Clustering Algorithms for Data Stream

Karisha Nadhe[1], Prof. P. M. Chawan[2]

[1]*Student, Dept of CS & IT, VJTI Mumbai, Maharashtra, India*

[2]*Professor, Dept of CS & IT, VJTI Mumbai, Maharashtra, India*

Abstract:

Nowadays streaming data is delivered by more and more applications, due to this crucial method for data and knowledge engineering is considered to be clustering data streams. It is a two step process. A normal approach is to summarize the data stream in real-time with an online process into so called micro-clusters. Local density estimates are represented by micro-clusters by assembling the information of many data points which is defined in an area. A traditional clustering algorithm is used in a second offline step, in which larger final clusters are formed by reclustering the micro-clusters. For reclustering, the pseudo points which are used are actually coordinator of the micro-clusters with the weights which are density estimates. However, in the online process, information about density in the area between micro-clusters is not preserved and reclustering is based on possibly inaccurate assumptions about the distribution of data within and between micro-clusters (e.g., uniform or Gaussian). In this paper we depicted various algorithms that are used for clustering and their workings in brief.

Keywords: Data mining, data stream clustering, density-based clustering, micro-cluster, reclustering.

1. Introduction

Data mining is basically used to extract useful information from large sets of data. Clustering is the commonly utilized data mining strategy. In this process, the classification of objects is done into different groups by partitioning sets of data into a series of subsets or clusters. Detecting possible intrusions and also tracking the network data utilized to identify changes in traffic patterns is the illustration of clustering. An additional requirement for this case is, as data is produced it must be processed. The sequence of data points is ordered and unbounded in data stream[1],[2]. For applications like GPS data from smart phones, web click-stream data, computer network monitoring data, telecommunication connection data, readings from sensor nets, stock quotes, etc. such data streams are generated. The reclustering approaches join micro-clusters which are separated by a small area of low density and also are close together. These approaches completely ignore the data density in the area between the micro-clusters. To solve this problem, Tu and Chen introduced an extension to the grid-based D-Stream algorithm based on the concept of attraction between adjacent grids cells and showed its effectiveness.

And to solve this problem for micro- cluster-based algorithms we have discussed an algorithm. Here the concept of a shared density graph has been introduced which explicitly captures the density of the original data between micro-clusters during clustering and then it shows how the graph can be used for reclustering micro-clusters. The density in the shared region between micro-clusters is estimated by this algorithm directly from the data.

2. Related Work

One of the well-researched areas is Density-based clustering and here we give a brief overview. We can see many prototypical density-based clustering approaches like DBSCAN and several of its improvements[3]. In the first step the density around each data point is estimated by DBSCAN by counting the number of points in a user specified eps-neighbourhood and identifies core, border and noise points by applying user-specified thresholds. In a second step, core points that are density-reachable are joined into a cluster. Finally, clusters are assigned by border points. Other approaches are based on kernel density estimation e.g., DENCLUE or use shared nearest neighbours e.g., SNN, CHAMELEON.

2.1 DBSCAN

To discover the clusters and the noise in a spatial database, the algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) is designed. To apply this algorithm the appropriate parameters Eps and MinPts of each cluster and at least one point from the respective cluster should be known[4]. Then, using the correct parameters we could retrieve all points that are density-reachable from the given point. To get this information in advance for all clusters of the database there is no easy way. However, to determine the parameters Eps and MinPts of the "thinnest", i.e. least dense, cluster in the database there is a simple and effective heuristic. For this reason, DBSCAN uses the same values for Eps and MinPts for all clusters which are called the global values. For these global parameter values, the density parameters of the “thinnest” cluster are considered to be good candidates specifying the lowest density which is not considered to be noise.

DBSCAN starts with an arbitrary point p and retrieves all points that are density-reachable from p wrt. Eps and MinPts. This is done to find a cluster. The result depends on p. This procedure yields a cluster wrt. Eps and MinPts if p is a core point. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database. Since we use global values for Eps and MinPts, if two clusters of different density are “close” to each other then DBSCAN may merge two clusters into one cluster. Let the distance between two sets of points S1 and S2 be defined as dist (S1, S2) = min $\{dist(p,q) \mid p\in S1, q\in S2\}$. Then, only if the distance between the two sets is larger than Eps two sets of points having at least the density of the thinnest cluster will be separated from each other. Consequently, for the detected clusters with a higher value for MinPts a recursive call of DBSCAN may be necessary.

