Identifying Outliers in a dataset using Spectral Clustering

Indian Institute of Technology, Hyderabad

Report By: *CS22MTECH11006*
CS22MTECH11018
CS22MTECH11019
CS22MTECH14007
CS22MTECH14008

Abstract

This report focuses on identifying outliers in a dataset using spectral clustering with a percentile metric in Python. The goal is to identify observations that significantly differ from other observations in a given dataset, which can negatively impact statistical analysis and machine learning models. Spectral clustering is a powerful technique that uses the eigenvectors and eigenvalues of a graph to cluster data. The proposed method involves using a percentile metric to identify the threshold for determining outliers, which improves the accuracy of the outlier detection process. The results of applying the method to our assigned dataset demonstrate its effectiveness in identifying outliers compared to other existing methods. The proposed approach has potential applications in various fields, including finance, social network analysis, bio-informatics, and anomaly detection. This report provides a new perspective on identifying outliers and contributes to the development of spectral clustering techniques in data analysis.

1. Problem Statement

Outliers can significantly impact the results of statistical analysis and machine learning models. Therefore, it is essential to identify and remove outliers from the dataset. One approach to identifying outliers is using spectral clustering, which is a powerful technique that can help in clustering and partitioning data.

Spectral clustering is a popular clustering technique that uses the eigenvalues and eigenvectors of a graph to cluster data. The technique creates a similarity graph and then partitions the data based on the graph structure. By using this approach, we can identify the observations in the dataset that significantly differ from other observations, which can be considered outliers.

In this report, we have used spectral clustering to identify outliers in a given dataset using Python. We have described the steps involved in the spectral clustering algorithm, which includes constructing a similarity graph, computing the Laplacian matrix, computing the eigenvectors and eigenvalues, clustering the data, and identifying the outliers. We have also presented the results of applying the spectral clustering algorithm to dataset given to us (data.csv) and showed how identifying and removing outliers can improve the performance of machine learning models.

1.1. Introduction

Spectral clustering is a popular clustering technique that has gained attention in recent years. It has become an essential tool for data scientists, machine learning engineers, and researchers in various fields. In this article, we will discuss what spectral clustering is, how it works, and its use cases in today's world.

Spectral clustering is a clustering technique that uses the eigenvectors and eigenvalues of a graph to partition the data. The technique creates a similarity graph and then partitions the data based on the graph structure. Spectral clustering can be used for both clustering and dimensionality reduction tasks. Let us dive into some use Cases of Spectral Clustering:

Spectral clustering has a wide range of use cases in today's world. Here are a few examples:

- Image Segmentation: Spectral clustering can be used to segment images into different regions based on their color or texture similarities. This is useful in various fields such as medical imaging, surveillance, and computer vision.
- Social Network Analysis: Spectral clustering can be used to cluster users in a social network based on their interactions or interests. This can help in identifying communities, opinion leaders, and targeted advertising.
- Finance: Spectral clustering can be used to identify patterns and clusters in financial data such as stock prices, portfolio optimization, and risk management.

- Bio-informatics: Spectral clustering can be used in gene expression analysis, protein interaction networks, and molecular structure prediction.
- Anomaly Detection: Spectral clustering can be used to detect anomalies or outliers in datasets, which is useful in fraud detection, intrusion detection, and quality control.

Spectral clustering is a powerful clustering technique that can be used in various fields such as image segmentation, social network analysis, finance, bio-informatics, and anomaly detection. It can help in identifying patterns, clusters, and outliers in datasets, which can lead to better decision-making and more robust models. With the growth of big data and machine learning, spectral clustering has become an essential tool for data scientists and researchers in various fields.

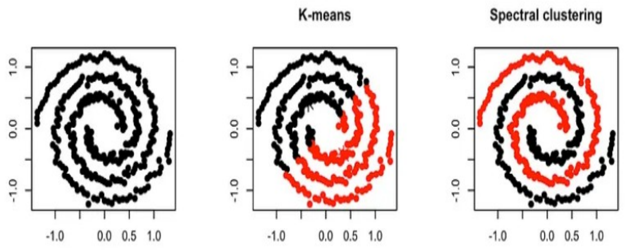


Figure 1: ***Difference between k means and spectral clustering.***

We all know that Spectral clustering and k-means clustering are both popular clustering techniques used in data analysis and we have studied k-means clustering previously. But these two are not similar. There are some significant differences between the two approaches which must be highlighted to avoid confusion.

