International Journal on Artificial Intelligence Tools
Vol. XX, No. X (2007) 1–18

World Scientific
www.worldscientific.com

OutRank: A GRAPH-BASED OUTLIER DETECTION FRAMEWORK USING RANDOM WALK

H. D. K. MOONESINGHE

Department of Computer Science & Engineering
Michigan State University
East Lansing, MI 48824
moonesin@cse.msu.edu

PANG-NING TAN

Department of Computer Science & Engineering
Michigan State University
East Lansing, MI 48824
ptan@cse.msu.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

This paper introduces a stochastic graph-based algorithm, called OutRank, for detecting outliers in data. We consider two approaches for constructing a graph representation of the data, based on the object similarity and number of shared neighbors between objects. The heart of this approach is the Markov chain 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 robust 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 false alarm rate are achieved using the proposed framework.

Keywords: Outlier detection; random walk; Markov chain.

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, the well-known PageRank algorithm[15] used for ranking the results of search queries is based on the model of a random surfer traversing the hyperlinks of a Web graph. 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.[1,2,3,8,10,17] 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 adjacency matrix of the graph into transition probabilities, we model the problem as a Markov chain process 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.

The random walk model is designed to find nodes that are most "central" to the graph. To illustrate, consider graph A shown on the top left panel of Fig. 1, which consists of 4 nodes connected to a central node labeled as node 1. Upon applying the random walk model to the transition matrix constructed from graph A, the probability score of each node is plotted on the right hand panel of Fig. 1. Clearly node 1 has the highest score compared to other remaining nodes. To illustrate the effect of outliers on the random walk model, consider the graph B shown in Fig. 1, which is obtained by removing the edges between nodes (3,5) and nodes (4,5) of graph A. Node 5 can be considered as an outlier of the graph. As can be seen from the probability score distribution for graph B, the random walk model assigns the lowest score to the outlying node.

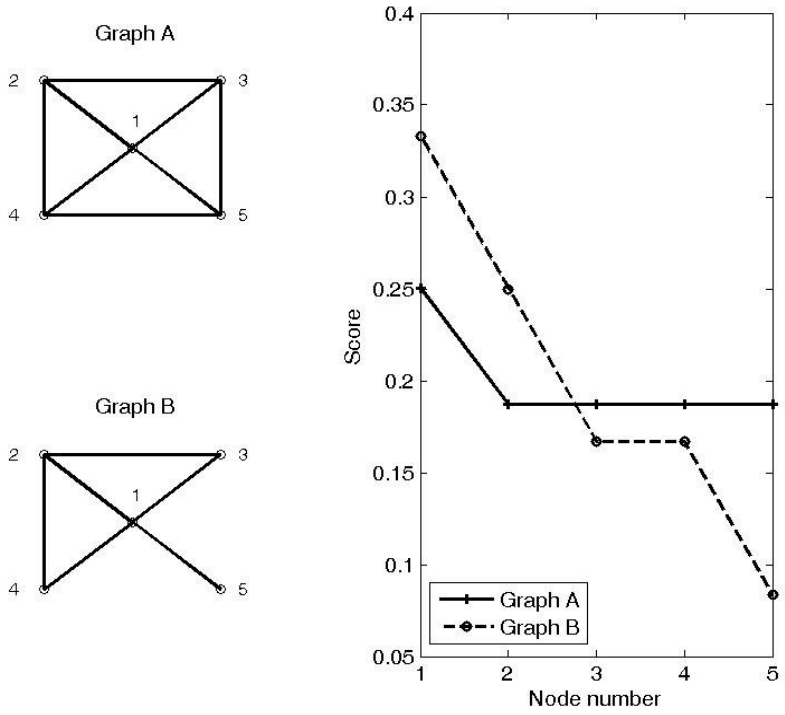


Fig. 1. Outlier detection with random walk

A major 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, one potential challenge of using the random walk approach is to determine the neighborhood graph from which the outliers can be detected. We examine two approaches for doing this: (1) building the graph using an appropriate similarity measure and (2) building the graph based on the number of shared neighbors. We perform extensive experiments to compare the performance of our approach against distance-based and density-based outlier detection algorithms. Experimental results using both real and synthetic data sets show that our proposed algorithm outperforms other approaches and yields higher detection rates with lower false alarm rates.

