Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Adaptive Sampling and Learning for Unsupervised Outlier Detection

Zhiruo Zhao and **Chilukuri K. Mohan** and **Kishan G. Mehrotra**
zzhao11@syr.edu and mohan@syr.edu and mehrotra@syr.edu
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-4100, USA

Abstract

Unsupervised outlier detection algorithms often suffer from high false positive detection rates. Ensemble approaches can be used to address this problem. This paper proposes a novel ensemble method which adopts the use of an adaptive sampling approach, and combines the outputs of individual anomaly detection algorithms by a weighted majority voting rule in a complete unsupervised context. Simulations on well-known benchmark problems show substantial improvement in performance.

Introduction

Outlier detection algorithms have been used in many applications such as intrusion detection and credit card fraud detection (Chandola, Banerjee, and Kumar 2009). Recently, several kNN based detection algorithms have been proposed, including local density based methods (Breunig et al. 2000)(Jin et al. 2006), and rank based methods (Huang, Mehrotra, and Mohan 2012) (Huang, Mehrotra, and Mohan 2013). Many of these algorithms output an outlier score associated with each object, thresholded to determine whether a point is outlier. Although they were reported to have been successful in capturing anomalies, most such algorithms suffer from high false positive rates, a problem that can be addressed using 'Ensemble' approaches.

A well-known ensemble approach is the 'AdaBoost' supervised classification algorithm (Freund and Schapire 1995), which trains T rounds of $weak\ learners$ over the training set, in each round focusing on the 'hard' examples from the previous round, then combines their output by weighted majority voting. AdaBoost calls a 'booster' in each round to draw a subsample D_t from D with a set of sampling probabilities, initially uniform. A $weak\ learner$ or $base\ method\ h_t$ is trained over D_t, then the probabilities are increased for incorrectly classified examples. After T rounds, each $weak\ learner\ h_t$ is assigned with a weight α_t which is lower if error is higher. The final decision output is made by applying weighted majority voting over all h_t for $t = 1, 2, ..., T$. The training error will be substantially reduced if all the $weak\ learners$ are better than random guessing. AdaBoost has gained considerable popularity for its elegance and performance.

For clustering problems, an Adaptive Clustering algorithm was proposed in (Topchy et al. 2004), with an adaptive sampling ensemble method for better partitioning generation. They use a consistency index to determine the sampling probability and adaptively focus more on points in inconsistent clustering label regions. A consensus function is used for the final decision output, using an agglomerative algorithm applied on the co-association similarity matrix. Empirical study has shown improved performance compared to the classical k-means clustering algorithm.

The application of ensemble methods for unsupervised outlier detection is an emerging research topic, addressed in only a few works so far, such as (Aggarwal 2013; Zimek, Campello, and Sander 2014; Lazarevic and Kumar 2005; Abe, Zadrozny, and Langford 2006; Keller, Müller, and Böhm 2012); of these the work in (Abe, Zadrozny, and Langford 2006) is the only attempt to apply AdaBoost to solve the outlier detection problem. Their proposed method converts the outlier detection problem to an AdaBoost-solvable classification problem by drawing some number of points from an underlying distribution, and marking them as outliers, whereas all the points in the original data set are considered to be inliers. Their resampling method is based on *minimum margin active learning* by iteratively assigning lower sample rejection probabilities to points with low margin values, because they have less consistency. The weights for each classifier are adjusted by the classification error rate in each subsample.

By contrast to the above approaches, this paper proposes a novel adaptive learning algorithm for unsupervised outlier detection which uses the score output of the base algorithm to determine the 'hard' examples, and iteratively resamples more points from such examples in a complete unsupervised context. The method is evaluated on multiple well-known datasets, and our simulations show better results than the base algorithm as well as other existing outlier detection approaches.

The next section describes the adaptive sampling approach. This is followed by discussing the combination method and the proposed algorithm. Experimental simulations and results on benchmark problems are then presented.

