High-Dimensional Data Clustering using Hubness Based Clustering Algorithms

Pradeepa S \(^1\) Dr. R. Thamilselvan \(^2\)

\(^1\) PG scholar
\(^2\) Department of Information Technology
Kongu Engineering College, India

Abstract—Clustering is an unsupervised process of grouping elements together, so that elements assigned to each cluster are more similar to each other than to the other data points. Clustering becomes difficult due to increasing sparsity of such data as well as the increasing difficulty in distinguishing distances between data points. Also, most data of interest today in data-mining applications is complex and is usually represented by many different features. Traditional approaches for clustering in low dimensional data can also be used for clustering high dimensional data by observing a lower dimensional feature subspace. But the performance of standard machine learning algorithms becomes degraded while handling high dimensional data. The number of data points are required to represent any distribution grows exponentially with number of dimensions which leads to bad density estimate for higher dimensional data. The difficulties in dealing with high dimensional data are considered to be an aspect of the curse of dimensionality. In this existing system the value of k or the range of k is neither directly nor indirectly specified by the users. The proposed method describes a novel perspective on the problem of clustering high-dimensional data and also for specifying k value by using Visual Access Tendency (VAT). Instead of attempting to avoid the curse of dimensionality by observing a lower dimensional feature subspace, the proposed method embrace the dimensionality by high-dimensional phenomena. Hubness, the number of times a data point appears among the k nearest neighbors of other data points in a data set, can be successfully exploited in clustering. Hubness score can be used as a good measure of point centrality within a high-dimensional data cluster by using hubness based clustering algorithms. In addition to that the proposed methodology uses VAT to find the number of clusters the high dimensional data with more accuracy in the automated manner. The experimental tests conducted were proves that the proposed methodology provides better result than the existing approaches in terms of more accuracy and improved time consumption. The quality of clusters is measured in terms of silhouette index and the results obtained would be promising.

Key words: Clustering, high-dimensional data, curse of dimensionality, nearest neighbors, hubness

I. INTRODUCTION

Data Mining is the process of mining knowledge from large amounts of data. Data mining is also called as knowledge mining. Data mining have great importance in competitive business environment. As the information technology grows, the database also grows. Data Mining is needed, in order to analyze and retrieve a high level of knowledge from the database. Nowadays, large quantities of data are being acquired. Data could be large in terms of size and dimensionality. The performance of data mining can be used for variety of information repositories. The applications of data mining include fraud detection, customer relationship management, banking, manufacturing and production, text mining, web mining, etc. Data mining functionalities are used to specify the kinds of patterns to be found in data mining tasks. Data mining tasks can be divided into two categories: descriptive and predictive. Descriptive mining task is to characterize the general properties of the data. Predictive mining task is to perform inference on the current data in order to make predictions. Data predictive is categorized into frequent Mining, association, correlation, classification, prediction, cluster analysis, evolution analysis and outlier analysis.

Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in the other clusters. A cluster of object can be treated collectively as one group and so considered as a form of data compression. In clustering, first partition the set of data into groups based on data similarity and then assigns labels to the relatively small number of groups. Advantages of a clustering-based process are that it is adaptable to changes and helps single out useful features that distinguish different groups. Clustering is the main task of explorative data mining. Clustering is the task of grouping a set of objects into groups called clusters, such that the objects in the same cluster are more similar to each other but the objects in different clusters are dissimilar. The objects are described by a set of dimensions (attributes). The resulting clusters should have high intra-class similarity and low inter-class similarity. Clustering techniques tries to group objects based on similarity (distance) measure. Clustering algorithms have been classified into four groups: hierarchical, partitional, density-based, grid-based and subspace algorithms. Algorithms from the subspace search for clusters in lower dimensional projection of the original data, and are preferred when dealing with high dimensional data\(^{[1]}\)[2][3]. This preference lies in the observation that having more dimensions usually leads cause curse of dimensionality, where the performance of many standard machine-learning algorithms becomes degraded. This is mostly due to two effects: the empty space phenomenon and concentration of distances. Mostly high-dimensional data sets tend to be sparse, because the number of data points required representing the distribution grows exponentially with the number of dimensions. This causes density estimates for high-dimensional data to be bad, causing difficulties for density-based approaches. The second one is a counterintuitive property of high-dimensional data representations, where all distances between data points tend to become harder to distinguish as dimensionality increases, which can cause problems with distance-based algorithms.
Hence these difficulties can be overcome by using hubness [6], the possibility of some data points to occur frequently in k-nearest neighbor lists of other data points in the data set.

