Constructing Training Sets for Outlier Detection

Li-Ping Liu Xiaoli Z. Fern
EECS, Oregon State University
{liuli, xfern}@eecs.oregonstate.edu

Abstract

Outlier detection often works in an unsupervised manner due to the difficulty of obtaining enough training data. Since outliers are rare, one has to label a very large dataset to include enough outliers in the training set, with which classifiers could sufficiently learn the concept of outliers. Labeling a large training set is costly for most applications. However, we could just label suspected instances identified by unsupervised methods. In this way, the number of instances to be labeled could be greatly reduced. Based on this idea, we propose CISO, an algorithm Constructing training set by Identifying Suspected Outliers. In this algorithm, instances in a pool are first ranked by an unsupervised outlier detection algorithm. Then, suspected instances are selected and hand-labeled, and all remaining instances receive label of inlier. As such, all instances in the pool are labeled and used in the training set. We also propose Budgeted CISO (BCISO), with which user could set a fixed budget for labeling. Experiments show that both algorithms achieve good performance compared to other methods when the same amount of labeling effort are used.

1 Introduction

Outlier detection[5][14], also referred to as anomaly detection, is a technique of detecting patterns in data that do not conform to the expected behavior [5]. Techniques of outlier detection fall into three categories: detecting with no prior knowledge of the data(unsupervised), modeling both normality and abnormality(supervised), and modeling only normality (semi-supervised) [14]. Most algorithms developed in recent years are unsupervised or semi-supervised. Although supervised methods often have good performances in outlier detection [4], their application is limited due to the difficulty of obtaining sufficient training data.

The difficulty of constructing training datasets for supervised outlier detection lies in the rareness of outliers. A training set of moderate size contains only few examples of outliers, which is often not enough for learning the concept of outliers. To include sufficient outlier examples, one needs to label a large training set, which is often very costly.

Supervised outlier detection problems can be considered as imbalanced classification problems [11][13], since outliers are usually much rarer than normal instances. However, the difficulty of labeling training set is not addressed by existing research on imbalanced classification. Most algorithms proposed for imbalanced classification assume that a reasonable training set is provided. Active learning [26] developed in recent years aims at reducing the effort in labeling the training set. In this paradigm, a classifier is first trained on an initial training set, and then is improved with more labeled instances that are selected according to various principles. In our problem, we start with no training data at all.

Traditionally, instances in training set are all labeled by the expert. In the task of outlier detection, the extreme rareness of outliers often makes this prohibitive. One straightforward idea is to detect suspected outliers and have their labels verified by an expert. All other instances can then be used directly as inliers without hand-labeling. As such, a large training set could be constructed with only a small number of suspected instances hand-labeled. Based on this idea, we propose an algorithm for training set Construction by Identifying Supsected Outliers (CISO) for supervised outlier detection. In CISO, an unsupervised outlier detection technique is first used to rank unlabeled instances in a pool. Highly ranked instances are suspected outliers while lower ranked instances are more likely to be normal. The next step is to find a cutting point dividing suspected outliers and the remaining instances that are considered normal. A good cutting point should generate a suspected set that contains all outliers and as few inliers as possible, so that all outliers can be verified by an expert with the least labeling effort. Based on this intuition, our algorithm searches for a cutting point to guarantee with high probability that all outliers are included in the suspected set to be hand labeled. Since the suspected set is typically much smaller than the whole set of examples in the pool, it is thus expected that we can obtain a large training set by manually labeling only a small portion of the examples. Sometimes a user might want to have control over the total labeling budget. For this need, we also proposed Budgeted CISO

(BCISO), in which a labeling budget could be specified.

For both methods, strict measures are taken to guarantee with high probability that all outliers are included in the suspected set. Although in rare cases a few outliers are labeled as inliers, having a large training set still gives the trained classifier an edge in its detection performance, since the larger training set reduces the variance of the trained classifier.

In our method, data is given without label, and we are attempting to construct a training set with the same distribution as the original dataset. The objective is to minimize the effort of hand labeling. This is different from training set selection[23], in which instance labels are known, and selection is for lower burden or higher accuracy of learners.

The rest of this paper is organized as follows. In next section, related work is reviewed. Section 3 gives a detailed description of our method. Then, empirical evaluation of the proposed methods is presented in Section 4. And the last section summarizes the paper.

2 Related Work

Outlier detection has drawn much attention in recent years. Most algorithms proposed for outlier detection are unsupervised methods. Without labeling information, these methods try to find instances deviating from high-density populations. These instances are identified with different approaches, such as distance-based approaches [17][18], density-based approaches [3][27], clustering-based approaches [12], and information theoretic approaches [2], etc.. Surveys of existing algorithms can be found in [5] and [14]. Particularly, we would like to refer to [25], which uses isolation trees to identify outliers from normal instances. This method has been shown to be very effective and efficient. In our work, this method is used to score the unlabeled instances. Carrasquilla [4] benchmarks several algorithms, both supervised and unsupervised, on large datasets. Its results show that supervised methods generally perform better than unsupervised ones, which supports our idea that outlier detection performance can be improved by labeling some instances.

Supervised outlier detection is overlapped with imbalanced classification [11] [13], because in most cases only a small fraction of the whole dataset are outliers and the inlier class and outlier class are highly imbalanced. To better classify imbalanced classes, different techniques are developed, such as stratified sampling [7][24][6], cost-sensitive learning [9][28] and modifying existing learning methods [1][15]. To our best knowledge, all of the above mentioned work assumes that a labeled training set is given.

