UNIVERSITÉ DE MONTRÉAL

NO-MEAN CLUSTERING ALGORITHM

GEOFFROY MOURET
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

NO-MEAN CLUSTERING ALGORITHM

présenté par: MOURET Geoffroy
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. LEMIRE Michel, M.Ing., président
M. BRAULT Jean-Jules, Ph.D., membre et directeur de recherche
M. PARTOVI NIA Vahid, Ph.D., membre et codirecteur de recherche
M. PAL Christopher, Ph.D., membre externe

DÉDICACE

À ma famille.

ACKNOWLEDGEMENTS

The present work would not have been possible without the valuable help of my academic advisors. I would like to thank my supervisor Professor Jean-Jules Brault for his guidance during this journey. Having an open door and a comprehensive person to talk to when I was still looking for a research project helped me a lot in the early stages of my master's degree. His feedback on this work brought new and interesting perspectives to the problem.

I want to express my deepest gratitude to my co-supervisor Professor Vahid Partovi Nia. For setting me on the right track, for all these discussions, for the financial help and countless academical opportunities, thank you so much.

I also thank the MITACS internship program and the CRSNG for their financial support.

I could not stress enough how precious the support of my family and friends has been. Having a home on both sides of the ocean helped me go through these past two years both in good and bad times. To my family, Mom, Dad, Capucine, Charlotte, Nicolas, Tiphaine and your own families, I am grateful for all your love and I miss you every minute I spend away from you. To my friends in Canada, old and new, local and imported, I am so glad I found you. The memories I forged during these two years with you are priceless. To my friends back in France, your encouragement and enthusiasm mean the world to me. To the three of you out there, for cheering me up every time I felt down, for being so far, yet so close, for helping me with matrix inversions and semicolons, for all the spoilers and scoubidous, I hope that what we lived is only the beginning a neverending story.

To Marc-Antoine, FF.

RÉSUMÉ

Ce mémoire présente un algorithme d'agrégation (clustering) Bayésien de données, l'algorithme no-mean (sans-moyennes). L'agrégation de données est un domaine de l'intelligence artifielle qui consiste à regrouper des objets dans différentes classes de manière non-supervisée, *i.e.* sans connaissance a priori sur la nature des classes.

Les applications de l'agrégation de données sont nombreuses, de la classification de génomes à la comparaison d'objets mécaniques en passant par la compression de données ou la reconnaissance de formes. L'objectif étant de regrouper les objets de telle sorte que tous les objets d'une même classe soient similaires entre eux. On compare pour cela leurs caractéristiques (features) à l'aide de différentes méthodes.

Une des méthodes les plus connues à l'heure actuelle est l'algorithme k-means (k-moyennes). Les causes de son succès sont multiples. Celui-ci est en effet très simple à implémenter, ce qui en fait un outil rapide mais néanmoins efficace. Son principal inconvénient provient de la forte dépendance du résultat à l'initialisation de l'algorithme. Pour un même jeu de données, plusieurs passages de l'algorithme k-means peuvent mener à des résultats différents et souvent sous-optimaux.

Les alternatives à cette technique sont nombreuses. On peut citer l'algorithme d'espérance-maximisation pour les modèles de mélanges ou encore les techniques d'agrégation hierarchisées. Ces techniques souffrent bien souvent d'un compromis rapidité-performance où l'optimisation de modèles plus réalistes se fait au coût de calculs plus lourds. Notre objectif est simple : améliorer la performance de k-means tout en gardant la même complexité de calcul, *via* une approche Bayésienne.

Pour cela nous avons tout d'abord montré comment l'algorithme k-means est relié à un modèle de mélanges de Gaussiennes. La minimisation de la variance intra-cluster par k-means est un cas particulier de certains algorithmes qui maximisent la vraisemblance de tels mélanges. Plus précisément, l'optimisation d'un regroupement de données par k-means revient à ajuster un mélange de Gaussiennes à l'aide de l'algorithme d'espérance-maximisation.

Sur cette base, nous avons ensuite montré qu'en considérant que les moyennes des agrégats suivaient une distribution normale multi-variée, k-means est également un cas particulier d'échantillonnage de Gibbs appliqué à l'ajustement d'un modèle Bayésien. Cette méthode échantillonne les moyennes d'un groupe en fonction des objets dans celui-ci puis échantillonne le nouveau groupe d'un objet en fonction des moyennes nouvellement calculées.

Le recuit simulé est une méthode qui consiste modifier les paramètres d'une distribution dans l'échantillonnage de Gibbs pour diminuer progressivement la variabilité de cet

échantillonage. Pour une diminution logarithmique des paramètres d'échelle des distributions dans l'échantillonage, le recuit simulé assure une convergence vers le maximum de ces distributions, *i.e.*, les points les plus probables. Dans notre cas, l'application du recuit simulé permet de faire transiter l'échantillonnage de Gibbs vers le k-means, mais en ayant guidé celui-ci vers des conditions initiales favorables.

L'algorithme no-mean est l'étape suivante de cette réflexion. Nous avons montré qu'en intégrant la distribution des données sur les moyennes des classes, il est possible d'obtenir une distribution conditionnelle sur la classe d'un objet en fonction de tous les autres. En d'autres termes, connaissant les classes des autres objets, il est possible d'obtenir la probabilité de chaque classe pour un objet. Cela nous permet d'appliquer un échantillonneur de Gibbs directement sur les classes elles-mêmes, sans avoir à échantillonner les moyennes. L'algorithme no-mean est la combinaison d'un tel échantillonneur avec la méthode du recuit simulé.

La principale innovation de notre algorithme est que nous sommes capable de calculer la probabilité d'une classe pour un objet en temps constant, ce qui le rend compétitif avec k-means en termes de temps de calcul. Sans notre optimisation, la complexité a priori du calcul empêcherait toute application pratique de la méthode.

Nous avons testé l'algorithme no-mean sur des jeux de données simulés en accord avec notre modèle a priori et validé la bonne performance de no-mean, en termes de résultats et de temps de calcul. Notre algorithme améliore la qualité des résultats obtenus par k-means pour la même initialisation dans 70% des cas. Il apparaît que notre algorithme est d'autant plus performant que le nombre de dimensions est grand, ainsi que pour un grand nombre d'agrégats.

Pour des jeux de données réels, nous avons appliqué la méthode à un problème de classification de nuages et d'intrusions dans un système informatique. Nous avons ensuite comparé nos résultats avec ceux obtenus par k-means pour les mêmes initialisations et par `k-means++`, une variante récente qui optimise l'initialisation du problème. Les résultats sont très satisfaisants pour la classification de nuages (jusqu'à 7 fois plus performant que k-means et `k-means++`) mais restent mitigés en ce qui concerne les intrusions. La distribution particulière dans le deuxième cas nous empêche de pouvoir appliquer efficacement notre modèle et la grande taille du jeux de données pose des problèmes de convergence qui limitent la performance moyenne de notre algorithme. Celui-ci reste cependant capable d'obtenir dans les meilleurs cas des résultats qui améliorent de plusieurs ordres de grandeur ceux obtenus par k-means et `k-means++`.

Nous avons également testé la possibilité d'utiliser l'algorithme no-mean pour la sélection de variable. En comparant la probabilité d'un regroupement pour une variable avec la probabilité qu'aucun regroupement n'existe pour cette variable, nous sommes capables de distin-

guer si celle-ci est importante pour une classification particulière. Les résultats préliminaires sur simulations sont encourageants mais l'absence de résultats probants pour de vrais jeux de données exige une étude plus poussée de la méthode et l'exploration de pistes alternatives.

Finalement, à l'aide des outils présentés dans ce mémoire, nous évoquons l'idée d'appliquer ce modèle à la sélection du nombre de classes. Ce problème étant particulièrement complexe, nous nous sommes contentés de présenter quelques pistes dont nous pensons qu'elles méritent une certaine attention.

Avec l'application d'une approche Bayésienne inspirée de l'algorithme k-means au problème de regroupement de données, nous proposons une méthode compétitive dont l'efficacité a été montrée en simulations et sur jeux de données réels. En abordant le problème d'un point de vue probabiliste, nous diminuons l'impact de l'initialisation sur le résultat, tout en conservant des temps de calculs du même ordre de grandeur.

ABSTRACT

This thesis devises and presents the no-mean algorithm, a Bayesian clustering algorithm. Data clustering is a sub-domain of artificial intelligence which aims to gather observations into different groups in an unsupervised fashion.

Clustering has a broad range of applications, including (but not limited to) genome classification, data compression or even pattern recognition. By comparing the objects' features, clustering methods organize objects into classes so that objects within the same class are close to each other.

Among these methods, the k-means algorithm is one of the most famous. The k-means has a solid foundation, easy to implement, fast and efficient. However, its strong dependence to initial values is a major issue. Multiple initializations lead to different results and the convergence to the optimal solution is not guaranteed.

The expectation-maximization (EM) algorithm for mixture models or hierarchical clustering are examples of alternatives to the k-means algorithm. Improvements are often made at the cost of a speed versus performance trade-off. More realistic models offer better results but are more complex and more difficult to compute. Our main objective is to improve the k-means algorithm performance while keeping the same computational complexity.

We show the link between the k-means and Gaussian mixtures. The k-means is a special case of the EM algorithm applied to a mixture of Gaussian distributions. We then prove that if a Gaussian prior is assumed on the cluster centers distribution, the k-means is also a special case of the Gibbs sampling applied to fit a Bayesian model with some priors. The Gibbs sampler consists in the successive sampling of centers and then clusters using the conditional distributions.

Simulated annealing progressively decreases the variance of the sampled values of a Gibbs sampler by modifying the parameters of the distribution. For a logarithmic rate of decrease of these parameters, the Gibbs sampler converges to the maximum of the sampled distributions. Simulated annealing applied to our Gibbs sampler pushes it to behave like k-means after enough iterations.

Integration of the marginal data distribution over cluster centers allows us to conditionally sample the new cluster of an object, given the current partitioning. In this case, the Gibbs sampler can be applied directly to the clusters, without sampling the cluster means, leading to the no-mean algorithm. Our proposed method, the no-mean algorithm, is the Gibbs sampler coupled with simulated annealing.

The computational efficiency of our implementation is the main innovation of this work.

The computational complexity for one iteration of the no-mean algorithm is competitive with the k-means. Our implementation allows us to sample new clusters for the Gibbs sampler in a constant time, when a basic implementation would have a higher computational complexity.

The no-mean provides satisfactory results on simulated datasets, both in terms of performance and computation time. For a given initialization, our algorithm performs better than the k-means 70% of the time. The increase of performance over the k-means is more important as the number of dimensions and clusters increases.

We applied the no-mean algorithm to two real datasets for the clustering of clouds and computer networks intrusions. The k-means and the `k-means++`, a variant that optimizes the initial centers, have been applied to these datasets and we compared the results of the three methods. While the no-mean presents better results overall for the cloud classification problem (performance is increased up to 7 times compared to the k-means and the `k-means++`), its performance on the intrusion dataset is not as positive. The fact that the distribution of the data in this dataset does not fit well with our model and the large number of observations make it more difficult for the no-mean algorithm to converge to satisfactory solutions. Though the average behavior of the no-mean is not optimal on the intrusion dataset, the best results still improve on those obtained by the `k-means++` by several orders of magnitude.

We also initiated a variation of the no-mean algorithm for variable selection. By comparing for a variable the probability of a given clustering with the probability of observations being all in the same cluster, we determine whether this variable has an impact on the clustering. Though preliminary results on simulated data are promising, the absence of conclusive results for real datasets calls for a more thorough study of the variable selection variant.

Finally, we suggest another variation of the no-mean algorithm for the selection of the number of clusters. Since the questions raised by this problem go beyond the framework of this thesis, we only present a few leads for further studies.

Our Bayesian approach to the k-means algorithm led to a competitive data clustering method. Its performance has been verified on both simulated and real datasets. By considering the probabilistic aspect of the problem, we reduce the impact of the initialization on the final result with the same computational complexity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF NOTATIONS

n	Number of objects to be clustered.
p	Number of variables, features, or dimensions.
$\mathbf{Y}$	Data matrix of n rows and p columns.
i	As an index, denotes an object.
j	As an index, denotes a variable.
$\mathbf{y}_i$	i-th object of a dataset.
$\mathbf{y}_j$	Vector of length n containing the value of the j-th variable for each object of a dataset.
k	Total number of clusters.
c	As an index, denotes a cluster label.
n_c	Number of observations in cluster c.
$\mathbf{d}$	Vector of length n containing the cluster labels for a dataset.
$\mathbf{Y}_c$	Matrix of all observations in cluster c.
$\mathbf{y}_{c_i}$	i-th observation of cluster c.
$\mathbf{y}_{cj}$	Vector of length n_c containing the value of the j-th variable for each object in cluster c.
$\bar{\mathbf{y}}_c$	Mean of observations in cluster c (univariate case).
$S_{c,j}$	Sum of the squared values of the j-th variable for all observations in cluster c.
$B_{c,j}$	Sum of the values of the j-th variable for all observations in cluster c.
S_W	Within-cluser dissimilarity.
RI	Rand index.
$L(\cdot)$	Likelihood function.
$\ell(\cdot)$	Log-likelihood function.

CHAPTER 1

INTRODUCTION

As datasets keep growing bigger, need for efficient and fast machine learning algorithms has never been stronger. The parallel computing offers great improvement in computation time. The parallel computing comes under the assumption that the algorithm can either process different sub-datasets or sub-tasks simultaneously. Clustering provides a way to find such sub-datasets by dividing data into groups, each group sharing some common features.

We propose a new algorithm, the no-mean algorithm, which is a probabilistic and improved version of the k-means. The k-means algorithm is one of the most popular clustering algorithms, due to its good performance and computational complexity. This new version aims to improve the quality of clustering by resolving the k-means' initial value problem. Chapter 1 presents a brief overview of the objectives behind data clustering, also the context in which our study takes place and then gives the outline of this thesis.

1.1 Clusters

Clusters have different meanings in different contexts. The term exists for computer clusters, which is a system consisting of many computers linked together to act like a single and more powerful structure. We do not have the concept of partitioning a set of objects, but this definition integrates the idea of a group of units that have some properties in common; to the point it can be considered as a single object. Data clusters are the memory units on a computer disk but are themselves constituted of contiguous disk sectors to improve writing and reading speed.

Another common use of the word cluster appears for clusters of galaxies. Whereas the previous examples concerned structures created and controlled by humans, galaxy clusters are natural formations. They consist of galaxies agglomerated together by gravity, making them the one of the largest natural formations in the universe.

These are all examples of clusters that can be seen in common language but when it comes to machine learning, clustering takes a whole new meaning. According to Hastie *et al.* (2009, p. 501), cluster analysis is "*grouping or segmenting a collection of objects into subsets or* clusters, *such that those within each cluster are more closely related to one another than objects assigned to different clusters*". On the contrary to previous examples, such clusters result from an active process, that is, dividing a dataset. And not only do we need to find a

partition of our data, we need to find a good one.

1.2 Cluster analysis

1.2.1 Applications

Clustering is a task that humans are good at, while is challenging to program. Thousands of years ago, our human ancestors were already able to distinguish between different animals and classify them as threats, or preys. People were already clustering stars into constellations for a better reading of the sky even before we learned about clusters of galaxies.

Clustering has evolved considerably since then. Finding groups of similar objects in a dataset is much more than parallel computing and multi-threading. By clustering observations, we capture a better representation of our data.

The range of applications of data clustering is wide and goes from genetics (finding groups of genes that have the same impact on a physical condition or groups of patients presenting close genetic profiles), to cosmology (classify stars or exoplanets and their likelihood to host life), to aeronautics (sorting construction parts), and robotics (gathering units that collect similar information).

Since each cluster contains information about its objects, they can be used to represent a dataset in a very compact way. Clustering shows that the tens of thousands clients of a company can be divided into 5 groups of consuming habits? Study each of these habits and predict the behaviour of a client based on what you expect from this group. Clustering allows us to measure variations through time for each group and adapt your strategy to take these changes into account. One can argue that this representation is made at the cost of client specificity. On the contrary, instead of confronting one client to your whole database to determine his characteristics, one only has to see what makes him special in his own group.

Some algorithms also work better when they are applied to datasets with strong clustering features. It becomes easier to get specific results either because the discrimination between objects is stronger or the parameters shaping the data can be found more accurately. In this case, clustering helps to find the best sub-groups depending on the features needed for the algorithm.

Clustering is highly linked to the representation of data. Most of the time, the final clusters will depend on the spatial distribution of objects. The goal is to find the representation that makes the clustering process as efficient as possible. The measure of performance for a given clustering is an indication of whether a representation is appropriate. Using this information as a performance index for the representation instead has practical applications. The no-mean algorithm proposed in this thesis can also perform variable or feature selection simultaneously.

1.2.2 Obstacles

In biological classification, living creatures are clustered into different categories, called classes. One class is made of orders, which are made of families, that consist of genus and finally the species themselves. This classification is solely based on similar morphological features between species. The phylogenetic nomenclature is another approach to biological classification which sorts species according to their evolution path, or to their genetic mutation. Hence, for the same species, biologists have two ways of clustering them.

Partitioning depends on how we perceive our data. In the definition of cluster analysis of Hastie *et al.* (2009, p. 501), having two objects "*closely related to one another*" is not straightforward to formalize. In order to achieve efficient clustering, we have to take multiple issues into account. We need to define a distance between objects with "good" properties. The choice of features is an important topic, as well as the question of whether the set of parameters offers a good separation between the clusters. We will have a closer look at these issues in Chapter 3.

Checking whether two objects are *closely* related is the key issue to perform data clustering. Indeed, for well-separated clusters on two dimensional space, visual inspection of data helps to distinguish different groups. However, if the number of dimensions is higher or if the clusters get entangled (see Fig.1.1), human eyes and computers need a lower dimensional representation space that offers good visual separability to guide forward a good clustering.

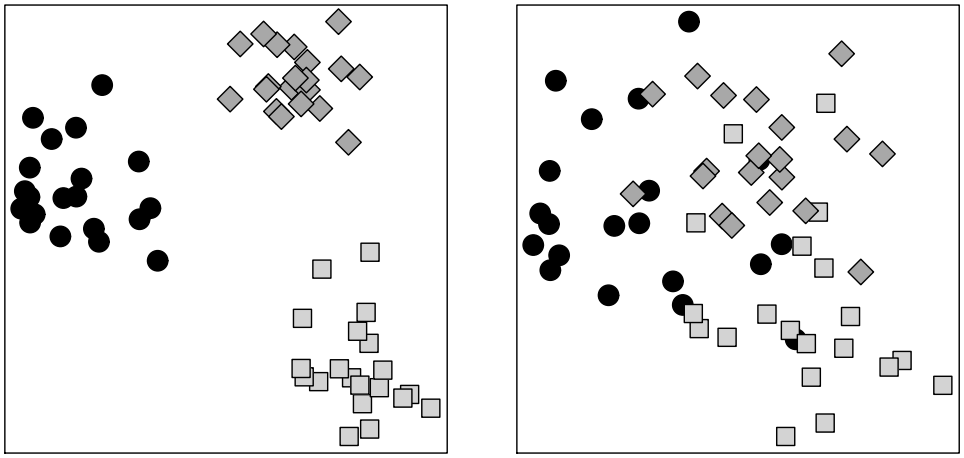


Figure 1.1 Visualization of 3 clusters in 2 situations.

Even if we consider optimal conditions, clustering algorithms need alternatives other than the brute force exploration of every possible partitioning of the data. Indeed, depending on the number of clusters, the amount of configurations goes from n^2 (for $n-1$ clusters) to 2^n

(for 2 clusters). Since most situations require a lot less clusters than the size of a dataset, the number of solutions often grows exponentially with the number of objects to be clustered, but not faster than $n!$. This prevents us from using brute force, but also rises the issue of improper initialization. Algorithms that optimize their clustering from step to step (like the k-means), search for an optimal solution in a space so wide, meaning that one may end up with only a locally optimal solution. Our goal with the no-mean algorithm is to avoid this trap by adding random mutations and search the grouping space wide-enough before converging to a solution.

