Outlier Detection Using Random Walks

H. D. K. Moonesinghe, Pang-Ning Tan
Department of Computer Science & Engineering
Michigan State University
East Lansing, MI 48824
(moonesin, ptan)@cse.msu.edu

Abstract

The discovery of objects with exceptional behavior is an important challenge from a knowledge discovery standpoint and has attracted much attention recently. In this paper, we present a stochastic graph-based algorithm, called OutRank, for detecting outlying objects. In our method, a matrix is constructed using the similarity between objects and used as the adjacency matrix of the graph representation. The heart of this approach is the Markov model that is built upon this graph, which assigns an outlier score to each object. Using this framework, we show that our algorithm is more powerful than the existing outlier detection schemes and can effectively address the inherent problems of such schemes. Empirical studies conducted on both real and synthetic data sets show that significant improvements in detection rate and a lower false alarm rate are achieved using our proposed framework.

1. Introduction

Random walk methods have been widely used for a variety of information retrieval tasks, including web search [15], keyword extraction [13], and text summarization [4][14]. These methods represent the data as a stochastic graph and perform random walk along all the paths on the graph to assess the *centrality* or *importance* of individual objects in the data. For example, in text summarization [4], the random walk method can be used to identify a sentence that is most representative of other sentences in a collection of documents.

This paper explores the use of random walk models for outlier detection as an alternative to previously used outlier detection algorithms [1][2][3][6][8][9][10][12] [16] [17]. The heart of this approach is to represent the underlying dataset as a weighted undirected graph, where each node represents an object and each (weighted) edge represents similarity between objects. By transforming the edge weights into transition probabilities, we model the system as a Markov chain and find the dominant eigenvector of the transition probability matrix. The values in the eigenvector are then used to determine the outlierness of each object.

As will be shown in this paper, a key advantage of using our random walk approach is that it can effectively capture not only the outlying objects scattered uniformly but also small clusters of outliers. In real-life applications such as intrusion detection [11], the small clusters of outliers often correspond to interesting events such as denial-of-service or worm attacks. Although existing density-based algorithms show high detection rate over distance-based algorithms for datasets with varying densities, they can be less effective when identifying small clusters of outliers. This is because these algorithms consider the density of a predefined neighborhood for outlier detection, and in some cases small clusters of outliers with similar density to normal patterns cannot be distinguished. A random walk based model solves this problem by defining the outlierness of an object with respect to the entire graph of objects; i.e. it views the outlierness from a global perspective.

Nevertheless, a key challenge of using the random walk approach is defining the appropriate similarity metric for constructing the neighborhood graph. We examine two approaches for doing this. The first approach (known as **OutRank-*a***) uses cosine similarity between objects whereas the second approach (known as **OutRank-*b***) uses shared-nearest neighbor density to define similarity. We perform extensive experiments to compare the performance of both approaches against distance-based and density-based outlier detection algorithms. Experimental results using both real and synthetic data sets show that our proposed algorithms outperform other approaches and yield higher detection rates with lower false alarm rates. We also found OutRank-*b* to be more robust than OutRank-*a*.

The main contributions of this paper are summarized below:

I. We investigate the effectiveness of random walk based approach for outlier detection.
II. We propose two outlier detection algorithms OutRank-*a* and OutRank-*b*, which are capable of

detecting outliers even when the normal patterns have similar densities as outliers.

III. Our method is based on an automatic, data dedicated threshold for defining the neighborhood of an object. In contrast, existing algorithms require users to specify a neighborhood parameter, which is not a trivial task.

The remainder of the paper is organized as follows. In section 2, we introduce our proposed outlier detection model. Section 3 presents several outlier detection algorithms. In section 4, we perform an extensive performance evaluation on real and synthetic data sets. Finally, we conclude our work in section 5.

2. Modeling Outliers Using a Graph

In this section we develop our framework to discover outlying objects in a database. According to Hawkins [6], outliers can be defined as follows:

Definition 1: (Outlier) An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.

Most outlier detection schemes adopt Hawkin's definition of outliers and thus assume that outliers are isolated points far away from other normal points. As such, these outliers can be easily detected by existing distance or density based algorithms. However, in this paper we focus on outliers that might be concentrated in certain regions, thus forming small clusters of outliers.

