Chapter 6

Outlier Ensembles

"Talent wins games, but teamwork and intelligence wins championships."– Michael Jordan

6.1 Introduction

Ensemble analysis is a popular method used to improve the accuracy of various data mining algorithms. Ensemble methods combine the outputs of multiple algorithms or *base detectors* to create a unified output. The basic idea of the approach is that some algorithms will do well on a particular subset of points whereas other algorithms will do better on other subsets of points. However, the ensemble combination is often able to perform more robustly across the board because of its ability to combine the outputs of multiple algorithms. In this chapter, will use the terms *base detector* and *component detector* interchangeably to denote the individual algorithms whose outputs are combined to create the final result.

Ensemble analysis is used frequently in various data mining and machine learning applications such as clustering, classification, outlier analysis, and recommender systems. Furthermore, it has also been applied to various data-types such as multidimensional data, time-series data, ratings data, and networks. The ubiquity of these methods across various problem domains and data types is a result of their relative success in various settings. Ensemble methods regularly win various data mining competitions, the most well-known of which is its victory in the Netflix Prize challenge for recommender systems [34].

Relative to other problem domains, ensemble analysis is a recent field in outlier detection. The reasons for its relative recency (compared to other data mining problems) were discussed in the first position paper on outlier ensembles [31]. One of the key differences between outlier detection and many other problems (like classification) is that the former is an *unsupervised* problem unlike the latter. Supervised ensemble algorithms enjoy the luxury of using the class labels in the analytical process. For example, methods like boosting use the class labels to evaluate the accuracy of data points and weight them accordingly. Such methods cannot be generalized to outlier detection easily because of the unavailability of ground-truth.

© Springer International Publishing AG 2017

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_6

In spite of these differences, it has been shown that the field of outlier ensembles shares a very large number of practical [31] and theoretical [32] characteristics with classification ensembles. As discussed in section 6.3, one can formulate a modified bias-variance trade-off for the outlier analysis setting as is available for the classification setting. As a result, many types of algorithms can be generalized directly from classification to outlier detection, and the results are also similar. The main difference from classification is that *the dependent variable (outlier label) is unobserved*. This means that we cannot adapt classification-ensemble algorithms that use these labels to outlier detection. Furthermore, issues such as parameter tuning remain a larger challenge in the context of unsupervised problems. All these challenges require subtle changes to algorithms for classification ensembles in order to generalize them to outlier detection.

The most common use-case for outlier ensembles is the setting of subspace outlier detection. This is primarily because of the uncertainty associated with identifying relevant subspaces for a given data point. Outlier ensembles are most useful in settings in which they are able to reduce the uncertainty associated with difficult algorithmic choices. In fact, the first subspace-outlier detection algorithm [4] can also be viewed as an ensemble method in which a maximization combination function is used on the scores to report outliers. Subsequently, as discussed in the previous chapter, most of the prominent methods for subspace outlier detection tend to use ensemble methods. The reasons for this lie in the natural similarities between subspace outlier detection and ensemble analysis. Just as ensemble analysis works well in cases in which different algorithms work better on different subsets of points, high-dimensional outlier detection works best in cases where different subspaces are relevant to different sets of points. Therefore, by using ensemble components to explore different subsets of dimensions, one is often able to combine the results of these different exploratory base detectors to create stable and accurate results.

There are two key design choices in the construction of an ensemble method:

1. **Choice of base detector:** Selecting the base detector is one of the first steps of an ensemble method. These base detectors might be completely different from one another, have different settings of the parameters, or they might use reconstructed data sets from the base data.

2. **Methodology for score normalization and combination:** Different detectors might create scores on different scales. For example, an average k-nearest neighbor detector will produce a raw distance score, whereas the LOF algorithm will produce a normalized value. Furthermore, although the general convention is to output larger scores for outliers, some detectors use the opposite convention and output smaller scores for outliers. Therefore, it is desirable to convert the scores from various detectors into normalized values that can be meaningfully combined. After the scores have been normalized, an important issue is the choice of the combination function used for computing the ensemble score of a point as a function of its scores returned by various base components. The most common choices include the averaging and maximization combination functions, which are discussed later in this chapter.

The design of a base detector and its combination method both depend on the specific goals of a particular ensemble method. This aspect critically depends on the theoretical foundations of outlier ensemble analysis, which decompose the error of an outlier detection technique into two components, known as the *bias* and the *variance*. These quantities have exactly similar interpretations to their counterparts used in the classification domain. The underlying theoretical assumption is that the data set is generated from some unknown base

distribution of data points. Note that if we actually had access to the base distribution, we can effectively have access to an infinite resource because we can generate as many training instances as we want from the data. However, we never actually have access to the base distribution, and we only have access to a single instance of the training data set. Trying to create a model with a finite training data set will inevitably lead to errors because of the inability to fully model the complexities of the base distribution. Even if we had infinite data, the specific model used might not be appropriate for the data distribution at hand. These two sources of error contribute to the variance and the bias, respectively. An intuitive understanding of these two sources of error is as follows:

1. **Variance:** It is assumed that the training data sets are generated from a base distribution for scoring outliers. Typically, the analyst only has access to a single finite data set drawn from the base distribution, as a result of which she will obtain a different outlier score for the same point when different finite data sets are used to score it using the same algorithm. For example, the k-nearest neighbor score of the same out-of-sample test point will be different over two different sets of 100 training points. This difference in results obtained from different training data sets (drawn from the same distribution) is a manifestation of model variance, which is a component of the error.

2. **Bias:** Even though one does not have ground-truth available in outlier detection problems, it is assumed that some *hypothetically ideal set of scores* do exist, which are not available to the algorithm. This is different from classification, where one at least has a (possibly noisy) sample of the dependent variable. Note that a particular outlier detection model might not appropriately reflect these hypothetically ideal scores, and therefore the *expected* scores returned by the algorithm will vary from the true scores. For example, consider a setting in which the generating process (theoretical base distribution) causes outliers uniformly at random throughout the entire range of the data space. On the other hand, if we use a multivariate extreme-value detector like the Mahalanobis method (section 2.3.4) to compute the scores, we are expected to perform poorly because of the inherent mismatch between the theoretical distribution of outliers and the modeling assumptions. This source of error is referred to as the bias. It is generally much harder to reduce bias in outlier detection problems in a controlled way because the ground-truth is not available as a "guiding post" to correct the errors in the assumptions of the model. Most bias-reduction methods in supervised settings use the ground truth in one form or the other.

Most outlier detectors are designed to work with one or more base detectors, and use a repeated application of these base detectors in order to generate scores with improved bias or variance. This chapter will provide several examples of both bias and variance reduction.

This chapter is organized as follows. In the next section, we will provide a broad overview of the various ways in which different base detectors are combined to create an ensemble. The theoretical foundations of outlier ensembles are introduced in section 6.3. In section 6.4, we will discuss several ensemble methods for variance reduction. Ensemble methods for bias reduction are discussed in section 6.5. A detailed discussion of several score combination methods is provided in section 6.6. The conclusions and summary are presented in section 6.7.

6.2 Categorization and Design of Ensemble Methods

As discussed in Chapter 1, ensemble methods can be categorized in two different ways. These two types of categorization are mutually orthogonal. The first type of categorization defines model-centric and data-centric ensembles as follows:

1. **Model-centric ensembles:** In this case, the different base detectors of the ensemble correspond to different models (algorithms), different parameter settings of the same model, or different randomized instantiations of the same model.

2. **Data-centric ensembles:** In this case, the different base detectors of the ensemble correspond to the application of the same model on different derivatives of the data. These different derivatives may be constructed by drawing samples from the data, drawing random projections from the data, weighting the data points, weighting the dimensions, or adding noise to the data.

After all the base detectors have been executed, the scores from the different models are combined to provide the final result. This combination is often performed by taking the average or the maximum of the scores from various detectors, although other possibilities will also be discussed in this chapter.

It is noteworthy that data-centric ensembles can be considered special cases of model-centric ensembles, when the process of drawing the training data from the base data is considered a part of the model execution. In fact, some data-centric ensembles like feature bagging can be better explained by viewing them from a model-centric perspective. However, explicitly distinguishing between model-centric and data-centric ensembles is helpful in distinguishing between their subtle theoretical differences [35].

There is, however, a different way of categorizing the different ensemble methods in terms of the level of independence of the base detectors. This methodology for categorizing the base detectors is as follows:

1. **Independent ensembles:** In this case, the different components of the base detector are executed independently of one another. The scores from the different detectors are then combined. Most of the existing ensemble methods fall into this category.

2. **Sequential ensembles:** In this case, the base detectors of the ensemble are executed one after the other in order to create successively refined models. An example of such an algorithm is one in which outliers are successively removed from the data to create a better model of normal data. Either the results from the final execution of the ensemble are reported, or the results from various components are combined into a unified prediction.

Note that independent ensembles can be either model-centric or data-centric; similarly, sequential ensembles can be either model-centric or data-centric. A large number of models have been proposed that belong to these various categories. However, sequential ensembles tend to be less explored than other types of models. This is, in part, because sequential ensembles are inherently designed for settings like classification in which ground-truth is available to use as a "guiding post" for future executions of the ensemble method. A more detailed discussion of these different categories of ensemble algorithms is provided in [35].

6.2.1 Basic Score Normalization and Combination Methods

Although the topic of score combination will be discussed in greater detail in section 6.6, it is necessary to touch on this topic early in this chapter in order to explain the various ensemble algorithms properly.

Throughout this chapter, we will assume that the data set $\mathcal{D}$ contains N points, which is typically both the training and test data. It is common not to distinguish between training and test data in unsupervised problems, although it is much easier to perform theoretical analysis by distinguishing between them. In cases where the training and test data set are distinguished from one another, the number of training points is denoted by N and the number of test points is denoted by n. It is assumed that the number of base detectors is m, and the scores of these base detectors for the ith point are denoted by $s_1(i) \ldots s_m(i)$. The goal of the combination process is to create a unified score for each data point from the m base detectors. As in all the other chapters, the dimensionality of the data set is assumed to be d.

The first step in any ensemble combination is that of score normalization to account for the fact that the different algorithms may use scores on different scales, or they might even have a different ordering of the scores. For example, a k-nearest neighbor algorithm would typically output scores on a different scale than the LOF algorithm. Similarly, an EM-algorithm might output fit values in which outliers would tend to have lower scores. On the other hand, most algorithms assume the opposite convention. Therefore, it is important to normalize scores, so that one can meaningfully combine various algorithms without inadvertently over-weighting one of the algorithms.

The simplest approach to the ordering issue is to flip the sign of the scores. By flipping the sign of the scores, an output in which smaller scores indicate greater outlier tendency will be converted into an output in which larger scores indicate greater outlier tendency. Subsequently, the scores can be converted into comparable values by using one of the following two methods:

1. **Range-based scaling:** In this case, the maximum and minimum scores of each detector are computed. Let these values for the jth detector be max_j and min_j respectively. Then, for the ith data point, the score $s_j(i)$ of the ith point by the jth detector is normalized to the following scaled value $S_j(i)$:

$$S_j(i) = \frac{s_j(i) - min_j}{max_j - min_j} \tag{6.1}$$

   The resulting score will lie between 0 and 1 in each case. One problem with this approach is that the values of the scores will depend crucially on the values of max_i and min_i. In most cases, the point taking on the score max_i is the strongest outlier in the data, whose score is rather unstable and can be much larger than those of the other points. This can sometimes reduce the discrimination among the remaining scores in a drastic way, even if many of them happen to be outliers.

2. **Standardization:** Standardization is a more reliable approach for converting the scores into normalized values. In effect, the approach converts the scores into the Z-values introduced in Chapter 2. Let μ_j and σ_j be the mean and standard deviation of the scores returned by the jth detector. Then, the standardized score $S_j(i)$ of the ith data point by the jth detector is defined as follows:

$$S_j(i) = \frac{s_j(i) - \mu_j}{\sigma_j} \tag{6.2}$$

Note that standardization uses the assumption that the 1-dimensional scores follow a normal distribution. Although this assumption is almost never true, this type of normalization can often provide reasonably robust results. More complex assumptions on the distributions of the scores have the drawback of being too sensitive to the specific characteristics of the data set at hand.