1.3 General objective

The general objective of this research is to develop the no-mean algorithm, a clustering algorithm that offers a lower within-cluster variation than the k-means algorithm for the same initialization but with the same computational complexity. This algorithm also allows variable selection.

1.4 Thesis objectives

Specific objectives are defined as steps towards the final goal and are as follows

1. Assessing the mathematical proof that the algorithm minimizes the within-cluster variation.
2. Demonstrating the relationship between the no-mean and the k-means algorithm.
3. Simulating data to study the impact of the different algorithm parameters on its performance.
4. Comparing the k-means with the no-mean algorithm.
5. Studying the feasibility of variable selection.

1.5 Thesis structure

The thesis continues as follows. Chapter 2 lists classic clustering methods and provides a literature review of the state-of-the-art. Comparisons between the k-means algorithm and the proposed version are developed in Chapter 3. Chapter 4 is the methodology behind the no-mean algorithm, including the optimization techniques used to improve its computation time. Results are finally presented in Chapter 5 and then discussed with potential developments in the Conclusion.

CHAPTER 2

LITERATURE REVIEW

Cluster analysis presents itself in an extensive spectrum of forms. From clustering trees to unsupervised neural networks, Section 2.1 presents an overview of famous or historical methods. Section 2.2 focuses specifically on the k-means algorithm and its variants, while Section 2.3 concludes this review with some insight on Bayesian clustering.

2.1 The state of clustering

Cluster analysis is an active field in machine learning and hundreds of papers are published each year to introduce new methods or variations of existing algorithms (Jain, 2010). New problems are tackled and new challenges arise everyday with new datasets or the introduction of other unsupervised learning methods.

The challenge in clustering comes from its unsupervised nature, meaning that there is no information available about what are the possible clusters. They have to be constructed based on geometrical properties. This leads to two main approaches: hierarchical methods and partitioning methods.

Hierarchical clustering

Hierarchical clustering is divided into two types of algorithms: agglomerative or divisive. Both techniques construct a hierarchy of clusters, so that each cluster is contained within a bigger one. This hierarchy is represented using a dendrogram (see Figure 2.1). A dendogram is a tree with each node representing a fusion of two clusters. Single observations are the leaves of the tree and the whole dataset is the root.

In agglomerative (bottom-up) hierarchical clustering (Ward, 1963), clusters are formed by merging the two closest clusters into one. Defining a distance between clusters is a key issue in hierarchical clustering. There are 3 common *linkages* set as the distance between two groups of observations (Johnson, 1967). A linkage is based on the metric distance s (see Table. 2.1 and Figure 2.2 for geometrical insight):

- Single (or minimum) linkage defines the distance as the minimal distance between 2 points in these clusters, sometimes called the nearest neighbours linkage.
- Complete (or maximum) linkage defines the distance as the maximal distance between 2 points in these clusters, sometimes called the furthest neighbours linkage.

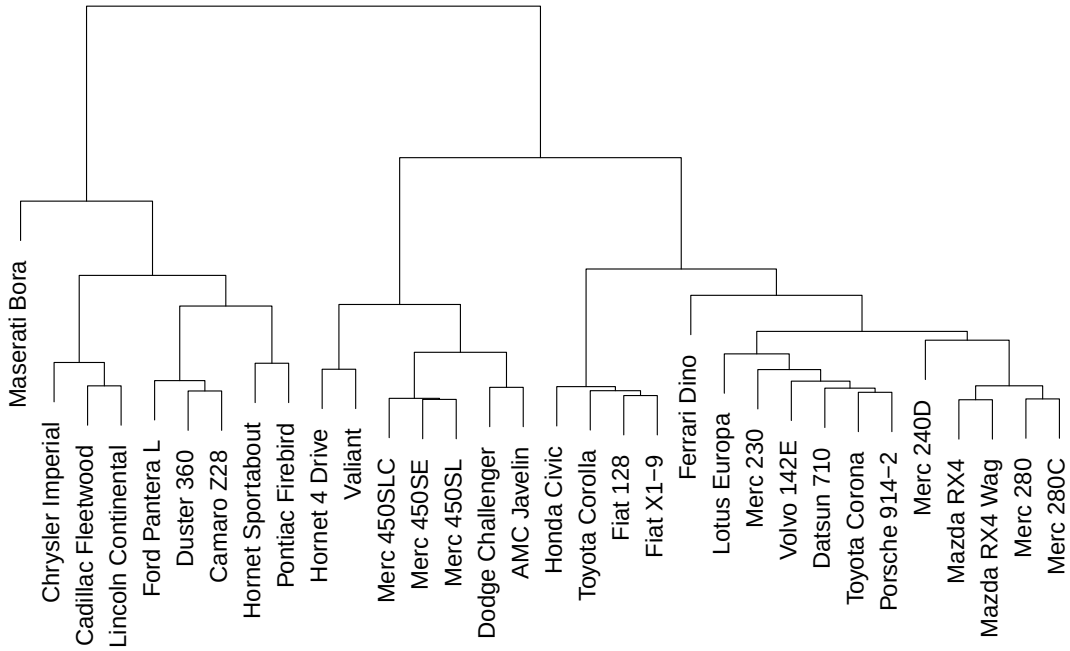


Figure 2.1 Hierarchical clustering for the `mtcars` dataset in `R`.

– Average linkage defines the distance as the average distance between points in one cluster to all points in the other cluster.

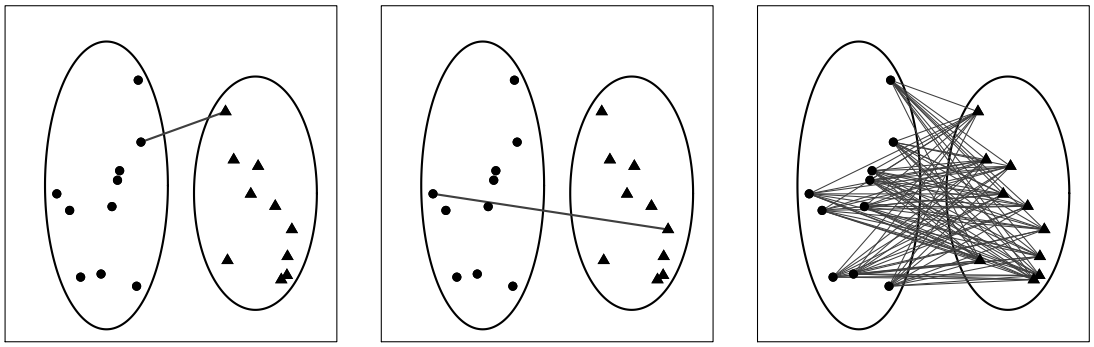
An inconvenience of the single linkage criterion is its tendency to agglomerate observations into one big cluster, contrary to the complete linkage for which clusters tend to be of similar size. Average linkage (also known as *unweighted pair group method with arithmetic mean*) is an alternative often used in biological applications. All methods have a naive implementation of time complexity $O(n^3)$, improved to at most $O(n^2)$ with some computational tricks (Sibson, 1973; Defays, 1977; Murtagh, 1983; Day and Edelsbrunner, 1984).

All linkages presented in Table 2.1 have been generalized in the Lance-Williams algorithm (Lance and Williams, 1967). Distances between clusters are upgraded recursively based on what clusters were merged at the previous step, along with some parameters. Single, complete, and average linkages are all specific cases of the Lance-Williams version.

Divisive (or top-bottom) hierarchical clustering is a more complex operation. Indeed, whereas the first step of an agglomerative algorithm only needs to check $\frac{n(n-1)}{2}$ combinations,

Table 2.1 Linkage clustering criteria, suppose X is the set of data in one cluster and Y is the data in another cluster, $s(x, y)$ is the distance between two points (x, y).

Linkage	Definition
Single linkage criterion	$s(X,Y) = \min_{x \in X, y \in Y} s(x,y)$
Complete linkage criterion	$s(X,Y) = \max_{x \in X, y \in Y} s(x,y)$
Average linkage criterion	$s(X,Y) = \frac{1}{n_X n_Y} \sum_{x \in X, y \in Y} s(x,y)$



(a) (b) (c)

Figure 2.2 (a) Single Linkage (b) Complete Linkage (c) Average Linkage.

the number of 2-clusters partitions of a dataset is $2^{n-1} - 1$. In practice, it is impossible to go through all combinations and divisive algorithms need techniques to split clusters. Bisecting k-means (Steinbach *et al.*, 2000) does it so by finding 2 clusters inside a group of observations.

The main problem with this approach is to define which cluster to split. Savaresi *et al.* (2002) present approaches such as splitting the most populated cluster or the one with the largest within-cluster variance.

While hierarchical clustering returns a full set of partitions from individual clusters to the whole dataset, its computational complexity is still too high for big data applications. The quest for optimal stopping criteria (Mojena, 1977; Milligan and Cooper, 1985; Jung *et al.*, 2003) remains strong in order to provide more efficiency to an already powerful and visual technique.

Partitional clustering

In contrast to constructing clusters by grouping or splitting data in successive clusters like hierarchical methods, partitional clustering algorithms aim at forming a fixed number of clusters.

Clustering a dataset into k groups is equivalent to labeling each observation with an integer $1, \ldots, k$. The labels of a whole dataset are in a vector of size n, denoted by $\mathbf{d} \in \{1, \ldots, k\}^n$, where d_i is the cluster to which the i-th observation is allocated. Clustering a set of 5 observations with $\mathbf{d} = (1, 1, 2, 2, 3)^\top$ means that the two first objects are in the same cluster, observations 3 and 4 are in another cluster and observation 5 is a singleton.

In the past decades, much attention has been paid to what is called the finite mixture model and its applications to clustering. In this context, each cluster has its own statistical model, meaning that observations $\mathbf{y}_c$ in cluster c are sampled from a given distribution function, with parameters $\boldsymbol{\theta}_c$, $f_c(\mathbf{y}_c) = f(\mathbf{y}_c|\boldsymbol{\theta}_c)$. Data come from distribution c with a probability p_c, which leads to the marginal distribution

$$f(\mathbf{y}_i) = \sum_{c=1}^{k} p_c f(\mathbf{y}_i|\boldsymbol{\theta}_c) \tag{2.1}$$

and the associate likelihood

$$L(\mathbf{y}; \mathbf{d}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{c=1}^{k} p_c f(\mathbf{y}_i|\boldsymbol{\theta}_c). \tag{2.2}$$

If the mixture model is known, the probability that each observation comes from a given cluster c is inferred using the Bayes' theorem

$$\Pr(c|\mathbf{y}_i) = \frac{\Pr(\mathbf{y}_i|c)\Pr(c)}{\Pr(\mathbf{y}_i)} = \frac{f(\mathbf{y}_i|\boldsymbol{\theta}_c)p_c}{\sum_c p_c f(\mathbf{y}_i|\boldsymbol{\theta}_c)}, \tag{2.3}$$

then a Bayes Classifier is $\hat{c}(\mathbf{y}_i) = \underset{c}{\operatorname{argmax}}\ \Pr(c|\mathbf{y}_i)$.

The purpose of probabilistic clustering is twofold: find appropriate models and fit these models to a dataset. The former is often constrained by the latter in the sense that sophisticated models are generally more complex to optimize.

Fitting a mixture model is implemented through maximizing the likelihood with respect to the mixtures probabilities and mixture distribution parameters for a given set of data

$$\left\{\hat{p}_1, \ldots, \hat{p}_k, \hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_k\right\} = \underset{p_1,\ldots,p_k,\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_k}{\operatorname{argmax}} L(\mathbf{p}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k) = \underset{p_1,\ldots,p_k,\theta_1,\ldots,\boldsymbol{\theta}_k}{\operatorname{argmax}} \prod_{i=1}^{n} \sum_{c=1}^{k} p_c f(\mathbf{y}_i|\boldsymbol{\theta}_c), \tag{2.4}$$

which is a non-convex optimization problem, in terms of $(\mathbf{p}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k)$.

Historically, mixture of Gaussians have been the first to be considered (Behboodian, 1970).

Hasselblad (1966) proposed a gradient descent algorithm to estimate the parameters of

a mixture of normal distributions. Like most gradient descent techniques, such algorithms converge to a local maximum of the likelihood.

In Gaussian mixtures, the optimization depends on the flexibility of each mixture. Different criteria have been suggested depending on whether mixtures are considered to have the same covariance matrix (Friedman and Rubin, 1967), have all their own covariance matrix (Scott and Symons, 1971) or a combination of both (Banfield and Raftery, 1993).

The Expectation-Maximization (Dempster *et al.*, 1977; Wu, 1983), also known as the EM algorithm, is an alternative to the gradient descent. Considering a dataset $\mathbf{Y}$ with unknown labels $\mathbf{d}$ to be clustered using a mixture of k components with parameters $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k\}$, the EM algorithm maximizes the marginal likelihood defined in (2.4).

This method is a two-step iterative process

1. Expectation step: Compute the expected value of the log-likelihood using the conditional distribution of the labels $\mathbf{d}$ given $\mathbf{Y}$ and the current parameters estimates

$$\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\mathbf{d}|\mathbf{Y},\boldsymbol{\theta}^{(t)}}\left\{\log L(\mathbf{Y}, \mathbf{d}, \boldsymbol{\theta})\right\} = \sum_{i=1}^{n} \log \left\{ \sum_{c=1}^{k} \Pr(d_i = c|\mathbf{Y}, \boldsymbol{\theta}) f(\mathbf{y}_i|\boldsymbol{\theta}_c) \right\}. \quad (2.5)$$

   The computation of $\Pr(d_i = c|\mathbf{Y}, \boldsymbol{\theta}^{(t)}) = p_{c,i}^{(t)}$ is straightforward using Bayes' Theorem, as seen in (2.3)

$$p_{c,i}^{(t)} = \Pr(d_i = c|\mathbf{y}_i, \boldsymbol{\theta}^{(t)}) = \frac{\Pr(\mathbf{y}_i|d_i = c, \boldsymbol{\theta}^{(t)}) \Pr(d_i = c)}{\Pr(\mathbf{y}_i|\boldsymbol{\theta}^{(t)})} = \frac{f(\mathbf{y}_i|\boldsymbol{\theta}_c^{(t)}) p_c^{(t)}}{\sum_c p_c^{(t)} f(\mathbf{y}_i|\boldsymbol{\theta}_c^{(t)})}. \quad (2.6)$$

2. Maximization step: Maximize $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ with respect to $\boldsymbol{\theta}$ to find the new set of parameters

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \sum_{i=1}^{n} \sum_{c=1}^{k} p_{c,i}^{(t)} \log\left[f(\mathbf{y}_i|\boldsymbol{\theta}_c)\right]. \quad (2.7)$$

The advantage of the EM algorithm over gradient-descent techniques is that the maximization step is linear in terms of $\boldsymbol{\theta}$ for mixtures of exponential family distributions (Sundberg, 1974), such as Bernoulli, Poisson, normal, gamma, beta, multinomial, and Dirichlet distributions. The EM algorithm does not require to compute derivatives in these cases and is therefore much faster in terms of computing speed (McCulloch, 1997).

For more complex distributions, Meng and Rubin (1993) developed the ECM algorithm, a variant of the EM algorithm. The maximization step is replaced by multiple steps of maximization of one parameter while the remaining parameters are kept fixed. It presents the same results and properties as the EM algorithm with simpler computations.

The optimization step may converge to local extrema which is the main drawback of the EM algorithm. Multiple initial values are usually required in order to obtain satisfactory results.

The mixture model is known as a flexible modeling tool. However, the EM algorithm in its basic form is not always the best suited method and we will see throughout this thesis how different methods such as the k-means or Bayesian algorithms provide great alternatives to the EM algorithm.

2.2 k-means

Classic approach

Among all the clustering techniques, the k-means is probably one of the most famous. It is one of the oldest clustering method and is still one of the most used algorithm nowadays. The idea behind the k-means is simple. The goal is to find the partition that minimizes the within-cluster sum of squares

$$\sum_{c=1}^{k} \sum_{i=1}^{n_c} \|\mathbf{y}_{c_i} - \boldsymbol{\mu}_c\|^2, \tag{2.8}$$

where $\mathbf{y}_{c_i}$ are observations inside cluster c and $\boldsymbol{\mu}_c$ are the means of these data points. The term "k-means" appears with MacQueen (1967) and its algorithm:

1. Initialize k centers (means) by randomly selecting observations.
2. Select a new data point and allocate it to the closest cluster.
3. Update the cluster means each time you move an observation from one cluster to another.
4. Repeat step 2 and 3 until convergence.

By providing the proof of its convergence, MacQueen (1967) introduces an algorithm that has practical application but many times gives a locally optimal solution.

The original idea goes up to Steinhaus (1956). His intention was to solve the sum-of-squares for the clustering problem as introduced by Dalenius (1950). His algorithm can be seen as a continous version of the k-means which divides a multivariate continuous space described by a probability distribution $\rho(x)$ to minimize

$$\sum_{c=1}^{k} \int_{V_c} \|x - \mathbb{E}\left[X | X \in V_c\right]\|^2 \rho(x) \mathrm{d}x. \tag{2.9}$$

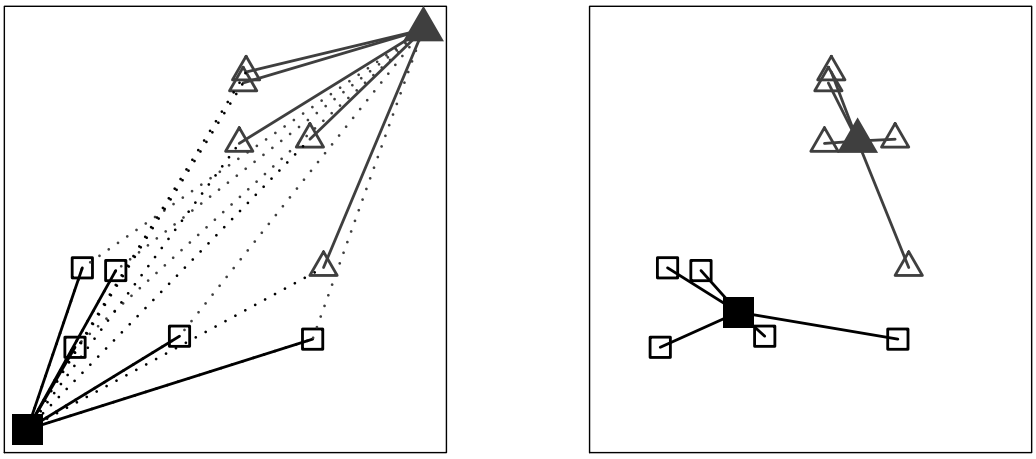
Steinhaus (1956) also discusses the existence and uniqueness of a solution as well as the asymptotic behaviour when $k \to \infty$.

Without explicitly using the k-means denomination, other works introduced a similar method. The first implementation of what is known today as the k-means algorithm has been developed by Lloyd (1982). The paper was published in 1982 but was initially written in 1957 as a technical report in Bell Labs.

After Lloyd (1982) but before the publication of Lloyd (1982), Forgy (1965) presented the same technique, except that he applied it to continuous rather than discrete mixtures. In computer science, the Lloyd-Forgy version of the k-means algorithm can also be applied to various problems such as the construction of a Voronoi tessellation (Du *et al.*, 1999). Lloyd-Forgy differs from MacQueen's algorithm because it updates the means after all observations have been assigned to their new clusters. Therefore, Lloyd-Forgy algorithm can be seen as the batch version of the k-means, whereas MacQueen's is an online version. The Lloyd-Forgy version is:

1. Initialize k centers (means) by randomly selecting k observations.
2. Allocate all data points to the closest clusters.
3. Update the cluster means.
4. Repeat step 2 and 3 until convergence.

