Chapter 6

SUPERVISED OUTLIER DETECTION

"True, a little learning is a dangerous thing, but it still beats total ignorance." – Abigail van Buren

1. Introduction

The discussion in the previous chapters focussed on the problem of unsupervised outlier detection in which no prior information is available about the abnormalities in the data. In such scenarios, many of the anomalies found correspond to noise, and may not be of any interest to an analyst. It has been observed [284, 315, 440] in diverse applications such as system anomaly detection, financial fraud, and web robot detection that *the nature of the anomalies is often highly specific to particular kinds of abnormal activity in the underlying application.* In such cases, unsupervised outlier detection methods may often discover noise, which may not be specific to that activity, and therefore may not also be of any interest to an analyst. The goal of supervised outlier detection is to incorporate application-specific knowledge into the outlier analysis process, so as to obtain more meaningful anomalies with the use of learning methods. Because of the rare nature of anomalies, such data is often limited, and it is hard to create robust and generalized models on this basis. Nevertheless, the general observation has been that the incorporation of learning methods can significantly improve the robustness of the outlier analysis process. *The general recommendation for outlier analysis is to always use supervision where possible.*

In most real data domains, some examples of normal or abnormal data may be available. This is referred to as *training data*, and can be used to create a *classification model*, which distinguishes between normal and anomalous instances. The problem of classification has been widely

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-1-4614-6396-2_6,

studied in its own right, and numerous algorithms are available in the literature [146] for creating supervised models from training data. In many cases, different kinds of abnormal instances may be available, in which case the classification model may be able to distinguish between them. For example, in an intrusion scenario, different kinds of intrusion anomalies are possible, and it may be desirable to distinguish among them.

So how is the supervised outlier detection problem different from classification? The supervised outlier detection problem may be considered a very difficult special case (or variation) of the classification problem, depending upon the following possibilities, which may be present either in isolation or in combination.

- *Class Imbalance:* Since outliers are defined as rare instances in the data, it is natural that the distribution between the normal and rare class will be very skewed. From a practical perspective, this implies that the optimization of classification accuracy may not be meaningful, especially since the misclassification of positive (outlier) instances is less desirable than the misclassification of negative (normal) instances. In other words, false positives are more acceptable than false negatives. This leads to cost-sensitive variations of the classification problem, in which the objective function for classification is changed.

- *Contaminated Normal Class Examples (Positive-Unlabeled Class Problem):* In many real scenarios, the data may originally be present in unlabeled form, and manual labeling is performed for annotation purposes. In such cases, only the positive class is labeled, and the remaining "normal" data contains some abnormalities. This is natural in large scale applications such as the web and social networks, in which the sheer volume of the underlying data makes contamination of the normal class more likely. For example, consider a social networking application, in which it is desirable to determine spam in the social network feed. A small percentage of the documents may be spam. In such cases, it may be possible to recognize and label some of the documents as spam, but many spam documents may remain in the examples of the normal class. Therefore, the "normal" class may also be considered an unlabeled class. In practice however, the unlabeled class is predominantly the normal class, and the anomalies in it may be treated as contaminants. The classification models need to be built to account for this. Technically, this case can be considered a form of partial supervision [306], though it can also be treated as a

difficult special case of full supervision, in which the normal class is more noisy and contaminated. Standard classifiers can be used on the positive-unlabeled version of the classification problem, as long as the relative frequency of contaminants is not extreme. In cases where the unlabeled class does not properly reflect the distribution in the test instances, the use of such unlabeled classes can actually *harm* classification accuracy [301].

A different flavor of incomplete supervision refers to missing training data about an *entire* class, rather than imperfect or noisy labels. This case is discussed below.

- *Partial Training Information (Semi-supervision or novel class detection):* In many applications, examples of one or more of the anomalous classes may not be available. For example, in an intrusion detection application, one may have examples of the normal class, and *some* of the intrusion classes, as new kinds of intrusions arise with time. In some cases, examples of one or more normal classes are available. A particularly commonly studied case is the *one class variation*, in which only examples of the normal class are available. The only difference between this extreme case and the unsupervised scenario is that the examples of the normal class are typically guaranteed to be free of outliers. In many applications, this is a very natural consequence of the extreme rarity of the outlier. For example, in a bio-terrorist attack scenario, no examples of anomalous classes may be available, since no such event may have occurred in the first place. Correspondingly, the examples of the training class are also guaranteed to be free of outliers. This particular special case, in which the training data contains only normal classes, is much closer to the unsupervised version of the outlier detection problem. This will be evident from the subsequent discussion in the chapter.

It is evident that most of the above cases are either a special case, or a variant of the classification problem, which provides different challenges. Furthermore, it is possible for some of these conditions to be present in combination. For example, in an intrusion detection application, labeled data may be available for some of the intrusions, but no labeled information may be available for other kinds of intrusions. Thus, this scenario requires the determination of both rare classes and novel classes. In some cases, rare class scenarios can be reduced to partially supervised scenarios, when only the rare class is used for training purposes. Therefore, the boundaries between these scenarios are often blurred in real applications. Nevertheless, since the techniques for the different scenar-

ios can usually be combined with one another, it is easier to discuss each of these challenges separately. Therefore, a section will be devoted to each of the aforementioned variations of the supervised outlier detection problem.

The discussion in this chapter will be focussed on more generic forms of multidimensional data, rather than specific kinds of data such as temporal and spatial data. This is because the general principles of supervised outlier detection are often independent of specific data type and can be easily generalized to more complex data domains. The goal of this chapter is to provide an understanding of how classification methods need to be *modified* in order to address the challenges of supervised outlier analysis. Therefore, a working knowledge of the classification problem is assumed [146] for the purposes of this chapter. Furthermore, a section on supervised methods will be included in many of the chapters which address the more complex data types.

A particular form of supervision is *active learning*, when human experts may intervene during the outlier detection process in order to identify relevant instances. Very often, active learning may be accomplished by providing an expert with candidates for outliers, which are followed by the expert explicitly labeling these pre-filtered examples. In such cases, *label acquisition* is combined with *model construction* in order to progressively incorporate more human knowledge into the outlier analysis process. Such a human-computer cooperative approach can sometimes provide more effective results than automated techniques.

Paucity of training data is a common problem, when the class distribution is imbalanced. Even in a modestly large training data set, only a small number of rare instances may be available. Typically, it may be expensive to acquire examples of the rare class. Imbalanced class distributions could easily lead to training algorithms which show differentially overfitting behavior. In other words, the algorithm may behave robustly for the normal class, but may overfit the rare class. Therefore, it is important to design the training algorithms, so that overfitting is avoided.

This chapter is organized as follows. The next section will discuss the problem of rare-class detection in the fully supervised scenario. The semi-supervised case of classification with positive and unlabeled data will be studied in section 3. Section 4 will discuss the problem of novel class detection. This is also a form of semi-supervision, though it is of a different kind. Methods for outlier detection with human supervision are addressed in section 5. The conclusions and summary are presented in section 6.