In summary, the main contributions of this paper are as follows:

(i) We investigate the effectiveness of random walk approach for outlier detection. More specifically, we show that our proposed framework, called *OutRank*, is capable of detecting outliers even when the normal patterns have similar densities as outliers.
(ii) We investigate two different approaches for constructing the graph upon which the random walk model is applied.
(iii) We also analyze the different choices of similarity measures on the random walk model.
(iv) 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. *Graph Representation*

In order to determine the link structure of the graph we compute the similarity between every pair of objects. Let $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$ be the vector representation of any two objects drawn from a d-dimensional space R^d. While there are many possible choices of similarity measures, we experiment with the following metrics:

Cosine Similarity. The similarity between X and Y is defined as follows:

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

RBF Kernel. The similarity between X and Y is defined as follows (where σ is a user specified kernel width parameter):

$$\text{rbf\_similarity}(X, Y) = \begin{cases} 0 & \text{if} \quad X = Y \\ \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|X-Y\|^2}{2\sigma^2}} & \text{otherwise} \end{cases} \tag{2}$$

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 pairs of objects in the database is represented by the $n \times n$ similarity matrix *Sim*, where n is the number of objects. We use the similarity matrix to represent the adjacency matrix of the graph. In the graph representation, each node corresponds to an object in the database. Two nodes X and Y are connected by an edge if their similarity is greater than zero, and the weight of the edge is taken as *Sim(X, Y)*.

2.2. *Markov chain model*

Based on the graph representation, we model the problem of outlier detection as a Markov chain process. The Markov chain modeled here corresponds to 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 the weighted votes given by other nodes in the graph. Here higher 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. The connectivity of a node is computed iteratively using the following expression.

Definition 2. (Connectivity) Connectivity *c(u)* of node *u* is defined as follows:

$$c_t(u) = \begin{cases} a & \text{if} \quad t = 0 \\ \sum_{v \in adj(u)} (c_{t-1}(v) / |v|) & \text{otherwise} \end{cases} \tag{3}$$

where *a* is its 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*.

Given *n* nodes, v_1, v_2, …, v_n, we can initially assign each node a uniform connectivity value (e.g. $c_0(v_i) = 1/n$, $1 \le i \le n$) and recursively apply Eq. (3) to refine its value, taking into account the modified connectivity values computed for its neighboring nodes. This iterative procedure is known as the power method and is often used to find the dominant eigenvector of a stochastic matrix. Upon convergence, Eq. (3) can be written in matrix notation as follows:

$$c = S^T c \tag{4}$$

where *S* is 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*. These properties follow from the well-known *Perron-Frobenius* theorem.[7]

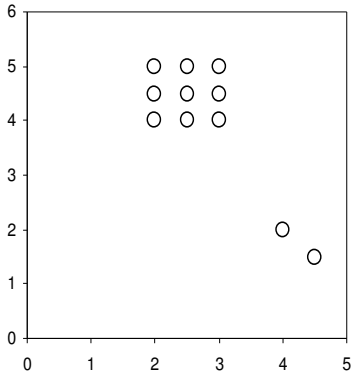
The transition matrix (*S*) of our Markov model is obtained by normalizing the rows of our similarity matrix (*Sim*) defined earlier:

$$S[i, j] = \frac{Sim[i, j]}{\sum_{k=1}^{n} Sim[i, k]} \tag{5}$$

This normalization ensures that the elements of each row of the transition matrix sum to 1, which is an essential property of a stochastic matrix. It is also assumed that the transition probabilities in S do not change over time. In general, the transition matrix S computed from data might not be *irreducible* or *aperiodic*. To ensure convergence, instead of using Eq. (4), we may compute the steady state distribution for the following modified matrix equation:

$$c = d.1 + (1-d)S^T c \tag{6}$$

where S is the row normalized transition matrix, d is known as the damping factor, and **1** is the unit column vector $[1\ 1 \ldots 1]^T$. For the proof of convergence of this equation, readers should refer to Ref. 18. Intuitively, the modification can be viewed as allowing the random walker to transit to any nodes in the graph with probability d even though they are not adjacent to the currently visited node. As an example, consider the 2-dimensional data with 11 objects shown in Fig. 2. Clearly object 1 and object 2 are outliers while the rest of the objects are normal.