Adaptive Sampling

The main idea of adaptive sampling is to give greater emphasis to the examples that are hard to identify. To extract such examples from the data set, we need to answer two questions:

1. How can we determine whether an object cannot be easily classified as an inlier or an outlier?
2. How can we combine the outputs from different iterations?

In this section, we focus on the first question. We first review how classification and clustering ensemble methods can be used to determine the hard-to-identify objects and adjust their sampling weights. We then discuss our proposed adaptive sampling method for outlier detection.

The popular ensemble classification method AdaBoost is adaptive with respect to the training errors of various weak hypotheses, and in each iteration increases sampling probability for points with a high error rate. In the adaptive clustering ensemble approach, clustering consistency is used to estimate the difficulty of assigning a clustering label to an object. Objects near cluster boundaries are assigned higher sampling probabilities, and the boundaries are successively refined in later iterations.

In AdaBoost, the classification decision boundaries are refined gradually by resampling the objects that are more likely to be near the decision boundaries. Adaptive clustering is also focused on cluster boundaries refinement. The outlier detection problem is more difficult: unlike classification and clustering, there is no clear boundary in outlier detection. Instead, most outlier detection algorithms output a score for each object indicating the probability of that object being an outlier; we now discuss our proposed method to use outlier scores to determine the 'decision boundary' in outlier detection adaptive learning.

Justification with local density-based kNN algorithms

Local density based methods consider the outlier score of a point o to be proportional to the density of its k nearest neighbors over the local density of o. The outlier score $Score(o)$ of an object o can be rewritten as follows:

$$Score(o) \propto \frac{DEN_k(N_k(o))}{den_k(o)},$$

where $N_k(o)$ is the set of k nearest neighbors of o, $den_k(o)$ is the local density of o, and $DEN_k(N_k(o))$ is the averaged local density of o's k nearest neighbors. For an inlier i, $Score(i) \approx 0$; for an outlier o, $Score(o) \gg 0$.

Many algorithms have been proposed to solve the outlier detection problem in an unsupervised context, but all have high false positive rates. An algorithm misidentifies an inlier p to be an outlier when p's neighborhood is dense;then $Score(p)$ is high due to other inliers in this neighborhood.

Likewise, if an outlier q is in a sparse neighborhood, other outliers in this neighborhood force the score of q to be small; and therefore q is declared to be an inlier. Our sampling approach aims to sample more from the 'boundary' area, in order to resolve such problems.

In our approach, we remove the obvious inliers and obvious outliers from the dataset. The benefits from such sampling are: (1) removal of the outliers from $N_k(q)$ will increase $den_k(N_k(q))$; (2) removal of the inliers from $N_k(p)$ will decrease $den_k(N_k(p))$. Hence, after such subsampling, the 'boundary' points can be separated better, while the obvious outliers remain identifiable.

Decision Boundary Points

Many outlier detection algorithms assign a score for each object, roughly indicating the relative probability of the object being an outlier (or inlier). Objects with higher scores are considered more likely to be outliers. Most existing outlier detection algorithms perform well on capturing the obvious outliers, but may fail to capture the hard-to-detect outliers. We use the state-of-the-art local density based anomaly detection algorithm LOF (Breunig et al. 2000) in the following as our base detection algorithm.

A simple example is illustrated in Figure 1, where non-anomalous (inlier) data points occur within a two-dimensional Gaussian cluster, outside which three outliers have been inserted: points p, q, and r. We apply the LOF anomaly detection algorithm with $k = 3$, obtain outlier scores for each object, and normalize the scores to be in [0,1]. The mapping of original objects into the score space is shown in Figure 2.

We observe that the object on the left top corner (p) is the most obvious outlier among all of the three outliers, and was assigned the highest outlier score as shown on Figure 2; this illustrates that the detection algorithm performs very well when detecting the easy-to-detect outliers.