Another approach, which is discussed in [213], is to convert scores into probabilities with the use of the EM-algorithm. Such an approach is discussed in detail in section 2.4.4 of Chapter 2. However, there is an important detail that needs to be kept in mind when combining probabilistic outputs via addition in an ensemble method. In such cases, additional steps are required to transform the scores to a more discriminative scale. For example, in some settings (such as the fit computation with the EM-algorithm), the fit drops off exponentially with increasing tendency to be an outlier. This is because the fit value exponentiates the squared Mahalanobis distance inside a Gaussian probability density function. In such cases, it may become difficult to differentiate between scores of abnormal points, whereas two normal points may be differentiated well. This is the opposite of what we want. For example, there is virtually no numerical distance between a weak outlier with fit probability of 10^{-2} and a strong outlier with a fit probability of 10^{-4}. Furthermore, there is a modest difference between the fit probability of 10^{-4} (strong outlier) and a fit probability of 0.2 (normal point). However, the difference between two normal points with fit values of 0.2 and 0.5 is higher than both the previous cases. In such cases, the ensemble components in which outliers are erroneously treated as normal points will dominate an additive combination of base-detector scores. Clearly, this can be a problem in any setting where the detector outputs probabilities. These situations should be addressed with the use of a logarithmic function on the scores before combination [31]. The use of the logarithms on probability outputs can also be theoretically justified because log-likelihoods are inherently additive. Summing the logarithms in a score combination function is equivalent to using the product of the arguments inside the logarithm function. One can therefore view the ensemble score as a product of the corresponding probabilities just like a naive Bayes method. The robustness of the averaged log-likelihood fit has also been observed in [184].

After the scores of various detectors have been computed, they are combined into a unified score. Although we will discuss the combination functions in more detail in section 6.6, we provide a brief overview of two commonly used combination functions (with their intuition) in order to facilitate further discussion. The two most commonly used combination functions are averaging and the maximum. These combination functions are defined as follows:

1. **Averaging:** In this case, the outlier score of a data point is its mean score over various detectors. In other words, the score of the ith data point is computed as follows:

$$\text{Average}(i) = \frac{\sum_{j=1}^{m} S_j(i)}{m} \tag{6.3}$$

   As we will discuss later in section 6.6, the main effect of averaging is to reduce the variance of the scores and thereby improve accuracy. This is a well-known result from the classification domain [617], and generalizes trivially to outlier ensembles because of the corresponding extension [32] of bias-variance theory to outlier detection.

2. **Maximum:** In this case, the outlier score of a data point is its maximum score over various detectors. In other words, the score of the ith data point is computed as follows:

$$\text{Maximum}(i) = \max_{j=1}^{m} S_j(i) \tag{6.4}$$

The effect of the maximization function is somewhat more complex because it can often improve bias but increase variance. This makes the overall effect unpredictable. Nevertheless, we can often gain the advantages of the bias-centric improvements by combining it with the averaging function. These theoretical effects and the advanced combination functions will be discussed in section 6.6.

Rank-centric variants of these combination functions can also be developed. These rank-centric variants are sometimes more stable, but the specific performance will depend on the detector at hand.

6.3 Theoretical Foundations of Outlier Ensembles

The bias-variance trade-off is often used in the context of supervised learning. Although it might seem at first sight that labels are required to quantify the bias-variance trade-off, it turns out that bias-variance theory can also be adapted to outlier detection with some modifications [32]. The trade-off is somewhat more abstract in the outlier-detection setting because of the unavailability of labels. Nevertheless, it has been shown [32] that the similarities in the theoretical analysis in the two cases lead to easier adaptation of many types of classification ensembles to outlier analysis. The discussion of this section is fully based on this recent work.

Most outlier detection algorithms output scores to quantify the "outlierness" of data points. After the scores have been computed, they can be converted to binary labels. Note that the bias-variance trade-off can be developed either for the scoring version of the problem or it can be developed for the binary labeling version of the problem. It is more interesting to develop this trade-off for the scoring version of the problem because most outlier ensemble algorithms combine the scores from different algorithms. An important observation about outlier scores is that they are *relative*. In other words, if all scores are multiplied by the same positive quantity, or translated by the same amount, it does not change various metrics (e.g., area under curve [AUC] of receiver operating characteristic curves [ROC]) of the outlier detector, which depend only on the ranks of the scores. This creates a challenge in quantifying the bias-variance trade-off for outlier analysis because the uniqueness of the score-based output is lost. The traditional notion of the bias-variance trade-off works with the mean-squared error (MSE). A quantification like the MSE is designed only for dependent variables with a clear mathematical interpretation of their absolute value (rather than only a relative interpretation). Measures such as the ROC AUC provide only an incomplete interpretation of the scores (in terms of relative ranks). It is possible to work with crisper definitions of the scores that allow the use of more conventional error measures like MSE. One such approach, which preserves uniqueness of scores, is that the outlier detectors always output standardized scores with zero mean, unit variance, and a crisp probabilistic interpretation. Note that one can always apply [31] a standardization step as a post-processing phase to any outlier detector without affecting the ROC; this also has a natural probabilistic interpretation (discussed below).

Consider a data instance denoted by $\overline{X_i}$, for which the outlier score is modeled using the training data $\mathcal{D}$. It is assumed that all training data points are generated from some base distribution, which is unavailable to the analyst in practice. We can assume that an ideal outlier score y_i exists for this data point, even though it is unobserved. The ideal score is output by an unknown function $f(\overline{X_i})$, and it is assumed that the scores that are output by this ideal function also satisfy the zero mean and unit variance assumption over all possible

points generated by the base data distribution:

$$y_i = f(\overline{X_i}) \tag{6.5}$$

The interpretation of the score y_i is that by applying the (cumulative) standard normal distribution function to y_i, we obtain the relative outlier rank of $\overline{X_i}$ with respect to all possible points generated by the base data distribution. In a sense, this crisp definition directly maps the score y_i to its (percentile) outlier rank in $(0, 1)$. In this sense, $f(\overline{X_i})$ is like an oracle that cannot be computed in practice; furthermore, in unsupervised problems, we do not have any examples of the output of this oracle.

This score y_i can be viewed as the analog to a numeric class variable in classification and regression modeling. In problems like classification, we add an additional term to the right-hand side of Equation 6.5 corresponding to the *intrinsic noise* in the dependent variable. However, unlike classification, where the value of y_i is a part of the *observed* data for training points, the value y_i in unsupervised problems only represents a theoretically ideal value (obtained from an oracle) which is *unobserved*. Therefore, in unsupervised problems, the labeling noise[1] no longer remains relevant.

Since the true model $f(\cdot)$ is unknown, the outlier score of a test point $\overline{X_i}$ can only be estimated with the use of an outlier detection model $g(\overline{X_i}, \mathcal{D})$ using base data set $\mathcal{D}$. The model $g(\overline{X_i}, \mathcal{D})$ is only a way of approximating the unknown function $f(\overline{X_i})$, and it is typically computed algorithmically. For example, in k-nearest neighbor outlier detectors, the function $g(\overline{X_i}, \mathcal{D})$ is defined as follows:

$$g(\overline{X_i}, \mathcal{D}) = \alpha \text{KNN-distance}(\overline{X_i}, \mathcal{D}) + \beta \tag{6.6}$$

Here, α and β are constants which are needed to standardize the scores to zero mean and unit variance in order to respect the constraint on the absolute interpretation of the outlier scores. It is important to note that the k-nearest neighbor distance, α, and β depend on the specific data set $\mathcal{D}$ at hand. This is the reason that the data set $\mathcal{D}$ is included as an argument of $g(\overline{X_i}, \mathcal{D})$.

If the function $g(\overline{X_i}, \mathcal{D})$ does not properly model the true oracle $f(\overline{X_i})$, then this will result in errors. This is referred to as *model bias* and it is directly analogous to the model bias used in classification. For example, the use of k-nearest neighbor algorithm as $g(\overline{X_i}, \mathcal{D})$, or a specific choice of the parameter k, might result in the user model deviating significantly from the true function $f(\overline{X_i})$. Similarly, if we use a linear outlier detection model to score a setting in which the normal data points are arranged in a nonlinear spiral (like Figure 5.7 of Chapter 5), the model will have a high level of bias. A second source of error is the *variance*. The variance is caused by the fact that the outlier score directly depends on the *specific instantiation* of the data set $\mathcal{D}$ at hand. Any data set is a finite set (assumed to be drawn from some unknown base distribution). Different instantiations of these draws will cause different results for the same outlier detection algorithm. Even if the *expected* value of $g(\overline{X_i}, \mathcal{D})$ correctly reflects $f(\overline{X_i})$, the estimation of $g(\overline{X_i}, \mathcal{D})$ with limited data would likely not be exactly correct because of these varied results from different draws. If the data set $\mathcal{D}$ is relatively small, then the variance in the estimation of $g(\overline{X_i}, \mathcal{D})$ will be significant and it will have detrimental results on the accuracy. In other words, $g(\overline{X_i}, \mathcal{D})$ will not be the same as $E[g(\overline{X_i}, \mathcal{D})]$ over the space of various random choices of training data sets $\mathcal{D}$. This phenomenon is also sometimes referred to as *overfitting*. The model variance is high when

[1] If there are errors in the feature values, this will also be reflected in the hypothetically ideal (but unobserved) outlier scores. For example, if a measurement error causes an outlier, rather than an application-specific reason, this will also be reflected in the ideal but unobserved scores.

the same point receives very different scores across different choices of training data sets drawn from the same base distribution.

Although one typically does not distinguish between training and test points in unsupervised problems, one can easily do so by cleanly separating the points used for model building, and the points used for scoring. For example, a k-nearest neighbor detector would determine the k closest points in the training data for any point $\overline{X_i}$ in the test data. We choose to demarcate training and test data because it makes our analysis intuitively similar to that of classification; however, it can be easily adapted[2] to draw approximately the same basic conclusions.

Let $\mathcal{D}$ be the training data, and $\overline{X_1} \ldots \overline{X_n}$ be a set of n test points whose (hypothetically ideal but unobserved) outlier scores are $y_1 \ldots y_n$. It is assumed that these out-of-sample test points remain fixed over different instantiations of the training data $\mathcal{D}$, so that one can measure statistical quantities such as the score variance. We use an unsupervised outlier detection algorithm that uses the function $g(\cdot, \cdot)$ to *estimate* these scores. Therefore, the resulting scores of $\overline{X_1} \ldots \overline{X_n}$ using the training data $\mathcal{D}$ are $g(\overline{X_1}, \mathcal{D}) \ldots g(\overline{X_n}, \mathcal{D})$, respectively. The mean-squared error, or MSE, of the detectors of the test points over a particular realization $\mathcal{D}$ of the training data is obtained by the averaging the squared errors over different test points:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \{y_i - g(\overline{X_i}, \mathcal{D})\}^2 \tag{6.7}$$

The *expected MSE*, *over different realizations of the training data*, generated using some random process, is as follows:

$$E[MSE] = \frac{1}{n} \sum_{i=1}^{n} E[\{y_i - g(\overline{X_i}, \mathcal{D})\}^2] \tag{6.8}$$

The different realizations of the training data $\mathcal{D}$ can be constructed using any crisply defined random process. In the traditional view of the bias-variance trade-off, one might assume that the data set $\mathcal{D}$ is generated by a hidden process that draws it from a true distribution. The basic idea is that *only one instance of a finite data set* can be collected by the entity performing the analysis, and there will be some variability in the results because of this limitation. This variability will also lead to some loss in the accuracy over a setting where the entity actually had access to the distribution from which the data was generated. To the entity that is performing the analysis, this variability is hidden because they have only one instance of the finite data set. Other unconventional interpretations of the bias-variance trade-off are also possible. For example, one might construct each instantiation of $\mathcal{D}$ by starting with a larger base data set $\mathcal{D}_0$ and use random subsets of points, dimensions, and so on. In this alternative interpretation, the expected value of the mean-squared error is computed over different instantiations of the random process extracting $\mathcal{D}$ from $\mathcal{D}_0$. Finally, one might even view the randomized process of extracting $\mathcal{D}$ from $\mathcal{D}_0$ as a part of the base detector. This will yield a randomized *base detector* $g(\overline{X_i}, \mathcal{D}_0)$, but a fixed data set $\mathcal{D}_0$. Therefore, the random process is now defined with respect to the randomization in base detector, rather than the training data selection process. These different interpretations will provide different bias-variance decompositions (see section 6.3.1). It is important to clearly

[2]It is noteworthy that the most popular outlier detectors are based on distance-based methods. These detectors are lazy learners in which the test point is itself never included among the k-nearest neighbors at prediction time. Therefore, these learners are essentially out-of-sample methods because they do not include the test point within the model (albeit in a lazy way).

define the underlying random process in order to properly analyze the effectiveness of a particular ensemble method. For now, we will work with the conventional interpretation that the random process generates the different training data sets from a base distribution.

