MMD and Ward criterion in a RKHS : application to Kernel based hierarchical agglomerative clustering

Na Li

Thèse
de doctorat
de l'UTT

Na LI

MMD and Ward Criterion in a RKHS. Application to Kernel based Hierarchical Agglomerative Clustering

Spécialité :
Optimisation et Sûreté des Systèmes

2015TROY0033

Année 2015

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Na LI

Le 1er décembre 2015

MMD and Ward Criterion in a RKHS. Application to Kernel based Hierarchical Agglomerative Clustering

JURY

M. D. BRIE	PROFESSEUR DES UNIVERSITES	Président
M. S. CANU	PROFESSEUR DES UNIVERSITES	Rapporteur
M. T. DENOEUX	PROFESSEUR DES UNIVERSITES	Rapporteur
M. N. LEFEBVRE	RESEARCHER	Directeur de thèse
M. R. LENGELLÉ	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. Y. LU	ASSOCIATE PROFESSOR	Examinateur

Acknowledgements

My deepest gratitude goes first and foremost to Professor Regis LENGELLE, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of my PHD study. Without his consistent and illuminating instruction, this thesis could not have reached its present form. He is more than a supervisor, also a friend, a family, one of the important peoples during my five years study in France. It is the destiny to work with him, I am so happy to work with him.

Second, I would like to express my heartfelt gratitude to Nicolas LEFEVBVRE, for his instructive advice, useful suggestions and selfless contributions on my thesis, especially for programming. He is always so nice with me. I appreciate a lot the time we worked together.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends who gave me their help and time in helping me work out my problems during the difficult course of the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last years, kernel methods have owned great success in supervised machine learning, whose best known elements are Support Vector Machines, showing their ability of generating more generalized solutions and its capacity of analyzing large data sets. However in real world applications, for example analysis of pharmacological data, it is useful to detect the pattern structure of the data without knowing the label information. This makes the problem to be unsupervised. Obviously, extending kernel methods to the unsupervised case would be a good idea to solve the problem. In unsupervised problems, the focus task is to find out the optimal label combination so that data with the same label are more similar to each other than to those with different labels. Trying all possible label combinations is a time consuming task (a NP-hard problem), so this makes it necessary to do a heuristic research to avoid the previous issue.

Clustering, as a useful tool for unsupervised classification, is the task of grouping objects according to some measured or perceived characteristics of them and it has owned great success in exploring the hidden structure of unlabeled data sets. It is well worth extending kernel methods into clustering both for binary and multiple clusters, proposing heuristics to be time saving, seeking new formulation of the optimization problem and considering associated constraints.

1.1 Clustering

Clustering has been early applied in many disciplines including psychology [34], biology [75] and computer security [6] and then it owed increasing attention in patter recognition, image processing [41] and information retrieval [68] etc. Clustering algorithms have a long history. They originated in anthropology by Driver and Kroeber [17]. Survey papers in this field exist and they reviewed clustering problem from different points of view including clustering algorithms [42], clustering problem description under a mathematical scheme [33] and applications of clustering algorithms [2] etc. More influential survey papers can be found in [5], [10], [21] and books on clustering also exist [39], [4] and [76] etc.

1.1.1 Structure of clustering task

A typical clustering procedure consists of four important steps which are shown in Fig. 1.1.

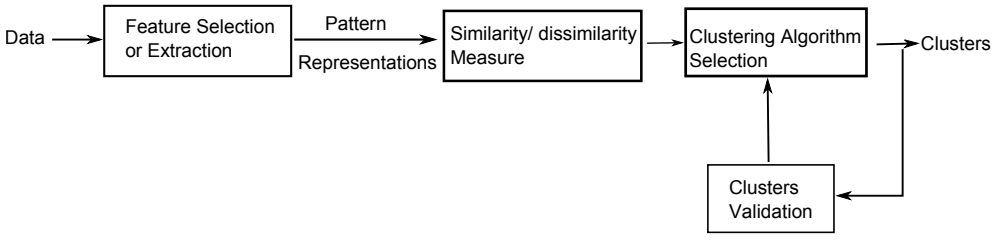


FIGURE 1.1: Clustering procedure consisting of four steps with a feedback pathway

1. **Feature Selection or extraction**. It is a task of identifying most effective features and generating novel features by using transformations. It's a crucial step as good feature selection and extraction will greatly decrease the calculation complexity and improve the effectiveness of clustering algorithm. More work has been done in [7], [40] and [10].

