TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ÁNGEL POC LÓPEZ

CLUSTERING ALGORITHMS FOR HIGH-DIMENSIONAL DATA

Bachelor of Science Thesis

Examiner: prof. Tapio Elomaa
Examiner and topic approved on 16 January 2018

ABSTRACT

ÁNGEL POC LÓPEZ: Clustering Algorithms for High-Dimensional Data
Tampere University of technology
Bachelor of Science Thesis, 22 pages
May 2018
Bachelor's Degree Programme in Information Technology
Major: Computer Science (University of Zaragoza)
Examiner: Tapio Elomaa

Keywords: cluster analysis, high-dimensional data, curse of the dimensionality

More and more data are produced every day. Some clustering techniques have been developed to automatically process this data, however, when this data is characteristically high-dimensional, conventional algorithms do not perform well. In this thesis, problems related to the *curse of the dimensionality* are discussed, as well as some algorithms to approach the problem. Finally, some empirical tests have been run to check the behavior of such approaches. Most algorithms do not really cope well with high-dimensional data. DBSCAN, some of its derivations, and surprisingly k-means, seem to be the best approaches.

PREFACE

First, I would like to thank my supervisor Tapio Elomaa for guiding me throughout this process and for being always available for having an appointment and commenting the process and the results of the thesis. Posteriorly, I would also like to thank professor Javier Fabra for making all the Erasmus paperwork between universities easier and for being always careful with me when having any doubt about the process. Last, I would like to thank my Erasmus friends, my friends from the university and from the school and above all my family for being always as supportive as they are.

Tampere, 15.5.2018

Ángel Poc López

CONTENTS

LIST OF TABLES AND FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

4C	Computing Correlation Connected Clusters algorithm
CASH	Clustering in Arbitrarily Oriented Subspaces based on the Hough transform algorithm
CLIQUE	Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications algorithm
COPAC	COrrelation PArtition Clustering
DBSCAN	Density-Based Spatial Clustering of Applications with Noise algorithm
DOC	Density-based Optimal projective Clustering algorithm
DISH	Detecting Subspace cluster Hierarchies algorithm
ERiC	Exploring Relationships among Correlation clusters algorithm
ORCLUS	arbitrarily ORiented projected CLUSte generation algorithm
PCA	Principal Component Analysis
P3C	Projected Clustering via Cluster Cores algorithm
PreDeCon	subspace PREference weighted DEnsity CONnected clustering
PROCLUS	fast algorithm for PROjected CLUStering
SUBCLU	density-connected SUBspace CLUstering

k	number of clusters or the k-th cluster
k-means	algorithm for data clustering
k-NN	k Nearest Neighbors
x	data sample, depending on the index it can make reference to a specific dimension of such sample

1. INTRODUCTION

According to Jain [1], we can define *cluster analysis* as the compilation of methods and techniques for grouping or clustering data in consonance with the mathematical characteristics or the patterns of the data, which can show underlying relations. In machine learning, supervised learning and unsupervised learning can be differentiated. In the former one, data is labelled according to how do we want the data to be trained. In the latter one, where cluster analysis belongs, there is no previous knowledge of how the data must be labelled.

People nowadays produce increasingly data. Scientists are able to run experiments that create such an amount of information that they cannot actually process it manually, nor even extract some conclusions at a glance. In the Internet era, society creates data at every step they take in a web site or every time they share something in a social network. The total amount of data in the world was 4.4 zettabytes in 2013, in 2014 we produced 2.5 exabytes of data every day and it is expected that the amount of data will rise up to 44 zettabytes in 2020 [2].

It is important to analyze the data because it may hide some important characteristics that can be useful for further research. In biology, clustering has been widely used to extract information about *microarray data*, also known as *gene expression data*. Depending on the way the microarray data is clustered, the result can be either genes with common functions or different genetic relationships or disorders. Another important application everybody uses in their day-to-day is making use of customer recommendation systems like those that can be found in online shops or online TV platforms. It should be noted that companies spend a lot of money to get to know the different groups of people that buy their products and so they can address each of those groups individually. Again, the technique used to distinguish the different customers is typically data clustering. but those applications are only some examples among the wide variety that can be found in real life. In 2007 there were 1660 entries published in Google Scholar including the word *data clustering* [1].

This thesis aims to unify the existing information about data clustering for high dimensional data and to clarify the different approaches there exist to this day to solve a problem with the mentioned characteristics. There will also be some empirical tests to compare the performance of such approaches. This work does not intend to be another paper on k-means but to be a brief document to learn to how handle the data clustering problem in general and to quickly help the user, in order to find a suitable method to solve his

problem. This thesis is going to take as main reference work an article about data clustering of Kriegel, Kröger, and Zimek [3].

In the following section a context for cluster analysis is going to be created, as well as some definitions and notation needed to understand the thesis. In Section 3, some basic algorithms are going to be described because they will be further exploited in Section 4, as well as some approaches that lie out of the classification pursued classification. In Section 4, a classification for clustering algorithms is presented, giving some descriptions and also some sample algorithms. Section 5 follows with a set of experimental tests on some selected algorithms. Finally, Section 6 explains the conclusions at which the thesis has come up with.

2. BACKGROUND

Machine learning is a branch of computer science and artificial intelligence that attempts to teach computers how to learn a series of behaviors. It is used to predict a set of outputs given some pre-arranged inputs or *features* of interest. The outputs can be widely different depending on the objective of the desired application, from an integer or a set of integers to a Boolean value explaining the membership to a class or different ways to group some data.