We take a graph based approach to solve this problem. Here we model objects in the database as a graph, where each node represents an object and each edge represents a similarity between them. Each edge is also assigned a weight, which is equal to the similarity between the nodes of the corresponding edge. There are two major issues that need to be addressed: first, how to determine the link structure of the graph based on the similarity of nodes; second, how to discover the outlying objects using this graph model. Following sections describe these issues in detail.

2.1. Similarity of objects

In order to determine the link structure of the graph we compute the similarity among objects as follows:

Definition 2: (object_similarity) Let $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$ be any two objects drawn from a d-dimensional space R^d. The similarity between X and Y is defined by the *cosine* between two corresponding vectors:

$$\text{object\_similarity}(X, Y) = \begin{cases} 0 & \text{if} \quad X = Y \\ \dfrac{\sum_{k=1}^{d} x_k y_k}{\sqrt{\sum_{k=1}^{d} x_k^2} \cdot \sqrt{\sum_{k=1}^{d} y_k^2}} & \text{otherwise} \end{cases} \quad (1)$$

Note that the similarity between an object to itself is set to zero to avoid self loops in the underlying graph representation. Such loops are ignored since they are common to every node, and therefore it is not very useful to distinguish normal objects from outliers.

The relationship between all objects in the database is represented by the similarity matrix $Sim_{n \times n}$, where n is the number of objects. Each entry in the matrix ($Sim[i,j]$) corresponds to the similarity between object i and object j as defined above. We take this similarity matrix as the adjacency matrix of the graph. In the graph representation, two nodes X and Y are connected by an edge if their similarity measure is greater than zero, and the weight of that edge is taken as the *object_similarity(X, Y)*.

2.2. Markov chain model

Using our similarity matrix and the corresponding graph representation, we model the problem of outlier detection as a Markov chain. The Markov chain modeled here is a random walk on a graph defined by the link structure of the nodes. We hypothesize that under this representation, if an object has a low connectivity to other objects in the graph, then it is more likely to be an outlier.

Connectivity is determined in terms of weighted votes given by the other nodes in the graph. Here high connectivity nodes convey votes with more weight than that conveyed by the lesser connectivity nodes. The weight of the vote from any node is scaled by the number of nodes adjacent to the source node. We formulate this scenario by considering nodes with connectivity values and distributing them among neighboring nodes. This can be expressed by defining the connectivity of a node.

Definition 3: (connectivity) Connectivity $c(u)$ of node u at t is expressed as

$$c_t(u) = \begin{cases} a & \text{if} \quad t = 0 \\ \sum_{v \in adj(u)} (c_{t-1}(v) / |v|) & \text{otherwise} \end{cases} \quad (2)$$

where a is any arbitrary initial value, t is the iteration step, $adj(u)$ is the set of nodes linked to node u, and $|v|$ denotes the degree of node v.

For n nodes, p_1, p_2, …,p_n, we can arbitrarily assign each node an initial connectivity value (e.g. $c_0(p_i) = 1/n$, $1 \le i \le n$) and recursively carryout the computation to refine the connectivity value for every node at each iteration step. This iterative procedure is called the power method and is often used to find the dominant eigenvector of a matrix. The refinement of the connectivity value of each node is done by modeling this scenario as a Markov chain. Here equation (2) can be expressed in matrix notation as follows:

$$c = S^T c \tag{3}$$

where S is called the transition matrix and c is the stationary distribution representing connectivity value for each object in the dataset. For a general transition matrix, neither the existence nor the uniqueness of a stationary distribution is guaranteed, unless the transition matrix is *irreducible* and *aperiodic* (*Perron-Frobenius* theorem [7]).

The transition matrix (S) of our Markov model is obtained by normalizing the similarity matrix (Sim) defined earlier:

$$S[i, j] = \frac{Sim[i, j]}{\sum_{k=1}^{n} Sim[i, k]} \tag{4}$$

This normalization ensures that the elements of each row of the transition matrix sum to 1, which is an essential property of the Markov chain. Also, the transition probabilities in S do not change over time.