In spatial databases for the task of class identification clustering algorithms are attractive. However, when applied to large spatial databases these algorithms suffer from severe drawbacks.

2.2 DENCLUE

The DENCLUE algorithm employs a cluster model based on kernel density estimation. A cluster is defined by the local maximum of the estimated density function. Data points going to the same local maximum are put into the same cluster.

The DENCLUE algorithm works in two steps. In the first step map of the relevant portion of the data space is constructed which is called a pre-clustering step. The map is used to speed up the calculation of the density function which requires efficiently accessing of neighbouring portions of the data space. The actual clustering step is the second step, in which the algorithm identifies the density-attractors and the corresponding density-attracted points.

New approach was introduced to clustering in large multimedia databases. They had formally introduced the notion of influence and density functions as well as different notions of clusters which were both based on determining the density-attractors of the density function. For an efficient implementation of their approach, they used a map-oriented representation of the data they could efficiently determine a local density function which they introduced[5]. They had shown the generality of their approach, i.e. that they were able to simulate a locality-based, partition-based, and hierarchical clustering.

2.3 SNN

For low to medium dimensional data, density based algorithms such as DBSCAN, CLIQUE, MAFIA and DENCLUE have shown to find clusters of different sizes and shapes, although not of different densities[6]. However, in high dimensions, the notion of density is perhaps even more troublesome than that of distance. In particular, the traditional Euclidean notion of density, the number of points per unit volume, becomes meaningless in high dimensions.

An alternative to a direct similarity is to define the similarity between a pair of points in terms of their shared nearest neighbours. That is, the common or shared near neighbours confirms the similarity between two points. If point A is close to point B and if they are both close to a set of points C then we can say that A and B are close with greater confidence since their similarity is “confirmed” by the points in set C. This idea of shared nearest neighbour was first introduced by Jarvis and Patrick. A similar idea was later presented in ROCK. In the Jarvis – Patrick scheme, a shared nearest neighbour graph is constructed from the proximity matrix as follows[6]. Let i and j be two points. The strength of the link between i and j is now defined as:

$$str(i, j) = \sum (k+1-m) * (k+1-n) \text{ , where } im = jn$$

In the equation above, k is the near neighbour list size, m and n are the positions of a shared near neighbour in i and j’s lists. At this point, all edges with weights less than a user specified threshold are removed and all the connected components in the resulting graph are our final clusters.

2.4 CHAMELEON

Chameleon is an agglomerative hierarchical clustering algorithm that overcomes the limitations of some clustering algorithms. The key feature of Chameleon algorithm, to identifying the most similar pair of clusters it accounts for both interconnectivity and closeness. Due to this it avoids the limitations that are experienced in other algorithms. Furthermore, to model the degree of interconnectivity and closeness between each pair of clusters a novel approach is used by Chameleon. The internal characteristics of the clusters are considered by the approach themselves. Thus, it can automatically adapt to the internal characteristics of the merged clusters and it does not depend on a static, user-supplied model. Chameleon operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items[7]. This sparse graph representation allows Chameleon to scale to large data sets and to successfully use data sets that are available only in similarity space and not in metric spaces. Data sets in a metric space have a fixed number of attributes for each data item, whereas data sets in a similarity space only provide similarities between data items. Chameleon finds the clusters in the data set by using a two-phase algorithm. During the first phase, Chameleon uses a graph-partitioning algorithm to cluster the data items into several relatively small sub-clusters. During the second phase, it uses an algorithm to find the genuine clusters by repeatedly combining these sub-clusters.

2.5 DBSTREAM

This algorithm is supposed to be the first micro-cluster-based online clustering component that explicitly captures the density between micro-clusters via a shared density graph. The density information in this graph is then exploited for reclustering based on actual density between adjacent micro-clusters. We discuss the space and time complexity of maintaining the shared density graph.

The basic idea of this work is that if we can capture not only the distance between two adjacent MCs but also the connectivity using the density of the original data in the area between the MCs, then the reclustering results may be improved. In the following we develop DBSTREAM which stands for density-based stream clustering.