We can mainly distinguish two classes of machine learning: *supervised learning* and *unsupervised learning*. Supervised learning consists on developing a function that maps the inputs to some desired output given a certain known label that discriminates the samples. In unsupervised learning the purpose is to find out or learn some properties of the data from the inputs without having a predefined label that separates them.

Data *clustering* or *cluster analysis* is a special case of unsupervised learning in which samples are classified in different groups given certain properties of their features. Depending on the technique used, groups can overlap each other or not.

A classical method used in cluster analysis is *k-means* [1]. This method divides the data into k clusters depending on the closeness of the data point to a given set of means. Algorithm *k-means* is well-known, but it shows a poor performance in more complicated scenarios as high-dimensional data clustering. In those cases, conventional distance measures are not useful anymore and there may also be some redundant or noisy features that cannot be manually extracted. In addition, we should keep in mind that the number of samples needed to represent the different cluster distributions grows with the number of dimensions [4]. To overcome the last problem, we can use *feature selection* and *dimensionality reduction* techniques. In these techniques we use methods to automatically analyze the different *dimensions* of the data and consequently select or stimulate some features and remove some other ones introducing redundancies or, directly, noise.

Following, some notation is going to be presented to be used during the rest of the article. A database, $\mathcal{D}$, is a set of data points (samples), which are points accommodated in a d-dimesional space and is presented as $< x_1, x_2, \dots, x_{d-1,}\, x_d >$. A feature is any of the characteristics x_i of a data point, that defines the point in some way. It must be said that x_i can also be used to denote any of the n samples of database, and x_{ij} will describe the j-th feature of the i-th data point of such database.

This work focuses on the clustering problem when there is high-dimensional data, which means that we do not only have a few features, but we are working in the scale of dozens, hundreds or even thousands of dimensions. Whenever a k appears on the text it will be

associated either with the final number of clusters after processing the data or with a specific cluster.

One of the most important things in data clustering is the selection of the distance measures, they will always have to be used either directly to the data points or after some transformations, to measure similarities. The most important one, because of its simplicity and its widely accepted use, is the *Euclidean distance*, which is the distance in a straight line connecting two points. To describe the Euclidean distance, we must present the *Euclidean norm* or ℓ^2 *norm*, which indicates the length of a vector in a d-dimensional space:

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_{d-1}^2 + x_d^2}\,.$$

Therefore, if we want to measure the distance between the samples x_1 and x_2, i.e., calculate the Euclidean distance, we must subtract the two vectors describing the points, x_1 and x_2:

$$d(x_1\,, x_2) = \sqrt{\sum_{i=1}^{d} (x_{1i-} x_{2i})^2}.$$

However, that is not the only distance function that can be used, we can also use the *Manhattan distance*, based on the *Manhattan norm* or ℓ^1 *norm*, which only measures distances on axis-parallel directions, that is, not following a straight line between points as above.

This distance measures are only useful if there is only numerical data present in the datasets. However, there can also be *nominal* or *boolean* attributes as well as *strings* or images. In the case of nominal attributes some transformations should be made, or some specific measure considering that all possible values for the variables are equidistant. When working with strings some edit distance measures can be used, in a similar way dissimilarities on DNA strings are calculated. If some images must be treated, point to point differences could be used or some hash function if it is desired to reduce the computational time.

The choice of these different distance measures mentioned before or another desired one in the spectrum of the existing ones, must be carefully studied, because depending on the distance function used, the shape of the final clusters may vary [3] [5].

Finally, a couple of concepts related with the k-means like algorithms are going to be described. A *medoid* is a similar concept to the mean, being the closest point in a dataset to the mean, whereas a *centroid* is the point in the d-dimensional space that represents the mean, and do not necessarily has to be an existing point in the database.

3. RELATED WORK

In this section some classic approaches to the cluster analysis problem are going to be introduced as well as some innovative approaches that had never been used in our field of interest and that seem to perform well specifically in high-dimensional data clustering (nor in the case of low dimensionality problem).

3.1 *K*-means algorithm

This method [1] divides the data in k clusters depending on the closeness of the data point to a given set of means, it tries to minimize the squared error between the samples and each cluster mean.

Let μ_k be the mean of the k-th cluster and the points in such cluster be c_k, then, the squared error is

$$J(c_k) = \sum_{x_i \in c_k} \|x_i - \mu_k\|^2,$$

and the goal is to minimize the global squared error, $J(C)$, over all the clusters

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2.$$

This problem is known to be an *NP-hard* problem and therefore the algorithm used to optimize the function follows a greedy algorithm approach, which implies that the final solution might just be a local minimum.

There are three steps in this algorithm:

1. Initialize the k means randomly.
2. Assign each sample to the closest mean, i.e. cluster, using a given distance function.
3. Update each cluster mean given the new assigned samples.

One should iteratively repeat steps 2 and 3 until the clusters remain stable.

As a distance function is used, there may problems in high dimensional data sets because the distance between samples turns to be less distinguishable as the number of features increases [4].

3.2 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN [5] is based on the notion that points within a cluster have higher density than either points outside them or points in noisy areas, as human eye can perceive from Figure 1. Each point, member of a cluster, must have a certain number of points within a specific radius. The selection of the distance function between points is of highly importance because it will determine the shape of the cluster. A very common choice in cluster analysis is the *Euclidean Distance*, because of its simplicity. The method defines a series of properties that must be met to create a cluster.