Active learning [26] is another field related to our work. The basic idea of active learning is to achieve greater accuracy with fewer labeled instances. Most active learning algorithms also assume that there exists an initial training set, with which a learner is trained at first. However, in our case, there is no initial training data at all. The problem of choosing the initial set for labeling is studied by [16]. In their method, instances are clustered first, and instances nearest to the cluster centers are selected and labeled. This method is for building initial training set of active learning. In our case, no more instances are to be added in training set constructed. Moreover, our data is highly imbalanced and a lot of outliers could not form clusters of their own.

3 Labeling for Supervised Outlier Detection

3.1 Problem Analysis Having sufficient outlier examples in the training set is crucial for the success of supervised outlier detection. However, outliers are rare, and the training set would have to be quite large in order to contain enough outliers. Suppose 0.5% of a dataset are outliers, then one need a training set of 20000 instances to contain 100 outliers. It would be very costly to have all these instances hand labeled. Here we define the "labeling set" to be the set of instances that are labeled by hand. Traditionally, the labeling set equals to the training set.

Since a dataset generally contains much fewer outliers than normal instances, one potential solution to this problem is to focus the labeling efforts on the (suspected) outliers. Specifically, if we can ensure with high probability that all true outliers in the data are included in the suspected set (which naturally will also contain some normal instances), we can rather safely label the remaining instances as inliers without requiring a human to manually label them. This allows us to obtain a large training set by hand labeling only a small fraction of the data, i.e. the labeling set is just the suspected set. Based on this idea, a key question that we try to solve in this work is how to identify the labeling set such that it contains the most outliers and minimal number of inliers.

Instances in a dataset can be first ranked by an unsupervised algorithm, and those with high outlier scores are suspected outliers. However, deciding the ratio of suspected outliers(size of labeling set) in the dataset is not a trivial problem. This ratio is closely related to two factors: the outlier rate and the ranking quality. User usually could have a rough estimation of outlier rate but scarcely the prior knowledge of the ranking quality. So it is not practical to have user set the ratio directly. In this situation, the feedback of labeling instances with high scores becomes important for deciding the ranking quality as well as the size of

labeling set.

It should be noted that there is a possibility that a few outliers would be excluded from labeling set and thus mistakenly labeled as inliers. First, strict measures can be taken to avoid such cases. Even if it happens, the training set obtained has much larger size, which could compensate for the labeling noise because large training set could reduce the variance of classifiers.

3.2 Detect Suspected Outliers For this purpose, we first create a ranking of all instances with an unsupervised outlier detection algorithm. In our case, we need the algorithm to output a ranking of instances based on their likelihood of being outliers. In this work, we use Isolation Forest [25] to create a ranking of all unlabeled instances. Isolation Forest constructs random trees to isolate instances in the dataset. The outlier score of an instance is calculated as its average depth in random trees. This algorithm has been shown to be effective and efficient. However, user could choose any other algorithm for their own applications. With a ranking of instances, the following procedure is the same.

Suppose we have a pool of n unlabeled instances, and they are ranked as $(x_1, x_2, \cdots, x_n)$, in decreasing order of their outlier scores. Let's assume that the pool contains m outliers, where m is unknown. The task is to decide a cutting point n_s such that the first n_s instances will be hand-labeled, and remaining instances are labeled as inliers without manual verifications. The goal is to minimize n_s while ensuring $(x_1, x_2, \cdots, x_{n_s})$ contains all outliers in the pool. The problem is how to decide the value of n_s.

To solve this problem, we make a mild assumption that the user could provide a rough estimate of the outlier rate in the dataset based on their domain knowledge. Usually, we could not expect user to give an exact value of the outlier rate. In our method, an interval for the outlier rate is sufficient. Suppose the interval is given as $[b_l, b_u]$, where b_l and b_u specify the lower and upper bound of the outlier rate.

We also have to make a reasonable assumption on the data and their outlier scores. One observation is, instances with high rankings often have higher probability to be outliers. In the tail of the ranking, the probability of being outlier is almost zero. Let p_i denote the probability that the i-th ranked instance is an outlier. Our algorithm makes the assumption that p_i decreases at least exponentially fast as i increases, i.e.

$$\frac{p_{i+1}}{p_i} \geq \frac{p_{i+2}}{p_{i+1}}, \quad i = 1,\ 2,\ \cdots,\ n-2 \tag{3.1}$$

Figure 1 shows probabilities estimated on the SMTP dataset and the HTTP dataset with pool size of 200

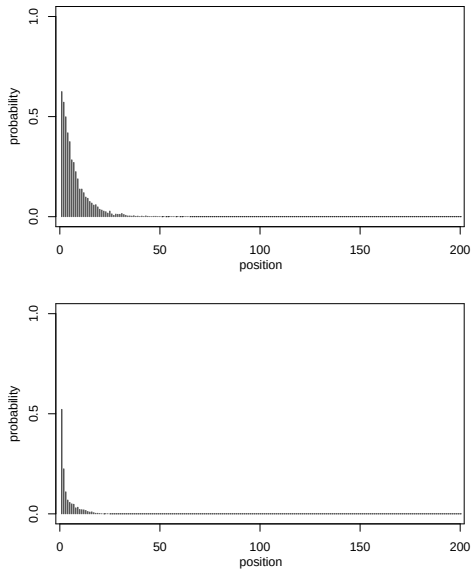

Figure 1: Estimated probability of being an outlier at each position of the ranking. Above, the Ann-thyroid dataset; below, the HTTP dataset. 200 instances are randomly selected and ranked for 1000 times, then ranked labels are averaged.

(see Section 4 for detailed information of datasets). It can be seen from the figure that our assumption roughly holds on these datasets.