2. The Fully Supervised Scenario: Rare Class Detection

The problem of rare-class detection or *class imbalance* is a common one in the context of supervised outlier detection. The straightforward use of evaluation metrics and classifiers which are not cognizant of this class imbalance may lead to very surprising results. For example, consider a medical application in which it is desirable to identify tumors from medical scans. In such cases, 99% of the instances may be normal, and only 1% are abnormal.

Consider the trivial classification algorithm, in which every instance is labeled as normal without even examining the feature space. Such a classifier would have a very high absolute accuracy of 99%, but would not be very useful in the context of a real application. In fact, many forms of classifiers (which are optimized for absolute accuracy) may show a degradation to the trivial classifier. For example, consider a k-nearest neighbor classifier, in which the majority class label in the neighborhood is reported as the relevant class label. Because of the inherent bias in the class distribution, the majority class may very often be normal even for abnormal test instances. Such an approach fails because it does not account for the *relative* behavior of the test instances with respect to the original class distribution. For example, if 49% of the training instances among the k-nearest neighbors of a test instance are anomalous, then that instance is *much* more likely to be anomalous *relative* to its original class distribution. By allowing changes to the classification criterion, such as reporting non-majority anomalous classes as the relevant label, it is possible to improve the classification accuracy of anomalous classes. However, the *overall* classification accuracy may degrade. Of course, the question arises whether the use of measures such as overall classification accuracy is meaningful in the first place. Therefore, the issue of *evaluation* and *model construction* are closely related in the supervised scenario. The first step is to identify how the rare class distribution relates to the objective function of a classification algorithm, and the algorithmic changes required in order to incorporate the modifications to the modeling assumptions.

There are two primary classes of algorithms which are used for handling class imbalance:

- *Cost Sensitive Learning:* The objective function of the classification algorithm is modified in order to weight the errors in classification differently for different classes. Classes with greater rarity have higher costs. Typically, this approach requires algorithm-

specific changes to different classifier models in order to account for costs.

- *Adaptive Re-sampling:* The data is re-sampled so as to magnify the *relative proportion* of the rare classes. Such an approach can be considered an *indirect form* of cost-sensitive learning, since data re-sampling is equivalent to implicitly assuming higher costs for misclassification of rare classes.

Both these methodologies will be discussed in this section. For the case of the cost-sensitive problem, it will also be discussed how classification techniques can be heuristically modified in order to approximately reflect costs. A working knowledge of classification methods is assumed in order to understand the material in this section. The reader is also referred to [146] for a description of the different types of classifiers.

For the discussion in this section, it is assumed that the training data set is denoted by $\mathcal{D}$, and the labels are denoted by $L = \{1, \ldots k\}$. Without loss of generality, it can be assumed that the normal class is indexed by 1. The ith record is denoted by $\overline{X_i}$, and its label l_i is drawn from L. The number of records belonging to the ith class are denoted by N_i, and $\sum_{i=1}^{k} N_i = N$. The class imbalance assumption implies that $N_1 >> N - N_1$. While imbalances may exist between other anomalous classes too, the major imbalance occurs between the normal and the anomalous classes.

2.1 Cost Sensitive Learning

In cost sensitive learning, the goal is to learn a classifier, which maximizes the weighted accuracy over the different classes. The *misclassification cost* of the ith class is denoted by c_i. Some models [145] use a $O(k \times k)$ cost matrix to represent the full spectrum of misclassification behavior. In such models, the cost is dependent not only on the class identity of the misclassified instance, but is also dependent on the specific class label *to which* it is misclassified. A simpler model is introduced here, which is more relevant to the rare class detection problem. Here the cost only depends on the origin class, and not on a combination of the origin and destination class. The goal of the classifier is to learn a training model which minimizes the *weighted misclassification rate*.

The choice of c_i is picked in an application specific manner, though numerous heuristics exist to pick the costs in an automated way. The work in [497] proposes methods to learn the costs directly in a data driven manner. Other simpler heuristic rules are used often in many practical scenarios. For example, by choosing the value of c_i to be proportional to $1/N_i$, the *aggregate* impact of the instances of each class on

the weighted misclassification rate is the same, in spite of the imbalance between the classes. Such methods are at best rule-of-thumb techniques for addressing imbalance, though more principled methods also exist in the literature. Many such methods will be discussed in this chapter.

2.1.1 MetaCost: A Relabeling Approach. A general framework known as MetaCost [145] uses a *relabeling* approach to classification. This is a *meta-algorithm*, which can be applied to any classification algorithm. In this method, the idea is to relabel some of the training instances in the data, by using the costs, so that normal training instances, which have a reasonable probability of classifying to the rare class are relabeled to that rare class. Of course, rare classes may also be relabeled to a normal class, but the cost-based approach is *intended to* make this less likely. Subsequently, a classifier can be used on this more balanced training data set. The idea is to use the costs in order to move the decision boundaries in a cost-sensitive way, so that normal instances have a greater chance of misclassification than rare instances, and the *expected misclassification cost* is minimized.

In order to perform the relabeling, the classifier is applied to each instance of the training data and its classification prediction is combined with costs for re-labeling. Then, if a classifier predicts class label i with probability $p_i(\overline{X})$ for the data instance $\overline{X}$, then the expected misclassification cost of the prediction of $\overline{X}$, *under the hypothesis that it truly belonged to* r, is given by $\sum_{i \neq r} c_i \cdot p_i(\overline{X})$. Clearly, one would like to minimize the expected misclassification cost of the prediction. Therefore, the *MetaCost* approach tries different hypothetical classes for the training instance, and relabels it to the class which minimizes the expected misclassification cost. A key question arises as to how the probability $p_i(\overline{X})$ may be estimated from a classifier. This probability clearly depends upon the specific classifier which is being used. While some classifiers explicitly provide a probability score, not all classifiers provide such probabilities. The work in [145] proposes a bagging approach [80] in which the training data is sampled with replacement (bootstrapping), and a model is repeatedly constructed on this basis. The training instances are repeatedly classified with the use of such a bootstrap sample. The fraction of predictions (or votes) for a particular class across different training data samples are used as the classification probabilities.

The challenge of such an approach is that relabeling training data is always somewhat risky, especially if the bagged classification probabilities do not reflect intrinsic classification probabilities. In fact, each bagged classification model-based prediction is *highly correlated* to the

others (since they share common training instances), and therefore the aggregate estimate is not a true probability.

In practice, the estimated probabilities are likely to be *very* skewed towards one of the classes, which is typically the normal class. For example, consider a scenario in which a rare class instance (with global class distribution of 1%) is present in a local region with 15% concentration of rare class instances. Clearly, this rare instance shows informative behavior in terms of *relative* concentration of the rare class in the locality of the instance. A vanilla 20-nearest neighbor classifier will virtually *always*[1] classify this instance to a normal class in a large bootstrapped sample. This situation is not specific to the nearest neighbor classifier, and is likely to occur in many classifiers, when the class distribution is very skewed. For example, an unmodified Bayes classifier will usually assign a lower probability to the rare class, because of its much lower *a-priori* probability, which is factored into the classification. Consider a situation, where a hypothetically perfect Bayes classifier has a prior probability of 1% and a posterior probability of 30% for the correct classification of a rare class instance. Such a classifier will typically assign far fewer than 30% of the votes to the rare class in a bagged prediction, especially[2] when large bootstrap samples are used. In such cases, the normal class will win every time in the bagging because of the prior skew. This means that the bagged classification probabilities can sometimes be close to 1 for the normal class in a skewed class distribution.

