A Study of the Efficiency and Accuracy of Data Stream Clustering for Large Data Sets

Michael Hahsler & Matthew Bolaños

Intelligent Data Analysis Group
Southern Methodist University

2013 INFORMS Annual Meeting
Minneapolis Convention Center, October 6–9, 2013

SMU | Bobby B. Lyle School of Engineering

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.

Table of Contents

Motivation

Clustering large data sets is important for many applications:

- Cluster customers for segmentation (Amazon, Netflix, etc.)
- Cluster documents to organize the information space (PubMed, arXiv, etc.)
- Cluster meteorological data to categorize phenomena (e.g., storms)
- Cluster genetic sequences for classifying organisms.

Data is typically large with respect to

- Number of objects
- Number of expected clusters
- Dimensionality

Motivation

Clustering large data sets is important for many applications:

- Cluster customers for segmentation (Amazon, Netflix, etc.)
- Cluster documents to organize the information space (PubMed, arXiv, etc.)
- Cluster meteorological data to categorize phenomena (e.g., storms)
- Cluster genetic sequences for classifying organisms.

Data is typically large with respect to

- Number of objects
- Number of expected clusters
- Dimensionality

Problems:

- Conventional clustering methods become very expensive.
- Large data are noisy.

Investigate data stream clustering methods.

Table of Contents

Clustering

Clustering [Jain et al., 1999], the grouping of similar objects, is an important task in statistics/data mining/machine learning.

Hard clustering with noise

Partition a set of objects

$$\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$$

into a set of clusters

$$\mathcal{C} = \{C_1, C_2, \ldots, C_k, C_\epsilon\},$$

where k is the number of clusters and C_ϵ contains all objects not assigned to a cluster (noise). We restrict clustering here to hard (non-fuzzy) clustering where $C_i \cap C_j = \emptyset$ for all $i, j \in \{1, 2, \ldots, k, \epsilon\}$ and $i \neq j$.

- The number k is often user-defined.
- Most conventional algorithms do not allow for unassigned objects (noise).

k-means

k-means

Tries to find a grouping $\mathcal{C}$ which minimizes the squared distance of each object to the representative (center) of it's cluster.

$$\arg\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||_2$$

where μ_i is the mean of the objects in C_i and $||x||_2$ is the ℓ^2-norm.

- Simple iterative heuristic.
- Small overhead in terms of space complexity (store centers).
- Time complexity: $O(nkb)$ where b is the number of iterations.
- Assumes spherical shaped clusters and no noise.
- Algorithm requires access to all objects in each iteration. Becomes a problem when data set does not fit into main memory.

Hierarchical Clustering

Agglomerative hierarchical clustering

1. Start with n sets containing one object each.
2. Iteratively combine the two closest sets till there is only one set left. Distance between sets is computed using a user-specified linkage function (single, average, complete link).

- Can find spherical clusters (average and complete link) or arbitrarily shaped clusters (single link).
- Problem: Distance computation takes $O(n^2)$ time and space.

Density-based Clustering

A typical density-based clustering algorithm is DBSCAN [Ester et al., 1996].

DBSCAN

1. Determines for each object if it is in a sufficiently dense area (MinPts withing a radius ϵ)
2. Join neighbors into clusters using the idea of reachability (shared neighbors)

- Like hierarchical clustering with single link, DBSCAN can also find arbitrary shaped clusters.
- DBSCAN can remove noise (low density areas).
- Problem: Time complexity is $O(n\,log(n))$ or $O(n^2)$.

Clustering Large Data

- Iterative extensions of classical algorithms.
- Parallel/distributed computing (e.g., k-means with MapReduce)

- Cheap preprocessing of the data before clustering
  - Sampling (e.g., CLARA [Kaufman and Rousseeuw, 1990])
  - Single pass algorithms like BIRCH [Zhang et al., 1996]
  - Canopy clustering [McCallum et al., 2000]
  - Online component of data stream clustering algorithms.

BIRCH

BIRCH

- BIRCH [Zhang et al., 1996] builds a height balanced tree (also known as a CF tree) to store information about subclusters.
- New objects are either added to an existing subcluster or create a new one.
- After all objects are added to the CF tree, the user is presented with a list of subclusters which can be recluster with another method like k-means.

- Time complexity is $O(n)$
- Space complexity is independent of n and only depends on k', the number of subclusters.
- Subclusters in the CF tree are very similar to micro-clusters in data stream clustering.