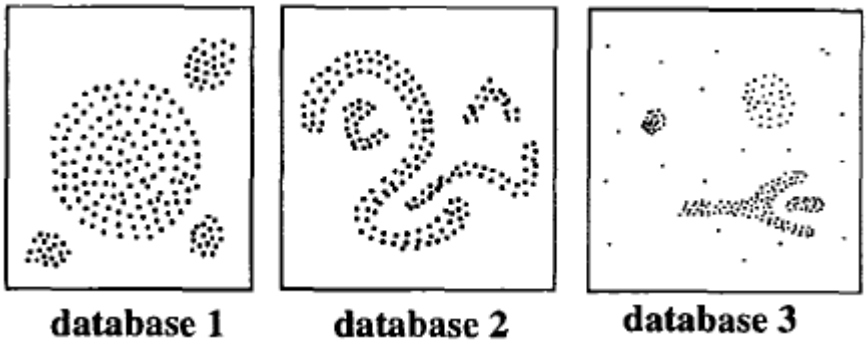

Figure 1: *Sample databases for Density Based Clustering [3].*

A direct approach would require having minimum number of points (*MinPts*) in the Eps-neighborhood, $N_{Eps}(p)$, of certain point p. The Eps-neighborhood of a point is the set of points inside the circumference described from the sample with radius Eps. This approach fails because there are two types of points within a cluster, the core points (inside the cluster) and border points (at the frontier of the cluster). Using this approach on border points would certainly fail to correctly cluster the samples. Because of that we are going to describe some other properties of density-based systems.

A point p is *directly density-reachable* from q if $N_{Eps}(q)$ contains p and the number of Eps-neighbors of q is greater than *MinPts.* In addition, a point p is *density-reachable* from q if those points are joined by a chain of directly density-reachable points. Moreover, a point p is *density-connected* to q if there is a point o from which both p and q are density-reachable. Given these definitions, we can finally define a cluster as a subset of points of a database meeting the following conditions:

1. For all points p and q, if p is a member of a cluster C and q is density-reachable from p, then q belongs to C.
2. All points within a cluster are density-reachable to each other.

We can implement an algorithm to create clusters meeting the previous conditions, in any case, in high-dimensional situations, it is difficult to see and distinguish low density

regions from the rest due to the data sparsity. Some approaches focused on the high-dimensional problem are based on this DBSCAN clustering system as we will show later.

3.3 Hubness application to Cluster Analysis

Hubness is a phenomenon that had been observed in high dimensional data, but nobody had tried to take profit of it applying it in the cluster analysis problem. Instead of trying to reduce dimensionality, we will take advantage of some characteristics of high-dimensional data spaces.

Hubness is the propensity of some data points in high-dimensional data spaces to appear with higher frequency in *k-nearest neighbors (k-NN) lists* of points than any other samples of the dataset [4]. K-nearest neighbor lists have been previously used to make k-NN graphs, in which we can use graph clustering, or in density-based clustering as shown before. Hubness is an implicit property of the spaces we are dealing with; therefore it cannot be applied to low-dimensional datasets.

Data are lying on various hyperspheres centered at the distribution mean and, in high-dimensional data, the variance is low. This implies in a practical case that, points closer to the mean will be closer to all other points, and therefore, they will appear more frequently in more k-NN lists. Elements with low hubness are probably further from the rest and will be outliers. It should also be kept in mind that some hubness points might be near two or more different clusters.

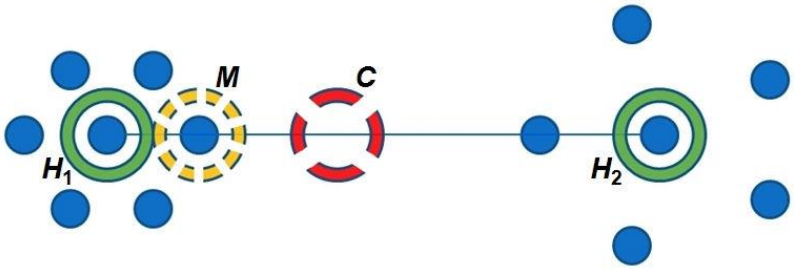

Figure 2: *Graphical example. C marks the centroid, M the medoid and the green circles mark points with high hubness [4].*

Then, after seeing how hubness points behave, we could think that they are the *medoids* of the cluster or perhaps that they are a good approximation to those as we can see in Figure 2. Experiments have demonstrated, indeed, that some hubness points are medoids. So, this approach is based on looking for the hubness points to use them as data center approximations.

The algorithm proposed by Tomasev et al. [4] frequently shows an improvement over some *centroid-based* approaches on high-dimensional datasets, on both synthetic and real data, also behaving well under noisy conditions.

4. ALGORITHMS FOR HIGH-DIMENSIONAL DATA

This section is going to get deeper in the clustering problem, distinguishing different types of clustering problems for high-dimensional data, and different ways to address them.

As we have commented before, high-dimensional data clustering has some intrinsic problems, such as *local feature relevance* or *local feature correlation*¸ meaning that different subsets of the features or different correlations are more important for different clusters, respectively.

Feature selection techniques are a way to address the high-dimensional problem, however, techniques like *Principal Component Analysis* (PCA) only choose one subspace in which the clustering will be subsequently done and that counterposes some principles discussed above.