Next, we address how to use the score space to distinguish potential outliers and inliers. We use the obvious and often used method: choose a threshold θ such that all objects with an outlier score greater than θ will be identified as outliers and all other objects are marked as inliers; θ determines the decision boundary. In the example of Figure 2, if we choose $\theta = 0.3$, then two real outliers p,q are identified, but five false positives are introduced, and we have one false negative r (undetected outlier). Points near the threshold boundary are more likely to be hard to identify, i.e., whether they

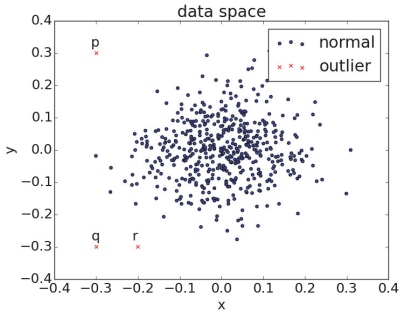


Figure 1: Toy example with 3 outliers

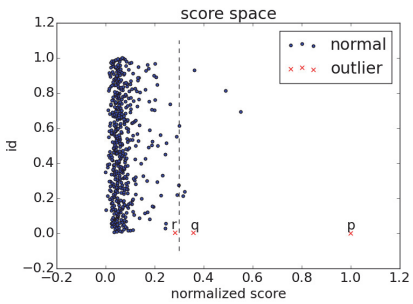


Figure 2: Converted to normalized score space

are true outliers.

Our approach is to re-evaluate such points that are near a decision boundary. In the following, we describe our method of how to adjust the sampling weights.

Sampling Weights Adjustment

As stated above, we consider normalized scores for each object o such that $Score(o) \in [0,1]$; $Score(o) \approx 0$ identifies a clear inlier, $Score(o) \approx 1$ identifies a definite outlier, and $Score(o) \approx \theta$, implies that o is more likely to be a boundary point, where $\theta \in [0,1]$ is an appropriately chosen threshold.

Most false positives and false negatives are expected to occur at boundary points, with existing algorithms. In our approach, a new weight is assigned to each object o, roughly measuring the expectation that it is a boundary point, as follows:

$$WB(o) = \begin{cases} \frac{Score(o)}{\theta} & Score(o) < \theta \\ \frac{1-Score(o)}{1-\theta} & Score(o) \geq \theta. \end{cases} \quad (1)$$

This weight assignment increases sampling probability for boundary points and decreases the sampling probability for points that are easy to detect (clear inliers and outliers). Instead of iteratively refining the boundaries as in classification or clustering problems, we iteratively re-evaluate the points near the boundaries in outlier detection, in order to reduce the number of false positives.

Final Outputs Combination with Different Weighting Schemes

It is very important in ensemble methods to determine how to combine different outputs from all the weak learners (individual modules). A commonly used method is weighted majority voting in which the weight of a learner is assigned by measuring the quality of its results. At iteration t, $t = 1, 2, ..., T$, for object o, if a learner's output is $h_t(o)$, the final output is denoted by:

$$H(o) = \sum_{t=1}^{T} \beta_t h_t(o)$$

where β_t indicates how important the learner h_t is in the final combination. In classification, one can use the trained error as an indication of how well a learner performed. The absence of labeled training data makes it difficult to measure the goodness of a clustering or outlier detection algorithm in an ensemble approach. If the outputs from all the iterations are independent, we can use the correlation between them as an indication of how well one individual learner performs. However, in the adaptive learning process, the input of each learner is dependent on the previous one, and this makes the process of selection of weights (for each learner) even harder. We now propose heuristics for how to assign weights β_t to the different learners and evaluate them empirically.

The problem of measuring how well a learner performs in an iteration is essentially the question of how many 'real' outliers were captured in that iteration. Obviously, the objects with higher scores are more anomalous than the ones with lower scores. But a more pertinent question is that of determining the size of the gap between scores of anomalous vs. non- anomalous objects.

In Figure 3, we show the zoomed histograms of normalized scores for two datasets: (1) with no outlier and (2) for the same dataset with 3 outliers inserted. We observe that the outliers get larger scores, and also the gap in scores between the inliers and outliers increases, in Figure 3a, the 'gap' is from 0.7 to 0.8 while in Figure 3b, the 'gap' is from 0.6 to 0.9.

