New method for automated clustering on data streams.

Jadwiga Piechota
AGH University of Science and Technology
Krakow, Poland
jadwiga.piechota@onet.pl

13.09.2019

Abstract

Clustering is one of the most common techniques for grouping a set of objects based on their statistical features. In machine learning, it is the unsupervised classification that distinguish patterns that can occur along the data. In mobile application data for analysis comes from a wide range of sensors provided by mobile device. To make results up to date, it is necessary to deal with stream of data that passes over a pipeline. Learning process has to be applied multiple times, depends on environments conditions change. In this paper we introduce a new approach of DBScan (ang. Density-based Spatial clustering of applications with noise) clustering algorithm that takes into account a stream of data received from GPS (ang. Global Positioning System) sensor. DBScan creates clusters that point the most frequently visited region by the user of app. We also make a quick comparison between our and other methods of clustering that can be used with streaming data.

1 Introduction

Context-aware systems have made a big contribution to development of ambient intelligence. Awareness and understanding external factors are the main sources for reasoning such systems. Environment, through a wide range of sensors, provides information that can be used in further computation. The amount of data that has to be processed by application is huge and has the significant reason for model update. Changing conditions and appropriate reaction in response cause such systems can be interpreted as intelligent as they can make the decision without human interaction.

In this work, the primary focus is on adapting mobile applications to user experience. Personalization poses an important role here in reflecting customer satisfaction. Making software individual oriented we allow that any changes are making impact on real-time results. Application evolves to be up to date with actual data. Additional advantage is that this technique is often used by recommendation systems. Thus, getting insights on user preferences is the key concept in business strategy developing intelligent solutions.

On the other hand, due to the mobility of devices on which context-aware applications are deployed, there is a bunch of risk factors that we have to face. Dynamics behavior, that characterizes this environment, is a big challenge for software developers. Data, that needs to be processed, flows in large quantities and has to be interpreted as system should react to new conditions and try to adjust the parameters to actual situation. Constant updates from sensors are a big challenge and designer has to take into consideration constrains that come from mobile devices. The scope of limitations include memory resources, consuming CPU (ang. Central Processing Unit) and energy efficiency. However, mobile application must be characterized by fast responsiveness and low latency. In this case, at the application development stage, the right balance between dependencies should be found.

One of the most commonly used contextual information in described systems is geolocation. The source of this data is GPS sensor that returns actual position of device on the Earth globe. This value consists of two fields: longitude and latitude that can be applied in further computations. It is claimed that the accuracy of this technology ranges within a radius of 10 meters or even better [Djuknic and Richton, 2001]. This precision has a big potential and different research areas have a benefit from this. One of them is the use of clustering methods for determining places on the map that are especially important for the user. Such actions are aimed at learning habits and as a consequence, to better match the displayed content of application. This approach is more apt because the current location can be just temporary and has random character. Cluster consists of points that are similar

to each other and thus, the results that are received are more reliable. In proposed solution, DBScan clustering algorithm was applied.

The main contribution of this work is creating an advanced version of DBScan clustering algorithm – DBScanSTREAM. The functionality that was added fits perfectly into context-aware systems. Constant stream of data can be processed by proposed application and results in up to date cluster creation. What separate new algorithm from the basic one is fact that old locations are forgotten and are not involved in the creation of new clusters. In results, there is a bigger chance that using the new approach will be more effective and results in reflecting reality.

The structure of the paper is organized as follows. In Section II history of similar algorithms is described, and related works are discussed. The DBScanSTREAM approach is introduced in Section III. Section IV explains how new algorithm has been implemented in mobile, context-aware devices. Experimental results of created application on real data are presented in Section V. Finally, Section VI summarizes what has been done and provides conclusions and remarks.

2 History and related work

The notion of clustering was introduced in anthropology in 1932 [Driver and Kroeber, 1932] and referred to the grouping of similar objects. As it was said in [Jain et al., 1999], in many research communities the term clustering is used to describe methods for grouping of unlabeled data. Two basic types of clustering algorithms exist: partitioning and hierarchical. There are three main steps for pattern clustering according to [Jain and Dubes, 1988]: (1) pattern definition, (2) proximity measure that is adapted to the data type and (3) running the clustering algorithm. That means that this clustering method can be defined as dividing domain into groups in the way that elements belonging to the same group are more similar to each other than to the rest of the objects.

DBScan clustering algorithm was designed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996 [Ester et al., 1996]. It is based on density distribution of objects that are subject to grouping in closely related neighborhood. Objects placed in low-density regions are marked as outliers and are excluded from clustering in actual iteration. Comparison of objects are obtained by computing a distance between analyzing point and rest of the points in the database. Optimally the algorithm needs just three parameters: database, epsilon neighborhood and minimum number of points required for cluster existence.

A slightly different approach of clustering was described in [Aggarwal et al., 2003]. The authors of this publication presented a new algorithm for streaming data named CluStream. Main philosophy besides this term is focused on how to manage a big volume of data flooding. The solution, they came up with, turn out to be creation of two components – online (for periodical statistics) and offline (for summary statistics). Additionally, the number of clusters in CluStream is constant and in case new cluster has to be attached, the oldest one is removed or merged with another one.

To meet the challenges posed by lots of organizations that produce data in growing rate, it was necessary to change a way of thinking. Examining of requirements that have to be fulfill results in introducing of concept tracking streams of data changes on the clustering models. The key was to design the pattern that has to be followed by algorithms dealing with such problem, without consuming lots of memory and processing resources. This kind of solution was proposed by Barbara in [Barbará, 2002] by pointing out the most important rules. These criteria formed the basis during the construction of the algorithm proposed in this work.