After computing the transition matrix S, we need to make S both *irreducible* and *aperiodic* in order to converge to a unique stationary distribution. In the past, this problem has been solved by Page et al. [15] in their *PageRank* computation by reserving a low probability value for restarting the random walk. We take their approach and give the revised equation (3) as:

$$c = d + (1 - d)S^T c \tag{5}$$

where d is known as the *damping factor*. This scenario can be viewed as a random walk on a Markov chain where the walker visits one of the adjacent states with probability (1-d) or requests another random state with probability d.

Consider the sample 2-dimensional database with 11 objects shown in Figure 1. Clearly object 1 and object 2 are outliers and the rest corresponds to a normal pattern. Note that x and y represent the (x,y) co-ordinates in the 2-D space.

Object	x	y
1	4.0	2.0
2	4.5	1.5
3	2.0	4.0
4	2.0	4.5
5	2.0	5.0
6	2.5	4.0
7	2.5	4.5
8	2.5	5.0
9	3.0	4.0
10	3.0	4.5
11	3.0	5.0

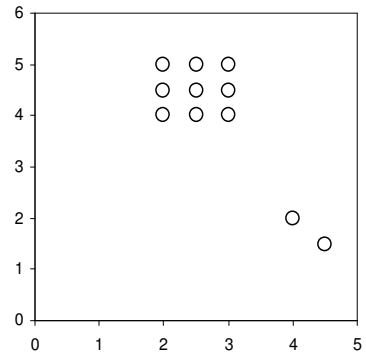

Figure 1. Sample 2-D data set

Using an arbitrary initial connectivity vector and applying equation (2), our model for the sample 2-D dataset converges to a stationary distribution after 112 iterations (using the damping factor-d = 0.1). Final connectivity values and the rank for each object are shown in Table 1. Note that object -1 and object -2 are ranked as the most outlying objects.

Table 1. Outlier rank for sample 2-D dataset

Object	Connectivity	Rank
1	0.0835	2
2	0.0764	1
3	0.0930	5
4	0.0922	4
5	0.0914	3
6	0.0940	9
7	0.0936	7
8	0.0930	6
9	0.0942	10
10	0.0942	11
11	0.0939	8

3. Algorithms

This section describes our proposed algorithm based on the above framework for detecting outliers in a given dataset. Two variants of the algorithm *OutRank* are presented.

3.1. OutRank-*a*: using object similarity

In this algorithm, we use cosine similarity to form the transition probability matrix for our Markov model. We use power method to compute the stationary distribution of the Markov chain. Here we initialize the initial connectivity vector (at t=0) to an arbitrary value ($1/n$, where n is the total number of objects). Also, we set

the damping factor to 0.1. The complete algorithm for OutRank-*a* is shown below.

Algorithm OutRank-*a*

Input: Similarity matrix $Sim_{n \times n}$ with n objects, error tolerance ε.
Output: Outlier ranks c.
Method:

```
1:  for i=1 to n  do          // forms transition matrix S
2:    let totSim=0.0;
3:    for j=1 to n do
4:       totSim=totSim+Sim[i][j];
5:    end
6:    for j=1 to n do
7:       S[i][j]=Sim[i][j]/totSim;
8:    end
9:  end
10: let d=0.1                 // damping factor
11: let t=0;
12: let c_0 =(1/n).1          // arbitrary assignment
13: repeat
14:     c_{t+1}= d/n + (1-d)S^T c_t
15:     δ = ||c_{t+1} - c_t||_1
16:     t = t+1;
17: until (δ<ε)
18: rank c_{t+1} from min(c_{t+1}) to max(c_{t+1})
19: return c_{t+1};
```

3.2. OutRank-*b*: using shared neighbors

In OutRank-*a*, nodes are considered adjacent if their corresponding cosine similarity measure is non-zero. So even the nodes with low similarity values are considered adjacent, and that similarity value is used as the weight of the link. In this section we propose an alternative algorithm called OutRank-*b*, which uses a similarity measure that considers the number of neighbors shared by the objects. For example, consider two objects, p_1 and p_2. Suppose p_1 has a set of neighbors $\{p_3, p_4, p_5, p_7\}$ and p_2 has a set of neighbors $\{p_3, p_4, p_6, p_7\}$ (see Figure 2). The set of neighbors shared by both p_1 and p_2 is $\{p_3, p_4, p_7\}$. In this algorithm, we take the cardinality of this set as the similarity measure.