Under the above assumption and given the outlier rate interval, our algorithm incrementally determines the proper value of n_s. At first, no instances are labeled and n_s equals to 0. Starting from the top ranked instance, each step one more instance is labeled, and n_s increases by 1, and the following three rules are checked in order to determine when to stop.

1. If the number of labeled instances n_s is less than nb_u, label more instances without checking the remaining two rules.

2. Let m_s be the total number of examples that have been labeled as outliers. If m_s are less than nb_l, label more instances without checking the third rule.

3. Divide the labeled instances into two equal parts. And let m_1 be number of outliers in the first $\frac{n_s}{2}$ instances, and m_2 be number of outliers of the second part, n_u be the number of unlabeled instances and ϵ is a small constant number, then if $\frac{2n_u m_2^2}{m_1 n_s} < \epsilon(m_1 + m_2)$, stop labeling more instances.

We first introduce the third rule, which is the most important stopping criterion. When n_s instances has

been labeled, The average outlier rates of the first half and the second half of n_s instances are estimated as

$$q_1 = \frac{2m_1}{n_s} \tag{3.2}$$
$$q_2 = \frac{2m_2}{n_s}$$

respectively. The ratio of the two rates is

$$r = \frac{q_2}{q_1} = \frac{m_2}{m_1}. \tag{3.3}$$

Since we assume that the outlier probability of the ranked instances decreases at least exponentially fast, the average outlier rate of next $\frac{n_s}{2}$ unlabeled instances would be at most rq_2. And instances further down in the ranking would have a further smaller outlier rate. Therefore, the average outlier rate of all remaining unlabeled instances, q_u, would be smaller than rq_2.

$$q_u \leq rq_2 = \frac{2m_2^2}{m_1 n_s}. \tag{3.4}$$

Let m_u denote the estimated number of outliers in the remaining n_u unlabeled instances. Then,

$$m_u \leq n_u q_u \leq n_u \frac{2m_2^2}{m_1 n_s}. \tag{3.5}$$

If the right side is bounded by

$$n_u \frac{2m_2^2}{m_1 n_s} \leq \epsilon(m_1 + m_2), \tag{3.6}$$

then we have

$$\begin{aligned} m_u &\leq n_u \frac{2m_2^2}{m_1 n_s} \\ &\leq \epsilon(m_1 + m_2), \\ &\leq \epsilon m, \end{aligned} \tag{3.7}$$

where m is the number of outliers in the pool. As indicated in this result, under condition of (3.6), the number of outliers in unlabeled instances would not exceed ϵ of number of outliers in the whole pool, or, $1-\epsilon$ of outliers in the pool are labeled. In our experiment we set ϵ as 0.01. Generally, when labeling stops, the first $\frac{n_s}{2}$ instances contain a large part of all outliers, and the second part includes smaller number of outliers in the long tail of the ranking.

Now let us introduce the first rule, which is used to decide how many instances need to be labeled at first. Based on the analysis above, we clearly need to label more instances than just the outliers in dataset. So one should first label as many instances as outliers in the data without checking other rules. We use nb_u as the estimation of number of outliers in the pool. With these instances labeled, the third rule would have a good estimation of outlier rate.

The second rule states that we need to identify at least nb_l outliers before stop labeling. This helps to rule out unreasonable results when too few outliers are identified. In some cases, the ranking of instances is not very good, and only a small portion of the top ranked instances are true outliers. The estimation of outlier rates and especially their ratio is not reliable. In such cases, this rule would have expert label more instances. User should provide a conservative value for b_l, since if it is higher than the actual outlier rate in the pool, the algorithm would label the whole pool of instances.

Two factors might undermine the performance of CISO. Firstly, the outlier rates q_1 and q_2 may not accurately reflect the true probability of the outliers. This can be solved by setting a large value of b_u. When more instances are labeled, q_1 and q_2 are more accurate estimation of probabilities of being outliers. Secondly, when the assumption of exponentially decreasing probability does not hold, there would be more outliers in the tail of ranking than estimated. However, Figure 1 shows that even though the outlier rate does not necessarily decrease exponentially fast as we move down the ranking, it surely decreases very fast to a very small value. So most outliers would be included in the labeling set even when this assumption does not hold.

One interesting observation in our experiment is, outliers with low outlier scores often lie in centers of small outlier clusters. Although some of them have low rankings and are labeled as inliers, they usually cause less trouble than expected.

3.3 Budget Labeling We now consider a more practical application scenario where we are given a total budget for labeling. In this subsection, we propose Budgeted CISO(BCISO), where the user specifies a labeling budget B. That is, only B instances could be hand labeled by the expert. The problem here is how to decide the pool size with given labeling set size, which is opposite to that of last section.

CISO cannot be directly applied here for the budgeted learning setting, because given a large pool of unlabeled examples, the number of instances labeled with CISO is variable depending on how the outliers is distributed in terms of their ranking. One observation about CISO is that n_s selected by CISO is generally proportional to the pool size. In BCISO, we start from a small pool and run CISO on it. If the first CISO does not consume B (generally it is the case), the pool will be gradually expanded, and the number of instances labeled will also increases correspondingly. When it it

Algorithm 1: $CISO(X, b_l, b_u)$

INPUT: X - unlabeled instance pool,
$[b_l, b_u]$ - estimated outlier rate interval
OUTPUT: (X, y) - labeled training set

$(x_1, x_2, \cdots, x_n) = Rank\ (X, OutlierScore(X))$
for i = 1:n
  query label y_i of x_i
  sat.r1=$(i >= nb_u)$
  $m = sum(y[1:i] == 1)$
  sat.r2=$(m >= nb_l)$
  m_1=$sum(y[1:i/2] == 1)$ $m_2 = m - m_1$
  $n_u = n - i$ $n_s = i$ $\epsilon = 0.01$
  sat.r3= $(n_u \frac{2m_2^2}{m_1 n_s} \leq \epsilon(m_1 + m_2))$
  if(sat.r1 & sat.r2 & sat.r3)
    break
  end if
end for
y[i+1:n]=0