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

Fig. 2. Sample 2-D data set

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

Using uniform probabilities as the initial connectivity vector and after applying Eq. (6), the connectivity vector converges to its stationary distribution after 112 iterations (where d is chosen to be 0.1). The final connectivity values and the rank for each object are shown in Table 1. Note that object-1 and object-2 are correctly identified as the most outlying objects.

3. Algorithms

This section describes our proposed algorithm based on the above random walk model for outlier detection. Two variants of the *OutRank* algorithms are presented.

3.1. *OutRank-a: using object similarity*

In OutRank-*a* algorithm, we form the transition matrix for the Markov chain model using the similarity matrix discussed earlier. We then use the power method to compute the stationary distribution of its connectivity vector. The connectivity vector is initialized to be a uniform probability distribution (1/n, where n is the total number of objects) and the damping factor is set to 0.1. The pseudo code for the OutRank-*a* algorithm 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 c0 =(1/n).1            // initialization
13: repeat
14:     ct+1= d/n + (1-d)S^T ct
15:     δ = ||ct+1 - ct||1
16:     t = t+1;
17: until (δ<ε)
18: rank ct+1 from min(ct+1) to max(ct+1)
19: return ct+1;
```

3.2. *OutRank-b: using shared neighbors*

In OutRank-*a*, nodes are considered adjacent if their corresponding similarity measure is non-zero. So even nodes with low similarity values might be 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 defined in terms of the number of neighbors shared by the objects. For example, consider two objects, v_1 and v_2. Suppose v_1 has a set of neighbors { v_3, v_4, v_5, v_7} and v_2 has a set of neighbors { v_3, v_4, v_6, v_7} (see Fig. 3). The set of neighbors shared by both v_1 and v_2 is {v_3, v_4, v_7}. In this algorithm, we take the cardinality of this set as the similarity measure.

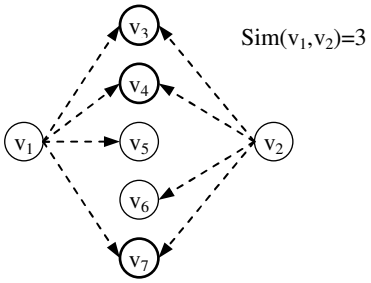


Fig. 3. Similarity based on the number of shared neighbors

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 short the similarity measure used in OutRank-*b* corresponds to 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 yield a high false alarm rate. In order to find a suitable threshold, we consider the distribution of similarity values of the corresponding data set. Let *X* be the set of similarity values. Let μ and σ be the mean and standard deviation of *X* respectively. Experimentally we observe that any *T* value within the interval [μ - σ, μ) gives higher detection rate; i.e. *T* should be chosen to be any value within one standard deviation below the mean.

