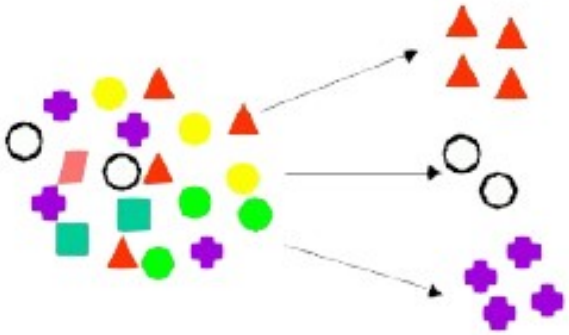


Efficient Evidence Accumulation Clustering for large datasets/big data

Diogo Alexandre Oliveira Silva

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Ana Fred 1 and Helena Aidos 2

Examination Committee

Chairperson:	Professor Full Name
Supervisor:	Professor Full Name 1 (or 2)
Member of the Committee:	Professor Full Name 3

Month 2015

Dedicated to someone special...

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: palavra-chave1, palavra-chave2,...

Abstract

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords...

Keywords: keyword1, keyword2,...

Contents

List of Tables

List of Figures

Glossary

API	Application Programming Interface.
CPU	Central Processing Unit.
EAC	Evidence Accumulation Clustering.
GPGPU	General Purpose computing in Graphics Processing Units.
GPU	Graphics Processing Unit.
HAC	Hierarchical Agglomeration Clustering.
PCA	Principal Component Analysis.
PC	Principal Component.
QK-Means	Quantum K-Means.
Qubit	Quantum bit.
SL-HAC	Single-Linkage Hierarchical Agglomeration Clustering.
SVD	Singular Value Decomposition.

Chapter 1

Clustering

1.1 The problem of clustering

Hundreds of methods for data analysis exist. Many of these methods fall into the realm of machine learning, which is usually divided into 2 major groups: *supervised* and *unsupervised* learning. Supervised learning deals with labeled data, i.e. data for which ground truth is known, and tries to solve the problem of classification. Examples of supervised learning algorithms are Neural Networks, Decision Trees, Linear Regression and Support Vector Machines. Unsupervised learning deals with unlabeled data for which no extra information is known Clustering algorithms, expectation-maximization and Principal Component Analysis are examples of unsupervised algorithms.

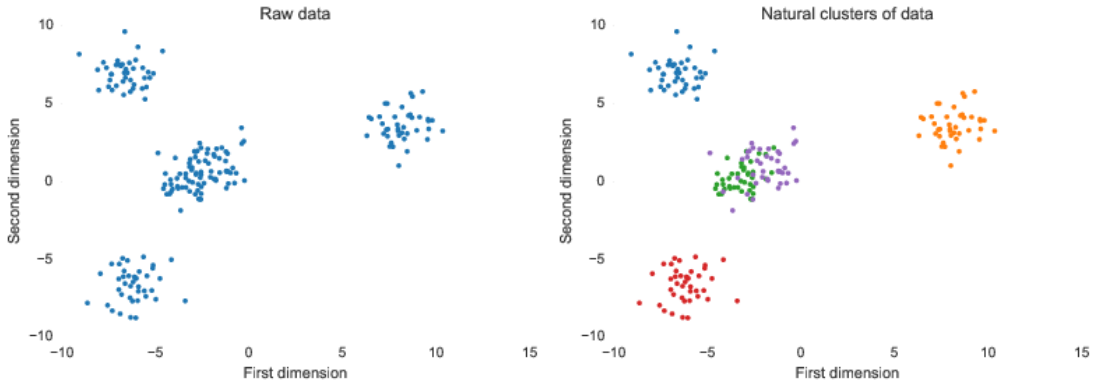
Cluster analysis methods are unsupervised and the backbone of the present work. The goal of data clustering, as defined by [1], is the discovery of the *natural grouping(s)* of a set of patterns, points or objects. In other words, the goal of data clustering is to discover structure on data. The methodology used is to group patterns (usually represented as a vector of measurements or a point in space [2]) based on some similarity, such that patterns belonging to the same cluster are typically more similar to each other than to patterns of other clusters. Clustering is a strictly data-driven method, in contrast with classification techniques which have a training set with the desired labels for a limited collection of patterns. Because there is very little information, as few assumptions as possible should be made about the structure of the data (e.g. number of clusters). And, because clustering typically makes as few assumptions on the data as possible, it is appropriate to use it on exploratory structural analysis of the data. The process of clustering data has three main stages [2]:

- **Pattern representation** refers to the choice of representation of the input data in terms of size, scale and type of features. The input patterns may be fed directly to the algorithms or undergo *feature selection* and/or *feature extraction*. The former is simply the selection of which features of the originally available should be used. The later is deals with the transformation of the original features such that the resulting features will produce more accurate and insightful clusterings, e.g. Principal Component Analysis. It should be noted that
- **Pattern similarity** refers to the definition of a measure for computing the similarity between two

patterns.

- **Grouping** refers to the algorithm that will perform the actual clustering on the dataset with the defined pattern representation, using the appropriate similarity measure.

As an example, Figure 1.1a shows the plot of a simple synthetic dataset - a Gaussian mixture of 5 distributions. No extra information other than the position of the points is given, since clustering algorithms are unsupervised methods. Figure 1.1b presents the desired clustering for this given dataset.



(a) Input data, unlabeled.

(b) Desired labels.

Figure 1.1: Gaussian mixture of 5 distributions. Fig. 1.1a shows the raw input data, i.e. how the algorithms ”see” the data. Fig. 1.1b shows the desired labels for each point, which here means their corresponding Gaussian.

1.2 Definitions and Notation

This section will introduce relevant definitions and notation within the clustering context that will be used throughout the rest of this document.

A *pattern* $\mathbf{x}$ is a single data item and, without loss of generalization, it consists of a set of d *features* x_i that characterize that data item, $\mathbf{x} = (x_1, \ldots, x_d)$, where d is referred to as the dimensionality of the pattern. A *pattern set* (or data set) $\mathcal{X}$ is then the collection of all n patterns $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The number of features is usually the same for all patterns in a given pattern set.

In cluster analysis, the desired clustering, typically, is one that reflects the natural structure of the data, i.e. the original true clustering. In other words, one wants to group the patterns that came from the same state of nature when they were generated, the same *class*. A class, then, can be viewed as a source of patterns and the effort of the clustering algorithm is to group patterns from the same source. Throughout this work, these classes will be also be referred to as the ”natural” or ”true” clusterings. *Hard* clustering (or partitional) techniques assign a class label l_i to each pattern $\mathbf{x}_i$. The whole set of labels of a pattern set $\mathcal{X}$ is given by $\mathcal{L} = l_1, \ldots, l_n$. Closely related to the whole set of labels is the concept of a *partition*, which completely describes a clustering in a different way. A partition P is a collection of k *clusters*. A cluster C is a subset of nc patterns $\mathbf{x}_i$ taken from the pattern set, where the patterns belonging to one subset don’t belong to any other in the same partition. A clustering *ensemble* $\mathbb{P}$ is a

set of partitions from a given pattern set. The relationship between the above concepts is condensed in the following expressions:

$$\mathbb{P} = \left\{P^1, P^2, \dots P^N\right\} \tag{1.1}$$

$$P^j = \left\{C_1^j, C_2^j, \dots C_{k_j}^j\right\} \tag{1.2}$$

$$C_i^j = \left\{x_1, x_2, \dots x_{nc_i^j}\right\} \tag{1.3}$$

Finally, a *similarity measure* is a way of quantifying how similar two patterns are in the feature space, e.g. a metric (such as the euclidean distance), Pearson correlation.

1.3 Characteristics of clustering techniques

Clustering algorithms may be described according to different properties. For the sake of completeness, a small discussion of some properties will be layed out in this section.