Algorithm 2: $BCISO(X_u, b_l, b_u, B)$

INPUT: X_u - unlabeled instances,
$[b_l, b_u]$ - estimated outlier rate interval,
B - budget
OUTPUT: (X, y) - labeled training set

$X = sample\ (X_u, 5/b_l)$, $y = \{-1\}^{5/b_l}$
while (TRUE)
  $(X, y) = Rank((X, y), OutlierScore(X))$
  $B_c = 0$
  for (i in 1:$nrow(X)$)
    if($y_i == -1$)
      query y_i, B_c++
    end if
    if($B_c >= B$) **goto 14** **end if**
    if(all three rules satisfied) **break** **end if**
  end for
  $X = (X, sample(X_u, k))$, $y = (y, \{-1\}^k)$
end while
if(any rule is NOT satisfied)
  (X, y)=$(x_j, y_j)_{j \notin \{i+1:1.5i\} || y_j != -1}$
end if
y[y==-1]=0

reaches B, we stop adding more instances to the pool.

To achieve this goal, we will first create a small initial pool, and apply CISO to this pool. If the budget is not consumed, we will add k additional random instances to the pool and run CISO again. Each time CISO is applied to the updated pool, labeled instances would not be labeled again. The procedure is repeated until the entire budget is consumed.

The initial pool should be small so that n_s selected on it would be smaller than the budget, and it also should be large enough to contain outliers. In our algorithm, the initial size is set to $\frac{5}{b_l}$, with which the pool contains outliers with probability about $(1 - \exp(-5))$. The setting of k is flexible. Small value of k just makes BCISO have more loops. In our algorithm, k is set to 10.

One issue is, the last run of CISO may not terminate when the budget is consumed. In this case, unlabeled instances in interval $[n_s + 1, 1.5n_s]$ of the ranked pool would be discarded, where n_s is selected by the last CISO. This is because outliers that are not successfully identified mainly reside in this interval, and these outliers should not be used as inliers. It is also possible that no outlier is labeled out with the budget B, which suggests that the user should consider a larger B for labeling.

The computational cost of CISO and BCISO is not a problem compared with the labeling cost. CISO takes $O(n)$ calculations and BCISO takes $O(n^2)$ calculations to reach a result. Since n is usually a small number (less than 10000), both algorithm could finish running in very short time.

The pseudo code of CISO and BCISO is shown in Algorithms 1 and 2.

3.4 Training Classifiers After the training set is constructed, we need to train a classifier next. Outlier detection data usually has the problem of imbalanced classes, and optional measures include random sampling, setting cost to different classes and modifying existing classifiers.

One notable fact here is, there might be some labeling noise in our training set, i.e. a few instances labeled as inliers might actually be outliers while labels of outliers are pure. According to our observation, these false negatives are often near the centers of small clusters of outliers. They are hard to be isolated and receive low outlier scores because they are surrounded by other outliers. Although such instances are labeled as normal, they are more similar to outliers. It means if they are absent in training, the trained classifiers could make right decision at their positions. Based on this fact, we choose random forest with under-sampling[7] for the task of outlier classification. In this method, major class is under-sampled to make data more balanced. The under-sampling rate is often very low due to the high ratio between normal instances and outliers. So falsely labeled instances have little chance to be sampled, and they would be absent from the training sets of most base classifiers. These classifiers often could make right predictions at positions of false

negatives in training data. When all base classifiers vote, the effect of false labels can be largely canceled out.

According to our experiments, supervised algorithms significantly outperform unsupervised algorithms given enough training instances.

4 Empirical Results

In this section, we empirically show that CISO and BCISO could build large training sets with low labeling noise, and the training set could greatly improve classifier performance. We conduct two experiments for each dataset. The first experiment shows that most outliers are indeed labeled out. The second experiment compares performances of different outlier detecting methods and the results show that our methods outperforms other methods in general.

4.1 Experimental Setup Six datasets are used to evaluated all methods in comparison. Information of these datasets is shown in Table 1. The HTTP dataset and the SMTP dataset are two subsets of the KDD CUP 99 network intrusion data. HTTP and SMTP connections are taken out to form these two datasets. Covtype, Shuttle, Ann-thyroid are from UCI repository[1]. Mammography is a dataset of medical tests[2]. Since Isolation Forest can only handle numeric features, binomial features are removed and multinomial features are converted into numeric features for all datasets.

For each dataset, we run 100 random experiments and all results are averaged. In each random run, each dataset is randomly split into three folds. One third is used as training data, since only a small number of instances is further chosen from training data. The instances in training data are labeled only with queries. The remaining two thirds are used as testing data, which could give a better evaluation of different algorithms.

In each run, CISO and BCISO randomly draw unlabeled instances from the training data. Both methods need an outlier rate interval on the given dataset. We assume that the user could provide a rough percentage of outliers, which is usually estimated with domain knowledge. In our experiment, we try to make it as reasonable as possible. For each dataset, we first calculate the true outlier rate on the training data. Then we set the upper bound of the interval as below. If the actual outlier rate is smaller than 1%, we double it and round it to percentage, otherwise, we add 1% to the actual outlier rate and round it to percentage. The lower bound the interval is set as $\frac{1}{4}$ of the upper bound. For example, the interval for HTTP dataset is set as $[0.25\%, 1\%]$, and it is set as $[0.5\%, 2\%]$ for SMTP dataset. As our analysis before, our method is not sensitive to interval setting.

