Mining Outliers with Ensemble of Heterogeneous Detectors on Random Subspaces

Hoang Vu Nguyen, Hock Hee Ang, and Vivekanand Gopalkrishnan

Nanyang Technological University, Singapore

Abstract. Outlier detection has many practical applications, especially in domains that have scope for abnormal behavior. Despite the importance of detecting outliers, defining outliers in fact is a nontrivial task which is normally application-dependent. On the other hand, detection techniques are constructed around the chosen definitions. As a consequence, available detection techniques vary significantly in terms of accuracy, performance and issues of the detection problem which they address. In this paper, we propose a unified framework for combining different outlier detection algorithms. Unlike existing work, our approach combines non-compatible techniques of different types to improve the outlier detection accuracy compared to other ensemble and individual approaches. Through extensive empirical studies, our framework is shown to be very effective in detecting outliers in the real-world context.

1 Introduction

The problem of detecting abnormal events, also called outliers, has been widely studied in recent years [1–3]. Researchers have developed several techniques to mine outliers in static databases and also recently in data streams. Existing outlier detection methods can be classified as distance-based [3–5], density-based [1, 2] and evolutionary-based [6]. There are many ways in practice to define what outliers exactly are, e.g., r-neighborhood Distance-based Outlier [3], k^{th} Nearest Neighbor Distance-based Outlier [5] (a.k.a. k-NN) and Cumulative Neighborhood [4]. Since detection methods are usually constructed around specific outlier notions, their detection qualities vary significantly among datasets. For example, a recent study [7] shows that the Nearest-Neighbor (NN) method performs well when outliers are located in sparse regions whereas LOF [1] performs well when outliers are located in dense regions of normal data. Existing techniques usually compute distances (*in full feature space*) of every data sample to its neighborhood to determine whether it is an outlier or not [1–3, 6]. This causes two side-effects. First, for high-dimensional datasets the concept of locality as well as neighbors becomes less meaningful [8]. Second, not all features are relevant for outlier mining. More specifically, popular distance functions like Euclidean and Mahalanobis are extremely sensitive to noisy features [7]. Despite the presence of the curse of dimensionality, it is difficult in practice to choose a relevant subset of features for the learning purpose [6, 9, 10].

H. Kitagawa et al. (Eds.): DASFAA 2010, Part I, LNCS 5981, pp. 368–383, 2010.

While the nature of data is unpredictable, there is a need for an efficient technique to combine different outlier detection techniques to overcome the drawback of each single method and yield higher detection accuracy. The motivation here is similar to the advent of ensemble classifiers in the machine learning area [9, 11]. With the feasibility of ensemble learning and subspace mining demonstrated, the natural progression would be to combine them both. Lazarevic and Kumar [10] propose the first solution for *semi-supervised* ensemble outlier detection in feature subspace. That work assumes the existence of outlier scores where a *combine* function can be applied directly. However, this is not practically true since different detection methods can produce outlier scores of different *scales*. For example, it can be recognized that the scores produced using k^{th} Nearest Neighbor Distance-based Outlier [5] are smaller in scale than those using Cumulative Neighborhood [4]. Furthermore, as pointed out in Section 3.3, different detection techniques also produce different *types* of score vectors. In particular, some vectors are real-valued while others are binary-valued. This leads to the need of a unified notion of outlier score and an efficient technique to specifically deal with scores' heterogeneity. The availability of such notion would facilitate the task of combination.

Problem Statement. Consider a dataset DS with N data samples in dim dimensions. While most of the data samples in DS are normal, some are outliers, and our task is to detect these outliers. While few outliers can be found when all dimensions are taken into account, most of them can only be identified when looking at some subsets of features. In addition, some features of DS are noisy, and cause the full distance computation to be inaccurate if they are included. Given a set of base outlier detection technique(s), our goal is to build an efficient method to combine the results obtained from them while overcoming their individual drawbacks when applying on DS. The ensemble framework should: (a) alleviate of the curse of dimensionality and noisy features, (b) efficiently combine outlier score vectors of base techniques having different *scales* and different *characteristics*, and (c) provide higher detection quality than each individual base technique used in the ensemble (when applied on full feature space). In order to address this problem, we present the *Heterogeneous Detector Ensemble on Random Subspaces* (HeDES) framework. The advantage of using HeDES lies in its ability to incorporate various heuristics for combining different types of score vectors. The main contributions of this work can be summarized as follows:

- We introduce a unified notion of outlier score function and show how existing outlier definitions can be represented using it. We demonstrate how to identify different types of outlier scores in literature by using this new notion of outlier score function.
- We propose a generalized framework for ensemble outlier detection in feature subspaces - HeDES. Unlike the existing simple framework [10], HeDES is able to combine different techniques producing outlier scores of different scales or even different types of scores (e.g., real-valued v/s. binary-valued).

Through extensive empirical studies, we demonstrate that the HeDES framework can outperform state-of-the-art detection techniques and is therefore suitable for outlier detection in real-world applications. The rest of this paper is organized as follows. Related work and background knowledge are presented in the next section. Details of our approach are provided in Section 3 and empirical comparison with other current-best approaches is discussed in Section 4. Finally, the paper is summarized in Section 5 with directions for future work.