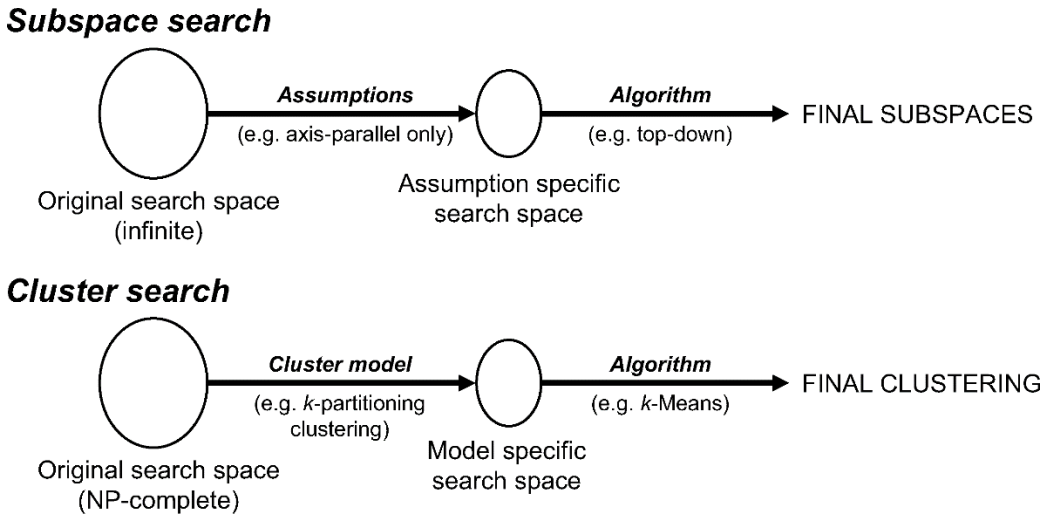

Figure 3: *Problems to be solved in high-dimensional data clustering [3].*

In the cluster analysis problem we have to solve both cluster and subspace searches at the same time, we can have a visual reference in Figure 3. Both problems have a huge search space (in the case of subspace searching it can be infinite) and some heuristics must be developed to reduce the time consumption. As a consequence of the application of heuristics, the solution can be non-optimal. Naturally, we discard the brute force approach of calculating each possible subspace projection because the number of subspaces can be infinite.

Three types of cluster searching approaches are going to be discussed: finding clusters in parallel-axis subspaces, pattern recognition approaches and finally finding clusters in

arbitrarily oriented subspaces. A great portion of the knowledge for this part has been acquired from Kriegel et al. survey [3].

4.1 Axis-Parallel Subspaces

One simple idea when working with high-dimensional data is to minimize the problem to axis-parallel subspaces. In a dataset with d dimensions we can pass from infinite subspaces to $2^d - 1$ possible subspaces, nevertheless, with the size of d that is being dealt in this thesis some heuristics have to be applied.

We can categorize the algorithms following two ways, on the one hand, depending on how the algorithm is going to be implemented, i.e. bottom-up or top-down approach, or on the other hand, based on the presumptions or simplifications made to the problem.

Top-down algorithms start from the full-dimensional space and try to refine and determine the features that constitute each subspace, and that after projecting will create a cluster. There are some constraints related to the problem, and to break them, researchers tend to use the *locality assumption*, that leads to the assumption that the subspace that stablishes a cluster can be learned from the local neighbors of the members.

Bottom-up algorithms start from all one-dimensional subspaces, and most of them evolve relying on the reverse of the *monotonicity property*: *if a space T does not contain a cluster, then no space containing T will contain a cluster*.

Now, according to the other classification of algorithms, the first subset to be analyzed is the *Projected clustering algorithms*. These algorithms attempt to bind uniquely each data sample to a cluster, they usually work as normal clustering algorithms, but the difference resides in a specialized distance function that depicts the subspaces of the clusters, finding the projection to be made. These algorithms rely on the fact that some data points are close when they are projected using certain subset of the features, but they are not close anymore when they are projected using the rest of them. *Projected clustering algorithms* can use full dimensional, partitional, hierarchical and density-based techniques [6]. PROCLUS [7] is a k-medoid algorithm like the one discussed in Section 3, the k-means algorithm. It initializes M medoids, of which it will choose k to start working with. The current medoids are refined minimizing the standard deviation of the distance from the medoids to the neighbors of such medoid, after that, the samples are adjudicated to the medoids having in mind the important features of each medoid, i.e., the subspace. If some of the current medoids can be replaced by one of the medoids left improving the results, it will be changed. After optimizing the system noisy samples are detected and taken out of the clusters. There are some variations of this algorithm. PreDeCon [8] is based in the previously mentioned DBSCAN, however, in this case it is using a specialized function to detect the desired subspaces. As in DBSCAN it requires a number of parameters which are difficult to guess.

A second class of algorithms or a subclass of the previous one is the "*soft" projected clustering algorithms*, which assume that the number of clusters is already known and the only thing to be made is to optimize a function in which all dimensions play a role and none of them is discarded.

To continue, we present the *subspace clustering algorithms.* We must distinguish between this subset and the projected clustering algorithms because in the clustering literature there has frequently been some confusions or misconceptions. Subspace clustering algorithms aim to find all clusters in all possible subspaces. This is a problem since the aforementioned dimensionality. They start with all the one-dimensional clusters and keep on merging each other, all of the approaches are developed via bottom-up algorithms. CLIQUE [9] is the first approach proposed in subspace clustering and it divides the space in units of the same size, ξ, along all dimensions. Only the partitions that include a minimum number of τ points will be considered as dense, and therefore, a cluster will be that a group of adjacent dense partitions. SUBCLU [10], in contrast to CLIQUE, can find clusters in arbitrary shapes (in the relevant subspace, not the full-dimensional one). It is based in the density-connection principle of DBSCAN but in addition it adds some more definitions to apply restrictions on the subspace search. It outputs the same clusters than applying DBSCAN to those resulting subspaces.

