УДК 637.02.73

Improved and Accelerated Multiresolution Linear Algorithm for Data Clustering

Smirnov Arthur A. asmirn3@uic.edu
University of Illinois at Chicago, Chicago, IL, USA
Sharlay Valery V.
St Petersburg State University of Aerospace Instrumentation
194021 St. Petersburg, Bolshaya Morskaya St., 67
Mehta Parth J.
Amazon USA
Seattle, State of Washington, USA

The paper describes one of the important problems related to databases – clustering problems. Clustering issues have been very important in research in the past not only in the area of database management but also in information security. We are describing what this new method does and how it improves. We develop a new method for local correlation clustering.
Keywords: Artificial Intelligence Techniques, Data Clustering, Databases, MDL method

---

Улучшенный и ускоренный мультимасштабный линейный алгоритм кластеризации данных

Смирнов А.А.
Университет шт. Иллинойс в Чикаго, США
Шарлай В.В.
Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67
Мехта Парт Дж. Pmehta25@uic.edu
Сиэтл, штат Вашингтон, США

Данная статья описывает важные проблемы, связанные с базой данных – проблемы кластеризации. Проблемы кластеризации являются довольно-таки важными на протяжении долгого времени не только в разделе управления базами данных, но и в мерах защиты информации. Мы описываем, как был разработан данный метод. Мы разработали новый метод для локальной корреляции кластеризации.
Ключевые слова: Техники искусственного интеллекта, кластеризация данных, базы данных, метод минимального описания длины.

Introduction

In this paper we are presenting the new method for linear clustering[1]. This is a fast algorithm that locates and then marks in smaller spaces of big-size data – in multidimensional data. The current methods are mostly space and time super-linear [2]. Our method was designed in such way so it improves the basic methods by providing an enhanced and optimized implementation strategy for the counting tree. This method will be able to handle cases when the whole tree is not suitable in memory. Since this could be the case – then operational system disk cache issue should be given a serious consideration. Our idea was to describe the tree in tables which will be located in main memory. Each level is representing a certain level of the tree. The tables consist of key entries.

Methodology

There exists a group of different cluster search algorithms which vary by many factors. Those cluster search algorithms find a few dislocated clusters in all the subsets which cause producing some overlapping clusters [3]. One of the most common problems these days face an issue of the objects being defined such as it is independent of the particular algorithm to detect these clusters. The second important problem that these algorithms face is the definition of the density – this definition is based on user-defined parameters which makes it hard to mark up the clusters in big sets of data. We redefined a new way of how the problem can be formulated that works at extracting parallel regions that have been marked up in the data sets.

The detection of correlations between different features in a certain set of data is a very important task due to the following reason: the correlation has an ability to show whether or not the features are dependent. There is a well-known algorithm called the principal components analysis (PCA) which can shown whether a linear, global or multidimensional dependency can be shown [7]. The method we develop has a determinate result and is therefore very robust.

Clustering can be a very time and space consuming task and may also suffer from the curse of dimensionality and similar functions that use all input features with equal relevance which may not that very efficient. Our idea is to present such an algorithm that will be discovering clusters in smaller spaces (subspaces) spanned by different combinations of dimensions via local weighs of features. This approach avoids any risk of loss of information encountered in the global dimensionality reductions techniques. We associate each clusters with a weighted vector; the weight of such vectors depends on relevance of nearby features around a corresponding vector.

Design of System & Classification

The system consists of three different levels. The top level is Halite. The second level contains four main elements: Dimension, Usability, Efficiency, and Generality. The bottom level consists of five elements listed in the order: Time, Clustering Area, Clustering Sub-Area, Results, Speed; all the elements listed herein produce the outcome into the database storage, which is the final product of this scheme. Please refer to the scheme below:

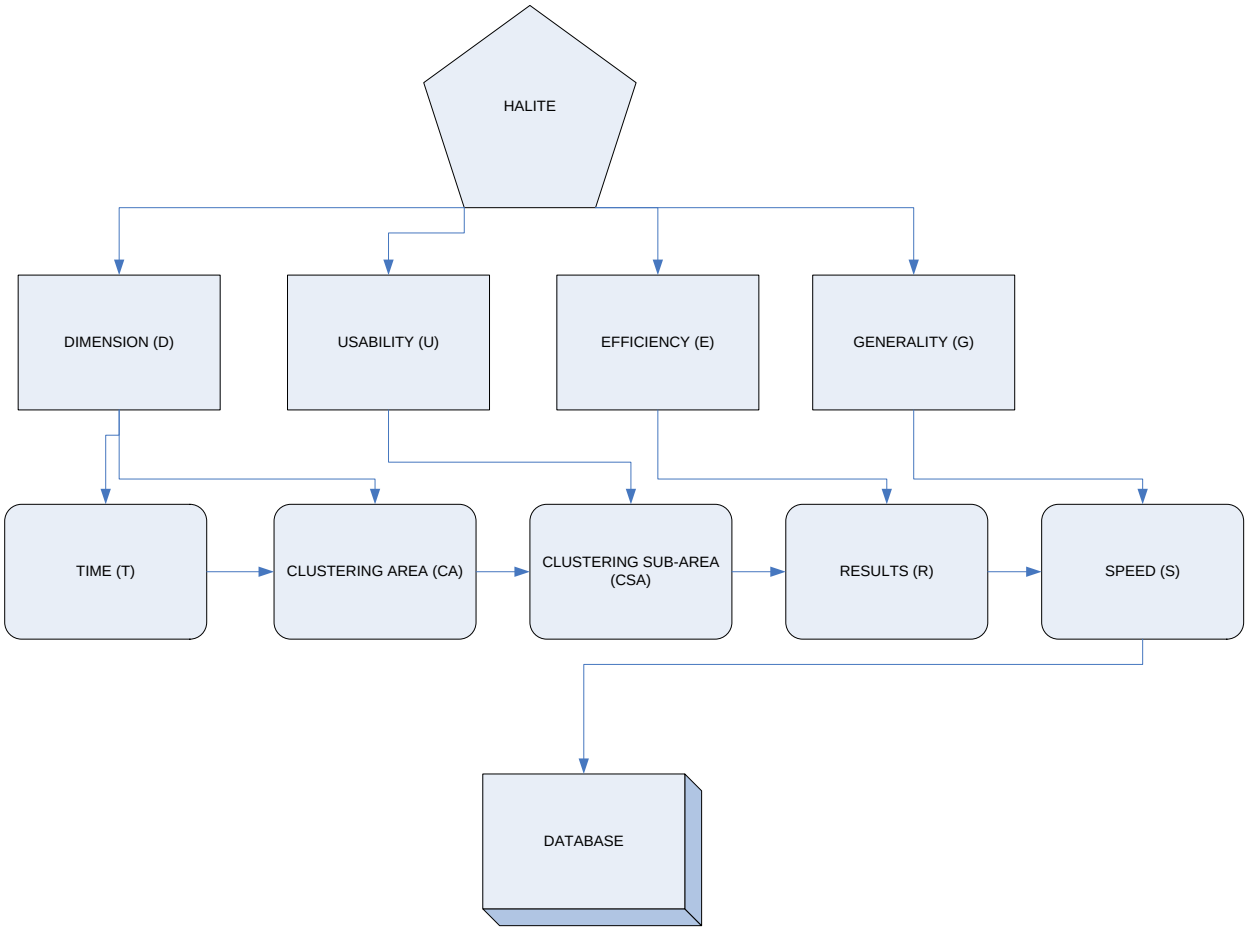


Fig.1 – Design of System

The dataset has to be partitioned in such a way that each point can only belong to one cluster (meaning that the same point cannot belong to two different clusters regardless of how the clusters are divided)[3]. The same rule is applicable when the system is a multidimensional system.

Since the information has to be protected, we have to encode the input data by selecting the minimum description length (MDL method) [6, 7]. This method is used on the top Halite level in order to automatically set up the value of the density threshold. This level also includes a compression-based analysis to mark up the points that are potentially suspicious of belonging to more than one cluster (it could be two or three adjacent clusters).

The next stage of our algorithms implies looking for β clusters. β clusters are detected by applying a variety of convolution masks over each level of the counting tree. Counting trees are such trees that are made of tables and being stored in the main memory. The content of the

counting trees stores value entries of the data related to all non-empty cells. The masks are integers that are outcomes of the Laplacian Filter, a second derivative operator that acts based upon the transitions in density.

Since Halite method for local correlation has a linear space complexity, thus the numbers of points and clusters will be defined at some point. In this implementation of the method, we found out that we require about 30-48% of the data size (amount of memory depends on the distribution of the points). In this case we may face a severe problem if a large dataset is presented. In such case we have to use the operational system's disk cache which at some point eventually may become a bottleneck. In order to overcome this issue, Halite method has a table-based implementation that never uses disk cache regardless of the input dataset. By using this little trick, we can analyze large datasets easily. We would have to represent the counting trees by tables that are stored in the main memory or on the disk. Each table will be representing only one level of the tree which will be storing the same information listed above in the tree description.