2 Literature Review

Distance-based outlier detection techniques in general exploit the distance of a data sample to its neighborhood to determine whether it is outlier or not. Distances can be computed either using only one neighbor [5] or using k nearest neighbors [4]. The notion of distance-based outlier was first introduced by Knorr and Ng [3] and then refined in [5]. Breunig *et al.* [1] propose the first notion of density-based outliers. The outlier score used, called Local Outlier Factor (LOF), is a measure of difference in neighborhood density of a data sample p and that of data samples in its local neighborhood. LOF for data samples belonging to a cluster is approximately equal to 1, while that for outliers should be much higher. Experimental results from [7] show that LOF outperforms other detection techniques in most cases. Papadimitriou *et al.* [2] introduce a new definition of density-based outliers. Instead of using the k nearest neighbors of a data sample p in computing its outlier score, they employ the r-neighborhood of p. The outlier score of each data sample, called MDEF, is used to compare against the normalized deviation of its neighborhood's scores and standard-deviation is employed in the outlier flagging decision. This removes the need of using any static cutoff or score ranking.

Both distance-based and density-based techniques involve the computation of distances from each data sample to its neighborhood. However, for high-dimensional datasets the concept of locality as well as neighbors becomes less meaningful [8]. This limitation is addressed by an evolutionary-based technique introduced by Aggarwal and Yu [6]. The method first performs a grid *discretization* of the data by dividing each data attribute into $\emptyset$ equi-depth ranges. Then, a genetic approach is employed to mine subspaces whose densities are in the top smallest values. Nevertheless, it suffers the intrinsic problems of evolutionary approaches - its accuracy is unstable and varies depending on the selection of initial population size as well as the crossover and mutation probabilities. The problem of mining in subspaces has also been studied in supervised learning [8, 9]. Ho [9] point out that constructing different classifiers by using randomized initial conditions or data perturbations cannot ensure high classification accuracy. Instead, randomly sampling subsets of feature space (i.e., feature subspaces) for different classifiers seems to be a very promising solution. Likewise, Lazarevic and Kumar [10] tackled the outlier detection problem using an ensemble of outlier techniques built on the problem subspaces. By assuming that information about normal behavior in the underlying dataset is known, they reported findings

similar to that of [9]. Their technique, called Feature Bagging, consists of two variants of combine functions: *Breadth First* and *Cumulative Sum*. The Breadth First combine method (a) first sorts all outlier score vectors, (b) then takes the data samples with highest outlier score from all outlier detection algorithms, and (c) finally appends their indices at the end of the final index vector (and so on). On the other hand, Cumulative Sum simply sums up all the score vectors and returns the result as the final outcome. Nevertheless, Feature Bagging does not specify clearly how to integrate outlier scores with different scales and different characteristics (e.g. *real-valued* vector vs. *binary* vector). Furthermore, Breadth First is reported to be sensitive to the order of detection algorithms applied [10]. Another notion of ensemble outlier mining is presented in [12]. However, like Feature Bagging, no consideration is given to the heterogeneity of outlier scores produced by different techniques. Furthermore, it lacks of details on how to process the score vectors to make its proposed combine functions be applicable whereas a direct application is impossible (c.f. Section 3). In addition, several aspects of the ensemble outlier detection problem (as mentioned in Section 1) are not discussed.

Abe *et al.* [13] propose an approach for constructing an ensemble of dichotomizers for mining outliers using artificially generated data. Their approach, called Active Outlier, first reduces the problem of outlier detection to classification. Active learning (a form of data sub-sampling) is used to construct a set of dichotomizers, combined results of which are used to identify outliers. Active Outlier is indeed a type of ensemble learning using data sub-sampling. As mentioned in [9, 10], building ensembles using data perturbation cannot enrich the homogeneity or de-correlate the relationship among learners in the ensemble as efficiently as feature sub-sampling. Our empirical studies on real-life datasets (c.f., Section 4) support this claim.

3 Methodology

The HeDES framework is a *generalized* framework for mining outliers in subspaces using ensemble of outlier detection techniques (henceforth termed detectors). In the following, we present the details of constructing the ensemble and explain how it is applied in HeDES.

3.1 Ensemble Construction

The process of constructing the ensemble of detectors is displayed in Algorithm 1. In each of the total R rounds, we first sample a detector T from the pool of techniques considered ($\mathcal{T}$) on a *round-robin* basis. Practically, R should be chosen as a multiple of the pool size. Next, we form a subspace S where T will operate by randomly choosing N_f features from the full feature space. Here, N_f is sampled from the uniformly distributed range $[\lfloor dim/2 \rfloor, dim-1]$. The pair (T, S) is then added to the ensemble. By sampling N_f from the range $[\lfloor dim/2 \rfloor, dim-1]$ instead of fixing it to $\lfloor dim/2 \rfloor$ like in [9], we increase the possibility of generating

different subsets of features for each detector in the ensemble. Since the detection capability of each detector relies on its own notion of dissimilarity measure, this increases the chance that they generalize their prediction in ways different to each other. Hence, the above process of constructing the ensemble takes advantage of high-dimensional feature space and weakens the curse of dimensionality.