Note that even though the training data $\mathcal{D}$ might have different instantiations because it is generated by a random process, the test points $\overline{X_1} \ldots \overline{X_n}$ always remain fixed over all instantiations of the random process. This is the reason that we have chosen to demarcate the training and test data; it allows us to evaluate the effects of changing the training data (with a random process) on the same set of test points. If the predictions of the same test points vary significantly over various instantiations of the random process, we say that the model has high *variance*. On the other hand, if the expected prediction of each test point is poor, we say that the model has high *bias*. The basic idea is to decompose the error of the classifier into these two components.

The term in the bracket on the right-hand side of Equation 6.8 can be re-written as follows:

$$E[MSE] = \frac{1}{n}\sum_{i=1}^{n} E[\{(y_i - f(\overline{X_i})) + (f(\overline{X_i}) - g(\overline{X_i}, \mathcal{D}))\}^2] \tag{6.9}$$

Note that we can set $(y_i - f(\overline{X_i}))$ on the RHS of aforementioned equation to 0 because of Equation 6.5. Therefore, the following can be shown:

$$E[MSE] = \frac{1}{n}\sum_{i=1}^{n} E[\{f(\overline{X_i}) - g(\overline{X_i}, \mathcal{D})\}^2] \tag{6.10}$$

This right-hand side can be further decomposed by adding and subtracting $E[g(\overline{X_i}, \mathcal{D})]$ within the squared term:

$$\begin{aligned} E[MSE] =& \frac{1}{n}\sum_{i=1}^{n} E[\{f(\overline{X_i}) - E[g(\overline{X_i}, \mathcal{D})]\}^2] \\ &+ \frac{2}{n}\sum_{i=1}^{n} \{f(\overline{X_i}) - E[g(\overline{X_i}, \mathcal{D})]\}\{E[g(\overline{X_i}, \mathcal{D})] - E[g(\overline{X_i}, \mathcal{D})]\} \\ &+ \frac{1}{n}\sum_{i=1}^{n} E[\{E[g(\overline{X_i}, \mathcal{D})] - g(\overline{X_i}, \mathcal{D})\}^2] \end{aligned}$$

The second term on the right-hand side of the aforementioned expression evaluates to 0. Therefore, we have:

$$E[MSE] = \frac{1}{n}\sum_{i=1}^{n} E[\{f(\overline{X_i}) - E[g(\overline{X_i}, \mathcal{D})]\}^2] + \frac{1}{n}\sum_{i=1}^{n} E[\{E[g(\overline{X_i}, \mathcal{D})] - g(\overline{X_i}, \mathcal{D})\}^2] \tag{6.11}$$

$$= \underbrace{\frac{1}{n}\sum_{i=1}^{n} \{f(\overline{X_i}) - E[g(\overline{X_i}, \mathcal{D})]\}^2}_{\text{Bias}^2} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} E[\{E[g(\overline{X_i}, \mathcal{D})] - g(\overline{X_i}, \mathcal{D})\}^2]}_{\text{Variance}} \tag{6.12}$$

The first term in the aforementioned expression is the (squared) bias, whereas the second term is the variance. Stated simply, one obtains the following:

$$E[MSE] = \text{Bias}^2 + \text{Variance} \tag{6.13}$$

This derivation is very similar to that in classification, although the intrinsic error term is missing because of the ideal nature of the score output by the oracle. The bias and variance are specific not just to the algorithm $g(\overline{X_i}, \mathcal{D})$ *but also to the random process used to create the training data sets* $\mathcal{D}$.

Although this analysis does not seem to be general because it makes the assumption of zero mean and unit variance on the scores, it holds as long as the outputs of the base detector and oracle have the same mathematical interpretation. For example, we could very easily have made this entire argument under the assumption that both the base detector $g(\overline{X_i}, \mathcal{D})$ and the oracle $f(\overline{X_i})$ directly output the relative ranks in $(0, 1)$. In that case, the ensemble would work with the ranks as the base detector output.

6.3.1 What is the Expectation Computed Over?

Note that the bias-variance condition of Equation 6.13 provides a condition in terms of the expected value over a set of random variables. Therefore, a natural question to investigate is the nature of the random process that generates these random variables. The *traditional* view of the bias-variance trade-off is that the random process draws the training data sets from some base distribution which is unknown to the analyst. In fact, the analyst sees only *one finite instantiation* of the data set and the random process is hidden from her. Therefore, there is a hidden variance of the predicted score of a particular test instance (over the prediction that another analyst may obtain on that test instance with a different instantiation of the training data). This hidden variance increases the error of the ensemble method. If the analyst were given access to the base distribution instead of a single instance of the training data, it is theoretically possible for her to reduce the variance to 0 by repeatedly scoring the test point over different generated instantiations of the training data and then averaging the scores. On the other hand, the bias is inherent to the *choice of the model* used by the analyst. In other words, even if the analyst were given access to the base distribution and could generate an unlimited number of data sets, she would still be left with the bias because of the error in *assumptions* made by the model.

There are, however, other non-traditional views of the bias-variance trade-off that are relevant to other types of ensemble algorithms. It is not necessary to compute the bias and variance over different choices of the training data. One can also choose the random process to define the base detector; for example, an isolation forest or a feature bagging approach makes randomized choices during its execution. In such cases, one can also define the random process in terms of the random choices made by the algorithm, and define a variance over such settings. Such a variance is referred to as *model-centric* variance, whereas the first type of variance is referred to as *data-centric* variance. These different views of the bias-variance trade-off are useful in different settings. For example, some ensemble methods such as subsampling are best explained from a data-centric perspective (i.e., random process draws the data set from an unknown base distribution), whereas other ensemble methods are best explained from a model-centric perspective. A detailed discussion of these issues is provided in [35].

6.3.2 Relationship of Ensemble Analysis to Bias-Variance Trade-Off

Ensemble analysis is a way of combining different models in order to ensure that the bias-variance trade-off is optimized. In general, one can view the prediction $g(\overline{X}, \mathcal{D})$ of a base detector as a random variable, depending on a random process over either the selection of the

base data $\mathcal{D}$, or the construction of the detector $g(\cdot, \cdot)$ itself, which might be randomized. The overall mean-squared error of this random variable is reduced with respect to the unknown oracle output $f(\overline{X})$ by the ensemble process. This is achieved in two ways:

1. *Reducing bias:* Some methods such as boosting reduce bias in classification by using an ensemble combination of highly biased detectors. The design of detectors is based on the performance results of earlier instantiations of the detector in order to encourage specific types of bias performance in various components. The final combination is also carefully designed in a weighted way to gain the maximum advantage in terms of overall bias performance. However, it is generally much harder to reduce bias in outlier ensembles because of the absence of ground truth.

2. *Reducing variance:* Methods such as bagging, bragging, wagging, and subagging (subsampling) [105, 106, 107], can be used to reduce the model-specific variance in classification. In this context, most classification methods generalize *directly* to outlier ensembles. In most of these methods the final ensemble score is computed as an average of the scores of various detectors. The basic idea is that the average of a set of random variables has lower variance. Therefore, the use of the averaging combination can be credited to the original work in classification [98, 105, 266]; however, it was subsequently adopted by most other outlier ensemble methods [344, 367].

The "unsupervised" nature of outlier detection does not mean that bias and variance cannot be defined. *It only means that the dependent variables are not available with the training data, even though an "abstract," but unknown ground truth does exist.* However, the bias-variance trade-off does not rely on such an availability *to the base algorithm.* None of the steps in the aforementioned computation of MSE rely on the need for $g(\overline{X}_i, \mathcal{D})$ to be computed using examples of the output of oracle $f(\cdot)$ on points in $\mathcal{D}$. This is the reason that variance-reduction algorithms for classification generalize so easily to outlier detection.

6.4 Variance Reduction Methods

The general idea in variance reduction is to average the scores obtained from multiple instantiations of a randomized algorithm. The randomization might be caused by randomized choices of the data (as in bagging and subsampling), or it might be caused by randomized choices of the model (as in isolation forests). In each case, the effect of averaging results in variance reduction and therefore provides improved performance. This is a well-known result in the classification domain [617], and it applies directly to outlier ensembles because of similar theoretical foundations in the two problems [32]. In general, *unstable* base detectors, in which the outlier scores/ranks vary significantly between different runs provide the best improvements. However, better *incremental* improvements do not always mean that the final ensemble performance will be better.

In order to show the "typical" relative performance of some randomized base detectors and ensemble combination, we will use an example. A hypothetical example of various randomized detectors together with the area under curve (AUC) of their receiver operating characteristic (ROC) is shown in Figure 6.1. The figure shows the hypothetical AUC values for 200 randomized runs of the base detector for each of three algorithms. A box plot[3] is used to summarize the 200 different AUC values. The ensemble performance is also shown with a square marker, and it generally performs better than most of the base detectors.

[3]See section 2.2.2.3 of Chapter 2 for a description of box plots.

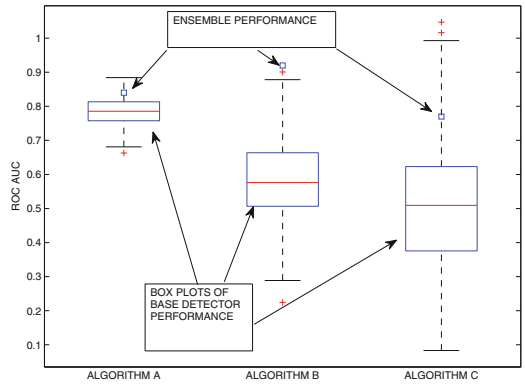


Figure 6.1: Illustrative example of base detector (box plot) and ensemble (square marker) performances for base detectors with varying levels of stability. Relative ensemble performance does not always reflect the relative base-detector performance. Unstable algorithms like algorithms 'B' and 'C' with thicker box plots lead to better incremental improvements.

This is because the base detectors include a variance component, which is significantly reduced in the ensemble combination. Unstable algorithms often[4] have "thicker" box plots (i.e., large inter-quartile ranges between the upper and lower ends of the box). As shown in Figure 6.1, thicker box plots often lead to larger *incremental* improvements. For example, algorithm 'B' has much better incremental improvement than algorithm 'A,' when compared with its median base detector performance. However, larger incremental improvements do not always lead to be the best ensemble performance. Even though algorithm 'C' has a larger incremental improvement than algorithm 'A,' it does not outperform algorithm 'A' in terms of the final ensemble performance. This is primarily due to the poor bias-centric characteristics of algorithm 'C.' Therefore, the overall choice of the base-detector in such methods often depends on a trade-off between the perceived[5] quality of the detector and its stability.

In this section, we will discuss several variance reduction methods for outlier detection. Although some of these methods, such as feature bagging, rotated bagging, and isolation forests, have already been discussed in the chapter on high-dimensional data, we will include discussions on these methods for completeness. We will also provide a slightly different perspective on these methods from the point of view of ensemble analysis. The discussion will also clarify the connections between ensemble analysis and subspace methods for high-dimensional outlier detection.

