Clustering Large Datasets using Data Stream Clustering Techniques

Matthew Bolaños[1], John Forrest[2], and Michael Hahsler[1]

[1] Southern Methodist University, Dallas, Texas, USA.
[2] Microsoft, Redmond, Washington, USA.

Abstract. Unsupervised identification of groups in large data sets is important for many machine learning and knowledge discovery applications. Conventional clustering approaches (k-means, hierarchical clustering, etc.) typically do not scale well for very large data sets. In recent years, data stream clustering algorithms have been proposed which can deal efficiently with potentially unbounded streams of data.

This paper is the first to investigate the use of data stream clustering algorithms as light-weight alternatives to conventional algorithms on large non-streaming data. We will discuss important issue including order dependence and report the results of an initial study using several synthetic and real-world data sets.

1 Introduction

Clustering very large data sets is important for may applications ranging from finding groups of users with common interests in web usage data to organizing organisms given genetic sequence information. The data are often not only so large that they do not fit into main memory, but they even have to be stored in a distributed manner. Conventional clustering algorithms typically require repeated access to all the data which is very expensive in a scenario with very large data.

Data stream clustering has become an important field of research in recent years. A data stream is an ordered and potentially unbounded sequence of objects (e.g., data points representing sensor readings). Data stream algorithms have been developed in order to process large volumes of data in an efficient manner using a single pass over the data while having only minimal storage overhead requirements. Although these algorithms are designed for data streams, they obviously can also be used on non-streaming data.

In this paper we investigate how to use data stream clustering techniques on large, non-streaming data. The paper is organized as follows. We introduce the problems of clustering large data sets and data stream clustering in Sections 2 and 3, respectively. Section 4 discusses issues of the application of data stream clustering algorithms to non-streaming data. We present results of first experiments in Section 5 and conclude with Section 6.

2 Clustering large data sets

Clustering groups objects such that objects in a group are more similar to each other than to the objects in a different group (Kaufman and Rousseeuw (1990)). Formally clustering can be defined as:

Definition 1 (Clustering). *Partition a set of objects $\mathscr{O} = \{o_1, o_2, \ldots, o_n\}$ into a set of clusters $\mathscr{C} = \{C_1, C_2, \ldots, C_k, C_\varepsilon\}$, where k is the number of clusters and C_ε contains all objects not assigned to a cluster.*

We restrict clustering here to hard (non-fuzzy) clustering where $C_i \cap C_j = \emptyset$ for all $i, j \in \{1, 2, \ldots, k, \varepsilon\}$ and $i \neq j$. Unassigned objects are often considered noise, however, many algorithms cannot leave objects unassigned (i.e., $C_\varepsilon = \emptyset$). The ability to deal with noise becomes more important in very large data sets where manually removing noise before clustering is not practical. The number k is typically user-defined, but might also be determined by the clustering algorithm. In this paper, we assume that the objects are embedded in a d-dimensional metric space ($o \in \mathbb{R}^d$) where dissimilarity can be measured using Euclidean distance. We do not deal with the problem of finding the optimal number of clusters, but assume that a reasonable estimate for k is available.

The most popular conventional clustering methods are k-means type clustering, hierarchical clustering and density-based clustering. All these methods have in common that they do not scale well for very large data sets since they either need several passes over the data or they create data structures that do not scale linearly with the number of objects. We refer the reader to the popular book by Jain and Dubes (1988) for details about the various clustering methods.

To cluster large data sets, researchers have developed parallel computing approaches, most notably using Google's MapReduce framework (e.g., for k-means see Zhao et al. (2009)). On the other hand, researchers started earlier to reduce the data size by sampling. For example, CLARA (Kaufman and Rousseeuw (1990)) uses sampling and then applies Partitioning Around Medoids (PAM) on the samples and returns the best clustering. Another algorithm, BIRCH (Zhang et al. (1996)), builds a height balanced tree (also known as a cluster feature or CF tree) in a single pass over the data. The tree stores information about subclusters in its leaf nodes. During clustering, each data point is either added to an existing leaf node or a new node is created. Some reorganization is applied to keep the tree at a manageable size. After all data points are added to the CF tree, the user is presented with a list of subclusters. BIRCH was developed by the data mining community and resembles in many ways the techniques used in data stream clustering which we will discuss in the next section.