This suggests that the effect of cost weighting can sometimes be overwhelmed by the erroneous skews in the probability estimation attained by bagging. In this particular example, even with a cost ratio of 100 : 1, the rare class instance will be wrongly relabeled to a normal class. This moves the classification boundaries in the opposite direction of what is desired. In fact, in cases where the unmodified classifier degrades to a trivial classifier of always classifying to the normal class, the expected misclassification cost criterion of [145] will result in relabeling all rare class instances to the normal class, rather than the intended goal of selective relabeling in the other direction. In other words, relabeling

[1]The probability can be (approximately) computed from a binomial distribution to be at least equal to $\sum_{i=0}^{9} \binom{20}{i} \cdot 0.15^i \cdot 0.85^{20-i}$ and is greater than 0.999.

[2]The original idea of bagging was not designed to yield class probabilities [80]. Rather, it was designed to perform robust prediction for instances, where either class is an almost equally good fit. In cases, where one of the classes has a "reasonably" higher (absolute) probability of prediction, the bagging approach will simply boost that probability to almost 1, when counted in terms of the number of votes. In the rare class scenario, it is expected for unmodified classifiers to misclassify rare classes to normal classes with "reasonably" higher probability.

may result in a further *magnification* of the errors arising from class skew. This leads to degradation of classification accuracy, *even from a cost-weighted perspective.*

In the previous example, if the *fraction* of the 20-nearest neighbors belonging to a class are used as its probability estimate for relabeling, then much more robust results can be obtained with *MetaCost.* Therefore, the effectiveness of *MetaCost* depends on the quality of the probability estimate used for re-labeling. Of course, if good probability estimates are directly available from the training model in the first place, then a test instance may be directly predicted using the expected misclassification cost, rather than using the indirect approach of trying to "correct" the training data by re-labeling. This is the idea behind weighting methods, which will be discussed in the next section.

2.1.2 Weighting Methods. Most classification algorithms can be modified in natural ways to account for costs with some simple modifications. The primary driving force behind these modifications is to implicitly treat each training instance with a weight, where the weight of the instance corresponds to its misclassification cost. This leads to a number of simple modifications to the underlying classification algorithms. In most cases, the weight is not used explicitly, but the underlying classification model is changed to reflect such an implicit assumption. Some methods have also been proposed in the literature [496] in order to incorporate the weights explicitly into the learning process. In the following, a discussion is provided about the natural modifications to the more common classification algorithms.

Bayes Classifier The modification of the Bayes classifier provides the simplest case for cost-sensitive learning. In this case, changing the weight of the example only changes the a-priori probability of the class, and all other terms within the Bayes estimation remain the same. Therefore, this is equivalent to multiplying the Bayes probability in the unweighted case with the cost, and picking the largest one. Note that this is the same criterion that is used in *MetaCost*, though the latter uses this criterion for *relabeling* training instances, rather than predicting test instances. When good probability estimates are available from the Bayes classifier, the test instance can be directly predicted in a cost-sensitive way.

Proximity-based Classifiers In nearest neighbor classifiers, the classification label of a test instance is defined to be the majority class from its k nearest neighbors. In the context of *cost-sensitive* classification, the *weighted* majority label is reported as the relevant one, where the

weight of an instance from class i is denoted by c_i. Thus, fewer examples of the rare class need to be present in a neighborhood of a test instance, in order for it to be reported as the relevant one. In a practical implementation, the number of k-nearest neighbors for each class can be multiplied with the corresponding cost for that class. The majority class is picked *after* the weighting process. A discussion of methods for k-nearest neighbor classification in the context of data classification may be found in [506].

Rule-based Classifiers In rule-based classifiers, frequent pattern mining algorithms may be adapted to determine the relevant rules at a given level of support and confidence. A rule relates a condition in the data (eg. ranges on numeric attributes) to a class label. The support of a rule is defined as the number of training instances which are relevant to that rule. The confidence of a rule is the fractional probability that the training instance belongs to the class on the right hand side, if it satisfies the conditions on the left-hand side. Typically, the data is first discretized, and all the relevant rules are mined from the data, at pre-specified levels of support and confidence. These rules are then prioritized based on the underlying confidence (and sometimes also the support). For a given test instances, all the relevant rules are determined, and the results from different rules can be combined in a variety of ways (eg. majority class from relevant rules, top matching rule etc.) in order to yield the final class label.

Such an approach is not difficult to adapt to the cost-sensitive case. The main adaptation is that the weights on the different training examples need to be used during the computation of measures such as the support or the confidence. Clearly, when rare examples are weighted more heavily, the confidence of a rule will be much higher, when its right hand side corresponds to a rare class because of the weighting. This will result in the selective emphasis of rules corresponding to rare instances. Some methods for using rule-based methods in imbalanced data classification are proposed in [245, 247].

2.1.3 Decision Trees. In decision trees, the training data is recursively partitioned, so that the instances of different classes are successively separated out at lower levels of the tree. The partitioning is performed by using conditions on one or more features in the data. Typically, the split criterion uses the various entropy measures such as the gini-index for deciding the choice of attribute and the position of the split. For a node containing a fraction of instances of different classes denoted by $p_1 \ldots p_k$, its gini-index is denote by $1 - \sum_{i=1}^{k} p_i^2$.

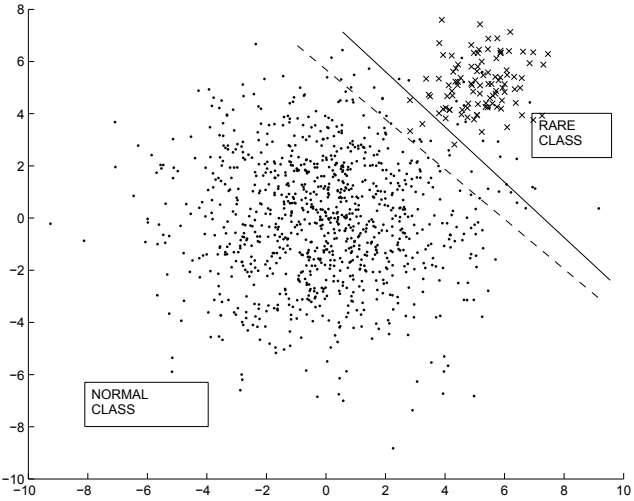

Figure 6.1. Optimal hyperplanes will change because of weighting of examples

Better separations of different classes lead to lower gini-index. The split attribute and partition point is decided as one which minimizes the gini-index of the children nodes. By using costs as weights for the instances, the computation of the gini-index will be impacted so as to selectively determine regions of the data containing higher proportions of the rare class. Some examples of cost-sensitive decision trees are discussed in [450, 463].