II. RELATED WORK

T.N. Tran et al (8) discussed with problems for clustering High resolution and high dimension satellite images due to clusters of different sizes, shapes and densities. They proposed k-nearest neighbor (KNN) density-based rule for a high dimensional dataset and to develop a new KNN density-based clustering (KNNCLUSST) for complex dataset based on the rule. The KNN_rule makes a clear link between density-estimation techniques to unsupervised classification. The method can obtain clusters different in size, shape and density. KNNCLUSST is stable, clear and easy to understand and implement. The number of clusters is automatically determined using minimum user input. The complexity and performance degrades in this method.

Chris Ding et al (2) dealt with the concept to data clustering, requiring that for any data point in a cluster, its k-nearest neighbors and mutual nearest neighbors should also be in the same cluster. The properties of the cluster k-nearest neighbor consistency and define kNN and kMN consistency enforcing and improving algorithms. The kMN/kNN consistent K-means allows several connected components per cluster, which is thus somewhere between the connected component approach and usual K-means. Thus represents the unified framework for the proposed algorithm. The proposed the cluster kNN /kMN consistency is considered as a unsupervised quality measure of data clustering. When the clustering accuracy improve which enhances the kNN /kMN consistency.

Liping Jing et al (3) focus on data sparsity problem in clustering high-dimensional data. They propose EWKM algorithm, a new k-means type subspace clustering algorithm for high-dimensional sparse data. The algorithm generates better clustering results than other subspace clustering algorithms in both synthetic and real world data. The new algorithm is also scalable to large data sets but lacks in clustering accuracy.

David Arthur et al (1) discussed with the problem of speed and accuracy of k-means algorithm. They proposed algorithm that rectifies the problem by augmenting k-means with a simple, randomized seeding technique, obtain an new algorithm that is O(log k)-competitive with the optimal clustering.

Milos Radovanovic et al (5) dealt with the problem of curse of dimensionality by exploring a new aspect of the dimensionality curse, referred to as hubness, that affects the distribution of k-occurrences: the number of times a point appears among the k nearest neighbors of other points in a data set. This phenomenon, showing that it is an inherent property of data distributions in high-dimensional vector space, discuss its interaction with dimensionality reduction, and explore its influence on a wide range of machine-learning tasks directly or indirectly based on measuring distances, belonging to supervised, semi-supervised, and unsupervised learning families. The accuracy is outperformed in several classification and clustering algorithms but it doesn’t have any concern about efficiency and quality.

Nenad Tomasev et al (6) focus on the problem of Object recognition from images in automatic image processing. They proposed nearest neighbor method using Hubness solve the problem of Object recognition. In Hubness, some influential points called hubs play important role in the nearest neighbor classification provides better classification accuracy and recognition accuracy. Algorithms such as HW-KNN, H-FNN and HIKNN often showed significant improvements in classification accuracy when compared to the basic kNN algorithms. This algorithm also produce higher robustness in cases of bad hubness. This suggests that incorporating the information about k-occurrences is potentially quite useful, so hubness should not be disregarded when building nearest-neighbor-based image classification algorithms.