As can be seen, the choice of the threshold *T* depends only on the dataset (i.e. mean and standard deviation of the corresponding data set) and can be automatically derived 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 detection rate is sensitive to the threshold parameter, the detection rate of our algorithm is not that sensitive to the value of *T*. We will demonstrate this further in the experimental section.

The pseudo code for the OutRank-*b* algorithm is summarized below. The algorithm forms the transition matrix based on the number of shared neighbors between every pair

of objects. After computing its transition matrix, the rest of the computation is similar to that of OutRank-*a*.

Algorithm OutRank-*b*

Input: 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

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:

$$P=\frac{TP}{TP+FP} \qquad FA=\frac{FP}{FP+TN} \tag{7}$$

$$R=\frac{TP}{TP+FN} \qquad F=\frac{2TP}{2TP+FP+FN} \tag{8}$$

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	
						euc	*cos*	*euc*	*cos*
2D-Data	2	2	20	482	0.93	10	20	10	16
Austra	14	2	22	400	0.25	4	2	5	12
Zoo	16	3	13	74	0.45	40	40	20	24
Diabetic	8	2	43	510	0.80	30	40	30	32
Led7	7	5	248	1489	0.55	330	390	330	220
Lymph	18	3	4	139	0.90	2	2	2	20
Pima	8	2	15	492	0.70	20	2	10	2
Vehicle	18	3	42	465	0.95	50	56	30	14
Optical	62	8	83	2756	0.65	10	2	20	40
KDD-99	38	2	1000	11000	0.35	500	500	5	200

The remainder of this section is organized as follows. In subsection 4.1, we evaluate our proposed algorithms against existing approaches, including a distance-based outlier detection algorithm called K-dist[8] and a density-based algorithm known as LOF.[2] In subsection 4.2 we analyze the performance of our algorithms by varying the percentage of outliers. In subsection 4.3, we discuss the effect of shared neighbor approach and in subsection 4.4 we analyze RBF kernel for outlier detection. Subsection 4.5 presents the effect of threshold selection for OutRank-*b*.

4.1. *Comparison with other approaches*

Table 3 shows the results of applying various algorithms to synthetic and real life data sets. The K-dist algorithm uses the distance between an object to its *k*-th nearest neighbor to be the outlier score. The LOF algorithm, 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. When experimenting with LOF and K-dist, we used 2 different distance measures: Euclidean distance and (1 – cosine) distance. In LOF, we have experimented with various K_L threshold 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 the threshold K_D (Table 2 shows all such thresholds selected under Euclidean (*euc*) and 1 – cosine (*cos*) distance measures). So the results reported in this paper for LOF and K-dist represent the optimal precision that these algorithms can achieve.

For OutRank-*a*, we chose cosine as the similarity measure. We show in subsection 4.4 that the choice of similarity measure (cosine or RBF kernel) does not affect the