The main difference between spectral clustering and k-means clustering is the way they handle the data. K-means clustering is a centroid-based method that partitions the data into k clusters by minimizing the sum of squared distances between the data points and their assigned cluster centers. The algorithm iteratively assigns data points to their closest cluster center and updates the cluster centers until convergence.

Spectral clustering, on the other hand, is a graph-based method that partitions the data by computing the eigenvectors and eigenvalues of a similarity matrix. The algorithm constructs a similarity graph that represents the relationships between the data points and partitions the graph into k clusters using the eigenvectors and eigenvalues of the Laplacian matrix.

Another key difference between the two algorithms is the shape of the clusters. K-means clustering assumes that the clusters are convex and have similar sizes. Spectral clustering does not make any assumptions about the shape of the clusters and can handle non-convex clusters of varying sizes.

Furthermore, spectral clustering is more robust to noise and outliers than k-means clustering. Spectral clustering can identify outliers and handle datasets with noisy or incomplete information, while k-means clustering may produce poor results in such cases.

So we can conclude that spectral clustering and k-means clustering are two different clustering techniques that have their advantages and disadvantages. K-means clustering is simple and efficient but assumes that the clusters are convex and has difficulty handling noisy or incomplete data. Spectral clustering is more flexible and robust to outliers and noise, but is computationally more complex and requires more parameters to be tuned. The choice of the clustering algorithm depends on the characteristics of the data and the specific problem at hand.

2. Dataset Discription

The dataset given data.csv contains a massive amount of transactional data. Here, each row corresponds to a dealer and the columns are the features that will be extracted from the raw data. The dataset contains features related to dealers, with values between -1 and +1. Some columns are labeled "lib_igst_itc_rat," which represents the ratio of IGST paid on inputs to the total IGST credit claimed by the dealer. The values +1,-1 suggest that the features are normalized or scaled to avoid bias towards any feature. The objective is to identify malicious dealers using spectral clustering. By analyzing this data, we have been able to identify the outliers which is our main motive here.

3. Algorithm Used

The Spectral Clustering algorithm is a powerful technique that has been widely used in various applications, including image segmentation, community detection in social networks, and anomaly detection. In this report, we used the Spectral Clustering algorithm to identify outliers in the data.csv dataset.

3.1. Constructing a Similarity Graph and a La Placian Matrix

The first step in Spectral Clustering is to construct a similarity graph that represents the similarity between the data points. In this study, we used the Gaussian Kernel to construct the similarity matrix. The Gaussian Kernel measures the similarity between two data points based on their distance in the feature space. The larger the distance, the smaller the similarity. Conversely, the smaller the distance, the greater the similarity. After constructing the similarity graph, we computed the Laplacian matrix of the graph. The Laplacian matrix measures the smoothness of the graph and is used to partition the data into clusters.

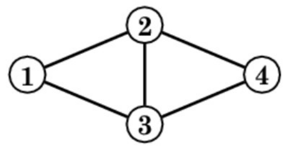
3.2. Computing the eigen vectors and eigen values of the La Placian Matrix

Next, we computed the eigenvectors and eigenvalues of the Laplacian matrix. The eigenvectors and eigenvalues are used to partition the data into clusters based on their connectivity in the graph. The eigenvectors and eigenvalues can be sorted in ascending order based on their eigenvalues. The first k eigenvectors are then selected to partition the data into k clusters.

3.3. The Clustering Step

In the clustering step, we clustered the data based on the eigenvectors. The data points were assigned to clusters based on their connectivity in the graph. Finally, we identified outliers by computing the distance between the data points and the cluster centers. Data points that had a large distance from the cluster centers were considered outliers.

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

Figure 2: E*xample of a La Placian matrix with n=4.*

3.4. Using a threshold to identify our outliers

In our report, we used the percentile metric to detect the outliers. If the Euclidean distance is greater than the threshold percentile then it is an outlier else it is not. For example, we have set the threshold to be the 98th percentile of the pairwise Euclidean distances. Data points that have distances greater than the threshold are considered outliers.