Finally, we have the hybrid algorithms, which neither try to create a unique bind between a point and a cluster nor try to find all clusters in all subspaces, indeed, they sometimes produce overlapping clusters. Other algorithms just focus on interesting subspaces. DOC [11] is a density-based approach relying on hypercubes of a given size, that uses a Monte Carlo algorithm to approximate the clusters. Each time DOC is called, it will probably output only one cluster, so it has to be iteratively called several times. If the points of the previous clusters are not excluded, DOC clusters may overlap. It does not usually compute all clusters in all subspaces as subspace clustering proposed. DiSH [12] is able to discover the hierarchies between subspaces via hierarchical subspace clustering, and it can detect clusters of different dimensions, size and density. Finally, P3C [13] follows a bottom-up strategy starting with all one-dimensional intervals, which are a restriction on the values of the attributes, and reduces the problem to an Expectation-Maximization problem used to refine the clusters. It ends-up with a matrix containing the probabilities for each data sample to belong to any of the clusters.

4.2 Pattern Recognition Approaches

Pattern recognition algorithms follow a different approach, if closeness has been defined until now as the Euclidean distance after projecting the samples, now closeness will be related with similar behavior in axis-parallel subspaces. It has also been noticed that rows were mainly used for points and columns for features, but in pattern recognition approaches you can generally transpose the data without affecting the output. Most of the

work for the so called *biclustering* algorithms, which are going to be presented later, has been done by Madeira and Oliveira [14] focusing on the application for biological data.

Let A be a matrix with a set of rows X and a set of columns Y, respectively. Now, if we talk about the element a_{xy}, we will be talking about an element of the matrix correspondent to the one placed at the row x and column y. On the other side, if we talk about the A_{IJ}, we are defining a subset of A in which $I \subseteq X$ and $J \subseteq Y$. Biclustering algorithms seek to find subsets of a matrix, which will be called from now on biclusters, that follow a given set of properties. These algorithms are supported by some statistical definitions like the mean. The mean of a row can be calculated as follows:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij} \, .$$

The mean of a column is analogously:

$$a_{Ij} = \frac{1}{|I|} \sum_{j \in J} a_{ij} \, .$$

And finally, the mean of a bicluster is:

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} \, .$$

We can distinguish among four different types of biclusters: *constant biclusters, biclusters with constant values in rows or columns, biclusters* with *coherent values* and finally *biclusters* with *coherent evolutions.*

Constant biclusters are those biclusters whose values are perfectly identical. If the problem is relaxed it can be said that the values are similar instead of identical. It is a clear example of an axis-parallel case. *Block clustering* [15] is an algorithm that solves the given problem. If the variance of every bicluster A_{IJ} is calculated:

$$VAR(A_{IJ}) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ}),$$

the perfect bicluster is the one with variance equal to zero and hence we can split the matrix into two partitions minimizing the variance of the obtained biclusters and iteratively repeating this process of splitting until k clusters are obtained.

Biclusters with constant values on rows are another kind of categorization in which values of rows or columns follow a certain pattern. In the case of constant rows, they can be calculated with the following formula $a_{ij} = \mu \cdot r_i$, where μ is a constant value for the

bicluster which is modified by r_i in each row and the dot means addition or multiplication. It must be cleared out that the multiplicative model is equivalent to the additive one when taking logarithms. In the case of constant columns, it can be calculated analogously as $a_{ij} = \mu \cdot c_i$, where again c_i is a variable dependent to the column. The most common approach is to apply a transformation to reduce the problem to a *constant biclusters* one, and therefore apply *block clustering* directly there.

Biclusters with coherent values present a real step-forward with respect to the biclusters previously mentioned because they present coherent values on both rows and columns at the same time. We can define them combining both the equations that characterized biclusters with constant values on rows and constant values on columns:

$$a_{ij} = \mu \cdot r_i \cdot c_i \,.$$

It has to be reminded that patterns with negative correlation cannot have coherent values. Cheng and Church [16] proposed a similarity score, the *mean squared residue*, H, to create a model to define biclusters. In that model, a submatrix A_{IJ} is a δ-bicluster if its similarity measure is lower than δ. We consider that a bicluster is perfect if δ is equal to zero, and in that conditions, an element of the bicluster is defined as:

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ},$$

which agrees perfectly with the previous equation defining coherent biclusters.

If we define H as:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}) \,.$$

The objective is to simultaneously calculate all the biclusters, minimizing the following function:

$$\sum_{I,J} H(I,J),$$

that is, to minimize the residuals given by the similarity measure. However, this algorithm has some limitations like the obligation to indicate the number of clusters before running the algorithm, and some inefficiencies due to the fact that some areas must be masked in order to continue processing the rest of biclusters. FLOC [17] is an algorithm based in δ-biclusters, but in this case some random movements are introduced to add or remove some rows or columns to some biclusters, allowing, also, overlapping between clusters.

Biclusters with coherent evolutions differ from the rest of the aforementioned biclusters in that the latter detect changes between pairs of rows or columns no matter the quantity that it is changing, just the fact that it follows a common pattern. Ben-Dor et al. [18] define the biclustering problem as an *order-preserving submatrix* (OPSM) problem, in which the idea is to find a permutation of the columns of a matrix so that the values of the rows of a given submatrix are ordered in a strictly increasing order. Other algorithms like the one proposed by Liu and Wang [19] follow the same approach but in their case some elements are allowed to have similar values.

The results of this pattern-based categorization might be interesting sometimes, but these results do not usually have a spatial interpretation, like clustered points lying on the same subspace.

4.3 Arbitrarily-Oriented Subspaces

Previous clustering systems had some limitations like the obvious one of the parallel-axis subspaces or that pattern matching methods were so simplistic and only some types of correlations were allowed. Now we are going to present some other type of algorithms, which we will call *oriented clustering*, *generalized subspace/projected clustering* or *correlation clustering* algorithms, which do not have any constraint on the location or orientation of the clusters, i.e., they can be on any subspace on $\mathbb{R}^d$, and here, clusters become hyperplanes of different dimensionalities, where pattern recognition approaches do not suit anymore.

