LODES: Local Density Meets Spectral Outlier Detection

Saket Sathe[*] Charu Aggarwal[†]

Abstract

The problem of outlier detection has been widely studied in existing literature because of its numerous applications in fraud detection, medical diagnostics, fault detection, and intrusion detection. A large category of outlier analysis algorithms have been proposed, such as proximity-based methods and local density-based methods. These methods are effective in finding outliers distributed along linear manifolds. Spectral methods, however, are particularly well suited to finding outliers when the data is distributed along manifolds of arbitrary shape. In practice, the underlying manifolds may have varying density, as a result of which a direct use of spectral methods may not be effective. In this paper, we show how to combine spectral techniques with local density-based methods in order to discover interesting outliers. We present experimental results demonstrating the effectiveness of our approach with respect to well-known competing methods.

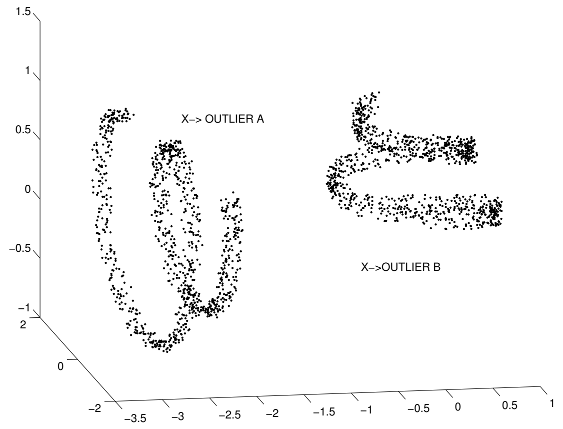


Figure 1: Outliers embedded with arbitrary manifolds.

1 Introduction.

The problem of outlier detection is to determine data records that are significantly different from other data records. Numerous methods have been proposed in the literature for outlier detection, such as linear models, proximity-based methods and subspace methods [1]. In spite of the vast variety of methods available, the outlier detection problem continues to be challenged in cases where the outliers are embedded in local nonlinear subspaces.

More recently, methods that can explore lower dimensional subspaces [2] in the data in order to discover outliers have been proposed. Most of these methods are, however, designed for discovering outliers either in subspaces of the original attributes, or in linear combinations of the underlying subspaces. In practice, the data is aligned along lower dimensional manifolds of arbitrary shape. Examples of such manifolds are illustrated in Figure 1. It is clear that both the data points A and B are outliers. Note that the shape of the relevant data patterns in localities of data points A and B can be projected to lower dimensions only if *nonlinear* mappings are used. Furthermore, the shape of the data patterns in the localities of A and B are quite different. Such outliers are unlikely to be discovered easily in lower dimensional subspaces of the data using existing subspace methods because they are hidden in local *non-linear* subspaces, which cannot be easily discovered by existing subspace methods. This problem becomes even more challenging when there is a significant variation in the local *density* of the underlying points. In such cases, one needs to account not only for the shape of the relevant nonlinear subspace but also for the variation in the underlying density.

Spectral methods have the merit that they can often discover an embedding of the data in which the lower dimensional clusters of arbitrary shape can be discovered. In addition, they have a desirable property that the use of Euclidean distances in the spectral embedding can adjust to the manifold structure of the data. Although spectral methods can often adapt to varying local subspaces and densities to a limited degree, the computations of the outlier scores will still be significantly distorted. This is, in part, because the similarity graphs in spectral methods are constructed using full dimensional distances, although the cut-off distance threshold for the k-nearest neighbor graph varies with data locality. Therefore, the use of off-the-shelf spectral methods may not be the most effective solution for the underlying task.

In an attempt to rectify the aforementioned problems, we propose the LODES algorithm. LODES combines spectral methods with local density-based methods and has the ability to adapt to the varying local properties of the data more effectively. It proposes a novel local density-based spec-

[*]IBM Research - Australia. ssathe@au.ibm.com
[†]IBM T. J. Watson Research Center. charu@us.ibm.com

tral embedding that is tailored to outlier analysis and is significantly different from state-of-the-art spectral embeddings proposed for clustering data [14, 12, 9]. Furthermore, the core idea in LODES is to iteratively adjust the similarity graph of the data points in conjunction with the results of the embedding achieved. Concretely, the local density-based spectral embedding from a previous iteration is used for improving the similarity graph for the next iteration, such that outliers are gradually segregated from inliers during these iterations. This graph-improvement process is combined with an iterative eigenspace exploration strategy that can efficiently sift through the unimportant eigenvectors and discover eigenvectors that are relevant for finding outliers hidden in local non-linear subspaces.

This paper is organized as follows. The remainder of this section discusses related work. The background and motivations are discussed in Section 2. Section 3 discusses the overall spectral embedding method that is tailored to outlier detection. In Section 4, we discuss how this spectral embedding can be leveraged to create an effective outlier scoring mechanism. Section 5 discusses the experimental results, while the conclusions are presented in Section 6.

1.1 Related Work. The problem of outlier detection has been widely studied in the literature because of its numerous applications including fraud detection and intrusion detection [1, 4]. The most fundamental class of outlier detection methods are the distance-based methods [6, 8, 13]. Techniques for speeding up distance-based methods with sampling have been discussed in [3]. The LOF method [6] uses local density in order to discover outliers. Subsequently, subspace methods were introduced in [2], in which the basic idea was to discover outliers in locality-specific subspaces of the data. Methods for discovering high-contrast subspaces, or using statistical methods to isolate important subspaces, were explored in detail in [7, 10, 11]. Angle-based methods have also been designed to perform multivariate extreme value analysis in high-dimensional data [15]. A spectral method, known as OutDST is proposed in [5]. While this interesting work is able to leverage several advantages of spectral methods, it is primarily designed for labeling outliers, as opposed to scoring them. Also, it does not fully leverage the locality-specific properties of some of the outliers. As our experimental results in Section 5 will show, the performance of OutDST can be extremely sensitive to the data set and the parameter settings. On the other hand, we provide a more robust approach that can perform robustly in most scenarios.