2. **Similarity or dissimilarity measure.** Before discovering clusters where data are more close to each other within the cluster, it is important to define the closeness. As we may have different feature styles with different scales, the choice of distance measure should be well adapted to the feature. The

most famous metric is the Euclidean distance which is for continuous features [42]

$$d_2(x_i, x_j) = (\sum_{k=1}^{d}(x_{ik} - x_{jk})^2)^{1/2}$$
$$= \| x_i - x_j \|_2$$

It is a special case of the Minkowski metric with $p = 2$

$$d_2(x_i, x_j) = (\sum_{k=1}^{d}(x_{ik} - x_{jk})^p)^{1/p}$$
$$= \| x_i - x_j \|_p$$

More distance metric forms and their applications are given in [85] and techniques for metric selection could be found in [42].

3. **Clustering algorithm selection.** This step is in fact a combination of similarity measure and a proper criterion function selection. Of course it is the most important step and more details about clustering algorithms will be given later.

4. **Cluster validation.** This step concerns to solve questions of how many clusters are hidden behind de the patterns and whether the clustering results obtained are meaningful. Criteria have been proposed to evaluate the algorithms used and generally there are internal indices, external indices and relative indices [39]. A well known research has been done by Milligan and Cooper [60]. where they compared 30 indices.

1.1.2 User's dilemma

A large quantity of existing clustering algorithms makes it difficult to select an appropriate one to better solve the task. In spite of several methods having been proposed to try to compare different algorithms [18], until now no effective method to help us analysis the following difficulties and they are still great challenges for practitioners:

- What is a cluster?

- How to select feature and which method should be used to extract features?
- what is the right metric for a given data set?
- which similarity measure should be used?
- which algorithm should be considered?
- How many clusters do we have?
- How to validate a cluster?
- How to deal with large scale data set?

There is no universal clustering algorithm that could solve all the problems. Most of algorithms are very problem dependent. Thus trying to understand difference between clustering techniques and finding a better way to better adapt to the clustering task is the purpose of investigators.

1.1.3 Clustering algorithms

Fig. A.1 illustrates a taxonomy of clustering algorithms. Of course this is only one possible of taxonomy among others. In general, clustering algorithms could be divided into two classes: hierarchical clustering algorithms and partitioning clustering algorithms.

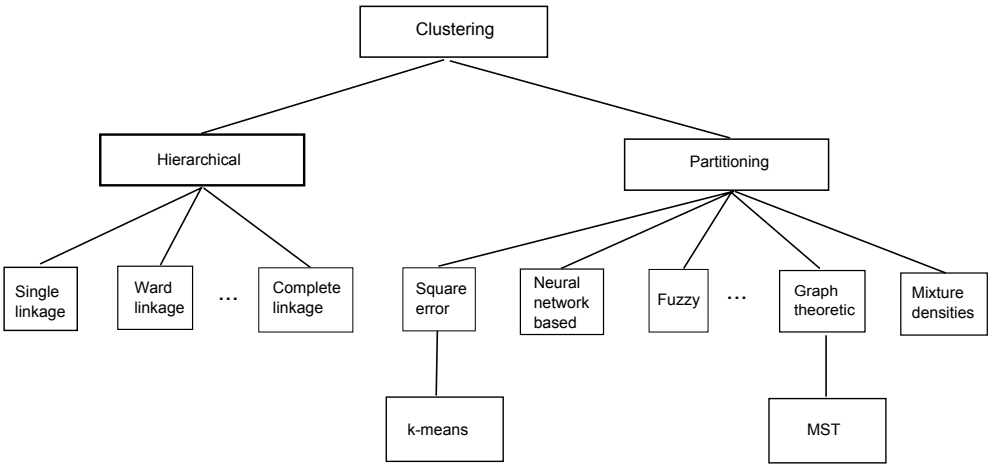
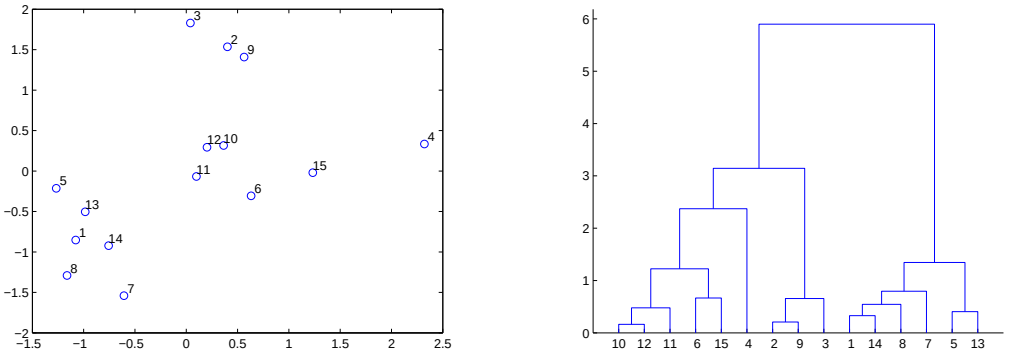