Nenad Tomasev et al (4) dealt with the problem of handling curse of dimensionality in high dimensional data. The phenomenon, related to nearest-neighbor learning methods, is known as hubness and refers to the emergence of very influential nodes (hubs) in k-nearest neighbor graphs. Based on this fuzzy measures are proposed for k-nearest neighbor classification, all based on hubness, which express fuzziness of elements appearing in k-neighborhoods of other points. The fuzzy k-nearest neighbor classification offers better confidence measures for label assignments, which is a highly desirable property. The accuracy improved on an average.

It is clear that clustering high dimensional data lacks in accuracy, quality, efficiency and effectiveness. Hubness naturally occurs in high dimensional data can effectively used with k nearest neighbor to provide better cluster quality.

III. SYSTEM IMPLEMENTATION

Hubness is an aspect of the curse of dimensionality pertaining to nearest neighbors, unlike the much discussed distance concentration phenomenon. Let \( D \subseteq \mathbb{R}^d \) be a set of data points and let \( N_k(x) \) denote the number of k-occurrences of data point \( x \in D \), i.e., the number of times \( x \) occurs in k-nearest neighbor lists of other data points from \( D \). When the dimensionality of data increases, the distribution of k-occurrences becomes considerably skewed. As a consequence, some data points refer to as hubs, are included in many more k-nearest-neighbor lists than other data points. Here after, the number of k-occurrences of point \( x \in D \) as its hubness score. Hubness, as a phenomenon, appears in high-dimensional data as an inherent property of high dimensionality, and is neither artifact of finite samples nor a peculiarity of some specific data sets. Naturally, the exact degree of hubness may still vary and is not uniquely determined by dimensionality.

A. K-Nearest Neighbor Classifiers

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (distance functions). A case is classified by a majority frequency of its neighbors, with that case is assigned to the class most common amongst its K nearest neighbors measured by a distance function. If the value of \( K=1 \), then in that case is simply assigned to the class of its nearest neighbor. Consider an example with data points
\( x_1=2, \) \( x_2=3, \) and \( x_3=4, \) \( x_4=5. \) Using the equation (4.1) the Euclidean distance between two points is computed as 2.82.

1) Distance Functions

\[ \text{Euclidean} \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \]  
(3.1)

\[ \text{Manhattan} \sum_{i=1}^{k} |x_i - y_i| \]  
(3.2)

\[ \text{Minkowski} \left( \sum_{i=1}^{k} |x_i - y_i|^q \right)^{\frac{1}{q}} \]  
(3.3)

It should be noted that all three distance measures are valid only for continuous variables. For categorical variable the hamming distance must be used. Before finding K-nearest Neighbor classifier the values of each attribute should be normalized. This helps prevent attributes with initially large ranges from outweighing attribute with smaller ranges. Min-max normalization can be used to transform a value \( v \) of a numeric attribute \( A \) to \( v' \) in the range \([0, 1] \) by computing

\[ v' = \frac{v - \text{min}_A}{\text{max}_A - \text{min}_A} \]  
(3.4)

where \( \text{min}_A \) is minimum values of attribute \( A \) and \( \text{max}_A \) is the maximum values of attribute \( A \).

B. Hub-Based Clustering

Hubness is as a kind of local centrality measure, it may be possible to use hubness for clustering in various ways. To test this hypothesis, opt for an approach that allows observations about the quality of resulting clustering design to be related directly to the effect of hubness. It is expected of hubs are located near the centers of sub clusters in high-dimensional data. natural way to test this hypothesis by approximating these centers is to compare the hub-based approach with some centroid-based technique. For this reason, the algorithm is made to resemble K-means, by being iterative approaches for defining clusters around separated high-hubness data elements. Fig 3 shows that not only might taking hubs as centers in following iterations provide quicker convergence, but that it also might prove helpful in finding the best end configuration. The example shows the red dashed circle marks the centroid (C), yellow dotted circle the medoid (M), and green circles denote two elements of highest hubness (H1,H2), for neighborhood size 3.

