performance was either confidence expressed by accuracy (and not the prediction accuracy) or the efficiency of

the filter expressed by accuracy. As long as the generalization performance is concerned, no evidence was found so far to support that overall performance can be increased as a result of atypical removal. To the best of the author's knowledge, for 0% noise in the training data, the highest *overall* achievement, obtained so far, is the test accuracy almost the same as that of the baseline.

For instance, in the study by Wilson and Martinez (2000b), when no noise was added to the training data, all their tested instance reduction techniques showed generalization accuracies weaker than that of the baseline (a KNN with 0% noise in the training data and no filtering). Brodley and Friedl (1999) also found that when no noise was introduced, filtering did not make a significant difference although there was no guarantee that the datasets themselves were noise free. John (1995) also experienced that Robust-C4.5, specialized in the removal of outliers, degraded performance in some datasets by throwing out points that seemed to be outliers.

Therefore, the best achieved so far by removing outliers, noise, or atypicals is to keep the overall (average) performance on the intact test set almost the same as that of the baseline with no removal. In our experiments, there are some datasets for which the test accuracy shows statistically significant improvement after the removal of atypicals; but we cannot make a conclusion only based on such results when we are interested in the overall performance.

All classifiers using the ASR algorithm, after removing atypical points from their training set, were able to keep their performance comparable to the baseline performance for which training was done without any removal. In the case of KNN (a local learner), 1% difference in the performance of ASR and the baseline was observed. Even this difference was vanished when ASR with the check option was used. This shows that ASR, as a wrapper method, is capable of identifying atypical points, as defined in this research, and removing them from the training set without sacrificing the performance on the intact test set. Considering that sample subset selection is a combinatorial problem with an exponential number of possible state, for a linear approximation (ASR) the obtained results are not disappointing at all.

Note that even if the solution represented in this research does not show any significant increase in the test performances, logically, we cannot exclude the chance that with a better algorithm and a finer search we may be able to enhance the test performance in the future.

Further tests on all 20 datasets and 7 classifiers were conducted using two boosting ensembles: one with intact training set (Ada-boost) and another with the atypical points removed from the training set (S-boost). Since the presence of misclassified points is an essential element for the boosting process to increase the performance of the classifier, S-boost should perform worse than Ada-boost unless the removed misclassified points were not of much use to the classification task. The results showed that when a significant portion of misclassified points were removed from the training set, Ada-boost and S-

boost ensembles had very close performance measures. This indicates that ASR is able to extract atypical points from a dataset without taking most of the informative points from the set of misclassified points. ASR may also have taken some points that are perhaps not atypical points (it is not perfect) but it does generate about the same test accuracy as the baseline classifier.

The comparison between ASR and the Mahalanobis filter method was also done. The results showed that ASR was more accurate in identifying atypical points, it was more consistent with the classifier in use by keeping its performance as high as that of the baseline (the classifier with no removal from the training set), and it was able to remove 30% more points from the dataset than the Mahalanobis filter.

The proposed algorithm (ASR) has the following features:

1. ASR has a linear running time to approximate sample subset selection for atypical detection. The closest wrapper methods are quadratic that is not practical for sample subset selection.

2. In ASR, as a wrapper method, there is no predetermined criterion for atypical points but they are found based on the possible negative effect of misclassified points to the classification task. That is, ASR only uses the classifier itself and any performance measure (like accuracy) to identify atypical points in the data. This is more consistent

with any classifier in use and therefore, more cognitively sound from the AI point of view.

3. ASR finds atypical points for the whole dataset (training and test sets) while keeping the test set intact, unlike error-reject and CP curves, by applying a cross-validation technique. The results are more comprehensive and even more reliable than partitioning data into training and test sets.

4. ASR generates a non-monotonic curve that is useful for atypical removal. Unlike error-reject and CP curves, ASR gives a cut off point for atypicals.

5. ASR does not need an error-reject option (CF or posterior probability estimate) to give ranks to the obtained atypical points, so it can be used for any classifier. The rank of atypical points can be found easily, at the end, by considering their frequencies in K-folds of the CV process.

6. In ASR, there is no need for new performance measures. The cleaned dataset and the trained classifier are free from the possible influence of atypical points.

7. Compared to the Mahalanobis filter, ASR is more specific in identifying atypical points, more consistent with the classifier in use by keeping its performance as high as that of the baseline, and able to remove 30% more points from the dataset than the filter. The points removed by ASR are always a subset of misclassified set.

8. ASR is a non-parametric method of atypical detection. It has no pre-determined parameter (such as $\alpha = 0.01$) that may hurt the performance of the classifier. Our experiments proved that even if there is no conflict in inductive bias of the classifier and the outlier filtering method, a pre-determined parameter may cause the removal of too many or too few points.

8.2 Future work

There are certain points that can help better understand and resolve the issues related to atypical detection for a wrapper approach such as ASR. These points can be the subject of the future research.