The baseline in comparison is RAND method. In each run, a set of instances is randomly selected from training data. All these instances are manually labeled and form training set. This is what we traditionally do.

In CISO, the number of instances for labeling could not be controlled exactly. So we run CISO first, and $\bar{n}_s$ is the average number of instances labeled in all runs. Then the BCISO and RAND methods all use $\bar{n}_s$ as their labeling budget. In this way, we can include CISO in comparison with different methods. We first compare the training set constructed by these three methods. Then, classification performances on training sets are compared, and the performance of the unsupervised method is also included. The R implementation of Random Forest [3] is used as classifier. We set all parameters to their default values except the sampling size. Sample sizes for outlier class and inlier class are both set to be the number of outliers in the training set.

We evaluated the detection performances of all algorithms with Area Under Precision-Recall Curve(AUPRC). In many previous studies, outlier detection performance has been evaluated by AUC(Area Under ROC Curve) [22] [20][25]. However, when data is highly skewed, AU-PRC is more informative than ROC in evaluating performances[8]. With Isolation Forest, AUC value on Covtype dataset is 0.88, which seems to be a good value. However, the outliers detected by Isolation Forest have many normal instances mixed in, and significant improvement can be expected. The AU-PRC value for such a performance is 0.055, and it could be improved to 0.91 by supervised method. In this example, AU-PRC value gives people better sense of performances. Calculation of AU-PRC involves the false negative value, which is often the key focus of user's concern. We often care more about purity of outliers detected. Take medical test for example, if some instances are predicted as outliers, more medical tests need to be done on them to find out which are true outliers. More false negatives mean more further tests. In ROC evaluation, the true negative value instead of the false negative value plays an important role. However, true negative value is often less important since we often put much more attention on suspected cases rather than normal cases. In this sense, AU-PRC is a more reasonable evaluation measure than AUC.

[1] http://archive.ics.uci.edu/ml/

[2] Thanks to Prof. Nitesh V. Chawla for providing this dataset

[3] http://cran.r-project.org/web/packages/randomForest/index.html

Table 1: Information of six datasets

datasets	instances	dimension	outlier class	outliers
HTTP	623091	25	attack	4045 (0.65%)
SMTP	96554	27	attack	1183 (1.2%)
Covtype	286048	11	class 4 vs. class 2	2747 (0.96%)
Mammography	11180	7	class 1	260 (2.3%)
Shuttle	58000	10	class 2, 3, 5, 6, 7	3511 (6.1%)
Ann-thyroid	7200	7	class 1	166 (2.3%)

Table 2: Training set for HTTP dataset

B		264	485	801	973
CISO	n	4000	6000	8000	10000
	n_s	264.2 ±180.3	484.9 ±267.7	801.4 ±326.7	973.3 ±352.9
	m	25.7 ±4.6	38.8 ±5.6	51.9 ±6.3	64.5 ±7.2
	$\frac{m_s}{m}$	87.2 ±17.7	90.8 ±17.3	96.6 ±11.6	98.3 ±7.7
BCISO	n	3959.1 ±1981	5242.4 ±2228	7802.6 ±3226	9386 ±3598
	m	25.9 ±12.6	34.8 ±15	51.4 ±22.3	63.1 ±24.9
	$\frac{m_s}{m}$	96.3 ±8.5	98.2 ±5.8	99.3 ±2.9	99.9 ±0.3
RAND	m_r	1.5 ±1.3	3 ±1.8	5 ±2.1	6.7 ±2.5
	v	25	5	1	0

Table 3: Training set for SMTP dataset

B		56	89	128	155
CISO	n	2000	3000	4000	5000
	n_s	56.4 ±14.8	89.4 ±24.4	127.8 ±37.7	155 ±33.6
	m	24.8 ±4.7	37 ±5.5	49.6 ±6.6	61.9 ±7.1
	$\frac{m_s}{m}$	99.3 ±1.7	99.4 ±1.3	99.4 ±1.1	99.3 ±1.1
BCISO	n	1900.4 ±376.1	2896.1 ±680.9	4027.2 ±848.7	4547.7 ±1168
	m	23.7 ±3.7	35.6 ±8.3	50.1 ±9.3	56.8 ±14
	$\frac{m_s}{m}$	99.5 ±1.4	98.8 ±2.8	99.3 ±1	99.5 ±1
RAND	m_r	0.7 ±0.9	1 ±0.9	1.8 ±1.5	2 ±1.3
	v	51	35	21	13

Table 4: Training set for Covtype dataset

B		580	808	1154	1468
CISO	n	1000	1500	2000	2500
	n_s	580.2 ±237.2	808 ±227	1153.7 ±308.4	1467.9 ±344.3
	m	9.4 ±3.8	14.3 ±4.3	19.2 ±5	24 ±5.4
	$\frac{m_s}{m}$	98.4 ±6.2	98.7 ±6.9	99.2 ±2.7	99.6 ±1.4
BCISO	n	1058.5 ±248.1	1289.8 ±318.1	1791 ±428.8	2378.5 ±509.1
	m	10.2 ±4.2	13.4 ±5.4	17.3 ±6.9	23.9 ±7.8
	$\frac{m_s}{m}$	99.3 ±3.4	99.8 ±1	99.5 ±1.7	99.5 ±1.7
RAND	m_r	5.5 ±2.1	7.8 ±2.8	10.7 ±3.4	14.2 ±3.5
	v	0	0	0	0