![Hub example](Image)

**Fig. 1: Hub example**

Traditional methods such as partitional uses the centroid value which always depend on all current cluster elements, while hubs value always depend on their neighboring elements and, therefore, they must contain localized centrality information. Consider two types of hubness namely global hubness and local hubness. Local hubness is a phenomenon that is defined as a restriction of global hubness on any given cluster. Thus, the local hubness occurrence value represents the number of k-occurrences of a point in k-NN lists of elements within the same cluster.

The fact that hubs emerge close to centers of dense sub regions might suggest some sort of a relationship between hubness and the density estimate at the observed data point. There are some important differences. Generally at first hubness does not depend on scale. Let D1 and D2 be two unique sets of points. Let \( D_k(x) \) be the set of points, where \( x \) is among the k-nearest neighbors. Hence, the hubness score of \( x \) is given by \( N_k(x) = | D_k(x) | \). For each \( x_i \in D_k(x) \), whether point \( x \) is among the k-nearest neighbors of \( x_i \) depends on two things: distance \( (x,x_i) \), and the density estimate of data point \( x_i \) but not on the density estimate at point \( x \). Consequently, a hub might be a k-neighbor for points where density is high and low points. Therefore, there is no direct correspondence between the magnitude of hubness and point density. Also, needs the state representation for representing the exact volume of the neighborhood around a given point. For finding hubness distance matrix is alone enough.

Computational complexity of hubness-based algorithms is determination hubness scores. Several fast approximate approaches are available. It is possible to construct an approximate k-NN graph (from which hubness scores can be read) in \( \Theta(n \log n) \) time, where the user-defined value \( t > 1 \) expresses the desired quality of graph construction.

1) K-Hubs Clustering

To employ hubs for clustering is to use the value of it as Centroids. Instead of finding mean for subsequent iterations hubness value can be opted. The algorithm convergences as like in K-Means method until there are no re-assignments. The algorithm is referred to as K-Hubs

```
(1) Algorithm: K-Hubs
    - Input: dataset(DS)
    - Output: Set of Clusters

initializeClusterCenters();
Cluster [] clusters = form Clusters();
repeat
    for all Cluster c \in clusters do
        Data Point h =findClusterHub(c);
        SetClusterCenter(c, h);
    end for
    clusters = formClusters();
    until noReAssignments
    return clusters

(2) Procedure for finding hubness score
Hubscore(Data Set)
Input: dataset(DS)
Output:Hub-score
Initialize i to 1
For each record r1 in DS
    For each record r2 in DS
        Rec-Dis=find Euclidean(r1,r2);
    endfor
    Sorted_dist=sort Dist(Rec-Dis);
    KNN-rec-list[i]=compute KNNlist(Sorted_dist);
    Increment i by 1
endfor
//calculate hub in DS
Initialize i to 1
For each Hub in DS
    For each kNN in KNN-rec-list[i]
        if(Hub exist in KNN)
            Hubscore[hub]++;
        Increment i by 1
```

All rights reserved by www.ijserd.com 1524
endfor  
return Hubscore;  

2) Hubness Proportional Clustering  
Even though points with highest hubness scores are prime candidates for cluster centers, there is no need to disregard the information about hubness scores of other points in the data. A squared hubness-proportional stochastic scheme based on the widely used simulated annealing approach to optimization is implemented in this algorithm Hubness Proportional Clustering (HPC). The temperature factor was introduced to the algorithm, so that it may start as being entirely probabilistic and eventually end by executing deterministic K-hubs iterations. A trivial temperature schedule in the getProbFromSchedule(t) function have to be chosen to use. The number of probabilistic iterations \( N_{prob} \) is passed as an argument to the algorithm and the probability \( \theta = \min(1, \frac{1}{N_{prob}}) \). Different probabilistic schemes are possible and might even lead to better results.  