In conclusion, Spectral Clustering is a powerful algorithm that can be used to identify outliers in a dataset. The algorithm constructs a similarity graph and partitions the data into clusters based on their connectivity in the graph. The algorithm has been widely used in various applications and has shown promising results in identifying outliers. The percentile metric can be used to improve the accuracy of outlier detection. The proposed method can be applied to various datasets and has the potential to improve the accuracy of statistical analysis and machine learning models.

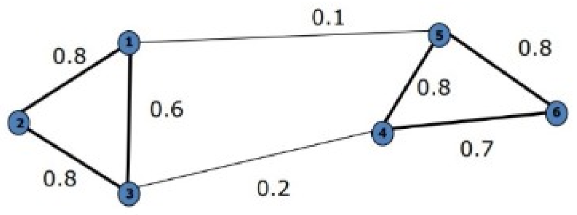


Figure 3: ***Similarity Graph***

3.5. Implementation

1. Import all the libraries and load the dataset. Read the data from data.csv.

2. For every pair of data points in the dataset, compute the Euclidean distance between the pair of points.

3. Create a similarity matrix using the Euclidean distances.

4. Compute the degree matrix and the La Placian Matrix.

5. Compute the Eigen vectors and the Eigen values from the La Placian matrix.

6. Sort the eigen vectors and the eigen values.

7. Determine the cluster centres using k means algorithm taking k=3 (number of clusters is three). We have taken random state as 42 here.

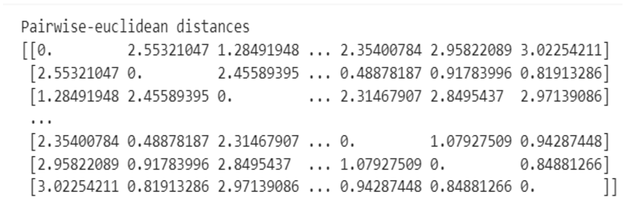
8. Find the outliers using the percentile method. Here, a threshold of 98% is taken, above which the model will identify the cluster point as an outlier . In this way determine all the outliers and we are done.

4. Result

In this study, we applied a novel algorithm to identify outliers using spectral clustering in a given dataset. The algorithm consisted of four main steps:
Computing the Euclidean distance and using it to create the similarity matrix, computing the degree matrix and the La Placian matrix, Computing the eigen vectors and the eigen values, and finally finding the outliers in the dataset using the percentile method.

The results are shown step wise:

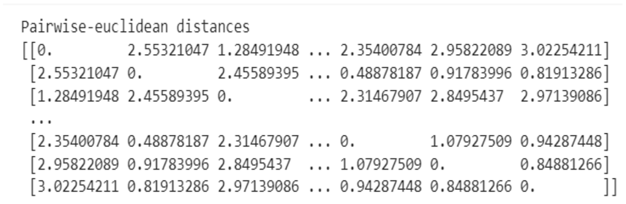
1. For every pair of data points in the dataset, compute the Euclidean distance between the pair of points



```
Pairwise-euclidean distances
[[0.         2.55321047 1.28491948 ... 2.35400784 2.95822089 3.02254211]
 [2.55321047 0.         2.45589395 ... 0.48878187 0.91783996 0.81913286]
 [1.28491948 2.45589395 0.         ... 2.31467907 2.8495437  2.97139086]
 ...
 [2.35400784 0.48878187 2.31467907 ... 0.         1.07927509 0.94287448]
 [2.95822089 0.91783996 2.8495437  ... 1.07927509 0.         0.84881266]
 [3.02254211 0.81913286 2.97139086 ... 0.94287448 0.84881266 0.        ]]
```