6.4.1 Parametric Ensembles

The most straightforward application of variance reduction is that of addressing the issue of parameter choice in outlier detection algorithms. For example, in a k-nearest neighbor algorithm, it is often impossible for an analyst to determine the correct value of k for a

[4]Strictly speaking, instability is measured in terms of the outlier scores rather than the AUC. Nevertheless, instability in scores is often reflected in the AUC as well.

[5]In unsupervised problems, it is impossible to measure accuracy in a concrete way because labels are unavailable for computing AUC in real application settings.

Algorithm *ParameterEnsemble*(Data Set: $\mathcal{D}$, Ensemble Components: T);
begin
{ Parameters of algorithm $\mathcal{A}$ are $\theta_1 \ldots \theta_r$ }
$t = 1$;
for each θ_i identify "reasonable" range $[min_i, max_i]$
repeat
for each i select θ_i randomly from $[min_i, max_i]$;
Compute score vector $\mathcal{S}(t)$ by executing algorithm $\mathcal{A}(\mathcal{D}, \theta_1 \ldots \theta_r)$;
Standardize score vector $\mathcal{S}(t)$;
$t = t + 1$;
until$(t = T)$;
return averaged score vector $\frac{\sum_{i=1}^{T} \mathcal{S}(i)}{T}$;
end

Figure 6.2: Using ensembles for reducing parameter-centric uncertainty in unsupervised outlier detection

particular algorithm. This is a particularly vexing problem in unsupervised problems like outlier detection, because no ground-truth is available to validate the effectiveness of a particular choice of parameters. In such cases, the use of a particular choice of parameters is no better than random guessing, and this results in an implicit *model-centric* variance in the output.

A typical ensemble approach [31] is to use a range of different values of k, and then average the scores. Although the overall performance might not be as good as that of using the best value of k, it is usually better than the median performance over the different values of k that are tested. This broad approach is applicable to virtually any parameterized algorithm. In cases, where multiple parameters are available, one might sample the parameters in a "reasonable" range of values, and return the averaged values of the scores. Note that the analyst still has the responsibility of identifying a "reasonable" range of values, although this is usually much easier to estimate (based on data set size and dimensionality) than the setting in which the precise value of a parameter needs to be identified on a particular data set. After the algorithm has been applied over the various instantiations of the parameters, the scores of each point are averaged. The overall approach is illustrated in Figure 6.2. The resulting detector has reduced variance than that of a detector that selects one of these reasonable choices of parameters randomly. It is noteworthy that this particular view of variance reduction is over different randomized instantiations of the base detector rather than over different randomized draws of the training data from the base distribution. Therefore, such an approach reduces model-centric variance.

Although the parametric-ensemble in Figure 6.2 samples over various choices of parameters, this is needed only in cases where the parameter space is too large to try all choices. In cases where the parameter space is small, one might simply run the algorithm over all reasonable choices of parameters and average the scores. Good results have been shown in [184] by averaging the outlier scores from multiple mixture-model clusterings, each of which is generated using a different set of parameters. In general, methods like clustering and histograms are so sensitive to the choice of parameters that it makes sense to use them only in this type of ensemble-centric setting.

Algorithm *FeatureBagging*(Data Set $\mathcal{D}$, Ensemble Components: m);
begin
  repeat
  Sample an integer r from $\lfloor d/2 \rfloor$ to $d-1$;
  Select r dimensions from the data $\mathcal{D}$ randomly to
    create an r-dimensional projection;
  Use base detector on projected representation to compute scores;
  until m iterations;
  Report score of each point as a combination function
    of its m scores from different subspaces;
  { The most common combinations are maximization and averaging }
end

Figure 6.3: Feature Bagging

6.4.2 Randomized Detector Averaging

Many outlier detection models are inherently randomized because they depend on the use of randomly chosen initialization points. For example, if a data point is scored using its distance to the closest centroid of a k-means algorithm, the scores may vary significantly from one execution to the next because of the effect of the initialization point. This is a manifestation of model-centric variance, which reduces the detector accuracy. Therefore, in all such cases, it is extremely important to run the model multiple tries and average the scores in order to reduce the effect of variance on modeling accuracy. In many cases, parametric ensemble methods are paired with this type of randomization to increase diversity. Many outlier detection models, such as density estimation, mixture-modeling, clustering and histograms, are inherently suitable for this type of paired setting. This is because many of these models have parameters and initialization points to which the performance is very sensitive (i.e., high model-centric variance).

6.4.3 Feature Bagging: An Ensemble-Centric Perspective

Feature bagging has been discussed in the context of high-dimensional data in section 5.2.3 of Chapter 5. Here, we provide an ensemble-centric view. The feature bagging method [344] samples different subsets of dimensions. The basic idea is to sample a number r between $\lfloor d/2 \rfloor$ and $d-1$, and then select r dimensions randomly from the data set. The base detector is applied to this lower-dimensional projection. The scores across different base detectors are then combined with the use of either the maximization or the averaging combination function. The former is used in an indirect way by ranking the data points in each component by their scores, and reporting the best rank of each point over all detectors. A pseudocode of the feature bagging method is illustrated in Figure 6.3.

Although feature-bagging might seem like a data-centric ensemble method, it is better explained using a model-centric view of the bias-variance trade-off. Feature bagging (with averaging) is a method that reduces (model-centric) detector variance. Feature bagging with a particular subset of dimensions has a data-centric bias that depends on the selected dimensions. However, if one views the step of randomly selecting the subset of dimensions as a *part* of the component detector, then each such (randomized) detector has exactly the same model-centric bias, and the aforementioned variability in the bias across different

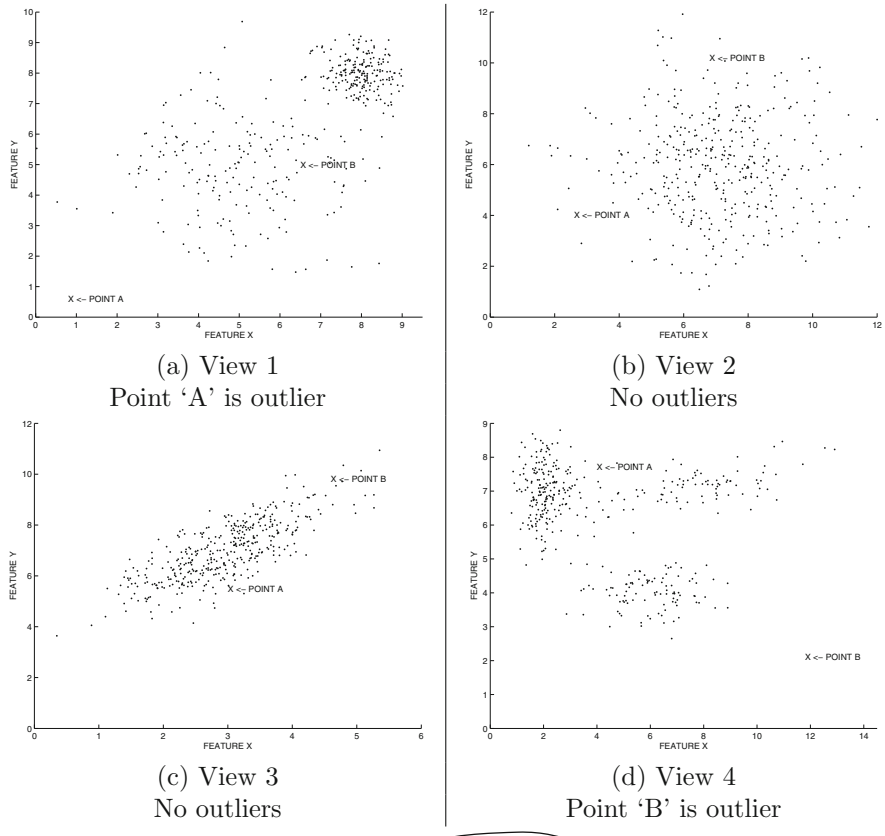
dimension-specific instantiations now becomes a part of this (randomized) detector's model-centric variance. Note that the random process for quantifying the bias-variance trade-off in this case is over the different randomized base detectors, which are applied to a fixed data set $\mathcal{D}$. In such cases, using an average combination is able to achieve variance reduction. The smaller the subset of dimensions that are selected, the greater the variance reduction that is achieved. This is because the underlying detectors tend to be relatively uncorrelated if few overlapping dimensions are selected by different detectors. However, if all dimensions are informative, the bias characteristics of such an approach are likely to work against feature bagging because down-selecting the dimensions will lose information. Therefore, there are subtle trade-offs between the bias and variance with the use of the feature bagging method.

6.4.3.1 Connections to Representational Bias

Even though feature bagging uses global subspaces within each base component, the ensemble combination is implicitly able to discover *locally relevant subspaces* as in subspace methods [4]. The reason for this is that *ensemble methods are often able to discover solutions that are more general than their individual components*. This point can be explained with Dietterich's notion of *representational bias* [170]. For example, we have repeated a pictorial example from Chapter 5 in Figure 6.4. In this case, point 'A' is an outlier in view 1, whereas point 'B' is an outlier in view 4. One can consider each of these views as a hypothesis about the model that best exposes the outliers. For example, the space $\mathcal{H}$ of all hypothesis being considered by the global projections of feature bagging is shown in Figure 6.4(e). However, neither of these hypotheses can expose both outliers 'A' and 'B' because of the inherent representational bias of the space of hypotheses being considered. In fact, the true hypothesis does not even lie in the space of all hypotheses (i.e., hypotheses of global projections) being considered by the feature bagging method. This situation is shown in Figure 6.4(e), where hypothesis h_1 corresponds to view 1, whereas the hypothesis h_6 corresponds to view 4. Because the true hypothesis f lies outside the space of all hypotheses being considered by global projections, one cannot discover both the outliers 'A' and 'B' in a single global projection even if one had access to a data set of infinite size. Nevertheless, the averaging or the maximization functions over different projections are often able to expose both outliers, and are therefore much closer to the true hypothesis f especially if many of these components are able to score these outliers properly.

The averaging function is better able to approximate the true hypothesis by converting the variability in *data-centric* bias across different hypothesis in $\mathcal{H}$ into a *model-centric* variance. This model-centric variance is computed over the randomized process[6] of generating different feature subsets, which can be reduced by averaging. It is noteworthy that variance reduction is a subtle concept and may be computed differently, depending on how one understands the random process over which the bias and variance are computed. By using different random processes, one can obtain different decompositions of the error into the bias and variance. In this specific example, we have shown that it is sometimes possible to use model-centric variance reduction to reduce data-centric (representational) bias by converting the variability of bias (over the different instantiations of the constituent models) into a portion of the model-centric variance. These notions are discussed in greater detail in [35].

[6]Recall that the data-centric bias is computed over the randomized process of generating different training data sets. In traditional bias-variance theory, this is the conventional understanding of bias. However, it is often helpful to interpret the bias-variance theorem in a more generic way over different types of random processes.

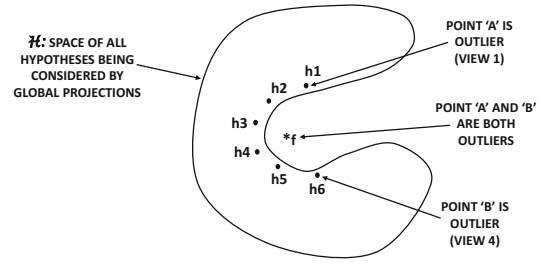


(a) View 1
Point 'A' is outlier

(b) View 2
No outliers

(c) View 3
No outliers

(d) View 4
Point 'B' is outlier



(e) Explaining feature bagging using representational variance [170]

Figure 6.4: The ensemble can reduce representational bias by capturing hypotheses that are more powerful than their individual members. It is possible to use variance reduction on the variability in representational bias.