(2) Algorithm: HPC  
- Input: dataset (DS)  
- Output: Set of Clusters  
InitializeClusterCenters();  
Cluster[] clusters = formClusters();  
float \( t = t_0 \); initialize temperature  
repeat  
float \( \theta = \text{getProbFromSchedule}(t) \);  
for all Cluster \( c \) do clusters  
if randomFloat(0, 1) < \( \theta \) then  
DataPoint \( h = \text{findClusterHub}(c) \);  
setClusterCenter(c, h);  
else  
for all DataPoint \( x \) do  
setChoosingProbability(\( x \), \( N_k^2(\mathbf{x}) \));  
end for  
normalizeProbabilities();  
DataPoint \( h = \text{chooseHubProbabilistically}(c) \);  
setClusterCenter(c, h);  
end if  
end for  
clusters = formClusters();  
\( t = \text{updateTemperature}(t) \);  
until noReassignments  
return clusters  

3) Hubness Proportional K-means  
This algorithm hubness-proportional K-means (HPKM) use point hubness scores to guide the search, but choose a centroid-based cluster configuration in the end. It is nearly identical to HPC, the only difference being in the deterministic phase of the iteration, as the configuration cools down during the annealing procedure: instead of reverting to K-hubs, the deterministic phase executes K-means updates.  

(3) Algorithm: HPKM  
- Input: dataset (DS)  
- Output: Set of Clusters  
InitializeClusterCenters();  
Cluster[] clusters = formClusters();  
float \( t = t_0 \); initialize temperature  
repeat  
float \( \theta = \text{getProbFromSchedule}(t) \);  
for all Cluster \( c \) do clusters  
if randomFloat(0, 1) < \( \theta \) then  
DataPoint \( h = \text{findClusterCentroid}(c) \);  
setClusterCenter(c, h);  
else  
for all DataPoint \( x \) do  
setChoosingProbability(\( x \), \( N_k^2(\mathbf{x}) \));  
end for  
normalizeProbabilities();  
DataPoint \( h = \text{chooseHubProbabilistically}(c) \);  
setClusterCenter(c, h);  
end if  
end for  
clusters = formClusters();  
\( t = \text{updateTemperature}(t) \);  
until noReassignments  
return clusters  

4) Determination Of Number Of Clusters Using Vat  
Visual Access Tendency (VAT) which is used to detect the information of number of data clusters (or classes) in visual form. The VAT has been introduced by author Bezdek. The visual pattern apparent will presents with the more clarity of visual results for data classes (or clusters). A very beginner of data clustering can makes use of this tool for accessing of data clusters from the organized data. Visualization tool helps for achieving the best clustering results  

![Fig 2: Processing steps of VAT](image-url)  

The organization of data consists of a set of data objects which is further clustered based on the similarity features for every two-element subset data objects. Initially, the data is organized as data matrix \( (n \times m) \) form which can be shown as follows with \( n \) number of objects and \( m \) number of properties. These \( n \) data objects are compared using the distance measures such as Euclidean, Manhattan, and Minkowski etc for obtaining the dissimilarity matrix \( R \). The following matrix is represents the dissimilarity matrix \( R \) (with size \( n \times n \)).  
The dissimilarity matrix \( R \) is the input of VAT, and the VAT tool outputs the VAT Image with finite number of visual squared shaped dark blocks. Thus, we detect the number of clusters by counting of square shaped dark blocks along the diagonal of VAT Image.  
The VAT uses the logic of Prim’s algorithm for the major purpose of changing the current indices of the data objects. In this way, the indices of data objects are reordered by the VAT tool. Reordering the indices of data objects would show the number of clusters by squared shaped dark blocks along the diagonal in the VAT Image. The main aspect of VAT tool is to display the hidden clustering.
structure for a set of data objects. The logical steps of VAT tool are described as follows:

1. Choose the longest edge weight $e(v_i,v_j)$ from the dissimilarity matrix $R$.
   - Index=$1$; $P(index)=v_i$; Set $I = \{ \}$; $J = \{1,2,...,n\}$; $I = I \cup \{j\}$ and $J = J – \{j\}$.
2. Use Prim’s Logic.
   - For the interactions $t = 2,\ldots, n$.
   - Select $(i,j) \in \arg \min_{(i,j) \in E} \{R_{ij}\} \ ; P(t) = j$.
   - $I = I \cup \{j\}$ and $J = J – \{j\}$.
3. Reordered dissimilarity matrix $[RR_{ij}]=[RP(i),P(j)]$.
4. Display the VAT Image of $R$.