performance significantly. Meanwhile, the threshold T for OutRank-*b* is determined empirically from the data using the approach described in Section 3.

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 Fig. 4) 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. Fig. 4 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		Euclidean		1 – Cosine	
		a	*b*	K-dist	LOF	K-dist	LOF
2D-Data	P	1.0000	1.0000	0.8500	0.5500	0.9500	0.5500
	FA	0.0000	0.0000	0.0064	0.0195	0.0022	0.0195
Austra	P	0.7727	0.9545	0.0454	0.1363	0.4545	0.0909
	FA	0.0132	0.0026	0.0555	0.0502	0.0317	0.0529
Zoo	P	0.9230	1.0000	0.7692	0.9230	1.0000	1.0000
	FA	0.0163	0.0000	0.0491	0.0163	0.0000	0.0000
Diabetic	P	0.8837	0.8139	0.7209	0.5813	0.5116	0.4884
	FA	0.0107	0.0171	0.0256	0.0385	0.0450	0.0471
Led7	P	0.9516	0.9799	0.8467	0.2217	0.9758	0.7419
	FA	0.0096	0.0040	0.0306	0.1555	0.0048	0.0516
Lymph	P	0.5000	1.0000	1.0000	0.7500	1.0000	0.7500
	FA	0.0148	0.0000	0.0000	0.0074	0.0000	0.0074
Pima	P	1.0000	1.0000	0.9333	0.9333	0.1333	0.1333
	FA	0.0000	0.0000	0.0020	0.0020	0.0273	0.0273
Vehicle	P	0.6190	0.6428	0.1666	0.3095	0.6190	0.2619
	FA	0.0378	0.0354	0.0827	0.0685	0.0378	0.0733
Optical	P	0.5300	0.6024	0.1686	0.0722	0.2530	0.0482
	FA	0.0145	0.0123	0.0258	0.0288	0.0232	0.0296
KDD-99	P	0.8880	0.8990	0.0080	0.2520	0.2810	0.3510
	FA	0.0112	0.0101	0.0992	0.0748	0.0719	0.0649

We have experimented with various K_L values but LOF always had problems finding the outliers. When the K_L value is less than the maximum size of the outlier clusters, then LOF may miss some of the outlying objects and identifies some of the normal points in cluster C_1 as outliers. This is because LOF computes the outlier score of an object by taking the ratio between the densities of an object and its K_L-th nearest neighbor, and if the neighborhoods under consideration for an object in C_1 and in some outlying cluster (say O_3) have similar densities then it is difficult to distinguish outliers since in this case the outlier scores computed by LOF can be similar for both objects. Also, when a larger K_L value is used, it can possibly identify 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 Ref. 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.

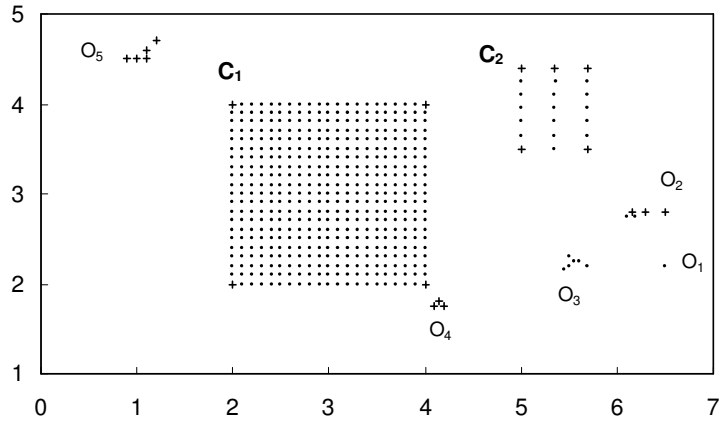


Fig. 4. Learning results of LOF on *2D-Data*

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. We speculate that there may be many small clusters of outliers in those datasets and since these algorithms are less effective in identifying such outliers their performance becomes low. Our algorithm performed significantly better than both LOF and K-dist with a lower false alarm rate, since as expected random walk model can handle this situation well.