After identifying all the detectors to be used in the ensemble, we adjust their weights by running the ensemble against an *unlabeled* training set. The intuition behind this weight-adjust is that some detection techniques are more powerful than others on some certain types of data. For example, recent study by Lazarevic *et al.* [7] shows that the Nearest-Neighbor (NN) method outperforms LOF when outliers are located in sparse regions whereas LOF [1] yields higher performance than NN when outliers are located in dense regions of normal data. Even though the detectors in the ensemble are applied on the same dataset during testing, the subspaces where they operate are homogeneous. Furthermore, subspace distributions are different whereas detectors' prediction performance is dependent on their respective subspace. Thus, our argument on detectors' superiority over the others in some certain data still holds in our ensemble learning. Since the nature of subspaces is unpredictable, assigning fixed weights for detectors is not a good solution. Intuitively, had we known which detectors would work better, we would give higher weights to them. In the absence of this knowledge, a possible strategy is to use the result of detectors on a separate validation dataset, or even their performance on the training dataset, as an estimate of their future performance.

Algorithm 1. CONSTRUCTING HEDES

```
for i = 1 to R do
    Choose a detector T_i ∈ 𝒯
    Randomly sample N_f from [⌊dim/2⌋, dim − 1]
    Randomly sample a subset of features S_i of size N_f from the feature set of DS
    Add (T_i, S_i) into the ensemble
Apply the ensemble to the synthetic training dataset
Adjust the weight of each detector in the ensemble
```

This paper, similar to AdaBoost [14], employs the latter strategy. However, since the training set is unlabeled, a direct weight-adjust is not straightforward. To overcome this problem, we construct a labeled synthetic training dataset from the original (unlabeled) one by applying the technique presented in [13]. In brief, the synthetic set is comprised of normal data drawn from the original one and artificially generated outliers. The artificial outliers here are created by using a uniform distribution U that is defined within a bounded subspace whose minimum and maximum are limited to be 10% beyond the observed minimum and maximum, respectively. Let the original training set be S_{tr}, we construct the set of artificial outliers S_{out} of size $|S_{tr}|$ according to U on the bounded domain.

Algorithm 2. MINING OUTLIERS WITH HEDES

```
1  Normalize DS
2  foreach detector type j do
3      TVS_j = ∅
4  for i = 1 to R do
5      Choose the detector (T_i, S_i) from the ensemble
6      j = type of T_i
7      RVS_i = apply T_i to DS projected on S_i
8      TVS_j = TVS_j ∪ {RVS_i}
9  foreach detector type j do
10     VS_j = SUBCOMBINE(TVS_j)
11 VS_FINAL = COMBINE(VS_1, VS_2, ...)
```

The synthetic training set is then set to be $S_{tr} \cup S_{out}$. More details are given in [13]. The use of this set helps us estimate the performance of each detector in the ensemble and adjust its weight correspondingly despite the lack of knowledge on anomalous behavior. Since outlier detectors in the ensemble are unsupervised, they are less susceptible to the overfitting problem. In other words, the weights trained are loosely coupled with the synthetic training set. Furthermore, this artificial data generation has been shown to be successful in training highly accurate classifiers [13]. Thus, the weights obtained in the training phase are likely to have very high generalization capability on unseen test data. By using the weight-adjusted scheme, the effect of detection techniques that are not as relevant as the others can be reduced. This becomes even more critical when irrelevant techniques may lead to a significantly wrong assignment of outlier score (c.f., Section 4).

3.2 HeDES Framework

Our proposed approach, HeDES, is described in Algorithm 2, and functions as follows. The testing dataset is passed through the ensemble. For every pair (T, S) in the ensemble, we apply T to DS projected on subspace S and obtain a raw vector score. This raw vector score is stored together with other vector scores generated by the same detector type j in TVS_j. After finishing R rounds, each set of vector scores (vectors in the same set are of the same type) are combined separately using SUBCOMBINE function to yield a vector score VS_j. Finally, the COMBINE function is invoked using all the VS's obtained to produce the final vector score VS_{FINAL}. The interpretation (combination) of VS and VS_{FINAL} depends on the specific combine functions utilized which are explored in detail in Section 3.4. Note that the two most important components in this framework are: (a) the outlier score function, and (b) the (SUB)COMBINE functions. The main difference between the simple subspace ensemble framework in [10] and our generalized framework lies in the multi-staged combine function which allows much more flexible integration among the heterogeneous types of outlier

detection techniques. It is highlighted that similar to other ensemble classifiers [9, 14], ensemble outlier detection method is a *parallel learning* algorithm [10]. Since each round of running is independent of the other, a parallel implementation can be employed for faster learning.

3.3 Outlier Score Function

Assume a metric distance function D exists on DS, using which we can measure the dissimilarity between two arbitrary data samples in any arbitrary subspace. A general approach that has been used by most of the existing outlier detection methods [1, 3, 6] is to assign an *outlier score* (based on the distance function) to each individual data point, and then design the detection process based on this score. The use of the outlier score is analogous to the mapping of multi-dimensional datasets to $\mathbb{R}$ space (the set of real numbers). In other words, we can define the outlier score function (F_{out}) which maps each data sample in DS to a unique value in $\mathbb{R}$. Intuitively, to create an outlier score function, we first identify a set of measurements based on some specified criteria, then define a mechanism g for combining them, and finally generate a function (F_{out}) based on g. Most the existing techniques utilize only a single measurement, i.e., g becomes a *uni-variable* function that is related directly to the only measurement taken into account. With reference to the k-NN [5], let the measurement considered be the distance from a data pattern p to its k^{th} nearest neighbor (D^k), then a possible choice of F_{out} is $F_{out} = g(D^k) = D^k$.