2.1.4 SVM Classifier. SVM classifiers work by learning hyperplanes, which optimally separate the two classes in order to minimize the expected error. Thus, SVM classifiers can be modeled as an optimization problem, where the goal is to learn the coefficients of the underlying hyperplane. For example, a two-class example has been illustrated in Figure 6.1. The optimal separator hyperplane for the two classes is illustrated in the same figure with the solid line. However, it is possible to change the optimization model by incorporating weights (or costs) into the optimization problem. This shifts the decision boundary, so as to allow erroneous classification of a larger number of normal instances, while correctly classifying more rare instances. The result would be a reduction in the overall classification accuracy, but an increase in the cost-sensitive accuracy. For example, in the case of Figure 6.1, the optimal separator hyperplane would move from the solid line to the dotted line in the figure. The issue of class-boundary re-alignment for SVMs in the context of imbalanced data sets has been explored in detail in

[443, 470]. While these models are not designed with the use of example re-weighting, they achieve similar goals by using class-biased penalties during the SVM model creation.

2.2 Adaptive Re-sampling

In adaptive re-sampling, the different classes are differentially sampled in order to enhance the impact of the rare class on the classification model. Sampling can be performed either with or without replacement. Either the rare class can be oversampled, or the normal class can be under-sampled, or both. The classification model is learned on the re-sampled data. The sampling probabilities are typically chosen in proportion to their misclassification costs. This enhances the proportion of the rare costs in the sample used for learning. It has generally been observed [143], that under-sampling has a number of advantages over over-sampling. When under-sampling is used, the sampled training data is much smaller than the original data set. In some variations, all instances of the rare class are used in combination with a small sample of the normal class [106, 278]. This is also referred to as *one-sided selection*. Under-sampling also has the advantage of being efficient without losing too much information, because:

- The model construction phase for a smaller training data set requires much less time.
- The normal class is less important for modeling purposes, and most of the rare class is included for modeling. Therefore, the discarded instances do not take away too much from the modeling effectiveness.

2.2.1 Relation between weighting and sampling. Since cost-sensitive learning can be logically understood as methods which weigh examples differently, a question arises as how these methods relate to one another. Adaptive re-sampling methods can be understood as methods which sample the data in proportion to their weights, and then treat all examples equally. From a practical perspective, this may often lead to similar models in the two cases, though sampling methods may throw away some of the relevant data. It should also be evident that a direct weight-based technique retains more information about the data, and is therefore likely to be more accurate. This seems to be the case from many practical experiences with real data [102]. On the other hand, adaptive re-sampling has distinct *efficiency* advantages because it works with a much smaller data set. For example, for a data set containing 1% of labeled anomalies, it is possible for a re-sampling technique to work

effectively with 2% of the original data, when the data is re-sampled into an equal mixture of the normal and anomalous classes. This translates to a performance improvement of a factor of 50.

2.2.2 Synthetic Over-sampling: SMOTE. Over-sampling methods are also used in the literature, though less frequently so than under-sampling. One of the problems of over-sampling the minority class is that a larger number of samples with replacement leads to repeated samples of the same record. This could lead to over-fitting, and does not necessarily help the effectiveness of the classifier. In other to address this issue, it was suggested [103] that synthetic over-sampling could be used to create the over-sampled examples in a way which provides better effectiveness. The *SMOTE* approach works as follows. For each minority instance, its k nearest neighbors are found. Then, depending upon the level of over-sampling required, a fraction of them are chosen randomly. A synthetic data example is generated on the line segment connecting that minority example to its nearest neighbor. The exact position of the example is chosen uniformly at random along the line segment. The *SMOTE* algorithm has been shown to provide more robust over-sampling than a vanilla over-sampling approach. This approach forces the decision region of the re-sampled data to become more general than one in which only members from the rare classes in the *original* training data are over-sampled.

2.2.3 One Class Learning with Positive Class. It is possible to take adaptive re-sampling to its logical extreme by not including any examples of the normal class. This artificially transforms the problem to the semi-supervised scenario, though the nature of the semi-supervision is quite different from naturally occurring scenarios. In most natural forms of semi-supervision, the positive class is missing, and copious examples of the normal class may be available. Here the normal class examples are removed from the data. This problem is also different from the positive-unlabeled classification problem. Such a problem may sometimes occur naturally in scenarios where the background class is too diverse or noisy to be sampled in a meaningful way.

In such cases, unsupervised models can be constructed on the subset of the data corresponding to the positive class. The major difference is that *higher fit* of the data to the positive class corresponds to greater outlier scores. This is the reverse of what is normally performed in outlier detection. The assumption is that the representative data contains only anomalies, and therefore outliers are more likely to be similar to this data. Proximity-based classifiers are very natural to construct in the

one-class scenario, since the propensity of a test instance to belong to a class can be naturally modeled in terms of distances.

In the case of SVM classifiers, it is possible to create a two-class distribution by using the origin as one of the classes [396]. Typically, a kernel function is used in order to transform the data into a new space in which the dot product corresponds to the value of the kernel function. In such a case, an SVM classifier will naturally create a hyperplane which separates out the combination of features which describe the one class in the data. However, the strategy of using the origin as the second class in combination with a feature transformation is not necessarily generic and may not work well in all data domains. This differential behavior across different data sets has already been observed in the literature. In some cases, the performance of vanilla one-class SVM methods is quite poor, without careful changes to the model [382]. Other one-class methods for SVM classification are discussed in [250, 323, 382, 445].

2.2.4 Ensemble Techniques. A major challenge of under-sampling is the loss of the training data, which can have a detrimental effect on the quality of the classifier. A natural method to improve the quality of the prediction is to use ensemble techniques, in which the data instances are repeatedly classified with different samples, and then the majority vote is used for predictive purposes. In many of these methods, all instances from the rare class are used, but the majority class is under-sampled [106, 312]. Therefore, the advantages of selective sampling may be retained without a significant amount of information loss from the sampling process. In addition, a special kind of ensemble known as the *sequential ensemble* has also been proposed in [312]. In the sequential ensemble, the choice of the majority class instances picked in a given iteration depends upon the behavior of the classifier during previous iterations. Specifically, only majority instances which are correctly classified by the classifier in a given iteration are not included in future iterations. The idea is to reduce the redundancy in the learning process, and improve the overall robustness of the ensemble. Note that this is a *supervised* sequential ensemble, and is exactly analogous to the sequential ensemble method introduced in Chapter 1 for general-purpose outlier analysis.

2.3 Boosting Methods

Boosting methods are commonly used in classification in order to improve the classification performance on difficult instances of the data. The well known *Adaboost* algorithm [394] works by associating each training example with a *weight*, which is updated in each iteration,

depending upon the results of the classification in the last iteration. Specifically, instances which are misclassified, are given higher weights in successive iterations. The idea is to give higher weights to "difficult" instances which may lie on the decision boundaries of the classification process. The overall classification results are computed as a combination of the results from different rounds. In the tth round, the weight of the ith instance is $D_t(i)$. The algorithm starts off with equal weight of $1/N$ for each of the N instances, and updates them in each iteration. In practice, it is always assumed that the weights are normalized in order to sum to 1, though the approach will be described below in terms of (unscaled) relative weights for notational simplicity. In the event that the ith iteration is misclassified, then its (relative) weight is increased to $D_{t+1}(i) = D_t(i) \cdot e^{\alpha_t}$, whereas in the case of a correct classification, the weight is decreased to $D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t}$. Here α_t is chosen as the function $(1/2) \cdot \ln((1 - \epsilon_t)/\epsilon_t)$, where ϵ_t is the fraction of incorrectly predicted instances on a weighted basis. The final result for the classification of a test instance is a weighted prediction over the different rounds, where α_t is used as the weight for the tth iteration.