1. ASR removes the collection of misclassified points in groups because they were misclassified together. This grouping method may not be in harmony with the combinatorial nature of sample subset selection. Is there a different method of grouping for subset generation that does not increase the running time of ASR? If yes, how is its performance compared to that of ASR? One option, of course, is to use the classifiers that have reject option (CF) to rank the points for removal. But this makes the sample subset selection method less general and perhaps more time consuming. Is the performance of such a wrapper method significantly better than ASR?

2. In getAtps we perform a backward elimination. In backward elimination, once some points are removed we cannot bring them back into the next subsets. This is a greedy approach and often reduces the quality of the result of any combinatorial optimization problem. Better solutions like sequential replacement algorithms are also available but they are more complicated and computationally more expensive. One can try these algorithms and compare the results with ASR. Are they practical for rather small datasets; is their performance better than ASR? How about using genetic algorithms to search the space of possible subsets? Can we use fast versions of GA with the ability to rank the atypical points?

3. The effect of atypical removal on the decision surface is also of interest. One expects that the removal of typical points may help smooth the decision surface. Can atypical removal prevent overfitting to some degree? This can be studied on both generated and real datasets. In weaker classifiers, one may expect to see more change in the decision surface after removing atypical points. Does the process of atypical removal bring weaker classifiers closer to an optimal decision surface?

References

Aha, D.W. Breslow, L.A. and Muñoz-Avila, H. (2000), "Conversational case-based reasoning" Applied Intelligence, 14, pp. 9-32.

Barnett, V. and Lewis, T. (1994), "Outliers in statistical data, 3rd ed.", John Willey & Sons, New York.

Beckman, R.J. and Cook, R.D. (1983), "Outlier....s", in Technometrics 25, No. 2, p. 119-149.

Bernoulli, D. (1777), "The most probable choice between several discrepant observations and the formation therefrom of the most likely induction", in C.G. Allen (1961), Biometrika, 48, 3-13.

Blake, C.L. and Merz, C.J. (1998), UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Blum, A.L. (1997), "Selection of relevant features and examples in machine learning", Artificial Intelligence, 97, pp 245-271.

Boser, B.E. Guyon, I.M. and Vapnik, V.N. (1992), "A Training Algorithm for Optimal Margin Classifieres", In 5th annual workshop on computational learning theory, Pittsburgh, 1992, ACM.

Breiman, L. (1998), "Arcing classifiers", The Annals of Statistics, 26(3), pp 801–849.

Breiman, L. (1996), "Bagging predictors", Machine Learning 24, 123-140.

Breiman, L. Friedman, J.H. Olshen, R.A. and Stone, C.J. (1984), "Classification and Regression Trees", Monterey, CA, Wadsworth Intl. Group.

Brodley, C.E. and Friedl, M.A. (1999), "Identifying mislabeled training data", J. of Artificial Intelligence Research, 11, 131-167.

Burges, C.J.C. (1998), "A tutorial on support vector machines for pattern recognition", Kluwer academic publishers, Boston, Manufactured in the Netherlands.

Chang, C.C. and Lin, C.J. (2002), a Library for Support Vector Machines (Version 2.33), http://www.csie.ntu.edu.tw/~cjlin.

Chow, C.K. (1970), "On optimum recognition error and reject tradeoff", IEEE Transactions on Information Theory, Vol. IT-16, No. 1.

Dash, M. and Liu, H. (1997), "Feature selection for classification", Intelligent Data Analysis 1 (1997), 131-156.

Domeniconi, C. and Gunopulos, D. (2001), "Adaptive Nearest Neighbor Classification using Support Vector Machines", TR, UCR-CSE-01-04.

Duch, W. Adamczak, R. Grabczewski, K. and Zal, G. (1999), "Hybrid neural-global minimization method of logical rule extraction", Journal of Advanced Computational Intelligence, 3 (5): 348-356.

Fumera, G. Roli, F. and Giacinto, G. (2000), "Reject option with multiple thresholds", Pattern Recognition, 33(12), pp. 165-167.

Fumera, G. and Roli, F. (2002), "Support Vector Machines with Embedded Reject Option", Proc. of Int. workshop on Pattern Recognition with Support Vector Machines (SVM2002), Niagara Falls, Canada. Springer, LNCS Vol. 2388, pp. 68-82.

Gamberger, D. Lavrac, N. and Dzeroski, S. (1996), "Noise elimination in inductive concept learning: A case study in medical diagnosis", In the 7th international workshop on algorithmic learning theory, pp199-212. Springer.

Guyon, I. Matic, N. and Vapnik, V. (1996), "Discovering informative patterns and data cleaning", In Fayyad, U.M., Piatetsky-Shapiro, G., Smith, P. and Uthurasamy, R.,

(Eds), Advances in knowledge discovery and data mining, pp. 181-203. AAAI-MIT press.

Han, J. and Kamber M. (2001), "Data Mining Concepts and Techniques", Morgan Kaufman Pub., NY.

