Chapter 7

Supervised Outlier Detection

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

7.1 Introduction

The discussions in the previous chapters focus 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 or other uninteresting phenomena. It has been observed [338, 374, 531] in diverse applications such as system anomaly detection, financial fraud, and Web robot detection that *interesting anomalies are often highly specific to particular types of abnormal activity in the underlying application*. In such cases, an unsupervised outlier detection method might discover noise, which is not specific to that activity, and therefore may not be of interest to an analyst. In many cases, different types of abnormal instances could be present, and it may be desirable to distinguish among them. For example, in an intrusion-detection scenario, different types of intrusion anomalies are possible, and the specific type of an intrusion is important information.

The goal of supervised outlier detection is to empower learning methods with application-specific knowledge so as to obtain application-relevant anomalies. This knowledge often includes examples of such relevant anomalies, although other types of supervision are also possible. Because of the rare nature of anomalies, such examples are often limited. This causes challenges in creating robust models. Nevertheless, even when a small amount of data is available for supervision, its incorporation usually improves outlier-detection accuracy in a significant way. *The general recommendation for outlier analysis is to always use supervision where possible.*

The supervision is provided by examples of normal or abnormal data. This is referred to as *training data*, and it can be used to create a *classification model* that distinguishes between normal and anomalous instances. The problem of classification has been widely studied in its own right, and numerous algorithms are available in the literature [33, 176] for creating supervised models from training data.

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_7

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. This is because the problem is associated with several challenging characteristics, 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 natural optimization function for classification (which is accuracy) is changed to *cost-sensitive* accuracy.

- **Contaminated normal class examples (Positive-unlabeled class problem):** In many real scenarios, only the positive class is labeled, and the remaining "normal" data contains some abnormalities. This is natural in large-scale settings like 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. Technically, this case can be treated as a difficult special case of full supervision, in which the normal class is noisy and contaminated. Off-the-shelf classifiers can be used on the positive-unlabeled version of the classification problem, as long as the relative frequency of contaminants is not extreme.

- **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 types of intrusions emerge 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 examples of only the normal class are available. 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. No changes are required to existing algorithms, other than properly distinguishing between the training and test data.

It is possible for these scenarios to be present in combination, and the boundaries between them are often blurred. The goal of this chapter is to elucidate the required *modifications* to classification methods to address these different settings. Therefore, a working knowledge of classification methods is assumed [176].

All classification problems, including rare-class problems, are heavily dependent on the feature representation used for the learning process. For example, kernel methods are often used in order to make nonlinearly separable classes linearly separable via an implicit transformation. However, in *feature engineering*, this transformation is performed explicitly with an understanding of the domain characteristics of the problem at hand. Rare-class learning

is one such domain in which the output scores of outlier detection algorithms can be used as engineered features for more effective learning. Therefore, the use of unsupervised outlier detection algorithms will be explored for feature engineering in supervised problems.

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, which can lead to poor results. In order to address this problem, *active learning* is used to label training examples in a guided way. This is achieved by providing an expert with pre-filtered candidate points for labeling. Such labeling can sometimes be accomplished with infrastructures like *Amazon Mechanical Turk*, and it comes at a per-instance cost of labeling. Therefore, it is important to present judiciously chosen examples to the expert so that the decision boundary between the rare and normal class is learned with as few examples as possible. In such cases, label acquisition is combined with model construction in order to progressively incorporate more expert knowledge into the outlier analysis process.

Although most of this chapter will focus on settings in which ground-truth (supervision) is already available, this chapter will also investigate the connections between *unsupervised* outlier detection and supervised regression modeling. An interesting approach, which was proposed recently [417, 429], shows how one can use repeated applications of *off-the-shelf* regression models for unsupervised outlier detection. The basic idea is to use regression modeling to predict each of the attributes from the remaining attributes and then combining the errors of these models to create an outlier score (see section 7.7). This approach has the merit that it opens the door to the use of hundreds of off-the-shelf regression models for effective *unsupervised* outlier detection.

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 7.3. Section 7.4 discusses the case in which only a subset of the classes are observed whether they are rare or normal. Unsupervised feature engineering methods for rare-class detection are discussed in section 7.5. Active learning methods are discussed in section 7.6. Section 7.7 shows how one can use repeated applications of off-the-shelf regression models for unsupervised outlier detection. The conclusions and summary are presented in section 7.8.

7.2 Full Supervision: 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 that are not cognizant of this class imbalance might 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 the remaining 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. However, this (rather useless) classifier is often hard to outperform from an *absolute* accuracy point of view. In many cases, apparently "reasonable" algorithms also degrade to this trivial classifier in terms of performance. 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 modifying the prediction criterion, such as reporting a non-majority anomalous class as the relevant label of a test instance, it is possible to improve the classification accuracy of anomalous classes at the expense of the accuracy on the normal class. Because of the predominance of the normal class in accuracy computation, such a classifier will almost always have lower accuracy then the aforementioned trivial classifier; yet, it is more *useful* from an application-specific perspective. Therefore, the question arises whether the use of measures such as overall classification accuracy is meaningful in the first place. It is important to understand that the issue of *evaluation* and *model construction* are closely related in all learning problems. In order to design useful rare-class detection models, one must first design meaningful evaluation mechanisms from an application-specific perspective. In the rare-class setting, the use of absolute accuracy is not a meaningful evaluation mechanism.

The proper evaluation mechanisms in these settings weigh the errors of anomalous instances differently from those of normal instances. The basic assumption is that it is more costly to misclassify anomalous instances as compared to normal instances. For example, misclassifying a fraudulent transaction (which could lead to millions of dollars in losses) is more costly than misclassifying a normal transaction (which causes annoyance to the end-user being incorrectly warned). Therefore, the model should optimize a cost-weighted variant of the accuracy. The underlying algorithms are also changed to incorporate these modified modeling assumptions. There are two primary types of algorithms that are popular in class-imbalanced settings:

- **Cost-sensitive learning:** The objective function of the classification algorithm is modified in order to weight the classification errors in a differential way for the normal classes and the rare classes. The typical assumption is that the misclassification of a rare class incurs higher costs. In many cases, this change in the objective function requires only minor changes to existing classification models.

- **Adaptive re-sampling:** The data are 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. After all, the presence of a larger *relative* number of instances of a particular class in the sample (with respect to original data) tends to bias the prediction algorithms in favor of that class.

Both these methodologies will be discussed. A working knowledge of classification methods is required to understand the material in this chapter. The reader is also referred to [176] 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 label indices are denoted by $L = \{1, \ldots k\}$. Without loss of generality, it can be assumed that the normal class takes on the label-index of 1 in the set L and the remaining classes, which are presumably rare, are indexed from 2 to k. 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 is denoted by N_i. The total number of instances in the data is denoted by N, and therefore we have $\sum_{i=1}^{k} N_i = N$. The class imbalance assumption implies that $N_1 \gg N - N_1$. It is also possible for the various anomalous classes to be imbalanced with respect to one another. For example, in a network intrusion-detection application, the intrusions of various types may have widely varying frequencies.

