

HIERARCHICAL CLUSTERING IN TEMPORAL DATABASES

Dr.G.Sreesharma
Professor, Pondicherry University, India

ABSTRACT

This project report discusses about the design and implementation of an incremental system for clustering the data stream that are present in time series databases. The Online Divisive Agglomerative Clustering system implemented in this project work continuously maintains a tree-like hierarchy of clusters that evolves with data, using a top-down strategy. Using correlation-based dissimilarity measure, each node is split by the farthest pair of streams. This system uses a merge operator which re-aggregates a previously split node, in order to react to changes in the correlation structure between time series. The split and merge operators act in response to changes in the diameters of existing clusters. Expanding the structure in this way leads to a decrease in the diameters of the clusters. This system has been designed to process thousands of data streams that flow at high rate. The main advantage of this system is the reduction in update time and memory usage. This system has been implemented using Java language in Windows platform.

INTRODUCTION

In recent days, techniques for storing information are becoming important as the web spread out, with constant increase in communication capabilities, creating a global network interaction of both data and processes. The traditional setting for data analysis had turned gathering data into one of the most difficult tasks in data mining applications. In fact, not rarely the amount of data available from a given source (e.g. sensor networks) is so high that traditional batch systems, which are usually based on memory storage and multiple readings of the same data, simply cannot be used. In recent real-world applications, data flows continuously from a data stream at high speed, producing examples over time. Traditional models cannot adapt to the high speed arrival of new examples (Rodrigues et al 2008). This way, new algorithms have been developed that aim to process data in real-time. These algorithms should be capable of processing each example in constant time and memory, while they consistently supply a compact data description at each given moment. In this context, quicker responses are usually requested. Hence it is necessary to continuously maintain a decision model at any time, which should reflect the behavior of most recent data.

Time series data is perhaps the most common kind of data explored by data miners. Clustering is probably the most frequently used data mining algorithm, used in exploratory data analysis. Data clustering techniques that work in real-time discovery must allow the update of the clustering definition based only on the current model and on new examples.

As data set have grown in size and complexity, there has been a shift away from direct hands on data analysis toward indirect, automatic data analysis using more complex and sophisticated tools. Major improvements in computer technology have aided data collection. However, the captured data needs to be converted into information and knowledge to become useful. Data mining is the entire process of applying computer based methodology, including new techniques for knowledge discovery to data.

Data mining identifies trends within data that go beyond simple analysis. Through the use of sophisticated algorithms, non-statistician users have the opportunity to identify key attributes of business processes and target opportunities. However, abdicating control of this process from the statistician to the machine may result in false-positives or no useful results at all. Although data mining is a relatively new term, the technology is not. For many years, businesses have used powerful computers to sift through volumes of data such as supermarket scanner data to produce market research reports (although reporting is not always considered to be data mining). Continuous innovations in computer processing power, disk storage, and statistical software are dramatically increasing the accuracy and usefulness of data analysis.

Clustering Algorithm

The algorithm that has been used in this work is Online Divisive Agglomerative Clustering. In this system, the Online Divisive-Agglomerative Clustering that constructs a hierarchical tree-shaped structure of clusters using a top-down strategy. The clustering technique is used to grow the tree and controlling the growth and also taking care of its splitting and aggregation. These processes are done and the tree is constructed. The Clustering algorithm is controlled by the database manager and finally a hierarchical tree is obtained. Thus using these components the decision process is made effective using the algorithms and thus a hierarchical tree structure is constructed.

HIERARCHICAL APPROACH

In this system, the Online Divisive-Agglomerative Clustering which is an algorithm for incremental clustering of streaming time series that constructs a hierarchical tree-shaped structure of clusters using a top-down strategy. The leaves are the resulting clusters, with each leaf grouping a set of variables. The union of all leaves is the complete set of variables. The intersection of any two leaves is the empty set. The system encloses an incremental distance measure and executes procedures for expansion and aggregation of the tree-based structure, based on the diameters of the clusters. The main assumption of the system is that decisions taken over a sample of the most recent data are, in the limit and under certain conditions, equivalent to those taken over an infinite set of observations. Given this, the system continuously monitors existing clusters' diameters over time. The diameter of a cluster is the maximum distance between variables of that cluster. For each existing cluster, the system finds the two variables defining the diameter of that cluster.