FIGURE 1.2: A taxonomy of clustering.

Hierarchical algorithms provide a view of data structure by a dendrogram which represents nested patterns and similarity levels. By breaking the dendrogram at different levels, different clusterings could be obtained with different numbers of clusters. Fig. 1.3 illustrates a data set consisting 15 points and the dendrogram represent similarities between points/sub groups.



(A) Data set (B) Corresponding dendrogram

FIGURE 1.3: A dendrogram obtained using Ward linkage.

A key technique in Hierarchical algorithms is the definitions of distance between clusters. A famous formula has been proposed by Lance and Williams [51]. Different distance measures could be generated by using this formula including single linkage, complete linkage, average linkage and ward linkage etc [39], [64]. More details about this formula and distance measures will be given later in this document.

Partitioning algorithms In partitioning algorithms, there is no nested cluster structures like in hierarchical algorithms. Usually data sets are partitioned by optimizing an objective function. The most well-known partitioning algorithm is K-means, which was first proposed in 1957. After that a lot of clustering algorithms have been proposed. Besides those listed in Fig. A.1, other clustering algorithms emerged in literatures include SOM, fuzzy c-means and CLIQUE etc.

As the most famous partitioning algorithm, K-means is simple and is easily implemented in solving real world problems. It is efficient in time consumption and provides competitive clustering results in many problems [42]. Disadvantages of K-means have been well studied. It is important to identify a initial partitions but no effective method is proposed to solve this problem. Besides it can not be

guaranteed that K-means could converge to a global optimum [85]. Another disadvantage which may restrict K-means to be considered in real clustering problem is that the number of clusters is unknown while to implement K-means, firstly we should provide the number of clusters.

Compared to K-means, hierarchical clustering algorithms are proved to be more versatile [42]. Even though hierarchical methods are more time consuming, new hierarchical clustering have been proposed to help handling large-scale data sets [31], [32], [87].

1.2 Kernel based clustering

Kernel-based clustering, has owned great success because of its ability to perform linear tasks in some non linearly transformed spaces by using a transform function $\phi(.)$. In machine learning, the kernel trick has been firstly introduced by Aizerman [1]. It became famous in Support Vector Machines (SVM) initially proposed by Cortes and Vapnik [14]. SVM has shown better performances in many problems and this success has brought an extensive use of the kernel trick into other algorithms like kernel PCA [69], non linear (adaptive) filtering [65] etc. Kernel methods have been widely used in supervised classification tasks like SVM and then they were extended to unsupervised classification.