Fig. 2.3 illustrates steps 2 and 3 of this version.



(a) (b)

Figure 2.3 Lloyd-Forgy algorithm: (a) Allocation step – (b) Updating step.

Hartigan and Wong (1979) proposed a slightly different version of the k-means. The algorithm checks whether adding an observation to another cluster would result in a smaller

within-cluster variation while taking the resulting mean into account. Using one algorithm or the other is often a personal choice (Morissette and Chartier, 2013) but recent works tend to favor Hartigan's algorithm over Lloyd's (Slonim *et al.*, 2013; Telgarsky and Vattani, 2010), by proving that local solutions found by the Hartigan k-means are a subset of Lloyd's with improved results and faster convergence.

Improving the computation time

Almost 60 years after the first appearance of the k-means, many people still use it in its simplest form. Most statistical programming languages have an implementation of the k-means and while some of them, like `R` (R Development Core Team, 2008), directly give the option to use Lloyd's, MacQueen's or Hartigan's version, some others, like `Matlab` (MATLAB, 2010) require to tune different parameters instead.

Keeping the basic idea does not mean that computing optimization cannot be implemented. The use of GPUs and the CUDA (NVIDIA Corporation, 2007) interface recently led to massively parallel versions that improved the computation time by two orders of magnitude (Farivar *et al.*, 2008; Li *et al.*, 2010).

Alsabti *et al.* (1997) and Pelleg and Moore (1999) propose to increase the computation speed by finding the closest cluster to an observation using k-d trees, a known structure to accelerate the search for nearest neighbours (Bentley, 1975).

Improving the accuracy

A major flaw of the k-means, as noticed by those who published it, is its locally optimum solution and there is no guaranty that the final result is the global optimum. A common revision is to run the algorithm multiple times with different initializations and keep the best result found.

More elaborate techniques have been proposed, like Phanendra Babu and Narasimha Murty (1993) and Hall *et al.* (1999) who used a genetic algorithm for the initialization.

Hochbaum and Shmoys (1985) developed a simple initialization technique in which each center is selected to be as far away as possible from previously selected centers. The idea is that centers should not be close to one another at first for a good optimization.

With the same approach but more recently, Arthur and Vassilvitskii (2007) introduced the `k-means++` algorithm which consist of the addition of a probabilistic initialization step to Lloyd's algorithm. Starting with one observation as a cluster center, other centers are sampled successively with a probability proportional to the distance to previous centers. This method is similar to Hochbaum and Shmoys (1985) but centers are initialized in a probabilistic way.

This gives more flexibility of initialization and they show this method improves not only the accuracy but also the speed of the k-means.

However, the initialization step using `k-means++` becomes relatively slow for large datasets as it requires to check all the data at each new initialization. Bahmani *et al.* (2012) solves this issue with what they call `k-means‖`, a parallel version of `k-means++` leading to faster initialization as well as less iterations of Lloyd's algorithm.

Instead of looking at the initialization step to solve the problem of locally optimum solution, the fuzzy (or soft) k-means clustering (Dunn, 1973) allows an observation to belong to different clusters at once, with weights proportional to one over the distance. The means of each cluster is then calculated as a weighted mean of all observations. This approach gives more flexibility and provides results that show great improvement in image processing (Ahmed *et al.*, 2002).

The initialization of the k-means is one of the major issues that this algorithm faces. Moreover, the k-means does not provide any indication about the optimal number of clusters, nor does it take into account the distribution of data.

While other clustering techniques possess such features, some variations of the k-means still offer interesting properties. The Euclidean distance in the within-cluster variation can be replaced by other dissimilarity distance that fit better the data distribution, like the Mahalanobis distance which takes the covariance between variables into account.

Concerning the choice of the number of clusters k, we keep it fixed in this thesis but other methods implement tools to select an appropriate number of cluster. The x-means algorithm (Pelleg and Moore, 2000) uses the Bayesian Information Criterion (BIC) to estimate the number of clusters k while iterating through the data. The BIC is an approximation of the Minimum Description Length (Rissanen, 1978) used in data compression. The GAP statistic (Tibshirani *et al.*, 2001) is another valid technique for the choice of k.

Despite all these improvements, the k-means remains massively used in its primal form. Its simplicity and speed are the two major assets and new techniques focus mostly on improving either results or computational complexity.

2.3 Bayesian clustering

Bayesian clustering is closely related to density-based clustering. Instead of optimizing the marginal likelihood with respect to the partitioning, the clustering $\mathbf{d}$ itself is seen as random variable and the maximization is made upon the posterior likelihood $f(\mathbf{d}|\mathbf{Y})$.

According to the Bayes' rule $f(\mathbf{d}|\mathbf{Y}) = f(\mathbf{Y}|\mathbf{d})f(\mathbf{d})/f(\mathbf{Y})$, where $f(\mathbf{Y}|\mathbf{d})$ is the marginal distribution and $f(\mathbf{d})$ is the prior distribution over possible partitions. Since $f(\mathbf{Y})$ does not

depend on clustering $\mathbf{d}$, the posterior is maximized using $f(\mathbf{d}|\mathbf{Y}) \propto f(\mathbf{Y}|\mathbf{d})f(\mathbf{d})$.

$f(\mathbf{Y}|\mathbf{d})$ is usually obtained by integrating over a parametric family $\boldsymbol{\theta}$ with respect to some distribution $f(\boldsymbol{\theta})$

$$f(\mathbf{Y}|\mathbf{d}) = \int f(\mathbf{Y}|\mathbf{d},\boldsymbol{\theta})f(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} = \prod_{c=1}^{k} \int \prod_{\{\mathbf{y}_i|d_i=c\}} f(\mathbf{y}_i|\boldsymbol{\theta}_c)f(\boldsymbol{\theta}_c)\mathrm{d}\boldsymbol{\theta}_c \tag{2.10}$$

$f(\mathbf{d})$ is a prior distribution over potential partitions. For the sake of simplicity, uniform distribution is often assumed but more elaborate priors have been proposed (Ewens, 1972; Pitman and Yor, 1997; Crowley, 1997; McCullagh and Yang, 2006).

Once the conditional distributions are known, we can sample from the posterior distribution. Markov Chain Monte Carlo methods, such as the Metropolis-Hasting (Metropolis *et al.*, 1953; Hastings, 1970) or the Gibbs sampling (Geman and Geman, 1984) techniques, can converge slowly, see Marin *et al.* (2005) for more details.

Among other Bayesian clustering techniques, we find Autoclass (Hanson *et al.*, 1991; Cheeseman *et al.*, 1996), or more recently latent Dirichlet allocation (Blei *et al.*, 2003; Li and McCallum, 2006).

CHAPTER 3

METHODOLOGY

In Chapter 2 we reviewed a wide range of methods for cluster analysis. Each of these methods presenting pros and cons that are subject to the nature of the dataset. This chapter will focus on two subjects: the k-means, and mixture clustering. Despite its age, 30 years of computing optimization has kept the k-means algorithm one of the most popular clustering algorithm both for its simplicity and its computational efficiency. We show how mixture clustering allows for more distributional specificity and why the k-means is related to fitting mixtures. Mixture clustering is also related to Bayesian clustering and our k-means-inspired algorithm is motivated by Bayesian clustering. We look at the performance when comparing two groupings in Section 3.1. Section 3.2 presents the details of the k-means algorithm and its flaws. In Section 3.3, the core of the mixture model and its application in clustering are explained and linked to the k-means. Bayesian clustering is introduced in Section 3.4.

3.1 Comparing clustering performances

Indicator for the quality of a given clustering is non unique. Some methods such as the within-cluster dissimilarity or the Rand index (Rand, 1971) provide useful comparison tools.

3.1.1 Observations

For simplicity, consider identical objects with no missing values. Let $\mathbf{y}_i \in \mathbb{R}^p$ be one of such objects, $i = 1, \ldots, n$.

In general, the feature does not need to be a continuous variable, but can instead be:

– Categorical: such as 0/1, a color, the sex of a person...

– Discrete: $\mathbb{N}$, such as age, number of items purchased...

– Bounded: $\mathbb{R}^+$, such as price, percentage...

Assuming that our data are real values is not as restrictive as it seems, since many problems can be transformed to fit this context. Centering features (and sometimes scaling after careful studies) is appropriate in many applications.

3.1.2 Within-cluster dissimilarity

The *within-cluster dissimilarity* is a measurement of the efficiency of a clustering if we refer to Hastie's definition of cluster analysis (Hastie *et al.*, 2009, p. 501) as "*grouping or*

segmenting a collection of objects into subsets or clusters, *such that those within each cluster are more closely related to one another than objects assigned to different clusters*". The goal here is to make sure that objects within the same group are closer to one another than they are to objects in other groups. This *closeness* is measured by what we call the *dissimilarity* between objects.

If $\mathbf{y}_i$ is the vector associated to the i-th object, we note the dissimilarity measure between a pair of observations $s(\mathbf{y}_i, \mathbf{y}_{i'})$ and we define the total dissimilarity S_T measured over all pairs of objects

$$S_T = \sum_{i=1}^{n} \sum_{i'=1}^{n} s(\mathbf{y}_i, \mathbf{y}_{i'}) \tag{3.1}$$

Suppose an observation $\mathbf{y}_i$ is allocated to cluster c, then observations within the same cluster $\mathbf{y}_{i'}$ are the ones for which $d_{i'} = c$ and those in other clusters have $d_{i'} \neq c$. This separation allows us to decompose the total dissimilarity into the dissimilarity within all clusters $S_W(\mathbf{d})$ and the dissimilarity between points that are not in the same clusters $S_B(\mathbf{d})$

$$S_T = S_W(\mathbf{d}) + S_B(\mathbf{d}), \tag{3.2}$$

in which

$$S_W(\mathbf{d}) = \sum_{c=1}^{k} \sum_{\{i|d_i=c\}} \sum_{\{i'|d_{i'}=c\}} s(y_i, y_{i'}), \tag{3.3}$$

$$S_B(\mathbf{d}) = \sum_{c=1}^{k} \sum_{\{i|d_i=c\}} \sum_{\{i'|d_{i'}\neq c\}} s(y_i, y_{i'}). \tag{3.4}$$

Although S_T is fixed for a given set of objects. Both S_W the within-cluster dissimilarity and S_B the between-cluster dissimilarity depend on the clustering $\mathbf{d}$. Then with more dissimilarity between clusters, the dissimilarity within these cluster decreases. In other words, maximizing S_B is equivalent to minimizing S_W. In this configuration, comparing two partitions can be achieved by comparing the within-cluster dissimilarity they induce for a dataset. The best clustering is defined by

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmin}} \;\; S_W(\mathbf{d}). \tag{3.5}$$

In practice, S_W is computed using the definition of dissimilarity between two observations $s(\mathbf{y}_i, \mathbf{y}_{i'})$. From now on, we consider the squared Euclidean distance as the measure of

dissimilarity

$$s(\mathbf{y}_i, \mathbf{y}_{i'}) = \sum_{j=1}^{p} (y_{ij} - y_{i'j})^2 = \|\mathbf{y}_i - \mathbf{y}_{i'}\|^2. \tag{3.6}$$

The choice of the Euclidean distance is justified by its good computational properties, its easy parallel implementation, and its connection to the log-likelihood of the Gaussian distribution.

The within-cluster dissimilarity can be averaged over cluster in a simpler form

$$S_W(\mathbf{d}) = \sum_{c=1}^{k} \sum_{\{i|d_i=c\}} \|\mathbf{y}_i - \boldsymbol{\mu}_c\|^2, \tag{3.7}$$

where $\boldsymbol{\mu}_c = \bar{y}_{c_i}$ is the mean of observations in cluster c.

The mean $\boldsymbol{\mu}_c$ can be treated as an additional parameter in S_W. We will see how this incorporates into the k-means algorithm and the EM algorithm in Sections 3.2 and 3.3.

Minimizing S_W is a complex task. The optimization is made over a large discrete space of cardinality called the Stirling number of the second kind $B(n, k)$, the number of groupings of n observations into k clusters

$$B(n, k) = \frac{1}{k!} \sum_{c=1}^{k} (-1)^{k-c} \begin{pmatrix} k \\ c \end{pmatrix} c^n. \tag{3.8}$$

The lower bound of $B(n, k)$ (Rennie and Dobson, 1969) is given by

$$L(n, k) = \frac{1}{2}(k^2 + k + 2)k^{n-k-1} - 1. \tag{3.9}$$

The number of clusters is usually small compared to the number of observations. Considering less than $\frac{n}{2}$ clusters leads to an exponential number of groupings, too large to consider the complete exploration of the space.

3.1.3 Rand index

The Rand index (Rand, 1971) and its adjusted version (Hubert and Arabie, 1985) measure the similarity between two clustering. It can be used as an evaluation method for a clustering problem, under the assumption that the true grouping is known.

The Rand index is the proportion of pairs of observations that are in the same cluster or belong to different clusters in both partitions. If $\mathbf{d}$ and $\mathbf{d}'$ are two partitions of a same

dataset Y, then the Rand index RI is

$$RI(\mathbf{d}, \mathbf{d}') = \frac{2}{n(n-1)} \sum_{i<j} \left\{ \mathbb{I}(d_i = d_j, d'_i = d'_j) + \mathbb{I}(d_i \neq d_j, d'_i \neq d'_j) \right\}, \tag{3.10}$$

where $\mathbb{I}$ is the indicator function. The RI takes values between 0 and 1, 0 meaning that no pairing is identical in the two partitions, 1 meaning that both partitions are exactly the same.

The adjusted Rand index takes into account the fact that some pairings may be identical in two partitions by chance. It computes the expected index and subtracts it to the usual Rand index. This result is scaled to stay between -1 (for completely different clusterings) and $+1$ (for identical clusterings).

3.2 k-means clustering

This section presents the k-means algorithm and its link to the within-cluster dissimilarity. Its performance in terms of computation and results are also studied and discussed at the end of this section.

3.2.1 General idea

The k-means algorithm aims to construct clusters by updating an initial partition. One step of the k-means consists in two sub-steps:

1. Assignment step: allocate each observation to the closest cluster (the cluster with the closest center).
2. Update step: update the center of all clusters with the new allocations.

This process is repeated until convergence.

3.2.2 Algorithm

The pseudo-code for the Lloyd-Forgy k-means algorithm is presented below.

```
1: while change == TRUE do
2:   change == FALSE
3:   //Assignment step
4:   for i = 1 to N do
5:     min[i] = +∞; cluster = cluster[i];
6:     for c = 1 to K do
7:       if dist(y[i], μ[c]) < min[i] then
8:         cluster[i] = c; min[i] = dist(y[i], μ[c]);
9:       end if
10:    end for
11:    change = change OR (cluster ≠ cluster[i]);
12:  end for
13:  //Update step
14:  for c = 1 to K do
15:    μ[c] = mean(y[i] ∈ c);
16:  end for
17: end while
```

Algorithm 1 k-means pseudocode

3.2.3 Within-cluster dissimilarity minimization

We define the within-cluster dissimilarity S_W

$$S_W(\mathbf{d}, \boldsymbol{\mu}) = \sum_{c=1}^{k} \sum_{\{i|d_i=c\}} \|\mathbf{y}_i - \boldsymbol{\mu}_c\|^2, \tag{3.11}$$

where $\|\mathbf{y}_i\|^2 = \sum_{j=1}^{p} y_{ij}^2$.

Theorem 1 *The k-means algorithm returns a clustering that locally minimizes the within-cluster dissimilarity.*

Proof The minimization is a consequence of the decrease of the within-cluster dissimilarity after each step, using coordinate descent. One coordinate is $\underset{\sim}{\mathbf{d}}$ and another is $\boldsymbol{\mu}$.

Assignment step: coordinate $\underset{\sim}{\mathbf{d}}$.

Let $\mathbf{y}_i$ be an observation during the assignment step, d_i and d'_i its cluster before and after this step, S_W and S'_W the within-cluster dissimilarity before and after the re-assignment

$$S'_W - S_W = \|\mathbf{y}_i - \boldsymbol{\mu}_{d'_i}\|^2 - \|\mathbf{y}_i - \boldsymbol{\mu}_{d_i}\|^2.$$

The new cluster being chosen so that $d'_i = \underset{c}{\operatorname{argmin}} \|\mathbf{y}_i - \boldsymbol{\mu}_c\|^2$, it is straightforward to see that $S'_W \leq S_W$.

Update step: coordinate $\boldsymbol{\mu}_c$.

Let S_W be the within-cluster dissimilarity and $\mathbf{d}'$ the new clustering at the beginning of the update step

$$S_W(\mathbf{d}', \boldsymbol{\mu}) = \sum_{c=1}^{k} \sum_{\{i|d'_i=c\}} \|\mathbf{y}_i - \boldsymbol{\mu}_c\|^2.$$

Then,

$$\frac{\partial S_W}{\partial \boldsymbol{\mu}_c} = \frac{\partial}{\partial \boldsymbol{\mu}_c} \sum_{\{i|d'_i=c\}} \|\mathbf{y}_i - \boldsymbol{\mu}_c\|^2 = \sum_{\{i|d'_i=c\}} 2(\mathbf{y}_i - \boldsymbol{\mu}_c)^\top = -2n_c\boldsymbol{\mu}_c^\top + 2 \sum_{\{i|d'_i=c\}} \mathbf{y}_i^\top$$

$$\Rightarrow \frac{\partial S_W}{\partial \boldsymbol{\mu}_c} = 0 \Leftrightarrow \boldsymbol{\mu}_c = \frac{1}{n_c} \sum_{\{i|d'_i=c\}} \mathbf{y}_i = \bar{\mathbf{y}}'_c.$$

Updating the centers to the new cluster means minimizes the within-cluster dissimilarity.

If $S_W^{(t)}$ is the within-cluster dissimilarity at the beginning of step (t), we showed that $S_W^{(t+1)} \leq S_W^{(t)}$. Since the number of possible partitionings is finite, S_W is bounded below by its minimum.

$S_W^{(t)}$ is a bounded and decreasing sequence and therefore converges to a local minimum.

End of proof.

3.2.4 Discussion

Time complexity

The assignment step consists in the measure of k distances for each of the n observations. For p dimensions, the computation of the distance between two observations is of order $O(p)$. Therefore, the assignment step has a time complexity of $O(nkp)$.

The update step is the computation of the cluster centers for a total of n observations, resulting in a $O(np)$ time complexity.

Therefore, the computation time of a whole step of the k-means algorithm is of order $O(nkp)$.

The complexity of k-means is probably its strongest asset. Being linear in terms of the number of observations n, dimensions p, and the number of clusters k makes the fitting relatively fast in terms of these parameters.

Initialization

The k-means suffers from converging to a local optimum. The convergence proven in Theorem 1 is only local and strongly depends on the initialization.

The usual initialization process consists in the random selection of k observations as initial centers. To avoid the problem of convergence to a "bad" local minimum, the algorithm can be run multiple times with different initializations.

The `k-means++` (Arthur and Vassilvitskii, 2007) is a recent initialization technique. This process aims to select centers to be further apart and avoid the initialization of multiple centers in the same natural cluster. The process is as follows:

1. Select the first center at random.
2. For center j, define the probability of sampling an observation $\mathbf{y}_i$ as a new center with a probability p_i proportional to $\min_{1 \leq c \leq j-1} \text{dist}(\mathbf{y}_i, \boldsymbol{\mu}_c)$.

3.3 Mixture clustering

In this section, we present the mixture distribution problem and the associated algorithm, the EM algorithm. We also discuss how this method aims to minimize the within-cluster dissimilarity and its relationship with k-means.