In the imbalanced and cost-sensitive scenario, the *AdaCost* method has been proposed [158], which can update the weights based on the cost of the instances. In this method, instead of updating the misclassified weights for instance i by the factor e^{α_t}, they are instead updated by $e^{\beta_-(c_i)\cdot\alpha_t}$, where c_i is the cost of the ith instance. Note that $\beta_-(c_i)$ is a function of the cost of the ith instance and serves as the "adjustment" factor, which accounts for the weights. For the case of correctly classified instances, the weights are updated by the factor $e^{-\beta_+(c_i)\cdot\alpha_t}$. Note that the adjustment factor is different depending upon whether the instance is correctly classified. This is because for the case of costly instances, it is desirable to increase weights more than less costly instances in case of misclassification. On the other hand, in cases of correct classification, it is desirable to reduce weights less for more costly instances. In either case, the adjustment is such that costly instances get relatively higher weight in later iterations. Therefore $\beta_-(c_i)$ is a non-decreasing function with cost, whereas $\beta_+(c_i)$ is a non-increasing function with cost. A different way to perform the adjustment would be to use the same exponential factor for weight updates as the original *Adaboost* algorithm, but this weight is further multiplied with the cost c_i [158], or other non-decreasing function of the cost. Such an approach would also provide higher weights to instances with larger costs. The use of boosting in weight updates has been shown to significantly improve the effectiveness of the imbalanced classification algorithms.

Boosting methods can also be combined with synthetic oversampling techniques. An example of this is the *SMOTEBoost* algorithm, which combines synthetic oversampling with a boosting approach. A number of interesting comparisons of boosting algorithms are presented in [246, 248]. In particular, an interesting observation in [248] is that the effectiveness of the boosting strategy is dependent upon the quality of the learner that it works with. When the boosting algorithm starts off with a weaker algorithm to begin with, the final (boosted) results are also not as good as those derived by boosting a stronger algorithm.

3. The Semi-Supervised Scenario: Positive and Unlabeled Data

In many data domains, the positive class may be easily identifiable, though examples of the negative class may be much harder to model simply because of their diversity and inexact modeling definition. Consider for example, a scenario where it is desirable to classify or collect all documents which belong to a rare class. In many scenarios, such as the case of web documents, the types of the documents available are too diverse, and it is hard to define a representative negative sample of documents from the web.

This leads to numerous challenges at the *data acquisition stage*, where it is unknown, what kinds of negative examples one might collect for contrast purposes. The problem is that the universe of instances in the negative class is rather large and diverse, and the collection of a representative sample may be difficult. For very large scale collections such as the web and social networks [493], this scenario is quite common. A number of methods are possible for negative data collection, none of which are completely satisfactory in terms of being *truly representative* of what one might encounter in a real application. For example, for web document classification, one simple option would be to simply crawl a random subset of documents off the web. Nevertheless, such a sample would contain contaminants which do belong to the positive class, and it may be hard to create a purely negative sample, unless a significant amount of effort is invested in creating a clean sample. The amount of human effort involved in human labeling in rare class scenarios is especially high because the vast majority of examples are negative, and a manual process of filtering out the positive examples would be too slow and tedious. Therefore, a simple solution is to use the sampled background collection as the unlabeled class for training, but this may contain positive contaminants. This could lead to two different levels of challenges:

- The contaminants in the negative class can reduce the effectiveness of a classifier, though it is still better to use the contaminated training examples rather than completely discard them.
- The collected training instances for the unlabeled class may not reflect the true distribution of documents. In such cases, the classification accuracy may actually be *harmed* by using the negative class [301].

A number of methods have been proposed in the literature for this variant of the classification problem, which can address the aforementioned issues.

While some methods in the literature treat this as a new problem which is distinct from the fully supervised classification problem [306], other methods [152] recognize this problem as a noisy variant of the classification problem, to which traditional classifiers can be applied with some modifications. An interesting and fundamental result proposed in [152] is that the accuracy of a classifier trained on this scenario differs by only a constant factor from the true conditional probabilities of being positive. The underlying assumption is that the labeled examples in the positive class are picked randomly from the positive examples in the combination of the two classes. These results provides strong support for the view that learning from positive and unlabeled examples is essentially equivalent to learning from positive and negative examples.

There are two broad classes of methods which can be used in order to address this problem. In the first class of methods, heuristics are used in order to identify training examples which are negative. Subsequently, a classifier is trained on the positive examples, together with the examples, which have already been identified to be negative. A less common approach is to assign weights to the unlabeled training examples [293, 306]. The second case is a special one of the first, in which each weight is chosen to be binary. It has been shown in the literature [307], that the second approach is superior. An SVM approach is used in order to learn the weights. The work in [507] uses the weight vector in order to provide robust classification estimates.

3.1 Difficult Cases and One-Class Learning

While the use of the unlabeled class provides some advantage to classification in most cases, this is not always true. In some scenarios, the unlabeled class in the training data reflects the behavior of the negative class in the test data very poorly. In such cases, it has been shown, that the use of the negative class actually *degrades* the effectiveness of classifiers. In such cases, it has been shown in [301] that the use of one-

class learners provides more effective results than the use of a standard classifier. Thus, in such situations, it may be better to simply discard the training class examples, which do not truly reflect the behavior of the test data. Most of the one-class SVM classifiers discussed in the previous section can be used in this scenario.

4. The Semi-Supervised Scenario: Novel Class Detection

The previous section discussed cases, where it is difficult to obtain a clean sample of normal data, when the background data is too diverse or contaminated. A more common situation in the context of outlier detection is one in which no training data is available about one or more of the anomalous classes. Such situations can arise, when the anomalous class is so rare that it may be difficult to collect concrete examples of its occurrence, even when it is recognized as a concrete possibility. Some examples of such scenarios are as follows:

- In a bio-terrorist attack application, it may be easy to collect normal examples of environmental variables, but no explicit examples of anomalies may be available, if an attack has never occurred.

- In an intrusion or viral attack scenario, many examples of normal data and previous intrusions or attacks may be available, but new forms of intrusion may arise over time.

This is truly a semi-supervised version of the problem, since training data is available about some portions of the data, but not others. Therefore, such scenarios are best addressed with a combination of supervised and unsupervised techniques. It is also important to distinguish this problem from one-class classification, in which instances of the *positive class* are available. In the one-class classification problem, it is desirable to determine other examples, which are as *similar* as possible to the training data, whereas in the novel class problem, it is desirable to determine examples, which are as *different* as possible from the training data.

In cases, where only examples of the normal class are available, the only difference from the unsupervised scenario is that the training data is guaranteed to be free of outliers. The specification of normal portions of the data makes the determination of further outliers easier, because this data can be used in order to construct a model of what the normal data looks like. Another distinction between unsupervised outlier detection and one-class novelty detection, is that novelties are often defined in a temporal context, and eventually become a normal part of the data.