A lot of kernel-induced clustering algorithms have emerged due to the extensive use of inner products in linear methods. Most of these algorithms are kernelized versions of the corresponding conventional algorithms. Surveys of kernel-induced methods for clustering have been done in [22, 48, 63]. The first proposed and the most well-known kernel-induced algorithm is kernel K-means by Scholkopf [69]. A further version has been proposed by Girolami [26]. After that, several kernel-induced algorithms have emerged such as kernel fuzzy c-means and kernel Self Organizing Maps etc. Compared with the corresponding conventional algorithms, kernelized criteria have shown better performance especially for non-linearly separable data sets.

Besides of the dilemmas existing in clustering, we have more challenges in kernel-based clustering:

- How to select a good kernel function for a given learning task?
- How to deal with large-scale data sets as the introduction of the kernel method might increase the calculation complexity?
- Which similarity measure should be considered in the kernel induced feature space?

We try to do research on some of these problems but not all of them will be studied. A recently proposed criterion attracts our attention which has been proposed to measure distance between two clusters and is easily to be calculated in the context of kernel method.

1.3 Research motivation

Maximum Mean Discrepancy (MMD) is a *distance* measure between two probability density functions. It has been used to compare distributions. Maximum Mean Discrepancy (MMD) is straightforward to estimate in a Reproducing Kernel Hilbert Space (RKHS) as density functions of distributions can be embedded in a RKHS. Study of probabilistic distance measure in RHKS could be found in [88].

We prove that the square of Maximum Mean Discrepancy is strongly connected to the Ward criterion, which is widely used in Hierarchical Agglomerative Clustering. Hierarchical clustering depends on distance calculations which are based on inner products. We can perform HAC after mapping the input data onto a higher dimension space using a nonlinear transform which induces wider classes of classifiers. This is one of the main ideas of kernel methods, where the transformed space is selected as a RKHS in which distance calculations can be easily evaluated with the help of the kernel trick [1].

We propose to consider Kernel-based Hierarchical Agglomerative algorithms using Ward linkage as a tool to perform clustering, considering dilemmas induced by kernel-based clustering and trying to propose better criteria to provide satisfactory results.

1.4 Outline

This document is organized as follows. Chapiter 2 introduces fundamental elements of kernel theory and basic kernel functions which we will consider in our research. In chapter 3, MMD theory and a detailed description of Hierarchical Agglomerative Clustering (HAC), especially distance measure, will be given. In chapter 4, we discuss a kernel parameter selection method and propose better criteria for kernel parameter selection. Chapter 5 provides methods for determination of number of clusters. In chapter 6, we propose an iterative kernel based HAC which enables to partition the data set and, at the same time, select the kernel parameter and determine the number of clusters. Methods will be proposed to evaluate the effectiveness of the considered criteria. Finally we conclude our work and give some perspectives.

Chapter 2

Kernel theory

Contents

The purpose of this chapter is to introduce a general framework of kernel theory. We firstly give a brief introduction of kernel method and highlight its advantages and key techniques. Then we review fundamentals notions which are necessary to kernel theory. Definitions of kernel and kernel matrix will be given and also methods to validate a kernel. The ways to characterize kernels provide tools to construct kernels by changing kernels or combining them to obtain a new one.

Examples of the corresponding embedding effect in feature space of these kernel transforms will be given as well. Finally we introduce some basic kernels which are popular in real applications.

2.1 Introduction

Kernel methods aim at embedding non-linearly separable data in a high dimensional feature space where algorithms based on linear algebra and statistics could be used. The key idea of kernel methods is that inner product of two elements in the feature space can be computed by the kernel function without explicitly figure out the coordinates in the feature space.

Kernel is a function k that for all $x, y \in X$ satisfies

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \tag{2.1}$$

where ϕ is a mapping from X to a feature space $\mathcal{F}$

$$\phi : x \longmapsto \phi(x) \in \mathcal{F} \tag{2.2}$$

Kernel methods have owned great success and the following aspects should be highlighted:

- Data points are embedded from a input space to a vector space (feature space)
- Coordinates in the feature space are not necessary, similarity information could be obtained by inner products
- Pairwise inner products can be calculated by the kernel function.
- It is expected that the initial problem is transformed to be a linear problem in the feature space even though it is non-linear in the input space

These could be illustrated by Fig. A.2.