6.4.3.2 Weaknesses of Feature Bagging

The feature bagging method proposes to always randomly select between $\lfloor d/2 \rfloor$ and $d-1$ dimensions; one does not always gain the best variance reduction by selecting so many dimensions because of correlations among different detectors. Correlations among detectors hinder variance reduction. One might even select the same subset of dimensions repeatedly, while providing drastically worse bias characteristics. In particular, consider a 6-dimensional data set. The number of possible 3-dimensional projections is 20, the number of possible 4-dimensional projections is 15, and the number of 5-dimensional projections is 6. The total number of possibilities is 41. Therefore, most of the projections (and especially the 4- and 5-dimensional ones) will be repeated multiple times in a set of 100 trials, and not much variance can be reduced from such repetitions. On the other hand, the 3-dimensional projections, while more diverse in overlap and repetition, will have deteriorated bias characteristics. This will also be reflected in the final ensemble performance. Here, it is important to note that the most diverse dimensions provide the worst bias characteristics and vice versa. The rotated bagging method [32] solves some of these problems.

6.4.4 Rotated Bagging

The *rotated bagging* method[7] was proposed in [32], in which the data is rotated to a random axis system before selecting the features. By rotating the axis system and using a lower-dimensional projection, it is often possible to expose the outliers in at least some of the projections. The combination function is often able to discover the different outliers in different subspaces. Since real data has significant correlations, one can afford to use a much lower dimensionality than $\lfloor d/2 \rfloor$ to represent the data without losing too much information. In real data sets, the implicit dimensionality usually does not grow much faster than $\sqrt{d}$ with increasing dimensionality d. Therefore, a total of $2 + \lceil \sqrt{d}/2 \rceil$ orthogonal directions are selected from the rotated axis-system as the set of relevant feature bags. The approach is not designed to work for three or less dimensions. Therefore, a constant value of 2 is added up front to prevent its use in such cases. The component detectors will be more uncorrelated in high-dimensional cases, which yields a better opportunity for variance reduction. The overall algorithm works as follows:

1. Determine a randomly rotated axis system in the data.
2. Sample $r = 2 + \lceil \sqrt{d}/2 \rceil$ directions from rotated axis system. Project data along these r directions.
3. Run the outlier detector on projected data.

After running the detectors, the scores can be averaged with a primary goal of variance reduction. However, other schemes such as maximization can also be used. It is important to use standardization on the scores before the combination.

In order to determine the $r = 2 + \lceil \sqrt{d}/2 \rceil$ randomly rotated mutually orthogonal directions, a $d \times r$ random matrix Y is generated, such that each value in the matrix is uniformly distributed in $[-1, 1]$. Let the tth column of Y be denoted by $\overline{y_t}$. Then, the r random orthogonal directions $\overline{e_1} \ldots \overline{e_r}$ are generated using a straightforward Gram-Schmidt orthogonalization of $\overline{y_1} \ldots \overline{y_r}$ as follows:

[7] Although the rotated bagging scheme is described in Chapter 5, we repeat the description here for completeness of this chapter. In the modern publishing era, selective chapter-wise electronic access has become increasingly common.

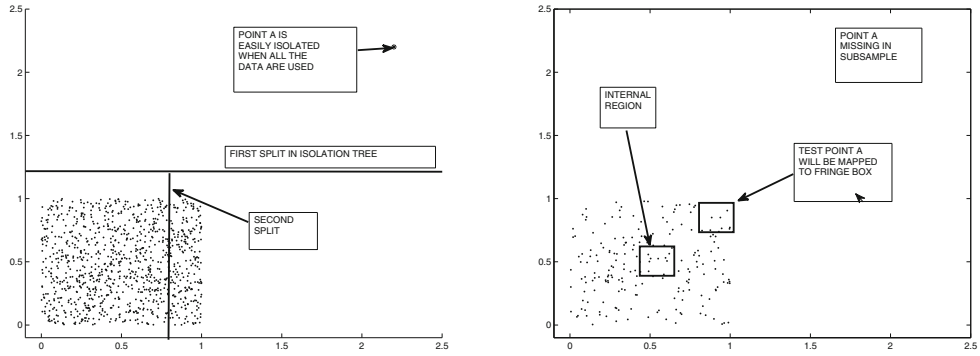
1. $t = 1$; $\overline{e_1} = \frac{\overline{y_1}}{|\overline{y_1}|}$
2. $\overline{e_{t+1}} = \overline{y_{t+1}} - \sum_{j=1}^{t} (\overline{y_{t+1}} \cdot \overline{e_j}) \overline{e_j}$
3. Normalize $\overline{e_{t+1}}$ to unit norm.
4. $t = t + 1$
5. if $t < r$ go to step 2

Let the resulting $d \times r$ matrix with columns $\overline{e_1} \ldots \overline{e_r}$ be denoted by E. The $N \times d$ data set D is transformed and projected to these orthogonal directions by computing the matrix product DE, which is an $N \times r$ matrix of r-dimensional points. The rotated bagging method can be viewed as a subspace outlier detection method in which the subspaces may be defined by any randomly chosen directions, rather than only features from the original space. Because of the difficulty in choosing discriminative features in arbitrary directions (especially for unsupervised problems like outlier detection), it is crucially important to exploit ensemble methods. It has been shown in [32] that rotated bagging can provide more accurate results than feature bagging. Rotated bagging can be explained in a similar way to feature bagging, except that the outliers are discovered in arbitrarily oriented subspaces of the underlying data. The reason that rotated bagging is effective in the context of high-dimensional data is described in section 5.3.3 of Chapter 5.

6.4.5 Isolation Forests: An Ensemble-Centric View

Isolation forests are discussed in detail in section 5.2.6 of Chapter 5 in the context of subspace outlier detection in high-dimensional data. Here, we revisit this discussion in the context of outlier ensembles. An isolation forest is constructed from multiple isolation trees. At a conceptual level, isolation forests also explore random subspaces in the different branches of a tree structure (like feature bagging). The score of a point quantifies the depth required to isolate a point in a single node of the tree with random splits. Outlier points are usually isolated quickly with a few splits. Since different branches of the tree use different splits, isolation forests explore random *local* subspaces unlike feature bagging, which explores a single global subspace in each ensemble component. Furthermore, the scoring is done in a more direct way in isolation trees by quantifying how easy it is to find a local subspace of low dimensionality in which a data point is isolated, and it is not dependent on the vagaries of a particular base detector like LOF (as in feature bagging). As a result, isolation forests are potentially more powerful than feature bagging.

In the most efficient version of isolation trees, a *subsample of the data* is used to recursively partition it with random splits. In each iteration, an attribute is randomly selected and a split point is randomly chosen between the minimum and maximum values of the attribute. The process of splitting is recursively repeated so as to isolate the instances into nodes with fewer and fewer instances until the points are isolated into singleton nodes containing one instance. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are quite different from the normal data. Thus, data points that have noticeably shorter paths in the branches of different trees are more likely to be outliers. Out-of-sample points can also be scored by determining their relevant paths in the tree, based on the univariate split criteria at the various nodes. The process of subsampling adds to the diversity. The final combination step is performed by averaging the path



(a) Data set of 1000 points (b) Subsample of 200 points

Figure 6.5: An outlier test point is mapped to a fringe region when it is not included in the subsample

lengths of the data points in the different trees of the isolation forest. Refer to section 5.2.6 of Chapter 5 for a more detailed discussion of isolation forests.

Although subsampling provides diversity benefits in the isolation forest, there are often some detrimental bias-centric effects of subsampling on the base detectors. These detrimental effects arise as a result of how out-of-sample points are scored in isolation forests. It is noteworthy that out-of-sample points might often map to empty regions in the subsample. From the perspective of the isolation tree (built on the subsample), this empty region might be merged with a normal region of the space, and therefore the score will correspond to the depth of this normal region. Nevertheless, in most cases, the empty regions often map to the *fringes* of normal regions. The fringes of normal regions also tend to be less dense and receive lower depth scores, which is desirable. There are indeed some occasional cases in which scoring strong outliers like fringe regions has detrimental accuracy effects. An example of a data set of 1000 points is illustrated in Figure 6.5(a), in which point 'A' is an obvious outlier. Such an outlier will often be isolated by the very first or second split in an isolation tree when it is included in the training data. However, if the point 'A' is missing from the subsample (cf. Figure 6.5(b)), the structure of the isolation tree will be defined by only the normal points. When the score of point 'A' is computed as a test point, it will be equal to the depth of a fringe box in the isolation tree, as shown in Figure 6.5(b). In cases where the isolation tree is constructed down to a single point, the score of the point 'A' will be equal to that of a fringe point in a cluster. Although fringe points are only weak outliers, the saving grace is that they will often have a lower depth compared to internal points *in expectation*. As a result, the isolation tree can still differentiate between outliers and normal points to a large extent. Nevertheless, it is clear that the outlier score of point 'A' in the out-of-sample setting will often be much weaker (i.e., have larger depth) than it ought to be. Although the approach in [367] is able to gain the diversity improvements from subsampling for some data sets, this improvement is not consistent. This because the diversity gains from the randomized process of tree construction are more significant than those of subsampling, and the difference in the treatment of out-of-sample points from in-sample points affects the subsampling component in a negative way.

The differential treatment of fringe points and interior points causes another problem in isolation forests for some pathological data sets; outliers in sparse but central regions of

the data set are sometimes scored[8] like inliers. In other words, the isolation forest tends to differentially favor specific types of outliers, such as the multivariate extreme values discussed in section 2.3.

6.4.6 Data-Centric Variance Reduction with Sampling

Bagging [98] and subsampling [105, 106, 107] are common methods used for *data-centric* variance reduction in classification. Although these methods were originally designed for the classification domain, they generalize easily to outlier detection because of significant theoretical similarities [32] between the classification and outlier detection domains. In *data-centric* variance reduction, the assumption is that an unknown base distribution exists from which all the data is generated, but the analyst only has access to only a finite sample of the data. Therefore, the prediction results observed by the analyst would be slightly different (i.e., *variant*) from another analyst who had access to a different sample from the same distribution (even if the same algorithm was used in both cases). This difference in results between the two analysts contributes to additional error (beyond modeling errors) in the form of variance. Therefore, the goal is to use this finite resource (data set) as *efficiently* as possible to minimize the variance.

Now consider an ideal setting in which the analyst actually had access to the base distribution, and could go back to it as many times as she wished in order to draw multiple samples. In such a case, each prediction (from a sample) would have a variance that is the second term of Equation 6.12. By drawing the sample an infinite number of times and averaging the results, the analyst could theoretically reduce the contribution of the second term in Equation 6.12 to 0 and therefore reduce the error by *variance reduction*. Unfortunately, however, analysts do not have access to the base distribution in practice. They only have access to a *single finite instance* of the base data. However, it turns out that applying the base detector once on this finite resource is not the most *efficient* way to minimize the variance term. Rather, one should use this finite resource to roughly simulate the aforementioned process of drawing samples from the base distribution. Two examples of such simulations are bagging and subsampling, which are closely related methods. Although such simulations are clearly imperfect (because a finite data set is being used), they reduce the error *most of the time* over a single application of the base detector [98] because of variance reduction. It has been theoretically shown in [98] that such a simulation reduces the error for unstable base detectors. A similar result has been shown in the case of outlier detection in [35].

6.4.6.1 Bagging

In bagging, samples are drawn from the data with replacement. The size of the sample drawn is typically the same as base data, although some variants of bagging do not impose this restriction. Because of the process of sampling with replacement, some points may occur multiple times in the sample, whereas others might not occur at all. Such a sampling technique is also referred to as *bootstrapping*. The outlier detector is applied to each bootstrapped sample. For example, for a k-nearest neighbor detector, the original data is treated as the test data, whereas the k-nearest neighbors are determined from among the boot-strapped sample. For any given test point, care must be taken to exclude its copies in the (bootstrapped) training data while computing its k-nearest neighbors. Each point

[8]For example, a single outlier at the center of normal data points on a unit sphere in 10 dimensions can receive an inlier-like score [35].

is scored multiple times using detectors constructed on different bootstrapped samples and the scores of each point are averaged. The averaged scores provide the final result.