7.2.1 Cost-Sensitive Learning

In cost-sensitive learning, the goal is to learn a classifier that maximizes the weighted accuracy over the different classes. The *misclassification cost* of the ith class is denoted by c_i. Some models [175] use a $k \times k$ cost matrix to represent all the pair-wise values of misclassification costs (i.e., cost of misclassifying class i to class j). 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. In this chapter, we consider a simpler model in which the misclassification cost depends only on the origin class to which the instance belongs. The destination class is not relevant to the cost. Such a simplified model is more relevant to the rare-class detection problem. Therefore, we can denote the misclassification cost of the ith class by c_i. The goal of the classifier is to learn a training model that minimizes the *weighted misclassification rate*. Therefore, if $n_i \leq N_i$ is the number of examples misclassified for the ith class, the goal is to minimize the following objective function:

$$J = \sum_{i=1}^{k} c_i \cdot n_i \tag{7.1}$$

The only difference from the traditional classification accuracy measure is the use of the weights $c_1 \ldots c_k$ in the objective function.

The choice of c_i is regulated by application-specific requirements, and is therefore a part of the input. However, in the absence of this input, some natural assumptions are sometimes used. 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, although more principled methods also exist in the literature. For example, the work in [601] proposes methods to learn the costs directly in a data-driven manner.

7.2.1.1 MetaCost: A Relabeling Approach

A general framework known as *MetaCost* [175] uses a *relabeling* approach to classification. This is a *meta-algorithm*, which can be wrapped around any existing 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 that 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. Consider a setting in which 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 class index* 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 that 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 on

the specific classifier at hand. While some classifiers explicitly provide a probability score, this is not true in general. The work in [175] proposes a bagging approach [98] in which the training data is sampled with replacement (bootstrapping), and a model is repeatedly constructed on this basis. The size of the bootstrapped sample might be smaller than the training data to improve efficiency. These models are used to classify the training instances repeatedly. The fraction of predictions (or votes) for each class across different training data samples is used as its classification probability.

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 prediction is *highly correlated* with 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. For example, an unmodified Bayes classifier will usually assign a lower probability to the rare class because of the much lower a priori probability of the rare class. Consider a situation in which a Bayes classifier assigns a posterior probability of 30% to a rare-class instance and the prior probability of the rare class is only 1%. In spite of increasing the prior probability by a factor of 30, 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 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 [175] 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 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*. This problem is caused by the fact that bagging is an extremely imperfect simulation of sampling from a base distribution; it works in some restricted settings like ensemble learning but one cannot generalize this principle to arbitrary applications which are heavily susceptible to the correlations in predictions across different bags.

[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. This example also suggests that it is extremely important to use very unstable classifiers with this approach. An example would be to use a 1-nearest neighbor classifier.

[2]The original idea of bagging was not designed to yield class probabilities [98]. 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.

In general, it is possible for some classification algorithms to work effectively with meta-cost under the following conditions:

1. It is extremely important to use an unstable algorithm as the base classifier. Stable algorithms will lead to probability estimates that are close to 0 or 1.

2. Even though the approach in [175] proposes the use of smaller bootstrapped samples solely for efficiency considerations, an unobserved side-effect is that it will reduce overlap among different samples. Reducing overlap among samples is helpful for reducing correlation among classifiers, which is likely to improve the probability estimation. Smaller samples will also lead to unstable classifiers.

Note, however, that excessive instability can also have detrimental effects on accuracy.

It is also possible to use soft estimates of classification probability. In the previous example of the nearest neighbor classifier, 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.

7.2.1.2 Weighting Methods

Most classification algorithms can be modified in natural ways to account for costs. 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 modification is directly motivated by the presence of costs in the accuracy objective of Equation 7.1 that needs to be optimized. This leads to a number of simple modifications to the underlying classification algorithms. 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. In other words, the prior probability in the unweighted case needs to be multiplied with the cost. Note that this criterion is also used in *MetaCost*, although the latter uses this criterion for *relabeling* training instances rather than predicting test instances. When good probability estimates are available from the Bayes classifier, it makes more sense to use them to directly predict test instances rather than to relabel training instances.

Proximity-Based Classifiers

In nearest-neighbor classifiers with binary classes, the classification label of a test instance is defined as 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. The majority class is therefore selected by reporting the class with the maximum *weight* as the relevant one. Because of the weighting,

a test instance may be classified as an anomaly, even when fewer examples of the rare class are present in its neighborhood. A discussion of methods for k-nearest neighbor classification in the context of data classification may be found in [609].

Rule-Based Classifiers

Many rule-based classifiers leverage association rule mining with support and confidence parameters. A rule relates a condition in the data (e.g., ranges on numeric attributes) to a class label. The condition occurs on the left-hand side (antecedent) of the rule, whereas the class label occurs on the right-hand side (consequent) of the rule. The support of a rule is defined as the number of training instances that 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. The data is first discretized, and all relevant rules are mined from it 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 identified as the rules whose antecedents match the test instance. These rules might sometimes predict conflicting class labels. Therefore, the results from different rules can be combined using a variety of heuristics (e.g., majority class from relevant rules or top-matching rule) 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 support and confidence of a rule will be much higher for rules with the rare class in the consequent. This will result in the selective emphasis of rules corresponding to prediction of rare instances. Some methods for using rule-based methods in imbalanced data classification are proposed in [298, 300].

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 denoted by $1 - \sum_{i=1}^{k} p_i^2$. Better separations of different classes lead to lower Gini index. Typically, the data is first discretized and then converted into a binary representation. The split attribute is the one that minimizes the Gini index of the children nodes. By using costs as weights for the instances, the values of the fractions $p_1 \ldots p_k$ are computed. This will also impact the computation of the Gini index, so as to selectively create nodes in which a higher proportion of data points belong to the rare class. This type of approach generally tends to lead to better separation between the normal class and the rare class. In cases where leaf nodes do not exclusively belong to a particular class, instances are weighted by their misclassification costs for labeling that leaf node. Some examples of cost-sensitive decision trees are discussed in [546, 566].

Support-Vector Machines

Support-vector machine (SVM) classifiers work by learning hyperplanes that optimally separate the two classes in order to minimize a carefully defined penalty function. This penalty

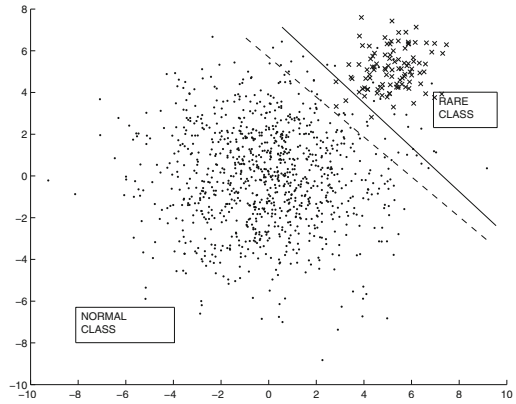


Figure 7.1: Optimal hyperplanes will change because of weighting of examples

function penalizes training data points for being on the wrong side of the decision boundary and for even being close to the decision boundary (on the correct side). The notion of "closeness" to the decision boundary is defined by the *margin*, which is simultaneously maximized. Thus, an SVM classifier can be modeled as an optimization problem, in which the goal is to learn the coefficients of the underlying hyperplane in order to optimize a weighted combination of these two objectives. For example, a two-class example has been illustrated in Figure 7.1. 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. Therefore, violation of the linear margin constraints incurs a cost of 1 for the normal class, whereas it incurs a cost of c_2/c_1 for the rare class. Since it is assumed that $c_2 \gg c_1$, this differential weighting 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 7.1, 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 [535, 570]. These methods use class-biased penalties during the SVM model creation. At a basic level, the linear decision boundary of the SVM may be written as follows:

$$\overline{W} \cdot \overline{X} + b = 0 \tag{7.2}$$

Here, the coefficient vector $\overline{W}$ and the bias b defines the hyperplane. Normal data points satisfy $\overline{W} \cdot \overline{X} + b \geq 0$ and anomalous data points satisfy $\overline{W} \cdot \overline{X} + b \leq 0$. Therefore, we punish the violation of these constraints. In the cost-sensitive setting, the normal points are punished by 1 unit, whereas the violations on rare-class instances are punished by $\frac{c_2}{c_1} > 1$. Note that the differential costs in the two cases will push the decision boundary towards making fewer errors on the rare class. Furthermore, we not only punish the violation of the decision boundary, but also the presence of points that are too close to the decision boundary (on the correct side). In other words, we penalize the violation of even stricter constraints than the one implied by the linear decision boundary. These stricter constraints correspond to two parallel hyperplanes on either side of the decision hyperplane, which are $\overline{W} \cdot \overline{X} + b \geq 1$ (for normal points) and $\overline{W} \cdot \overline{X} + b < -1$ (for anomalous points). The goal is to ensure that very few points lie between these two hyperplanes. This is achieved by using

penalties to discourage points from lying in these regions, no matter which class they might belong to.