Tracing steps for VAT Tool Algorithm for the Input $R$ (Dissimilarity Matrix). Finally, the VAT image is displayed in MATLAB by executing the command of imshow (RR). Hence, the number of clusters information from this visualized image can be extracted. The number of clusters is given as input to the above three algorithms and compare the results before and after.

After obtaining the number of clusters from VAT tool for every dataset the value is fed into the algorithm 3 and 4 instead of choosing the input value as a random one. The performance is compared based on the algorithms with VAT input and algorithm without VAT input.

IV. RESULT ANALYSIS

A. K-Nearest Neighbor Classifier

Before finding K-nearest Neighbor classifier the values of each attribute should normalized. This helps prevent attributes with initially large ranges from outweighing attribute with smaller ranges. Min-max normalization can be used to transform a value $v$ of a numeric attribute $A$ to $v’$ in the range $[0, 1]$ by using equation (4). Given object $x$, find the $k$ most similar objects to $x$. Then $x$ has $k$ nearest neighbors. Variety of distance or similarity measures can be used to identify and rank neighbors. For that this requires comparison between $x$ and all objects in the database. The similarity is defined by means of distance metric. Here the distance metric used is Euclidean distance. Euclidean distance between two points or tuples, where the two values are $X_1=(x_{11},x_{12},...,x_{1n})$ & $X_2=(x_{21},x_{22},...,x_{2n})$.

$$dist(X_1,X_2)=\sqrt{\sum_{i=1}^{n}(x_{1i}-x_{2i})^2} \quad (5)$$

Fig. 3: Quality of GKH clustering on neighborhood size ($k$)

The GKH algorithm with wine dataset to the initialization of dimensionality $d=13$, $n=178$ instances, and $K=3$ clusters in the data. Neighborhood size is varied to different values starting from 5 to 20 Thus the results are displayed in figure 3. The silhouette index was used to estimate the clustering quality. The algorithm provides better quality when the neighborhood value is set to 10 and 20.

B. K-Hubs Algorithm

Fig. 4: Quality of GKH with varying number of clusters ($K$)

For selecting proper neighborhood size we ran the series of tests which gives better results when setting the value of $k$ as 10 and 20. In our experiment, neighborhood size value is set to be 10. To determine number of clusters the series of experiments are conducted with wine data set of dimensionality $d=13$, $n=178$ instances is shown in fig 4. The cluster size ($K$) is set be 3 and 5. The clustering quality is measured using silhouette index which produce good results when the number of clusters $K=3$.

After analyze and compare the performance offered by Global K-Hubs (GKH), Local K-Hubs (LKH) clustering system. The performance is evaluated by the parameters such as silhouette index. Based on the comparison and the results from the experiment show the Global K-Hubs (GKH) clustering approach works better than the LKH clustering.

Fig. 5: Comparison of LKH and GKH algorithm against cluster quality

In the above figure 5 shows the graph for wine dataset in that we are comparing the clustering quality in terms of silhouette index of the Global K-Hubs (GKH) clustering with Local K-Hubs (LKH) clustering. In this graph, x axis will be the number of data and y axis will be silhouette index rate. From the graph we can easily understand that the Global K-Hubs (GKH) clustering has higher silhouette index rate (higher clustering quality) which is taken the output result. So the GKH clustering algorithm is well effective compared to the LKH clustering approach in silhouette index rate.