Hashemi, S. and Trappenberg, T.P. (2002), "Using SVM for Classification in Datasets with Ambiguous data", The 6th World Multiconference on Systemics, Cybernetics, and Informatics. Orlando, Florida (USA), July 2002.

Hashemi, S. (2002), "Coverage-Performance Curves for Classification in Datasets with Atypical Data", Proceedings of the 1st IEEE Int. Conference on Machine Learning and Cybernetics, Beijing, Nov. 2002.

Japkowicz, N. (2003), "Class imbalances, are we focusing on the right issue?", Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC, 2003.

John, H.G. Kohavi, R. and Pfleger, K. (1994), "Irrelevant Features and the Subset Selection Problem", In International Conference on Machine Learning, p 121-129.

John, H.G. (1995), "Robust Decision Trees: Removing Outliers from Databases", In Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining, p 174-179, Montreal, CA.

Jude, C.M. and McClelland, G.H. (1989), "Data Analysis, a model-comparison approach", Harcourt Brace Jovanovich, Publishers.

Kohavi, R. and John, G.H. (1997), "Wrappers for feature subset selection", AIJ special issue on relevance.

Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection", IJCAI, 1995.

Komorowski, J. and Øhrn, A. (1999), "Modelling prognostic power of cardiac tests using rough sets", Artificial Intelligence in Medicine 15, pp.167–191

Krogh, A. and Vedelsby, J. (1995), "Neural network ensembles, cross-validation, and active learning", In Advances in Neural Information Processing Systems, Vol. 7, pp. 231-238 Cambridge, MA, MIT Press.

Krzanowski, W.J. (1998), "Principles of Multivariate Analysis", Oxford University Press, Oxford.

Maclin, R. and Opitz, D. (1997), "An Empirical Evaluation of Bagging and Boosting", In Proceedings of the 14th National Conference on Artificial Intelligence, p 546-551.

Millar, A.M. and Hamilton, D.C. (1999), "Modern outlier detection methods and their effect on subsequent inference", J. Statist. Comput. Simul. Vol. 64, pp. 125-150.

Miller, A.J. (1990), "Subset Selection in Regression", Chapman and Hall, NY.

Opitz, D. and Maclin, R. (1999), "Popular Ensemble Methods: an empirical study", Journal of Artificial Intelligence Research, 11, 169-198.

Opitz, D. and Shavlik, J. (1996), "Actively searching for an effective neural-network ensemble", Connection Science, 8 (3/4), pp. 337-353.

Provost, F.J. (1992), "Policies for the selection of bias in inductive machine learning", PhD thesis, University of Pittsburgh, Computer Science Department. Report no. 92-34.

Ripley, B.D. (1996), "Pattern recognition and neural networks", Cambridge Univ. Press.

Schapire, R. Freund, Y. Bartlett, P. and Lee, W. (1998), "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods", The Annals of Statistics, May 1998.

Schölkopf, B. Williamson, R. Smola, A. Shawe-Taylor, J. and Platt, J. (2000), "Support Vector Method for Novelty Detection", Advances in Neural Information Processing Systems 12, pp. 582-588.

Skalak, D.B. (1994), "Prototype and Feature Selection by Sampling and Random Mutation Hill-Climbing Algorithms", Proceedings of the Eleventh International Conference on Machine Learning, pp. 293-301, New Brunswick, New Jersey, 1994.

Skalak, D.B. (1993), "Using a Genetic Algorithm to Learn Prototypes for Case Retrieval and Classification", Proceedings of the AAAI-93 Case-Based Reasoning Workshop, Washington, D.C., American Association for Artificial Intelligence, July 1993.

Trappenberg, T.P. and Back, A.D. (2000), "A classification scheme for applications with ambiguous data", International Joined Conference on Neural Networks, IJCNN.

Trappenberg, T.P. Back, A.D. and Amari, S.-I. (1999), "A Performance Measure for Classification with Ambiguous Data", BSIS Technical Reports No. 99-67, May, 1999.

Vapnik, V. (1998), "Statistical Learning Theory", John Wiley and Sons Inc., New York.

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

Wettschereck, D. Aha, D.W. and Mohri, T. (1997), "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms", Artificial Intelligence Review.

Williams, G. Baxter, R. He, H. Hawkins, S. and Gu L. (2002), "A Comparative Study of RNN for Outlier Detection in Data Mining", the Proceedings of the 2nd IEEE Intl. Conf. on Data Mining (ICDM02), Maebashi City, Japan.

Wilson, D.R. and Martinez, T.R. (2000a), "An integrated instance-based algorithm", Computational Intelligence, Volume 16, Number 1, pp. 1-28.

Wilson, D.R. and Martinez, T.R. (2000b), "Reduction Techniques for Instance-Based Learning Algorithms", Machine Learning, 38 (3): pp. 257-286.

Witten, I.H. and Frank, E. (2000), "Data Mining, Practical machine learning tools and techniques with Java implementation", Morgan Kaufmann Pub. Ca.