3.3.1 General idea

Mixture clustering is a common approach in which clusters are seen as distinct probability distributions. Observations in cluster c, denoted by $\mathbf{Y}_c$, are distributed according to $\mathbf{Y}_c \sim F_c$, where F_c is a distribution function. The F_c is usually a parametric class, parametrized with $\boldsymbol{\mu}_c$.

The complete likelihood is given by the product of the density functions for all clusters

$$L(\mathbf{d}, \boldsymbol{\mu}) = f(\mathbf{Y} \mid \mathbf{d}, \boldsymbol{\mu}) = \prod_{c=1}^{k} f(\mathbf{Y}_c \mid \boldsymbol{\mu}_c) = \prod_{c=1}^{k} \prod_{\{i \mid d_i = c\}}^{n_c} f(\mathbf{y}_{c_i} \mid \boldsymbol{\mu}_c), \tag{3.12}$$

where $\mathbf{d}$ is the vector of labels and $\boldsymbol{\mu}$ the list of parameters $\boldsymbol{\mu}_c$ for each distribution.

This likelihood can be rewritten as

$$L(\mathbf{d}, \boldsymbol{\mu}) = f(\mathbf{Y} \mid \mathbf{d}, \boldsymbol{\mu}) = \prod_{i=1}^{n} \prod_{c=1}^{k} f(\mathbf{y}_i \mid \boldsymbol{\mu}_c)^{\mathbb{I}(d_i=c)}. \tag{3.13}$$

If we consider the marginal likelihood over possible clustering, with a prior distribution $\Pr(d_i = c) = p_c$ for allocation of observations to class c

$$L(\boldsymbol{\mu}) = f(\mathbf{Y} \mid \boldsymbol{\mu}) = \prod_{i=1}^{n} \sum_{c=1}^{k} p_c f(\mathbf{y}_i \mid \boldsymbol{\mu}_c). \tag{3.14}$$

Gaussian distributions with identical variance is a common choice. The univariate Gaussian mixtures are easily extended to higher dimensions

$$L(\mu_1, \ldots, \mu_k, p_1, \ldots, p_k, \sigma^2) = \prod_{i=1}^{n} \sum_{c=1}^{k} \frac{p_c}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2}(y_i - \mu_c)^2 \right\}. \tag{3.15}$$

The maximization of the likelihood with respect to mixture parameters is a non-convex problem. The numerical solution is an optimal clustering in the sense that each observation is allocated to the cluster with the highest likelihood.

3.3.2 Expectation-Maximization (EM) algorithm

Non-convex problems are difficult to handle. Most solutions are only locally optimal as there is no guarantee that the numerical minimum (or maximum) is a global solution.

The Expectation-Maximization algorithm also returns a local maximum of the mixture likelihood but through indirectly maximizing the likelihood. The EM comes from the two steps of the algorithm:

– **Expectation (E) step:** Compute the expected value over possible clustering given observations $\mathbf{Y}$ and current parameters $\boldsymbol{\mu}^{(t)}$

$$\mathbb{E}_{\mathbf{d}|\mathbf{Y},\boldsymbol{\mu}^{(t)}} \left[\log L(\mathbf{d}, \boldsymbol{\mu}^{(t)}) \right] = \sum_{i=1}^{n} \sum_{c=1}^{k} \Pr(d_i = c|\mathbf{Y}, \boldsymbol{\mu}^{(t)}) \log \left[f(\mathbf{y}_i \mid \boldsymbol{\mu}^{(t)}) \right], \tag{3.16}$$

where $\Pr(d_i = c|\mathbf{Y}, \boldsymbol{\mu}^{(t)})$ is given by Bayes Theorem and $\mathbb{E}$ stands for mathematical expectation.

$$\Pr(d_i = c|\mathbf{y}_i, \boldsymbol{\mu}^{(t)}) = \frac{\Pr(\mathbf{y}_i|d_i = c, \boldsymbol{\mu}_c^{(t)}) \Pr(d_i = c)}{\Pr(\mathbf{y}_i|\boldsymbol{\mu}_c^{(t)})} = \frac{f(\mathbf{y}_i|\boldsymbol{\mu}_c^{(t)}) p_c^{(t)}}{\sum_c p_c^{(t)} f(\mathbf{y}_i|\boldsymbol{\mu}_c^{(t)})}. \tag{3.17}$$

– **Maximization (M) step:** Maximize the previous quantity with respect to the parameters

$$\boldsymbol{\mu}^{(t+1)} = \underset{\boldsymbol{\mu}}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{d}|\mathbf{Y},\boldsymbol{\mu}^{(t)}}\left[\log L(\mathbf{d},\boldsymbol{\mu}^{(t)})\right]. \tag{3.18}$$

In the case of univariate Gaussian mixtures, these two steps have straightforward solutions

– **Expectation step:**

$$\begin{aligned}\mathbb{E}_{\mathbf{d}|\mathbf{Y},\boldsymbol{\mu}^{(t)}}\left[\log L(\mathbf{d},\boldsymbol{\mu}^{(t)})\right] = &\sum_{i=1}^{n}\sum_{c=1}^{k}\Pr(d_i=c|\mathbf{Y},\boldsymbol{\mu}^{(t)})\times \\ &\left(\log p_c^{(t)} - \frac{1}{2}\log\sigma^2 - \frac{1}{2}\log(2\pi) - \frac{(y_i-\mu_c)^2}{2\sigma^2}\right),\end{aligned} \tag{3.19}$$

where

$$\Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)}) = \frac{\frac{p_c^{(t)}}{\sqrt{2\pi{\sigma^{(t)}}^2}}\exp\left(-\frac{(y_i-\mu_c^{(t)})^2}{2{\sigma^{(t)}}^2}\right)}{\sum_c \frac{p_c^{(t)}}{\sqrt{2\pi{\sigma^{(t)}}^2}}\exp\left(-\frac{(y_i-\mu_c^{(t)})^2}{2{\sigma^{(t)}}^2}\right)}. \tag{3.20}$$

– **Maximization step:** Derivatives with respect to the different parameters have closed-form solutions.

$$\begin{cases} (1) \quad p_c^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)}) & \text{for } 1\le c\le k, \\ (2) \quad \mu_c^{(t+1)} = \frac{\sum_{i=1}^{n} y_i \Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)})}{\sum_{i=1}^{n}\Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)})} & \text{for } 1\le c\le k, \\ (3) \quad {\sigma^{(t+1)}}^2 = \frac{\sum_{i=1}^{n}\sum_{c=1}^{k}(y_i-\mu_c^{(t+1)})^2\Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)})}{\sum_{i=1}^{n}\Pr(d_i=c|y_i,\boldsymbol{\mu}^{(t)})}. \end{cases} \tag{3.21}$$

The EM algorithm for multivariate normal mixtures has the same steps with vector expressions.

This method ensures that the likelihood increases after each step (Dempster *et al.*, 1977). There are multiple stopping criteria based on the absolute or relative variation of a parameter. For a parameter $\boldsymbol{\theta}$, the absolute variation stopping criterion is reached when the difference between two successive values of this parameter is below a chosen threshold $\|\boldsymbol{\theta}^{(t+1)}-\boldsymbol{\theta}^{(t)}\| \le \delta$. The log-likelihood, the maximum absolute variation for means or the variance are possible parameters as the variation between steps will decrease as the algorithm converges. The relative variation, $\|\boldsymbol{\theta}^{(t+1)}-\boldsymbol{\theta}^{(t)}\|/\|\boldsymbol{\theta}^{(t)}\|$, is also used as a stopping criterion but is not recommended for means as there is no guaranty that these means will not reach 0 at some point of the

process.

3.3.3 Within-cluster dissimilarity minimization

Theorem 2 *The maximization of the complete likelihood of a Gaussian mixture is equivalent to the minimization of the within-cluster dissimilarity.*

Proof

The within-cluster dissimilarity S_W is given by

$$S_W(\mathbf{d}, \boldsymbol{\mu}) = \sum_{c=1}^{k} \sum_{\{i|d_i=c\}}^{n_c} (y_{c_i} - \mu_c)^2 .$$

Maximization of the complete likelihood is to be made on the following value:

$$L(\mathbf{d}, \boldsymbol{\mu}) = \prod_{c=1}^{k} \prod_{\{i|d_i=c\}}^{n_c} f\left(y_{c_i} \mid \boldsymbol{\mu}_c\right), \text{with } y_{c_i} \overset{iid}{\sim} \mathcal{N}\left(\mu_c, \sigma^2\right).$$

Hence

$$\begin{aligned} L(\mathbf{d}, \boldsymbol{\mu}) &= \prod_{c=1}^{k} \prod_{\{i|d_i=c\}}^{n_c} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y_{c_i} - \mu_c\right)^2\right) \\ &\propto \exp\left(-\frac{1}{2\sigma^2} \sum_{c=1}^{k} \sum_{\{i|d_i=c\}}^{n_c} \left(y_{c_i} - \mu_c\right)^2\right) \\ &\propto \exp\left(-\frac{1}{2\sigma^2} S_W(\mathbf{d}, \boldsymbol{\mu})\right). \end{aligned}$$

We assume σ^2 to be known. The maximization of the complete likelihood over the clustering $\mathbf{d}$ and the associated means $\boldsymbol{\mu}$ is therefore equivalent to the minimization of the within-cluster dissimilarity over these parameters.

End of proof.

3.3.4 Discussion

Time complexity

The expectation step consists in the computation of $\Pr(d_i = c|\mathbf{Y}, \boldsymbol{\mu}^{(t)})$ for $1 \leq i \leq n$ and $1 \leq c \leq k$. The time complexity of this step is $O(nkT)$, where T is the complexity of

computing $\Pr(d_i = c|\mathbf{Y}, \boldsymbol{\mu}^{(t)})$. According to (3.20), this can be computed in $O(p)$ operations, where p is the number of dimensions.

The maximization step requires $O(nk)$ operations for the update of p_c, $O(nkp)$ for the update of $\boldsymbol{\mu}_c$ and $O(nkp)$ operations for the update of σ.

The total time complexity of one step of the EM algorithm is $O(nkp)$.

Relationship with the k-means

The EM algorithm has the same time complexity as k-means. It also suffers from the initialization problem and is often disregarded in favor of k-means as they tend to provide similar results.

The k-means is a special case of the EM algorithm.

Theorem 3 *The maximization of the likelihood* (3.15) *coincides with the k-means if* $p_1 = \cdots = p_k = \frac{1}{k}$ *and* $\sigma^2 \to 0$.

Proof Consider the maximization of the following likelihood:

$$L(\mu_1, \ldots, \mu_k, p_1, \ldots, p_k, \sigma^2) = \prod_{i=1}^{n} \sum_{c=1}^{k} \frac{p_c}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_c)^2\right\}.$$

In the case in which the prior probabilities are all equal ($p_1 = \cdots = p_k = \frac{1}{k}$) and the within-class variance tends towards zero ($\sigma^2 \to 0$), the Expectation-Maximization (EM) algorithm coincides with the k-means algorithm.

The equivalence between the two algorithms corresponds to the equivalence of their two steps respectively:

- The assignment step of the k-means and the expectation step of the EM algorithm.
- The update step of the k-means and the maximization step of the EM algorithm.

***E*-step $\Leftrightarrow$ Assignment step**

Assuming the previous conditions at the beginning of the Expectation step of the EM algorithm:

$$p_{ic} = P(d_i = c \mid y_i) = \frac{\frac{p_c}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_c)^2\right\}}{\sum_{j=1}^{k} \frac{p_j}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_j)^2\right\}} = \frac{\exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_c)^2\right\}}{\sum_{j=1}^{k} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_j)^2\right\}}.$$

For each y_i, there is a unique cluster d_i so that $d_i = \underset{c}{\operatorname{argmin}}\,[(y_i - \mu_c)^2]$,

$$p_{ic} = \frac{\exp\left\{-\frac{1}{2\sigma^2}\left((y_i - \mu_c)^2 - (y_i - \mu_{d_i})^2\right)\right\}}{\sum_{j=1}^{k} \exp\left\{-\frac{1}{2\sigma^2}\left((y_i - \mu_j)^2 - (y_i - \mu_{d_i})^2\right)\right\}}.$$

Since $d_i = \underset{c}{\operatorname{argmin}} \left[(y_i - \mu_c)^2\right]$

$$\begin{cases} (y_i - \mu_j)^2 - (y_i - \mu_{d_i})^2 > 0 & \text{if } j \neq d_i, \\ (y_i - \mu_j)^2 - (y_i - \mu_{d_i})^2 = 0 & \text{if } j = d_i. \end{cases}$$

Hence

$$\begin{aligned} \lim_{\sigma^2 \to 0} \quad & \exp\left[-\frac{1}{2\sigma^2}\left((y_i - \mu_j)^2 - (y_i - \mu_{d_i})^2\right)\right] = \begin{cases} 0 & \text{if } j \neq d_i, \\ 1 & \text{if } j = d_i, \end{cases} \\ \Rightarrow \lim_{\sigma^2 \to 0} \quad & \sum_{j=1}^{k} \exp\left[-\frac{1}{2\sigma^2}\left((y_i - \mu_j)^2 - (y_i - \mu_{d_i})^2\right)\right] = 1 \\ \Rightarrow \lim_{\sigma^2 \to 0} \quad & p_{ic} = \mathbb{I}(c = d_i). \end{aligned}$$

The E-step of the EM algorithm consists in allocating each sample to the closest cluster mean with a probability one, which is equivalent to the assignment step of the k-means algorithm.

***M*-step $\Leftrightarrow$ Update step**

When fitting a Gaussian Mixture in the EM-algorithm, the means of each mixture are updated according to the following equation

$$\mu_c = \frac{\sum_{i=1}^{n} p_{ic} y_i}{\sum_{i=1}^{n} p_{ic}}.$$

Since $p_{ic} = \mathbb{I}(c = d_i)$, we get

$$\mu_c = \frac{1}{n_c} \sum_{\{i | d_i = c\}}^{n_c} y_i = \bar{y_c},$$

which is the mean of the samples currently allocated to the cluster c.

End of proof.

3.4 Bayesian clustering

Our proposed algorithm, the no-mean algorithm takes the Gaussian mixture model in a Bayesian framework and solves this new problem with the same time complexity as the k-means.

In the EM algorithm, the clustering parameter $\mathbf{d}$ is integrated out during the expectation step and the final clustering is obtained from the final mixture components. In Bayesian

clustering, the component parameters are integrated out in the marginal likelihood

$$f(\mathbf{Y}|\mathbf{d}) = L(\mathbf{d}) = \int L(\mathbf{d}, \boldsymbol{\mu})\mathrm{d}\boldsymbol{\mu} = \prod_{i=1}^{n} \int \prod_{\{i|d_i=c\}}^{n_c} f(\mathbf{y}_i|\boldsymbol{\mu}_c) f(\boldsymbol{\mu}_c)\mathrm{d}\boldsymbol{\mu}_c \tag{3.22}$$

which can be used to find the posterior $f(\mathbf{d}|\mathbf{Y}) \propto f(\mathbf{Y}|\mathbf{d})f(\mathbf{d})$, from which we can sample the final grouping. The prior distribution of the grouping parameter $f(\mathbf{d})$ will be assumed to be uniform in this thesis, but can also be changed to favor clusters of similar sizes (McCullagh and Yang, 2006; Heard *et al.*, 2006) . The assumption of uniformity simplifies the expression of the posterior $f(\mathbf{d}|\mathbf{Y}) \propto f(\mathbf{Y}|\mathbf{d})$.

To be useful in practice, the integral (3.22) also needs to be analytically tractable. If not, we may sample from the full posterior

$$f(\mathbf{d}, \boldsymbol{\mu} \mid \mathbf{y}) \propto f(\mathbf{y} \mid \mathbf{d}, \boldsymbol{\mu}) f(\boldsymbol{\mu}) = \prod_{c=1}^{k} \prod_{\{i|d_i=c\}}^{n_c} f(y_{ci} \mid \boldsymbol{\mu}_c) f(\boldsymbol{\mu}_c) \tag{3.23}$$

using a Markov chain Monte Carlo method, such as the Gibbs sampler.

Gibbs sampling (Gelfand *et al.*, 1992) is a technique that uses the conditional distributions of different sets of parameters $\boldsymbol{\theta}$. If we denote $\boldsymbol{\theta}_{-i}$ the set of parameters without θ_i, the Gibbs sampler is as follows

1. Choose an initial set $\boldsymbol{\theta}^{(0)}$.
2. Set $t = 0$.
3. For all parameters θ_i, sample $\theta_i^{(t+1)} \sim f(\theta_i|\boldsymbol{\theta}_{-i} = \boldsymbol{\theta}_{-i}^{(t)})$
4. Increment t and return to step 3.

For a large number of iterations, the vector of parameters become independent of the initial value and is sampled from the objective joint distribution $\boldsymbol{\theta}^{(t)} \sim f(\boldsymbol{\theta})$.

For Bayesian clustering, the parameters are the clusters means $\boldsymbol{\mu}_c$ and the grouping $\mathbf{d}$. We consider the Gaussian mixture model with a Gaussian prior on $\boldsymbol{\mu}_c$

$$\begin{cases} \mathbf{y}_i|d_i = c, \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(\boldsymbol{\mu}_c, \sigma^2\mathbf{I}_{p\times p}), \\ \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(0, \tau^2\mathbf{I}_{p\times p}). \end{cases} \tag{3.24}$$

where $\mathbf{I}_{p\times p}$ is the identity matrix and $\mathcal{N}_p$ is the p-variate normal distribution.

The conditional probability on the grouping d_i is discrete over possible values $c \in \{1, \ldots, k\}$

and given by

$$\begin{aligned}\Pr(d_i = c|\mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{Y}) &= \frac{\Pr(d_i = c, \boldsymbol{\mu}, \mathbf{Y}, \mathbf{d}_{-i})}{\Pr(\boldsymbol{\mu}, \mathbf{Y}, \mathbf{d}_{-i})} \\ &= \frac{\Pr(\mathbf{y}_i|d_i = c, \boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i}) \Pr(d_i = c, \boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i})}{\Pr(\boldsymbol{\mu}, \mathbf{Y}, \mathbf{d}_{-i})}.\end{aligned} \tag{3.25}$$

We have

$$\begin{aligned}\Pr(d_i = c, \boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i}) &= \Pr(\boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i}|d_i = c) \Pr(d_i = c) \\ &= \Pr(\boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i}) \Pr(d_i = c),\end{aligned} \tag{3.26}$$

since the whole dataset without the i-th observation and the grouping of this observation are independent. The prior distribution $f(\mathbf{d})$ is assumed to be uniform discrete, which means the term $\Pr(d_i = c, \boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i})$ is constant. The same reasoning leads to the independence of $\Pr(\boldsymbol{\mu}, \mathbf{Y}, \mathbf{d}_{-i})$ with respect to c.

We deduct

$$\Pr(d_i = c|\mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{Y}) \propto \Pr(\mathbf{y}_i|d_i = c, \boldsymbol{\mu}, \mathbf{Y}_{-i}, \mathbf{d}_{-i}) = \Pr(\mathbf{y}_i|d_i = c, \boldsymbol{\mu}_c), \tag{3.27}$$

leading to

$$\Pr(d_i = c|\mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{Y}) \propto \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y}_i - \boldsymbol{\mu}_c)^\top(\mathbf{y}_i - \boldsymbol{\mu}_c)\right\}. \tag{3.28}$$

We compute the conditional probability for the cluster means $\boldsymbol{\mu}_c$

$$\Pr(\boldsymbol{\mu}_c|\boldsymbol{\mu}_{-c}, \mathbf{Y}, \mathbf{d}) = \Pr(\boldsymbol{\mu}_c|\mathbf{Y}_c) = \frac{\Pr(\boldsymbol{\mu}_c, \mathbf{Y}_c)}{\Pr(\mathbf{Y}_c)} = \frac{\Pr(\mathbf{Y}_c|\boldsymbol{\mu}_c) \Pr(\boldsymbol{\mu}_c)}{\int \Pr(\mathbf{Y}_c|\boldsymbol{\mu}_c) \Pr(\boldsymbol{\mu}_c) \mathrm{d}\boldsymbol{\mu}_c}. \tag{3.29}$$

Using the distribution in (3.24), we deduct

$$\boldsymbol{\mu}_c|\boldsymbol{\mu}_{-c}, \mathbf{Y}, \mathbf{d} \sim \mathcal{N}_p\left(\frac{\bar{y}_c}{1 + \frac{\sigma^2}{n_c\tau^2}}, \frac{\tau^2}{1 + \frac{n_c\tau^2}{\sigma^2}}\mathbf{I}_{p\times p}\right). \tag{3.30}$$

Once again, the k-means can be seen as particular case of the Gibbs sampler on the Gaussian mixture model.