3 Data stream clustering

We first define data stream since data stream clustering operates on data streams.

Definition 2 (Data stream). *A data stream is an ordered and potentially unbounded sequence of objects* $\mathscr{S} = \langle \mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \ldots \rangle$.

Working with data streams imposes several restrictions on algorithms. It is impractical to permanently store all objects in the stream which implies that there is limited time to process each object and repeated access to all objects is not possible.

Over the past 10 years a number of data stream clustering algorithms have been developed. For simplicity, we will restrict the discussion in this paper to algorithms based on micro-clusters (see Gama (2010)). Most data stream clustering algorithms use a two stage online/offline approach.

Definition 3 (Online stage). *Summarize the objects in stream* $\mathscr{S}$ *in real-time (i.e., in a single pass over the data) by a set of* k' *micro-clusters* $\mathscr{M} = \{m_1, m_2, \ldots, m_{k'}\}$ *where* m_i *with* $i = \{1, 2, \ldots, k'\}$ *represents a micro-cluster in a way such that the center, weight, and possibly additional statistics can be computed.*

When the user requires a clustering, the offline stage reclusters the micro-clusters to form a final (macro) clustering.

Definition 4 (Offline stage). *Use the* k' *micro-clusters in* $\mathscr{M}$ *as pseudo-objects to produce a set* $\mathscr{C}$ *of* $k \ll k'$ *final clusters using clustering defined in Definition 1.*

Note that while k often is specified by the user, k' is not fixed and may grow and shrink during the clustering process. Micro-clusters are typically represented as a center and each new object is assigned to its closest (in terms of a proximity measure) micro-cluster. Some algorithms also use a grid and micro-clusters represent non-empty grid-cells. If a new data point cannot be assigned to an existing micro-cluster, typically a new micro-cluster is created. The algorithm may also do some housekeeping (merging or deleting micro-clusters) to keep the number of micro-clusters at a manageable size.

Since the online component only passes over the data once and the offline component operates on a drastically reduced data set which typically fits into main memory, data stream clustering algorithms can be used efficiently on very large, disk-resident data. For a comprehensive treatment of data stream clustering algorithms including some single-pass versions of k-means and related issues, we refer the interested reader to the books by Aggarwal (2007) and Gama (2010).

4 Data stream clustering of non-streaming data

Applying a data stream clustering algorithm to non-streaming data is straightforward. To convert the set of objects $\mathscr{O}$ into a stream $\mathscr{S}$, we simply take one object at a time from $\mathscr{O}$ and hand it to the clustering algorithm. However, there are several important issues to consider.

A crucial aspect of data streams is that the objects are temporally ordered. Many data streams are considered to change over time, i.e., clusters move, disappear or new clusters may form. Therefore, data stream algorithms incorporate methods to put more weight on current data and forget outdated data. This is typically done by removing micro-clusters which were not updated for a while (e.g., in CluStream; Aggarwal et al. (2003)) or using a time-dependent exponentially decaying weight for the influence of an object (most algorithms). For large, stationary data sets, where order has no temporal meaning and is often arbitrary, this approach would mean that we put more weight on data towards the end of the data set while loosing the information at the beginning. Some data stream clustering algorithms allow us to disable this feature, e.g., using a very large horizon parameter for CluStream forces the algorithm not to forget micro-clusters and instead merge similar clusters. For many other algorithms, the decay rate can be set to 0 or close to 0. For example in DenStream (Cao et al. (2006)) a value close to 0 reduces the influence of the order in the data. However, setting it to 0 makes the algorithm unusable since removing small micro-clusters representing noise or outliers depends on the decay mechanism. It is very important to established if and how this type of order dependence can be removed or reduced before applying a data stream clustering algorithm to non-streaming data.