Consider a data set $\mathcal{D}$ with anomalous points $\mathcal{O}$. Then, the penalty for the normal points is given by $\sum_{\overline{X} \in \mathcal{D}-\mathcal{O}} \max\{0, 1 - \overline{W} \cdot \overline{X} - b\}$ and the penalty for the anomalous points is given by $\frac{c_2}{c_1} \sum_{\overline{X} \in \mathcal{O}} \max\{0, 1 + \overline{W} \cdot \overline{X} + b\}$. In addition, we add a regularization term $\lambda ||\overline{W}||^2$. Here, $\lambda > 0$ is the regularization parameter. Therefore, the overall objective function may be written as follows:

$$J = \underbrace{\sum_{\overline{X} \in \mathcal{D}-\mathcal{O}} \max\{0, 1 - \overline{W} \cdot \overline{X} - b\}}_{\text{Penalty (normal points)}} + \underbrace{\frac{c_2}{c_1} \sum_{\overline{X} \in \mathcal{O}} \max\{0, 1 + \overline{W} \cdot \overline{X} + b\}}_{\text{Penalty (anomalous points)}} + \underbrace{\lambda ||\overline{W}||^2}_{\text{Regularizer}} \tag{7.3}$$

The differential weighting of the costs is particularly notable in this setting. Note that one can easily solve this optimization problem with gradient-descent. However, if kernel methods are to be used to determine non-linear decision boundaries, then a dual formulation is used frequently. Discussions of the dual formulation for SVMs may be found in [33], and its modification to the aforementioned cost-sensitive case is straightforward.

7.2.2 Adaptive Re-sampling

It is noteworthy that most of the weighting methods simply increase the weight of the rare class in order to bias the classification process towards classifying rare-class instances correctly. In adaptive re-sampling, a similar goal of increasing the impact of the rare class is achieved by differentially sampling the training data in favor of the rare class. After all, classification models are directly influenced by the frequency of presence of each class. For example, in the case of the naive Bayes classifier, the prior probabilities are proportional to the relative presence of each class.

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 types of sampling can be simultaneously executed. Sampling with replacement is necessary when it is needed to oversample a class. 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-class costs in the sample used for learning. It has generally been observed [173], 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 [124, 332]. This is also referred to as *one-sided selection*. Under-sampling has several advantages:

- 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.
- Because of the improved efficiency, one can draw multiple subsamples and average the predictions in order to create a more robust ensemble model.

Sampling methods are often used in the context of subsampling ensembles in classification [105, 106, 107], in which the samples are repeatedly drawn in order to create multiple

models. The prediction of a test point is obtained by averaging the predictions from different models. This type of approach significantly improves the accuracy of the base method because of its variance reduction effects of the ensemble method. This can be viewed as the supervised avatar of the subsampling approach discussed in Chapter 6.

7.2.2.1 Relationship between Weighting and Sampling

Since cost-sensitive learning can be logically understood as methods that weight examples of various classes in a differential way, a question arises as to how these methods relate to one another. Adaptive re-sampling methods can be understood as methods that 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, although 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 if *only a single instantiation is used in both cases*. This seems to be the case from many practical experiences with real data [132]. 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. This means that we can use 50 different instantiations of the sampling method at the same cost of one instantiation of the weighting method. By averaging the results from these different instantiations, one will typically obtain more accurate results with the use of sampling than in the case of weighting. In most cases, a small number of ensemble components (between 10 and 25) are required in order to gain most of the accuracy advantages.

In addition, a special type of ensemble known as the *sequential ensemble* has also been proposed in [371]. In the sequential ensemble, the choice of the majority class instances selected in a given iteration is regulated by the predictions of the classifier during previous iterations. Specifically, only majority instances that 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.

7.2.2.2 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 data point. This could lead to over-fitting and deteriorated bias-centric effects on accuracy. In order to address this issue, it was suggested [133] that synthetic over-sampling could be used to create the over-sampled examples in a way which provides better accuracy.

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. In addition to over-sampling the rare class, the *SMOTE* algorithm under-samples the minority class. 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. Another advantage of the *SMOTE* algorithm is that it can be easily combined with boosting algorithms to create more robust methods like *SMOTEBoost* (see next section). Although it has not been discussed in the original paper, a simpler approach for creating a very robust ensemble is to average the point-wise predictions from various (randomized) *SMOTE* instantiations; we refer to this variation as *SMOTEAverage*.

7.2.3 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 [478] works by associating each training example with a *weight* that is updated in each iteration depending on the accuracy of its prediction in the previous iteration. Misclassified instances are given larger weights. The idea is to give greater importance to "difficult" instances which may lie near the decision boundaries of the classification process. The overall classification results are computed as a combination of the results from different rounds. The basic idea is that the classifier has a bias caused by the incorrect shape of this decision boundary. By combining the predictions from multiple training models on biased data sets, one is able to roughly approximate the shape of the decision boundary correctly. In other words, a combination of the results on a set of biased *data sets* are used to correct for the inherent bias in a particular *model*.

Next, we describe the boosting method in more detail. In the tth round, the weight of the ith instance is $D_t(i)$. The algorithm is initialized with an equal weight of $1/N$ for each of the N instances, and updates them in each iteration depending on the correctness of the classification model (on the training instances). In practice, it is always assumed that the weights are normalized in order to sum to 1, although the approach will be described below in terms of (unscaled) relative weights for notational simplicity. In the event that the ith example is misclassified in the tth iteration, its (relative) weight is increased to $D_{t+1}(i) = D_t(i) \cdot e^{\alpha_t}$. 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. The weighting is typically implemented using sampling.

The rare-class setting also has an impact on the algorithmic design of boosting methods. A specific example is the *AdaCost* method [191], which uses the costs to update the weights. 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 on 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 [191], 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 [299, 301]. A general observation about boosting methods is that their primary focus is on bias correction. Therefore, it is important to use classifiers with high bias and low variance (e.g., linear SVM instead of kernel SVM). This is because we want to ensure that the incorrect classification of training examples is caused by bias rather than variance. In cases where the classifier has high variance, the boosting method could preferentially overweight examples based on random variations in algorithm output or other noisy examples. Similarly, boosting methods tend to work poorly in data sets with a large amount of noise.

7.3 Semi-Supervision: 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 in which it is desirable to classify or collect all documents that belong to a rare class. In many scenarios, such as the case of Web documents, the types of the documents available are too diverse. As a result, it is hard to define a representative negative sample of documents from the Web.