Theorem 4 *The Gibbs sampling approach for a Gaussian mixture model with a Gaussian prior distribution for cluster means and a uniform prior distribution for cluster labels, see* (3.24)*, coincides with the k-means algorithm if* $\tau^2 \to \infty$ *and* $\sigma^2 \to 0$.

Proof

The Gibbs sampling is also a two-step approach:

1. Sampling the cluster labels according to: $\Pr(d_i = c \mid \mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{y})$.
2. Sampling the means according to: $\Pr(\mu_c \mid \mathbf{d}, \mathbf{y})$.

These two steps are respectively equivalent to the assignment step and the update step of the k-means algorithm when $\sigma^2 \to 0$ and $\tau^2 \to \infty$.

Cluster sampling $\Leftrightarrow$ assignment step

The probability of a sample y_i to being sampled in a cluster c is given by:

$$\Pr(d_i = c \mid \mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{y}) \propto \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_c)^2\right\}.$$

Since $\sum_{c=1}^{k} \Pr(d_i = c \mid \mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{y}) = 1$:

$$\Pr(d_i = c \mid \mathbf{d}_{-i}, \boldsymbol{\mu}, \mathbf{y}) = \frac{\exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_c)^2\right\}}{\sum_{j=1}^{k} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu_j)^2\right\}}.$$

The proof of Theorem 3 shows that, as $\sigma^2 \to 0$, this probability is equal to one for the cluster for which the mean is the closest to the sample value, as in the assignment step of the k-means algorithm.

Mean sampling $\Leftrightarrow$ Update step

The conditional probability for the sampling of the cluster means is:

$$\boldsymbol{\mu}_c | \mathbf{Y}, \mathbf{d} \sim \mathcal{N}\left(\frac{\bar{y}_c}{1 + \frac{\sigma^2}{n_c\tau^2}}, \frac{1}{\frac{1}{\tau^2} + \frac{n_c}{\sigma^2}}\mathbf{I}_{p\times p}\right).$$

As $\sigma^2 \to 0$ and $\tau^2 \to \infty$:

$$\frac{\sigma^2}{n_c\tau^2} \to 0 \Rightarrow \frac{\bar{y}_c}{1 + \frac{\sigma^2}{n_c\tau^2}} \to \bar{y}_c.$$

$$\frac{1}{\frac{1}{\tau^2} + \frac{n_c}{\sigma^2}} \to 0 \Rightarrow \frac{1}{\frac{1}{\tau^2} + \frac{n_c}{\sigma^2}}\mathbf{I}_{p\times p} \to \mathbf{0}.$$

This probability tends towards a degenerate probability distribution at $\bar{y}_c$, the mean of the samples contained in this cluster, as in the update step of the k-means algorithm.

End of proof.

However, if we can get a closed-form solution for the integral in (3.22), there is no need to sample from the cluster means. The marginal likelihood for the Gaussian model in (3.24)

is analytically tractable and is a multivariate Gaussian

$$f(\mathbf{Y}|\mathbf{d}) = \prod_{i=1}^{n} \int \prod_{i|d_i=c}^{n_c} f(\mathbf{y}_i|\boldsymbol{\mu}_c) f(\boldsymbol{\mu}_c) \mathrm{d}\boldsymbol{\mu}_c = \prod_{c=1}^{k} f(\mathbf{Y}_c|\mathbf{d}), \tag{3.31}$$

where $f(\mathbf{Y}_c|\mathbf{d}) \sim \mathcal{N}_{n_c}(\mathbf{0}, \boldsymbol{\Omega})$, in which $\mathcal{N}_p$ denotes a p-variate Gaussian distribution and $\boldsymbol{\Omega}$ is a uniform positive semi-definite covariance matrix of size n_c (the number of observations in cluster c) with diagonals $\sigma^2 + \tau^2$ and off-diagonals τ^2.

Since $f(\mathbf{d}|\mathbf{Y}) \propto f(\mathbf{Y}|\mathbf{d})$, a Gibbs sampler can be applied with

$$\Pr(d_i = c|\mathbf{d}_{-i}, \mathbf{Y}) \propto f(\mathbf{Y}|d_1, \ldots, d_i = c, \ldots, d_n). \tag{3.32}$$

CHAPTER 4

NO-MEAN ALGORITHM AND ITS VARIANTS

Using the marginal likelihood to sample from the posterior distribution $f(\mathbf{d}|\mathbf{Y})$ ensures that in the long run the achieved grouping is independent of initialization. This chapter presents the no-mean algorithm, a Bayesian clustering algorithm. In Section 4.1, we introduce the core of the no-mean algorithm as a clustering algorithm. The first variant of the no-mean as a variable selection method is explained in Section 4.2. Section 4.3 presents another variation of the algorithm that even estimates the number of clusters.

4.1 No-mean algorithm for clustering

This section sets the context in which we apply the no-mean algorithm and detail its content.

4.1.1 General idea

Suppose the clusters are normally distributed with a Gaussian prior distribution on cluster means

$$\begin{cases} \mathbf{y}_i|d_i = c, \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(\boldsymbol{\mu}_c, \sigma^2\mathbf{I}_{p\times p}), \\ \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(0, \tau^2\mathbf{I}_{p\times p}). \end{cases} \tag{4.1}$$

Then, the marginal likelihood $L(\mathbf{d}) \equiv f(\mathbf{Y}|\mathbf{d})$ is the product of multivariate normal densities for each cluster $\mathbf{Y}_c \sim \mathcal{N}_{n_c}(\mathbf{0}, \boldsymbol{\Omega})$, where

$$\boldsymbol{\Omega} = (\sigma^2 + \tau^2) \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho \\ \rho & \cdots & \rho & 1 \end{pmatrix} \text{ and } \rho = \frac{\tau^2}{\sigma^2 + \tau^2}. \tag{4.2}$$

Since $f(\mathbf{d}|\mathbf{Y}) = \frac{f(\mathbf{Y}|\mathbf{d})f(\mathbf{d})}{f(\mathbf{Y})}$, for a uniform prior on the grouping distribution $f(\mathbf{d})$, the posterior distribution is proportional to the marginal $f(\mathbf{d}|\mathbf{Y}) \propto f(\mathbf{Y}|\mathbf{d})$.

The no-mean algorithm is a Gibbs sampler on this distribution combined with simulated annealing. Simulated annealing is a technique used to gradually reduce the randomness of

a process by decreasing the variance during the sampling. This means pushing $\sigma^2 \to 0$ and $\tau^2 \to \infty$ to slowly settle the search towards the "optimal" region. The convergence is ensured for a logarithmic rate change of the variance (Mitra *et al.*, 1985). However, this considerably slows the process and in practice we observed that exponential rates converge rapidly while keeping satisfactory results.

Iterations of the no-mean algorithm are as follow:

1. Initialize clusters $\mathbf{d}^{(0)}$ by allocating observations randomly to k groups as cluster means.
2. Initialize model parameters $\sigma^{2(0)}$ and $\tau^{2(0)}$.
3. For each data point, sample its new cluster using the conditional $\Pr(d_i^{(t+1)} = c|\mathbf{d}_{-i}^{(t)}, \mathbf{Y})$, with parameters $\sigma^{2(t)}$ and $\tau^{2(t)}$.
4. Update $\sigma^{2(t+1)} = \beta\sigma^{2(t)}$ and $\tau^{2(t+1)} = \alpha\tau^{2(t)}$, where α and β are the annealing rates. We suggest $\beta = \frac{1}{\alpha}$.
5. Repeat steps 3 and 4.

4.1.2 Algorithm

The pseudo-code for the no-mean algorithm is presented below.

```
1: for iter in 1 : numberOfIterations do
2:   for i in permutation(N) do
3:     for c = 1 to K do
4:       prob[c] = Pr(d[i] = c);
5:     end for
6:     cluster[i] = c with probability prob[c];
7:   end for
8:   sigmaSquare = sigmaSquare/alpha;
9:   tauSquare = alpha * tauSquare;
10: end for
```

Algorithm 2 No-mean

4.1.3 Computation time

The main obstacle to Bayesian clustering is the complexity of computing the marginal likelihood. If we consider the time complexity of one step of the no-mean algorithm, we easily find that it is of order $O(nkT)$ where T is the complexity of computing $\Pr(d_i = c|\mathbf{d}_{-i}, \mathbf{Y})$.

For our algorithm to be competitive with k-means, we need to compute this probability in order of $O(p)$ operations, where p is the number of variables.

We denote $L_j(\mathbf{d})$ the marginal likelihood for the j-th parameter, associated to j-th variable

$$f(\mathbf{d}|\mathbf{y}_j) \propto L_j(\mathbf{d}) = f(\mathbf{y}_j|\mathbf{d}) = \prod_{c=1}^{k} f(\mathbf{y}_{cj}|\mathbf{d}), \tag{4.3}$$

where

$$f(\mathbf{y}_{cj}|\mathbf{d}) = \prod_{c=1}^{k} \frac{1}{(2\pi)^{\frac{n_c}{2}}} \frac{1}{\sqrt{|\mathbf{\Omega}_{n_c}|}} \exp\left\{-\frac{1}{2}{\mathbf{y}_{cj}}^{\top}\mathbf{\Omega}_{n_c}^{-1}\mathbf{y}_{cj}\right\}. \tag{4.4}$$

The determinant $|\mathbf{\Omega}_{n_c}|$ and inverse $\mathbf{\Omega}_{n_c}^{-1}$ have simple closed-form expressions. If we denote $\rho = \frac{\tau^2}{\sigma^2+\tau^2}$, then

$$\mathbf{\Omega}_{n_c}^{-1} = \frac{1}{\tau^2}\left\{\mathbf{I}_{n_c\times n_c} - \frac{\rho}{1+(n_c-1)\rho}\mathbf{J}_{n_c}\right\}, \tag{4.5}$$

$$|\mathbf{\Omega}_{n_c}| = \frac{{\sigma^2}^{n_c}}{1-\rho}\left(1+(n_c-1)\rho\right), \tag{4.6}$$

where $\mathbf{I}_{n_c\times n_c}$ is the identity matrix of size n_c and $\mathbf{J}_{n_c}$ is a matrix of ones of size n_c.

Denote the log-likelihood $\ell_j(\mathbf{d}) = \log L_j(\mathbf{d})$,

$$\begin{aligned}
\ell_j(\mathbf{d}) &= \sum_{c=1}^{k} -\frac{n_c}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{\Omega}_{n_c}|) - \frac{1}{2}{\mathbf{y}_{cj}}^{\top}\mathbf{\Omega}_{n_c}^{-1}\mathbf{y}_{cj} \\
&= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{c=1}^{k}\left\{n_c\log\sigma^2 - \log(1-\rho) + \log(1+(n_c-1)\rho)\right. \\
&\qquad \left. + \frac{1}{\sigma^2}\mathbf{y}_{cj}^{\top}\left(\mathbf{I}_{n_c\times n_c} - \frac{\rho}{1+(n_c-1)\rho}\mathbf{J}_{n_c}\right)\mathbf{y}_{cj}\right\} \\
&= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 + \frac{k}{2}\log(1-\rho) \\
&\qquad - \frac{1}{2}\sum_{c=1}^{k}\left\{\log(1+(n_c-1)\rho)\frac{\mathbf{y}_{cj}^{\top}\mathbf{y}_{cj}}{\sigma^2} - \frac{\rho}{\sigma^2}\frac{\mathbf{y}_{cj}^{\top}\mathbf{J}_{n_c}\mathbf{y}_{cj}}{1+(n_c-1)\rho}\right\} \\
&= -\frac{n}{2}\log(2\pi\sigma^2) + \frac{k}{2}\log(1-\rho) - \frac{1}{2}\sum_{c=1}^{k}\left\{\log(1+(n_c-1)\rho) + \frac{S_{c,j}}{\sigma^2} - \frac{\rho}{\sigma^2}\frac{B_{c,j}^2}{1+(n_c-1)\rho}\right\},
\end{aligned}$$

where

$$S_{c,j} = \sum_{i|d_i=c} y_{i,j}^2, \tag{4.7}$$

$$B_{c,j} = \sum_{i|d_i=c} y_{i,j}. \tag{4.8}$$

Since $\sum_{c=1}^{k} S_{c,j} = \sum_{i=1}^{n} y_{i,j}^2 = \mathbf{y}_j^T \mathbf{y}_j$, the final version of the log-likelihood is

$$\begin{aligned} \ell_j(\mathbf{d}) = &- \frac{n}{2}\log(2\pi\sigma^2) + \frac{k}{2}\log(1-\rho) - \frac{\mathbf{y}_j^T\mathbf{y}_j}{2\sigma^2} \\ &- \frac{1}{2}\sum_{c=1}^{k}\left\{\log(1+(n_c-1)\rho) - \frac{\rho}{\sigma^2}\frac{B_{c,j}^2}{1+(n_c-1)\rho}\right\}. \end{aligned} \tag{4.9}$$

The marginal log-likelihood can be re-written as

$$\ell_j(\mathbf{d}) = a_j + b_j(\mathbf{d}), \tag{4.10}$$

where a_j is a constant independent of the clustering and b_j is a function of $\mathbf{d}$

$$a_j = -\frac{n}{2}\log(2\pi\sigma^2) + \frac{k}{2}\log(1-\rho) - \frac{\mathbf{y}_j^T\mathbf{y}_j}{2\sigma^2}, \tag{4.11}$$

$$b_j(\mathbf{d}) = -\frac{1}{2}\sum_{c=1}^{k}\left\{\log(1+(n_c-1)\rho) - \frac{\rho}{\sigma^2}\frac{B_{c,j}^2}{1+(n_c-1)\rho}\right\}. \tag{4.12}$$

Let consider an observation $\mathbf{y}_i$, $c_0 = d_i$ its current cluster and c_1 a potential different new cluster. We denote $\ell_j(\mathbf{d}) = \ell_j(d_1, \ldots, d_i = c_0, \ldots, d_n)$ the current likelihood, $\ell_j(\mathbf{d}_{c_0 \to c_1}) = \ell_j(d_1, \ldots, d_i = c_1, \ldots, d_n)$ the likelihood for the potential cluster, and $b_j(\mathbf{d}_{c_0 \to c_1}) = b_j(d_1, \ldots, d_i = c_1, \ldots, d_n) - b_j(d_1, \ldots, d_i = c_0, \ldots, d_n)$. Then,

$$\ell_j(\mathbf{d}_{c_0 \to c_1}) = \ell_j(\mathbf{d}) + b_j(\mathbf{d}_{c_0 \to c_1}). \tag{4.13}$$

Only 2 clusters are concerned in the changes in $b_j(\mathbf{d}_{c_0 \to c_1})$. The current cluster c_0 loses the observation $\mathbf{y}_i$ to the potential cluster c_1, which changes $B_{c_0,j}$ and $B_{c_1,j}$ to $(B_{c_0,j} - y_{i,j})$ and $(B_{c_1,j} + y_{i,j})$, and n_{c_0} and n_{c_1} to $(n_{c_0} - 1)$ and $(n_{c_1} + 1)$. Assuming that we do not move an

observation to a new cluster if it is a singleton (meaning $n_{c_0} \geq 2$), this means

$$\begin{aligned}
b_j(\mathbf{d}_{c_0 \to c_1}) = &-\frac{1}{2}\left(\log(1+(n_{c_0}-2)\rho) - \frac{\rho}{\sigma^2}\frac{(B_{c_0,j}-y_{i,j})^2}{1+(n_{c_0}-2)\rho}\right. \\
&\left. + \log(1+n_{c_1}\rho) - \frac{\rho}{\sigma^2}\frac{(B_{c_1,j}+y_{i,j})^2}{1+n_{c_1}\rho}\right) \\
&+ \frac{1}{2}\left(\log(1+(n_{c_0}-1)\rho) - \frac{\rho}{\sigma^2}\frac{B_{c_0,j}^2}{1+(n_{c_0}-1)\rho}\right. \\
&\left. + \log(1+(n_{c_1}-1)\rho) - \frac{\rho}{\sigma^2}\frac{B_{c_1,j}^2}{1+(n_{c_1}-1)\rho}\right) \\
= &\frac{1}{2}\log\left(\frac{1+(n_{c_0}-1)\rho}{1+(n_{c_0}-2)\rho}\right) + \frac{\rho}{2\sigma^2}\left(\frac{(B_{c_0,j}-y_{i,j})^2}{1+(n_{c_0}-2)\rho} - \frac{B_{c_0,j}^2}{1+(n_{c_0}-1)\rho}\right) \\
&+ \frac{1}{2}\log\left(\frac{1+(n_{c_1}-1)\rho}{1+n_{c_1}\rho}\right) + \frac{\rho}{2\sigma^2}\left(\frac{(B_{c_1,j}+y_{i,j})^2}{1+n_{c_1}\rho} - \frac{B_{c_1,j}^2}{1+(n_{c_1}-1)\rho}\right).
\end{aligned}$$

We can rewrite the likelihood for the potential cluster as follows

$$\ell_j(\mathbf{d}_{c_0 \to c_1}) = \ell_j(\mathbf{d}) + \ell_j(\mathbf{d}_{c_0 \to}) + \ell_j(\mathbf{d}_{\to c_1}), \tag{4.14}$$

where

$$\ell_j(\mathbf{d}_{c_0 \to}) = \frac{1}{2}\log\left\{\frac{1+(n_{c_0}-1)\rho}{1+(n_{c_0}-2)\rho}\right\} + \frac{\rho}{2\sigma^2}\left\{\frac{(B_{c_0,j}-y_{i,j})^2}{1+(n_{c_0}-2)\rho} - \frac{B_{c_0,j}^2}{1+(n_{c_0}-1)\rho}\right\}, \tag{4.15}$$

$$\ell_j(\mathbf{d}_{\to c_1}) = \frac{1}{2}\log\left\{\frac{1+(n_{c_1}-1)\rho}{1+n_{c_1}\rho}\right\} + \frac{\rho}{2\sigma^2}\left\{\frac{(B_{c_1,j}+y_{i,j})^2}{1+n_{c_1}\rho} - \frac{B_{c_1,j}^2}{1+(n_{c_1}-1)\rho}\right\}. \tag{4.16}$$

This representation is similar to a common problem in thermodynamics, the Boltzmann distribution. We can interpret this problem as the probability for a particle to move from one state to another. If each state c has an energy level E_c, then the probability of moving from a state c_0 to c_1 is proportional to $e^{-\frac{E_{c_1}-E_{c_0}}{kT}}$, where T is the temperature of the system and k is the Boltzmann constant. This means that the particle will more likely move to a state of lower energy, $E_{c_1} - E_{c_0}$ representing the "cost" of moving from c_0 to c_1.

In our case, $\ell_j(\mathbf{d}_{c_0 \to})$ represents the cost of moving the observation out of its original cluster c_0 and $\ell_j(\mathbf{d}_{\to c_1})$ is the cost of moving it into the cluster c_1.