2 Background and Motivation.

Until now, spectral embeddings have been largely studied in the context of data clustering, particularly for discovering non-convex clusters [14, 12]. We will show that these embeddings are unsuitable for the problem of outlier detection, when they are used directly without modifications. We will correspondingly design an embedding approach that is tailored to outlier detection.

2.1 Spectral Embedding of Data Points. We assume that we have a set of m data points, denoted by $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, where each data point is an n-dimensional vector (i.e., $\boldsymbol{x} \in \mathbb{R}^n$). The process of transforming the data points such that new co-ordinates are assigned to them in the transformed space is known as *embedding* the data points. Our objective behind computing a spectral embedding is to make the task of outlier detection easier in the transformed space by applying any well-established method. We will describe two of the most popular spectral embeddings: the first was proposed by Shi and Malik, which we denote by **SM** [14] and the second by Ng *et al.* [12], which we denote by **NJW**.

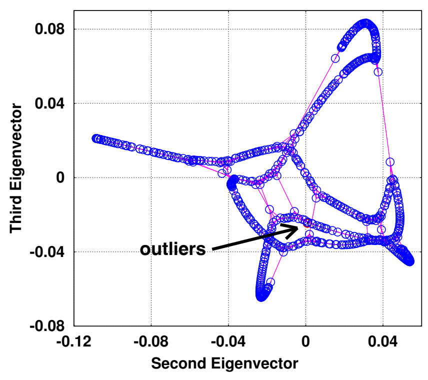
The first step of computing a spectral embedding is to construct a set R consisting of index pairs (i, j). A pair (i, j) is included in R if and only if the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are mutual k-nearest neighbors of each other. Two points are mutual k-nearest neighbors of each other if $\boldsymbol{x}_i$ is in the set of top-k similar points to $\boldsymbol{x}_j$ and vice-versa. The similarity between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is denoted as w_{ij} and is typically computed using a heat kernel. The entries in the set R can be thought of as edges of a weighted undirected graph, where the weights are the similarities w_{ij}. Such a graph is known as a **neighbourhood graph**.

In the second step the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are mapped to a vector $\boldsymbol{u}$, such that if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are highly similar, then the entries u_i and u_j are mapped close to each other. This task can be performed by solving the following optimization problem:

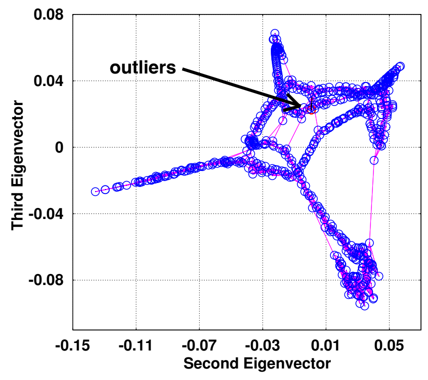
$$\begin{aligned} &\underset{\boldsymbol{u}}{\text{minimize}} \quad O = \sum_{(i,j)\in R} w_{ij}(u_i - u_j)^2 \\ &\text{subject to} \quad \boldsymbol{u}^\top \boldsymbol{D} \boldsymbol{u} = 1, \end{aligned} \tag{2.1}$$

where $\boldsymbol{D}$ is a diagonal matrix known as the **degree matrix**, and each of its diagonal entry $d_i = \sum_{j=1}^{m} w_{ij}$ is known as the **degree** of point $\boldsymbol{x}_i$. The weights w_{ij} can be materialized into a matrix $\boldsymbol{W}$ known as the **kNN matrix**. The objective function O in Eq. (2.1) can also be expressed as $O = 2\boldsymbol{u}^\top(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{u}$, where the matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is known as the **Laplacian** of the neighborhood graph and is positive semi-definite with non-negative eigenvalues.

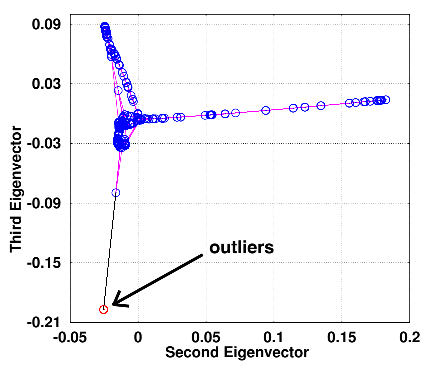
The optimization problem of Eq. (2.1) has multiple solutions. These solutions are the eigenvectors of the matrix $\boldsymbol{D}^{-1}\boldsymbol{L}$, where the eigenvectors associated with lower eigenvalues are considered better solutions. An h-dimensional embedding can be computed by storing the eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_h$ associated with the lowest h eigenvalues in a matrix $\boldsymbol{U} \in \mathbb{R}^{m\times h}$. Every row of $\boldsymbol{U}$ is a mapping of the original point $\boldsymbol{x}_i$ into a new point $\boldsymbol{y}_i \in \mathbb{R}^h$. The points $\boldsymbol{y}_i$ constitute the spectral embedding of the points $\boldsymbol{x}_i$. The embed-



(a) **SM**



(b) **NJW**



(c) A desired embedding

Figure 2: The second and third eigenvectors of the normalized Laplacian of **SM** and **NJW** embeddings are shown when $k = 6$. An arrow is drawn to show where the outliers are embedded. In the **SM** and **NJW** embeddings, the outliers are not clearly separated. The ideal kind of embedding (drawn using the embedding of this paper) is also shown in the final figure.

ding technique described so far is known as the **SM** spectral embedding. The matrices $\boldsymbol{D}^{-1}\boldsymbol{L}$ and $\boldsymbol{D}^{-1}\boldsymbol{W}$ are known as the normalized Laplacian and kNN matrix respectively. In the **NJW** embedding the normalized Laplacian and kNN matrices are given as $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ and $\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ respectively. The normalized matrices of **SM** are asymmetric while **NJW** are symmetric.