This leads to numerous challenges at the *data acquisition stage*, in which the collection of negative examples is difficult. 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 [595], this scenario is quite common. Although a number of methods have been proposed for negative data collection in such unstructured domains, none of which are fully 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 cleaning the sample of the negative class. The amount of human effort involved in labeling 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, and simply tolerate the presence of positive contaminants. This could lead to two different types of challenges:

- The contaminants in the negative class can reduce the effectiveness of a classifier, although it is still better to use the contaminated training examples rather than to completely discard them. This setting is referred to as *positive-unlabeled (PUC)* data classification.

- 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 [356]. Therefore, there are a few algorithms that discard the unlabeled data altogether and work with only the positive class.

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

Although some methods in the literature treat this as a new problem that is distinct from the fully supervised classification problem [362], other methods [183] 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 [183] 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. 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 types of methods that can be used in order to address the problem of contaminated normal class examples. In the first class of methods, heuristics are used in order to identify (unlabeled) training examples that are negative (i.e., normal). Subsequently, a classifier is trained on the positive examples, together with the unlabeled examples that have been *reliably* predicted to be negative. A less common approach is to assign weights to the unlabeled training examples [348, 362]. The second case is a soft version of the first, because the first setting can be viewed as an approach in which binary weights are used. It has been shown in the literature [363], that the second approach is superior. An SVM approach is used in order to learn the weights. Another work in [603] uses a probabilistic approach in which the weight vector is learned in order to provide robust estimates of the prediction probabilities.

Although the use of unlabeled data as the negative class is adequate in most settings, 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 [356]. Therefore, it may be better to simply discard the negative examples and build the model only from the anomalous examples (cf. section 7.4.1).

7.4 Semi-Supervision: Partially Observed Classes

The data might contain a normal class and several anomalous classes. In such cases, several interesting settings might arise in which some of the classes are observed, whereas others are not. For example, one might have examples of only the anomalous class, only the rare class, or a combination of some subset of the rare classes together with the normal class. In this context, it is noteworthy that missing anomalous classes are more common than missing normal classes. This is because anomalous classes are naturally so rare that it is difficult to collect examples of their occurrence. Some examples of various scenarios of missing classes are as follows:

- In some cases, it is difficult to obtain a clean sample of normal data, when the background data is too diverse or contaminated. In such cases, the anomalous class may be observed but the normal class may be missing.

- 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.

These versions of the problem are truly semi-supervised because training data are available about some portions of the data but not others. The case of positive and unlabeled data is much simpler because the unlabeled class can often be approximated as a negative class. Therefore, such scenarios are best addressed with a combination of supervised and unsupervised techniques. The cases in which examples of anomalous class are available are very different from those in which examples of only the normal class are available. In the former, outliers are points that are as *similar* as possible to the training data, whereas in the latter, outliers are points that are as *different* as possible from the training data. In cases where examples of only the normal class are available, the distinction from unsupervised outlier detection is minimal. A distinction between unsupervised outlier detection and one-class *novelty* detection is that the term "novelty" is often used in a temporal context in which the normal examples have arrived in the past.

7.4.1 One-Class Learning with Anomalous Examples

In most natural settings, the anomalous class is missing, and copious examples of the normal class may be available. There are a few cases in which only the anomalous class is available, although such situations are rare. As a result, there are very few methods addressing this setting. 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. Therefore, one cannot sample the background class to create unlabeled instances and use positive-unlabeled learning methods.

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 outlier detectors are very natural to construct in the one-class scenario, because the propensity of a test instance to belong to a class can be naturally modeled in terms of distances. Therefore, a data point may be scored on the basis of its average k-nearest neighbor distance, except that *lower* scores indicate a greater degree of outlierness. It is much harder to adapt other outlier detection methods to this variation of the one-class setting because of paucity of data and overfitting.

Another possibility is the adaptation of one-class SVMs to this setting. However, the SVM method in section 3.4 of Chapter 3 assumes the availability of examples from the normal class rather than the anomalous class. The main difference is that we are now looking for test points that are classified on the same side of the separator as the observed points (which are known to be anomalous). It is, however, much harder to learn this type of classifier because examples of the anomalous class are usually scarce. The use of a kernel function would typically cause overfitting. Certain variations of one-class SVMs have been shown [113] to work effectively in this setting (as well as the original setting of normal points only). This approach determines a linear separator that is attracted towards the center of the points in kernel feature space, rather than repelled away from the origin. The work in [113] shows that this approach can detect abnormalities in ball-bearing cages, when only examples of anomalies (faulty cages) are available.

7.4.2 One-Class Learning with Normal Examples

A more common setting is one in which only normal examples are available and none of the rare classes are observed. This problem is, of course, no different from the unsupervised outlier detection setting. *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.

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 variation of outlier detection and the unsupervised version of the problem is limited and artificial, especially when other labeled anomalous classes do not form a part of the training data.* In spite of this, the semi-supervised version of the (normal-class only) problem seems to have a distinct literature of its own, which 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. In general, when working with inductive learning methods for unsupervised outlier detection (like one-class SVMs), it is always a good practice to distinguish between training and test data (by cross-validation) even when they are not distinguished in the original data. This helps in avoiding overfitting.

7.4.3 Learning with a Subset of Labeled Classes

A more challenging scenario arises when labeled rare classes are present in the training data, but new (i.e., *novel*) classes appear from time to time. Such scenarios can arise quite often in many applications such as intrusion detection in which partial knowledge is available about *some* of the anomalies, but others may need to be modeled in an unsupervised way. For a given test point, two key 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 to identify outliers. 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 aforementioned batch-centric algorithms, this temporal aspect is not fully explored because a single snapshot of training data is assumed. Many applications such as intrusion detection are naturally

focused 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 types of novel classes from one another [46, 391, 392]. 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 9.4.3 of Chapter 9.

7.5 Unsupervised Feature Engineering in Supervised Methods

Feature engineering is the problem of selecting the best representation of the data set for a particular model. For example, consider a classification problem in which the data is distributed according to a single Gaussian distribution. Furthermore, rare-class instances are located far away from the data mean in all directions, whereas normal-class instances are clustered near the data mean. This situation is quite conceivable in real settings. If we used a linear support vector machine to construct a training model, it would work very poorly because the rare and normal class are not linearly separable. However, if we extracted the distance of each point from the data mean as one of the features, then the linear support vector machine will provide perfect separation. This is an example of feature engineering, in which extracting a more convenient representation can often overcome the inherent representational limitations of a model.

Although it is possible to use more complex models (e.g., kernel methods) to overcome these types of limitations, there are several challenges in using these models such as computational complexity and overfitting. Overfitting is a common problem when there is a limited number of instances of any class; this is common in rare-class learning because of the paucity of anomalous examples. Kernel methods also cannot control the opaque representation created by the kernel function and are therefore unable to perform feature selection in a controlled way. Feature engineering provides additional flexibility in selecting the correct representation to use out of a set of "reasonable" features.

What is a reasonable representation in the rare-class setting? What types of features are most likely to emphasize the difference between the rare class and the normal class? It is here that unsupervised outlier detection methods can play an important role because the different unsupervised methods are designed to emphasize the "typically" rare characteristics of data sets [395]. In our previous example of outliers at the extremities of a multivariate Gaussian distribution, the outlier score of the linear Mahalanobis method would be an excellent feature to use. Since different outlier detection algorithms emphasize different types of rare characteristics (which may be suitable for different data sets), one can use multiple outlier detection algorithms to extract different features. These features can either replace or be added to the original data representation. A rare-class learning algorithm can be trained on the augmented representation in order to obtain results of better quality.