Outlier score function classification. Among existing approaches to outlier detection problem, we can classify F_{out} into *global* and *local* score functions. An outlier score function is called *global* when the value it assigns to a data sample $p \in DS$ can be used to compare globally with other data samples. More specifically, for two arbitrary data samples p_1 and p_2 in DS, $F_{out}(p_1)$ and $F_{out}(p_2)$ can be compared with each other, and if $F_{out}(p_1) > F_{out}(p_2)$, p_1 has a larger possibility than p_2 to be an outlier. The definitions proposed by Angiulli *et al.* [4], Breunig *et al.* [1], and Ramaswamy *et al.* [5] straightforwardly adhere to this category. On the other hand, the definition of Knorr and Ng [3] can be converted to this category by taking the inverse of the number of neighbors within distance r of each data point. In contrast, a *local* outlier score function assigns to each data sample p, a score that can only be used to compare within some local neighborhood. Example of such a function is proposed in [2], where the local comparison space is the set of data samples lying within the *circle* centered by p and the *radius* is user-defined. The choice of a global or local outlier score function clearly affects later stages of the algorithm design process.

A classification of detection techniques using F_{out}. Using the notion of F_{out} defined above, existing outlier detection techniques can be classified into two types: (a) Threshold-based where a local F_{out} is usually used, and (b) Ranking-based where a global F_{out} is employed, (c.f., Definitions 1 and 2, respectively). According to this classification, the methods proposed in [4–6] using global score functions are classified as Ranking-based. On the other hand, LOCI [2] with local

score function is classified as Threshold-based. Although the technique in [3] utilizes a global F_{out}, it is classified as Threshold-based by letting $F_{out}(p) = \frac{1}{|S(p)|}$ and choosing $t = \frac{1}{1-P}$. In this case, a data sample $p \in DS$ is an outlier if $F_{out}(p) > t$, i.e., $F_{out}(p) > \frac{1}{1-P}$. Note that the threshold t in LOCI [2] is dynamic, whereas that of [3] is static (dependant on the pre-defined variable P).

Definition 1. [THRESHOLD-BASED] *Given a (dynamic or static) threshold t, a data sample p is an outlier of DS if $F_{out}(p) > t$.*

Definition 2. [RANKING-BASED OR TOP-n-OUTLIER] *Given a positive integer n, a data sample p is an n^{th} outlier of DS if no more than $n-1$ other points in DS have a higher value of F_{out} than p. An algorithm based on this definition outputs the top n outliers.*

When F_{out} is global, a Ranking-based technique is normally preferred since the assigned score values of data samples can be compared globally to produce the top points with largest scores. The resultant score vector is then real-valued and identical to the values that F_{out} assigns to data samples. On the other hand, if F_{out} is a local one, a Threshold-based approach becomes a reasonable choice. As a consequence, the score vector obtained contains only binary values (0 for non-outliers and 1 for outliers) since the scores produced by F_{out} are already *discretized* through a threshold-based test. Therefore, score vectors produced by different detection techniques are heterogeneous and need to be processed carefully to facilitate the COMBINE process.

Issue of converting F_{out} to the posterior probabilities. Assume by applying an outlier detector T with outlier score function F_{out} onto DS, we obtain the score vector: $RVS = \{F_{out}(p_1), F_{out}(p_2), \ldots, F_{out}(p_N)\}$. The problem of outlier detection is equivalent to a binary classification problem with two classes: O (outlier class) and M (normal class). One important question which has not been addressed well by the research community is how to compute the posterior probability $P(O|F_{out}(p_i))$ using the knowledge on RVS. Gao and Tan [15] propose two methods attempting to solve this problem. The first method bases on the assumption that the posterior probabilities follow a logistic sigmoid function and the normal and anomalous samples have similar forms of outlier score distribution (same covariance matrix). It then tries to learn the function's parameters using RVS. The second learner on the other hand models the likelihood probability distributions $P(F_{out}(p_i)|O)$ and $P(F_{out}(p_i)|M)$ as a Gaussian and an exponential distribution, respectively. The posterior probabilities are then computed using Bayes theorem. Among the two methods, mixture modeling is more suitable for ensemble learning as demonstrated in [15].

The main intuition leading to this mixture model is derived from the empirical studies using k-NN [5] as the score function. However, the argument used in [15] does not hold for density-based approaches, such as LOF, where density of a data sample is compared (divided) to that of its neighbors. Because of limited space, we omit the demonstration here. Our empirical studies (c.f., Section 4) point out that processing the outlier scores directly (like in HeDES and Feature Bagging) instead of converting to posterior probabilities will yield better detection results.