2.2 Problems with SM and NJW. The key problem with **SM** and **NJW** is that the embedding obtained by them is unsuitable for outlier detection. We will illustrate this point with a toy example shown in Figure 3. This data contains two intertwined spirals representing normal groups of blue data points. In addition, a group of 5 outliers are shown in red. The edges of the neighborhood graph are also shown in Figure 3.

As is common in spectral embeddings, we plot the second and the third eigenvector of the normalized Laplacian for **SM** and **NJW** in Figure 2(*a*) and (*b*) respectively. Observe that in both these types of embeddings, the outliers are entangled with the inliers. Their exact position in each case is shown by an arrow. This situation does not improve even if more eigenvectors are included. Such a scenario is common with spectral embeddings that are optimized to the clustering problem.

What is the reason for this behavior of the spectral embeddings? To answer this question, let us look at the non-diagonal entries $\overline{w}_{ij}$ of the normalized kNN matrix of **SM** and **NJW**:

$$(2.2) \qquad \textbf{SM:} \quad \overline{w}_{ij} = \frac{w_{ij}}{d_i}, \qquad \textbf{NJW:} \quad \overline{w}_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}.$$

The larger the value of $\overline{w}_{ij}$, the closer the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ will be embedded. On the other hand, the edges that connect outliers to inliers generally have significantly lower weight. The smaller normalization factors will tend to increase the weights of edges connecting outliers and inliers. This leads to a situation where the degree-based normalization does not lead to significantly different weights even for edges that connect outliers to inliers. Several straightforward adjustments of this principle do not seem to help significantly.

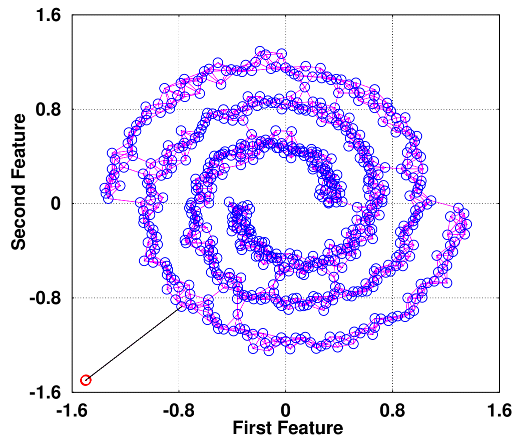


Figure 3: Toy data set where outliers are shown in red.

3 Local Density-Based Spectral Embedding.

In order to address the aforementioned problems, we will propose a spectral embedding that is extensively used by LODES for detecting outliers in local non-linear subspaces. This embedding is specifically designed for the outlier detection task and is based on the notion of local densities. It also has the ability to be used iteratively for slowly separating outliers from inliers. The spectral embedding used by LODES heavily relies on the following three concepts:

Definition 1. (Local Density) *The* local density *of a data point $\boldsymbol{x}_i$ is the density of data points in its neighborhood. We use the degree d_i of a data point as a proxy for the local density.*

This definition is based on the intuition that the higher the degree of a data point, the greater its local density. Although one could define the local density in a variety of ways, we choose this particular definition because of its natural connections with spectral clustering.

Definition 2. (Linkage Density) *Given two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the* linkage density *is the density in the region between these two data points. We use the similarity w_{ij} between data points as a proxy for the linkage density.*

This definition is based on the intuition that edges of larger weight are generally placed in dense regions.

Definition 3. (Symmetric Density Difference) *Given two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the symmetric density difference q_{ij} measures the squared difference in the local densities of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$:*

$$q_{ij} = (d_i - d_j)^2.$$

This quantity is symmetric because $q_{ij} = q_{ji}$. Based on these definitions, LODES proposes a normalized kNN matrix that is particularly effective for outlier detection. Let us denote the normalized kNN matrix of LODES as $\overline{\boldsymbol{W}}^{(\ell)}$. We define each entry $\overline{w}_{ij}$ in this matrix as follows:

(3.3)
$$\overline{w}_{ij} = \frac{\text{Linkage Density}}{\text{Symmetric Density Difference}} = \frac{w_{ij}}{q_{ij}} = \frac{w_{ij}}{(d_i - d_j)^2}.$$

The formulation in Eq. (3.3) can be intuitively explained as follows. A lower linkage density and a higher symmetric density difference indicates that points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from regions with very different local densities and should be mapped away from each other. On the other hand, when the linkage density is high and the symmetric density difference is low, it indicates that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to a region of high density and similar local density regions. Such points are therefore mapped close to each other. An important observation about outliers is that the *local* density difference between an outlier and its nearest neighbors is typically much larger. Therefore, the proposed adjustment explicitly incorporates this factor into the mapping.

Next, this normalized kNN matrix $\overline{\boldsymbol{W}}^{(\ell)}$ is used for computing the degree matrix $\overline{\boldsymbol{D}}^{(\ell)}$. The corresponding Laplacian is written as follows:

$$\overline{\boldsymbol{L}}^{(\ell)} = \overline{\boldsymbol{D}}^{(\ell)} - \overline{\boldsymbol{W}}^{(\ell)}. \tag{3.4}$$

Then, the first r eigenvectors of $\overline{\boldsymbol{L}}^{(\ell)}$ are computed. Like before, the row $\boldsymbol{y}_i$ corresponding to the original data point $\boldsymbol{x}_i$ is the new coordinate of $\boldsymbol{x}_i$ in the embedded space. The embedded points $\boldsymbol{y}_i$ often show clearer separation of outliers from inliers, as compared to the original data points $\boldsymbol{x}_i$. The process of computing the local density-based embedding of the data points $\boldsymbol{x}_i$ is known as **local densification**. We demonstrate the spectral embedding created by this adjusted methodology on the toy example of Figure 3. The resulting embedding is shown in Figure 2(*c*). It is clear that this embedding exposes the outliers much better than the ones obtained in Figure 2(*a*) and (*b*).

