Fall 2017

Final Project Report: Hubness and Some Questions

Yimeng Li and Vega Bharadwaj *CS 584 Data Mining*

1 Introduction

The number of k-occurences of point x from dataset D is called its hubness score. High hubness score indicates that this point is close to plenty of other points in the dataset. In a regular clustering algorithm, e.g. K-Means, a cluster medoid is also the point closest to all the points in the cluster. Since both of them, a cluster medoid and a high hubness point, share the same attribute, we believe it's highly possible that high hubness points are more likely to become cluster medoids comparing to points with a lower hubness score. In this project, we unveil the relationship between high hubness points and cluster medoids on both synthetic and real-world datasets.
Besides, low hubness scores also indicates that a point is far away from the majority of its peers, which is big sign of being an outlier or a noise. We happened to learn an algorithm called DBSCAN during the semester, which is quite effective in finding data outliers using density information. The second problem we are trying to solve is to explore the relation between low hubness points and DBSCAN-detected noise points on both synthetic and real-world datasets.

2 Related Work

Hubness has been observed in several fields of applications involving sound and image data during recent years (Aucouturie and Pachet, 2007; Doddington et al., 1998; Hicklin et al., 2005) and, in addition, Jebara et al. Hubness phenomenon in the construction of the neighborhood graph of learning (Tony Jebara et al 2009) [2]. Amina M et al. designed a hubness-based algorithm by introducing hubs into the k-means algorithm (Amina M et al 2015) [3]. Although people does not give much attention to the phenomenon of hubness in data clustering, the k-nearest-neighbor list is widely used in many clusters. The k-nearest-neighbor list computes the density estimate by the volume used to observe the space determined by the k nearest neighbors. Density-based clustering methods generally rely on this density estimation. The main goal of density-based clustering algorithms is to find high-density areas separated by low-density areas [4]. In high-dimensional space, this is often difficult to estimate because data is usually sparse. It is also important to choose the appropriate neighborhood size, because too small or too large k values may cause the density-based approach to fail. The k-nearest-neighbor list is often used to construct k-NN graphs and so it's used in graph clustering.

3 Solutions

To explore the relation between hubs and clusters, we did experiments on both synthetic datasets and real-world datasets. All the experiments are finished using MATLAB.

First part of the experiment is performed on the synthetic datasets. We run K-Means on each gaussian mixture. The K-Means algorithm is initialized by using K-Means++. In each iteration, we measure the distance from each cluster centroid to its closest neighbor and to the strongest hub inside the cluster, and scaled by the average intracluster distance. Both minimal and maximal distance among all the clusters are computed. We run the whole process several times and compute the average minimal and maximal distances to eliminate the noise brought in by K-Means random initialzation. In the end we compare the average minimal distance to a medoid and to a strong hubness point to see if they are the same point. And of course we also compare the maximal distances between them. Second part of the experiment is done on the real-world datasets. Before we running the real algorithms on these datasets, we first do some data preprocessing. For each point in a dataset, we normalize its attributes values across the whole dataset using minmax normalization. Then we go through the same computing process on each real dataset as we do on the gaussian mixtures.
Even though we implement all the experiments by ourselves, it's worth mentioning how we find the strongest hubness point in each cluster. We first call $knnsearch()$ function to find the nearest neighbors of each point in the cluster. Then we call $unique()$ function to pick out points that are the neighbors of other points. $histc()$ function is called to figure out the number of times each point being the nearest neighbor of other points. The point with the largest number is the strongest hubness point in the dataset. To compute the intraclass distance, we call $pdist2()$ function to compute the euclidean distance from each point in the cluster to the centroid. We sum up these distances and divide it by the number points in the cluster to get the intraclass distance. The cluster medoid is found by seeing the point which has the smallest distance to the cluster centroid.
To explore the relation between hubs and DBSCAN outliers, experiments are also finished using both synthetic datasets and real-world datasets.First part of the experiment is done on the synthetic datasets. We run DBSCAN on each gaussian mixture to find the outliers. The eps value, the radius of each core circle, is set up manually by reading the kdist graph of the whole dataset. Then we compute the hubness score of all the points in the dataset. For the DBSCAN detected noise points, we check if their hubness scores are two standard deviations lower than the average hubness value of all the points. Second part of the experiment is done on the real-world datasets. We still need to manually set up the eps value. Then we do the same comparison between the outliers hubness scores and the average hubness value.
How we find the strongest hubness point in the dataset is the same as what we do when we find the relationship between hubs and cluster medoids. The difference is that in the hubs and medoids experiment, we find the strongest hubness point of each cluster. The hubness point is computed by finding each point's nearest neighbors inside the cluster. But in the hubs and DBSCAN noise points experiment, each point's hubness score is decided by each point's nearest neighbors over the whole dataset, rather than just points in the same cluster. To compute the average and standard deviation of the hubness scores, we call MATLAB functions $mean()$ and $std()$ respectively.