Table 5: Training set for Mammography dataset

B		420	591	712	844
CISO	n	800	1000	1200	1400
	n_s	419.6 ±179.3	590.8 ±194.9	712.1 ±213.5	844.2 ±214.7
	m	18.3 ±3.9	22.7 ±4.4	27.4 ±4.7	32.1 ±5.2
	$\frac{m_s}{m}$	88.9 ±16	93.3 ±11.1	93.9 ±9.1	94.4 ±7.8
BCISO	n	709.8 ±178.8	874.4 ±180.4	1022 ±159.9	1199.8 ±219.6
	m	17.9 ±5.4	21 ±6	25.1 ±5.9	28.8 ±7.3
	$\frac{m_s}{m}$	95.7 ±6.1	96.7 ±3.7	96.7 ±3.9	97.7 ±3.1
RAND	m_r	9.9 ±3.1	13.3 ±3.4	16.8 ±4.3	19.1 ±4.1
	v	0	0	0	0

Table 6: Training set for Shuttle dataset

B		47	62	79	92
CISO	n	300	400	500	600
	n_s	47.1 ±14.5	62.2 ±14.5	78.7 ±14.2	91.5 ±15.4
	m	18.4 ±3.8	24.3 ±4.6	30.8 ±5.2	36.9 ±5.4
	$\frac{m_s}{m}$	99 ±2.2	98.9 ±1.9	99.3 ±1.4	99 ±1.5
BCISO	n	325.3 ±62.7	381.8 ±80.5	476 ±84.9	525.3 ±127.2
	m	19.9 ±2.8	24.4 ±3.6	30.2 ±4.6	33.9 ±6
	$\frac{m_s}{m}$	98.2 ±3	98.9 ±2.1	98.8 ±2.1	98.8 ±1.8
RAND	m_r	3.1 ±1.6	3.6 ±1.8	4.8 ±2.2	5.6 ±2.1
	v	3	2	1	0

Table 7: Training set for Ann-thyroid dataset

B		154	205	243	278
CISO	n	800	1000	1200	1400
	n_s	153.8 ±60.3	204.8 ±60.1	243.3 ±61.7	277.6 ±72.3
	m	19.3 ±3.8	23.9 ±4.1	28.9 ±4.7	33.2 ±4.6
	$\frac{m_s}{m}$	96.8 ±8.7	99.1 ±3.6	99.3 ±1.9	99.2 ±1.8
BCISO	n	806.4 ±207.5	923.6 ±279.2	1019.9 ±253.7	1215 ±316.1
	m	19.6 ±5	23 ±5.9	25.2 ±7.1	29.6 ±7.9
	$\frac{m_s}{m}$	98.9 ±2.6	99.1 ±2	99.6 ±1.1	99.5 ±1.2
RAND	m_r	3.6 ±1.7	4.3 ±2	5.5 ±2.2	6.1 ±2.3
	v	4	2	0	0

4.2 Constructing Training Sets In this subsection, training sets constructed by different methods are shown. In this experiment, we investigate labeling efficiency and outlier recall in the labeling set for our methods. The size of the training set constructed and the number of outliers included are compared between the three methods. All results are averaged over 100 runs.

Table 2–6 show information about the training sets constructed by the three methods for different datasets. In each table, the first row, B, is the number of instances labeled in BCISO and RAND. It is also the average number of instances manually labeled in CISO with different pool sizes. The size of the training set constructed by RAND is also B. For CISO method, the pool size n, the number of instances labeled n_s and the actual number of outliers in training set m are shown. We also include in table the percentage of outliers which are labeled out $\frac{m_s}{m}$, which indicates how much labeling noise is in training set. For BCISO method, we have shown the same values as CISO except n_s. For RAND method, the first row shows m_r, the number of positive instances included in training set. Sometimes, there are no outliers in RAND training sets and these training sets are invalid for training. Percentage of invalid training sets in 100 runs, denoted as v, is reported in the second row of RAND results. Mean values and standard deviations are given for values averaged.

From the results we could see that CISO and BCISO could build a much larger training set than labeling set. On the two datasets, HTTP, SMTP, the training sets built are more than ten times larger than labeling sets. On Shuttle dataset and Ann-thyroid dataset, the training sets are about five times of the labeling sets. On the other two datasets, they perform not as good, but they still save much effort of labeling those instances that are quite likely to be normal. The ratio between sizes of the labeling set and the training set depends on the outlier rate of dataset and the ranking of instances. If the dataset has a very low outlier rate, which is often the case, a small labeling set could contain all outliers. The ranking of instances is also very important. If most outliers have high rankings, CISO and BCISO would stop early and have less instances labeled. HTTP dataset and SMTP dataset have low outlier rates, and Isolation Forest has good performance on them[4], thus the labeling set is much smaller than training set. Although Covtype dataset also has low outlier rate, the instance ranking by Isolation Forest has a quite low AU-PRC score, and outliers are distributed in a long tail in the ranking. More instances need to be labeled to include most outliers. From the results of the rest three datasets, we can also see similar effect of the outlier rate and the quality of initial ranking.

For all datasets except HTTP, we can see most outliers are included in the CISO training set. In the ranking of instances from HTTP dataset, outliers have long tail in ranking and some of them are not included the labeling set. In BCISO training sets, only a few outliers are used as normal instances, which supports our idea of discarding some instances in the interval after the labeling set. From these tables, we could see in five datasets approximately 99% outliers are labeled out. For different pools sampled, rankings of outliers

[4]The performance of Isolation Forest is shown in Section 4.3

have large variances, which makes CISO have large variance on sizes of labeling sets. And similarly, BCISO has large variance on sizes of training sets constructed. Nevertheless, such variances do not have much influence on classification performance.

For the RAND method, the training set constructed has the same size as the labeling set, therefore, the training set constructed by RAND is much smaller than those of CISO and BCISO. When it labels the same number of instances as CISO and BCISO, no outliers are selected in some runs and the training set is invalid for later training. This happens on the HTTP dataset and the SMTP dataset, both of which have low outlier rates. Even though in other runs small number of outliers are labeled out, they are still not sufficient for training.