Denote $\ell(\mathbf{d}) = \sum_{j=1}^{p} \ell_j(\mathbf{d})$ for all the likelihoods mentioned above, then

$$\begin{aligned}
\Pr(d_i = c_1 | \mathbf{d}_{-i}, \mathbf{Y}) &= \texttt{cst} \times e^{\ell(d_1,\ldots,d_i=c_1,\ldots,d_n)} \\
&= \begin{cases} \texttt{cst} \times e^{\ell(\mathbf{d})} & \text{if } c_1 = c_0, \\ \texttt{cst} \times e^{\ell(\mathbf{d})} e^{\ell(\mathbf{d}_{c_0 \to})} e^{\ell(\mathbf{d}_{\to c_1})} & \text{otherwise.} \end{cases}
\end{aligned} \tag{4.17}$$

If we multiply each of these probability by the same constant, they remain the same. This leads to

$$\Pr(d_i = c_1|\mathbf{d}_{-i}, \mathbf{Y}) = \begin{cases} \texttt{cst} \times e^{-\ell(\mathbf{d}_{c_0\to})} & \text{if } c_1 = c_0, \\ \texttt{cst} \times e^{\ell(\mathbf{d}\to c_1)} & \text{otherwise.} \end{cases} \tag{4.18}$$

Equivalently

$$\Pr(d_i = c_1|\mathbf{d}_{-i}, \mathbf{Y}) = \begin{cases} \frac{e^{-\ell(\mathbf{d}_{c_0\to})}}{e^{-\ell(\mathbf{d}_{c_0\to})}+\sum_{c\neq c_0} e^{\ell(\mathbf{d}\to c)}} & \text{if } c_1 = c_0, \\ \frac{e^{\ell(\mathbf{d}\to c_1)}}{e^{-\ell(\mathbf{d}_{c_0\to})}+\sum_{c\neq c_0} e^{\ell(\mathbf{d}\to c)}} & \text{otherwise.} \end{cases} \tag{4.19}$$

The computational complexity of $\Pr(d_i = c_1|\mathbf{d}_{-i}, \mathbf{Y})$ is therefore the same as $\ell(\mathbf{d}_{\to c_1}) = \sum_{j=1}^{p} \ell_j(\mathbf{d}_{\to c_1})$. If the term $B_{c_1,j}$ is saved and is updated after each step of the algorithm, equation (4.16) can be computed in a constant time, leading to a time complexity of order $O(nkp)$ for each iteration of the no-mean algorithm.

We just showed that a smart implementation of the no-mean algorithm presents the same time complexity as k-means for each iteration, where a brute force approach would have led to a much slower algorithm of order at least $O(n^2k^2p)$.

4.1.4 Online no-mean versus batch no-mean

In online no-mean, the likelihood is updated after each allocation of an observation, by updating $B_{c_1,j}, B_{c_0,j}, n_{c_1}$ and n_{c_0}. In contrast, the batch no-mean version updates these parameters after all observations have been re-allocated.

The online version converges much faster than the batch version, especially for large initial values of σ^2. Large values of σ_0^2 lead to completely random clusters for the first iterations of the algorithm. If these clusters are kept as they are for the allocation of the whole dataset (batch version), it will take more time to create groups of close observations.

4.2 No-mean algorithm for variable selection

Sometimes we use features that are irrelevant to our clustering task. If we want to cluster a group of persons randomly sampled from a population into 2 clusters, assuming this will lead to a group of men and a group of women, the height and weight of a person will have more impact than his age. For such features, data are indistinguishable and cannot be separated into different clusters.

The marginal likelihood gives the probability that our dataset is clustered according to a given partition. The idea behind variable selection for the no-mean algorithm is to compare

each variable using the marginal likelihood of the current clustering versus the marginal likelihood when all observations are in the same cluster.

The likelihood without clustering ℓ_0 is easy to compute by transposing the result from (4.9) to a single cluster

$$\begin{aligned}\ell_{0j} = &-\frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2}\log(1-\rho) - \frac{\mathbf{y}_j^T\mathbf{y}_j}{2\sigma^2} \\ &-\frac{1}{2}\left\{\log(1+(n-1)\rho) - \frac{\rho}{\sigma^2}\frac{1}{1+(n_c-1)\rho}\left(\sum_{i=1}^{n} y_{i,j}\right)^2\right\}.\end{aligned} \tag{4.20}$$

This likelihood remains constant through the whole algorithm and only needs to be computed once at the first iteration.

Computing the whole marginal likelihood for the current partition might be harder as explained in Section 4.1. The no-mean algorithm only requires to compute a part of it to get the conditional probability for the Gibbs sampler. However, the change in the likelihood from moving an observation from one cluster to another is small and even computed during the sampling step as $\ell(\mathbf{d}_{c_0\to})$ and $\ell(\mathbf{d}_{\to c_1})$.

Appendix B shows the updated pseudo-code of no-mean to take these changes into account.

Though, this method showed satisfactory results on simulated toy examples, it did not perform well on real world datasets. Results of this variant are presented in Chapter 5 and are discussed in the concluding chapter of this thesis.

4.3 No-mean algorithm for flexible number of clusters

Since the marginal likelihood is a function of the partitioning, there is no indication that the number of clusters should be fixed to k. Our algorithm could be improved into a no-k-no-mean algorithm in which an observation can be allocated to a new cluster or a cluster can be emptied.

When computing the likelihood for each potential cluster, we also compute the likelihood if the current observation is moved to a cluster of its own and sample the new cluster using the corresponding probabilities. If an object is already a singleton, we may allow this observation to merge into another cluster.

This approach tends to separate all observations into their own clusters, which minimizes the within-cluster variation. This comes from the uniform prior distribution for the grouping parameter $f(\mathbf{d})$. This assumption is reasonable for a fixed number of clusters when we assume that all clusters are equiprobable. Consider a company looking for two types of clients in its

database. They do not care too much if one group is small and the other is big. But if they try to detect different groups, they are less likely to consider a lot of small groups.

A simple remedy is to consider the prior as a probability mass on the number of clusters $f(\|\mathbf{d}\|) = \Pr(\|\mathbf{d}\| = k) = f(k)$, where k is the number of clusters for the partitioning $\mathbf{d}$. The choice of a correct prior is too vast of a problem to be solved here but this variant shows that as long as we have a way to efficiently compute the likelihood (i.e. with the same time complexity as k-means), it opens the way to many applications. Our current suggestion is a truncated Poisson distribution for $f(k)$ with the mode around k, where k is the assumed number of clusters if the k-means were to be run.

CHAPTER 5

SIMULATION AND APPLICATION

The no-mean algorithm is developed as an alternative to the k-means. This Chapter presents comparison between these two methods. Section 1 compares both algorithms for simulated data that fit the prior model of the no-mean context. An application of the no-mean algorithm on real data is made in Section 2. The variation of the no-mean for variable selection is tested in simulations in Section 3.

5.1 Simulation

The Bayesian framework relies heavily on the notion of prior distribution. We assume that the data are distributed according to a model with some knowledge *a priori*. This section describes a dataset simulated according to the prior model used for the no-mean. The performance of the no-mean algorithm is then compared to the k-means and the linearity of the time complexity of no-mean is tested on this dataset.

5.1.1 Description

Theorem 3 shows that the k-means algorithm is linked to a Gaussian mixture model

$$\begin{cases} \mathbf{y}_i | d_i = c, \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(\boldsymbol{\mu}_c, \sigma^2 \mathbf{I}_{p\times p}), \\ \boldsymbol{\mu}_c & \overset{iid}{\sim} & \mathcal{N}_p(0, \tau^2 \mathbf{I}_{p\times p}). \end{cases} \tag{5.1}$$

To study the impact of the no-mean algorithm, we decided to simulate data according to this model:

- k: the number of clusters.
- N: the number of observations per cluster.
- p: the number of variables.
- τ^2: the variance of the simulated cluster centers.
- σ^2: the variance of observations within clusters.

We start by sampling k means using $\boldsymbol{\mu}_c \overset{iid}{\sim} \mathcal{N}_p(0, \tau^2 \mathbf{I}_{p\times p})$. For each of these clusters, we sample N observations using $\mathbf{y}_i | \boldsymbol{\mu}_c \overset{iid}{\sim} \mathcal{N}_p(\boldsymbol{\mu}_c, \sigma^2 \mathbf{I}_{p\times p})$. Each simulation is a matrix of $n = kN$ observations in p dimensions.

In addition to the parameters of the simulation, the no-mean algorithm has its own

hyperparameters:

- k: the number of clusters to return.
- σ_0^2: the initial within cluster variance.
- τ_0^2 or $\rho_0 = \frac{\tau_0^2}{\tau_0^2+\sigma_0^2}$: the other initial variance.
- S: the number of iterations of the algorithm.
- α: the annealing rate.

For the rest of this section, we suppose that the number of clusters for simulation study is equal to the one that we set for the no-mean algorithm. Our study focuses on the impact of these parameters on the performance of the no-mean algorithm.

5.1.2 Design of experiments

Experimental design aims at evaluating the impact of various parameters (or factors) on the outcome of an experiment. For known significant factors, each factor is considered at three levels: low, medium and high, also referred to as a 3^k experimental design. Then all combination of these factors are tested. Let say that we have two factors A and B and that we denote each level of these factors by -, 0 and +, then we test the result for $\{(A^-B^-), (A^-B^0), (A^-B^+), (A^0B^-), (A^0B^0), (A^0B^+), (A^+B^-), (A^+B^0), (A^+B^+)\}$. In general, for k factors, this methods requires 3^k tests.

From now on, we remove ambiguity with the notation k of k-means and 3^k experimental design by referring to the number of experiments with n.The goal of this method is to find the combination of factors that maximizes the outcome. If $X_1, \ldots, X_n$ are the n considered factors, we fit the outcome y with a second order polynomial over our 3^n observations

$$y = \beta_0 + \sum_{j=1}^{n} \beta_j x_j + \sum_{j=1}^{n} \sum_{j'=1}^{n} \beta_{j,j'} x_j x_{j'}. \tag{5.2}$$

In our case, Table 5.1 lists the factors and the associated values.

Table 5.1 Experimental design factor values for the no-mean algorithm.

	Value		
Factor	low	medium	high
σ_0^2	0.01	0.1	1
τ_0^2	0.1	1	10
S	5	20	35
α	1.2	3.1	5

These values are all symmetric (in log-scale for σ_0^2 and τ_0^2), so that they can be scaled and centered as -1, 0 and +1 for the fitting.

We simulate a dataset with the parameters presented in Table 5.2.

Table 5.2 Simulation parameters for experimental design of the no-mean algorithm.

Parameter	Value
k	50
N	100
σ^2	1
τ^2	250

The no-mean algorithm is run twice for each of the $3^4 = 81$ combinations of factors on our simulated dataset with the same initialization. Since the original clustering is known, we evaluate its average performance using both the Rand index, and the within-cluster dissimilarity, S_W in (3.7).

The outcome y in model (5.2) is either the Rand index or the within-cluster dissimilarity. Higher values of the Rand index and lower values of S_W are favorable.

The optimized outcomes are reported in Table 5.3.

Table 5.3 Optimized factors for experimental design.

Factor	Within-cluster dissimilarity		Rand index	
σ_0^2	0.98	high	1.51	high
τ_0^2	-0.99	low	-1.34	low
S	1.03	high	1.23	high
α	0.89	high	1.26	high

Table 5.3 indicates that the no-mean algorithm works better in a high number of iterations S, a high annealing rate α, a high initial within-cluster variance σ_0^2 and a low initial between-cluster variation τ_0^2. These results are for the most part in agreement with our intuition. Having such initial variances is equivalent to starting with high temperatures for the simulated annealing, meaning that observations can easily move between clusters. A large number of iterations was also expected to have a positive impact, since it allows the algorithm to have a finer convergence. The surprising outcome in this study is the high value of the annealing rate. The higher the annealing rate, the faster the algorithm becomes deterministic, with higher chances of converging to a sub-optimal solution.

A careful study of the results highlights a contrasting conclusion. The no-mean algorithm performs uniformly well when $\sigma_0^2 = \tau_0^2$, regardless of the other parameters, and performs uniformly worse for all other combinations of factors. Having almost constant results over different factors means that the results of the optimization are to be considered carefully.

Further investigation of the impact of hyperparameters, in comparison with k-means and for different configurations is presented in the next section.

5.1.3 Random Search for Hyperparameter Optimization

The number of factors impacting the performance of the no-mean algorithm is quite high, both in terms of hyperparameters and the problem parameters. We listed 9 of such parameters in Section 5.1.1. Doing a full grid-search over these factors in a similar fashion to the experimental design of Section 5.1.2 would lead to $3^9 \approx 20,000$ runs, and only if we decide to test for three levels.

We opted for a random search optimization (Bergstra and Bengio, 2012) over the following parameters

- Simulation set up
  - k: the number of clusters.
  - N: the number of observations per cluster.
  - p: the number of variables.
  - τ^2: the variance of the simulated cluster centers.
- No-mean algorithm
  - σ_0^2: the initial within cluster variance.
  - $\rho_0 = \frac{\tau_0^2}{\tau_0^2+\sigma_0^2}$.
  - S: the number of iterations of the algorithm.
  - α: the annealing rate.

The number of clusters for the algorithm is the same as the number of clusters in the simulation set up. Since the problem can be scaled, the variance within clusters for the simulation σ_0^2 was set to 1. For each of the parameters, 3 to 7 possible values were chosen.

For a given dataset and combination of parameters, each algorithm was run 10 times, with 10 different initial clusters. This process was repeated 430 times, the maximum number of simulations we could run on our computation system.

Performance of each run is measured by the within-cluster dissimilarity S_W and the Rand index RI. If we denote $S_W^{(k)}$, $S_W^{(n)}$, $S_W^{(0)}$, $RI^{(k)}$ and $RI^{(n)}$ the performance indicators for k-means $^{(k)}$, no-mean $^{(n)}$ or the original distribution $^{(0)}$, then

$$\begin{cases} S_W^+ & = \frac{S_W^{(k)} - S_W^{(n)}}{S_W^{(0)}} \\ RI^+ & = RI^{(n)} - RI^{(k)} \end{cases} . \tag{5.3}$$

are two indicators that take positive values when the no-mean algorithm performs better than the k-means algorithm. S_W^+ takes values between $-\infty$ and $+\infty$ and RI^+ between -1

and +1. The highest the value, the better the performance of no-mean.

For each set of parameters, the indicators are computed using the best and the average performance of each algorithm over 10 runs. Positive values of the indicators are indicated by black circles in Figures, negative values are indicated by gray squares.

As an example, Figure 5.1 shows the performance indicators for different values of the annealing rate α. The worst results for the no-mean algorithm are for $\alpha = 1$, meaning that the algorithm does not perform any simulated annealing. This result is coherent with our expectation since the simulated annealing is what allows the algorithm to converge to an optimal solution. Removing the annealing step maintains the allocation at a high level of randomness throughout the process. From now on, we will only consider simulations for which $\alpha > 1$, since it is always the case in practice.

Figure 5.1 (with the exception of $\alpha = 1$) shows that the no-mean algorithm is likely to produce a better result compared to the k-means. When it happens, the improvement on the performance is usually more important than when the k-means produces better results.

To assess whether the no-mean algorithm increases the quality of the clustering, we compute the percentage of simulations for which S_W^+ or RI^+ are positive (see Table 5.4). When only considering the best performance over 10 initializations, the no-mean algorithm outperforms k-means about 70% of the time. This number rises to 75% if we consider the average performance over the 10 initializations. Rerunning the k-means multiple times is a technique to improve its performance and it makes sense that the increase in performance is smaller when we consider the best case out of 10.

Table 5.4 Percentage of successful improvement of no-mean.

	$S_W^+ > 0$	$RI^+ > 0$
Best	71.88%	68.44%
Average	77.19%	74.38%

The k-means is known for performing poorly for high dimensional datasets. The "curse of dimensionality" (Bellman, 1956) states that for high dimensional problems, the distance between points tend to be uniformly distributed. It becomes harder to distinguish clusters as the distance between observations and cluster centers can easily shift in favor of one cluster or another and fall into a local minimum. Figure 5.2 presents the results for a various number of dimensions p. These graphs suggest that the no-mean algorithm outperforms k-means more and more as the number of variables increases.

Cluster separability is another factor of interest. Since all observations within a cluster are sampled with variance $\sigma^2 = 1$, we use $\frac{\tau^2 p}{k}$ as a measurement of cluster separability. τ^2 is natural as it represents the variance of cluster means. For higher dimensions, these means

will be naturally pushed further apart from one another, which decreases the probability of overlapping clusters. On the contrary, a high number of clusters increases this probability. Figure 5.3 shows the performance indicators for this parameter.

This example highlights the difference between the within-cluster dissimilarity and the Rand index as the performance indicators. The no-mean seems to largely outperform k-means in terms of S_W^+ for high separability, although it is not the case when we consider the Rand index instead. This happens because, for high separability, both algorithm are doing poorly compared to the true clustering in terms of S_W, which amplifies the difference between the two methods in S_W^+.

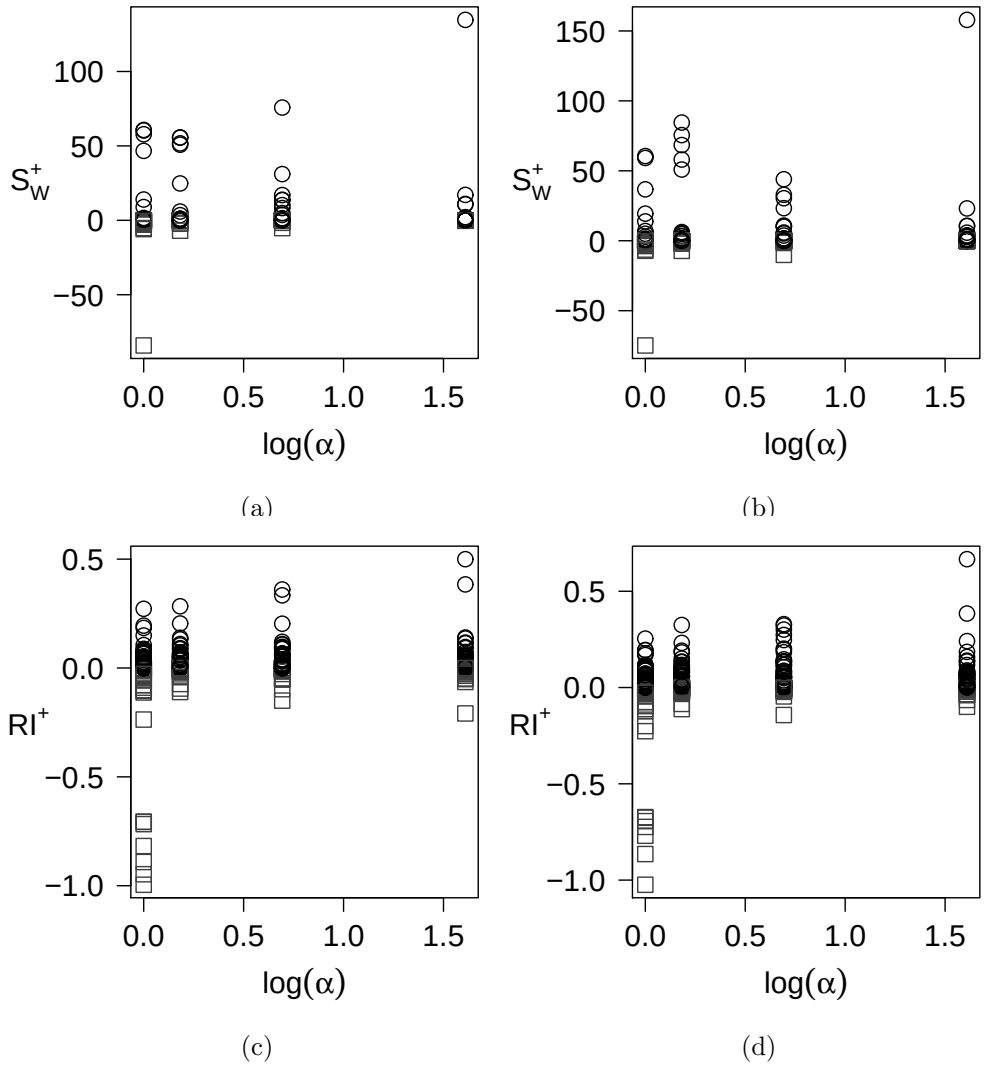


Figure 5.1 No-mean performance as a function of the annealing rate α
(a) S_W^+ on best performance – (b) S_W^+ on average performance
(c) RI^+ on best performance – (d) RI^+ on average performance.