Table of Contents

Data Stream Clustering

Data Stream

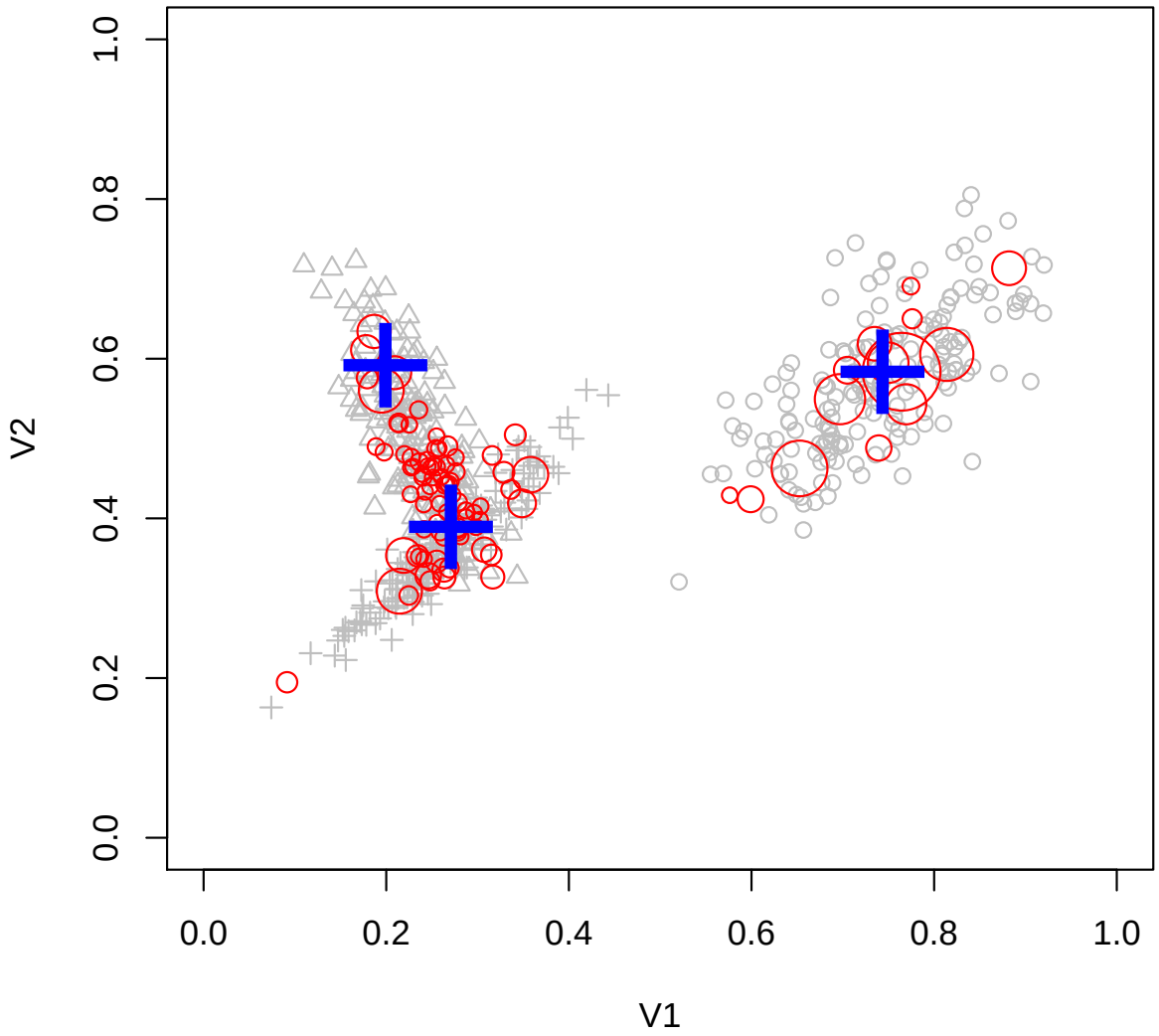
A data stream is an ordered and potentially unbounded sequence of objects $\mathcal{S} = \langle \mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \ldots \rangle$.

Over the last 10 years many algorithms for clustering data streams have been proposed [Guha et al., 2003, Aggarwal et al., 2003, Aggarwal et al., 2004, Cao et al., 2006, Tasoulis et al., 2006, Tasoulis et al., 2007, Udommanetanakit et al., 2007, Tu and Chen, 2009, Wan et al., 2009].