3.4 COMBINE Functions

As discussed in Section 2, Lazarevic and Kumar [10] introduce two combine functions (Cumulative Sum and Breadth First) which have been successfully used in ensemble-based outlier mining. Here, we present three novel combine functions which are *Weighted Sum*, *Weighted Majority Voting* and *OR Voting*. Unlike Breadth First, these functions are invariant to the order of the detectors. Since accuracy is the most critical factor in ensemble learning, this property becomes an advantage of our approach. Among them, the first two functions are shown to be very efficient in ensemble classification and have been widely employed in many practical applications [14, 16]. The intuition for utilizing weighted combine functions were also discussed in details above. Weighted Majority Voting is known to excel in combining class labels assigned by different classifiers in the ensemble. On the other hand, Weighted Sum in classification is normally applied on posterior probabilities [9]. Conversely, in HeDES, it is used to combine *normalized* outlier scores produced by different detectors of the ensemble. Finally, Or Voting is a natural combine function for integrating heterogeneous types of output scores as demonstrated later. It is important to note here that exploring all possible combine functions is not a focus in this paper. Nevertheless, our chosen combine functions are still able to encompass almost all available types of outlier scores in the field.

Although HeDES provides an easy extension to score vectors of various types (depending on the purpose of learners), in this paper score vectors are either real-valued or binary-valued. An natural approach (Ensemble Voting) to combine different types of score vectors is to simply normalize and *discretize* the real-valued score vectors (convert all score vectors to the same type), and thereafter integrate all the binary-valued score vectors (inclusive of the discretized real-valued score vectors) using Weighted Majority Voting. However, such a natural approach is not sufficient and does not produce good results (c.f., Section 4). The set of input score vectors to the (SUB)COMBINE function is classified into two groups in which the first group contains score vectors (TVS_R) resulting from applying Ranking-based techniques, whereas the second group contains score vectors (TVS_T) of Threshold-based ones. Our strategy is to apply some combine function on TVS_R and TVS_T separately to obtain VS_R and VS_T. Finally, a special combine function is used to integrate VS_R and VS_T to produce the final score vector VS_{FINAL}. It is noted that the problem of combining results of Ranking-based and Threshold-based techniques here is very similar to the problem of combining detection results of categorical and continuous features in mixed-attribute datasets as addressed in [17]. In both cases, we process real values and binary/categorical values separately. Eventually, a heuristic is used to integrate the results obtained. This is the base intuition for our Or Voting combine function.

Processing outlier score vectors. Because of the different nature between Ranking-based and Threshold-based techniques, outlier score vectors produced by them need different treatments. Assume the data samples in DS are $p_1, p_2, \ldots, p_N$. A detection technique T using a specific score function F_{out} is

applied to identify outliers in DS. We denote T's resultant score vector as $RVS = \{F_{out}(p_1), F_{out}(p_2), \ldots, F_{out}(p_N)\}$.

If T is a Ranking-based technique: Vectors of different Ranking-based techniques may have different scales [5]. Hence, to apply combine functions, real-valued vectors need to have equivalent scale. In other words, normalization is necessary. In HeDES, RVS is normalized using the standardization technique. One of the most important characteristics of this normalization technique is its ability to maintain the detectability of extreme values after performing normalization [18]. As argued in [10], this facilitates combining real-valued vectors since a data sample receiving a high score value by one detector, after summing up its score with those produced by other detectors, may still have large values and be flagged as outliers. We define the *normalized* value of $F_{out}(p_j)$ in RVS as: $Score_{norm}(p_j) = \frac{F_{out}(p_j)-m}{s}$ where $m = \frac{1}{N}(\sum_{i=1}^{N} F_{out}(p_i))$ and $s = \frac{1}{N}(\sum_{i=1}^{N} |F_{out}(p_i) - m|)$. By applying normalization, the range of outlier score becomes independent of the technique used. Since all normalized vectors score have comparable scale, it is feasible to integrate them.

If T is a Threshold-based technique: We preserve RVS as it is. This is because each individual element in RVS already indicates the posterior probability of being outlier for data points. Thus, if an ensemble employs techniques from both Ranking-based and Threshold-based, we need a special combine function. Since *Cumulative Sum* and *Breadth First* functions ignore the score vectors' heterogeneity, they are not suitable for use.

Weighted Sum. This function is used for vectors in TVS_R. Let us denote the *weight* of the detector $T_i \in \mathcal{T}$ at round i with score vector RVS_i as W_i. The final score vector of all vectors in TVS_R is defined as: $VS_R = \sum_i W_i \times RVS_i$. Weighted Sum is in fact a modified version of Cumulative Sum proposed in [10]. However, the weight-based strategy helps boost the performance of more efficient detectors. This cannot be obtained in equi-weight schemes.

Weighted Majority Voting. This combine function is used for processing vectors in TVS_T. Although similar to most of the existing ensemble classifiers [9, 14], the problem here is much simpler since we are only interested in two classes of data: normal (class M) and outlier (class O). Since all vectors in TVS_T only contain binary values, they are suitable for Weighted Majority Voting. As in the case of Weighted Sum, the weight of each vector is determined by the performance of the corresponding detection technique on training datasets.

OR Voting. This function is used for combining VS_R and VS_T. However, its input vectors must contain only binary values. Therefore, we perform a *discretization* process on VS_R where its top values are converted to 1, and the rest are converted to 0. Under this scheme, we have: $VS_{FINAL} = VS_R \vee VS_T$ where "$\vee$" is the usual Boolean operator.