Since data stream clustering algorithms use a single pass over the data, the resulting clustering may still be order dependent. This happens for algorithms, where the location of the created micro-clusters is different if objects are added in a different order. However, this type of order dependency typically only effects micro-cluster placement slightly and our results below indicate that after reclustering, the final clustering is only effected minimally.

Another issue is that data stream clustering algorithms dispose of the objects after they are absorbed by a micro-cluster and since the data stream is expected to be unbounded, not even the cluster assignments of the objects are retained. The only information that is available after reclustering is the set of micro-clusters $\mathscr{M}$ and the mapping of micro-clusters onto final clusters $f_{\text{macro}} : \mathscr{M} \mapsto \mathscr{C}$. In order to infer the mapping of objects in $\mathscr{O}$ to cluster labels, we find for each object o the closest micro-cluster and then use f_{macro} to retrieve the cluster label in $\mathscr{C}$. Note that this approach works not only for reclustering that produces spherical clusters but also for reclustering that produces arbitrarily shaped clusters (e.g., with single-linkage hierarchical clustering or density based clustering).

5 Comparing different clustering methods

To perform our experiments we use **stream**,[1] a R-extension currently under development which provides an intuitive interface for experimenting with data streams and data stream algorithms. It includes the generation of synthetic data, reading of disk-resident data in a streaming fashion, and a growing set of data stream mining algorithms (including some from the MOA framework by Bifet et al. (2010)). In this first study, we only evaluate sampling and the online component of three of the more popular clustering methods suitable for data streams.

[1] **stream** is available at `http://R-Forge.R-Project.org/projects/clusterds/`

Table 1. Data sets

Dataset	Number of objects	Dimensions	Clusters	Noise
Mixture of Gaussians	100,000	d^*	k^*	$n\%^*$
Covertype	581,012	10	7	unknown
16S rRNA	406,997	64	110	unknown

* Values correspond with the data sets' names.

- Reservoir Sampling (Vitter (1985))
- BIRCH (Zhang et al. (1996))
- CluStream (Aggarwal et al. (2003))
- DenStream (Cao et al. (2006))

For evaluation, we use the data sets shown in Table 1. We use several mixture of Gaussians data sets with k clusters in a d-dimensional hypercube created with randomly generated centers and covariance matrices similar to the method suggested by Jain and Dubes (1988).[2] Some clusters typically overlap. For one set we add $n = 20\%$ noise in the form of objects uniformly distributed over the whole data space.

The Covertype data set[3] contains remote sensing data from the Roosevelt National Forest of northern Colorado for 7 different forest cover types. We use for clustering the 10 quantitative variables.

The 16S rRNA data set contains the 3-gram count for the more than 400,000 16S rRNA sequences currently available for bacteria in the Greengenes database.[4] 16S sequences are about 1500 letters long and we obtain 64 different 3-gram counts for the 4 letters in the RNA alphabet. Since these sequences are mainly used for classification, we use the phylum, a phylogenetic rank right below kingdom, as ground truth.

All data sets come from a stationary distribution and the order in the data is arbitrary making them suitable for evaluating clustering of non-streaming data.

5.1 Evaluation method

We cluster each data set with each data stream clustering method. We reduce decay and forgetting in the algorithms by using appropriate parameters (horizon $= 10^6$ for CluStream and $\lambda = 10^{-6}$ for DenStream). Then we tune each algorithm for each data set to generate approximately 1000 micro-clusters to make the results better comparable. Finally, we recluster each data stream clustering algorithm's result using weighted k-means using the known number of clusters for k.

[2] Created with the default settings of function `DSD_Gaussian_Static()` in **stream**.