Also, when analyzing the datasets with several clusters of objects such as *Led7* and *Optical,* performance of density based algorithms became low. In both these datasets our algorithm showed better performance. Also, as shown in Fig. 5, our algorithm shows a lower 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*

In this section we compare the performance of various algorithms when the percentage of outliers is varied. We have compared OutRank-*b* against both LOF and K-dist with Euclidean distance on three of the larger data sets as shown in Fig. 5. The performance of K-dist algorithm on *kdd-99* dataset was not graphed because its precision and false alarm

rate are considerably worse than that for LOF and OutRank-*b*. In general the precision values for OutRank-*b* do not change significantly compared to LOF and K-dist when the percentage of outliers was varied. OutRank-*b* also shows a comparably lower false alarm rate, whereas other approaches deliver typically unacceptable rate for datasets such as *Led7* and *kdd-99*.

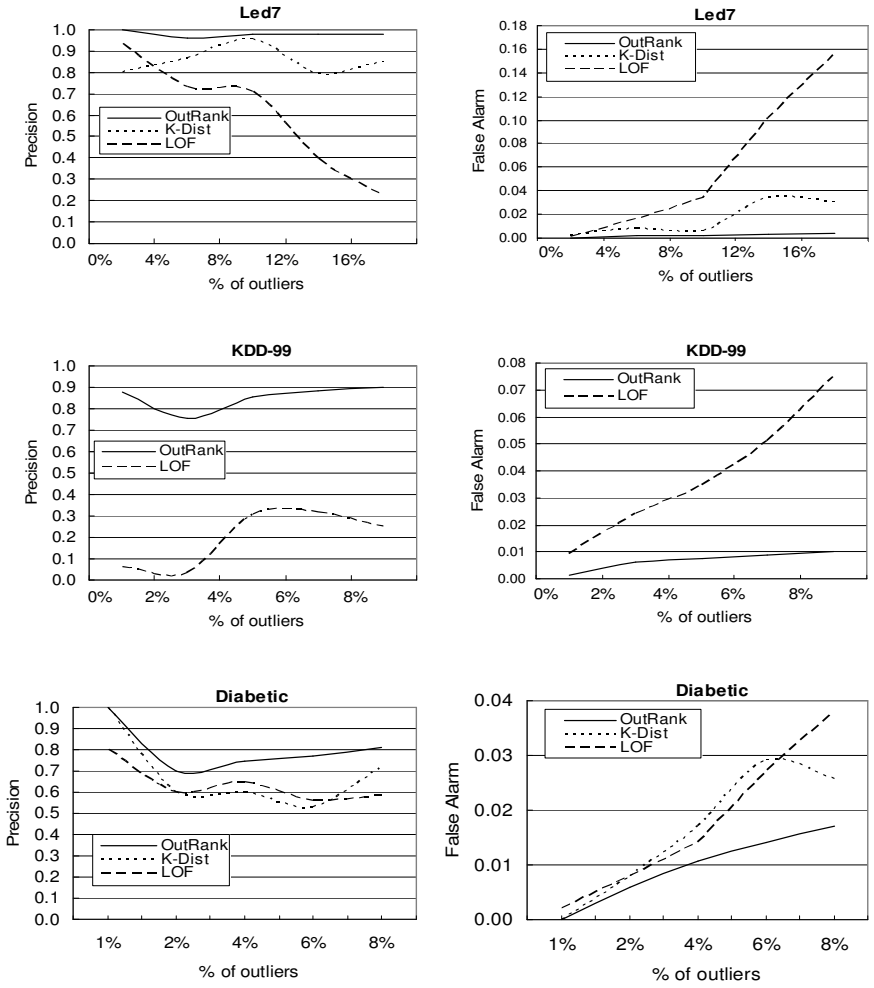


Fig. 5. Precision and false alarm rate while varying the % of outliers

4.3. *Effect of the shared neighbor approach*

Here we analyze the effect of shared neighbor approach used by OutRank-*b* on outlier detection. First we analyze the precision achieved by the shared neighbor approach. For this analysis we use 3 versions of OutRank algorithm. Apart from OutRank *a* and *b*, we design a new algorithm called OutRank-*c*, which is similar to OutRank-*a* but we apply a

threshold T on its similarity matrix to cutoff the low similarity nodes. Note that OutRank-*b* computes shared neighbors using this matrix. So, the only difference between OutRank-*b* and OutRank-*c* is that former uses shared neighbors for outlier detection. This way we can see whether the performance achieved by OutRank-*b* is resulted from the threshold effect or the shared neighbors.

Fig. 6 shows the precision for the three OutRank algorithms on the *Optical* dataset, while varying the percentage of outliers. In most cases we can see significant improvement of precision with OutRank-*b* over other algorithms. Furthermore, in situations where the percentage of outliers is sufficiently large, applying a threshold on the similarity matrix can have an adverse effect as shown in Fig. 6 for the 4% case (here, OutRank-*c* shows a slightly lower precision than OutRank-*a*), whereas the shared neighbor approach can minimize the thresholding effect.

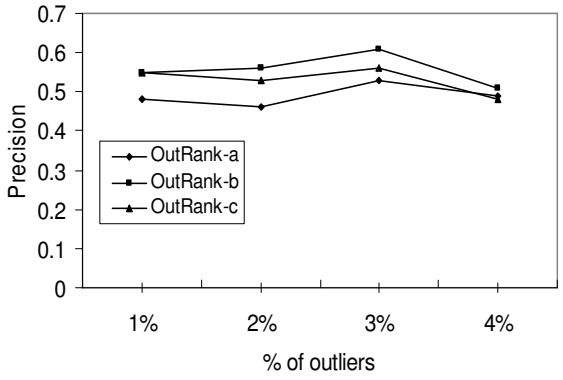


Fig. 6. Precision of algorithms on *optical* dataset