4.3 Detection Performance In this section, training sets constructed with different methods are evaluated with classification performance. The results shown in Table 8–13 are AU-PRC values with different number of labeled instances. Note that, AU-PRC values of the RAND method are averaged on only valid training sets. The row denoted as UNSUP shows performances of Isolation Forest.

We can see CISO and BCISO outperforms other methods in general. Compared with RAND, AU-PRC values of CISO and BCISO are significantly higher. Although some training sets built by the two methods have labeling noise, they have much larger sizes and classifier trained on them achieve better performance in classification. Compared with Isolation Forest, CISO and BCISO make improvement on all six datasets. On HTTP, Covtype, Mammography and Ann-thyroid datasets, the improvements are significant. On SMTP and Shuttle datasets, improvements are minor, since performance of unsupervised method is already very high. However, such improvements are still of significance if there are large quantity of testing instances.

The difference between CISO and BCISO is insignificant, since they generally build similar training sets. When the pool size is small, there are more noises in CISO training set, and its performance is a little lower than BCISO. When the pool size is large, instances discarded by BCISO have some effect, and the performance on the CISO training set becomes higher than that of BCISO. This can be seen from results on HTTP and Ann-thyroid datasets.

The RAND method is better than unsupervised learning method on HTTP, Covtype, Mammography and Ann-thyroid datasets, when performance of Isolation Forest is not good. On the other two datasets, its performances finally catch up with increasing size of training sets. Though the RAND method could overtake unsupervised method given enough training data, it would need much more labeling efforts.

5 Summary and Future Work

In most outlier detection applications, outliers are very rare in the data. Due to the labeling cost, training sets constructed in traditional ways tend to contain insufficient outliers for training. In this work, we present the CISO algorithm, which constructs training sets for outlier detection problems with greatly reduced labeling efforts. With a unsupervised outlier detection algorithm, instances are first ranked according to their outlier scores. Based on the ranking, only suspected outliers are labeled. The instances with high rankings are hand-labeled and those with low rankings receives the label of inlier. As a key novelty of the algorithm, we design strict measures to guarantee that most outliers are selected to be hand-labeled. With our method, one can construct large training sets by only hand labeling a small portion of instances. Experiments show that the training set constructed by CISO has large size and low labeling noise. Experiments also show that the CISO training sets typically lead to classifiers that significantly outperform classifiers built with traditionally labeled training set.

We also proposed Budgeted CISO (BCISO), with which user could set a labeling budget. In this method, new instances are incrementally added to the instance pool of CISO. New added instances are labeled if they fall in the suspected interval of ranking. After the labeling budget is reached, unlabeled suspected instances are discarded. BCISO achieves similar performance as CISO in the experiments.

In the future, we would like to improve the ranking of the instances by combining different outlier detection algorithms. Another direction might be incorporating outlier labeling into outlier detection algorithm and making it actively pose queries.

Acknowledgment

This work is supported by NSF Award 1055113. Thanks to members of CAMEL group for their suggestions.

Table 8: Performance Comparison on HTTP dataset

n_s	264	485	801	973
UNSUP	0.62±0.05	0.62±0.047	0.62±0.048	0.62±0.051
CISO	0.953±0.127	0.952±0.133	0.99±0.055	**0.994±0.032**
BCISO	**0.991±0.042**	**0.998±0.001**	**0.998±0.004**	0.99±0.057
RAND	0.617±0.202	0.76±0.24	0.876±0.183	0.93±0.14

Table 9: Performance Comparison on SMTP dataset

n_s	56	89	128	155
UNSUP	0.985±0.004	0.985±0.005	0.985±0.004	0.985±0.004
CISO	**0.991±0.008**	**0.992±0.002**	**0.993±0.003**	**0.993±0.002**
BCISO	**0.991±0.004**	**0.992±0.004**	**0.993±0.002**	**0.993±0.002**
RAND	0.96±0.025	0.936±0.113	0.966±0.021	0.964±0.031

Table 10: Performance Comparison on Covtype dataset

n_s	580	808	1154	1468
UNSUP	0.055±0.015	0.055±0.014	0.055±0.014	0.055±0.017
CISO	0.809±0.121	**0.869±0.082**	**0.897±0.043**	**0.909±0.03**
BCISO	**0.822±0.134**	0.862±0.075	0.881±0.069	0.907±0.046
RAND	0.726±0.164	0.797±0.101	0.842±0.075	0.879±0.051

Table 11: Performance Comparison on Mammography dataset

n_s	420	591	712	844
UNSUP	0.205±0.038	0.205±0.036	0.205±0.038	0.205±0.043
CISO	0.573±0.042	**0.582±0.04**	**0.589±0.034**	**0.596±0.036**
BCISO	**0.575±0.041**	0.572±0.038	0.585±0.034	**0.596±0.034**
RAND	0.541±0.074	0.553±0.044	0.572±0.046	0.574±0.035

Table 12: Performance Comparison on Shuttle dataset

n_s	47	62	79	92
UNSUP	0.94±0.015	0.94±0.014	0.94±0.015	0.94±0.018
CISO	**0.972±0.005**	**0.975±0.007**	**0.977±0.007**	**0.979±0.008**
CISOB	**0.972±0.006**	**0.975±0.007**	0.975±0.008	0.978±0.008
RAND	0.914±0.071	0.921±0.067	0.937±0.083	0.944±0.049

Table 13: Performance Comparison on Ann-thyroid dataset

n_s	154	205	243	278
UNSUP	0.519±0.073	0.519±0.075	0.519±0.07	0.519±0.076
CISO	0.942±0.024	0.95±0.018	**0.953±0.017**	**0.957±0.015**
CISOB	**0.947±0.018**	**0.951±0.018**	0.952±0.017	0.956±0.017
RAND	0.9±0.048	0.913±0.035	0.922±0.031	0.922±0.03

References

[1] R. Barandela, J. S. Sánchezb, V. García, E. Rangel. *Strategies for learning in class imbalance problems*. Pattern Recognition, 36(2003), pp. 849-851.