[3] Obtained from UCI Machine Learning Repository at `http://archive.ics.uci.edu/ml/datasets/Covertype`

[4] Obtained from Greengenes at `http://greengenes.lbl.gov/Download/Sequence_Data/Fasta_data_files/current_GREENGENES_gg16S_unaligned.fasta.gz`

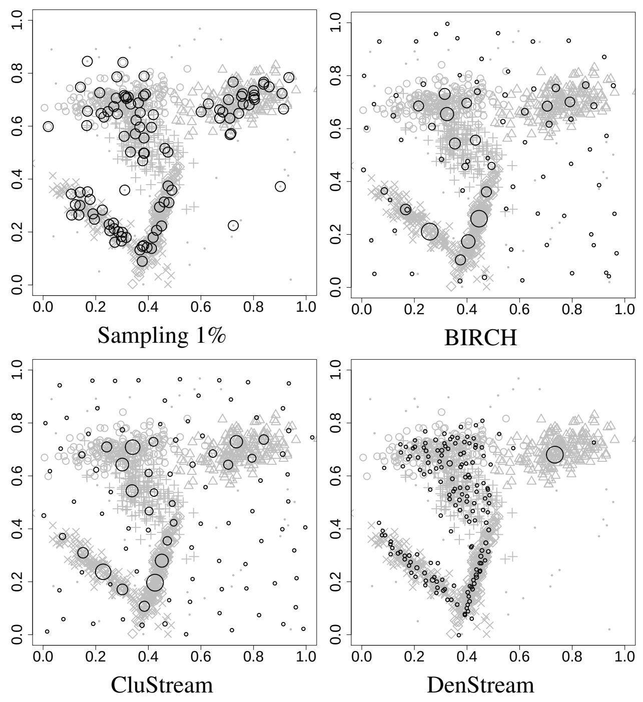

Fig. 1. About 100 micro-clusters placed by different algorithms on a mixture of $k = 5$ Gaussians in $d = 2$ (shown in gray) with 5% noise. Micro-cluster weights are represented by circle size.

We chose data sets with available ground truth in order to be able to use an external evaluation measure. Studies showed that the corrected Rand index (Hubert and Arabie (1985)) is an appropriate external measure to compare partitions for clustering static data (Milligan and Cooper (1986)). It compares the partition produced via clustering with the partition given by the ground truth using the Rand index (a measure of agreement between partitions) corrected for expected random agreements. The index is in the interval $[-1,1]$, where 0 indicates that the found agreements can be entirely explained by chance and the higher the index, the better the agreement. The index is also appropriate to compare the quality of different partitions given the ground truth (Jain and Dubes (1988)) as is done in this paper.

5.2 Results

First, we take a look at how different algorithms place micro-clusters since this gives us a better idea of how well reclustering will work. In Figure 1 we apply all four algorithms on a simple mixture of Gaussians data set with 10,000 data points and 5% noise. We either set or tuned all algorithms to produce about 100 micro-clusters. Reservoir sampling (1%) in Figure 1 concentrates on the dense areas and only selects few noise points. However, we can see that some quite dense areas do not have

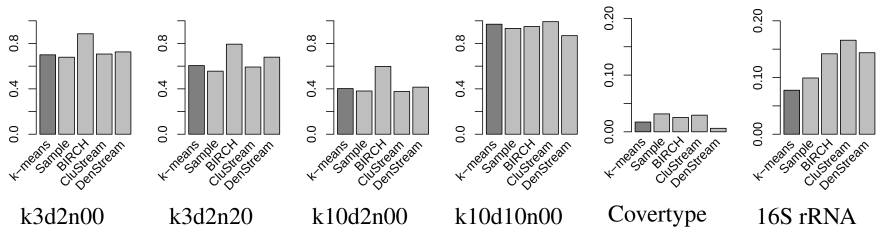

Fig. 2. Corrected Rand index for different data sets with *k*-means reclustering.

a representative. Also, no weights are available for sampling. BIRCH places micro-clusters relatively evenly with heavier micro-clusters in denser areas. Since BIRCH tries to represent all objects, noise creates many very light micro-clusters. CluStream produces results very similar to BIRCH. DenStream tends to create a single heavy cluster for a dense area, but often micro-cluster compete for larger dense areas resulting in a cloud of many very light clusters. A big difference to the other methods is that DenStream is able to suppress noise very well.