Points lying on a common hyperplane seem to follow certain linear dependencies or correlations among the dimensions of such plane. If a density-based algorithm was used, some points may be clustered in an interesting subspace, but if correlations are introduced into the problem, it may be noticed that those points do not really belong to the same cluster and hence, we should devise a new approach.

A good technique to seek for relevant directions of high variance in any direction is Principal Component Analysis (PCA). PCA is applied locally to discover clusters in different subspaces, assuming that the hyperplane in which the points lie is well defined by a local set of points.

To apply PCA the covariance matrix of the dataset $\mathcal{D} \subset \mathbb{R}^d$ has to be built first:

$$\Sigma_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \cdot \sum_{x \in \mathcal{D}} (x - x_{\mathcal{D}}) \cdot (x - x_{\mathcal{D}})^T .$$

In the resultant matrix of size $d \times d$ each element σ_{ij} of $\Sigma_{\mathcal{D}}$ represents the covariance between the dimensions i and j and the diagonal element represents the variance of such dimension. It must be said that there is a direct relation between covariance and

correlation, being explained by: $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$, where ρ_{ij} is the correlation between the dimensions i and j. Now, eigenvalue decomposition can be applied to $\Sigma_{\mathcal{D}}$ to obtain

$$\Sigma_{\mathcal{D}} = V_{\mathcal{D}} E_{\mathcal{D}} V_{\mathcal{D}}^T,$$

where $E_{\mathcal{D}}$ is the *eigenvalues* matrix of $\Sigma_{\mathcal{D}}$, containing its *eigenvalues* in decreasing order, and $V_{\mathcal{D}}$ is the *eigenvectors* matrix, representing a new orthonormal base, where the vectors are ordered corresponding to its eigenvalues. $E_{\mathcal{D}}$ can also be understood as the covariance matrix of the system using the new basis, in such system, there are no correlations among dimensions anymore, consequence of all non-diagonal elements being equal to zero. While the first λ eigenvectors (strong eigenvectors) create a hyperplane where the $\mathcal{D}$ points are accommodated, the last $d - \lambda$ eigenvectors (the weak eigenvectors) create a subspace where the points cluster densely when projected.

This information can be used to make algorithms for data clustering, there are some different approaches, differing in the selection of λ or the local points. ORCLUS [20] follows a similar path to PROCLUS [7], also initializing $M(> k)$ seeds, it uses a specialized distance function that measures distances among points when projected through the weak eigenvectors. The number of seeds is iteratively reduced until reaching the selected k by merging the most similar (closest on the projected subspace) clusters. There is a check and balance between the choice of M and the runtime, a greater M will lead to a better performance but it will consequently increase the execution time. On the other hand, 4C [21] follows a density-based approach like DBSCAN [5], some seeds are iteratively initialized and they evolve meeting some density-based properties based on distance functions created applying PCA to the *Eps-neighborhood* of the seed. COPAC [22] follows related ideas to 4C, but in its case it clusters points with the same number of strong eigenvectors increasing efficiency. ERiC [23] follows a similar path to COPAC with the addition of a subspaces hierarchy.

However, not all the approaches of *generalized subspace clustering* are attached to PCA. There are other techniques based on the Hough Transform as CASH [24]. The Hough Transform is a highly used transform in digital image processing that maps points from the spatial domain to the parameter domain, one of its applications is line fitting in object recognition problems. Each point in the spatial domain is mapped into an infinite set of points in the parameter domain through a line, for example, or a trigonometric function, which is more used. The intersection of two lines or curves in the parameter domain shows a line between two points in the spatial domain. The point to cluster is to find intersections in the parameter domain, getting rid of the sparsity of taking measures in the spatial domain. The idea of CASH is to find high-density regions on the parameter domain though a grid-based system for later doing a search for intersections. This algorithm has a poor runtime performance, that is exponential in d, in the worst case.

There are also some other ideas distancing from PCA analysis like exploiting self-similarity trough fractal dimensions also based on the *locality assumption*. Nevertheless, clustering algorithms based only in fractal dimensions do not appear to obtain good results. Finally, there are also some systems based on random sampling, consisting on taking $d + 1$ random samples iteratively n, choosing n such that there is a certain probability to get a sample from each of the existing clusters [25]. There is another approach based on RANSAC [26], used for model fitting, and with some applications as well on image processing such as line fitting. Anyway, the latter commented approaches do not seem to conclude neither efficient or effective results [3].

5. EXPERIMENTAL TESTS AND RESULTS

Once all different approaches have been examined and described and considering that Kriegel et al. [3] did not run any empirical tests on any of the mentioned algorithms in their survey, it has been decided to run those series of tests in this thesis. Some algorithms are going to be selected and tested with different datasets of different dimensionalities to see their real behavior.

The framework selected to run the experiments is the ELKI System [1], which is an environment for *Knowledge Data Mining* applications focused on clustering techniques. Their developers have also developed some of the algorithms commented in this thesis. This platform is implemented in Java and it uses some specific indexing structures to organize the memory space better according to each approach and therefore increase the running time. The platform has an intuitive and simple layout with some ready-to-use algorithms and distance metrics. After running the desired experiments, ELKI has also some visualization tools to help the user to understand how the points cluster in the data space. Besides, the platform has a great number of performance measures based on different evaluation metrics groups.