In order to extract different types of features, one can use the outlier scores from different models, different parameter settings of the same outlier detection model, or the execution of the same model on different feature bags. Since different outlier detection algorithms work well on different data sets, it is evident that the useful features will also be data-specific.

For example, the features extracted from LOF would be useful for a data set containing many local outliers, whereas the features extracted from the k-nearest neighbor method would be useful for a data set containing many global outliers. Serendipity also plays a role in extracting useful features. For example, the scores from a sampled feature bag may work very well for a particular data set, although there is some element of luck in being able to sample the correct bag.

Note that this approach leads to a significant redundancy in the feature representation of the rare-class learning problem. This is particularly true because many outlier detection models are similar and might lead to highly correlated features. However, one can use any number of supervised feature selection and regularization methods to minimize the effect of redundant features. For example, a linear support-vector machine or logistic regressor with L_1-regularization will automatically remove the effect of redundant or irrelevant features from the model. In this particular setting, the underlying data representation may be very high-dimensional, and therefore L_1-regularization may be more suitable than L_2-regularization. L_1-regularizers automatically set the coefficients of many features to 0 in a regression model, and therefore the effect of using the regularization is to perform guided feature selection. It has been shown in [395] that this type of feature engineering approach significantly enhances the representation of the data and therefore improves the classification results.

Why should such an approach work? Feature engineering methods work best when the features are encoded with an understanding of either the problem or application domain. It is noteworthy that the same data sets and evaluation measures are used for benchmarking both outlier detection and (supervised) rare-class detection problems. In many rare-class detection problems, unsupervised outlier detection algorithms achieve very high accuracy (AUC) values [35]. This is primarily because outlier detection algorithms have been developed over the years with a problem domain-specific understanding of the typical characteristics of anomalies. As a result, such features also enhance the effectiveness of supervised learning methods.

It is noteworthy that feature engineering methods can be either supervised or unsupervised. For example, this approach shares a number of similarities with a supervised feature engineering and ensemble method, referred to as *stacking* [33]. In stacking, the features are learned using the output of supervised classification algorithms rather than unsupervised outlier detection algorithms. Unsupervised feature engineering methods are often more effective when the amount of training data is small. If a lot of labeled data is available, then it is possible to partition the data between the feature learning part and the second-phase of prediction for better results (with stacking). In the rare-class setting, the amount of training data for at least one of the classes is very little. As a result, an unsupervised approach, such as the use of outlier scores, is appropriate in this setting.

7.6 Active Learning

It is expensive to manually label examples in order to identify the rare class. Therefore, the basic idea is to identify and label more *informative* examples that reduce the need to acquire too many of them. Such examples often occur near the decision boundary between the rare and normal class. In addition, suspected outliers are more valuable than suspected inliers for labeling, because the former are harder to acquire. By identifying such examples, it is possible to train classifiers with far fewer examples. This is particularly useful in reducing the cost of label acquisition. Therefore, the most important aspect of active learning is in

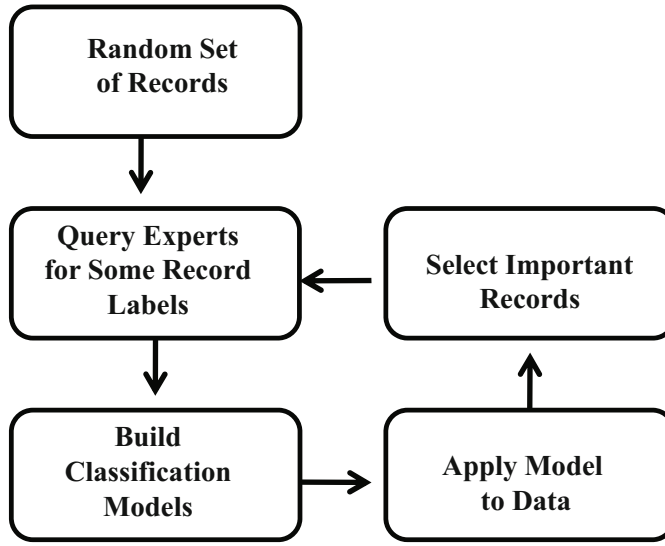


Figure 7.2: The overall procedure for active learning

identifying these critical examples near the decision boundary. Active learning is a common approach used in classification domains [11, 33, 486]. However, in anomaly and rare-class detection, a bigger issue is the *cold-start problem* in which few training examples are available at the very beginning. In fact, in a purely unsupervised setting, there might be no labeling to begin with. While the active learning for rare-class detection is not too different from the classification domain, there are some additional challenges from the cold-start point of view.

An interesting procedure for active learning from unlabeled data is proposed in [431]. 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 proper determination of the important instances is key to the effectiveness of this approach. An iterative approach is therefore used in which the examples are identified with the use of labels that are available, and these are then presented to the human expert in order to augment the labeled data.

In the setting discussed in [431], no labeled examples are available at the beginning of the process. The first iteration is special, in which a purely unsupervised approach is used for learning the important examples and presenting them to the user. The user labels these examples to provide the first set of training data. Subsequent iterations are similar to the traditional active learning setting for classification. The key lies in the process of properly selecting the examples to be labeled, which is discussed below. This procedure is performed iteratively until the addition of further examples is no longer deemed significantly helpful for further classification or a predefined labeling budget is exhausted. The overall procedure is illustrated in Figure 7.2. It should be noted that this approach can also be used in scenarios in which few 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 that are predicted to be clearly 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* in prediction that should be presented to the user for labeling. The intuition is that such examples lie on the decision boundary and by labeling them, one can gain the greatest knowledge about the decision boundaries between the different classes. Such an approach maximizes the learning of the contours separating different classes, while requiring the least amount of (potentially expensive) expert supervision. In the context of rare-class detection, however, the greater value of labeling rare-class examples is also taken into account. Outliers are far more valuable than inliers as examples, and, therefore, if a point is suspected of being an outlier (but not included), it should be labeled. In other words, some subtle changes need to be made to the usual rules for selecting examples in the rare-class setting. In other words, the two key selection criteria are as follows [431]:

- *Low likelihood:* These are data points that have low fit to the model describing the data. For example, if an EM algorithm is used for modeling, then these are points that have low fit to the underlying model. Since these are suspected outliers, it would be desirable to label them.
- *High uncertainty:* These are points that have the greatest uncertainty in terms of their membership to either the rare class or the normal class. Therefore, by acquiring the label of this example, one learns something new about the decision space, thereby helping the underlying algorithms.

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. This ranking is used in order to select the next data point to be labeled by the expert.

Note that the low-likelihood criterion is specific to the rare class and outlier detection setting, whereas the high-uncertainty criterion is common to all classification settings. Several surveys [11, 486] on active learning in classification describe methods to quantify the uncertainty in prediction of a data point. Some examples of such quantifications are as follows:

- Consider a case in which a Bayes classifier predicts the probability of a point belonging to the positive class to be very similar to the (estimated) fraction of outliers in the data. In such a case, this example might be informative because it lies on the decision boundary between the rare and anomalous class. The estimated fraction of outliers in the base data can be computed based on domain knowledge. However, in the rare-class setting, the acquisition of rare examples is more critical. Therefore, as long as the Bayes classifier predicts the probability of a point belong to the rare class to be *greater* than the estimated fraction of outliers, the point is considered informative. In a sense, this approach merges the low-likelihood criterion with the high-uncertainty criterion in a seamless way.
- Another approach is the principle of *query by committee* [485]. An ensemble of classifiers is trained, and points with the greatest *predictive disagreement* (by different classifiers) are selected. The basic idea is that different models make varying predictions on data points near the decision boundary between the normal and anomalous class and are therefore more informative for learning. A variety of such criteria based on ensemble learning are discussed in [394].