Using these concepts, we have evaluated three alternative weight assignments for β_t:

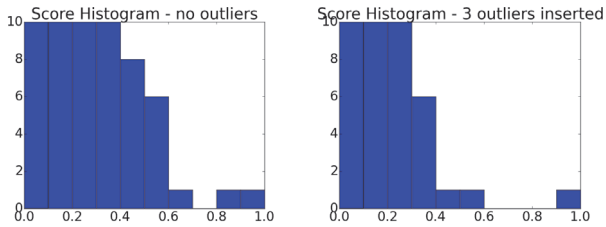
a) The simplest approach is to take the arithmetic average for all values of t. Thus, we assign:
$$\beta_t = 1/T; t = 1, 2, ..., T$$

b) Select a score threshold τ, and at each iteration t, calculate a_t= the number of objects with score greater than τ. Using this, we assign:
$$\beta_t = 1 - \frac{a_t}{|D_t|}$$
where $|D_t|$ is the size of the sample in tth iteration .

c) At each iteration t, obtain the histogram of the score output, calculate the 'gap' b_t between the right-most bin and the second right-most bin, and set:
$$\beta_t = \frac{b_t + 1}{\sum_{t=1}^{T} b_t + 1}.$$

Algorithm

The algorithm begins by giving equal sampling weight to all points in the original dataset D, such that for every point x_i, $w^0_{x_i}$=$1/|D|$. At each iteration, we draw N observations following the sampling distribution $\mathbf{p^t}$, in the set of observations, duplicates are removed and the scores are re-evaluated, this new set of observations is denoted as D_t. We adjust the sampling weights for all the points in D_t as mentioned above, and normalize their sampling weights w.r.t all the sampling weights in D_t; for unsampled data, the sampling weights remain unchanged. This process continues for $t = 1, 2, ..., T$.



(a) Dataset without Outliers (b) Dataset with 3 Outliers

Figure 3: Zoomed Score Histogram Example

The result of our sampling makes the sampling weights for possible boundary points (hard-to-identify points) higher in the following iterations, so the 'effective' sample size will decrease over iterations. To prevent the sample size from reducing too fast, we select $N = 2|D|$ in our experiments. Details of the algorithm (Algorithm 1) are shown below.

Algorithm 1

Input: Dataset D, and detection algorithm A.
Initialize the weight vector: $w^0_{x_i} = 1/|D|$ for each point x_i;
for iteration t = 1,2,...,T:
1. Set:

$$\mathbf{p^t} = \mathbf{w^t}/\sum_i w^t_{x_i}$$

2. Draw N observations from D using $\mathbf{p^t}$, remove the duplicates, denote this set of observations as D_t
3. Run A on D_t, get a score vector $\mathbf{Score^t}$, normalize the scores to [0,1]
4. $h_t(x_i)$= normalized score of x_i
5. Set the new weights vector to be:

$$w^{t+1}_{x_i} = (1-\alpha) * w^t_{x_i} + (\alpha) * WB(x_i);$$

(WB is defined in equation (1))

Output: Make the final decision

$$H(x_i) = \sum_{t=1}^{T} \beta_t h_t(x_i)$$

Experiments and Results

This section describes the datasets used for simulation, simulation results, and discussions of model parameters. In our results, the AUC (Area Under Curve), defined as the surface area under ROC curve, is used to measure the performance of detection algorithms. An ROC curve plots a function between the true positive rate and false positive rate. If the value of AUC is higher for an outlier detection algorithm, then, the algorithm is more capable of capturing outliers while introducing less false positives.