Note that the bootstrapping process is a simulation of the process of drawing data points from a base distribution. However, this simulation is imperfect because of two reasons:

1. One can no longer reduce the variance in the second term of Equation 6.12 to 0 because of correlations (overlaps) among the bootstrapped samples. Therefore, there is a (hidden) residual variance, which is not apparent to the analyst who only has access to a finite resource (data set) rather than the base distribution. The correlations among base detectors are rather large in the case of bagging with bootstrapped sample size equal to data size, and therefore a relatively large part of the variance cannot be reduced. It is often more rewarding to use bagging with smaller re-sample sizes because of less correlations among base detectors (and therefore better variance reduction).

2. Each sample no longer represents the base distribution faithfully. For example, the presence of so many repeated points is a manifestation of this imperfection. This would tend to have a detrimental bias-centric effect, although it is often quite small. However, in some (very occasional) settings, it is possible for these bias-centric effects to overwhelm the variance-reduction advantages.

In general, one can often improve the performance of most detectors with bagging. Some results are presented in [35], which show that bagging can indeed improve detector performance. To maximize the advantages of variance reduction, it is helpful to use unstable detectors, although it is not necessarily helpful to use an unstable base detector with very poor accuracy. After all, the absolute accuracy of the ensemble method is more important than its incremental improvement over the base detectors.

6.4.6.2 Subsampling

Subsampling is frequently used in the classification domain [105, 106, 107] for variance reduction. In subsampling, samples are drawn from the data *without* replacement. Then, each point from the original data set is treated as a test point, and it is scored against the model constructed on the subsample, whether that point is included in the subsample or not. Therefore, much like classification, subsampling distinguishes between the training and test data, although the training data is a subset of the test data. When using an instance-based (i.e., lazy) approach like k-nearest neighbor, care must be taken to exclude the test point from among the k-nearest neighbors, if it is included in the training subsample. This is important to avoid overfitting. However, when using explicit generalization (such as isolation forests and one-class SVMs), it is impossible to exclude a specific test point from the model, which is typically constructed *up-front* using the entire subsample. Therefore, there will always be a small amount of over-fitting for the in-sample points in the test data, as compared to the out-of-sample points. This effect is usually negligible, but it sometimes requires some thought during the design of the prediction process. The scores of each point from different subsample-based models are averaged as in classification [105].

An application of subsampling in outlier ensembles was proposed for the case of graph data[9], and the first subsampling method in multidimensional data was provided in the work on isolation forests [367]. However, the primary focus of the isolation forest was on improving computational efficiency of the ensemble by performing the

[9]The work in [17] subsamples edges of a graph (i.e., *entries* of the adjacency matrix) to create multiple clustering models and scoring the outlier tendency of edges (see section 12.3.2.3 of Chapter 12).

base model construction on a small subsample of size s. It was shown that one could *sometimes* obtain accuracy improvements with subsampling, although such improvements were dependent on the quirks of the underlying data distribution. In many cases, subsampling worsened the accuracy of the isolation forest. An explanation of this behavior is provided in section 6.4.5.

Although the accuracy improvement provided by subsampling is unpredictable in the isolation forest setting, better accuracy improvements have been observed for distance-based detectors [32, 621]. Even in these cases, the results tend to be sensitive to the size of the subsample in a somewhat unpredictable way mainly because the relative performance of the *base detector* varies with subsample size [32]. For example, if an outlier occurs together with a small cluster of other outliers in a particular data distribution, the base k-nearest neighbor detector might avoid the interfering effect of this cluster while scoring outlier points with a smaller subsample. In complex data distributions, larger subsample sizes may be beneficial to the bias performance of the base detector. On the other hand, variance reduction is usually better for smaller subsamples because of less overlaps and correlations among base detectors. The combination of the two effects can often result in an optimal sample size to use for a particular base detector and base distribution, although this optimal sample size cannot be estimated in an unsupervised problem like outlier detection. In practice, the optimal sample size is small for simpler base detectors and data distributions.

The computational benefit is significant for distance-based detectors, because their computational complexity increases superlinearly (quadratically) with the number of points. In the case of a quadratic detector, the complexity of subsampled detectors often depends on the testing phase, which requires $O(N \cdot s)$ time instead of $O(N^2)$ time. In general, subsampling has significant potential to improve the accuracy and efficiency of outlier detectors when implemented properly; a specific example is *variable subsampling* [32].

6.4.6.3 Variable Subsampling

One of the challenging aspects of subsampling is that the performance of a particular base detector is often crucially dependent on the size of the subsample in an unpredictable way. For example, if an original data set contains 1000 data points, and a value of $k = 50$ is used (at 5% of the data size) for a k-nearest neighbor detector, then this same parameter value will be at 50% of the data set size for a subsample of size 100. Clearly, the effects of data size on accuracy will be significant and unpredictable and in some cases fixing parameters can lead[10] to poorer performance with more data. In general, the *optimal* size of the required subsample at a particular parameter setting is unknown; furthermore, it is also impossible to estimate this optimal size in unsupervised problems. This is not as much of a problem when using subsampling in supervised problems like classification because it is possible to use cross-validation to determine the optimal subsample size for a particular parameter setting (or the optimal parameter setting for a particular subsample size).

Even in cases where the algorithm is parameter-free, the optimal size of the required subsample for a particular data set is typically unknown because it depends on the specific interaction between the detector and the data set at hand. Therefore, a natural solution to this dilemma is to use variable subsampling. Note that variable subsampling is more

[10]This observation has repeatedly been misinterpreted by various researchers to suggest (more generally) that one can improve the accuracy of *base* outlier detectors by simply using less data for model-building. As shown in [32], this is an incorrect claim. In general, the main problem is that it is often difficult to make asymptotically optimal parameter or algorithm design choices in unsupervised problems. For example, one does not observe better performance with less data in an exact k-nearest neighbor detector if one adjusts the value of k proportionally with data size.

powerful than simply varying the parameter value for various subsamples, because it also works for parameter-free detectors in which the design choices of the detector might regulate the optimal data size to be used. Parameter-free detectors are often particularly sensitive to subsample size in an unpredictable way.

Let N be the number of points in the base data set $\mathcal{D}$. The algorithm proceeds as follows:

1. Select f uniformly at random between $\min\{1, \frac{50}{N}\}$ and $\min\{1, \frac{1000}{N}\}$, where N is the number of points in the original data set $\mathcal{D}$.

2. Select $f \cdot N$ randomly sampled points from the original data $\mathcal{D}$, and apply the base outlier detector on this sample to create an outlier detection model. Score each point in $\mathcal{D}$ using this model.

At the end of the process, the scores of each data point in different components are averaged to create a unified score. However, before averaging, the N outlier scores from each detector should be standardized to zero mean and unit variance. This standardization is necessary because subsamples of different sizes will create outlier scores of different raw values for unnormalized k-nearest neighbor algorithms. It is noteworthy that the subsampling approach always selects between 50 and 1000 data points irrespective of base data size. For data sets with less than 1000 points, the maximum raw size would be equal to the size of the data set. For data sets with less than 50 points, subsampling is not recommended.

We now analyze the effect of such an approach on parameter choice, by using the k-nearest neighbor algorithm as an example. The merit of this approach is that it effectively samples for different values of model parameters. For example, varying the subsample size at fixed k effectively varies the *percentile value of* k in the subsample. In general, holding data size-sensitive parameters fixed, while varying subsample size, has an automatic effect of parameter space exploration. If we view each component detector *after* selecting the subsample size, then it has a bias, which is component dependent. However, if we view the randomized process of selecting the subsample size as a part of the component detector, then every component has the same bias, and the variability in the aforementioned component-dependent bias now becomes a part of this detector variance. One can reduce this variance with ensembling, with the additional advantage that the underlying component detectors of variable subsampling tend to be far less correlated with one another as compared to fixed subsampling. As a result, one can now aim for better accuracy improvements in the ensemble. Note that variable subsampling *combines data-centric variance-reduction with model-centric variance reduction*. Therefore, this approach provides variance reduction not only over different choices of the training data, but also over different randomized choices of k (in an implicit way). In other words, the approach becomes insensitive to specific parameterizations. However, the approach has implications beyond parametric ensembling because it also provides improvements for parameter-free base detectors, in which the optimal subsample size is not known for a particular algorithm. For example, it has been shown in [367] that certain data-specific training sample sizes enhance outlier discovery for parameter-free methods like isolation trees. Unfortunately, this optimal sample size cannot be estimated for unsupervised problems; therefore, variable subsampling reduces the underlying uncertainty. This makes the *VS* approach more general and desirable than simply varying the values of the parameters across detectors; it is independent of the nature of the parameters/design choices in the base detector and it *concurrently* achieves other forms of variance reduction in an implicit way. For data size-sensitive parameters, it is advisable to select them while keeping in mind that subsample sizes vary between 50 and 1000 points. Knowledge of subsample sizes eases the parameter selection process to some extent. For example, for distance-based

detectors, it is recommended [32] that a value of $k = 5$ will result in a percentile value of k varying between 0.5% to 10% of data size, which seems reasonable. For very simple and stable detectors like (raw) distance-based detectors, it makes sense to use small values of k in combination with small subsample sizes to reduce overlaps between subsamples (and correlations among base detectors). Therefore, one can also use $k = 1$ in combination with proportionally reduced samples sizes between 10 and 200 points [35]. However, this would not be an advisable approach for a more complex detector like the kernel Mahalanobis method, which tries to model more detailed characteristics of the data distribution.

It is noteworthy that variable subsampling works with raw subsample sizes between 50 and 1000, irrespective of base data size. By fixing the subsample size in a constant range, it would seem at first sight that the approach cannot take advantage of the larger base data sizes. This is, however, not the case; larger data sets would result is less overlap across different subsamples, and therefore less correlation across detectors. This would lead to better variance reduction. The idea is to leverage the larger base data size for better de-correlation across detectors rather than build more robust base detectors with larger subsamples; the former is a more efficient form of variance reduction. After all, the number of points required to accurately model a distribution depends on the absolute subsample size, rather than on the size of the original data set obtained by the data collector.

6.4.6.4 Variable Subsampling with Rotated Bagging (VR)

It is possible to combine the base detectors in variable subsampling and rotated bagging to create an even more diverse base detector. This will help in variance reduction. Furthermore, because of the reduction in terms of *both* points and dimensions, significant computational savings are achieved. The combined base detector is created is as follows:

1. Project the data into a random $2 + \lceil \sqrt{d}/2 \rceil$-dimensional space using the rotation method of section 6.4.4.
2. Select a variable size subsample using the approach described in section 6.4.6.3.
3. Score each point using the reduced data set.

The scores of these individual detectors can then be averaged into the final ensemble score. It is important to use Z-score normalization of the scores from the base detectors before combination. This approach is referred to as variable subsampling with rotated bagging (VR).

Rotated bagging has clear computational benefits because one is using only $\sqrt{d}$ dimensions. With increasing dimensionality, the benefit increases. When combined with variable subsampling, the benefits can be very significant. For example, for a data set containing ten million points and 100 dimensions (i.e., a billion entries), each ensemble component would use a data matrix of size at most 1000×7 (i.e., less than ten-thousand entries). In space-constrained settings, this can make a difference in terms of being able to use the approach at all. For 100 trials, the ensemble (containing quadratic base detectors) would be hundreds of times faster than a single application of the base method on the full data.

6.4.7 Other Variance Reduction Methods

The work in [35] discusses a number of other methods for outlier ensembles. Most of these methods have been adapted from their counterparts in the classification domain. Some examples are as follows:

1. **Randomized feature weighting:** The various features of the data are scaled randomly to induce diversity. First, all features are scaled to zero mean and unit variance. Subsequently, the scaling weight w_i of the ith dimension is defined using the following density function $f_X(x)$:

$$f_X(x) = \begin{cases} \frac{\alpha}{x^{\alpha+1}} & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.14}$$

   This density function is the Pareto distribution. The value of α is chosen uniformly at random from $(1, 2)$. Each feature value is multiplied with its scaling weight w_i before applying the base detector. The scores from various ensemble components are combined with the use of an averaging or maximization combination function. The approach can be viewed as a soft version of feature bagging.