While the basic local density-based normalization has a clear advantage over a vanilla spectral embedding, it still does not fully leverage the advantages of local density differences. In the next section, we will see how to build on this basic idea, and iteratively refine the local densification process to derive the maximum advantage from this approach. After a few iterations of the local densification process have been performed, the spectral embedding from the final round is used for computing the outlier scores.

4 The LODES Algorithm.

A straightforward way to detect and score outliers is to apply a k-nearest neighbor approach on the local density-based embedding proposed in the previous section. Although such an approach may detect some of the more obvious outliers, many outliers may not be uncovered by it. Even though the use of local density helps in tailoring the embedding to outlier detection to some extent, the overall algorithmic framework of discovering the eigenvectors in one shot is still optimized to clustering rather than outlier analysis. In fact, the presence of outliers can itself interfere with the discovery of an effective embedding. As a result, the more obvious outliers can often interfere with the discovery of less obvious outliers. Therefore, a more refined approach is required, which as we will see later, takes the form of an iterative algorithm.

The main challenge is caused by the fact that the presence of some of the eigenvectors in the embedding can subtly mask the presence of specific outliers. Therefore, it is important that these eigenvectors are distinguished and separated from the remaining eigenvectors to achieve a more effective analysis in later iterations. In many cases, the removal of some of the eigenvectors corresponds to removing subsets of data points that interfere with the local analysis. As a result, the local analysis becomes even more localized and sharpened. Therefore, an important aspect in this context is to identify the nature of such eigenvectors. In the following, we will discuss some common characteristics of these eigenvectors.

Sparse Eigenvectors: Such eigenvectors have a large number of zeros and only a small number of non-zero entries. The data points corresponding to these non-zero entries can often be marked as obvious outliers, as these data points are disconnected from all other data points in the neighborhood graph. For illustration, consider the following sparse eigenvector: $\boldsymbol{u}_2 = (0.5, 0, 0, 0, 0)^\top$. This eigenvector indicates that the data point $\boldsymbol{x}_1$ is an outlier. Since it contains useful information about outliers, these eigenvectors are retained separately by LODES and are used during the outlier scoring stage. Nevertheless, the presence of such an eigenvector often prevents the discovery of a low-dimensional embedding containing all the outliers especially if there are a modestly large number of such eigenvectors.

Low-Cardinality Eigenvectors: Such eigenvectors contain a small number of distinct values. Intuitively, such eigenvectors represent a small number of distinct groups of data points. These eigenvectors embed the data points as groups, where all points that have the same component in the eigenvector belong to the same group. While such an eigenvector is optimized to clustering, it does little in terms of exposing the outliers in the underlying data. To illustrate this point, consider the following example of an eigenvector: $\boldsymbol{u}_2 = (0.5, 0.5, -0.2, -0.2, -0.2)^\top$. This eigenvector contains 2 distinct values 0.5 and −0.2 and therefore two groups of size 2 and 4 respectively. This eigenvector provides little information about the outliers in the underlying data.

It is noteworthy that sparse eigenvectors have smaller eigenvalues as compared to low-cardinality eigenvectors because they often represent full disconnection between small groups of the points and the remaining neighbourhood graph. In the following, we will discuss how LODES can effectively handle these two types of eigenvectors.

4.1 An Iterative Approach. The basic iterative approach that is followed by LODES starts by computing a neighborhood graph in the first iteration, and then iteratively modifying the weights of the edges in subsequent iterations. The weights of the edges are successively modified in subsequent iterations by using the distances between points in the locally-densified spectral embedding of the previous iteration. The goal of this iterative modification is to successively expose relevant outliers, and also iteratively store away useful eigenvectors found in earlier iterations. In order to achieve this goal, the sparse and low-cardinality eigenvectors are always removed from the spectral embedding of a given iteration before the weights are recomputed. Note that only the weights of the neighborhood graph change in successive iterations, and the edges of the neighborhood graph never change across iterations.

We start by describing the first iteration of LODES and then describe how the information from one iteration is used in subsequent iterations to progressively separate the outliers. The pseudocode for the LODES algorithm can be found in Algorithm 1. The iteration number is denoted by the variable t. The first step, on Line 4, is to estimate the bandwidth of the heat kernel used for computing similarities. This is done using the following rule-of-thumb. We sample P random pairs of data points and compute the following:

$$\sqrt{\frac{1}{|P|}\sum_{(i,j)\in P}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}. \tag{4.5}$$

This value is used as an estimate of the bandwidth σ, which is used for computing similarities and the corresponding kNN matrix $\boldsymbol{W}_1$. Note that the iteration number is written as a subscript of the concerned matrix; for example, $\boldsymbol{W}_1$ is the kNN matrix from the first iteration, i.e. $t = 1$. Then, on Line 12, the locally dense and normalized Laplacian $\overline{\boldsymbol{L}}_1^{(\ell)}$ is computed as discussed in Section 3. The eigenvector matrix $\boldsymbol{U}_1$, with columns containing eigenvectors, is computed in Line 13. It is assumed that the columns in $\boldsymbol{U}_1$ are sorted in order of increasing eigenvalues.