At time t , if a given condition is met on this diameter, the system splits the cluster and assigns each of the chosen variables to one of the new clusters, becoming the pivot variable for that cluster. Afterwards, all remaining variables on the old cluster are assigned to the new cluster which has the closest pivot. New leaves start new statistics, assuming that only forthcoming information will be useful to decide whether or not this cluster should be split. This feature increases the system's ability to cope with changing concepts as; later on a test is performed to check if the previously decided split still represents the structure of variables. On stationary data streams, the overall intra-cluster dissimilarity should decrease with each split. This way, if a cluster is split into two child leaves, the diameter of the new clusters should be less or equal than the diameter of the parent node. If the diameter of a leaf is greater than its parent's diameter, then the previously taken decision no longer reflects the structure of data. The system re-aggregates on the cluster's parent, restarting statistics.

INCREMENTAL DISSIMILARITY MEASURES

The system must analyze distances between incomplete vectors, without having any of the previous values available. Thus, these distances must be incrementally computed. Since the decision has to be taken with statistical support, we use the Hoeffding bound to support our decisions, forcing the criterion the distance measure to be scaled. Let be independent random variables. Assume that the X_i are almost bounded; that is, assume for that

Then, for the sum of these variables we have the inequality which is valid for positive values of t (where $E[S]$ is the expected value of S).

The Pearson's correlation coefficient between time series as similarity measure, as done by deriving from the correlation between two time series a and b calculated in, the factors used to compute the correlation can be updated incrementally, achieving an exact incremental expression for the correlation,

$$\text{Corr}(a,b) = (P-(AB/n)) / (\sqrt{((A_2-(A_2/n)) * \sqrt{((B_2-(B_2/n)))))})$$

The sufficient statistics needed to compute the correlation are sufficient statistics easily updated at each time step: A = summation of (a_i) , B = summation of (b_i) , A_2 = summation of (a_i^2) , B_2 = summation of (b_i^2) , P = summation of $(a_i b_i)$. In ODAC, the dissimilarity between variables a and b is measured by an appropriate metric, the Rooted Normalized One Minus Correlation, given by $\text{rnomc}(a, b) = ((1 - \text{corr}(a,b))/2)^2$ with range $[0, 1]$. The cluster's diameter is to be the highest dissimilarity between two time series belonging to the same cluster, or the variable variance in the case of clusters with single variables.

GROWING & CONTROLLING THE HIERARCHY

Growing the Hierarchy

The main procedure of the ODAC system is to grow a tree shaped structure that represents the hierarchy of the clusters present in the data. The system incrementally updates, at each new example arrival, the sufficient statistics needed to compute the dissimilarity matrix, enabling its application to clustering data streams. The dissimilarity matrix for each leaf is only computed when it is being tested for splitting or aggregation, after receiving a minimum number of examples. When processing a new example, only the leaves are updated, avoiding computation of unneeded dissimilarities; this speeds up the process every time the structure grows. This is how the tree structure is grown.

Resolving Ties

There might emerge a case where the two top-most distances are nearly or completely equal. To distinguish the cases where the cluster has many variables nearly equidistant from the cases where there are two or more highly dissimilar variables, a tweak must be done. This application of the system to a data stream with high dimension, possibly with hundreds or thousands of variables, we turn to a heuristic approach.

The system is let to check for the real diameter until we force the splitting and aggregation tests, assuming the leaf has been fed with enough examples, hence it should consider the highest distance to be the real diameter.

Controlling the Growth

The hierarchy may grow unnecessarily, in order to control and avoid such a situation; we define another criterion that has to be fulfilled to perform the splitting. The splitting criterion should reflect some relation among the distances between variables of the cluster. Given this fact, we can impose a cluster to be split if it includes a high difference between the two variables.