C. HPC and HPKM algorithm

The proposed algorithm HPC and HPKM can be classified into two classification which is explained in section 4. Further based on the input to the algorithm mentioned two algorithms can be classified into the following: LHPC with VAT input, LHPC without VAT input, GHPC with VAT input, GHPC without VAT input, LHPKM with VAT input, LHPKM without VAT input, GHPKM with VAT input, GHPKM without VAT input. Based on the performance of algorithms in various measures such as accuracy, error rate, time complexity, Davies bouldin index, Beta index,
silhouette index are discussed and compared for several datasets.

The performance of good clustering algorithm is judged based on the low error rate production. By comparing GHPC with VAT, GHPC without VAT, LHPC with VAT, LHPC without VAT clustering system in terms of error rate the GHPC & GHPKM with VAT input and LHPC & LHPKM with VAT input produces low error. The efficiency and effectiveness of a clustering algorithm depends on the less running time. Based on that criteria here the algorithm is tabulated. From the analysis the conclusion can be made

<table>
<thead>
<tr>
<th>Datasets</th>
<th>GHPC Without VAT input</th>
<th>GHPC With VAT input</th>
<th>GHPKM Without VAT input</th>
<th>GHPKM With VAT input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>23</td>
<td>18</td>
<td>27</td>
<td>17</td>
</tr>
<tr>
<td>Iris</td>
<td>27</td>
<td>18</td>
<td>27</td>
<td>19</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>24</td>
<td>20</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>Wine</td>
<td>26</td>
<td>17</td>
<td>28</td>
<td>14</td>
</tr>
</tbody>
</table>

From the Table 1 the LHPC and GHPC with VAT produces effective result in few seconds than the other algorithms.

Silhouette index is an unsupervised measure of configuration validity, and average cluster entropy as a supervised measure of clustering homogeneity.

Fig. 6: Silhouette index Comparison of LHPC and GHPC

In the above Figure 6 of abalone dataset, based the comparison the LHPC and GHPC in terms of silhouette index for cluster quality. In this graph, x axis will be the number of data and y axis will be silhouette index rate The GHPC produces better results compared to other one.

Fig. 7: Silhouette index Comparison of LHPC and GHPC with VAT

In the above figure 7 shows the graph for abalone dataset in that we are comparing the clustering quality in terms of silhouette index of the LHPC with GHPC with VAT input. In this graph, x axis will be the number of data and y axis will be silhouette index rate. From the graph we can easily understand that the GHPC with VAT input has higher silhouette index rate (higher clustering quality).

V. CONCLUSION AND FUTURE WORK

Hubness for data clustering has been estimated using hubs, which is used to find approximately local data centers, but also frequently leads to improvement over the centroid-based approach. In existing only highest hubness score is used for cluster configuration that might degrades the performance. To avoid that problem in existing system in proposed methodologies both highest and lowest hubness score are used for cluster configuration. Based on the comparison and the results from the experiment show the Global Hubness proportional Kmeans (GHPKM) with VAT input and Global Hubness Proportional clustering (GHPC) with VAT input methods works better than the other methods of clustering. The results of above mentioned algorithms are compared with various performance measures such as accuracy, error rate, time complexity, Davies bound index, beta index. In that the GHPKM with VAT and GHPC with VAT out performs than other techniques.

The proposed algorithms represent only few possible approach to using hubness for improving high-dimensional data clustering. In future hubness can also intend to explore other closely related research directions.
including kernel mappings and shared-neighbor clustering. This would be a change to overcome the major drawback of the proposed methods—detecting only hyper spherical clusters, just as K-Means. Additionally, the research directions can explore methods for using hubs to automatically determine the number of clusters in the data.

VI. ACKNOWLEDGEMENTS

First of all I would like to extend my sincere gratitude for my supervisor Dr.R.Thamilselvan for providing me the opportunities of taking part in Master of Computer and Communication Program and his ideas and suggestions, which have been very helpful in the project. I am so deeply grateful for his help professionalism and valuable guidance throughout this project.

REFERENCES