Interpretation of VS_{FINAL}. If the pool of detection techniques $\mathcal{T}$ contains only Ranking-based techniques, we then flag those data samples having highest scores in VS_{FINAL} as outliers. In case $\mathcal{T}$ contains only Threshold-based

techniques, outliers are those points having score in VS_{FINAL} equal to 1. Finally, if $\mathcal{T}$ contains both Ranking-based and Threshold-based methods, outliers are those whose scores equal to 1 in VS_{FINAL}. Thus, the flagging mechanism for "mixed" $\mathcal{T}$ is similar to that of an ensemble containing only Threshold-based methods. This is because by applying the OR function, the real-valued vector VS_R is already converted to a binary-valued one. Similar to [10], the number of outliers to flag for Ranking-based methods depends on the specific dataset used.

4 Experimental Evaluation

To verify the effectiveness of the proposed combination framework, we conducted the experiments on several real datasets which are taken from UCI Machine Repository [1]. These datasets are used widely in outlier detection as well as in rare class mining [10], and are summarized in Table 1. The setup procedure (converting datasets into binary-class sets, etc.) employed here follows exactly that of Feature Bagging. In the field of outlier detection, ROC curve (as well as AUC) is an important metric used to evaluate detection quality. Similar to [4, 7, 10, 15], AUC (area under the ROC curve) was chosen as performance benchmark in this paper because of its proved relevance for outlier detection [7, 10]. In each experiment, due to space limitation, we only report how AUC changes when the number of rounds R is varied for KDD Cup 1999 dataset. This dataset is chosen as it has the largest number of instances as well as attributes among all the datasets considered, and hence is a good representative. For other sets, the results are similar and average AUC with $R = 10$ is presented (setting R to 10 was suggested in [10, 15]). For every dataset, each reported result is a 95% confidence interval of the AUC obtained by averaging the outcomes of running the algorithms 10 times on each of its generated binary-class sets. In our empirical studies, two different base detectors are considered: LOF [1] and LOCI [2], and are tested using full feature space. The former is known to be one of the best Ranking-based techniques [7] while the latter is a well-known Threshold-based technique [2]. By choosing these high quality base detectors, we are able to highlight the improvement of HeDES in detection accuracy. For LOF, the parameter $MinPts$ was set to 20. For LOCI, we chose $n_{min} = 20$, $n_{max} = 50$, $\alpha = 1/2$, and $k_\alpha = 3$. Those values were derived from the corresponding papers [1, 2]. Apart from the two base techniques, we compared our approach with other ensemble approaches including: Feature Bagging [10], Active Outlier [13], Mixture Model [15], and Ensemble Voting (c.f., Section 3.4). Feature Bagging uses two combine functions: Cumulative Sum and Breadth First. For each dataset under consideration, we choose to display the highest AUC value among the two for Feature Bagging. Active Outlier constructs an ensemble after t rounds of training, i.e. the ensemble contains t detectors. Here, t was set to R for fair comparison. Since Active Outlier does not use any base detector, its performance remains the same regardless of which base detector is chosen for other ensemble techniques.

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html

Table 1. Characteristics of datasets used for measuring accuracy of techniques

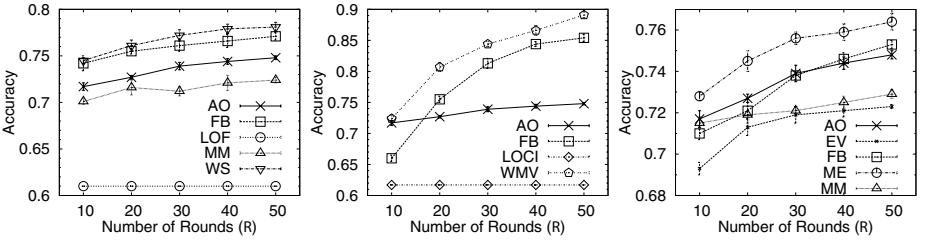
Dataset	Classes	Attributes	Instances	Outlier v/s. Normal
Ann-thyroid 1	3	21	3428	class 1 v/s. 3
Ann-thyroid 2	3	21	3428	class 2 v/s. 3
Lymphography	4	18	148	merged class 2 & 4 v/s. rest
Satimage	7	36	6435	smallest class v/s. rest
Shuttle	7	9	14500	class 2, 3, 5, 6, 7 v/s. 1
KDD Cup 1999	2	42	60839	class U2R v/s. normal
Breast Cancer	2	32	569	class 2 v/s. 1
Segment	7	19	2310	each class v/s. rest
Letter	26	16	6238	each class v/s. rest