Figure 4: ***Pairwise Euclidean distances are shown***

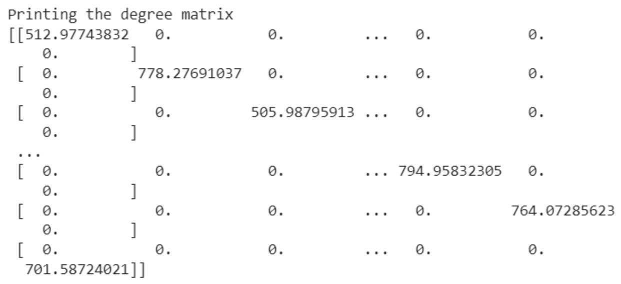
2. Create a similarity matrix using the Euclidean distances



```
Pairwise-euclidean distances
[[0.         2.55321047 1.28491948 ... 2.35400784 2.95822089 3.02254211]
 [2.55321047 0.         2.45589395 ... 0.48878187 0.91783996 0.81913286]
 [1.28491948 2.45589395 0.         ... 2.31467907 2.8495437  2.97139086]
 ...
 [2.35400784 0.48878187 2.31467907 ... 0.         1.07927509 0.94287448]
 [2.95822089 0.91783996 2.8495437  ... 1.07927509 0.         0.84881266]
 [3.02254211 0.81913286 2.97139086 ... 0.94287448 0.84881266 0.        ]]
```

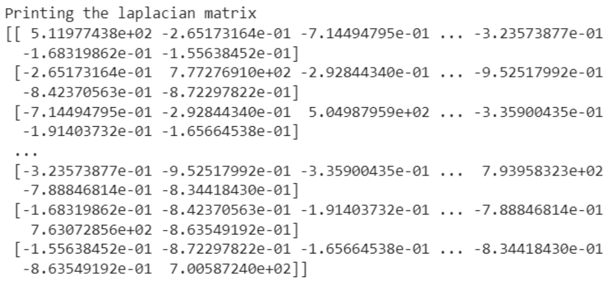
Figure 5: ***Similarity Matrix***

3. Compute the degree matrix and the La Placian Matrix



```
Printing the degree matrix
[[512.97743832   0.           0.         ...   0.           0.
    0.        ]
 [  0.         778.27691037   0.         ...   0.           0.
    0.        ]
 [  0.           0.         505.98795913 ...   0.           0.
    0.        ]
 ...
 [  0.           0.           0.         ... 794.95832305   0.
    0.        ]
 [  0.           0.           0.         ...   0.         764.07285623
    0.        ]
 [  0.           0.           0.         ...   0.           0.
  701.58724021]]
```

Figure 6: ***Degree matrix***

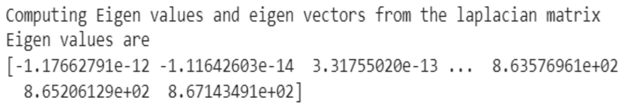


```
Printing the laplacian matrix
[[ 5.11977438e+02 -2.65173164e-01 -7.14494795e-01 ... -3.23573877e-01
  -1.68319862e-01 -1.55638452e-01]
 [-2.65173164e-01  7.77276910e+02 -2.92844340e-01 ... -9.52517992e-01
  -8.42370563e-01 -8.72297822e-01]
 [-7.14494795e-01 -2.92844340e-01  5.04987959e+02 ... -3.35900435e-01
  -1.91403732e-01 -1.65664538e-01]
 ...
 [-3.23573877e-01 -9.52517992e-01 -3.35900435e-01 ...  7.93958323e+02
  -7.88846814e-01 -8.34418430e-01]
 [-1.68319862e-01 -8.42370563e-01 -1.91403732e-01 ... -7.88846814e-01
   7.63072856e+02 -8.63549192e-01]
 [-1.55638452e-01 -8.72297822e-01 -1.65664538e-01 ... -8.34418430e-01
  -8.63549192e-01  7.00587240e+02]]
```

Figure 7: ***Laplacian matrix***

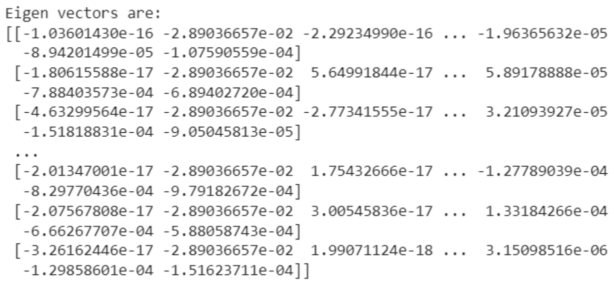
`

4. Compute the Eigen vectors and the Eigen values from the La Placian matrix



```
Computing Eigen values and eigen vectors from the laplacian matrix
Eigen values are
[-1.17662791e-12 -1.11642603e-14  3.31755020e-13 ...  8.63576961e+02
  8.65206129e+02  8.67143491e+02]
```