The splitting of leaf into a node with two child-leaves if the following condition is met:

$$(d_1 - d_0) |d_1 + d_0 - 2d| > e_k$$

This expression gives the global positioning of the mean with respect to the range of the existing distances, representing two inherent concepts: the farther the highest distance is from the minimum distance, the higher is the possibility of a split occurrence; also, the farther the mean distance is from the average of the maximum and minimum distances $((d_1 + d_0)/2)$, the higher is the probability of splitting.

MEMORY AND TIME COMPLEXITY

The ODAC system presents the required features of an adaptive learning system. For each leaf in which the diameter is known (with confidence level given by the Hoeffding bound) the system tests for aggregation first, so in case of concept drift it will not start to grow unnecessarily. This algorithm presents the method of merging the splitting with the aggregative procedure.

Algorithm ODAC

Input: A set of streaming time series $X = (x_1, x_2, \dots, x_n)$

Output: A hierarchical clustering structure S with leaves (clusters) $L = (l_1, l_2, \dots, l_m)$

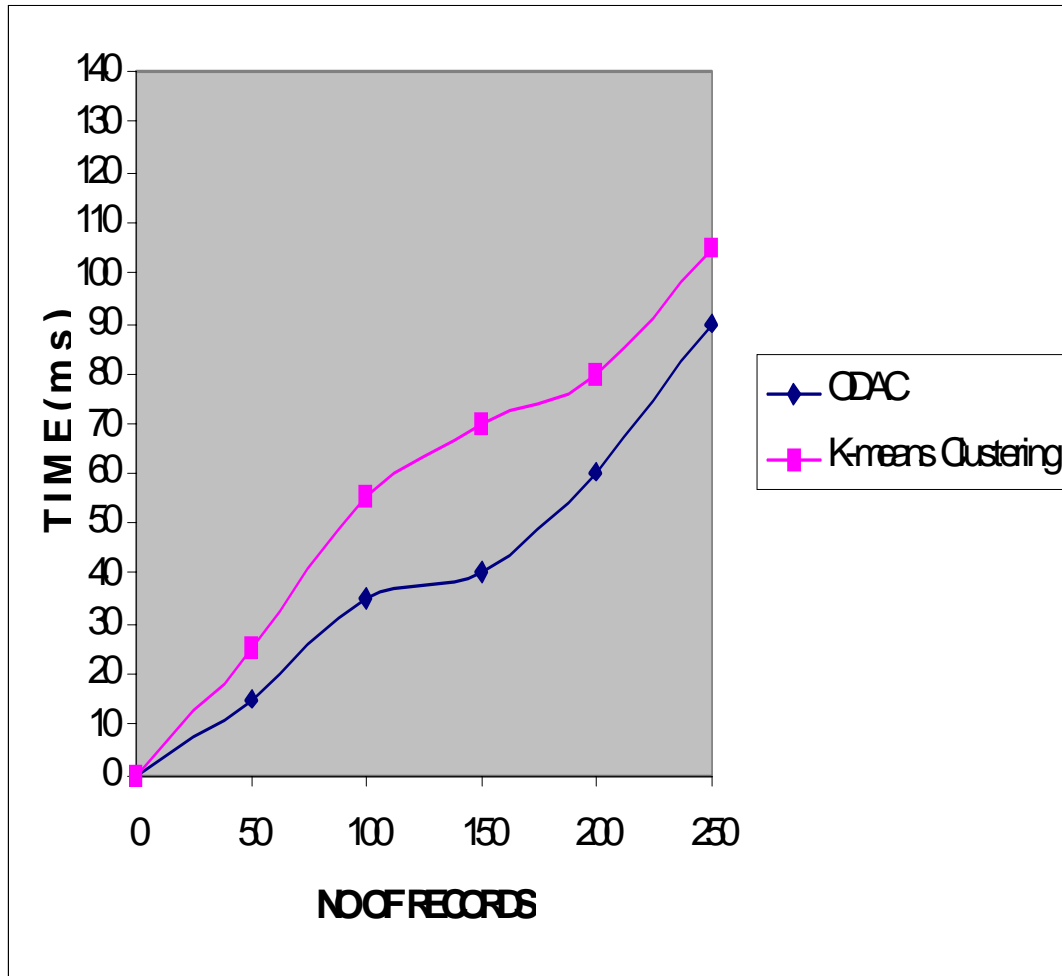
- 1: Repeat
- 2: Read new example X_i and update sufficient statistics on the leaves L
- 3: For each leaf l_k not yet tested do
- 4: Update dissimilarities and the Hoeffding bound e_k for this leaf
- 5: if TestAggregate (l_k) or TestSplit (l_k) then
- 6: announce new structure S ;
- 7: end if
- 8: end for
- 9: until EOF