Dataset Description

Five well-known datasets were used in our simulations:

1. **Packed Executables dataset (PEC)**[1] Executable packing is the most common technique used by computer virus writers to obfuscate malicious code and evade detection by anti-virus software. This dataset was collected from the Malfease Project to classify the non-packed executables from packed executables so that only packed executables could be sent to universal unpacker. In our experiments, we select 1000 packed executables as normal observations, and insert 8 non-packed executables as anomalies. All 8 features are used in experiments.
2. **KDD 99 dataset (KDD)**[2] KDD 99 dataset is available from DARPA's intrusion dataset evaluation program. This dataset has been widely used in both intrusion detection and anomaly detection area, and data belong to four main attack categories. In our experiments, we select 1000 normal connections from test dataset and insert 14 attack-connections as anomalies. All 41 features are used in experiments.
3. **Wisconsin Dataset (WIS)**[3] This dataset contains 699 instances and has 9 attributes. There are many duplicate instances and instances with missing attribute values. After removing all duplicate and instances with missing attribute values, 236 instances are labeled as benign class and 236 instances as malignant. In our experiments the final dataset consisted 213 normal instances and 10 malignant instances as anomalies.
4. **Database Basketball dataset (NBA)**[4] This dataset contains Basketball player statistics from 1951-2009 with 17 features. It contains all players statistics in regular season and information about all star players for each year. We construct our outlier detection evaluation dataset by selecting all star players for year 2009, use their regular season stats as the outlier data points, and select all the other players stats in year 2009 as normal points.
5. **Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set (ACT)**[5] This dataset collects data from 30 volunteers from age 19-48 wearing smartphones. The 3-axial linear acceleration and 3-axial angular velocity values from the sensors data are used for classification, with 561 features and 12 classes. To construct our outlier detection evaluation dataset, we consider the class with the least number of instances (sit-to-stand) as outlier points and the class with most number of instances (standing) as inlier points.

Performance Comparisons

We evaluated our ensemble approach with base algorithm LOF for different k values (the number of local neighbors). In Figure 4, we plot AUC for various k values, where the x-axis shows the ratio of k to the size of the dataset. A solid line shows the AUC values for base algorithm LOF, and a boxplot shows the results of our ensemble approach using

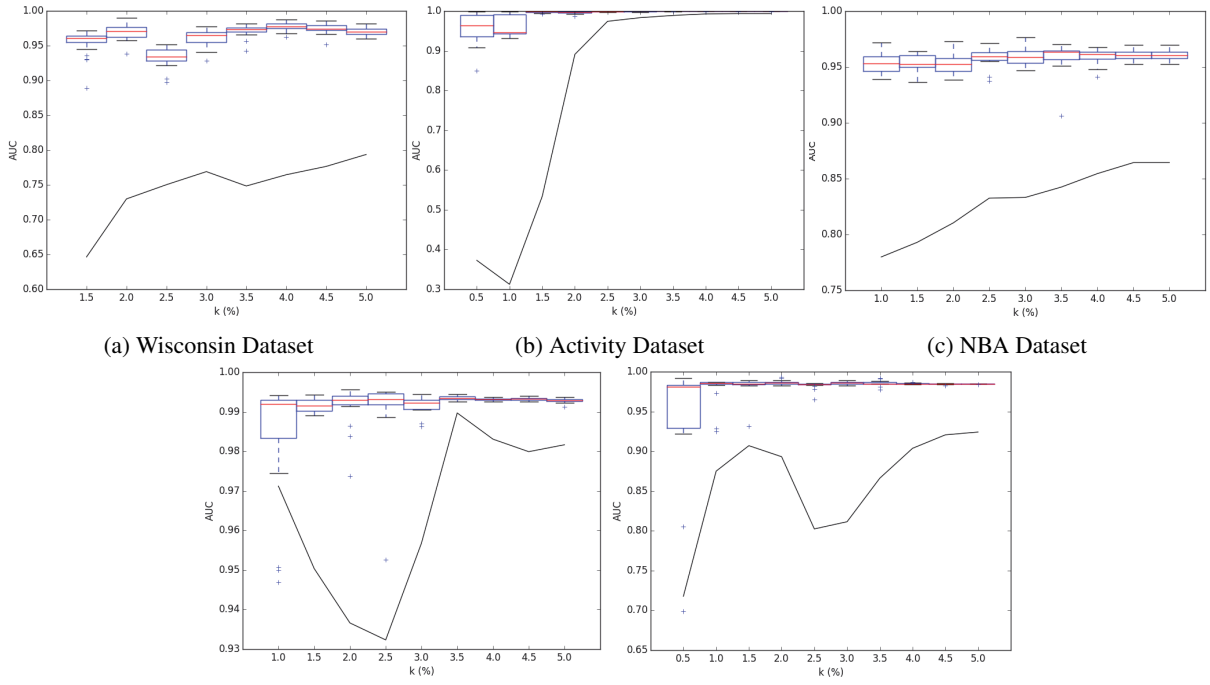
[1] http://roberto.perdisci.com/projects/cpexe