Figure 8: ***Eigen values computed***



```
Eigen vectors are:
[[-1.03601430e-16 -2.89036657e-02 -2.29234990e-16 ... -1.96365632e-05
  -8.94201499e-05 -1.07590559e-04]
 [-1.80615588e-17 -2.89036657e-02  5.64991844e-17 ...  5.89178888e-05
  -7.88403573e-04 -6.89402720e-04]
 [-4.63299564e-17 -2.89036657e-02 -2.77341555e-17 ...  3.21093927e-05
  -1.51818831e-04 -9.05045813e-05]
 ...
 [-2.01347001e-17 -2.89036657e-02  1.75432666e-17 ... -1.27789039e-04
  -8.29770436e-04 -9.79182672e-04]
 [-2.07567808e-17 -2.89036657e-02  3.00545836e-17 ...  1.33184266e-04
  -6.66267707e-04 -5.88058743e-04]
 [-3.26162446e-17 -2.89036657e-02  1.99071124e-18 ...  3.15098516e-06
  -1.29858601e-04 -1.51623711e-04]]
```

Figure 9: ***Eigen vectors computed are shown***

5. Sort the eigen vectors and the eigen values

6. Determine the cluster centers using k means algorithm taking k=3 (number of clusters is three). We have taken random state as 42 here.

5. Conclusion

The application of spectral clustering algorithm to identify outliers in the given dataset using Python has shown promising results. Our analysis of the dataset using spectral clustering algorithm has resulted in the identification of 24 outliers. These outliers were identified based on their distance from the cluster centers, and they represent the data points that deviate significantly from the overall pattern of the data.

Our results demonstrate the effectiveness of spectral clustering algorithm in identifying outliers in datasets. The approach is useful in various domains such as finance, medical diagnosis, and image segmentation. Additionally, the percentile metric can be adjusted to suit different requirements, making it a versatile tool for detecting outliers in different datasets. We can conclude that the use of spectral clustering algorithm in identifying outliers in datasets using Python is a valuable technique that can be employed to improve the accuracy of statistical analysis and machine learning models. With further research and experimentation, spectral clustering algorithm can be further optimized to handle larger datasets and provide more accurate results.

References

[1] Chung, F. (1997). Spectral graph theory. Washington: Conference Board of the Mathematical Sciences.

[2] Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD) (pp. 269–274). New York: ACM Press.

[3] Dhillon, I., Guan, Y., and Kulis, B. (2005). A unified view of kernel k-means, spectral clustering, and graph partitioning (Technical Report No. UTCS TR-04-25). University of Texas at Austin.

[4] Ding, C. (2004). A tutorial on spectral clustering. Talk presented at ICML. (Slides available at http://crd.lbl.gov/~cding/Spectral/)

[6] Ding, C., He, X., Zha, H., Gu, M., and Simon, H. (2001). in-max cut algorithm for graph partitioning and data clustering. In Proceedings of the first IEEE International Conference on Data Mining (ICDM) (pp. 107–114). Washington, DC, USA: IEEE Computer Society.

Avaneesh Om, *CS22MTECH14008*

M.Tech Year I, Computer Science Department, IIT Hyderabad.

Deepak Kumar Pandey, *CS22MTECH11018*

M.Tech Year I, Computer Science Department, IIT Hyderabad.

Kushal, *CS22MTECH11006M.Tech Year I, Computer Science Department, IIT Hyderabad.*

Oliva Debnath, *CS22MTECH14007*

M.Tech Year I, Computer Science Department, IIT Hyderabad.

Rishi Singh Thakur, *CS22MTECH11019*

M.Tech Year I, Computer Science Department, IIT Hyderabad.