In order to define the shared neighbors we need to find the neighbors of a given object (i.e. adjacent nodes of the graph representation). Here we limit the neighbors only to a set of nodes having high similarity values by using a threshold to cutoff low similarity neighbors. By doing so, outliers will have a fewer number of nodes than the normal objects in general, and this further helps to isolate outliers. In fact we can see that the similarity measure used here is the number of *high-similarity* shared neighbors.

Finding a suitable threshold T is vital to achieve a higher outlier detection rate. If the threshold is too small, many objects including both outliers and normal ones will have a higher number of shared neighbors, and therefore it will be harder to distinguish outliers. On the other hand, if the threshold is too high, then even the normal objects might have fewer shared neighbors and the algorithm will show a high false alarm rate.

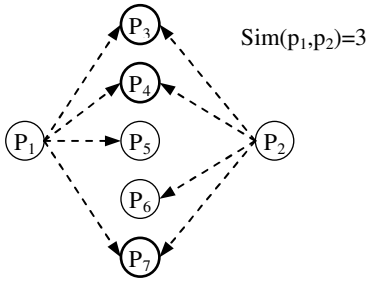

Figure 2. Shared neighbors

In order to find a suitable threshold, we consider the distribution of cosine similarity value of the corresponding data set. Let X be the set of cosine similarity values. Let μ and σ be the mean and standard deviation of X respectively. Experimentally we found that any T value inside the interval $[\mu - \sigma, \mu)$ gives higher detection rate (precision); i.e. T is any value within one standard deviation below the mean.

As we can see, the choice of the threshold depends only on the dataset (i.e. mean and standard deviation of the corresponding data set) and can be automatically derived without depending on user input. Existing algorithms such as LOF [2] and k-dist [8] use a threshold called minimum points (k) to define the neighborhood. Selection of threshold-k for these approaches is non-trivial and must be specified by the user. Also, unlike in previous approaches where precision is sensitive to the threshold, precision of our algorithm is not highly sensitive to the value of T chosen in this interval. We will discuss this further in the experimental section.

The following algorithm for OutRank-*b* shows the computation of the new similarity matrix based on shared neighbors using the cosine similarity matrix. Although it takes threshold T as an input, it can be automatically derived from the cosine similarity matrix as discussed earlier. After the similarity matrix based on the shared neighbors is derived, the rest of the computation is similar to that of OutRank-*a*.

Algorithm OutRank-*b*

Input: Cosine similarity matrix $M_{n \times n}$, threshold T, error tolerance ε.
Output: Outlier ranks.
Method:

```
1:  for i=1 to n do   // discretize M using T
2:    for j=i+1 to n do
3:      if M[i,j] ≥ T
4:        M[i][j]= M[j][i]=1;
5:      else
6:        M[i][j]= M[j][i]=0;
7:      end
8:    end
9:  end
10: for i=1 to n do   // compute new similarity scores
11:   for j=i+1 to n do
12:     let X= {M[i][1] to M[i][n]};
13:     let Y= {M[j][1] to M[j][n]};
14:     Sim[i][j]=Sim[j][i]=|X ∩ Y|;
15:   end
16:   Sim[i][i]=0;
17: end
18: call OutRank-a (Sim, ε)
```

4. Experimental Evaluation

In this section we describe the experimental environment used to evaluate our algorithms and the results obtained.

We compared the performance of our algorithms against a distance-based anomaly detection algorithm called k-dist [8] and a density-based algorithm known as LOF [2]. K-dist uses the distance between an object to its *k*-th nearest neighbor to be the outlier score. LOF, on the other hand, computes the outlier score in terms of the ratio between the densities of an object to the density of its *k* nearest neighbors.

We have performed extensive experiments on both synthetic and real data sets to evaluate the performance of our algorithms. The experiments were conducted on a SUN *Sparc* 1GHz machine with 4 GB of main memory. The data sets used for our experiments are summarized in Table 2. Here *D* is the dimension of databases and *C* is the number of clusters. Thresholds *T, K_L*, and *K_D* are parameters used with OutRank-*b*, LOF, and k-dist algorithms respectively.