The next step is to select a set of r eigenvectors in this order that address the sparsity and low-cardinality problems. An important observation is that the sparse eigenvectors are almost always the leading (i.e., leftmost) eigenvectors. Therefore, they can be removed by simply increasing the column index a of leftmost eigenvector (in matrix $\boldsymbol{U}_1$) passed into the next iteration. A threshold $\delta \in (0, 1)$ is used to determine the number of leftmost eigenvectors to drop. All the leftmost consecutive eigenvectors containing less than $\delta \cdot m$ non-zero entries are assumed to be sparse. The value of a is incremented just enough so that $\boldsymbol{u}_a$ is not sparse. This is achieved by the function `handleSparsity()`, which is called on Line 14 and is described from Lines 22-31. The next step involves scanning the columns of $\boldsymbol{U}_1$ from left to right and determining a value of b, such that at least r eigenvectors in $\boldsymbol{u}_a, \ldots, \boldsymbol{u}_b$ do not have low cardinality. An eigenvector is said to have low cardinality, if for a user-defined threshold $\tau \in (0, 1)$, there are less than $\tau \cdot m$ distinct values in the eigenvectors. This is achieved by the procedure `handleLowCardinality()`, which is called on Line 15 and described from Lines 33-42. In practice, reasonably small values (typically less than 4–6%) of the sparsity and cardinality thresholds are sufficient for LODES to function correctly. Additionally, LODES is robust to minor inconsistencies while setting these thresholds. These important properties of LODES are validated with extensive experimental evaluation in Section 5.4.

It is noteworthy that the sparse eigenvectors at the leading end are useful for discovering outliers and the corresponding data points are continuously added to a set $\mathcal{R}$. As we will see later, the outlier scores of these points are set to arbitrarily high values at the very end of the algorithm. The embedding given by the eigenvectors $\boldsymbol{u}_a, \ldots, \boldsymbol{u}_b$ is used in the subsequent iterations for a *similarity update step* in which the weights of the neighborhood graph are readjusted.

4.1.1 Similarity Update Step. In subsequent iterations, the weight matrix $\boldsymbol{W}_t$ is readjusted using the eigenvectors derived in the previous iteration from matrix $\boldsymbol{W}_{t-1}$. Note that no new edges are added to or removed from the neighbourhood graph. We use the eigenvectors $\boldsymbol{u}_a, \ldots, \boldsymbol{u}_b$ derived from the spectral embedding in the previous iteration and compute the similarity matrix $\boldsymbol{W}_t^{(sim)}$, where each entry (i, j) is the similarity between the embedded data points i and j using

Algorithm 1 The LODES spectral outlier detection algorithm.

```
Input: Data points: $X$, Number of mutual kNN: $k$, Sparsity
    and cardinality thresholds: $(\delta, \tau)$, Window size of eigen-
    vectors: $r$, Number of iterations: $T$
Output: Outliers scores $\{c_1, \dots, c_m\}$
 1: for $t = 1$ to $T$ do
 2:     if $t = 1$ then
 3:         $\hat{X} \leftarrow X$, $a \leftarrow 2$
 4:         Compute bandwidth $\sigma$ of $\hat{X}$ using Eq. (4.5)
 5:         Let $W_t$ be kNN matrix of $\hat{X}$ with bandwidth $\sigma$
 6:     else
 7:         $\hat{X} \leftarrow U_{t-1}(:, a : b)$
 8:         Compute bandwidth $\sigma$ of $\hat{X}$ using Eq. (4.5)
 9:         Let $W_t^{(sim)}$ contain all pairwise similarities of $\hat{X}$
            with bandwidth $\sigma$
10:         $W_t \leftarrow W_t^{(sim)} \circ W_{t-1}$
11:     end if
12:     $\overline{L}_t^{(\ell)} \leftarrow$ Compute Laplacian of $W_t$ as discussed in
        Section 3
13:     $U_t \leftarrow$ Compute the eigenvector matrix of $\overline{L}_t^{(\ell)}$
14:     $(a, \overline{\mathcal{R}}) \leftarrow$ handleSparsity($U_t, a, \delta$)
15:     $b \leftarrow$ handleLowCardinality($U_t, a, \tau, r$)
16:     $\mathcal{R} \leftarrow \mathcal{R} \cup \overline{\mathcal{R}}$
17: end for
18: $\{c_1, \dots, c_m\} \leftarrow$ outlierScore($U_T(:, a : b), k$)
19: Set the score of the points in $\mathcal{R}$ to $\max(\{c_1, \dots, c_m\})$
20: return $\{c_1, \dots, c_m\}$
21:
22: function handleSparsity($U, a, \delta$)
23:     $\hat{a} \leftarrow a$, $\mathcal{R} \leftarrow \emptyset$
24:     for $j = a$ to $m$ do
25:         if $|u_j \neq 0| \leq m \cdot \delta$ then
26:             $\hat{a} \leftarrow j + 1$
27:             $\mathcal{R} \leftarrow \mathcal{R} \cup \{i | u_{ij} \neq 0\}$
28:         end if
29:     end for
30:     return $(\hat{a}, \mathcal{R})$
31: end function
32:
33: function handleLowCardinality($U, a, \tau, r$)
34:     $b \leftarrow a$, $v \leftarrow 0$
35:     while $v < r$ and $b < m$ do
36:         $b \leftarrow b + 1$
37:         if $|u_b|_{\neq} > m \cdot \tau$ then
38:             $v \leftarrow v + 1$
39:         end if
40:     end while
41:     return $b$
42: end function
```

the same heat kernel approach. These similarity values are used to update the original kNN matrix W_{t-1} in the previous iteration to obtain W_t as follows:

$$W_t = W_t^{(sim)} \circ W_{t-1}, \tag{4.6}$$

where the symbol $\circ$ denotes the *Hadamard product*. The Hadamard product between two matrices corresponds to the element-wise multiplication between the two matrices. The idea of performing the Hadamard product is to change the relative values of the various entries in the kNN matrix W_{t-1} to more closely reflect the similarities in the sanitized spectral embedding from the previous iteration. This has the effect of keeping the same edges in the neighbourhood graph, but only modifying the similarity between the embedded data points. This update procedure helps the nodes in the neighbourhood graph to move closer or away from each other between iterations as the eigenvectors are refined. This approach accelerates the densification process and the detection and separation of the outliers hidden in local non-linear subspaces.