4.1 One Class Novelty Detection

Since the novel-class detection problem is closely related to the one-class problem in cases where only the normal class is specified, it is natural to question whether it is possible to adapt some of the one-class detection algorithms to this scenario. The major difference in this case is that it is desirable to determine classes which are as *different* as possible from the specified training class. This is a more difficult problem, because a data point may be different from the training class in several ways. If the training model is not exhaustive in describing the corresponding class, it is easy for mistakes to occur.

For example, nearest neighbor models are easy to adapt to the one class scenario. In the one-class models discussed in the previous section, it is desirable to determine data points which are as *close* as possible to the training data. In this case, the opposite is desired, where it is desirable to determine data points which are as *different* as possible from the specified training data. This is of course no different from the unsupervised methods for creating proximity-based outlier detection methods. *In fact, any of the unsupervised models for outlier detection can be used in this case.* The major difference is that the training data is guaranteed to contain only the normal class, and therefore the outlier analysis methods are likely to be more robust. Strictly speaking, when only examples of the normal class are available, the problem is hard to distinguish from the unsupervised version of the problem, at least from a methodological point of view. From a *formulation* point of view, the training and test records are not distinguished from one another in the unsupervised case (any record can be normal or an anomaly), whereas the training (only normal) and test records (either normal or anomaly) are distinguished from one another in the semi-supervised case.

One class SVM methods have also been adapted to novelty detection [397]. The main difference from *positive example training-based* one-class detection is that the class of interest lies on the *opposite* side of the separator as the training data. Some of the one-class methods such as SVMs are unlikely to work quite as well in this case. This is because a one-class SVM may really only be able to model the class present in the training data (the normal class) well, and may not easily be able to design the best separator for the class which is most *different* from the normal class. Typically, one-class SVMs use a kernel-based transformation along with reference points such as the origin in order to determine a synthetic reference point for the other class, so that a separator can be defined. If the transformation and the reference point is not chosen properly, the one-class SVM is unlikely to provide robust results in terms of identify-

ing the outlier. One issue with the one-class SVM is that the anomalous points (of interest) and the training data now need to lie on *opposite* sides of the separator. This is a more difficult case than one in which the anomalous points (of interest) and the training data need to lie on the same side of the separator (as was discussed in a previous section on positive-only SVMs). The key difference here is that the examples of interest are not available on the *interesting* side of the separator, which is poorly modeled.

It has been pointed out that the use of the origin as a prior for the anomalous class [91] can lead to incorrect results, since the precise nature of the anomaly is unknown a-priori. Therefore, the work in [91] attempts to determine a linear or non-linear decision surface which wrap around the surfaces of the normal class. Points which lie outside this decision surface are anomalies. It is important to note that this model essentially uses an indirect approach such as SVM to model the dense regions in the data. Virtually all unsupervised outlier detection methods attempt to model the normal behavior of the data, and can be used for novel class detection, especially when the only class in the training data is the normal class. *Therefore the distinction between normal-class only variations of the novel class detection problem and the unsupervised version of the problem are limited and artificial, especially when other labeled anomalous classes do not form a part of the training data.* Numerous analogues of unsupervised methods have also been developed for novelty detection, such as extreme value methods [383], direct density ratio estimation [214], and kernel-based PCA methods [220]. This is not surprising, given that the two problems are different only at a rather superficial level. In spite of this, the semi-supervised version of the (normal-class only) problem seems to have a distinct literature of its own. This is somewhat unnecessary, since any of the unsupervised algorithms can be applied to this case. The main difference is that the training and test data are distinguished from one another, and the outlier score is computed for a test instance with respect to the training data. Novelty detection can be better distinguished from the unsupervised case in temporal scenarios, where novelties are defined *continuously* based on the past behavior of the data. This will be discussed in more detail in Chapter 8 on temporal outlier detection, though a brief introduction is provided in the following subsections.

4.2 Combining Novel Class Detection with Rare Class Detection

A more challenging scenario arises, when labeled rare classes are present in the training data, but novel classes may also need to be detected. Such scenarios can arise quite often in many applications such as intrusion detection, where partial knowledge is available about *some* of the anomalies, but others may need to be modeled in an unsupervised way. Furthermore, it is important to distinguish different kinds of anomalies from one another, whether they are found in a supervised or unsupervised way. The labeled rare classes already provides important information about *some* of the outliers in the data. This can be used to determine different kinds of outliers in the underlying data, and distinguish them from one another. This is important in applications, where it is not only desirable to determine outliers, but also obtain an understanding of the kind of outlier which is discovered. The main challenge in these methods is to seamlessly combine unsupervised outlier detection methods with fully supervised rare class detection methods. For a given test data point two decisions need to be made, in the following order:

1. Is the test point a natural fit for a model of the training data? This model also includes the currently occurring rare classes. A variety of unsupervised models such as clustering can be used for thus purpose. If not, it is immediately flagged as an outlier, or a novelty.

2. If the test point is a fit for the training data, then a classifier model is used to determine whether it belongs to one of the rare classes. Any cost-sensitive model (or an ensemble of them) can be used for this purpose.

Thus, this model requires a combination of unsupervised and supervised methods in order to determine the outliers in the data. This situation arises more commonly in online and streaming scenarios, which will be discussed in the next section.

4.3 Online Novelty Detection

The most common scenario for novel class detection occurs in the context of *online* scenarios in concept drifting data streams. In fact, novel class detection usually has an implicit assumption of *temporal* data, since classes can be defined as novel only in terms of what has already been seen in the *past*. In many of the batch-algorithms discussed above, this temporal aspect is not fully explored, since a single snapshot of

training data is assumed. Many applications such as intrusion detection are naturally focussed on a streaming scenario. In such cases, novel classes may appear at any point in the data stream, and it may be desirable to distinguish different kinds of novel classes from one another [328, 329, 36]. Furthermore, when new classes are discovered, these kinds of anomalies may recur over time, albeit quite rarely. In such cases, the effectiveness of the model can be improved by keeping a memory of the *rarely recurring classes*. This case is particularly challenging because aside from the temporal aspects of modeling, it is desirable to perform the training and testing in an online manner, in which only one pass is allowed over the incoming data stream. This scenario is a true amalgamation of supervised and unsupervised methods for anomaly detection, and is discussed in detail in section 4.3 of Chapter 8.

In the streaming scenario containing only unlabeled data, unsupervised clustering methods [25, 26] can be used in order to identify significant novelties in the stream. In these methods, *novelties occur as emerging clusters in the data, which eventually become a part of the normal clustering structure of the data*. Both the methods in [25, 26] have statistical tests to identify, when a newly incoming instance in the stream should be considered a novelty. Thus, the output of these methods provides an understanding of the natural complementary relationship between the clusters (normal unsupervised models) and novelties (temporal abnormalities) in the underlying data. This issue will be discussed in some more detail in Chapter 8 on temporal outlier detection.