2. **Wagging:** In wagging, the various points are weighted by values drawn from a probability distribution. Either the uniform or the Gaussian distribution may be used. However. this approach requires the modification of base detectors to work with weighted points. For example, an average k-nearest neighbor algorithm weights the points while calculating the average distance to the k nearest neighbors. The same is true for the LOF algorithm in which average *reachability* distances need to be computed. Furthermore, LOF divides the reachability distance of a point with the harmonically averaged reachability distance in its locality to compute the final score (cf. Equation 4.4). The harmonic averaging[11] is also done in a weighted way. Therefore, the parametrizations of the algorithms do not change, although the computed values are affected by the weights. The test points are scored with the weighted base detectors. The scores across these different instantiations are then averaged.

3. **Geometric subsampling:** Geometric subsampling [35] is a variant of variable subsampling in which the number of points sampled is selected in a way that provides better diversity for certain types of detectors like distance-based detectors. Instead of always sampling fraction of the data between $\min\{1, \frac{50}{N}\}$ and $\min\{1, \frac{1000}{N}\}$, it samples a value g, which is drawn uniformly at random from $\log_2(\min\{1, \frac{50}{N}\})$ and $\log_2(\min\{1, \frac{1000}{N}\})$. Subsequently, the approach samples a fraction $f = 2^g$ of the data.

   In the following, we briefly describe the main advantage of geometric subsampling over variable subsampling for distance-based detectors. It is noteworthy that more than half the ensemble components in variable subsampling use between 500 and 1000 data points, in which the value of k/N varies only within a factor of 2. This property reduces the diversity of parameter space exploration and the results are often dominated by the samples larger than 500 points. Geometric subsampling increases the diversity of parameter-space exploration by ensuring that there is greater diversity in the value of k/N over different subsamples. As shown in [35], the geometric subsampling approach often provides higher-quality results than variable subsampling.

The first two of these ensemble combination methods have been adapted directly from classification. An experimental comparison of many of these methods is provided in [35]. In general, almost all variance-reduction methods from classification can be adapted to outlier detection because of the similarity of the bias-variance trade-off in the two cases.

[11] Consider a set of relative weights $w_1 \ldots w_k$ within a k-nearest neighbor locality which are scaled to sum to 1. Then, the weighted harmonic mean of $x_1 \ldots x_k$ is given by $1/(\sum_{i=1}^{k} [w_i/x_i])$.

6.5 Flying Blind with Bias Reduction

Bias reduction is a very difficult problem in the context of outlier detection. Note that bias can be viewed as the *inherent* error of a model because of the poor assumptions made in the model. The first term in Equation 6.12 corresponds to the (squared) bias. This term uses the oracle $f(\overline{X_i})$, which is unknown in unsupervised problems like outlier detection. In supervised problems like classification, (possibly noisy) examples of the output of this oracle are available in the form of labels. Because of the presence of $f(\overline{X_i})$ in the bias term, all bias-reduction methods in classification use labels in one form or the other.

A classical example of a bias-reduction method in classification is *boosting*. In this method, the accuracy of a detector on a training instance is used to sequentially readjust its weight in later iterations. Specifically, the weights of incorrectly classified examples are increased. Note that the determination of incorrectly classified examples requires knowledge of the underlying labels. This process is continued until the entire training data is classified with complete accuracy. The final classification of a test instance is performed as a weighted combination of the results from the individual learners. A detailed discussion of boosting is provided in Chapter 7.

Similar methods are, however, hard to construct in outlier detection. This is because the use of the ground-truth is almost always crucial in these cases. Nevertheless, a limited amount of bias reduction can still be *heuristically* achieved in these cases. We emphasize the fact that these methods have much greater uncertainty attached to their performance as compared to variance-reduction methods. This is, in part, because these techniques substitute the outputs of algorithms for the ground-truth, and there is an inherent circularity in making design choices about the algorithm on this basis. The nature of this circularity will become evident from the examples in the subsequent sections.

6.5.1 Bias Reduction by Data-Centric Pruning

A technique that is frequently used to improve outlier detection algorithms but is often not recognized as a meta-algorithm is the technique of iterative outlier removal. The basic idea in this case is that all outlier detection methods assume that the model is constructed on normal points in order to estimate scores. Therefore, the removal of outliers from the training data can sometimes be helpful in improving the correctness of this *assumption* (i.e., improving *bias*). Recall that errors in modeling assumptions cause bias. However, since the knowledge of the ground-truth is not available, how do we know which point to remove? It is here that the natural circularity of bias-reduction algorithms becomes evident. The basic idea here is that we can use the output from a base detector in a conservative way to remove outliers in subsequent iterations. By "conservative" we refer to the fact that we set a high bar for removal of a point. For example, one can convert the outlier score to a Z-value and set a large threshold requirement on the Z-value for removal of the outlier. Therefore, one can use the outlier detection algorithm $\mathcal{A}$ iteratively as shown in Figure 6.6.

The algorithm proceeds by successively refining the outliers from the underlying data $\mathcal{D}$ to keep a current version of a sanitized data set, referred to as $\mathcal{D}_{current}$. In each iteration, an outlier model $\mathcal{M}$ is constructed on $\mathcal{D}_{current}$. For example, if a one-class SVM model is to be constructed, then the data $\mathcal{D}_{curent}$ is used. The data set $\mathcal{D}$ is used as the test data and each point is scored against the model. If a k-nearest neighbor detector is used, then each point in $\mathcal{D}$ is scored using the k-nearest neighbors in $\mathcal{D}_{current}$. Subsequently, the outliers are removed from $\mathcal{D}$ to create the new data set $\mathcal{D}_{current}$. This process is repeated for several iterations as the outlier set is successively refined. The outliers discovered in the

Algorithm *IterativeOutlierRemoval*(Data Set: $\mathcal{D}$)
begin
  $\mathcal{D}_{current} = \mathcal{D}$;
  repeat
    Apply algorithm $\mathcal{A}$ to data set $\mathcal{D}_{current}$ to build model $\mathcal{M}$;
    Score each point in $\mathcal{D}$ using model $\mathcal{M}$ to determine outlier set $\mathcal{O}$;
    $\mathcal{D}_{current} = \mathcal{D} - \mathcal{O}$;
  until convergence or maximum number of iterations;
  return($\mathcal{O}$);
end

Figure 6.6: Iterative Outlier Removal

final iteration of the algorithm are reported.

6.5.2 Bias Reduction by Model-Centric Pruning

In the previous section, it was discussed how bias can be reduced with the use of data-centric pruning. It turns out that bias can also be reduced with the use of model-centric pruning methods. An example of such a method is the *SELECT* technique proposed in [461]. The basic idea is to remove the inaccurate models in the ensemble to improve the overall accuracy. From the perspective of Equation 6.12, one is now trying to remove models for which the first term $\sum_{i=1}^{n}(f(\overline{X_i}) - E[g(\overline{X_i}, \mathcal{D})])^2$ is large. Unfortunately, we do not have access to $f(\overline{X_i})$ in real settings. Therefore, the key idea in [461] is to use a robust ensemble output in lieu of the true ground-truth value $f(\overline{X_i})$ in order to determine the detectors that should be removed. This substituted value is also referred to as the *pseudo-ground truth*. It is noteworthy that the effectiveness of the scheme is crucially dependent on the correctness of the pseudo-ground truth; therefore, at least a reasonable number of base detectors in the original set of models should be accurate.

A very primitive[12] and bare-bones version of the *SELECT* scheme is as follows:

1. Normalize the scores from various detectors to the same scale. Numerous methods might be used for standardization, although the *SELECT* approach uses a technique discussed in [213]. Let the normalized score of the ith point for the jth detector be $O(i, j)$.

2. Compute the average score $a_i = \sum_{j=1}^{m} O(i, j)/m$ of the ith point over all ensemble components.

3. Retain only detectors that correlate well with the pseudo-ground-truth. This step is referred to as the *selection* step.

4. Report the combined score of the remaining detectors as the final score. Examples of such combination functions include the use of averaging or maximum. In principle, any of the combination methods discussed in section 6.6 may be used.

The third step of selecting the detectors requires further explanation. How to select detectors that correlate well with the pseudo-ground-truth? The first step is to compute a *global*

[12]This description is not precisely the same as discussed in [461], although it captures the basic idea behind the approach.

pseudo-ground truth G based on all the detectors. This artificial ground truth G is a vector of scores over all points, such that the score of each point is the average normalized score over all detectors.

First, the detectors are sorted based on their Pearson correlation coefficient of their score vectors with that of the pseudo-ground truth G, and the detector most correlated with the ground-truth is added to an empty set $\mathcal{L}$ to create a singleton ensemble. Subsequently, detectors are added to $\mathcal{L}$ iteratively based on correlation with the *current* ensemble represented by $\mathcal{L}$. The average (normalized) score of each point from the detectors in $\mathcal{L}$ is computed. The remaining detectors (i.e., detectors not in $\mathcal{L}$) are sorted based on their Pearson correlation with the current (normalized) average of the scores in $\mathcal{L}$. The process is repeated by adding the first detector in this ranked list to $\mathcal{L}$. In each iteration, it is also tested whether adding a detector improves the correlation of the ensemble constructed from $\mathcal{L}$ with the *global* pseudo-ground truth G over all detectors. If the correlation increases, then the current detector is added. Otherwise, the process terminates. This approach is referred to as *vertical selection*.

The vertical selection method focuses on *all* the points when computing correlations. The horizontal selection method gives greater importance to the anomalous points while computing the relationships between the various detectors. Therefore, in this case, the binary labeling of points is used rather than the scores. In general, any statistical thresholding on the scores can yield binary labels. A mixture-modeling approach (cf. section 2.4.4 of Chapter 2) is used in order to convert the score list from every detector into a list of binary labels. For each outlier point, its ranks are computed across the different detectors. Detectors in which many (pseudo) ground-truth outlier points are ranked very highly are selected. A method based on order-statistics is used to provide a crisp criterion for this selection. Details may be found in [461]. The basic idea in horizontal selection is that the top-ranked points are more important than the entire list because the anomalies are selected only from the top-ranked points.

6.5.3 Combining Bias and Variance Reduction

A variety of methods can be used to combine bias and variance reduction. Many of these techniques use biased point or dimension sampling methods. Some of these methods are discussed in the context of high-dimensional outlier detection.

1. One can use statistical selection of relevant subspaces rather than using completely randomized methods. Two such methods are the *HiCS* method [308] and the *OUTRES* method [402]. These methods are discussed in sections 5.2.7 and 5.2.8, respectively, of Chapter 5. Both these techniques perform different types of statistical selection of subspaces. For example, the *HiCS* method preselects subspaces based on a deviation test of their non-uniformity. The base algorithm is applied only to the selected subspaces. By using this approach, one is biasing the detector towards sharpened outlier scores. At the same time, the scores from various subspaces are averaged. Therefore, variance is reduced as well. In principle, one can use any other type of feature selection measure such as Kurtosis (cf. section 1.3.1 of Chapter 1) to bias the dimension selection.

2. One can bias the subsampling approach with a computed probability that a data point is an outlier. In other words, an iterative subsampling approach is used in which data points that are scored as outliers have a lower probability of being selected in the next subsample. The scores are then averaged over various iterations to provide the

final result. This approach can be considered a randomized variant of the method in section 6.5.1 and it is discussed in detail in [35].

Bias reduction can also be achieved in the final step of model combination. A variety of such methods are discussed in the next section.

6.6 Model Combination for Outlier Ensembles

Given the outlier scores from various detectors, a final step of ensemble-based approach is to combine the scores from various detectors. It was already discussed in section 6.2, how the scores of different detectors may be combined. As in that discussion, consider a setting in which the (normalized) score for the ith point by the jth detector is denoted by $S_j(i)$. It is assumed that there are N points and m detectors. The two combination functions already discussed earlier in this chapter are as follows:

1. **Averaging:** For the ith data point, the average of $S_1(i) \ldots S_m(i)$ is reported as its final outlier score.

2. **Median:** For the ith data point, the median of $S_1(i) \ldots S_m(i)$ is reported as its final outlier score [17, 35].

3. **Maximum:** For the ith data point, the maximum of $S_1(i) \ldots S_m(i)$ is reported as its final outlier score.