Algorithm 2 outlierScore() – Outlier scoring algorithm.

```
Input: Spectral embedding $U_T$, number of nearest neigh-
    bours $k$.
Output: Outlier scores $\{c_1, \dots, c_m\}$
 1: for $i = 1$ to $m$ do
 2:     Initialize $\Delta_1^{max} \dots \Delta_k^{max}$ to 0
 3:     for $j = 1$ to $k$ do
 4:         $p_j \leftarrow j$th nearest neighbor distance of $y_i$ in $U_T$
 5:         $\Delta_j \leftarrow p_j - p_{j-1}$
 6:         $\Delta_j^{max} \leftarrow \max_{s=1}^{j} \Delta_s$
 7:     end for
 8:     $c_i \leftarrow \frac{\sum_{j=1}^{k} \Delta_j^{max}}{k}$
 9: end for
10: return $\{c_1, \dots, c_m\}$
```

4.2 Outlier Scoring. The last step of the LODES algorithm computes the outlier scores c_i corresponding to each data point x_i. The spectral embedding U_T, which is obtained in the final iteration, is used for computing the scores. For each point y_i that is embedded according to U_T, the distance p_j to its jth nearest neighbor is computed for each $j \in \{1, \dots, k\}$. The embedded representation U_T is used for computing distances. For each value of $j \in \{1, \dots, k\}$ we compute the value of $\Delta_j = p_j - p_{j-1}$ and compute $\Delta_j^{max} = \max_{s=1}^{j} \Delta_s$. In this computation, the value of p_0 is 0 for the case where $j = 1$ and $\Delta_j = p_1 - p_0$. The outlier score of x_i is simply the arithmetic mean of $\Delta_1^{max}, \dots, \Delta_k^{max}$. The scoring approach works well because the spectral embedding is tailored for finding outliers that are well-separated from the inliers based on their local density difference and subsequent sanitization of eigenvectors. As we will show in the experimental section (Section 5), the resulting approach shows superior results compared to the state-of-the-art methods.

4.3 Computational Complexity. Let us consider the complexity of a single iteration of LODES. The cost of computing the kNN matrix is $\mathcal{O}(nm \log m)$, with the use of a *k-d tree* index. The kNN matrix and the Laplacian can be constructed in $\mathcal{O}(mk)$ time by using a sparse representation of the matrix. The eigendecomposition of a sparse Laplacian matrix containing $O(km)$ non-zero entries is given by $\mathcal{O}((m+km)g+g^2)$. Here, g is an user-defined integer that is much smaller than m (based on the Lanczos method). Therefore, the cost of a single iteration of LODES is dominated by the $O(km)$ number of entries in the sparse matrix. For T iterations this cost is given as $O(Tkm)$. As will be demonstrated in the experiments performed in Section 5, in practice the number of iterations required by LODES to separate outliers is insignificant as compared to the number of data points.

5 Experimental Evaluation.

In this section we perform an extensive experimental analysis of LODES and compare its performance with state-of-the-art outlier analysis methods. We start by discussing the details of the data sets in Section 5.1. The baselines are described in Section 5.2. The accuracy of LODES is compared with the baselines in Section 5.3. Finally, in Section 5.4, we analyze how sensitive the accuracy is to the parameter k.

5.1 Data Sets. We used 11 real data sets from the UCI Machine Learning Repository[1]. The important features of all these data sets are given in Table 1. In data sets with unbalanced classes, the majority classes are labeled as inliers, while the minority class is labeled as outliers. In data sets with reasonably balanced classes, a minority class is created by uniformly down-sampling one of the majority classes. The summary results are presented for all the 11 data sets. However, because of space constraints, more detailed results, such as the full Receiver Operating Characteristics (ROC) and sensitivity curves are presented only for a subset of 4 data sets, which are **Glass**, **Pendigits**, **Ecoli**, and **Vowels**. We refer to these four data sets as the **primary data sets**.

5.2 Baselines. We use a combination of state-of-the-art density-based techniques (LOF [6]), angle-based methods (FastABOD [15]), subspace exploration techniques (HiCS [7]), and spectral techniques (OutDST [5]) as baselines. LODES and OutDST use the bandwidth σ to construct the kNN matrix. Its value is estimated using Eq. (4.5). There are a few other parameters that are required by HiCS and OutDST, they are set to their default values. Concretely, for HiCS we set $M = 50$, $\alpha = 0.1$, $candidate\_cutoff = 400$. As OutDST requires the percentage of outliers present in the data set, this quantity is provided to it as input. OutDST's parameter γ is varied from 0.5 to 0.8 and a value of γ is chosen, such that the difference between the number of outliers found and present in the data set is minimal. The value of γ corresponding to the minimal difference is used for generating the final results. Lastly, for LODES the *default parameter setting* is as follows: $k = 10$, $r = 2$, $\tau = 1\%$, $\gamma = 2\%$, and we always executed the algorithm for 10 iterations. The value of k was always the same across all algorithms. Unless otherwise specified, these default parameter settings are used.

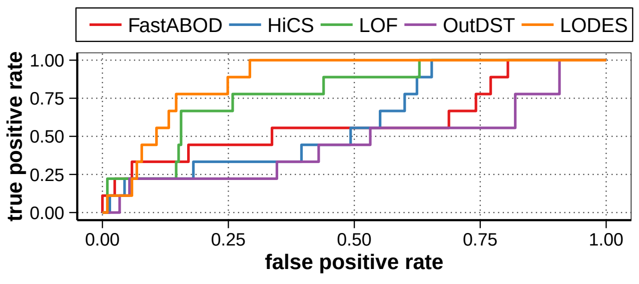
[1] `https://archive.ics.uci.edu/ml/datasets.html`

Table 1: Summary of the data sets.