To further understand the effect of using shared neighbor approach, we have also analyzed the values of the dominant eigenvector (*connectivity*) produced by the random walk model. As shown in Fig. 7 and Fig. 8, there is a sharp distinction between the connectivity values of nodes identified as outliers from those identified as normal. The presence of such rising slope is vital towards detecting outliers using the random walk approach. When comparing the connectivity values of OutRank-*a* to OutRank-*b*, it is clear that the shared neighbor approach tends to push down the connectivity values for outliers and pulls them up for normal points. This shows that the shared neighbor approach helps to improve the distinction between outliers and normal nodes, which makes this approach more suitable for outlier detection task.

4.4. *Choice of similarity measures*

This section examines the choice of similarity measure for our proposed random walk framework. The cosine similarity measure that we used in most of our experiments cannot effectively handle outliers that are co-aligned with other normal points. As an

alternative, we consider using the RBF kernel function given in Eq. 2 to define the transition probabilities of our random walk model. Table 4 compares the precision and false alarm rates of the OutRank-*a* algorithm using RBF kernel and cosine similarity measures.

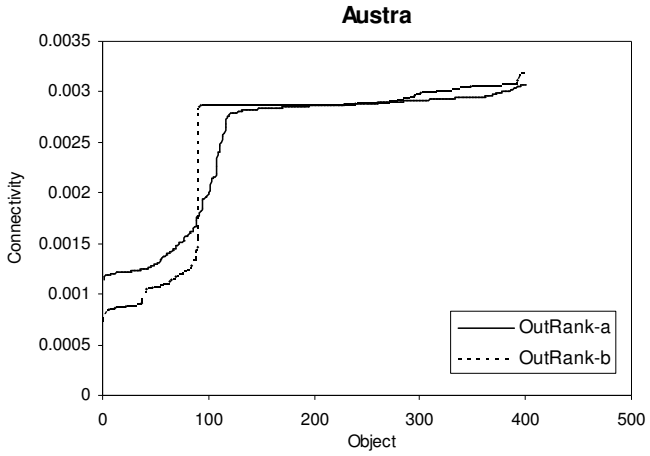


Fig. 7. Connectivity of objects in *Austra* dataset

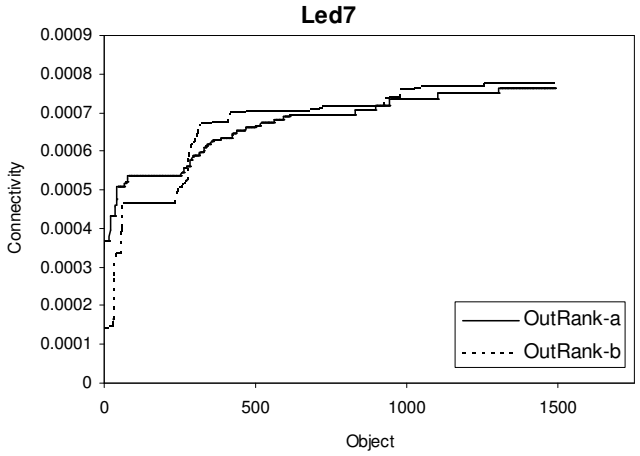


Fig. 8. Connectivity of objects in *Led7* dataset

When comparing the precision achieved by OutRank-*a* using RBF, we can see in six out of nine data sets, its precision is significantly lower than OutRank-*a* using cosine similarity. Nevertheless, when comparing the performance of OutRank-*a* using RBF against LOF and K-dist (using Euclidean distance), the RBF approach still shows better performance in most datasets. K-dist shows better performance in Diabetic, Lymph, and

Pima datasets and LOF shows better performance in Austra, Zoo, and Lymph datasets. Even in these datasets performance of RBF approach is not very low.

Table 4. Performance comparison with different similarity measures

Data Set		OutRank-*a* (RBF-Kernel)	OutRank-*a* (Cosine)
Austra	P	0.1000	0.7727
	FA	0.0529	0.0132
Zoo	P	0.7692	0.9230
	FA	0.0491	0.0163
Diabetic	P	0.6511	0.8837
	FA	0.0321	0.0107
Led7	P	0.9597	0.9516
	FA	0.0080	0.0096
Lymph	P	0.5000	0.5000
	FA	0.0148	0.0148
Pima	P	0.8000	1.0000
	FA	0.0062	0.0000
Vehicle	P	0.4762	0.6190
	FA	0.0520	0.0378
Optical	P	0.6265	0.5300
	FA	0.0115	0.0145
KDD-99	P	0.2710	0.8880
	FA	0.0729	0.0112

4.5. *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. Fig. 9 and Fig. 10 show 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.

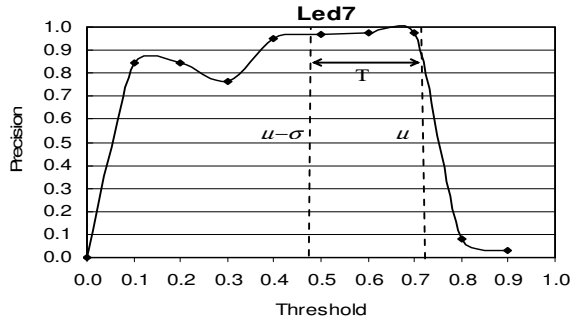


Fig. 9. Precision for different threshold values

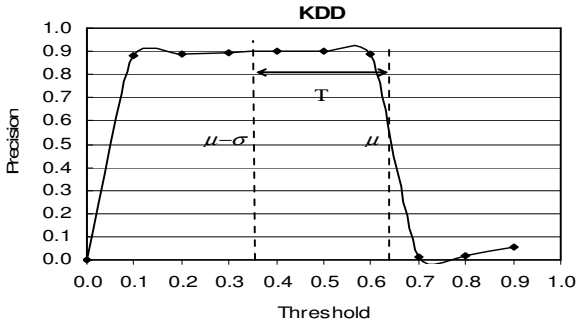


Fig. 10. Precision for different threshold values

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.
18. G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 3rd edition, 1998.