Example. Here we give an example of a kernel function and the corresponding embedding function.

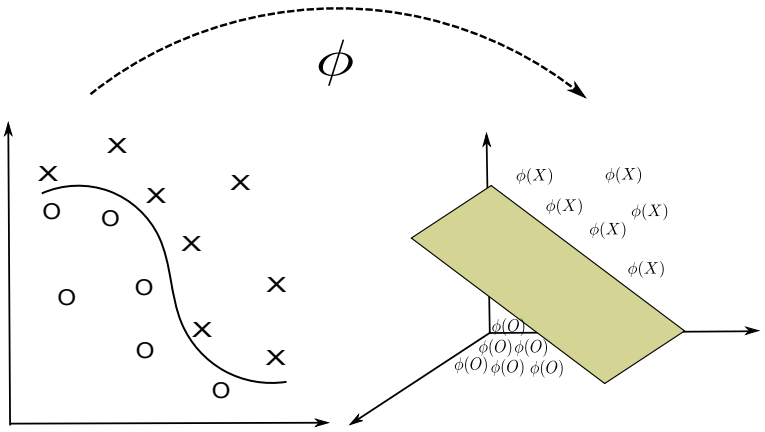


FIGURE 2.1: Data points are embedded from input space to a high dimensional space where the non-linear problem becomes linear, using transformation ϕ.

We consider a two-dimension input space $X \subseteq \mathbb{R}^2$ and its corresponding feature map

$$\phi : x = (x_1, x_2) \longmapsto \phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in \mathcal{F} \subseteq \mathbb{R}^3$$

Hence we have

$$\begin{aligned}\langle \phi(x), \phi(y) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (y_1^2, y_2^2, \sqrt{2}y_1y_2) \rangle \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= \langle x, y \rangle^2\end{aligned}$$

So the kernel function is

$$k(x, y) = \langle \phi(x), \phi(y) \rangle = \langle x, y \rangle^2$$

Consider another embedding

$$\phi : x = (x_1, x_2) \longmapsto \phi(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in \mathcal{F} \subseteq \mathbb{R}^4$$

It can be easily demonstrated that this embedding could be realized with the same kernel function

$$k(x, y) = \langle x, y \rangle^2$$

Remark. Kernel function is not the only element which determinates the feature space. With the same kernel function, different feature spaces could be created.

Consider an n-dimensional input space $X \subseteq \mathbb{R}^n$, with the same kernel function

$$k(x, y) = \langle x, y \rangle^2$$

the following embedding functions could be obtained

$$\phi : x \longmapsto \phi(x) = (x_i x_j)_{i,j=1}^{n} \in \mathcal{F} \subseteq \mathbb{R}^{n^2}$$

It illustrates that keeping the same kernel function, one could improve the complexity of the feature space, thus be able to gain algorithms' efficiency in a high-dimensional feature space, which is an important aspect that should be considered in real applications.

2.2 Fundamental elements for kernel theory

To well understand the kernel theory, the following fundamental elements need to be introduced. This section reviews definitions, properties and the corresponding proofs which are necessary for kernel theory. These notions play a key role in kernel validation and kernel characterization.

2.2.1 Hilbert space

Definition 1. The scalar field is defined as F. A ***inner product space***(pre-Hilbert space) [36] is a vector space $\mathcal{X}$ over F

$$\langle x, u \rangle$$

which satisfies

1. Symmetry

$$\langle x, u \rangle = \langle u, x \rangle \forall x, u \in \mathcal{X}$$

2. Bilinearity

$$\langle \alpha x + \beta u, w \rangle = \alpha \langle x, w \rangle + \beta \langle u, w \rangle \forall x, u, w \in \mathcal{X}, \forall \alpha, \beta \in \mathbb{R}$$

3.Strict Positive Definiteness

$$\langle x, x\rangle \geq 0 \forall x \in \mathcal{X}$$

$$\langle x, x\rangle = 0 \Longleftrightarrow x = 0.$$

Inner product space is sometimes mentioned as Hilbert space, however most of researchers require additional conditions as completeness and separability. We now give a formal Hilbert space definition.