Experiment on Ranking-based technique. This experiment aims to investigate the performance of the our proposed combine function, Weighted Sum, when applied to the Ranking-based technique. We compared our method against LOF, Feature Bagging (FB), Mixture Model (MM), and Active Outlier (AO). The results are shown in Figure 1 and Table 2. It can be observed that Weighted Sum strategy yields very good results in all test cases. Even in the case where the base technique, LOF, performs no better than random guessing due to high dimensionality of the dataset (Satimage), our approach is still able to bring very good improvement. The results also indicate that using full feature space in outlier detection may yield low accuracy, especially when the number of features is large and it is likely that some features are noisy. The performance of Mixture Model over the datasets used is worse than Active Outlier and Feature Bagging. This agrees with our argument about the applicability of Mixture Model on other notions of outliers. In particular, the outlier score proposed in LOF is density-based whereas k-NN is distance-based. Extensive studies in the field have pointed out the significant differences between these two notions. These in addition to the results obtained show that the assumption made in Mixture Model is not flexible enough to encompass the scores produced by LOF. For all ensemble techniques considered (including our approach), AUC value increases as the number of detectors included in the ensemble increases. However, Weighted Sum and Feature Bagging tend to work better than AO. This can be attributed to the fact that ensemble learning by subspace sampling produces more efficient learners than data sub-sampling one [9].

Experiment on Threshold-based technique. In this experiment, we study the effect of our proposed combine function, Weighted Majority Voting (WMV), for Threshold-based techniques. Thus, LOCI is selected as the base detector. Our approach's performance is assessed against LOCI, Feature Bagging (FB, also utilizes LOCI), and Active Outlier (AO). Mixture Model is omitted here since the posterior probabilities can be derived directly from the binary-valued scores. In fact, the results achieved by Mixture Model under this setting are the same as that of Feature Bagging. From Figure 1 and Table 3, it can be seen that Weighted Majority Voting yields the best or nearly best results in

Table 2. Ranking-based technique: *AUC* values of LOF, Feature Bagging, Mixture Model, Active Outlier, and Weighted Sum ($R = 10$)

Dataset	LOF	FB	MM	AO	WS
Ann-thyroid 1	0.869	0.869 ± 0.015	0.855 ± 0.021	0.856 ± 0.023	**0.892 ± 0.005**
Ann-thyroid 2	0.761	0.769 ± 0.003	0.759 ± 0.007	0.753 ± 0.009	**0.798 ± 0.008**
Lymphography	0.924	0.967 ± 0.009	0.921 ± 0.001	0.843 ± 0.041	**0.984 ± 0.004**
Satimage	0.510	0.558 ± 0.031	0.562 ± 0.025	0.646 ± 0.024	**0.703 ± 0.022**
Shuttle	0.825	0.839 ± 0.004	0.724 ± 0.017	0.843 ± 0.006	**0.861 ± 0.002**
Breast Cancer	0.805	0.825 ± 0.022	0.758 ± 0.012	0.822 ± 0.015	**0.866 ± 0.017**
Segment	0.820	0.847 ± 0.017	0.798 ± 0.005	0.836 ± 0.002	**0.882 ± 0.003**
Letter	0.816	0.821 ± 0.003	0.722 ± 0.014	0.824 ± 0.002	**0.848 ± 0.001**



(a) Ranking-based (b) Threshold-based (c) Ranking-based & Threshold-based

Fig. 1. *AUC* values of all competing approaches on the KDD Cup 1999 dataset

all cases (the margin with respect to the best one is negligible). For Feature Bagging, neither Cumulative Sum nor Breadth First works well in combining vectors of Threshold-based techniques. This indicates that specialized schemes are required. With the results achieved in this test, Weighted Majority Voting is shown to be a promising candidate.

Overall, we can observe that ensemble outlier detection (Feature Bagging, Weighted Majority Voting, Active Outlier) results in good improvements over the base technique. We again observe the same pattern as in the previous experiment: the accuracy of ensemble techniques grows as the number of detectors increases and that of Active Outlier is dominated by our approach's and Feature Bagging's.

Experiment on Ranking-based & Threshold-based techniques. So far in our empirical studies, the ensemble contains either only Ranking-based (LOF) or only Threshold-based (LOCI) detection techniques. We now investigate our last proposed combine strategy, the OR Voting, in an ensemble where both types of techniques are considered. Therefore, in this experiment, both LOF (Ranking-based) and LOCI (Threshold-based) are employed. We call our method under this setting Mixed Ensemble (ME). More specifically, we use Weighted Sum for Ranking-based technique whereas with Threshold-based technique, we apply Weighted Majority Voting. The results from each group are combined using the OR Voting. Our proposed approach is compared against Feature Bagging

Table 3. Threshold-based technique: *AUC* values of LOCI, Feature Bagging, Active Outlier, and Weighted Majority Voting ($R = 10$)

Dataset	LOCI	FB	AO	WMV
Ann-thyroid 1	0.871	**0.873 ± 0.003**	0.856 ± 0.023	0.872 ± 0.021
Ann-thyroid 2	0.747	0.754 ± 0.026	0.753 ± 0.009	**0.812 ± 0.015**
Lymphography	0.892	0.932 ± 0.007	0.843 ± 0.041	**0.987 ± 0.003**
Satimage	0.529	0.535 ± 0.022	0.646 ± 0.024	**0.654 ± 0.024**
Shuttle	0.822	0.856 ± 0.011	0.843 ± 0.006	**0.873 ± 0.004**
Breast Cancer	0.801	0.827 ± 0.002	0.822 ± 0.015	**0.842 ± 0.001**
Segment	0.835	**0.852 ± 0.002**	0.836 ± 0.002	0.850 ± 0.014
Letter	0.811	0.834 ± 0.016	0.824 ± 0.002	**0.872 ± 0.004**