Typical data stream clustering approach

1. **Online:** Summarize the data by a set of k' micro-clusters using a single pass over the data.
2. **Offline:** Recluster the k' micro-clusters into k final (macro-) clusters ($k \ll k'$). Micro-clusters are typically regarded as pseudo-points for a conventional clustering algorithm (e.g., k-means).

Example: Data Stream Clustering



Online: DenStream (micro-clusters: red circles)
Offline: weighted k-means with $k = 3$ (final clusters: blue crosses)

Order Dependency

- Objects in data streams are temporally ordered.
- Data streams are considered to change over time (clusters move, disappear or new clusters form).
- Data stream algorithms forget outdated data (e.g., time-dependent exponentially decaying weight for the influence of objects)

Approach for large data sets with arbitrary order

Established if and how order dependence can be removed or reduced before applying a data stream clustering algorithm to non-streaming data. Examples:

- **ClusStream:** Very large horizon parameter forces the algorithm not to forget micro-clusters and instead merge similar clusters.
- **DenStream:** Decay rate can be set to 0 or close to 0.

Order dependency due to single-pass nature of the algorithm typically only effects micro-cluster placement slightly. Results indicate that after reclustering, the final clustering is only effected minimally.

CluStream

CluStream [Aggarwal et al., 2003] was the first micro-cluster-based algorithm. A micro-cluster is a local summary typically represented by position, dispersion and weight.

CluStream

1. Collect data to create an initial set of k' micro-clusters using k-means.
2. For each additional object: either add the object to a micro-cluster or create a new micro-cluster.
3. If a new micro-cluster was created then delete or merge micro-clusters to keep k' constant.

For a stable clustering with no merging and deletion of micro-clusters: Time complexity of $O(nk')$.

DenStream

DenStream

Similar to CluStream but DenStream [Cao et al., 2006] initializes the micro-cluster list using DBSCAN on an initial set of objects and maintains two lists of micro-clusters.

- **Core-micro-clusters:** Dense clusters (defined by MinPoints)
- **Potential micro-clusters:** May develop into core-micro-clusters or be eventually deleted (noise).

DenStream incurs more overhead compared to CluStream for maintaining the data structure and k' is not held constant.

Table of Contents

The **stream** Package

The R-extension package **stream** provides an infrastructure for data stream mining.

- Define and create synthetic data streams.
- Connect to real data streams (via connections)
- Implement data stream mining algorithms (currently online component of clustering from MOA [Bifet et al., 2010] and native in R)
- "Stream" interface to classical algorithms (k-means, sampling, etc.)
- Implement reclustering (offline component).
- Performance evaluation and support for systematic experimentation.

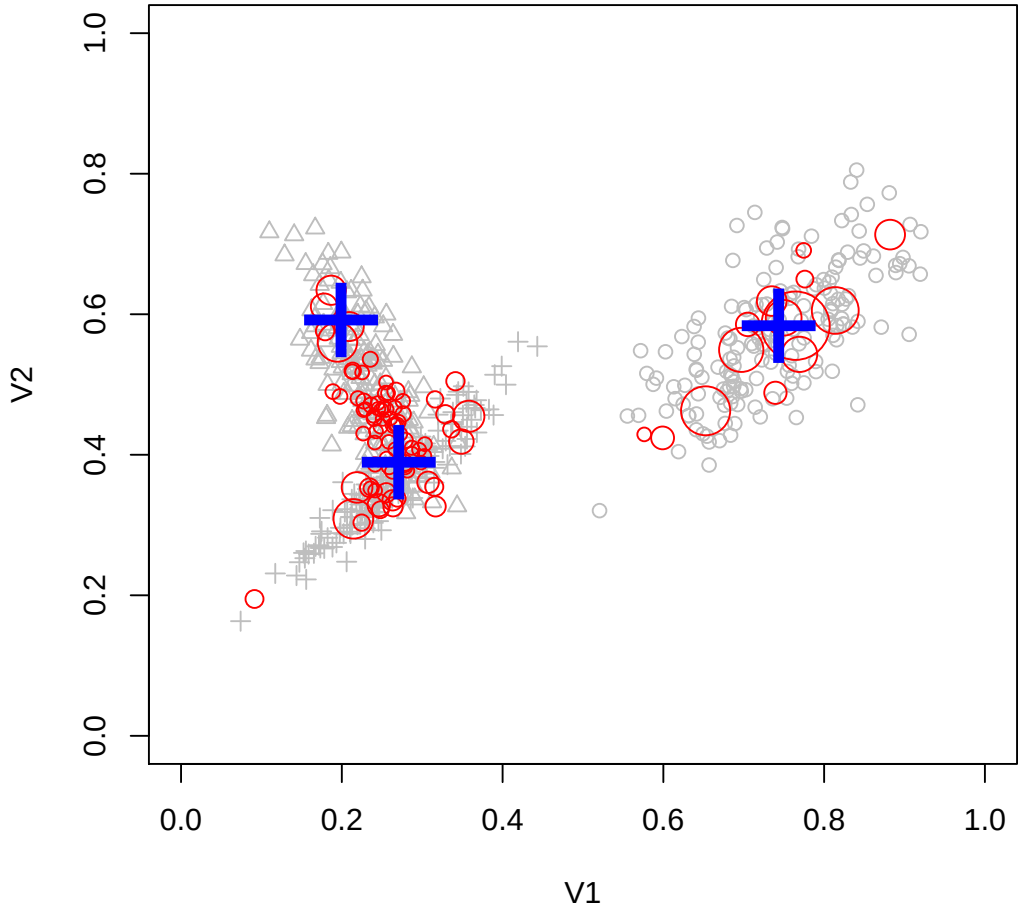
stream is available at from within R or at