Definition 2. A ***Hilbert space*** $\mathcal{H}$ is an inner product space which also should be separable and complete.

Completeness means that every Cauchy sequence $\{h\}_{n\geq 1}$ which satisfies

$$\sup_{m>n} \parallel h_n - h_m \parallel \longrightarrow 0, \quad \text{when} n \longrightarrow \infty$$

over $\mathcal{H}$ converges.

Separability refers to the property that there is a finite set of elements $h_1, ...h_N$ of $\mathcal{H}$ that satisfies

$$\min_i \parallel h_i - h \parallel < \epsilon, \forall \epsilon > 0, \forall h \in \mathcal{H}$$

Example. Here we give several Hilbert space examples

- Space with $\langle x, y\rangle = x^T y$
- Space with $\langle x, y\rangle = \sum_{i=1}^{\infty} x_i y_i$
- Space with inner product $\langle f, g\rangle = \int f(x)g(x)d(x)$ where functions f satisfy $\int f(x)^2 d(x) < \infty$

2.2.2 Reproducing Kernel Hilbert Space

Before introducing the definition of a Reproducing Kernel Hilbert Space (RKHS), we review the following notions which are essential in kernel definition and kernel validation.

Definition 3. A matrix M is **positive semi-definite matrix** [56] if

$$a'Ma \geq 0, \forall a \in \mathbb{R}^n$$

Definition 4. A symmetric function $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semi-definite if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j f(x_i, x_j) = \alpha' F \alpha \geq 0, \forall (\alpha_1, ...\alpha_n) \in \mathbb{R}^n, \forall (x_1, ...x_n) \in \mathcal{X}^n$$

Compared with a Hilbert space, the RKHS introduces the reproducing property.

Definition 5. $k(.,.)$ is a ***reproducing kernel*** [27] of a Hilbert space $\mathcal{H}$ if

$$f(x) = \langle k(x,.), f(.) \rangle_{\mathcal{H}}, \forall f \in \mathcal{H}$$

Remark. If a kernel k is a reproducing kernel, then k satisfies the finitely positive semi-definite property

Proof.

$$\langle k(x_i,.), k(y_j.) \rangle_{\mathcal{H}} = k(x_i, y_j), f \in \mathcal{H}$$

$$\begin{aligned}
\sum_{i,j=1}^{l} \alpha_i \alpha_j k(x_i, x_j) &= \sum_{i,j=1}^{l} \alpha_i \alpha_j \langle k(x_i,.), k(x_j.) \rangle_{\mathcal{H}} \\
&= \langle \sum_{i}^{l} \alpha_i k(x_i,.), \sum_{j}^{l} \alpha_j k(x_j,.) \rangle_{\mathcal{H}} \\
&= \| \sum_{i}^{l} \alpha_i k(x_i,.) \|_{\mathcal{H}}^2 \geq 0
\end{aligned}$$

□

Definition 6. An ***evaluation function*** is a function in space $\mathcal{H}$ which maps the function f to a value of f at point x ($\mathcal{H} \to \mathbb{R}$)

$$\delta_x(f) = f(x), \forall f \in \mathcal{H}$$

Definition 7. A ***Reproducing Kernel Hilbert Space*** [36] (RKHS) is a Hilbert space with a reproducing kernel, where the evaluation functions $f(.)$ are bounded,

i.e

$$\exists M > 0 \quad : |\delta_x(f)| = |f(x)| \leq M\|f\|_{\mathcal{H}}$$

2.2.3 Characterization of kernels

Given the above notions on kernel theory, so far, the way to verify that a function $k(x, y)$ is a kernel is that it maps data from one space to another higher dimensional space with an embedding map ϕ and inner product in this higher dimensional space which can be calculated by this function. In this section, we characterize the kernel function using an alternative way, which also enables us to construct new kernels.

Characterization of kernels [67] A function $k : X \times X \to \mathbb{R}$ can be written as

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$

where x, y are mapped into a Hilbert space by an embedding function ϕ, if and only if it satisfies the finitely positive semi-definite property.