2D-Data is the synthetic data set. The rest of the data sets are obtained from the UCI KDD archive. Some of these datasets contains more than one cluster of normal objects. For example, dataset *Zoo* contains 2 normal clusters and a smaller outlier cluster of 13 objects.

We employ several evaluation metrics to compare the performance of our algorithms: *Precision* (P), *False Alarm rate* (FA), *Recall* (R), and *F-measure* (F). These metrics are computed as follows:

$$P = \frac{TP}{TP + FP} \qquad FA = \frac{FP}{FP + TN} \qquad (6)$$

$$R = \frac{TP}{TP + FN} \qquad F = \frac{2TP}{2TP + FP + FN}$$

where TP (TN) is the number of true positives (negatives) and FP (FN) is the number of false positives (negatives). In all of our experiments number of actual outliers is equal to the number of predicted outliers and therefore P=R=F.

Table 2. Characteristics of the datasets

Data Set	D	C	No. of outliers	No. of instances	T	K_L	K_D
2D-Data	2	2	20	482	0.93	10	10
Austra	14	2	22	400	0.25	4	5
Zoo	16	3	13	74	0.45	40	20
Diabetic	8	2	43	510	0.80	30	30
Led7	7	5	248	1489	0.55	330	330
Lymph	18	3	4	139	0.90	2	2
Pima	8	2	15	492	0.70	20	10
Vehicle	18	3	42	465	0.95	50	30
Optical	62	8	83	2756	0.65	10	20
KDD-99	38	2	1000	11000	0.35	500	5

4.1. Comparison with other approaches

Table 3 shows the results of applying various algorithms to synthetic and real life data sets. The threshold of OutRank-*b* is selected as described in Section 3. In LOF, we have experimented with various K_L values for each dataset and selected the best K_L value that maximizes the precision. We did the same for k-dist algorithm when choosing K_D. So the result presented under LOF and k-dist represents the optimal value that these algorithms can achieve.

First let us analyze the 2D synthetic dataset, which is designed to view the difference between existing outlier detection schemes and our random walk based method. This dataset (see Figure 3) has two clusters (C_1, C_2) of normal patterns and several small clusters of outlier objects (O_1 to O_5). Note that cluster C_1 has a similar density to some of the outlying objects. Both our algorithms successfully captured all of the outliers and delivered a precision of 1.0. On the other hand LOF was unable to find some of the outlying clusters. Figure 3 shows the outlier objects detected by LOF (denoted with '+' symbol). Many of the outlying objects in O_1, O_2, and

O_3 regions were undetected. Even worse, it identified some of the normal objects in C_1 and C_2 as outliers.

Table 3. Experimental results

Data Set		OUTRANK		K-dist	LOF
		a	*b*		
2D-Data	P	1.0000	1.0000	0.8500	0.5500
	FA	0.0000	0.0000	0.0064	0.0195
Austra	P	0.7727	0.9545	0.0454	0.1363
	FA	0.0132	0.0026	0.0555	0.0502
Zoo	P	0.9230	1.0000	0.7692	0.9230
	FA	0.0163	0.0000	0.0491	0.0163
Diabetic	P	0.8837	0.8139	0.7209	0.5813
	FA	0.0107	0.0171	0.0256	0.0385
Led7	P	0.9516	0.9799	0.8467	0.2217
	FA	0.0096	0.0040	0.0306	0.1555
Lymph	P	0.5000	1.0000	1.0000	0.7500
	FA	0.0148	0.0000	0.0000	0.0074
Pima	P	1.0000	1.0000	0.9333	0.9333
	FA	0.0000	0.0000	0.0020	0.0020
Vehicle	P	0.6190	0.6428	0.1666	0.3095
	FA	0.0378	0.0354	0.0827	0.0685
Optical	P	0.5300	0.6024	0.1686	0.0722
	FA	0.0145	0.0123	0.0258	0.0288
KDD-99	P	0.8880	0.8990	0.0080	0.2520
	FA	0.0112	0.0101	0.0992	0.0748