`http://cran.R-project.org/web/packages/stream/`

Using the **stream** Package

```
> library(stream)

### define a data stream
> dsd <- DSD_GaussianStatic(k=3, d=2)
> get_points(dsd)
           V1        V2
    1 0.6051603 0.4822357

### cluster using DenStream
> micro <- DSC_DenStream(initPoints=100, horizon=1000)
> cluster(micro, dsd, 1000)

### recluster using weighted k-means
> macro <- DSC_Kmeans(k=3, weighted=TRUE)
> recluster(macro, micro)

### plot results
> plot(micro, dsd=dsd)
> points(get_centers(macro), col="blue", cex=5, lwd=2, pch="+")
```

stream: Example Code Output




```
> evaluate(micro, dsd, n=100)
 recall     precision          f1         ssq        rand numCluster numClasses
 0.1000000  1.0000000  0.1818182  4.5551937  0.7776000 30.0000000  3.0000000
> evaluate(macro, dsd, n=100)
 recall     precision          f1         ssq        rand numCluster numClasses
 0.9876543  0.9912281  0.9894380  7.8401944  0.9874000  3.0000000  3.0000000
```

Table of Contents

Empirical Study

How well do different "large data" preprocessing methods work?

We will compare the quality of pure k-means clustering (baseline) with

1. Reservoir Sampling [Vitter, 1985]
2. BIRCH [Zhang et al., 1996]
3. CluStream [Aggarwal et al., 2003]
4. DenStream [Cao et al., 2006]

   $+$ reclustering using weighted k-means.

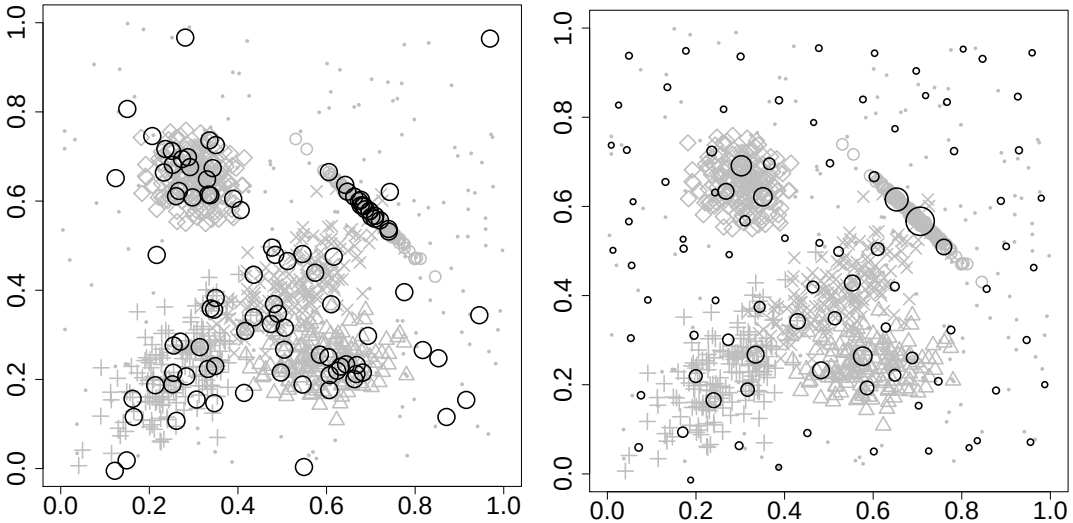
The experiments are performed using the R-extension package **stream**