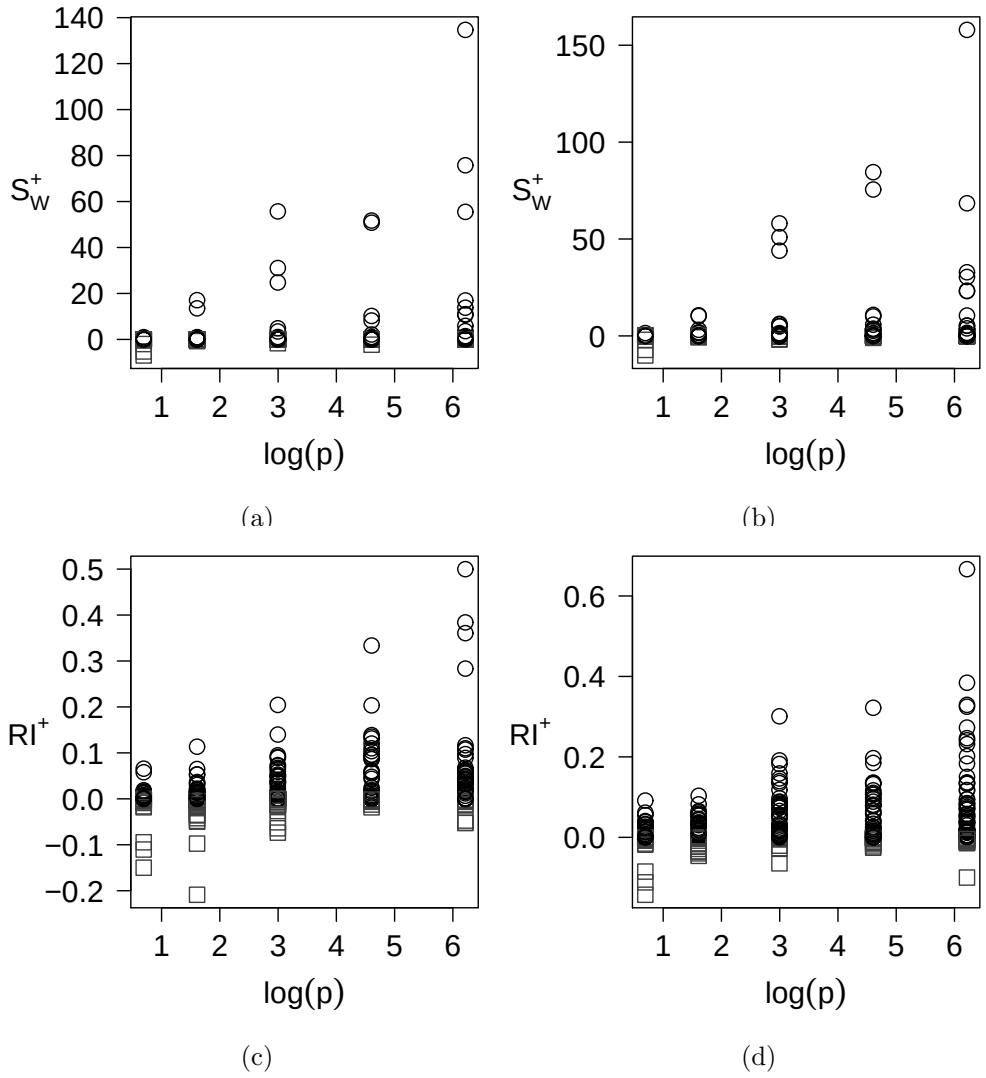


Figure 5.2 No-mean performance as a function of the number of dimensions p
(a) S_W^+ on best performance – (b) S_W^+ on average performance
(c) RI^+ on best performance – (d) RI^+ on average performance.

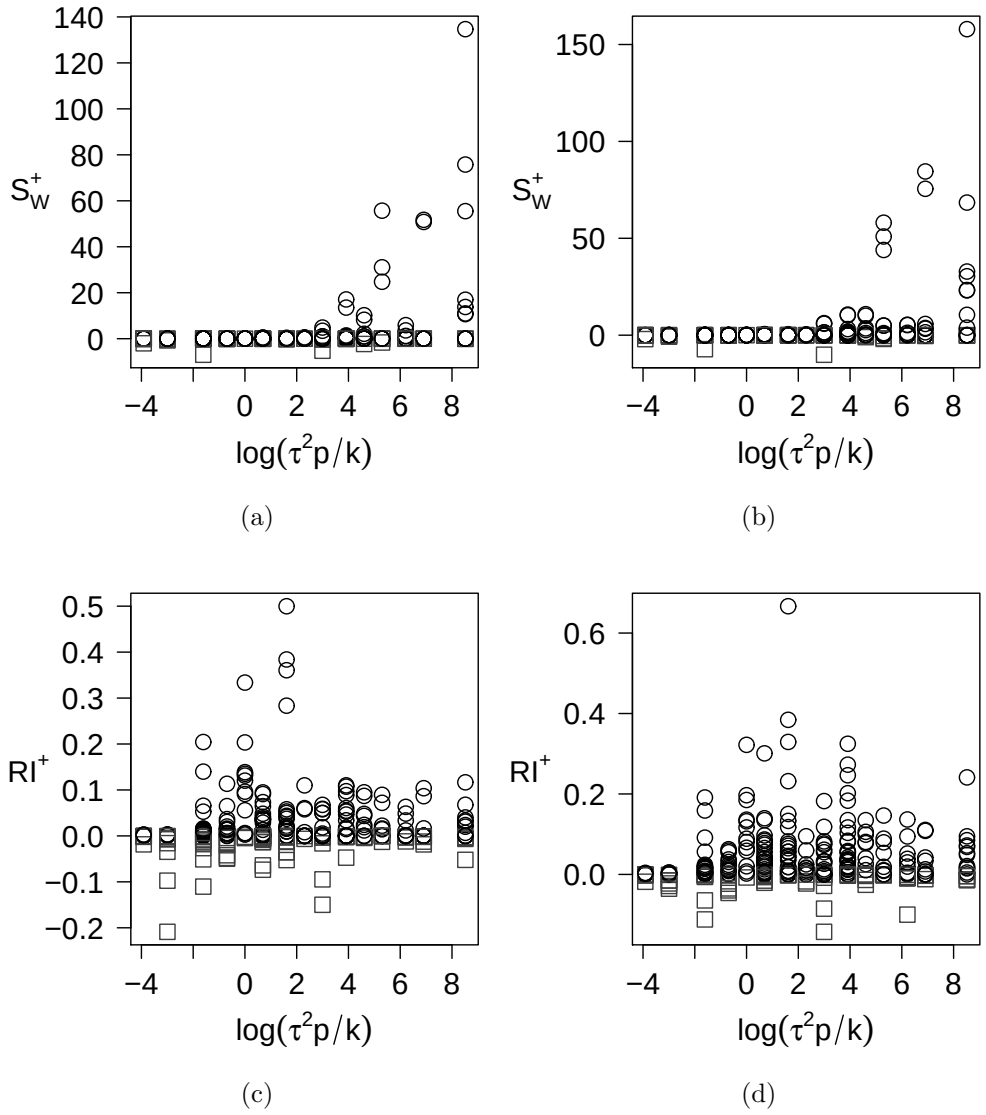
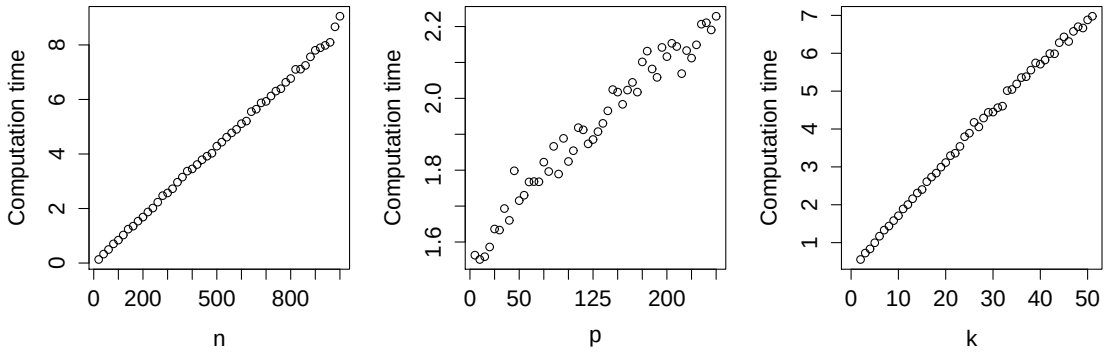


Figure 5.3 No-mean performance as a function of cluster separability $\frac{\tau^2 p}{k}$
(a) S_W^+ on best performance – (b) S_W^+ on average performance
(c) RI^+ on best performance – (d) RI^+ on average performance.

5.1.4 Time complexity

The time complexity of the k-means in terms of n, k and p has already been assessed multiple times. Regarding the no-mean algorithm, we established that one iteration of our implementation has a computational complexity of order $O(nkp)$.

Figure 5.4 shows the computation time for 40 steps of the no-mean algorithm as a function of n, p and k. We see that the computational time is a linear function of n, p and k, which confirms the computational complexity of $O(nkp)$. It is worth pointing out that the relationship between the computation time and p is in fact affine, meaning that doubling the number of dimensions will not double the computation time, thanks to the efficiency of column-wise operations in `R`.



(a) (b) (c)

Figure 5.4 Computation time of the no-mean algorithm for (a) n observations (b) p variables (c) k clusters.

5.2 Application

5.2.1 Problem description

To evaluate the performance of the no-mean algorithm, we applied it to existing clustering problems and compared the results with those obtained with the k-means and the `k-means++` (see Section 3.2.4). The datasets are those used in Arthur and Vassilvitskii (2007) for the comparison of the `k-means++` initialization technique to the usual k-means.

The first dataset is the Cloud dataset (Bache and Lichman, 2013). The detection and classification of clouds in satellite images is a key issue in the better understanding of the climatic impact of the atmospheric cloud layer. The observations are super-pixels of 16×16, extracted from a 512×512 pixels satellite image in visible light and infrared. The dataset consists in 1024 of these super-pixels from which 10 features are extracted:

– Visible light: mean, maximum, minimum, average, contrast, entropy, second angular momentum.
– Infrared intensity of the image: mean, max, min.

The second dataset is the first 10% of the KDD Cup 1999 Intrusion Dataset (Bache and Lichman, 2013). Classification of computer networks attacks helps preventing further intrusions and detecting suspicious behaviors. The observations are simulated connections to a military network environment (for a total of 949021 observations). Each observation has 34 features, such as the number of attempts, the length of the connection or the transmission error rate.

For these two datasets, all three methods are run 20 times each for $k = 10$, $k = 25$ and $k = 50$ on the standardized cloud dataset (as in the original study) and on the centered intrusion dataset. For each trial, the k-means and the no-mean use the same initialization. Performance is measured by the average and the best within-cluster dissimilarity S_W for each technique.

5.2.2 Data summary

Figure 5.5 shows the scatterplot for the cloud dataset. This dataset presents two characteristics:

– Variables 4 to 7 are strongly correlated.
– Apart for V_1 and V_6, no clear clusters seem to arise from this scatterplot.

The box-plots in Figure 5.6 represents the centered observations of the intrusion dataset without outliers. Figure (a) shows the different scales and Figure (b) the shift induced by outliers. Variables 2, 16, 17 and 26 have larger scales than the other parameters, making it harder to fit to our Bayesian model (3.24). When taking out the outliers, most observations are not centered around 0. This induces some issues since being an outlier for one parameter can force one observation out of what would otherwise be a good cluster.

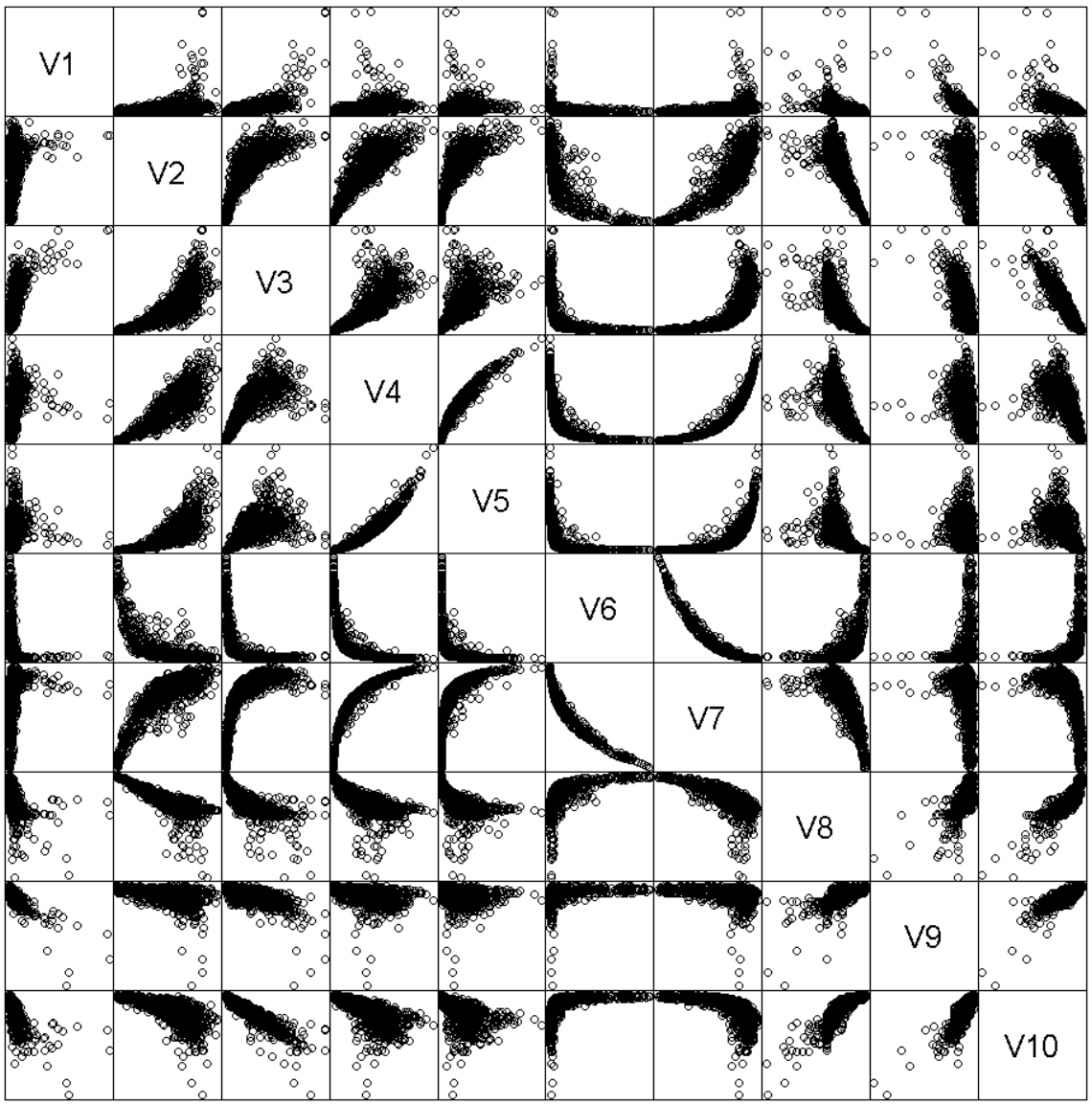
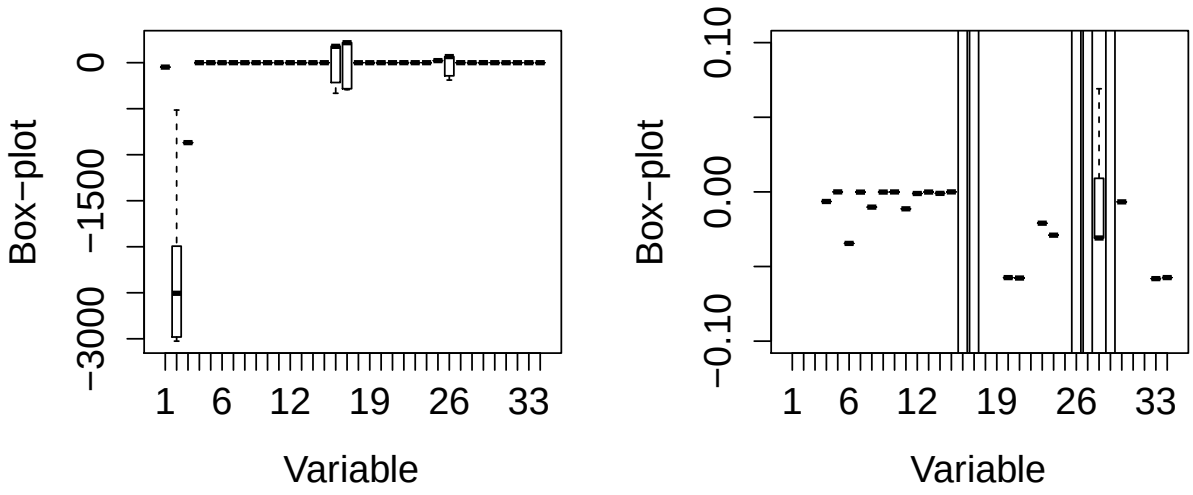


Figure 5.5 Scatterplot matrix for the cloud dataset.



(a) (b)

Figure 5.6 Box-plots for the intrusion dataset: (a) Full dataset (without outliers) (b) Zoomed in.

5.2.3 Results

The no-mean algorithm shows substantial improvements compared to both k-means and `k-means++` on the Cloud dataset (see Table 5.5). Both the average and the best performance are increased when using no-mean, and more drastically as the number of clusters increases, up to 7 times for $k = 50$.

Table 5.5 Experimental results on the Cloud dataset.

	Average S_W			**Best** S_W		
k	k-means	`k-means++`	no-mean	k-means	`k-means++`	no-mean
10	1555.8 ($\pm$13.4)	1554.3 ($\pm$14.6)	1543.7 ($\pm$12.0)	1503.7	1503.2	1503.1
25	912.74 ($\pm$12.5)	876.76 ($\pm$21.9)	363.20 ($\pm$18.5)	886.14	821.55	286.83
50	619.52 ($\pm$14.2)	555.77($\pm$10.7)	119.88 ($\pm$14.6)	556.22	530.28	81.75

Regarding the intrusion dataset (see Table 5.6), the no-mean algorithm is able to outperform the other techniques for $k = 25$ or $k = 50$ but only for its best performance. The size of the dataset and the large number of duplicates (about two thirds of the dataset are duplicates) make it harder for the no-mean algorithm to converge. This factors combined to the fact that outliers have a lot of impact on the parameters distribution (see Section 5.2.2) do not provide the best conditions for both the k-means and the no-mean algorithm.

Table 5.6 Experimental results on the Intrusion dataset.