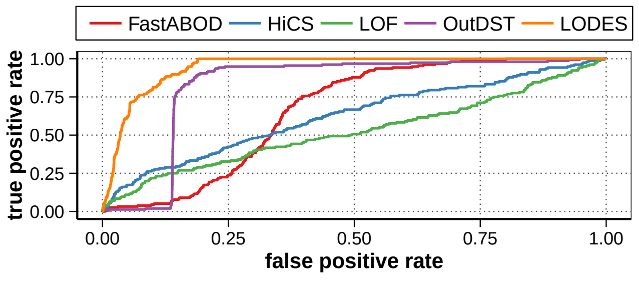
data set	points	attributes	percent outliers (%)
Glass	214	9	4.2
Pendigits	6870	16	2.2
Ecoli	336	7	2.6
Vowels	1456	12	3.4
Cardio	1831	21	9.6
Wine	129	13	7.7
Thyroid	3772	6	2.4
Vertebral	240	6	12.5
Yeast	1364	8	4.7
Seismic	2584	11	6.5
Heart	224	44	4.4

5.3 Comparing Accuracy. In this section, we will compare the accuracy of LODES with the baselines. The aforementioned default parameter settings are used for LODES. We used the *Receiver Operating Characteristics (ROC)* curves to generate the full trade-off between the true positive rate (recall) and the false positive rate. A summary measure of the accuracy is the area under the ROC curve (AUC). This summary measure is shown across all data sets, although the detailed ROC curves are shown only on the primary data sets. The ROC curves for these data sets are shown in Figure 4(a) to (d). It can be observed that the ROC of LODES significantly dominates the baselines. The key reason for this is the local-density based spectral embedding and the superior eigenspace exploration strategy used by LODES to navigate through the eigenvectors that are important for discovering outliers.

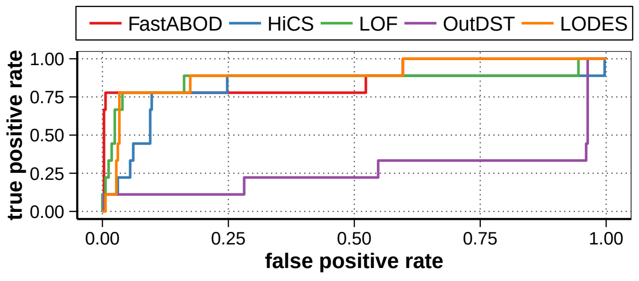
It is noteworthy that the OutDST method does not perform particularly well over the various data sets in spite of being a spectral method like our technique. This difference can be attributed to the tailoring of the iterative spectral method in our approach and the local density-based spectral embedding used for outlier analysis. Particularly interesting is the behavior of LoF, HiCS and LODES on the **Vowels** data set. The ROC curve for **Vowels** is shown in Figure 4(d). Although the AUC of HiCS and LoF on **Vowels** is better than LODES



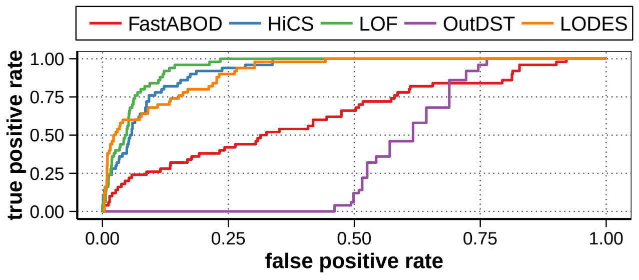
(a) **Glass**



(b) **Pendigits**



(c) **Ecoli**



(d) **Vowels**

Figure 4: ROC curves for the primary data sets.

by 1% and 3.5% respectively, LODES demonstrates the ability to capture outliers significantly earlier than HiCS. The large initial jump in the ROC of LODES can be attributed to the outliers detected in the sparse eigenvectors, which are found through successive local-densification of the spectral embedding. A similar initial jump in the ROC of LODES can be observed for the **Pendigits** data set shown in Figure 4(b).

Next, the AUC of all the methods and data sets is shown in Table 2. For a given data set only the top-2 winners are highlighted in bold. Except for the **Seismic** and **Vowels** data sets, the AUC of LODES is among the top-2 winners for all data sets. In some data sets, such as **Glass**, it shows significant improvement even over the second best result. In cases where it is second-best, it is usually a close second to the best-performing result. The main point here is that LODES is extremely consistent across different data sets compared to the state-of-the-art methods. This demonstrates the ability of LODES to detect outliers in high-dimensional manifolds, which are hard to detect using other techniques. At the same time, LODES can also easily find outliers that can be detected by other methods. Thus, the combination of local density and spectral techniques enables consistent and reliable detection of outliers in a large number of real data sets.

Table 2: Comparison of AUC. The top-2 AUCs are shown in bold, while the rest are greyed.

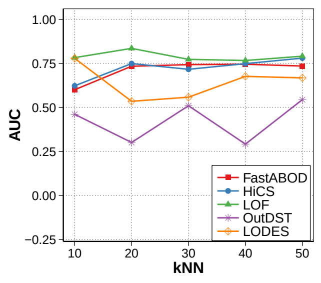
Data Set	LODES	HiCS	OutDST	FastABOD	LOF
Glass	**87.31**	60.4	46.61	60.05	**78.26**
Pendigits	**94.4**	62.42	**82.89**	66.39	52.55
Ecoli	**89.29**	81.34	26.5	**87.32**	86.30
Vowels	91.14	**92.17**	40.01	63.67	**94.67**
Cardio	**72.08**	63.02	30.78	**94.52**	59.67
Wine	**96.6**	48.5	**92.43**	82.5	62.18
Thyroid	**68.40**	**76.82**	51.2	55.58	67.14
Vertebral	**58.20**	56.60	48.2	34.80	**59.39**
Yeast	**81.4**	59.48	69.89	**84.55**	56.44
Seismic	63.43	60.58	**66.79**	**70.91**	57.39
Heart	**59.06**	52.10	**56.25**	40.14	30.28

Table 3: Comparison of F1-score for 10% highest ranked data points. The top-2 F1-scores are shown in bold, while the rest are greyed.