Details of other relevant methods for active learning are discussed in [11, 486].

7.7 Supervised Models for Unsupervised Outlier Detection

In section 1.2.1 of Chapter 1, it is pointed out that outlier detection models can be viewed as one-class avatars of classification models. In this section, we point out yet another connection to regression modeling. This particular connection has significant *practical* applicability because it enables the use of off-the-shelf classification and regression models for outlier detection.

Section 3.2.1 of Chapter 3 shows how one can use linear regression models to predict errors in a particular attribute in the data from the other attributes. Such models are very useful in contextual settings like time-series data where there is a clear distinction between contextual and behavioral attributes. However, it turns out that these models can also be used for unsupervised anomaly detection in multidimensional data. This broad idea has been explored recently [417, 429] in several anomaly detectors, although we focus on [429] because of its recency, extensive experimentation and clear explanations.

The basic idea [429] is to repeatedly apply a regression model $\mathcal{M}_k$ by selecting the kth attribute as the dependent variable and the remaining attributes as the independent variables. Cross-validation is used [33] to ensure that each data point receives an out-of-sample score. The squared-error in prediction of the dependent variable is used to compute the score for the instance for that iteration. This approach is repeated with each attribute as the dependent variable and a weighted average is used to construct the final score. The weight is defined by the ability of the regression model to predict that attribute; the weight ensures that irrelevant attributes like identifiers are not used. It is noteworthy that one is not restricted to using the linear models of section 3.2.1; in fact, *virtually any of the hundreds of off-the-shelf regression models may be used for prediction.* This makes the approach almost trivial to implement, and numerous options are available for the regression subroutine. The approach is referred to as *attribute-wise learning for scoring outliers (ALSO).*

Let $\epsilon_k^2(\overline{X})$ be the squared error of data point $\overline{X}$ using the model $\mathcal{M}_k$ in which the kth attribute is used as the dependent variable. Note that cross-validation is used to compute each $\epsilon_k^2(\overline{X})$ so that each point receives an out-of-sample score. It is important to standardize the data to unit variance as a preprocessing step in order to avoid problems associated with relative scaling of various attributes. Standardization ensures that the differences in $\epsilon_k^2(\overline{X})$ across different attributes are not regulated by the scale of the attributes.

It now remains to describe how one might choose the weight of each component of the regression model. The basic idea is to ensure that the components of the model corresponding to irrelevant dependent attributes, which cannot be predicted easily by other attributes, are not weighted significantly. How does one determine such irrelevant attributes? Consider a trivial regression model $\mathcal{T}_k$ that always predicts the mean of the kth dependent attribute for every data point irrespective of the values of the independent attributes. The root-mean-squared error of the kth such trivial model is always 1 because each attribute has been scaled to unit variance. Let $RMSE(\mathcal{M}_k)$ represent the root-mean-squared error of the kth attribute:

$$RMSE(\mathcal{M}_k) = \sqrt{\frac{\sum_{i=1}^{N} \epsilon_k^2(\overline{X_i})}{N}} \tag{7.4}$$

Then, the weight w_k of the model with the kth attribute as the dependent variable is given

by the following:

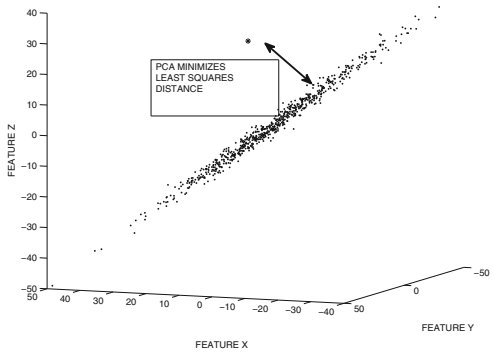
$$w_k = 1 - \min\{1, RMSE(\mathcal{M}_k)\} \tag{7.5}$$

The basic idea is that if the prediction performs worse than predicting the mean, the attribute is irrelevant and one should give the model a weight of 0. The maximum weight of the model can be 1 when it gives zero error. Then, the outlier score of the d-dimensional data point $\overline{X}$ is as follows:

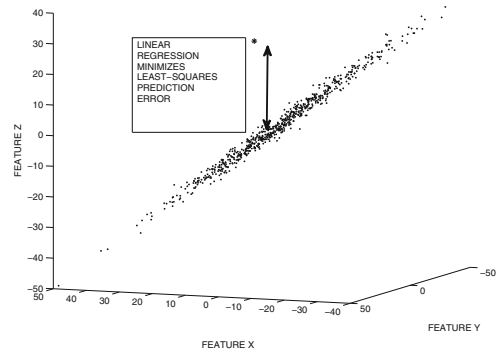
$$\text{Score}(\overline{X}) = \sum_{k=1}^{d} w_k \, \epsilon_k^2(\overline{X}) \tag{7.6}$$

Larger values of the score are more indicative of outlierness. Note that the score presented here is a simplified version of the one in [429], although it is equivalent because of the relative nature of outlier scores. It is noteworthy that the approach in [429] is able to achieve high-quality results with only a limited number of relatively simple base models, and it is possible that better results may be obtained with improved regression models. In particular, even though it has been shown that the use of M5 regression trees [453] as the base model provides excellent results, the use of its most obvious enhancement in the form of random forests for regression has not been tested. Since random forests are known to be very robust compared to most other classification and regression models in a wide variety of settings [195], it might be worthwhile to test the effectiveness of the approach with the use of random forests as the base regressor. One can also adjust the classifier or regressor depending on the domain that is tested (e.g., text versus images).

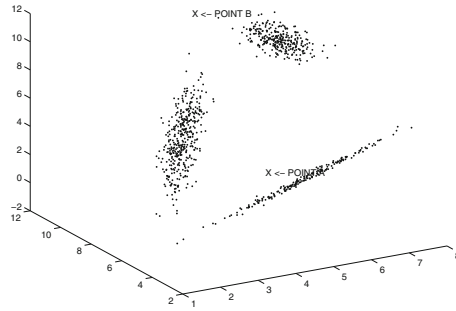
This approach also has natural connections with subspace outlier detection [4] because it provides an explanation of the outlierness of each data point in terms of *locally relevant* subspaces when certain types of base predictors are used. In this context, random forests provide the best interpretability in terms of subspace outlier detection. For example, if a random forest is used as the base learner and the attribute k has a large value of $w_k \, \epsilon_k^2(\overline{X})$, then the most erroneous paths in the underlying decision trees of $\mathcal{M}_k$ provide insights about locally relevant subspaces. This is somewhat easier to understand when the target attribute is symbolic, or a numeric target attribute is discretized into a symbolic value. For example, consider a setting in which the target attribute is a binary disease attribute $Alzheimer$ for which the value is drawn from $\{0, 1\}$. This is a disease that predominantly affects very old individuals except for a few rare (i.e., outlier) cases. Although the disease is primarily genetic, some factors like head-trauma further increase the risk. Therefore, a sequence of splits such as $Age \leq 30, Trauma = 0$ will typically predict $Alzheimer = 0$. However, some test instances satisfying this sequence of splits might have $Alzheimer = 1$, which suggests that these are outliers. In such cases, one has an immediate understanding of the causality of this outlier. Furthermore, the sequence (corresponding to tree path) can be concatenated with the target attribute value (or its numeric range) to infer a locally relevant subspace for the specific test point. In this particular example, the locally relevant subspace is $Age \leq 30, Trauma = 0, Alzheimer = 1$. Furthermore, as is desirable [31] for subspace outlier detection, more than one locally relevant subspace is implicitly used by the approach to score points (and interpret them). This is because we can repeat this process with all the random forests $\mathcal{M}_k$ and each random forest contains multiple decision trees. This is yet another reason why random forests should be used as base learners in this setting; they are likely to be robust because of their implicit use of multiple locally relevant subspaces.