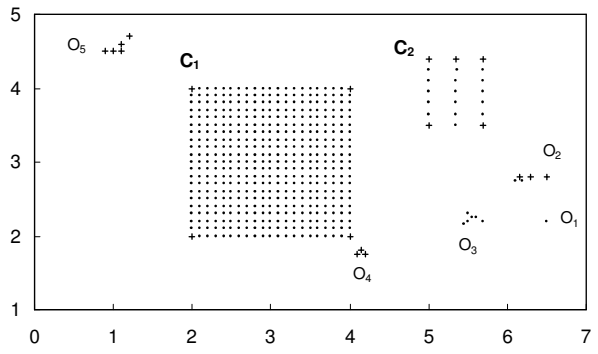

Figure 3. Learning results of LOF on *2D-Data*

We have experimented with various K_L values but LOF always had problems finding the outliers. When we use a smaller K_L value (< maximum size of outlying clusters) then it misses some of the outlying objects and identifies normal points in cluster C_1 as outliers. This is because the neighborhoods under consideration for an object in C_1 and in some outlying cluster have similar densities and therefore it is difficult to distinguish outliers. Also, when a larger K_L value is used, it identifies normal objects in cluster C_2 as outliers. On the other hand, distance based algorithms such as k-dist suffers from local density problem as described in [2]. Therefore they fail to identify outlier clusters such as O_4.

As a result distance and density based algorithms break down and deliver a higher false alarm rate.

When considering the real life data sets such as *optical* (hand written data), *kdd-99* (intrusion data) and *austra,* both density and distance based algorithms performed poorly because the datasets are high-dimensional and very sparse, a situation in which the notion of distance is likely to break down. Our algorithm performed significantly better than both LOF and k-dist with a lower false alarm rate.

Also, when analyzing the datasets with several clusters of objects such as *led7* and *optical,* performance of density based algorithms became very low. In both these datasets our algorithm showed better performance. Also, as shown in Figure 4, our algorithm shows a remarkably low false alarm rate in *led7*.

When OutRank-*b* is compared against OutRank-*a*, we found that, on average, it delivers a 20% improvement in precision. Also, a significant reduction in false alarm rate for datasets such as *Austra*, *Zoo* and *Lymph* can be seen. In *diabetic* dataset OutRank-*b* shows somewhat low precision compared to OutRank-*a*, and it is because of the choice of threshold.

4.2. Effect of the percentage of outliers

Figure 4 shows a comparison between algorithms on several large datasets when the percentage of outliers is varied. Performance of k-dist algorithm on *kdd-99* dataset was very poor and it was not graphed. Also, we have used our best approach: OutRank-*b* in these graphs. In general our algorithm delivered a consistent precision when the percentage of outliers was varied. LOF algorithm tends to decrease the precision heavily with the percentage of outliers in some cases such as in *led7*. Notice that our algorithm shows a comparably lower false alarm rate, whereas other approaches deliver typically unacceptable rate for datasets such as *led7* and *kdd-99*.

4.3. Effect of the shared neighbor approach

Here we analyze the effect of shared neighbor approach on the precision of OutRank-*b*. For the comparison we use 3 versions of OutRank algorithm. Apart from OutRank *a* and *b*, we designed a new algorithm OutRank-*c*, by applying threshold *T* on the similarity matrix used with OutRank-*a*. Note that OutRank-*b* computes shared neighbors using this matrix.