Complexity analysis can be done with respect to memory usage and time consumption.

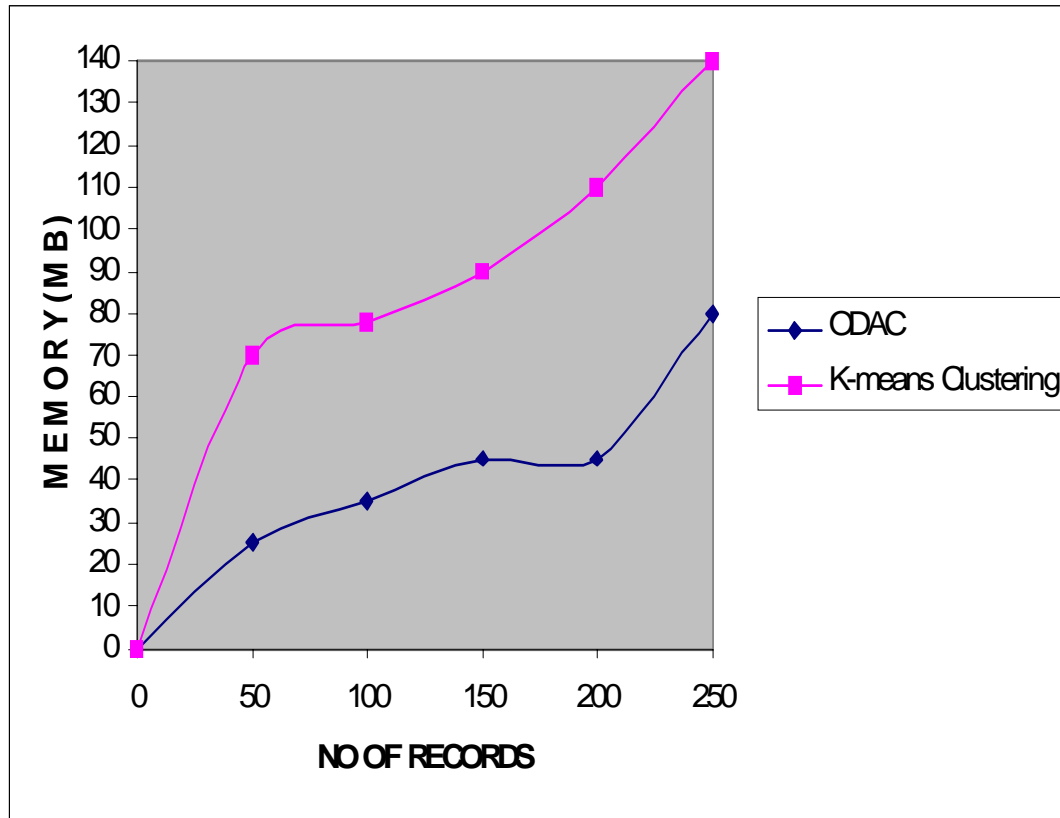
A system which aims at efficiently clustering data streams must comply with constant memory usage. In ODAC, the size needed to keep the sufficient statistics at each node with n variables is $O(n^2)$. Let us consider splitting this node into two new leaves, with n_1 and n_2 streams, where $n = n_1 + n_2$. Considering that $(n_1 + n_2)^2 > n_1^2 + n_2^2$ although the space used by children nodes is still $O(n_1^2 + n_2^2)$.