Three datasets have been selected to be working with:

- Pov-2: a toy dataset, only 2-dimensional, to see how the algorithms perform in typical and well-studied clustering problems. This dataset is composed of three clusters whose dimensions are drawn from different gaussian distributions with different parameters. There are 150 samples and none of them belongs to noise. This dataset is provided by the ELKI group.
- Subspaces-10: a 10-dimensional dataset, to test how the algorithms work when the dimensions start to increase. Here, the data is composed mainly of gaussian distributions with the addition of some uniform ones. There are four clusters and 350 samples of which the 14% are noise. This dataset also belongs to ELKI.
- Synthetic-control[2]: a 62-dimensional dataset. It can already be said that this one has high-dimensional characteristics. The data is drawn from synthetically generated control charts. There are 600 samples belonging to 6 classes with no noise. This data has been collected from a web with datasets to do research in the machine learning area.

[1] ELKI: Environment for Developing KDD-Applications Supported by Index-Structures - https://elki-project.github.io/ - [Accessed 16 05 2018]
[2] Synthetic control dataset - https://www.openml.org/d/377 - [Accessed 16 05 2018]

Higher dimensional datasets have also been tried, in the order of hundreds, but the framework did not seem to respond correctly when working with such data, and no conclusion could be reached.

ELKI framework offers 5 main groups of evaluation metrics based on the idea of having a ground truth to compare with: *pair counting* measures, *entropy-based* measures, *BCubed-based* measures, *Set-matching-based* measures and *Editing-distance* measures. According to Amigó et al. [27], the evaluation metrics must meet some properties to be able to react to any change in a cluster diminishing its quality. *Cluster homogeneity* explains that evaluation metrics should reward items of the same cluster to be similar. Complementing this, *cluster completeness* says that two groups of items belonging to the same category should be grouped into the same cluster. Following, *rag bag* states that it is better to introduce noise into an ill-conditioned cluster that into a clean one. Last, *cluster size versus quantity* exposes that it is preferable to have a small error into a big cluster than having multiple errors in small clusters.

The only group of evaluation metrics fulfilling the mentioned properties is the one containing the *BCubed-based* measures. *BCubed* metrics include the *precision*, which is the amount of samples in the cluster grouped according to the ground truth cluster, and *recall*, is the proportion of items from its ground truth category appear that in its corresponding cluster. The *F-measure* is a proportion that relates both *recall* and *precision,* being 0 the lower bound showing the maximum error and 1 the upper bound indicating a perfect match. It must be noted that a random classifier can easily get a *F-measure* of 0.5 so it will be expected a good clustering method to have a larger value. The algorithms we are going to use can create overlapping clusters, but extended *BCubed* metrics can deal with this problem. F-measure from *BCubed* evaluation metrics is the one that is going to be used in the proposed tests.

Group	Subgroup	Algorithm	$d = 2$	$d = 10$	$d = 62$	Rank
General		k-means	0.8478	0.8122	0.9224	1
		DBSCAN	0.8295	0.7039	0.9081	3
Axis-Parallel Subspaces	Projected	PROCLUS	0.8349	0.6500	0.6780	7
		PreDeCon	0.8295	0.7039	0.9081	3
	Subspace	CLIQUE	0.5238	0.8039	0.3053	9
		SUBCLU	0.4975			13
	Hybrid	DOC	0.7279	0.9686	0.4525	8
		DISH	0.7999	0.4254	0.4016	11
		P3C	0.6303	1		14
Arbitrarily-Oriented Subspaces	PCA-based	ORCLUS	0.7545	0.6104	0.2299	9
		4C	0.8295	0.7039	0.9081	3
		COPAC	0.8911	0.7037	0.9950	2
		ERiC	0.8000	0.4191		6
	Other	CASH	0.5276	0.4754		12

Table 1: *F-measure results for each algorithm with the different databases. Green color highlights the best results, orange denotes running time problems and finally red symbolizes Java memory exceptions.*

In Table 1 the F-measure results for the run experiments can be contemplated. In the last column, the rank is computed using the average of the individual ranks for each dataset. The total average has not been used as the distributions of the data cannot be directly compared.

The parameters for the tests have been manually selected to obtain the best results possible. All the algorithms have used Euclidean distance to compute closeness except those algorithms based on special distance functions.

As it can be appreciated in the table, the best method is surprisingly the simplest one, k-means, but this result is somehow unfair because the number of desired clusters was selected beforehand, and in a real unsupervised case that could not be done. The following methods with best results are DBSCAN and derivations of this method such as PreDeCon, 4C or COPAC, being the last one the only able to overcome its predecessor. One can also see that some methods start to fail with only 10 dimensions and that almost all of the supposed high-dimensional approaches fail to solve the problem for 62 dimensions. It also should be noted that most of the algorithms are very hard to parametrize.

The yellow boxes indicate that there has been a waiting time of plus 30 minutes and the parameters could not be improved or that they could not even be retrieved. It is interesting to note that one of the algorithms that fails with this problem is SUBCLU, that follows the Subspace algorithms approach aiming to find all clusters in all subspaces. The red box symbolizes a memory exception, making P3C unfit to work with high-dimensional datasets.

It can be baffling, but some of these algorithms, like PROCLUS [7], were only tested in the original research paper with datasets of 7 to 10 dimensions and hence, the behavior seen on the table can be understood. Others, like SUBCLU [10] are tested with up to 50 dimensions, but bad parametrization makes it difficult to obtain results.

6. CONCLUSIONS

In this work the characteristics of the problem of cluster analysis for high-dimensional data have been analyzed. In addition, some algorithmic approaches have been selected and studied to overcome this problem, starting from the more simplistic or basic ones, and advancing to more specific solutions, which have been further classified in different groups and subgroups.

Some empirical tests have been run on different datasets with different characteristics to see the real behavior of each approach and most of the algorithms did not cope with real high dimensional data. It must be remembered, as said in the beginning of this thesis, that distance measures have problems in such a sparse space, and even though these algorithms try to approach the high-dimensional problem, all of them make use of distance measures.