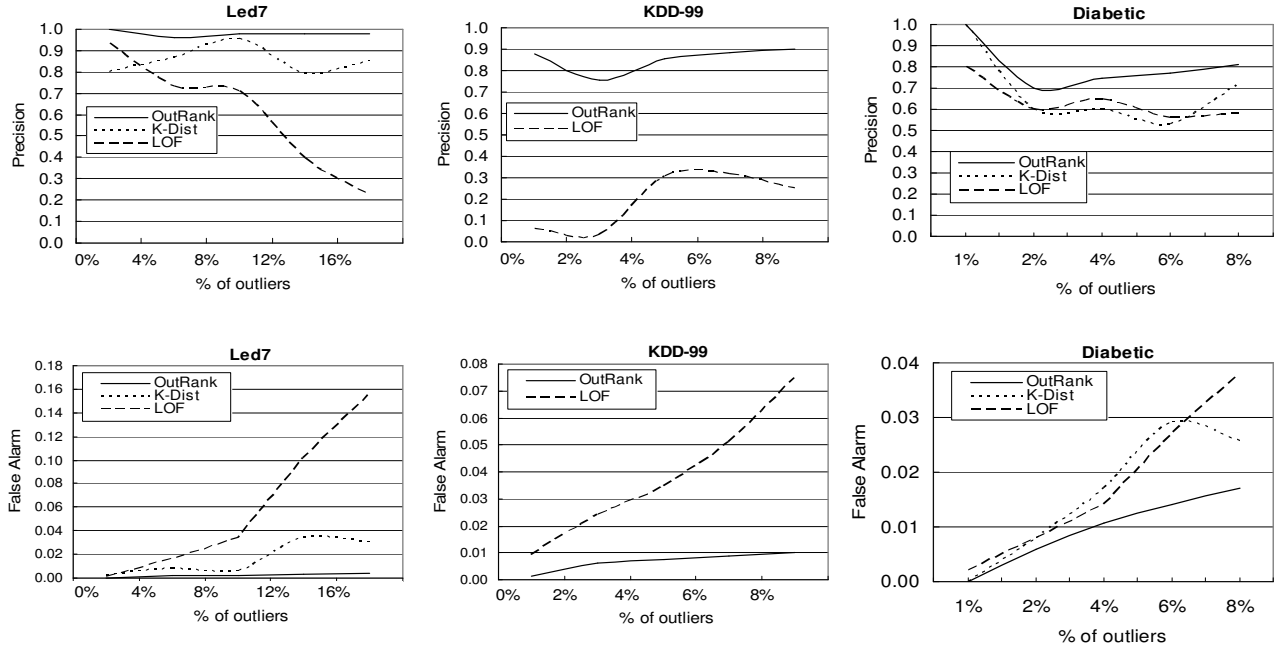

Figure 4. Precision and false alarm rate while varying the % of outliers

Figure-5 shows precision for 3 algorithms on dataset *optical*, while varying the percentage of outliers. In all cases we can see some improvement of precision with OutRank-b, over other algorithms. Also, in some situations mere discretization can have a negative effect as shown in Figure 5 for 4% case. But shared neighbor approach built upon this can minimize this effect.

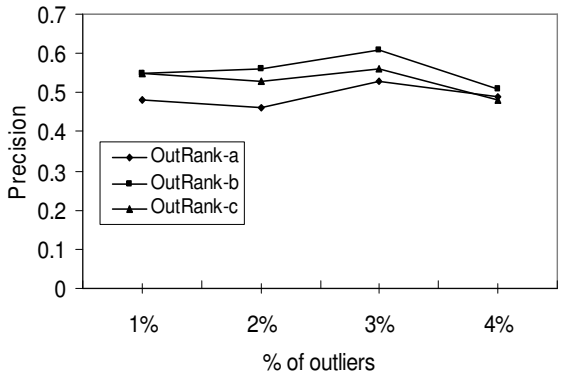

Figure 5. Precision of algorithms on *optical* dataset

4.4. Effect of threshold on the quality of solution

Let us analyze the effect of threshold T on OutRank-*b*. Note that OutRank-*a* does not use any threshold. Figure-6 shows precision for *led7* and *kdd-99* datasets when T is varied from 0.00 to 0.90. As expected, for higher and lower T values precision becomes low. Notice the interval $[\mu - \sigma, \mu)$ where our algorithm delivers the highest performance. Also, any $T \in [\mu - \sigma, \mu)$ shows similar performance in precision, and therefore our algorithm does not exhibit any unexpected sensitivity on the choice of threshold T.

5. Conclusions

This paper investigated the effectiveness of random walk based approach for outlier detection. Experimental results using both real and synthetic data sets confirmed that this approach is generally more effective at ranking most understandable outliers that previous approaches cannot capture. Also, the results revealed that our outlier detection model tends to do better when the percentage of outliers is gradually increased. In outlier detection algorithms, false alarm rate is considered as the limiting factor of its performance and the algorithms proposed here achieved the lowest false alarm rate using an unsupervised learning scheme.