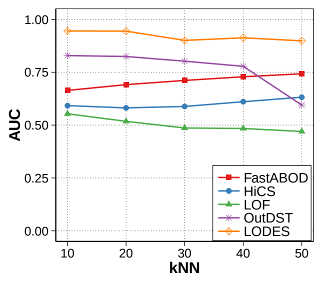
Data Set	LODES	HiCS	OutDST	FastABOD	LOF
Glass	**0.263**	0.132	0.132	**0.197**	0.132
Pendigits	**0.285**	**0.097**	0.007	0.017	0.081
Ecoli	**0.328**	0.188	0.047	**0.328**	**0.328**
Vowels	**0.328**	**0.328**	0.000	0.133	**0.400**
Cardio	**0.351**	0.185	0.033	**0.597**	0.206
Wine	**0.777**	0.079	**0.341**	0.253	0.000
Thyroid	0.055	**0.166**	0.026	0.047	**0.111**
Vertebral	**0.185**	0.111	0.000	0.000	**0.111**
Yeast	**0.358**	0.159	0.153	**0.441**	0.119
Seismic	0.131	**0.140**	0.107	**0.196**	0.103
Heart	**0.161**	0.000	**0.062**	0.000	0.000

As stated in [5], OutDST is not optimized to finding the ROC, and is primarily designed for labeling outliers, as opposed to scoring them. Therefore, in [5], the F1-score is used as a measure for demonstrating OutDST's effectiveness. We follow the same principle and compute the F1-score for all methods using the top 10% points. These F1-scores are shown in Table 3. Observe that LODES again shows significantly better performance against all the competitors, including OutDST. The F1-score of LODES is among the top-2 for 9 data sets, while HiCS is in the top-2 for 4 data sets, OutDST in 2 data sets, FastABOD in 5 data sets, and LOF in 4 data sets.

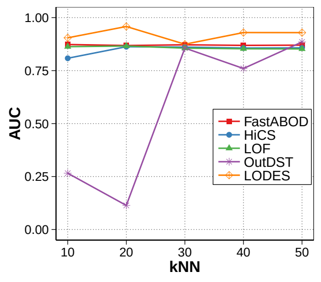
5.4 Analyzing Sensitivity to k. Finally, we analyze the parameter sensitivity of the parameter k, which regulates the number of nearest neighbors. Since this parameter is used by all the methods, its sensitivity can be studied in all methods. We plot the AUC against the value of k for the primary data sets in Figure 5(*a*) to *(d)*. It can be observed that all the methods, except OutDST, are largely insensitive to the value of k. This is of course a positive characteristic of both LODES and the baselines.



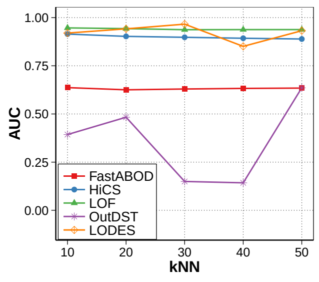
(a) **Glass**



(b) **Pendigits**



(c) **Ecoli**



(d) **Vowels**

Figure 5: Sensitivity of the outlier detection methods to the parameter k or the number of k-nearest neighbours.

OutDST demonstrates high sensitivity to k. On the other hand, even though LODES is an spectral outlier detection technique, like OutDST, its accuracy is not highly sensitive to k. An inappropriate value of k adds edges to the neighbourhood graph that are inconsistent with the local-density structure. Such inconsistent edges dramatically reduce the quality of the spectral embedding. The relative insensitivity of LODES to the parameter k can be attributed to its ability to handle inconsistent edges in the neighbourhood graph. The similarity update step of LODES adjusts the neighbourhood graph, such that the weights of these inconsistent edges are significantly reduced after each iteration. Thus, the iterative nature of LODES helps to mitigate the adverse effects exhibited by improper values of k, while other spectral techniques remain susceptible to it.

6 Conclusion.

Spectral methods are used widely in clustering because of their ability to discover clusters of varying shapes and sizes. Therefore, it is natural to explore whether such methods can also be used in the context of outlier analysis in order to discover outliers embedded in manifolds of arbitrary shape. However, spectral methods are often optimized to clustering and they do not work particularly well for outlier analysis. In this paper, we discussed an iterative approach for discovering a high-quality spectral embedding by combining local density-based methods with spectral methods. The embedding obtained from the proposed approach is tailored for outlier analysis. Our experimental results show significant performance improvements over several state-of-the-art outlier analysis methods.

References

[1] C. C. Aggarwal. Outlier Analysis, *Springer*, 2013.

[2] C. C. Aggarwal and P. S. Yu. Outlier detection for high-dimensional data, *ACM SIGMOD Conference*, 2001.

[3] S. Bay, and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *ACM KDD Conference*, 2003.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A Survey, *ACM Computing Surveys*, 2013.

[5] X. Dang, B. Misenkova, I. Assent, and R. Ng. Outlier detection with space transformation and spectral analysis. *SIAM Conference on Data Mining*, 2013.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. *SIGMOD*, 2000.

[7] F. Keller, E. Muller, K. Bohm. HiCS: High-Contrast Subspaces for Density-based Outlier Ranking, *ICDE Conference*, 2012.

[8] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. *VLDB Conference*, 1998.

[9] U. von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4), pp. 395–416, 2007.

[10] E. Muller, M. Schiffer, T. Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. *ICDE*, 2011.

[11] E. Muller, I. Assent, P. Iglesias, Y. Mulle, K. Bohm. Outlier Analysis via Subspace Analysis in Multiple Views of the Data, *ICDM Conference*, 2012.

[12] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering analysis and an algorithm. *NIPS*, pp. 849–856, 2001.

[13] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD*, 2000.

[14] J. Shi, and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*. 22(8), pp. 888–905, 2000.

[15] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based Outlier Detection in High-Dimensional Data, *KDD*, 2008.