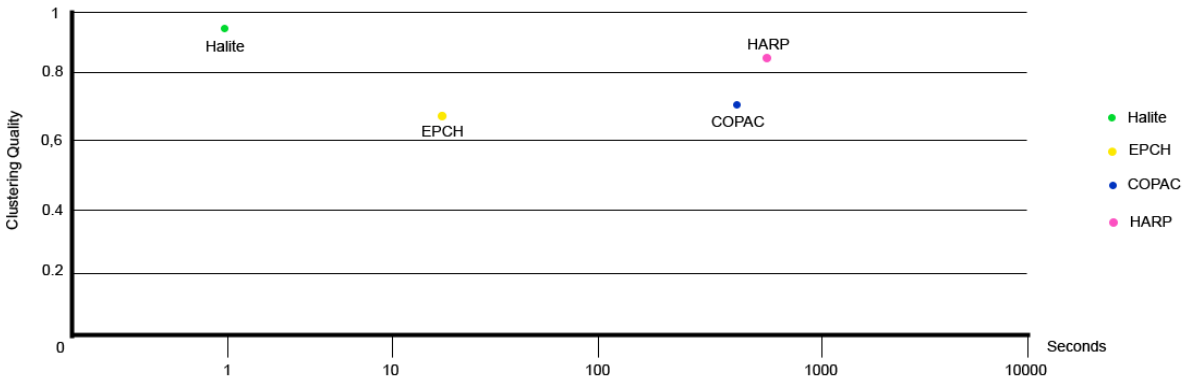
Testing Data

The peak question for us was to properly test large amounts of data. We got a large set of data from Amazon Web Services (AWS). This dataset consists of about 500,000 records and each record is being described by 35 unique categories in which the data can be split into other 1,000 different sub-categories. We tested our algorithm in two different ways on this large dataset. The first implementation was to using the scalability of the algorithm against the number of clusters for a certain given number of objects for a given number of clusters. In order to get the most accurate results, we used the same machine to run both algorithms – Sun Enterprise 6000, a unix-based server with a single processor. The results of running these algorithms will be presented in the final part of this paper. But, we can state that the results are encouraging. We can clearly see a linear increase in the number of both, clusters and records. It took us 57 min. to cluster all the records into 120 clusters. Many different artificial intelligent techniques can be applied in comparing results of running various algorithms.[5] We assume that this running time is acceptable in this case due to a large number of records. When compared, this algorithm is much faster in operating versus when the data consisting of mixed values is being clustered. The main factor in this case is a number of iterations that have to be implemented over the whole dataset.

One of the proposals on how to improve and expedite these algorithms would be is run these algorithms simultaneously on multiple processors which will get us better results. The other main issue that has to be taken care of is parallelizing an allocation operation of the objects.

Results

We have compared different algorithms and the graph below demonstrates obtained results:



F

As the graph above shows, Halite method beats all other three methods that we used in the project. Main two parameters that we consider important are the time and the quality of clustering[4]. The scale of clustering quality was translated from 100% to 1 for easier understanding. The other important parameter was amount of time taken to run the algorithm – as see above, the fastest one is Halite. The second fast algorithm is EPCH, it takes about 22 sec to run the algorithm all the way. The slowest one is HAPR, although its efficiency is higher than EPCH, which is the second fast one and COPAC, which is faster than HAPR but is less quality.

Overall, it is obvious that using Halite method will be both fast and qualitative. The second algorithm which we would recommend to use is HAPR, but it takes more time, but on the other hand, this algorithm is easier to implement. The middle version of all these algorithms is COPAC that is a bit less qualitative but a bit faster than HARP.

References (Список литературы)

1. Jagadish, H. V. "Linear clustering of objects with multiple attributes." *ACM SIGMOD Record*. Vol. 19. No. 2. ACM, 1990.
2. Nagashima, Umpei, et al. "An experience with super-linear speedup achieved by parallel computing on a workstation cluster: Parallel calculation of density of states of large scale cyclic polyacenes." *Parallel computing* 21.9 (1995): 1491-1504.
3. Jain, Anil K., M. Narasimha Murty, and Patrick J. Flynn. "Data clustering: a review." *ACM computing surveys (CSUR)* 31.3 (1999): 264-323.

4. Jain, Anil K. "Data clustering: 50 years beyond K-means." *Pattern Recognition Letters* 31.8 (2010): 651-666.
5. Smirnov, Arthur. *Artificial intelligence: Concepts and Applicable Uses*. LAP LAMBERT Academic Publishing, 2013
6. Koivisto, M. "An MDL Method for Finding Haplotype Blocks and for Estimating the Strength of Haplotype Block Boundaries M. Koivisto, M. Perola, T. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, E. Ukkonen, H. Mannila Pacific Symposium on Biocomputing 8: 502-513 (2003)." *Pacific Symposium on Biocomputing*. Vol. 8. 2003.
7. Huang, Lei, and Shunjun Wu. "Low-complexity MDL method for accurate source enumeration." *Signal Processing Letters, IEEE* 14.9 (2007): 581-584.
8. Wilson, Robert P., et al. "SUIF: An infrastructure for research on parallelizing and optimizing compilers." *ACM Sigplan Notices* 29.12 (1994): 31-37.
9. Jolliffe, Ian. *Principal component analysis*. John Wiley & Sons, Ltd, 2005.