Table 4. Ranking-based & Threshold-based techniques: *AUC* values of Feature Bagging, Mixture Model, Active Outlier, Ensemble Voting, and Mixed Ensemble ($R = 10$)

Dataset	FB	MM	AO	EV	ME
Ann-thyroid 1	0.870 ± 0.015	0.813 ± 0.013	0.856 ± 0.023	0.832 ± 0.012	**0.883 ± 0.020**
Ann-thyroid 2	0.768 ± 0.031	0.684 ± 0.001	0.753 ± 0.009	0.754 ± 0.012	**0.792 ± 0.004**
Lymphography	**0.955 ± 0.033**	0.735 ± 0.002	0.843 ± 0.041	0.901 ± 0.235	0.952 ± 0.014
Satimage	0.531 ± 0.003	0.517 ± 0.043	0.646 ± 0.024	0.544 ± 0.007	**0.780 ± 0.005**
Shuttle	0.853 ± 0.028	0.729 ± 0.013	0.843 ± 0.006	0.827 ± 0.024	**0.871 ± 0.016**
Breast Cancer	0.824 ± 0.013	0.755 ± 0.023	0.822 ± 0.015	0.837 ± 0.017	**0.864 ± 0.015**
Segment	0.845 ± 0.007	0.792 ± 0.016	0.836 ± 0.002	0.840 ± 0.004	**0.852 ± 0.006**
Letter	0.841 ± 0.004	0.785 ± 0.011	0.824 ± 0.002	0.836 ± 0.003	**0.877 ± 0.018**

(FB), Mixture Model (MM), Active Outlier (AO) and the natural combination approach (Ensemble Voting, a.k.a. EV). Ensemble Voting, similar to ensemble classifier using weighted majority voting (e.g., AdaBoost), is shown to yield very high accuracy in the classification problem [14]. However, through this experiment we point out that it is not very applicable for ensemble outlier detection. For Cumulative Sum of Feature Bagging, we simply sum up all score vectors after performing normalization. The AUC values of all methods are presented in Figure 1 and Table 4. Our approach (Mixed Ensemble) once again performs very well compared to other techniques. The results also show that when an ensemble contains both Ranking-based and Threshold-based techniques, natural sum-up scheme of Cumulative Sum as well as usual ensemble learning based on Weighted Majority Voting does not help much. Instead, we need special combine functions to deal specifically with different types of score vectors.

5 Conclusions and Future Work

In this paper, the problem of ensemble outlier detection in high-dimensional datasets were studied in detail. A formal notion of outlier score which helps to

identify different types of outlier score vectors was introduced. Using the new notion, we presented a heterogeneous detector ensemble on random subspaces (HeDES) framework using different relevant combine functions to tackle the problem of heterogeneity of techniques. Extensive empirical studies on several popular real-life datasets show that our approach can outperform contemporary techniques in the field. In future work, we are considering a systematic extension to test all possible combine functions. Furthermore, we intend to expand the scope of our empirical studies by performing experiments on more large and high-dimensional datasets with different base outlier detection techniques. These will bring us a better understanding about the benefit of ensemble outlier detection for real-world applications. Last but not least, we would like to investigate how the selection of subspaces and base outlier detection techniques affects the detection accuracy of the ensemble.

References

1. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. In: SIGMOD, pp. 93–104 (2000)
2. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: LOCI: Fast outlier detection using the local correlation integral. In: ICDE, pp. 315–324 (2003)
3. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: VLDB, pp. 392–403 (1998)
4. Angiulli, F., Basta, S., Pizzuti, C.: Distance-based detection and prediction of outliers. IEEE Transactions on Knowledge and Data Engineering 18(2), 145–160 (2006)
5. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: SIGMOD, pp. 427–438 (2000)
6. Aggarwal, C.C., Yu, P.S.: An effective and efficient algorithm for high-dimensional outlier detection. VLDB J. 14(2), 211–221 (2005)
7. Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: SDM (2003)
8. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: ICDT, pp. 217–235 (1999)
9. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
10. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: KDD, pp. 157–166 (2005)
11. Kong, E.B., Dietterich, T.G.: Error-correcting output coding corrects bias and variance. In: ICML, pp. 313–321 (1995)
12. He, Z., Deng, S., Xu, X.: A unified subspace outlier ensemble framework for outlier detection. In: Fan, W., Wu, Z., Yang, J. (eds.) WAIM 2005. LNCS, vol. 3739, pp. 632–637. Springer, Heidelberg (2005)
13. Abe, N., Zadrozny, B., Langford, J.: Outlier detection by active learning. In: KDD, pp. 504–509 (2006)
14. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

15. Gao, J., Tan, P.N.: Converting output scores from outlier detection algorithms into probability estimates. In: ICDM, pp. 212–221 (2006)
16. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2003)
17. Otey, M.E., Ghoting, A., Parthasarathy, S.: Fast distributed outlier detection in mixed-attribute data sets. Data Mining and Knowledge Discovery 12(2-3), 203–228 (2006)
18. Hawkins, D.M.: Identification of Outliers. Chapman and Hall, London (1980)