4 Experiments

4.1 Data

Two types of data are including synthetic and real-world data sets.
For synthetic datasets, we create 7 random generated gaussian mixtures by using $gmdistribution()$ from MATLAB. All the gaussian mixtures are generated from a given list of dimensions: 2, 5,

10, 20, 30, 50, 100 and are created from 10 Gaussian generators. So each mixture has 10 clusters. When we create the mixtures, the gaussian distribution mean is uniformly chosen from $[lower\_bound^{\mu}, upper\_bound^{\mu}]^{dimension}$, $lower\_bound^{\mu} = -20$, $upper\_bound^{\mu} = 20$ and the standard deviations is also randomly chosen from $[lower\_bound^{\sigma}, upper\_bound^{\sigma}]^{dimension}$, $lower\_bound^{\sigma} = 2$, $upper\_bound^{\sigma} = 5$. And we randomly generate 10000 points for each mixture.
For the real world datasets, we take several datasets from UCI dataset repository. And they are described in an ascending order of their dimension.

- Iris Dataset. It is perhaps the best known database to be found in the pattern recognition literature. It has 150 instances and the data dimension is 4 and it has 3 classes. It can be downloaded from https://archive.ics.uci.edu/ml/datasets/iris.

- Abalone Dataset. The data comes from predicting the age of abalone from physical measurements. It has 4177 instances and the data dimension is 8 and it also has 8 classes. It can be downloaded from https://archive.ics.uci.edu/ml/datasets/abalone.

- Breast Cancer Wisconsin (Prognostic) Dataset. Each record represents follow-up data for one breast cancer case. It has 198 instances and the data dimension is 32 and it has 2 classes. It can be downloaded from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Prognostic).

- Sonar Dataset. The data is obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. It has 208 instances and the data dimension is 60 and it has 2 classes. It can be downloaded from http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks).

- Hill-Valley Data Set. Each record represents 100 points on a two-dimensional graph. When plotted in order as the Y co-ordinate, the points will create either a Hill or a Valley. It has 606 instances and the data dimension is 100 and it has 2 classes. It can be downloaded from http://archive.ics.uci.edu/ml/datasets/hill-valley.

4.2 Experiment Setup

To make sure that the implementation computing hubness scores is correct, we replicate the experiment settings mentioned in [1] to see if I get similar results.
To evaluate our experiments on hubs distribution across clusters, we will draw some plots to show how the maximal distance and minimal distance evolve through iterations. Data dimensions below 10 are used to illustrate low-dimensional behavior while dimensions above 10 are used to illustrate high-dimensional behavior. The neighborhood size is set to 10 across all the datasets.
To set up the DBSCAN *eps* parameter, we read the kdist graph, a graph recording the distance of each point to its 10th nearest neighbor. Table ?? shows the *eps* value of each dataset.

Synthetic Datasets	Eps Value	Real Datasets	Eps Value
2-dim gm	2.0	Iris	0.35
5-dim gm	4.5	Abalone	0.2
10-dim gm	7.2	WPBG	1.6
20-dim gm	11	Sonar	2.4
30-dim gm	13.5	Hill and Valley	0.9
50-dim gm	18	...	...
100-dim gm	26	...	...

Table 1: Datasets and DBSCAN *eps* Parameter

4.3 Experimental Results

5 Conclusion

6 Contribution

Yimeng Li finished exploring the relation between hign-hubness points and cluster modoids. He also finished exploring the relation between low-hubness points and DBSCAN-detected noise points.