(a) PCA-based error (Chapter 3)



(b) Linear regression-based error



(c) Cannot be modeled by PCA or linear regression

Figure 7.3: Showing connections between regression methods and PCA. The use of linear regression as a base method for the approach in this section results in intuitively similar models to the PCA method of Chapter 3. However, supervised modeling also allows the use of complex base methods like random forests. Such methods can handle complex data distributions like (c).

7.7.1 Connections with PCA-Based Methods

Chapter 3 discusses dependent variable regression in section 3.2.1 and independent variable regression (PCA) in section 3.2.2. A soft version of the latter is the Mahalanobis method (section 3.3.1). The approach in the aforementioned section is an instantiation of dependent-variable regression, especially when linear regression is used as the subroutine for each of the decomposed subproblems. Furthermore, the error (outlier score) with the use of PCA will be highly correlated with the aggregate regression errors using this approach when linear regression is used in the latter case. For example, we have shown PCA-wise regression errors in Figure 7.3(a) and an attribute-wise (linear) regression error in Figure 7.3(b). These errors are shown with double-sided arrows for the same outlier in Figures 7.3(a) and (b), respectively, and they are clearly correlated. However, there are cases that cannot be handled so easily by either method.

For example, the distribution in Figure 7.3(c) cannot be modeled by either PCA or dependent-variable modeling because outlier 'A' is exposed in a different rotated subspace than outlier 'B.' However, if a random forest is used as the base regression method, then such outliers will be exposed as well because of the ability of random forests to model such complex nonlinear distributions. The decomposition of a single unsupervised problem into many supervised problems also has the advantage of allowing the most relevant type of feature selection in each subproblem. In principle, such outliers can also be exposed with nonlinear PCA (section 3.3.8), when a spectral kernel is used. Therefore, there are deep connections between the Mahalanobis method and the approach discussed in this section. The approach discussed in this section is potentially more powerful because the use of certain base regressors like random forests provides a complex model with embedded feature selection. On the other hand, soft PCA allows better score aggregation along independent component directions. In general, their relative performance might depend on the complexity of the underlying data set at hand.

One can also show that the Mahalanobis method is a variation of this approach in another way by modifying the weighting function of Equation 7.5 as follows:

$$w_k = \frac{1}{[RMSE(\mathcal{M}_k)]^2} \tag{7.7}$$

Consider a setting in which we transformed the data into a new d-dimensional PCA-based axis system in which the various dimensions are uncorrelated. If we apply (supervised) least-squares regression to this transformed data by using the aforementioned partitioning scheme into independent and dependent variables, the regression model would not be (theoretically) expected to learn anything more than predicting the mean of the dependent variable. This is because we have already removed all the inter-attribute correlations by transformation, and the transformed variables do not have predictive power with respect to each other with the use of a linear regression model. In such a case, the best that one can theoretically expect[3] from the regression model is to report the mean along each transformed dimension as the prediction for each point. Therefore, the weight w_k in Equation 7.7 will simply be the inverse of the variance of the attribute. In other words, if $\epsilon_i(\overline{X})$ is the error along the ith transformed dimension of data point $\overline{X}$, then the score according to Equation 7.6 (using

[3]For a PCA-transformed data set D, which is also mean-centered, the coefficients of linear regression can be computed as $(D^T D)^{-1} D^T \overline{y}$, where $\overline{y}$ is the dependent variable (section 3.2.1). We do not need to account for the bias term by adding an additional column of 1s to D because of mean-centering. However, $D^T \overline{y}$ is a vector of 0s because of the orthogonality property of SVD embeddings. Therefore, all coefficients will be 0, and one will always predict the dependent variable to be 0 (i.e., the mean).

the modified weight of Equation 7.7) is as follows:

$$\text{Score}(\overline{X}) = \sum_{k=1}^{d} \frac{\epsilon_k^2(\overline{X})}{\text{Variance of } k\text{th attribute}} \tag{7.8}$$

This is exactly the scoring function used by the Mahalanobis method (cf. Equation 3.17 in Chapter 3). One advantage of using this weight is that the outlier score can be modeled from a χ^2 distribution (see Chapter 3).

Although the Mahalanobis method has an advantage in some settings, it needs to be kept in mind that the supervised approach can discover outliers with complex base learners like random forests. In such cases, the underlying classifier can leverage the use of higher-order correlations among the dimensions, even after the first-order correlations have been removed. On the other hand, poor predictive power with respect to too many weakly relevant dimensions can continue to cause challenges in *ALSO*, no matter how one might adjust the scoring mechanism to account for irrelevant dimensions. This can be a particularly major problem in very high-dimensional domains like text in which it is almost impossible to predict the frequency of any particular word from other words even though there might be significant latent correlations in the aggregate. The second problem with the approach is that if two dimensions are perfectly correlated (e.g., Celsius and Fahrenheit readings) and a data point shows extreme values respecting that correlation, the approach will not detect the point as an outlier.

7.7.2 Group-wise Predictions for High-Dimensional Data

Many of the aforementioned challenges can be addressed using predictions on groups of sampled targets [35]. Both the Mahalanobis method and supervised learning can be viewed as special cases of this approach.

The approach is inherently ensemble-centric in nature. In each base detector, the first step is to randomly partition the attributes into r target variables and $(d - r)$ predictor variables. However, the r target variables, denoted by S_r, are not used in their original form but are transformed to latent concepts to sharpen their predictability. The r-dimensional target variables are transformed to uncorrelated targets using PCA and then standardized to zero mean and unit variance. Transformation might occasionally cause target attributes with zero variance (eigenvalues) when the value of r is of the same magnitude as the number of data points. Transformed dimensions with zero variance are dropped, as a result of which one might be left with only $r' \leq r$ targets. The $(d-r)$ predictor variables are used to predict each of the r' transformed attributes using r' individually trained models for the various target attributes. One can use[4] cross-validation in order to ensure out-of-sample scores. Subsequently, the scores $\epsilon_1(\overline{X}) \ldots \epsilon_{r'}(\overline{X})$ over these r' targets are averaged. The resulting score, which is specific to the target set S_r, is as follows:

$$\text{Score}(S_r, \overline{X}) = \frac{\sum_{k=1}^{r'} w_k \, \epsilon_k^2(\overline{X})}{r'} \tag{7.9}$$

Here, the weight w_k is set according to Equation 7.7. This concludes the description of a single base detector for the sampled target attribute set S_r. Note the presence of S_r as

[4]For larger data sets, one can also subsample a small part of the data as the training portion but score all points with the model. This is an efficient approximation which is natural to use in an ensemble-centric setting [35].

an argument of the score in Equation 7.9. The approach is repeated over several samples of $(d - r)$ predictors and r targets. The resulting scores for each data point are averaged in order to compute its final score. For very high-dimensional domains like text, it makes more sense to predict values along each of the r'-dimensional latent vectors rather than the individual word frequencies. The approach is referred to as *GASP* (Group-wise Attribute Selection and Prediction) [35].