Next, we look at the order dependence of each method. We use the same data set as above, but randomly reorder the data 10 times, run the four algorithms on it and then recluster with weighted *k*-means and $k = 5$. We assign the 10,000 objects to the found clusters using the method discussed above in Section 4 and then compare the assignments for the 10 different orders (45 comparisons) using the corrected Rand index. The average corrected Rank index is relatively high with .74 (sampling), .85 (BIRCH), 0.81 (CluStream) and 0.79 (DenStream). Sampling has the lowest index, however, this is not caused by the order of the data since random sampling is order independent, but by the variation caused by choosing random subsets. The higher index for the other methods indicates that, using appropriate parameters, order dependence is below the variability of a 1% sample.

Finally, we cluster and recluster the artificial and real data sets and report the corrected Rank index between the clustering and the known ground truth in Figure 2. We replicate each experiment for the artificial data 10 times and report the average corrected Rand index. For comparison, the result of *k*-means on the whole data set is reported. BIRCH performs extraordinarily well on the artificial data sets with low dimensionality where it even outperforming directly using *k*-means. For noisy data (k3d2n20), we see that all algorithms but DenStream degrade slightly (from k3d2n00). This can be explained by the fact, that DenStream has built-in capability to remove outliers. For higher-dimensional data (k10d10n00 and the real data sets) CluStream performs very favorably.

6 Conclusion

The experiments in this paper indicate a potential for using data stream clustering techniques for efficiently reducing large data sets to a size manageable by conventional clustering algorithms. However, it is important to carefully analyze the algorithm and remove or reduce the order dependency inherent in data stream clustering algorithms. More thorough experiments on how different methods perform is needed. However, the authors hope that this paper will spark more research in this area, leading to new algorithms dedicated to clustering large, non-streaming data.

Acknowledgments

This work is supported in part by the U.S. National Science Foundation as a research experience for undergraduates (REU) under contract number IIS-0948893 and by the National Institutes of Health under contract number R21HG005912.

References

AGGARWAL, C. (2007): *Data Streams: Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, New York, NY.

AGGARWAL, C. C., HAN, J., WANG, J. and YU, P. S. (2003): A framework for clustering evolving data streams. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB '03)*. 81–92.

BIFET, A., HOLMES, G., KIRKBY, R. and PFAHRINGER, B. (2010): MOA: Massive online analysis. *Journal of Machine Learning Research*, *99, 1601–1604*.

CAO, F., ESTER, M., QIAN, W. and ZHOU, A. (2006): Density-based clustering over an evolving data stream with noise. In: *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 328–339.

GAMA, J. A. (2010): *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, Boca Raton, FL, 1st edition.

HUBERT, L. and ARABIE, P. (1985): Comparing partitions. *Journal of Classification*, *2(1), 193–218*.

JAIN, A. K. and DUBES, R. C. (1988): *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

KAUFMAN, L. and ROUSSEEUW, P. J. (1990): *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York.

MILLIGAN, G. W. and COOPER, M. C. (1986): A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, *21(4), 441–458*.

VITTER, J. S. (1985): Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, *11(1), 37–57*.

ZHANG, T., RAMAKRISHNAN, R. and LIVNY, M. (1996): BIRCH: An efficient data clustering method for very large databases. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. ACM, 103–114.

ZHAO, W., MA, H. and HE, Q. (2009): Parallel k-means clustering based on MapReduce. In: *Proceedings of the 1st International Conference on Cloud Computing*. Springer-Verlag, Berlin, Heidelberg, CloudCom '09, 674–679.