Surprisingly, k-means has had the best results of the tests performed, to continue the work of this thesis some more tests with this approach should be performed, and some research should be made of using specialized distance measures on k-means. DBSCAN has also shown good results, and specifically COPAC, one of its derivations. It would be interesting to keep on working with more derivations of DBSCAN to see if they can be improved, as well as with COPAC itself.

Something else that has been noted is the difficulty of setting the parameters of each algorithm. Some approaches should be further studied to create solutions that automatically set the parameters according to some intrinsic characteristics of the data.

Finally, this thesis has focused on testing the data having a ground truth to compare with. However, in a real unsupervised learning problem that information is not available and despite the fact that there are some metrics based on the separability of the data, those metrics are also based on distance measures, and here the curse of the dimensionality is faced again. Therefore, some metrics should be developed for new data without any kind of labelling available.

REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters,* vol. 31, no. 8, pp. 651-666, 2010.

[2] M. Khoso, "Northeastern University - How Much Data is Produced Every Day?," 13 05 2016. [Online]. Available: http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day/. [Accessed 05 04 2018].

[3] H.-P. Kriegel, P. Kröger and A. Zimek, "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering," *ACM Transactions on Knowledge Discovery from Data,* vol. 3, no. 1, 2009.

[4] N. Tomasev, M. Radovanovic, D. Mladenic and M. Ivanovic, "The Role of Hubness in Clustering," *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, no. 3, pp. 739-751, 2013.

[5] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* pp. 226-231, 1996.

[6] G. Moise, A. Zimek, P. Kröger and H.-P. Kriegel, "Subspace and projected clustering: experimental evaluation and analysis," *Knowledge and Information Systems,* vol. 21, no. 3, 2009.

[7] C. C. Aggarwal, C. Procopiuc, J. L. Wolf and J. S. Park, "Fast Algorithms for Projected Clustering," *ACM SIGMOD,* vol. 28, no. 2, pp. 61-72, 1999.

[8] C. Böhm, K. Kailing, H.-P. Kriegel and P. Kröger, "Density Connected Clustering with Local Subspace Preferences," *Proceedings of the Fourth IEEE International Conference on Data Mining,* pp. 27-34, 2004.

[9] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *ACM SIGMOD,* vol. 27, no. 2, pp. 94-105, 1998.

[10] K. Kailing, H.-P. Kriegel and P. Kröger, "Density-Connected Subspace Clustering for High-Dimensional Data," in *International Conference on Data Mining*, 2004.

[11] C. M. Procopiuc, M. Jones, P. K. Agarwal and T. M. Murali, "A Monte Carlo algorithm for fast projective clustering," *Proceedings of the 2002 ACM SIGMOD international conference on Management of data,* pp. 418-427, 2002.

[12] E. Achert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman and A. Zimek, "Detection and Visualization of Subspace Cluster," *Advances in Databases: Concepts, Systems and Applications.,* pp. 152-163, 2007.

[13] G. Moise, S. Jörg and M. Ester, "P3C: A Robust Projected Clustering Algorithm," *Proceedings of the 6th International Conference on Data Mining (ICDM),* pp. 414-425, 2006.

[14] S. C. Madeira and A. L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," *IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS,* vol. 1, no. 1, 2004.

[15] J. A. Hartigan, "Direct Clustering of a Data Matrix," *Journal of the American Statistical Association,* vol. 67, no. 337, pp. 123-129, 1972.

[16] Y. Cheng and G. Church, "Biclustering of Expression Data," *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology,* pp. 93-103, 2000.

[17] Y. Yang, W. Wang, H. Wang and P. Yu, "δ-Clusters: Capturing Subspace Correlation in a Large Data Set," *Proceedings 18th International Conference on Data Engineering,* pp. 517-528, 2002.

[18] A. Ben-Dor, B. Chor, R. Karp and Z. Yakhini, "Discovering Local Structure in Gene Expression Data:," *Journal of Computational Biology,* vol. 10, no. 3-4, 2003.

[19] J. Liu and W. Wang, "OP-Cluster: Clustering by Tendency in High," *Proceedings - IEEE International Conference on Data Mining,* pp. 187-194, 2003.

[20] C. C. Aggarwal and P. S. Yu, "Finding generalized projected clusters in high dimensional spaces," *Proceedings of the 2000 ACM SIGMOD international conference on Management of data,* vol. 29, no. 2, pp. 70-81, 2000.

[21] C. Böhm, K. Kailing, P. Kröger and A. Zimek, "Computing Clusters of Correlation Connected objects," *Proceedings of the 2004 ACM SIGMOD international conference on Management of data,* pp. 455-466, 2004.

[22] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger and A. Zimek, "Robust, Complete, and Efficient Correlation Clustering," *Proceedings of the SIAM International Conference on Data Mining,* pp. 413-418, 2007.

[23] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger and A. Zimek, "On Exploring Complex Relationships of Correlation Clusters," *n Proc. 19th International Conference on Scientific and Statistical Database Management,* pp. 7-7, 2007.

[24] E. Achtert, C. Böhm, J. David, P. Kröger and A. Zimek, "Robust Clustering in Arbitrarily Oriented Subspaces," *Proceedings of the 2008 SIAM International Conference on Data Mining,* pp. 763-774, 2008.

[25] R. Haralick and R. Harpaz, "Linear Manifold Clustering," in *Machine Learning and Data Mining in Pattern Recognition*, Springer, 2005, pp. 132-141.

[26] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[27] E. Amigó, J. Gonzalo, J. Artiles and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval,* vol. 12, no. 4, pp. 461-486, 2009.