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[3] http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/

[4] http://databasebasketball.com/

[5] http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions



(a) Wisconsin Dataset (b) Activity Dataset (c) NBA Dataset

(d) KDD Dataset (e) PEC Dataset

Figure 4: AUC Performance Comparison with Base Algorithm (LOF) over Different k

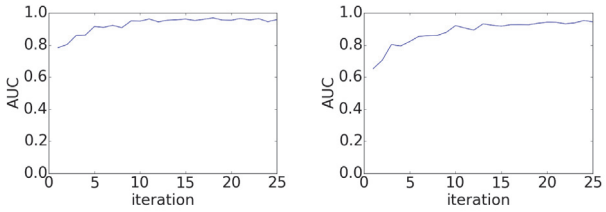
$\beta_t = 1 - \frac{a_t}{|D_t|}$, using 25 iterations in this experiment. Results show that for all k values, our approach outperforms the base algorithm for all the 5 datasets. The ensemble approach has large variations when k is small, which decreases as k increases.

We also compared our proposed ensemble approach with the Active-Outlier approach proposed in (Abe, Zadrozny, and Langford 2006), Feature Bagging approach proposed in (Lazarevic and Kumar 2005), and HiCS (high contrast subspace) outlier detection method proposed in (Keller, Müller, and Böhm 2012). In Active-Outlier approach, (Abe, Zadrozny, and Langford 2006) propose to use AdaBoost with decision tree classifier (CART) as the base classifier, and the synthetic outliers are generated from a specific distribution (Uniform, Gaussian). In Feature Bagging, we use the same number of ensemble members and use LOF as base method. In HiCS, we use LOF as base method. Feature Bagging and HiCS approaches are implemented in ELKI toolbox(Achtert et al. 2009), results for three different k values are reported. In our ensemble approach, we do not add any additional observations; we use LOF as the base outlier detection algorithm with $\alpha = 0.2$, $\tau = 0.95$ and results are reported for three different values of k. For fair comparison, we use the same number of iterations in all methods.

Table 1 presents the averaged AUC for performance over 20 repeats. Active-Outlier approach is denoted as *ActOut*, Feature Bagging is denoted as *FB*, while our proposed ensemble approach is denoted as *Ensemble*. As can be seen that the performance of our proposed ensemble approach is better in almost all cases and increases as k increases.

Effect of Model Parameters

Effect of Number of Iterations: In Fig. 5, we plot the AUC value versus the number of iterations for a synthetic dataset and a real dataset. As we can see, on the synthetic dataset, performance stabilized after 10 iterations and on the real dataset, performance stabilized after 20 iterations.



(a) Synthetic Dataset (b) Real Dataset

Figure 5: AUC Performance vs. Number of Iterations

Effect of Combination Approach: The simulations for comparing different possible combination approaches were set up as follows:

1. For each dataset, for each k value, apply the algorithm with each of the three different combination approaches.
2. Repeat the whole process 20 times, report the mean of AUC.
3. For each dataset, for each k value, rank the AUC performance of each combination approach, so the best one will have a rank of 1, the worst one has a rank of 3.