3 DBScanSTREAM algorithm

The presented algorithm has a few of changes compare to common DBScan. These changes allow for computing clusters in effective way on mobile devices where the scope of resources is limited and context changes in dynamic way.

3.1 Data representation

To decrease amount of data that has to be stored in memory, the new idea of cluster representation was proposed. Instead of keeping value of all the points that ale included in the cluster, this number was restricted to positions of four corners of the smallest rectangle that is capable to surround this cluster. This method is named MBR and stands for Minimal Bounding Rectangle and it was shown in (fig. 1). It was described in detail in [Gaffuri, 2009].

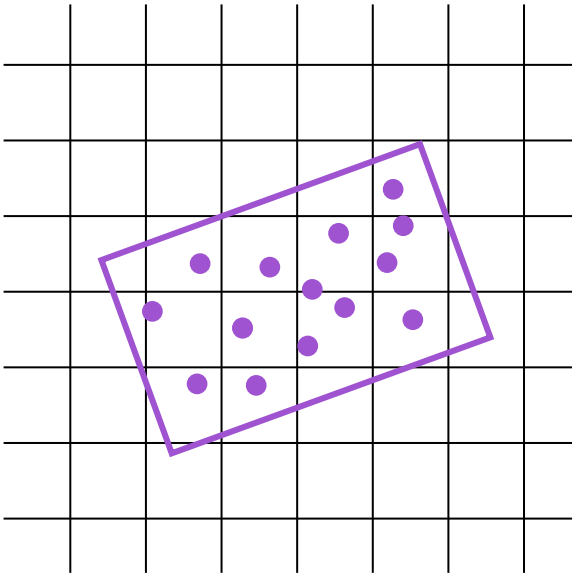

Figure 1: Minimum bounding rectangle that cover all points found for analyzing cluster.

3.2 Algorithm steps

First iteration of DBScanSTREAM executes resulting in computing clusters as it was done in the basic version of algorithm. Each next run starts with the download of clusters that were found in the previous iteration. The important thing to note that each cluster has dedicated weight responsible for its significance compare to the others. This value is decrease after every run of algorithm. This implies clusters that were not visited for a long time can be forgotten what helps keeping application up to date. Such weight may be increased what is mentioned in the next section. In each case the learning process operates on specific range of data. It is determined by timestamp which indicates the start point for reading location entries. In this place, there is invoked DBScan algorithm that computes clusters based on newly discovered records. The next step is to estimate the weight of cluster. It is done by assigning the value equals to the number of points that are covered by interpreted cluster. Last step relate to comparing and potential merging clusters form previous and the last one iteration. There are introduced rules simulating location change. This means we do not take into consideration points that were visited last long time ago. In that way each execution retrains our model in context of new information that has been processed by the pipeline depicted in the Listing 1.

Listing 1: DBScanSTREAM algorithm sequence.

```
if not first_execution then
    read previous_result from database
    if previous_result then
        get previous_clusters from previous_result
        get previous_execution_time from previous_result
    end if
end if
if not previous_execution_time then
    get all available location_data
else
    get location_data since previous_execution_time
end if
for previous_cluster in previous_clusters do
    decrease previous_cluster_weight
end for
new_clusters ← cluster location_data using DBScan
for new_cluster in new_clusters do
    new_clusterMBR ← find MBR of the new_cluster
    for previous_cluster in previous_clusters do
        if new_clusterMBR ∩ previous_clusterMBR > area_threshold then
            join new_clusterMBR and previous_clusterMBR
            join new_cluster_weight and previous_cluster_weight
        end if
    end for
end for
for cluster in all_clusters do
    if cluster_weight < cluster_threshold then
        remove cluster
    end if
end for
send clustering_result
```

3.3 Cluster actualization

This aspect was designed to adapt presented algorithm for streaming data. In order to avoid grouping points infinitely with no information provided, there was a must to devise a solution for updating clusters. The authors of this algorithm proposed the following strategy. DBScan run returns the new created clusters from database starting from defined timestamp. These results have to be compared with clusters available from the previous iteration. In this process two parameters are used: the area and weight. Based on the analysis outcome, new clusters can be joined together with the old ones or leaved as separated objects. If they are joined, their weights and areas are added together. This concept was designed because of inaccuracy in determining the MBR. Each execution of DBScanSTREAM decreases the value of weight for existing clusters to simulate their declining. In the mobile application, the user can change a rate of this process manually. This feature allows to avoid random visited places having impact on final rectangle formation. Before iteration results will be broadcasted, decision about throwing clusters away has to be made. It is done based on the rule that says a cluster is still valid if its weight to the power of three must be bigger than area of its MBR. This resolves the problem of infinite growing and absorption of clusters.

4 Experiments and results

5 Conclusions and remarks

References

[Aggarwal et al., 2003] Aggarwal, C., Han, J., Wang, J., and Yu, P. (2003). A Framework for Clustering Evolving Data Streams. *VLDB '03 Proceedings of the 29th international conference on Very large data bases*, 29:81–92.

[Barbará, 2002] Barbará, D. (2002). Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2):23–27.

[Djuknic and Richton, 2001] Djuknic, G. and Richton, R. (2001). Geolocation and assisted GPS. *Computer, IEEE*, 34(2):123 – 125.

[Driver and Kroeber, 1932] Driver, H. E. and Kroeber, A. L. (1932). Quantitative Expression of Cultural Relationships. *University of California Publications in American Archaeology and Ethnology*, Quantitative Expression of Cultural Relationships:211–256.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press*, pages 226–231.

[Gaffuri, 2009] Gaffuri, J. (2009). Three reuse example of a generic deformation model in map generalisation. *24th International Cartographic Conference, At Santiago, Chile.*

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs: Prentice Hall, ii edition.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323.