These different types of combination functions have different effects in terms of the bias and variance. The effect of both averaging and the median is very similar by promoting stability, although the averaging function is used more commonly. Several studies [32, 35, 344] have shown that the different combination functions may perform differently in various settings. For some data sets, the averaging function seems to perform better, whereas for other data sets, the maximization function seems to perform better. However, the averaging function seems to be relatively stable because its performance does not seem to vary too much over different instantiations from the same data distribution. The main problem with the maximization function is its lack of stability, and it sometimes makes more sense to use it in combination with rank-centric scores [344] in order to reduce the effects of run-away outlier scores.

In order to understand the nature of these combination functions, one needs to revisit the bias-variance trade-off. It is well-known from the theoretical results in the classification domain [98] that averaging reduces variance. Since the bias-variance trade-off in outlier detection is almost identical that in classification [32], it follows that averaging and the median will also reduce variance in outlier detection.

The effect of the maximization function is far more subtle as compared to the averaging function and is based on a heuristic observation of how outlier detectors often behave on real data sets. In real settings, one is often able to de-emphasize irrelevant or weak ensemble (poorly biased) components with the maximization function. Therefore, one is often able to reduce bias. Even though it might seem at first sight that using a maximization function might overestimate scores, it is important to keep in mind that outlier scores are relative, and one always applies the bias-variance trade-off on a normalized representation of the scores in order to meaningfully use measures such as the MSE with respect to an absolute interpretation of the scores. Therefore, one can no longer talk about overestimation or underestimation of the scores, because *relatively* overestimated scores always need

to be counterbalanced by relatively underestimated scores. The main problem with the maximization function is that it *might* increase variance, especially for small training data sets. The specific effect will depend on the data set at hand; indeed it has been shown in multiple papers [32, 344] that the different combination functions work better for different data sets. As a result of this instability (because of possible variance increase), the accuracy of the maximization function may sometimes fall below the base detector quality. However, in other cases, it can also provide large improvements. At the same time, the instability of the maximization scheme is often a matter of concern and discourages practitioners from using it. Not using the maximization function simply because of its instability leaves large potential improvements on the table. A reasonable solution is to combine the maximization function with variance amelioration in order to provide more accurate results. We will revisit such methods in section 6.6.2.

Next, we explain the bias reduction effects of the maximization combination. In many "difficult" data sets, the outliers may be well hidden, as a result of which many ensemble components may give them inlier-like scores. On the other hand, the variance of inlier points is often far more modest because there is little to distinguish them from their nearby points. In such cases, the scores of outlier points are often *relatively* underestimated in most ensemble components as compared to inlier data points. In order to explain this point, let us consider the feature bagging approach of [344], in which the outliers are hidden in small subsets of dimensions. In such cases, depending on the nature of the underlying data set, a large majority of subspace samples may not contain many of the relevant dimensions. Therefore, most of the subspace samples will provide significant underestimates of the outlier scores for the (small number of) true outlier points and mild overestimates of the outlier scores for the (many) normal points. This is a problem of *bias*, which is caused by the well-hidden nature of outliers. As discussed in [32], such types of bias are inherent to the problem of outlier detection. The scores of outlier points are often far more *brittle* to small algorithm modifications, as compared to the scores of inlier points. Using a maximization ensemble is simply a way of trying to identify components in which the outlier-like behavior is best magnified. At the same time, it needs to be kept in mind that bias-reduction in an unsupervised problem like outlier detection is inherently heuristic, and it might not work for a specific data set. For example, if a training data set (or subsample) is very small, then the maximization function will not work very well because of its propensity of pick out the high variance in the scores.

As discussed in [32], the maximization function often has the ability to pick out non-obvious outliers, which appear in only a few of the ensemble components. In the easy cases in which most outliers are "obvious" and can be discovered by the majority of the ensemble components, the averaging approach will almost always do better by reducing variance effects. However, if it can be argued that the discovery of "obvious" outliers is not quite as interesting from an analytical perspective, the maximization function will have a clear advantage.

6.6.1 Combining Scoring Methods with Ranks

A related question is whether using ranks as base detector output might be a better choice than using absolute outlier scores. After all, the metrics for outlier detection are based on the rank-wise AUCs rather than the score-wise MSEs. Ranks are especially robust to the instability of *raw* scores of the underlying detectors. A specific example of this is the fact that the LOF algorithm often returns ∞ scores because of harmonic normalization (cf. section 4.4.1.1 of Chapter 4). In such cases, the use of ranks has beneficial effects because

they reduce the effects of run-away outlier scores. On the other hand, ranks do lose a lot of relevant information discriminating between various points. In such cases, using ranks could increase bias-centric errors, which might also be manifested in the ranks of the final combination score.

6.6.2 Combining Bias and Variance Reduction

Clearly, the bias-variance trade-off suggests that different combination functions might do better in different settings. The averaging function does better in terms of variance whereas the maximization function often does better in terms of bias. Therefore, it is natural to balance the effort in reducing bias and variance by combining the merits of the two methods. Two such schemes were proposed in [32]. In each cases, it is important to normalize to Z-scores before applying the combination function:

1. **AOM Method:** For m ensemble components, the components are divided into approximately m/q buckets of q components each. First, a maximization is used over each of the buckets of q components, and then the scores are averaged over the m/q buckets. Note that one does not need to assign equal resources to maximization and averaging; in fact, the value of q should be selected to be less than m/q. For example, the implementation in [32] uses 100 trials with $q = 5$. This method is referred to as *Average-of-Maximum*, which is also abbreviated to *AOM*.

2. **Thresh Method:** A method suggested in [31], for combining the scores of multiple detectors, is to use an absolute threshold t on the (standardized) outlier score, and then adding the (thresholded and standardized) outlier scores for these components. The threshold is chosen in a mild way, such as a value of $t = 0$ on the standardized score. Note that values less than 0 almost always correspond to strong inliers. The overall effect of this approach is to reward points for showing up as outliers in a given component, but not to penalize them too much for showing up as strong inliers. The implementation in [32] used a threshold value of $t = 0$ on the Z-score. An important point is that such an approach can sometimes lead to tied scores among the *lowest ranked* (i.e., least outlier-like) points having a score of exactly $m*t$. Such ties are broken among the lowest ranked points by using their average standardized score across the m ensemble components. As a practical matter, one can add a small amount $\epsilon * avg_i$ proportional to the average standardized score avg_i of such points in order to achieve the desired tie-breaking. This approach is referred to as *Thresh*.

The *AOM* combination scheme is particularly useful when the maximum number of trials is not a concern from the computationally efficiency perspective. With simple schemes like averaging the benefits are saturated very quickly. However, to saturate the benefits of combining maximization and averaging (e.g., *AOM*) one would need a larger number of trials. However, it shown in [32] that even with the same number of trials, schemes like *AOM* often do better than averaging. With faster base detectors, one can run a far larger number of trials to gain the maximum accuracy improvements from *both* bias and variance reduction. This implies that there are significant computational advantages in designing efficient base detectors. The *Thresh* method can be viewed as a faster way of combining bias and variance reduction, when computational efficiency is important. Other ideas for combining bias and variance reduction include the use of *Maximum-of-Average (MOA)*.

6.7 Conclusions and Summary

Outlier ensembles have seen an increasing interest from the research community in recent years, which is motivated, in part, by increasingly challenging data sets that are resistant to accurate outlier discovery. An example of such a challenging instantiation is the high-dimensional case in which no single subspace can capture all the outliers. In spite of the superficial differences between the two problems, the theoretical foundations of outlier ensembles are similar to that in the supervised case. Variance-reduction methods are among the most popular techniques used in the context of outlier ensembles. Common variance-reduction techniques include feature bagging, subsampling, and isolation forests. Recently, a number of techniques have also been proposed for bias reduction. Bias-reduction methods are, however, more difficult to implement for outlier detection because of the unavailability of ground truth. The proper combination of the output of outlier detectors is very important for obtaining more accurate results. The most common combination functions include the averaging and the maximization function, which primarily have effects on variance and bias, respectively. These functions can also be combined to benefit from simultaneous bias and variance reduction.

6.8 Bibliographic Survey

Ensemble analysis has a rich history in the field of classification [98, 266, 617] and clustering [516]. The field is, however, far more nascent in the field of outlier analysis. Much of the earliest work in the field of outlier ensembles was motivated by the work on subspace outlier detection [4]. In particular, it has been shown [258] that the first scheme [4] on subspace outlier detection was an ensemble technique in which the maximization combination function was used. However, the approach was not formally claimed as an ensemble method. Soon after, a scheme by [344] proposed the problem of subspace outlier detection more formally in the context of feature bagging. This approach proposed the use of both the averaging and the maximization function, and is therefore credited with the first use of the averaging function in outlier ensembles, although averaging was used earlier in the classification domain for variance reduction. The work in [621] claims to provide a theoretical proof of the averaging function and subsampling effectiveness, which is incorrect; a correct proof based on the connections of the approach to classification is found in [32]. A number of other methods in subspace outlier detection were also proposed for statistical selection of relevant subspaces and their combinations [32, 308, 367, 368, 402, 473]. The isolation forest approach has been modified to discover clustered anomalies [369] and to the case of streaming data [532]. The work in [412] applied ensembles of heterogeneous detectors on high-dimensional data, whereas methods for combining specific types of one-class methods are discussed in [540]. Among the earliest works, there is also an interesting work [213] that shows how to convert scores to probabilities. This is useful for normalizing the outputs of detectors. The rotated bagging work [32] is also notable because of its ability to reduce the dimensionality of the data, and obtain strong outlier detection results by combining the results from many weak detectors. An independently proposed implementation of rotated bagging [436], referred to as *LODA*, uses 1-dimensional rotated projections in combination with histogram-based methods in order to obtain effective results in high-dimensional data. The *RS-Hash* method [476] samples axis-parallel subspace grid regions of varying size and dimensionality to score data points as outliers (cf. section 5.2.5 of Chapter 5).

A position paper describing the key challenges in outlier ensembles was first published

in [31]. This work stated the many challenges associated with outlier ensembles, and the way in which this area is different from classification ensembles. A taxonomy of the field of outlier ensembles is provided in [31] and various model combination methods are also discussed. In this paper, it was also proposed to use bagging techniques [98] for outlier detection, although experimental results were presented much later in a recent book [35].

Bagging is closely related to subsampling, which is used commonly in classification [105, 106, 107]. Entry-wise subsampling of a graph adjacency matrix is proposed in [17]. For multidimensional data, subsampling was proposed in the context of isolation forests [367]; however, there are some detrimental bias-centric effects of this subsampling. The use of subsampling with k-nearest neighbor detectors was proposed in [621]. However, this work provides an incorrect theoretical explanation. The correct theoretical explanation for outlier ensembles is provided in [32]. Many new variance-reduction methods for outlier ensembles are discussed in [35]. One merit of the isolation forest (and other random forest) methods is that they are generally very efficient. As a result, variants of this class of methods have been recently adapted to streaming anomaly detection [532, 571]. Many new variance-reduction methods, such as wagging, randomized feature weighting, and geometric subsampling, have been proposed [35]. These results show that the geometric subsampling approach is an extremely efficient method that works well in practice.

Recently, a number of bias-reduction methods have also been proposed for anomaly detection. As discussed in [32], the use of the maximization combination function is able to provide bias reduction. The *SELECT* scheme was proposed in [461], which proposes bias reduction with a model-centric setting. Another approach for reducing bias in the temporal setting by selecting more relevant models is proposed in [471]. As suggested in [32], one can also perform data-centric bias reduction by removing obvious outliers from the data or by using biased subsampling.

6.9 Exercises

1. Consider the data-centric bias reduction approach in which outliers are iteratively removed. Provide an example of an algorithm and a data set in which the use of such an approach might be counter-productive.

2. Implement the rotated bagging approach to variance reduction.

3. For a given data set, would you expect a k-nearest neighbor detector with a larger value of k to have greater variance, or would you expect a smaller value of k to have greater variance? Why? Which one would typically obtain larger improvement with subsampling at a fixed ratio? Which one would typically obtain a better accuracy?