Partitional and *hierarchical* algorithms are the two most studied kinds of algorithms [3]. A partitional algorithm is an hard clustering algorithm that will output a partiton where each pattern belongs exclusively to one cluster, e.g. K-Means. A hierarchical algorithm produces a tree-based data structure called *dendrogram*. A dendrogram contains different partitions at different levels which means that the user can easily change the desired number of clusters by simply traversing the different levels. This is an advantage over a partitional algorithm since a user can analyze different partitions with different numbers of clusters without having to rerun the algorithm. Hierarchical algorithms can be further split into two approaches: bottom-up (or *agglomerative*) and top-down (or *divisive*). The former starts with all patterns as distinct clusters and will group them together according to some similarity measure, building the dendrogram from the ground up, e.g. Single-Link, Average-Link. The later will start will all patterns in the same cluster and continuosly split it until all patterns are separated, building the dendrogram from the top level to the bottom, e.g. Principal Directon Divisive Partitioning[4] and Bisecting K-Means [5].

Another characteristic relates to how algorithms use the features for computing similarities. If all features are used simultaniously the algorithm is called *polithetic*, e.g. K-Means. Otherwise, if the features are used sequentially, it's called *monothetic*, e.g. [6].

Constrasting *hard* clustering algorithm are the *fuzzy* algorithms. A fuzzy algorithm will attribute to each pattern a degree of membership to each cluster. A partition can still be extracted from this output by choosing, for each pattern, the cluster with higher degree of membership. An example of a fuzzy algorithm is the Fuzzy C-Means [7].

Another characteristic is an algorithm's stochasticity. A *stochastic* algorithm uses a random search over the feature space at some point in its execution, possibly yielding different results in each run, e.g. K-Means can use a random initialization. A *deterministic* algorithm, on the other hand, will always produce the same result for a given input, e.g. Single-Link.

Finally, the last characteristic discussed is how an algorithm processes the input data. An algorithm

is said to be *incremental* if it processes the input incrementally, i.e. taking part of the data, processing it and then doing the save for the remaining parts, e.g. PEGASUS [8]. A *non-incremental* algorithm, on the other hand, will process the whole input in each run, e.g. K-Means. This discussion is specially relevant when considering large datasets that may not fit in memory or whose processing would take a too long for a single run and is therefore done in parallel.

1.4 K-Means

One of the most famous non-optimal solutions for the problem partitional clustering is the K-Means algorithm [9]. The K-Means algorithm uses K *centroid* representatives c_k for K clusters. Patterns are assigned to a cluster such that that the squared error (or, more accurately, squared similarity measure) between the cluster representatives and the patterns is minimized. In essence, K-Means is solution (although not necessarily an optimal one) to a optimization problem having the Sum of Squared Errors as its objective function, which is known to be a NP hard problem [1]. It can be mathematically demonstrated that the optimal representatives for the clusters are the means of the patterns of each cluster [3]. K-Means, then, minimizes the following expression, where the similarity measure used is the Euclidean distance:

$$\sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - c_k\|^2 \tag{1.4}$$

K-Means needs two initialization parameters: the number of clusters and the centroid initializations. It starts by assigning each pattern to its closer cluster based on the cluster's centroid. This is called the **labeling** step since one usually uses cluster labels for this assignment. The centroids are then recomputed based on this assignment, in the **update** step. The new centroids are the mean of all the patterns belonging to the clusters, hence the name of the algorithm.

The similarity measure used is typically the Euclidean distance. This tends to produce hyperspherical clusters [2]. Still, according to [1], other measures have been used such as the L1 norm, Mahalanobis distance, as well as cosine similarity [3]. The choice of similarity measure must be made carefully as it may not guarantee that the algorithm will converge.