	Average S_W ($\times 10^{13}$)			**Best** S_W ($\times 10^{13}$)		
k	k-means	`k-means++`	no-mean	k-means	`k-means++`	no-mean
10	16.0 ($\pm$0.37)	5.75 ($\pm$2.51)	46700 ($\pm$0.08)	12.5	1.49	46700
25	15.2 ($\pm$0.002)	0.506 ($\pm$0.04)	28000 ($\pm$10270)	15.2	0.351	0.018
50	10.8 ($\pm$3.00)	0.221 ($\pm$0.06)	14000 ($\pm$9600)	0.59	0.076	0.002

5.3 Variable Selection

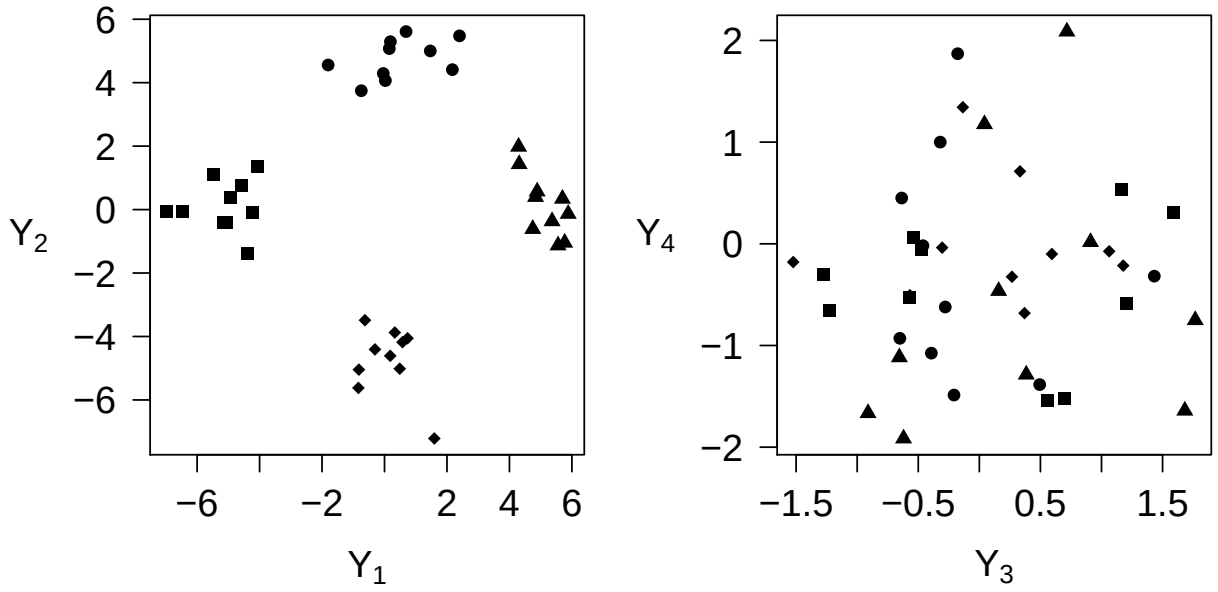
5.3.1 Description

One dataset is simulated according to the following protocol:

– The data is clustered into 4 distinct clusters in the first two dimensions centered around $(-5, 0), (5, 0), (0, -5)$ and $(0, 5)$, see Figure 5.7 (a).

– The other two dimensions consist in random gaussian noise, see Figure 5.7 (b).

The first simulation compares the log-likelihood (4.9) for each variable in different situations:



(a) (b)

Figure 5.7 Simulation: (a) Data in clustered dimensions (b) Data in random dimensions.

- With the true clustering (True **d**).
- With a random clustering (Random **d**).
- With each observation in its own cluster ($d_i = i$).
- With all observations in one cluster ($d_i = 1, \forall i$).

The second simulation runs the no-mean algorithm for variable selection on the same dataset.

5.3.2 Likelihood comparison

The results for the log-likelihood are presented in Table 5.7.

Table 5.7 Likelihood for variable selection.

	$\ell_1(\mathbf{d})$	$\ell_2(\mathbf{d})$	$\ell_3(\mathbf{d})$	$\ell_4(\mathbf{d})$
True **d**	**-81.53**	**-76.60**	-41.69	-41.72
Random **d**	-309.30	-275.93	-41.50	-41.53
$d_i = i$	-190.91	-170.85	-50.69	-50.72
$d_i = 1$	-318.80	-279.00	**-38.76**	**-38.80**

In our model of variable selection (see Section 4.2), a feature is important if the likelihood for the current clustering is higher than the likelihood without any clustering ($d_i = 1, \forall i$).

Our simulation confirms this result. For variables 1 and 2, the log-likelihood for all groupings is always greater than the log-likelihood without clustering. For variables 3 and 4, the likelihood without clustering is always greater than other groupings. These results confirm that variables 1 and 2 are useful for clustering and variables 3 and 4 are irrelevant to the clustering problem.

5.3.3 No-mean for variable selection

The no-mean algorithm for variable selection presented in Appendix B is run on the dataset in Figure 5.7. The algorithm converges to the true clustering and selects the first two variables, matching the expected result.

CHAPTER 6

CONCLUSION

As a rising field in machine learning and data analysis, data clustering has seen a lot of improvements in the last decades. New algorithms have brought shorter computation time or better performance but often at the cost of a trade-off between these two. We proposed the no-mean algorithm, a variant of the k-means algorithm, with better clustering behaviour while keeping the same time complexity.

6.1 Summary

The k-means is known for being one of the fastest clustering algorithm. The computational complexity of one of its iteration is linear with respect to the number of observations, clusters, and features.

The main drawback of the k-means is its convergence to non optimal solutions. Depending on the initial centers, the k-means may not converge to the best grouping, in terms of minimal within-cluster dissimilarity.

We showed that the k-means is a degenerate case of some probabilistic models. Mixture models and Bayesian clustering also aim to minimize the within-cluster dissimilarity and their respective solutions, the EM algorithm and the Gibbs sampler, behave similar to the k-means when the scale parameter of the mixing distribution tend to zero.

On this basis, we implemented our own version of the Bayesian clustering Gibbs sampler with simulated annealing, the no-mean algorithm. We assume a Gaussian prior distribution on the cluster centers and integrate the marginal likelihood over the centers distribution. Using the posterior, the Gibbs sampler is applied directly on the cluster labels without computing the mean at each iteration.

Gibbs sampling ensures convergence to the optimal clustering, for a logarithmic rate of simulated annealing. Therefore, from theoretical perspective, the no-mean algorithm is superior to the k-means. In practice, the logarithmic rate is too slow if one wants to converge in a number of steps comparable with the k-means. We observed that exponential rates provide satisfactory results both in simulations and in applications.

Our main innovation is the implementation of the Gibbs sampler with the same complexity as the k-means. Both algorithms computation times scale linearly with the parameters.

6.2 Limitations

For large or complex datasets, an exponential rate of simulated annealing greatly increases the chances of converging to a local optimum. We also saw that using such rate does not ensure reasonable convergence rates, see Table 5.6.

Even if we could afford a logarithmic rate of simulated annealing, the result would only be optimal with respect to the assumed statistical model. If the data do not fit this model, as seen with the intrusion dataset, there is no guaranty that the no-mean algorithm performs well. However, since our distributional assumption mimics the k-means, we may claim that the performance of the no-mean will still be an improvement compared with the k-means.

6.3 Future prospects

Our results show the potential of our algorithm for variable selection in clustering. Our model is still limited in the sense that our algorithm usually considers variables as irrelevant unless clusters are clearly separated. Applied to real datasets, the algorithm often selects no variable at all. Further studies are required to tackle this problem.

The other variation of the no-mean algorithm for the selection of the number of clusters is another interesting lead. The choice of an appropriate prior distribution over the potential groupings is a key issue, but not the only one. Sampling multiple data points into a new cluster at once or splitting/merging existing clusters could also be considered in our Bayesian framework to develop a new version of the no-k-no-mean algorithm.

BIBLIOGRAPHY

AHMED, M. N., YAMANY, S. M., MOHAMED, N., FARAG, A. A. and MORIARTY, T. (2002). A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data. *Medical Imaging, IEEE Transactions on*, 21, 193–199.

ALSABTI, K., RANKA, S. and SINGH, V. (1997). An efficient k-means clustering algorithm.

ARTHUR, D. and VASSILVITSKII, S. (2007). k-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.

BACHE, K. and LICHMAN, M. (2013). UCI machine learning repository.

BAHMANI, B., MOSELEY, B., VATTANI, A., KUMAR, R. and VASSILVITSKII, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, 5, 622–633.

BANFIELD, J. D. and RAFTERY, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 803–821.

BEHBOODIAN, J. (1970). On a mixture of normal distributions. *Biometrika*, 57, 215–217.

BELLMAN, R. (1956). Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42, 767.

BENTLEY, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 509–517.

BERGSTRA, J. and BENGIO, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13, 281–305.

BLEI, D. M., NG, A. Y. and JORDAN, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3, 993–1022.

CHEESEMAN, P., SELF, M., KELLY, J. and STUTZ, J. (1996). Bayesian classification.

CROWLEY, E. M. (1997). Product partition models for normal means. *Journal of the American Statistical Association*, 92, 192–198.

DALENIUS, T. (1950). The problem of optimum stratification. *Scandinavian Actuarial Journal*, 1950, 203–213.

DAY, W. H. and EDELSBRUNNER, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1, 7–24.

DEFAYS, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal*, 20, 364–366.

DEMPSTER, A. P., LAIRD, N. M. and RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 1–38.

DU, Q., FABER, V. and GUNZBURGER, M. (1999). Centroidal voronoi tessellations: applications and algorithms. *SIAM Review*, 41, 637–676.

DUNN, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3, 32–57.

EWENS, W. J. (1972). The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, 3, 87–112.

FARIVAR, R., REBOLLEDO, D., CHAN, E. and CAMPBELL, R. H. (2008). A parallel implementation of k-means clustering on gpus. *Proceedings of 2008 International Conference on Parallel and Distributed Processing Techniques and Applications*. 340–345.

FORGY, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21, 768–769.

FRIEDMAN, H. P. and RUBIN, J. (1967). On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62, 1159–1178.

GELFAND, A. E., SMITH, A. F. and LEE, T.-M. (1992). Bayesian analysis of constrained parameter and truncated data problems using gibbs sampling. *Journal of the American Statistical Association*, 87, 523–532.

GEMAN, S. and GEMAN, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 721–741.

HALL, L. O., "OZYURT, I. B. and BEZDEK, J. C. (1999). Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3, 103–112.

HANSON, R., STUTZ, J. and CHEESEMAN, P. (1991). *Bayesian classification theory*. NASA Ames Research Center, Artificial Intelligence Research Branch.

HARTIGAN, J. A. and WONG, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Applied Statistics*, 100–108.

HASSELBLAD, V. (1966). Estimation of parameters for a mixture of normal distributions. *Technometrics*, 8, 431–444.

HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2009). *The Elements of Statistical Learning*. No. 1. Springer, seconde édition.

HASTINGS, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109.

HEARD, N. A., HOLMES, C. C. and STEPHENS, D. A. (2006). A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: An application of Bayesian hierarchical clustering of curves. *Journal of the American Statistical Association*, 101, 18–29.

HOCHBAUM, D. S. and SHMOYS, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10, 180–184.

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

JAIN, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31, 651–666.

JOHNSON, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32, 241–254.

JUNG, Y., PARK, H., DU, D.-Z. and DRAKE, B. L. (2003). A decision criterion for the optimal number of clusters in hierarchical clustering. *Journal of Global Optimization*, 25, 91–111.

LANCE, G. N. and WILLIAMS, W. T. (1967). A general theory of classificatory sorting strategies ii. clustering systems. *The Computer Journal*, 10, 271–277.

LI, W. and MCCALLUM, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 577–584.

LI, Y., ZHAO, K., CHU, X. and LIU, J. (2010). Speeding up k-means algorithm by gpus. *2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*. IEEE, 115–122.

LLOYD, S. (1982). Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28, 129–137.

MACQUEEN, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. California, USA, vol. 1, 281–297.

MARIN, J.-M., MENGERSEN, K. and ROBERT, C. P. (2005). Bayesian modelling and inference on mixtures of distributions. *Handbook of Statistics*, 25, 459–507.

MATLAB (2010). *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

MCCULLAGH, P. and YANG, J. (2006). Stochastic classification models. *Proceedings of the International Congress of Mathematicians: Madrid, August 22-30, 2006: invited lectures*. 669–686.

MCCULLOCH, C. E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association*, 92, 162–170.

MENG, X.-L. and RUBIN, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80, 267–278.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H. and TELLER, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21, 1087–1092.

MILLIGAN, G. W. and COOPER, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50, 159–179.

MITRA, D., ROMEO, F. and SANGIOVANNI-VINCENTELLI, A. (1985). Convergence and finite-time behavior of simulated annealing. *Decision and Control, 1985 24th IEEE Conference on*. IEEE, vol. 24, 761–767.

MOJENA, R. (1977). Hierarchical grouping methods and stopping rules: an evaluation. *The Computer Journal*, 20, 359–363.

MORISSETTE, L. and CHARTIER, S. (2013). The k-means clustering technique: General considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology*, 9, 15–24.

MURTAGH, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26, 354–359.

NVIDIA CORPORATION (2007). *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. NVIDIA Corporation.

PELLEG, D. and MOORE, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 277–281.

PELLEG, D. and MOORE, A. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. *ICML*. 727–734.

PHANENDRA BABU, G. and NARASIMHA MURTY, M. (1993). A near-optimal initial seed value selection in k-means means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14, 763–769.

PITMAN, J. and YOR, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 855–900.

R DEVELOPMENT CORE TEAM (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

RAND, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66, 846–850.

RENNIE, B. and DOBSON, A. (1969). On stirling numbers of the second kind. *Journal of Combinatorial Theory*, 7, 116–121.

RISSANEN, J. (1978). Modeling by shortest data description. *Automatica*, 14, 465–471.

SAVARESI, S. M., BOLEY, D. L., BITTANTI, S. and GAZZANIGA, G. (2002). Cluster selection in divisive clustering algorithms. *Proceedings of SIAM International Conference on Data Mining*. SIAM, 299–314.

SCOTT, A. and SYMONS, M. J. (1971). Clustering methods based on likelihood ratio criteria. *Biometrics*, 387–397.

SIBSON, R. (1973). Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16, 30–34.

SLONIM, N., AHARONI, E. and CRAMMER, K. (2013). Hartigan's k-means versus lloyd's k-means: is it time for a change? *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, 1677–1684.

STEINBACH, M., KARYPIS, G., KUMAR, V. *ET AL.* (2000). A comparison of document clustering techniques. *Knowledge Discovery and Data Mining workshop on text mining*. Boston, vol. 400, 525–526.

STEINHAUS, H. (1956). Sur la division des corps materiels en parties. *Bulletin de l'Academie Polonaise des Sciences Cl. III. 4*, 801–804.

SUNDBERG, R. (1974). Maximum likelihood theory for incomplete data from an exponential family. *Scandinavian Journal of Statistics*, 49–58.

TELGARSKY, M. and VATTANI, A. (2010). Hartigan's method: k-means clustering without voronoi. *International Conference on Artificial Intelligence and Statistics*. 820–827.

TIBSHIRANI, R., WALTHER, G. and HASTIE, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B*, 63, 411–423.

WARD, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58, 236–244.

WU, C. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 95–103.

APPENDIX A

R implementation of the no-mean algorithm

```
# i in 1:n the number of samples
# j in 1:p the number of variables
# c in 1:k the number of clusters
# Y[i,j]: the data
# B[c,j]: the sum of data in cluster c for variable j
# B[c,]: the vectors of sums of data in cluster c for all variables
# d[i]: assigned cluster of sample i
# nC[c]: the number of samples in cluster c
# sigmaSquare: within cluster dispersion
# rho = tauSquare / (tauSquare + sigmaSquare): percentage of total
## dispersion explained by cluster dispersion

noMeans <- function(n, p, k, Y, sigmaSquare, rho, numberOfSteps,
      annealingRate, setInitial = FALSE, initialD = c()){
  Y = as.matrix(Y,ncol=p);

  if(setInitial){
    d = initialD;
    nC = vector(mode="double", length=k);
    for(c in 1:k)
      nC[c] = length(which(d==c));
  } else {
    d = sample(rep(1:k,length.out=n));

    nC = rep(n%/%k,k);

    if(n%%k>0)
      nC[1:(n%%k)] = n%/%k+1;
  }

  B = matrix(nrow=k,ncol=p);
```

```
for(c in 1:k)
  B[c,] = apply(as.matrix(Y[which(d==c),]),2,sum);


for(steps in 1:numberOfSteps){
  for(i in 1:n){

    c0 = d[i];
    l = vector(mode="double", length=k);
    myProb = vector(mode="double", length=k);

    # This compute the difference between the initial log-likelihood L[c0]
    # and the log-likelihood when the sample Y[i,] is moved from cluster c0
    # to the cluster c


    if(nC[c0] > 1){
      for(c in 1:k){
        if(c!=c0){
          l[c] = p/2*log((1+(nC[c]-1)*rho)/(1+nC[c]*rho)) +
            rho/(2*sigmaSquare)*(sum((B[c,]+Y[i,])^2)/(1+nC[c]*rho) -
            sum(B[c,]^2)/(1+(nC[c]-1)*rho));
        }

        else{
          l[c] = -p/2*log((1+(nC[c]-1)*rho)/(1+(nC[c]-2)*rho)) -
            rho/(2*sigmaSquare)*(sum((B[c,]-Y[i,])^2)/(1+(nC[c]-2)*rho) -
            sum(B[c,]^2)/(1+(nC[c]-1)*rho));
        }
      }

      # Avoid null values of the normalization parameter
      # by multiplying by exp(-max(L))
      myMax = max(l);
      myNorm = sum(exp(l - myMax));
      myProb = exp(l - myMax)/myNorm;
```

```
        # Sample the new cluster for this sample
        sampleD = sample(x = 1:k, size = 1, prob = myProb);
        # Update the information
        d[i] = sampleD;
        B[c0,] = B[c0,] - Y[i,];
        B[sampleD,] = B[sampleD,] + Y[i,];
        nC[c0] = nC[c0] - 1;
        nC[sampleD] = nC[sampleD] + 1;
      }

    }
    sigmaSquare = sigmaSquare / annealingRate;
    alpha = (1/annealingRate)^2;
    # Update rho considering tauSquare = annealingRate*tauSquare
    rho = rho / (alpha + (1-alpha)*rho);
  }

  return(d);
}
```

APPENDIX B

Pseudo-code of the no-mean algorithm for variable selection

Initialize clusters;
`selectedVariables` $= 1 : p$;
for j in $1 : p$ **do**
    Compute $\texttt{l}_0[j]$;
    Compute $\texttt{l}[j]$ with the initial clusters;
end for
for `iter` in $1 :$ `numberOfIterations` **do**
    for i in `permutation`(N) **do**
        for $c = 1$ to K **do**
            for j in $1 : p$ **do**
                Compute $\Delta\texttt{l}[c, j]$;
            end for
            $\Delta\texttt{l}[c] = 0$
            for j in `selectedVariables` **do**
                $\Delta\texttt{l}[c] = \Delta\texttt{l}[c] + \Delta\texttt{l}[c, j]$;
            end for
            `prob`$[c] = P(d[i] = c) \propto \Delta\texttt{l}[c]$;
        end for
        `cluster`$[i] = c$ with probability `prob`$[c]$;
        for j in $1 : p$ **do**
            $\texttt{l}[j] = \texttt{l}[j] + \Delta\texttt{l}[\texttt{cluster}[i], j]$;
        end for
    end for
    for j in $1 : p$ **do**
        if $\texttt{l}[j] > \texttt{l}_0[j]$ **then**
            Add j to `selectedVariables`;
        else
            Remove j from `selectedVariables`;
        end if
    end for
    `sigmaSquare` $=$ `sigmaSquare`/`alpha`;
    `tauSquare` $=$ `alpha` $*$ `tauSquare`;
end for

Algorithm 3 No-mean for variable selection