[2] C. Böhm, K. Haegler, N. S. Müller and C. Plant. *CoCo: coding cost for parameter-free outlier detection*. Proceedings of ACM Knowledge Discovery and Data Mining, (2009), pp. 85–126.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. *LOF: identifying density-based local outliers*. ACM SIGMOD Record, 29(2)(2000), pp. 93–104.

[4] U. Carrasquilla. *Benchmarking algorithms for detecting anomalies in large datasets*. MeasureIT, Nov.(2010), pp. 1–16.

[5] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly detection : a survey*. ACM Computing Surveys, 41(3)(2009), Article 15.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. *SMOTE: synthetic minority oversampling technique*. Journal of Artificial Intelligence Research, 16(2002), pp. 321-357.

[7] C. Chen, A. Liaw, and L. Breiman. *Using random forest to learn imbalanced data*. Tech. Report 666, Department of Statistics, UC Berkeley, (2004).

[8] J. Davis and M. Goadrich. *The relationship between precision-recall and ROC curves*. Proceedings of International Conference on Machine Learning, (2006), pp. 233–240.

[9] C. Elkan. *The foundations of cost-sensitive learning*. Proceedings of International Joint Conferences on Artificial Intelligence, (2001), pp. 973–978.

[10] J. Gao, P.-N. Tan. *Converting output scores from outlier detection algorithms into probability estimates*. Proceedings of IEEE International Conference on Data Mining, (2006) pp. 212–221.

[11] V. García, J. S. Sánchez, R. A. Mollineda, R. Alejo, and J. M. Sotoca. *The class imbalance problem in pattern classification and learning*. Data Engineering, (2007), pp. 283–291.

[12] Z. He, X. Xu, and S. Deng. *Discovering cluster-based local outliers*. Pattern Recognition Letters, 24(9-10)(2003), pp. 1641-1650.

[13] H. He and E. A. Garcia. *Learning from imbalanced data*. IEEE Transactions on Knowledge and Data Engineering, 21(9)(2009), pp. 574–584.

[14] V. Hodge and J. Austin. *A survey of outlier detection methodologies*. Artificial Intelligence Review, 22(2)(2004), pp. 85–126.

[15] X. Hong, S. Chen, and C.J. Harris. *A kernel-based two-class classifier for imbalanced data sets*. IEEE Transactions on Neural Networks, 18(1)(2007), pp. 28–41.

[16] J. Kang, K. R. Ryu and H.-C. Kwon. *Using cluster-based sampling to select initial training set for active learning in text classification*. Advances in Knowledge Discovery and Data Mining, 3056(2004), pp. 384–388.

[17] E. M. Knorr and R. T. Ng. *Algorithms for mining distance-based outliers in large datasets*. Proceedings of International Conference on Very Large Data Bases, (1998), pp. 392-403.

[18] E. M. Knorr and R. T. Ng. *Finding intensional knowledge of distance-based outliers*. Proceedings of International Conference on Very Large Data Bases, (1999), pp. 211–222.

[19] H.-P. Kriegel, M. Schubert and A. Zimek. *Angle-based outlier detection in high-dimensional data*. Proceedings of ACM Knowledge Discovery and Data Mining, (2008), pp. 444–452.

[20] H.-P. Kriegel, P. Kröger, E. Schubert and A. Zimek. *LoOP: local outlier probabilities*. In Proceedings of ACM Conference on Information and Knowledge Management, (2009), pp. 1649–1652.

[21] H.-P. Kriegel, P. Kröger, E. Schubert and A. Zimek. *Interpreting and unifying outlier scores*. In Proceedings of SIAM International Conference on Data Mining, (2011), pp. 1324.

[22] A. Lazarevic, V. Kumar. *Feature bagging for outlier detection*. Proceedings of ACM Knowledge Discovery and Data Mining, (2005), pp. 157–166.

[23] H. Liu, H. Motoda. *On issues of instance selection*. Data Mining and Knowledge Discovery, 6(2002), pp. 115-130.

[24] X.-Y. Liu and J. Wu and Z.-H. Zhou. *Exploratory under-sampling for class-imbalance learning*. Proceedings of IEEE International Conference on Data Mining, (2006), pp. 965–969.

[25] F. T. Liu, K. M. Ting and Z.-H. Zhou. *Isolation forest*. Proceedings of IEEE International Conference on Data Mining, (2008), pp. 413–422.

[26] B. Settles. *Active learning literature survey*. Tech. Report 1648, University of WisconsinMadison, (2009).

[27] J. Tang, Z. Chen, and A. Fu, and D. Cheung. *Enhancing effectiveness of outlier detections for low density patterns*. Proceedings of Pacific-Asia Knowledge Discovery and Data Mining, (2002), pp. 535–548.

[28] K. M. Ting. An instance-weighting method to induce cost-sensitive trees. IEEE Transactions on Knowledge and Data Engineering, 14(3)(2002), pp. 659–665.

[29] X. Yang, L. J. Latecki and D. Pokrajac. *Outlier detection with globally optimal exemplar-based GMM*. Proceedings of SIAM International Conference on Data Mining, (2009), pp. 145–154.