Moore-Aronszajn Theorem [3] states that for every positive definite function $k(.,.)$, there exists a unique RKHS.

Given a kernel k, we define a reproducing kernel feature map

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$$

$$\phi(x) = k(., x)$$

We consider the space

$$\mathcal{F} = \{\sum_{i=1}^{n} \alpha_i k(x_i, .) : n \in \mathbb{N}, x_i \in \mathcal{X}, \alpha_i \in \mathbb{R}, i = 1, ..., n\}$$

which is a set of points that are functions. Note that this space is also a vector space as it is closed under multiplication by a scalar and addition of functions. Let $f, g \in \mathcal{F}$ given by

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

$$g(x) = \sum_{j=1}^{m} \beta_j k(x_j, x)$$

then we have

$$\langle f, g\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_i\beta_j k(x_i, x_j) = \sum_{i=1}^{n}\alpha_i g(x_i)\sum_{j=1}^{m}\beta_j f(x_j)$$

It is clear that $\langle f, g\rangle$ is real-valued, symmetric and bilinear and so it satisfies the inner product properties, provided

$$\langle ., .\rangle \geq 0 \quad \forall f, g \in \mathcal{F}.$$

This could also be demonstrated by the assumption that all kernel matrices are positive semi-definite:

$$\langle f, f\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_i\alpha_j k(x_i, x_j) = \alpha' K\alpha \geq 0, \tag{2.3}$$

Where $\boldsymbol{\alpha}$ is the vector with elements $\alpha_i, i = 1, ...n$ and K is the kernel matrix.

Now we check the reproducing property. According to Eq. 2.2.3 when $g = k(x, .)$

$$\langle f, k(x, .)\rangle = \sum_{i=1}^{n}\alpha_i k(x_i, x) = f(x)$$

To validate that this space is a RKHS, we still need to verify the completeness and separability properties.

A sketch demonstration of separability property is that the kernel is continuous. For completeness, consider a Cauchy sequence $(f_n)_{n=1}^{\infty}$,

$$(f_n(x) - f_m(x))^2 = \langle f_n - f_m, k(x, .)\rangle^2 \leq \| f_n - f_m \|^2 k(x, x)$$

which shows that $f_n(x)$ is a bounded Cauchy sequence and has a limit.

Until now we have demonstrated that the function k which satisfies positive semi-definite property generate a unique RKHS.

Mercer Theorem The **Mercer-Hilbert-Schmit theorem** [8] states that if we have a kernel k that is positive (defined somehow), we can expand k in terms of eigenfunctions and eigenvalues of a positive operator. The Mercer theorem is in

fact equivalent to the finitely positive semi-definite property.

$$k(x,y) = \sum_{i=0}^{\infty} \lambda_i \phi_i(x)\phi_i(y)$$

where $\langle \phi_i \rangle_{i=0}^{\infty}$ is an infinite sequence of eigenfunctions and λ_i are eigenvalues with $\lambda_1 \geq \lambda_2 \geq ....$

2.2.4 Kernel matrix

Given the above basic notions which are necessary to construct the kernel theory, now we introduce a formal definition of a kernel:

Definition 8. $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if it satisfies the following conditions

1. k is symmetric: $k(x,y) = k(y,x)$
2. k is positive semi-definite.

Definition 9. Given a kernel k and a data set $X = \{x_1, ...x_n\}$, a matrix which contains evaluation information of pairs of data points can be obtained directly, known as kernel matrix or Gram matrix whose entries are K_{ij}.

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$$

The Gram matrix is symmetric since $K_{ij} = K_{ji}$ and it is positive semi-definite.

Proof. For any vector α we have

$$\begin{aligned}
\alpha' K \alpha &= \sum_{i,j=1}^{n} \alpha_i \alpha_j K_{ij} \\
&= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle \\
&= \langle \sum_{i=1}^{n} \alpha_i \phi(x_i), \sum_{j=1}^{n} \alpha_j \phi(x_j) \rangle \\
&= \| \sum_{i=1}^{n} \alpha_i \phi(x_i) \|^2 \geq 0
\end{aligned}$$

□

This property guarantees that we have a valid kernel (positive semi-definite property) and this enables us to manipulate kernels without considering the feature space.