The two steps outlined above (labeling and update) are executed iteratively until a stopping condition is met, usually the number of iterations, a convergence criteria or both. The initial centroids are usually randomly chosen, but other schemes exist to improve the overall accuracy of the algorithm, e.g. K-Means++ [10]. A detail of implementation is what to do with clusters that have no patterns assigned to them. One approach to this situation is to drop the empty clusters in further iterations. However, allowing the existence of empty clusters or dropping empty clusters is undesirable since the number of clusters is an input parameter and it's expected that the output contains the specified number of clusters. Other approaches exist dealing with this problem, such as equaling the centroid of an empty cluster to the pattern furthest away from its assigned centroid or reusing the old centroids as in [11]. After the first

iteration, the first step is executed with the new centroids. The end result of the algorithm is the labels produced in the first step of the last iteration.

K-Means is one of the earliest clustering algorithms to offer a partitional solution to the clustering problem

explain K-means algorithm

As an example, the output of the K-means algorithm to the data presented in Fig. 1.1 is represented in Fig. 1.2. The algorithm executed with 4 random centroids.

Figure 1.2: The output labels of the K-Means algorithm with the number of clusters (input parameter) set to 4.

The number of clusters was purposefully set to an ”incorrect” number to demonstrate that it is not trivial to discover, even in such a simple example. In this synthetic dataset, the number of clusters is not clear due to the two superimposed Gaussians. When no prior information about the dataset is given, the number of clusters can be hard to discover. This is why, when available, a domain expert may provide valuable insight on tuning the initialization parameters.

K-Means is a simple algorithm with reduced complexity $O(ntk)$, where k is the number of clusters and t is the number of iterations that it executes. Because of this, K-Means is often used as a basis for It's also widely used due to it's simplicity and efficiency. Because of this, it's also often used as a foundational step of more complex and robust algorithms, such as EAC. the algorithm works as follows

1.5 Single-Link

Do the same thing as K-Means but for SL

Bibliography

[1] Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. ISSN 01678655. doi: 10.1016/j.patrec.2009.09.011. URL http://dx.doi.org/10.1016/j.patrec.2009.09.011.

[2] a. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3): 264–323, 1999. ISSN 03600300. doi: 10.1145/331499.331504.

[3] Charu C Aggarwal and Chandan K Reddy. *Data clustering algorithms and applications*. ISBN 9781466558229.

[4] Daniel Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998. ISSN 1384-5810. doi: 10.1023/A:1009740529316. URL http://dx.doi.org/10.1023/A:1009740529316$\delimiter"026E30F$nhttp://www.springerlink.com/content/w15313n737603612/.

[5] Michael Steinbach, George Karypis, Vipin Kumar, and Others. A comparison of document clustering techniques. *KDD workshop on text mining*, 400(1):525–526, 2000. doi: 10.1.1.125.9225.

[6] Marie Chavent. A monothetic clustering method. *Pattern Recognition Letters*, 19(11):989–996, 1998. ISSN 01678655. doi: 10.1016/S0167-8655(98)00087-7.

[7] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy ¡i¿c¡/i¿-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191–203, 1984. ISSN 0098-3004. doi: 10.1016/0098-3004(84)90020-7. URL http://www.sciencedirect.com/science/article/pii/0098300484900207.

[8] U Kang, Charalampos E Tsourakakis, and Christos Faloutsos. PEGASUS : Mining Peta-Scale Graphs. *Knowledge and Information Systems*, 27(2):303–325, 2011.

[9] J B MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press*, volume 1, pages 281–297, 1967.

[10] D. Arthur, D. Arthur, S. Vassilvitskii, and S. Vassilvitskii. k-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 8:

1027–1035, 2007. doi: 10.1145/1283383.1283494. URL http://portal.acm.org/citation.cfm?id=1283494.

[11] Malay K Pakhira. A Modified k -means Algorithm to Avoid Empty Clusters. *International Journal of Recent Trends in Enineering*, 1(1), 2009.

Appendix A

Vector calculus

In case an appendix if deemed necessary, the document cannot exceed a total of 100 pages...

Some definitions and vector identities are listed in the section below.

A.1 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.1}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.2}$$