5. Human Supervision

A natural form of supervision in outlier detection is one in which a human expert may intervene in the outlier detection process in order to further improve the effectiveness of the underlying algorithms. One of the major challenges in outlier detection is that the anomalies found by an algorithm which is either purely unsupervised or only partially supervised may not be very useful. This is because unsupervised algorithms (or even supervised methods with a small amount of training data) may not be able to effectively distinguish between *useless noise* and *useful outliers*. In such cases, it may be valuable to add human supervision to outlier analysis in order to detect more meaningful outliers. The incorporation of human supervision can augment the limited knowledge of outlier analysis algorithms. Specifically, the augmentation may be done in several ways:

- An unsupervised or supervised outlier detection algorithm may present pre-filtered results to a user, and the user can provide

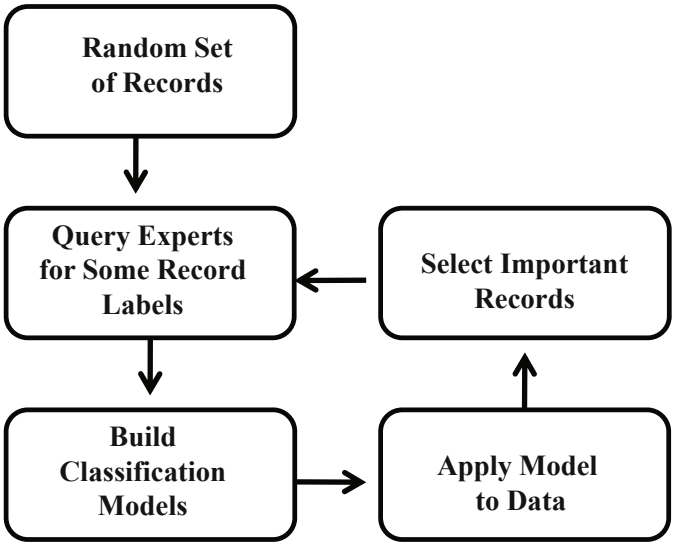

Figure 6.2. The overall procedure for active learning

their feedback on this small number of pre-filtered examples. This process would not be possible to perform manually on the original data set, which may large, and in which the vast majority of examples are normal [360].

- The user-provided examples can be combined with the results from an unsupervised algorithm to learn which outliers determined by the unsupervised algorithm are relevant. The combined results can then be used in order to train a traditional rare class detection model, as discussed earlier in this chapter. For example, an SVM approach was used in [512].

Each of the aforementioned methodologies are discussed in detail below.

5.1 Active Learning

An interesting procedure for active learning from unlabeled data is proposed in [360]. An iterative procedure is used in order to label some of the examples in each iteration. In each iteration, a number of interesting instances are identified, for which the addition of labels would be helpful for further classification. These are considered the "important" instances. The human expert provides labels for these examples. These are then used in order to classify the data set with the augmented labels.

The first iteration is special, in which a purely unsupervised approach is used for learning. These procedure is performed iteratively until the addition of further examples is no longer deemed helpful for further classification. The overall procedure is illustrated in Figure 6.2. It should be noted that this approach can also be used in scenarios in which a small number of positive examples are available to begin with.

A key question arises as to *which* examples should be presented to the user for the purposes of labeling. It is clear that examples which are very obviously positive or negative (based on current models) are not particularly useful to present to the user. Rather, it is the examples with the greatest *uncertainty* or *ambiguity*, which should be presented to the user in order to gain the greatest knowledge about the decision boundaries between the different classes. It is expected that the selected examples should lie on the decision boundaries, in order to maximize the learning of the contours separating different classes, with the use of least amount of expert supervision, which can be expensive in many scenarios.

A common approach to achieve this goal in active learning is the principle of *query by committee* [400]. . In these methods, an ensemble of classifiers is learned, and the greatest *disagreement* among them is used to select data points which lie on the decision boundary. A variety of such criteria based on ensemble learning are discussed in [331]. It is also possible to use the model characteristics directly in order to select such points. For example, two primary criteria which can be used for selection, are as follows [360]:

- *Low Likelihood:* These are data points which have low fit to the model describing the data. For example, if an EM algorithm is used for modeling, then these are points which have low fit to the underlying model.

- *High Uncertainty:* These are points which have the greatest uncertainty in terms of the component of the model to which they belong. In other words, in an EM model, such a data point would show relatively even soft probabilities for different components of the mixture.

All data points are ranked on the basis of the two aforementioned criteria. The lists are merged by alternating between them, and adding the next point in the list, which has not already been added to the merged list. Details of other relevant methods such as *interleaving* are discussed in [360].

5.2 Outlier by Example

The outlier by example method follows the same principle of learning from (user-provided) positive and unlabeled examples, which was discussed earlier in this chapter, except for the difference that an unsupervised approach is utilize to perform feature transformations on the examples, and augment the user provided examples by comparing the deviations of the objects with those of the user-provided examples. The algorithm proceeds in the following steps:

- *Feature Extraction:* In this step, all the objects are transformed to their MDEF-based representations as discussed in the section on the LOCI method in Chapter 4. This is done by using the LOCI method discussed in Chapter 4, except that different *sampling neighborhoods* are used in order to create a vector of deviations for different sampling neighborhoods. Thus, this approach transforms the objects into a vector representation of MDEF values.

- *Example Augmentation:* A major challenge with all supervised learning methods is the paucity of training examples for effective training. Therefore, the user-provided examples are augmented in order to increase the number of positive examples. Two kinds of outliers are added. The first kind are outliers for which any component of the MDEF vector is greater than a user-specified threshold. These are referred to as *outstanding outliers*. The second kind of examples are *artificially generated* from the user specified outliers, by creating MDEF values which lie between their current maximum MDEF value and the threshold K. Depending upon the number of outliers which need to be generated, equally spaced intervals between the MDEF value and the threshold are generated. Note that the representation of the data is still in the form of MDEF vectors, and the artificially generated data is also represented in this form.

- *Final Classification:* The augmented training data is used to learn an SVM classifier, which distinguishes the unlabeled examples from the positive examples.

An interesting observation about the technique above is that the additional labeling and augmentation is done with the use of *automated* techniques. This is different from the method of [360] in which labeling is done by human experts. In both methods, human experts and automated methods are involved, but in different parts of the process.

6. Conclusions and Summary

This chapter discusses the problem of supervised outlier analysis. In many real scenarios, training data is available, which can be used in order to greatly enhance the effectiveness of the outlier detection process. Many of the standard classification algorithms in the literature can be adapted to this problem, especially when full supervision is available. The major challenge of using the standard classification algorithms is that they may not work very well in scenarios where the distribution of classes is imbalanced. In order to address this issue, sampling and re-weighting can be used quite effectively.

The partially supervised variations of the problem are diverse. Some of these methods do not provide any labels on the normal class. This corresponds to the fact that the normal class may be contaminated with an unknown number of outlier examples. Furthermore, in some cases, the distribution of the normal class may be very different in the training and test data. One-class methods can sometimes be effective in addressing such issues.

Another form of partial supervision is the identification of novel classes in the training data. Novel classes correspond to scenarios in which the labels for some of the classes are completely missing from the training data. In such cases, a combination of unsupervised and supervised methods need to be used for the detection process. In cases where examples of a single normal class are available, the scenario becomes almost equivalent to the unsupervised version of the problem.