Both *ALSO* and the Mahalanobis method are special cases of the *GASP* technique. Setting $r = 1$ yields the *ALSO* technique (with minor scoring modifications), whereas setting $r = d$ yields the Mahalanobis method, assuming that each attribute is predicted as its mean in the absence of predictor attributes. Note that the latter can also find outliers caused by correlated extreme values. As the dimensionality of the data increases, it often makes sense to increase the raw value of r, albeit sublinearly with increasing data dimensionality (e.g., $r = \lfloor \sqrt{d} \rfloor$). The work in [35] presents experimental results for the case in which r is set to $d/2$ along with an ensemble-centric variation.

7.7.3 Applicability to Mixed-Attribute Data Sets

One of the nice characteristics of this approach is that it is directly generalizable to mixed-attribute data sets with minor modifications. Most regression models can be applied to mixed-attribute settings very easily. Furthermore, in cases in which the predictive attribute is categorical, one can use a classifier rather than a regression model. In such cases, the value of $\epsilon_k(\overline{X})$ is drawn from $\{0, 1\}$. All other steps of the approach, including the final computation of the score, remain the same.

7.7.4 Incorporating Column-wise Knowledge

The supervised approach is particularly useful in settings where we are looking for specific types of anomalies and it is known that certain attributes are more important than others for anomaly detection. For example, consider an unsupervised setting in which we are predicting fraudulent transactions. Even though some attributes such as age and gender have marginal predictive value, other attributes such as the size of the transaction are far more important. In such cases, it is possible to restrict the predictive process only over the relevant subset of attributes as the target, although all attributes might be used for predicting the target attribute. Therefore, the score of Equation 7.6 is aggregated only over the relevant subset of target attributes. In a sense, one is able to properly distinguish between primary and secondary attributes with this process. This approach is also used for outlier detection in contextual data types such as time-series data in which the prediction of behavioral attributes (e.g., temperature) is more important than that of the contextual attributes (e.g., time). Such relationships between prediction and outlier detection will be discussed in greater detail in Chapter 9 on time-series data.

7.7.5 Other Classification Methods with Synthetic Outliers

A related method for using supervised methods is to generate synthetic data for the outlier class, and treat the provided data set as the normal class. This creates a supervised setting because one can now use any off-the-shelf classifier for the two-class problem. As discussed in [491], the outliers are generated uniformly at random between the minimum and maximum range of each attribute. Subsequently, any binary classification model may be applied to this data set. This method is, however, not quite as desirable because it might introduce

bias in the form of synthetic data. The "typical" pattern of outliers in any given data set is far from random. This is because outliers are often caused by specific types of anomalies in the underlying system, and they are often clustered into small groups. Therefore, the use of randomly distributed data as outliers can result in poor performance for certain types of data sets.

7.8 Conclusions and Summary

This chapter discusses the problem of supervised outlier analysis. In many real scenarios, training data are available, which can be used in order to greatly enhance the effectiveness of the outlier detection process. Many off-the-shelf classification algorithms can be adapted to this problem with minor modifications. The most common modifications are to address class imbalance with sampling and re-weighting.

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. In other cases, no examples of the normal class may be available. 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.

In active learning methods, 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 more insightful results, because the human is involved in the entire process of label acquisition and final outlier detection.

Supervised methods can also be used for unsupervised anomaly detection by decomposing the anomaly detection problem into a set of regression modeling problems. This is achieved by setting each dimension in the data as a predicted variable and the other dimensions as the predictor variables. A weighted average of prediction errors over these different models is used to quantify the outlier score.

7.9 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 [132, 135, 182]. 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 receiver operating characteristic (ROC) curves and classification accuracy in the presence of costs and class imbalance are discussed in [174, 192, 302, 450, 451]. The impact of class imbalance is relevant even for feature selection [399, 616], because it is more desirable to select features that are more indicative of the rare class.

A variety of general methods have been proposed for cost-sensitive learning such as *MetaCost* [175], weighting [600], and sampling [124, 132, 173, 332, 600]. 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. Furthermore, they can often improve effectiveness because of their ability to use subsampled ensembles [32]. Numerous cost-sensitive variations of different classifiers have been proposed along the lines of weighting, including the Bayes classifier [600], nearest-neighbor classifier [609], decision trees [546, 566], rule-based classifiers [298, 300] and SVM classifiers [535, 570].

Ensemble methods for improving the robustness of sampling are proposed in [124, 371]. Since the under-sampling process reduces the number of negative examples, it is natural to use an ensemble of classifiers that combine the results from training 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 [133].

The earliest work on boosting rare classes was proposed in [305]. 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 [191]. Boosting techniques can also be combined with sampling methods, as in the case of the *SMOTEBoost* algorithm [134]. An evaluation of boosting algorithms for rare-class detection is provided in [299]. 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 [301]. It has been shown that the final result from the boosted algorithm is highly dependent on the quality of the base learner.

A commonly encountered case is one in which the instances of the positive (anomalous) class are specified, whereas the other class is unlabeled [183, 356, 348, 362, 363, 595, 603]. Since the unlabeled class is pre-dominantly a negative class with contaminants, it is essentially equivalent to a fully supervised problem with some quantifiable loss in accuracy [183]. 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 [356]. 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 (anomalous) instances of the class are used for training purposes. SVM methods are particularly popular for one-class classification [303, 384, 460, 479, 538]. However, most of these methods are designed for the case in which the observed single class corresponds to normal points. It is also possible to adapt these methods to cases in which the observed class corresponds to anomalous points. Methods for one-class SVM methods for scene classification are proposed in [580]. It has been shown that the SVM method is particularly sensitive to the data set used [460].

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. The special case of this setting is the traditional outlier detection problem in which no training data is available about *any* 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 exist in the combination of the supervised and unsupervised case, and numerous methods have been designed for the streaming scenario [391, 392, 46]. The special case, where only the normal class is available is not very

different from the unsupervised scenario. Numerous methods have been designed for this case such as single-class SVMs [480, 113], minimax probability machines [336], kernel-based PCA methods [462], direct density ratio estimation [262], and extreme value analysis [270]. Single class novelty detection has also been studied extensively in the context of the first story detection in text streams [622] and will be discussed in detail in Chapter 8. The methods for the text-streaming scenario use standard clustering or nearest neighbor models. In fact, a variety of stream-clustering methods [28, 29] 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 [388, 389]. Unsupervised feature engineering methods for improving the accuracy of rare-class detection are discussed in [395].

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 [618, 619]. These methods also add artificially generated examples to the training data 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 [431]. A nearest-neighbor method for active learning is proposed in [252]. 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 [149]. Methods that use query-by-committee to select such points with ensembles are discussed in [394, 485]. A recent approach that minimizes the amount of required data for supervised active learning is discussed in [224]. Surveys on supervised active learning may be found in [18, 486].

A selective sampling method that uses active learning in the context of outlier detection is proposed in [1]. A method has also been proposed in [366] 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. The decomposition of unsupervised outlier detection into a set of supervised learning problems is discussed in [35, 417, 429]. In particular, the work in [35] discusses a number of ensemble-centric variants of the decomposition process.

7.10 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.