There are several aspects of our framework that can still be improved. First, the cosine similarity measure that we used has both advantages and disadvantages. For example, it cannot effectively handle outliers that are co-aligned with other normal points. Despite this limitation, both OutRank-*a* and OutRank-*b* still outperform standard distance-based and density-based algorithms. We are currently investigating other similarity measures, such as those based on Euclidean-based distances, as the

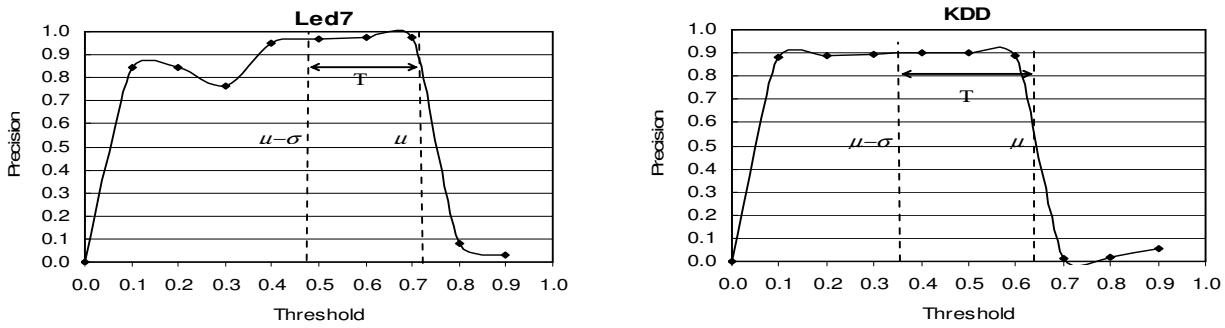

Figure 6. Precision for different threshold values

transition probabilities of our random walk model. While these measures work well in low dimensional data, they do not work well in high dimensions.

In our past work [5], we have examined semi-supervised learning techniques for outlier detection. We plan to further explore random walk based methods in a semi-supervised setting. Also, we will investigate the application of random walk algorithms for detecting anomalous substructures in graph databases.

Acknowledgement

We would like to thank the reviewers for their constructive and helpful comments.

References

[1] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc. of the ninth ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 29–38, 2003.

[2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of data*, pages 93–104, 2000.

[3] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. of the 17th Int'l Conf. on Machine Learning*, pages 255–262, 2000.

[4] G. Erkan, D. Radev. LexPageRank: Prestige in Multi-Document Text Summarization. In *Proc. EMNLP*, 2004.

[5] J. Gao, H. Cheng, and P.-N Tan. A Novel Framework for Incorporating Labeled Examples into Anomaly Detection. In *Proc of SIAM Int'l Conf on Data Mining*, Bethesda, MD, Apr 20-22, 2006.

[6] D. Hawkins. *Identification of outliers*. Chapman and Hall, London, 1980.

[7] D. L. Isaacson and R.W. Madsen, *Markov chains: theory and applications*, Wiley, New York, 1976.

[8] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proc. of the Seventh ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 293–298, 2001.

[9] T. Johnson, I. Kwok, and R. T. Ng. Fast computation of 2-dimensional depth contours. In *Proc. of the Fourth Int'l Conf. on Knowledge Discovery and Data Mining*, pages 224–228, 1998.

[10] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.

[11] C. Kruegel and G. Vigna. Anomaly Detection of Web-Based Attacks. In *Proc. of 10th ACM Conf. Computer and Comm. Security (CCS '03)*, pp. 251-261, Oct. 2003.

[12] V. B. T. Lewis. *Outliers in statistical data*. John Wiley & Sons, Chichester, 1994.

[13] R. Mihalcea, P. Tarau. TextRank: Bringing Order into Texts. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, July 2004.

[14] R. Mihalcea. Graph-based Ranking Algorithms for Sentence Extraction Applied to Text Summarization. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics, companion volume*, Barcelona, Spain, July 2004.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[16] F. Preparata and M. Shamos. *Computational Geometry: an Introduction*. Springer, 1988.

[17] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of data*, pages 427–438, 2000.