Table 1: Performance Comparison: Averaged AUC over 20 Repeats

DataSet	ActOut		k=1%			k=3%			k=5%		
	Unif	Gauss	FB	HiCS	Ensemble	FB	HiCS	Ensemble	FB	HiCS	Ensemble
WIS	0.865	0.921	0.446	0.564	**0.963**	0.407	0.539	**0.965**	0.448	0.761	**0.978**
NBA	0.903	0.810	0.822	0.709	**0.920**	0.855	0.864	**0.959**	0.884	0.885	**0.961**
ACT	0.5	**0.980**	0.833	0.922	**0.961**	0.977	**1.000**	0.999	0.993	**1.000**	**1.000**
PEC	0.907	0.686	0.762	0.768	**0.975**	0.775	0.808	**0.986**	0.701	0.894	**0.985**
KDD	0.873	0.685	0.844	0.778	**0.985**	0.780	0.836	**0.993**	0.787	0.795	**0.993**

4. For each dataset, calculate the sum of the above ranks for each combination approach, so the best one has the lowest sum.

In Table 2, we summarize the accumulated sums of AUC rankings over all different k values for the three approaches and denoted as Sum_a,Sum_b, and Sum_c respectively. The one with best detection performance will have the lowest Sum over the three different combination approaches.

We observe that combination approach a ranks once as the best, twice as the second, and twice as the third; combination approach c ranks twice as the best, three times as the third; while the combination approach b ranks twice as the best, three times as the second and none as the third. So we can say that combination approach b, where $\beta_t = 1 - \frac{a_t}{|D_t|}$ outperforms the other two combination approaches over the 5 datasets we considered.

Table 2: Performance over Sum of AUC Rankings with Different Combination Approaches

DataSet	Sum_a	Sum_b	Sum_c
WIS	42	**33**	73
NBA	55	52	**43**
ACT	41	**27**	73
PEC	59	45	**33**
KDD	**33**	41	57

Conclusion

Unsupervised outlier detection is a challenging task with many applications, but widely used algorithms are often prone to generating too many false positives. In this paper, we have proposed a novel adaptive sampling approach which succeeds in addressing this problem. Simulation results on five well-known data sets show the relative success obtained with the new approach, compared to the LOF base algorithm as well as other recent approaches.

References

Abe, N.; Zadrozny, B.; and Langford, J. 2006. Outlier detection by active learning. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06* 504.

Achtert, E.; Bernecker, T.; Kriegel, H.-P.; Schubert, E.; and Zimek, A. 2009. Elki in time: Elki 0.2 for the performance evaluation of distance measures for time series. In *Advances in Spatial and Temporal Databases*. Springer. 436–440.

Aggarwal, C. 2013. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter* 14:49–58.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. Lof: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, 93–104. ACM.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41(3):15.

Freund, Y., and Schapire, R. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. *Computational learning theory* 55:119–139.

Huang, H.; Mehrotra, K.; and Mohan, C. K. 2012. Algorithms for detecting outliers via clustering and ranks. In *Advanced Research in Applied Artificial Intelligence*. Springer. 20–29.

Huang, H.; Mehrotra, K.; and Mohan, C. K. 2013. Rank-based outlier detection. *Journal of Statistical Computation and Simulation* 83(3):518–531.

Jin, W.; Tung, A. K.; Han, J.; and Wang, W. 2006. Ranking outliers using symmetric neighborhood relationship. In *Advances in Knowledge Discovery and Data Mining*. Springer. 577–593.

Keller, F.; Müller, E.; and Böhm, K. 2012. Hics: high contrast subspaces for density-based outlier ranking. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, 1037–1048. IEEE.

Lazarevic, A., and Kumar, V. 2005. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 157–166. ACM.

Topchy, A.; Minaei-Bidgoli, B.; Jain, A. K.; and Punch, W. F. 2004. Adaptive clustering ensembles. *Proceedings - International Conference on Pattern Recognition* 1(i):272–275.

Zimek, A.; Campello, R. J.; and Sander, J. 2014. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter* 15(1):11–22.