Supervised methods are closely related to active learning in which human experts may intervene in order to add more knowledge to the outlier detection process. Such combinations of automated filtering with human interaction can provide insightful results. The use of human intervention sometimes provides the more insightful results, because the human is involved in the entire process of label acquisition and final outlier detection.

7. Bibliographic Survey

Supervision can be incorporated in a variety of ways, starting from partial supervision to complete supervision. In the case of complete supervision, the main challenges arise in the context of class imbalance and cost-sensitive learning [102, 105, 151]. The issue of evaluation is critical in cost-sensitive learning because of the inability to model the effectiveness with measures such as the absolute accuracy. Methods for interpreting ROC curves and classification accuracy in the presence of costs and class imbalance are discussed in [144, 159, 249, 376, 377]. The

impact of class imbalance is relevant even for feature selection [335, 511], because it is more desirable to select features which are more indicative of the rare class.

A variety of general methods have been proposed for cost-sensitive learning such as *MetaCost* [145], weighting [496], and sampling [106, 102, 143, 278, 496]. Weighting methods are generally quite effective, but may sometimes be unnecessarily inefficient, when most of the training data corresponds to the background distribution. In this context, sampling methods can significantly improve the efficiency. Numerous cost-sensitive variations of different classifiers have been proposed along the lines of weighting, and include the Bayes classifier [496], nearest neighbor classifier [506], decision trees [450, 463], rule-based classifiers [245, 247] and SVM classifiers [443, 470].

Ensemble methods for improving the robustness of sampling are proposed in [106, 312]. Since the under-sampling process reduces the number of negative examples, it is natural to use an ensemble of classifiers which combine the results of classifiers trained on different samples. This provides more robust results, and ameliorates the instability which arises from under-sampling. The major problem in over-sampling of the minority class is the over-fitting obtained by re-sampling duplicate instances. Therefore, a method known as *SMOTE* creates synthetic data instances in the neighborhood of the rare instances [103].

The earliest work on boosting rare classes was proposed in [252]. This technique is designed for imbalanced data sets, and the intuition is to boost the positive training instances (rare classes) faster than the negatives. Thus, it increases the weight of false negatives more the false positives. However, it is not cost-sensitive, and it also decreases the weight of true positives more than true negatives, which is not desirable. The *AdaCost* algorithm proposed in this chapter was proposed in [158]. Boosting techniques can also be combined with sampling methods, as in the case of the *SMOTEBoost* algorithm [104]. An evaluation of boosting algorithms for rare class detection is provided in [246]. Two new algorithms for boosting are also proposed in the same paper. The effect of the base learner on the final results of the boosting algorithm are investigated in [248]. It has been shown that the final result from the boosted algorithm is highly dependent on the quality of the base learner.

A particular case which is commonly encountered is one in which the instances of the positive class are specified, whereas the other class is unlabeled [152, 301, 293, 306, 307, 493, 507]. Since the unlabeled class is pre-dominantly a negative class with contaminants, it is essentially equivalent to a fully supervised problem, with some loss in accu-

racy which can be quantified [152]. In some cases, when the collection mechanisms for the negative class are not reflective of what would be encountered in test instances, the use of such instances may harm the performance of the classifier. In such cases, it may be desirable to discard the negative class entirely and treat the problem as a one-class problem [301]. However, as long as the training and test distributions are not too different, it is generally desirable to also use the instances from the negative class.

The one-class version of the problem is an extreme variation in which only positive instances of the class are used for training purpose. SVM methods are particularly popular for one-class classification [250, 323, 382, 396, 445]. Methods for one-class SVM methods for scene classification are proposed in [480]. It has been shown that the SVM method is particularly sensitive to the data set used [382].

An important class of semi-supervised algorithms is known as *novelty detection*, in which no training data is available about some of the anomalous classes. This is common in many scenarios such as intrusion detection, in which the patterns in the data may change over time, and may therefore lead to novel anomalies (or intrusions). These problems are combination of the supervised and unsupervised case, and numerous methods have been designed for the streaming scenario [328, 329, 36]. The special case, where only the normal class is available is not very different from the unsupervised scenario, other than the fact that it may have an underlying temporal component. Numerous methods have been designed for this case such as single-class SVMs [397, 91], minimax probability machines [282], kernel-based PCA methods [383], direct density ratio estimation [214], and extreme value analysis [220]. Single class novelty detection has also been studied extensively in the context of the first story detection in text streams [515], and will be discussed in detail in Chapter 7. The methods for the text streaming scenario are most highly unsupervised, and use standard clustering or nearest neighbor models. In fact, a variety of stream clustering methods [25, 26] discover newly forming clusters (or emerging novelties) as part of their output of the overall clustering process. A detailed survey of novelty detection methods may be found in [325, 326].

Human supervision is a natural goal in anomaly detection, since most of the anomalous instances are not interesting, and it is only by incorporating user feedback that the interesting examples can be separated from noisy anomalies. Methods for augmenting user-specified examples with automated methods are discussed in [512, 513]. These methods also add artificially generated examples to the training data, in order to increase the number of positive examples for the learning process. Other

methods are designed for *selectively* presenting examples to a user, so that only the relevant ones are labeled [360]. A nearest-neighbor method for active learning is proposed in [207]. The effectiveness of active learning methods for selecting good examples to present to the user is critical in ensuring minimal human effort. Such points should lie on the decision boundaries separating two classes [121]. Methods which use query by committee to select such points with ensembles are discussed in [331, 400]. A selective sampling method which uses active learning in the context of outlier detection is proposed in [1]. A method has also been proposed in [309] as to how unsupervised outlier detection algorithms can be leveraged in conjunction with limited human effort in order to create a labeled training data set.

8. Exercises

1. Download the *Arrythmia* data set from the *UCI Machine Learning Repository*.

- Implement a 20-nearest neighbor classifier which classifies the majority class as the primary label. Use a $3:1$ ratio of costs between the normal class, and any other minority cost. Determine the overall accuracy and the cost-weighted accuracy.
- Implement the same algorithm as above, except that each data point is given a weight, which is proportional to its cost. Determine the overall accuracy and the cost-weighted accuracy.

2. Repeat the exercise above for the quantitative attributes of the *KDD CUP 1999 Network Intrusion* data set of the *UCI Machine Learning Repository*.

3. Repeat each of the exercises above with the use of the *MetaCost* classifier, in which 100 different bagged executions are utilized in order to estimate the probability of relabeling. An unweighted 10-nearest neighbor classifier is used as the base learner. For each bagged execution, use a 50% sample of the data set. Determine the overall accuracy and the cost-weighted accuracy.

4. Repeat Exercises 1 and 2 by sampling one-thirds the examples from the normal class, and including all examples from the other classes. An unweighted 20-nearest neighbor classifier is used as the base learner. Determine the overall accuracy and the cost-weighted accuracy.

5. Repeat Exercise 4, by using an ensemble of five classifiers, and using the majority vote.

6. Repeat Exercises 1 and 2 with the use of cost-sensitive boosting. An unweighted 10-nearest neighbor classifier is used as the base learner.