A system which aims at efficiently clustering data streams must also comply with constant execution time, with respect to the number of examples. When updating the system with a new example, the update of sufficient statistics results in $O(n^2)$ operations. This is, as expected, quadratic in the number of variables. The update does not depend on the number of examples seen. Therefore, the system's update time is constant in respect to the number of examples, satisfying the data stream requirements. The splitting

procedure (algorithm 1) needs to compute the two maximum, the minimum and the mean of the dissimilarity matrix. This procedure is linear in the number of distances, hence $O(n^2)$, quadratic with the number of variables. When computing dissimilarities, assuming the worst case scenario where only one leaf exists, the number of dissimilarities computed is also $O(n^2)$, thus quadratic with respect to the number of variables. An important feature of this algorithm is that every time a split is performed on a worst case scenario and at most $(n/2)^2$ (best case scenario). As a result, the time complexity of each iteration of the system is constant with respect to the number of examples, and decreases with every split; it is therefore capable of addressing continuous data streams.



Time Vs Number of Records



Memory Vs Number of Records

From these figures, it is observed that ODAC takes less time and consumes less memory in comparison with the K-means clustering algorithm.

CONCLUSIONS AND FUTURE WORK

This Online Divisive Agglomerative Clustering is designed towards thousands of data streams that flow at high-rate (Pedroso Rodrigues et al (2008)). Two main characteristics are update time and memory consumption. Both reduce whenever the tree structure grows. This is a major achievement accomplished by ODAC since only dissimilarities at the leaves must be computed. This way, every time the system grows it becomes faster, overcoming the bottleneck of having to compute all dissimilarities at root level, which is known to have quadratic complexity on the number of streams. In phase1 of this project work ODAC algorithm has been implemented and tested.

Further works in this direction could be the design and implementation of a temporal database with suitable indexes and with more historical information. In phase2 of this project work, a new clustering algorithm will be proposed and implemented.

REFERENCES

- [1] P. Rodrigues P, J. Gama, and J. P. Pedroso, "Hierarchical Clustering Of Time Series Data Streams", in Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining. , Vol 20, No 5, pp. 615 -627.
- [2] P. Rodrigues P, J. Gama, and J. P. Pedroso, "ODAC: Hierarchical Clustering Of Time Series Data Streams," in Proceedings of the Sixth SIAM International Conference on Data Mining, April 2006, pp. 499 -503.
- [3] Ying Zhao and George Karypis, "Hierarchical Clustering Algorithms For Document Dataset" University of Minnesota, Technical Report pp 1 -22, 2003.
- [4] A.Kannan, C.J.Date and S.Swamynathan, "Introduction to Database Systems", Pearson Education Asia Ltd, 2006
- [5] Jessica Lin, Michail Vlachos, Eamonn Keogh and Dimitrios Gunopulos, "Iterative Incremental Clustering of Time Series," IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 3, pp. 515 -528, 2003.
- [6] Bruce Walter, Kavitha Bala, Milind Kulkarni, Keshav Pingali, "Fast Agglomerative Clustering for Rendering," Journal of Universal Computer Science, Vol.11, No.8, pp. 1426-1439, 2007
- [7] Sergio M. Savaresi, Daniel L.Boley, Sergio Bittanti and Giovanna Gazzaniga , " Cluster Selection in Divisive Clustering algorithms" University of Minnesota, Technical Report pp 24 -51, 2005.
- [8] Martin Ester, Hans-Peter Kreigel, Jorg Sander, Michael Wimmer, Xiaowei Xu," Incremental Clustering for Mining in a Data Warehousing Environment" Proceedings of the 24th International Conference on Very Large Data Bases, pp 323 -333, 1998.