Data Sets

Dataset	# 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

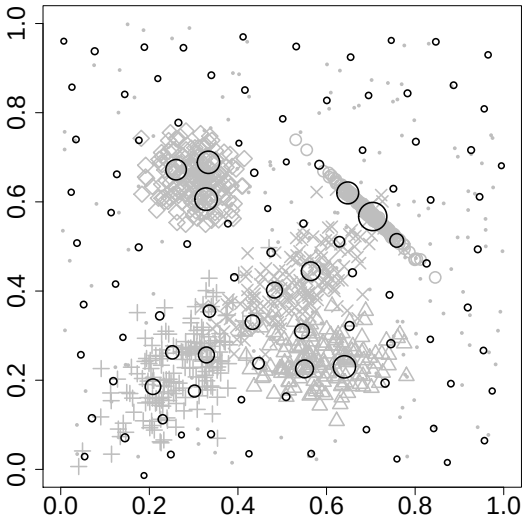
- **Mixture of Gaussians:** k clusters in a d-dimensional hypercube; random centers and covariance matrices [Jain and Dubes, 1988]; uniform noise.
- **Covertype data set:** remote sensing data from the Roosevelt National Forest, CO; 10 quantitative variables; ground truth: 7 different forest cover types. (UCI Machine Learning Repository)
- **16S rRNA data set:** 3-gram count for 400,000+ 16S rRNA sequences (approx. 1500 letters each); ground truth: Phylum annotation. (Greengenes database)

Micro-Cluster Placement on a Mixture of 5 Gaussians

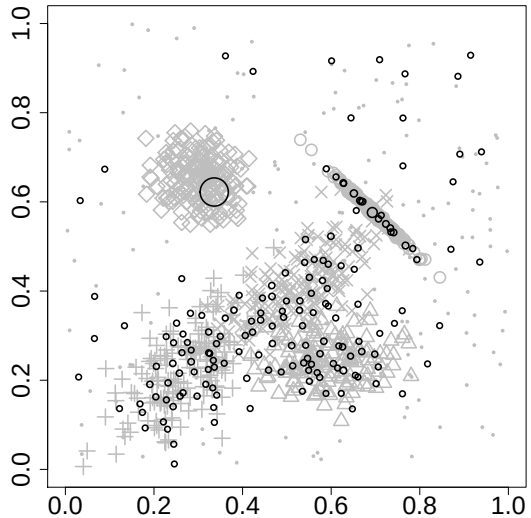


Sampling 1%

BIRCH



CluStream



DenStream

Order Dependence

- Single pass algorithms are order dependent.
- Use Mixture of 5 Gaussians with 10,000 objects.
- Randomly reorder 10 times and cluster.
- Compare the 10 cluster assignments obtained using the corrected Rand index (1 = perfect agreement; corrected for agreement by chance).

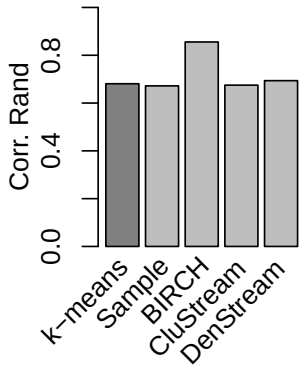
	Sampling (1%)	BIRCH	CluStream	DenStream
Corr. Rand	.74	.85	.81	.79

Findings

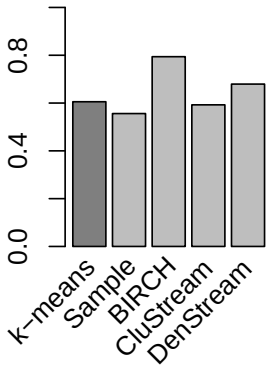
- Random sampling is order independent and used as the baseline.
- Order dependence of other methods is below the variability of a 1% sample.