2.5.1 Leader-Based Clustering

Leader-based clustering was introduced by Hartigan[8] as a conventional clustering algorithm. To apply the idea to data streams it is considered straight-forward. DBSTREAM represents each MC by a leader which is a data point defining the MC's center and the density in an area of a user-specified radius r i.e. threshold around the center. This is similar to DBSCAN's concept of counting the points is an eps-neighbourhood, however, here the density is not estimated for each point, but only for each MC which can easily be achieved for streaming data[3].

2.5.2 Competitive Learning

New leaders are chosen as points which cannot be assigned to an existing MC. The positions of these newly formed MCs are most likely not ideal for the clustering. To remedy this problem, we use a competitive learning strategy introduced in to move the MC centers towards each newly assigned point. To control the magnitude of the movement, we use a neighbourhood function h() similar to self-organizing maps. In our implementation we use the popular Gaussian neighbourhood function defined between two points, a and b, as

$$h(\mathbf{a},\mathbf{b}) = \exp\left(-||\mathbf{a}-\mathbf{b}||^2/(2\sigma^2)\right)$$

With σ = r/3 indicating that the used neighbourhood size is +-3 standard deviations[1].

2.5.3 Capturing Shared Density

Capturing shared density directly in the online component is a new concept.

The shared density between two MCs, i and j, is estimated by $\hat{\rho}_{ij} = \frac{s_{ij}}{A_{ij}}$, where sij is the shared weight and Aij is the size of the overlapping area between the MCs[3].

A shared density graph Gsd(V,E) is an undirected weighted graph, where the set of vertices is the set of all MCs, i.e., V(Gsd)=MC, and the set of edges E(Gsd)={(vi,vj) | vi,vj $\in$ V(Gsd) $\wedge$ $\hat{\rho}_{ij}$ > 0} represents all the pairs of MCs for which we have pairwise density estimates. Each edge is labeled with the pairwise density estimate $\hat{\rho}_{ij}$.

In a typical clustering most MCs will not share density with each other. A very sparse shared density graph is the result of this condition. We can exploit this fact for more efficient storage and manipulation of the graph. All fixed-radius nearest-neighbours are needed to be found during clustering. Therefore, no additional increase in search time is incurred while obtaining shared weights.

3. Conclusion

In this paper, we have analyzed some of the important algorithms which are required for clustering data streams. We also discussed the first data stream clustering algorithm called DBSTREAM which explicitly records the density in the area shared by micro-clusters and uses this data for reclustering. Experiments also show that shared-density reclustering already performs extremely well when the online data stream clustering component is set to produce a small number of large MCs. To achieve comparable results other popular reclustering strategies can only slightly improve over the results of shared density reclustering and need significantly more MCs. Since it implies that we can tune the online component to produce less micro-clusters for shared-density reclustering this is an important advantage. This improves performance and, in many cases, the saved memory more than offset the memory requirement for the shared density graph.

REFERENCES

[1] C. Aggarwal, Data Streams:Models and Algorithms, (series Advances in Database Systems). NewYork, NY, USA: Springer-Verlag, 2007.

[2] J. Gama, Knowledge Discovery from Data Streams, 1st ed. London, U.K.: Chapman & Hall, 2010.

[3] M. Hahsler and M. Bolanos, "Clustering Data Streams Based on Shared Density between Micro-Clusters," IEEE transactions on knowledge and data engineering, vol. 28, no. 6, june 2016.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 1996, pp. 226–231.

[5] A. Hinneburg, E. Hinneburg, and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in Proc. 4th Int. Conf. Knowl. Discovery Data Mining, 1998, pp. 58–65.

[6] L. Ertoz, M. Steinbach, and V. Kumar, "A new shared nearest neighbour clustering algorithm and its applications," in Proc. Workshop Clustering High Dimensional Data Appl. 2nd SIAM Int. Conf. Data Mining, 2002, pp. 105–115.

[7] G. Karypis, E.-H. S. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," Computer, vol. 32, no. 8, pp. 68–75, Aug. 1999.

[8] J. A. Hartigan, "Clustering Algorithms", 99th ed. New York, NY, USA: Wiley, 1975.