Cluster Quality

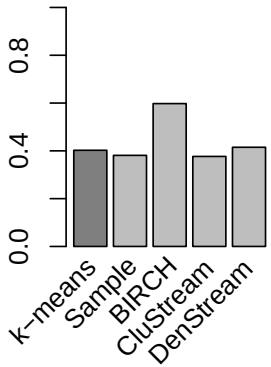
Compare clustering methods ($+k$-means reclustering) with ground truth.



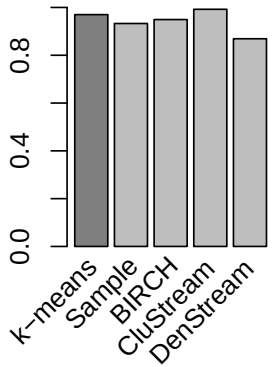
k3d2n00



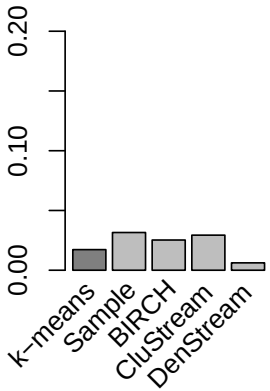
k3d2n20



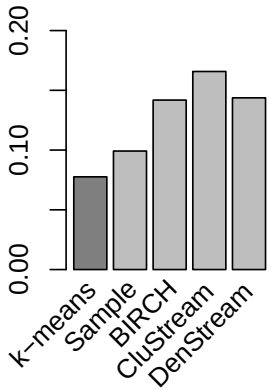
k10d2n00



k10d10n00



Covertype



16S rRNA

Conclusion

- Data stream clustering is fast and scalable (single pass over the data)
- Much work has been done over the last 10 years
- Order dependency is not very strong
- Clustering quality is comparable to k-means and typically better than sampling
- Many algorithms can deal with noisy data

Detailed results will appear in

Matthew Bolaños, John Forrest, and Michael Hahsler. *Clustering large datasets using data stream clustering techniques.* In Studies in Classification, Data Analysis, and Knowledge Organization. Springer-Verlag, 2013.

Thank you!

Visit us at

http://lyle.smu.edu/IDA

References I

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)*, pages 81–92.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004).
A framework for projected clustering of high dimensional data streams.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB '04)*, pages 852–863.

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*, pages 328–339. SIAM.

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

Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (2003).
Clustering data streams: Theory and practice.
IEEE Transactions on Knowledge and Data Engineering, 15(3):515–528.

Jain, A. K. and Dubes, R. C. (1988).
Algorithms for clustering data.
Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

References II

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999).

Data clustering: a review.

ACM Computing Surveys, 31(3):264–323.

Kaufman, L. and Rousseeuw, P. J. (1990).

Finding groups in data: an introduction to cluster analysis.

John Wiley and Sons, New York.

McCallum, A., Nigam, K., and Ungar, L. H. (2000).

Efficient clustering of high-dimensional data sets with application to reference matching.

In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, New York, NY, USA. ACM.

Tasoulis, D., Adams, N., and Hand, D. (2006).

Unsupervised clustering in streaming data.

In *IEEE International Workshop on Mining Evolving and Streaming Data. Sixth IEEE International Conference on Data Mining (ICDM 2006)*, pages 638–642.

Tasoulis, D. K., Ross, G., and Adams, N. M. (2007).

Visualising the cluster structure of data streams.

In *Advances in Intelligent Data Analysis VII*, Lecture Notes in Computer Science, pages 81–92. Springer.

Tu, L. and Chen, Y. (2009).

Stream data clustering based on grid density and attraction.

ACM Transactions on Knowledge Discovery from Data, 3(3):1–27.

References III

Udommanetanakit, K., Rakthanmanon, T., and Waiyamai, K. (2007).

E-stream: Evolution-based technique for stream clustering.

In *ADMA '07: Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, pages 605–615. Springer-Verlag, Berlin, Heidelberg.

Vitter, J. S. (1985).

Random sampling with a reservoir.

ACM Transactions on Mathematical Software, 11(1):37–57.

Wan, L., Ng, W. K., Dang, X. H., Yu, P. S., and Zhang, K. (2009).

Density-based clustering of data streams at multiple resolutions.

ACM Transactions on Knowledge Discovery from Data, 3(3):1–28.

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*, pages 103–114. ACM.