Here again we recall the trick of kernel methods: Similarity measures of points in the feature space could be calculated by using the kernel function in the original space, without giving an explicit mapping function ϕ. Hence the Gram matrix could be considered as an interface between the input data and the representation in the feature space.

The positive semi-definite property can also be used to validate the intermediate methods designed to improve the efficiency of algorithms in the feature space. Data representation in the feature space is implicit and the only way is to manipulate the Gram matrix where the positive semi-definite property should be guaranteed. By example, adding a constant to the diagonal of Gram matrix introduces a soft margin in classification. Obviously the properties of Gram matrix are important in the algorithm execution. This matrix plays central roles in kernel methods because all the information about the data set should be extracted from this matrix.

2.3 Kernel constructions

In the previous section, we introduced a method to demonstrate that a function is a valid kernel function when it satisfies the positive semi-definitive property. This provides one way to create new kernels. A series of rules could be justified by this property to transform a kernel or combine kernels to obtain a new one. After operations on kernel matrices, if this property is still preserved, we could still embed the data in a feature space.

Two ways are available to implement kernel constructions:

- Operations on kernel functions
- Operations on kernel matrices

The origin of these two methods is consistent: The Gram matrix keeps to be positive semi-definite.

2.3.1 Operation on kernel function

In this part we list some properties also named as closure properties which enable us to manipulate kernel functions to create more complex kernels.

Proposition 1. (Closure Properties) [37], [57] Let $\phi : X \to \mathbb{R}^N$, k_1 and k_2 be kernels over $X \times X$ where $X \in \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(.)$ a real-valued function on X, $M(n \times n)$ a symmetric positive semi-definite matrix and k_3 a kernel over $\mathbb{R}^N \times \mathbb{R}^N$. Then we have the following functions k being kernels.

$1.k(x, y) = k_1(x, y) + k_2(x, y)$
$2.k(x, y) = \alpha k_1(x, y)$
$3.k(x, y) = k_1(x, y)k_2(x, y)$
$4.k(x, y) = f(x)f(y)$
$5.k(x, y) = k_3(\phi(x), \phi(y))$
$6.k(x, y) = x'My$

proof Let K_1 and K_2 be the $(n \times n)$ corresponding kernel matrices of kernels k_1 and k_2 and $\alpha \in \mathbb{R}^n$ be any vector. Recall that K is a positive semi-definite matrix if and only if $\alpha' M\alpha \geq 0, \forall\alpha$.

1. $K_1 + K_2$ is kernel matrix corresponding to $k_1 + k_2$. we have

$$\alpha'(K_1 + K_2)\alpha = \alpha' K_1\alpha + \alpha' K_2\alpha \geq 0$$

So $K_1 + K_2$ is the positive semi-definite and k is a kernel.

2. $\alpha' aK_1\alpha = a\alpha' K_1\alpha \geq 0$, so ak_1 is a kernel.

3. M is called a *Schur product* of matrix K_1 and K_2 where $M_{ij} = K_{1ij}K_{2ij}$. A *Schur product* of two positive semi-definite matrix is still positive seme-definite since the eigenvalues of the product are product of corresponding eigenvalues of the two matrices which are positive. So k is a kernel.

4. Let $\phi : x \mapsto f(x)$. $k(x, y) = f(x)f(y)$ is the corresponding kernel.

5. Since k_3 is a kernel, the corresponding kernel matrix is positive semi-definite, which means that k is a kernel.

6. As M being a symmetric positive semi-definite matrix, it can be written in terms of $V'\Lambda V$, where Λ is a diagonal matrix containing all the non-negative eigenvalues of M. Therefore we have

$$k(x,y) = x'My = x'V'\Lambda Vy = x'A'Ay = \langle Ax, Ay\rangle$$

where $A = \sqrt{\Lambda}V$. Obviously k is a kernel function with $\phi : x \mapsto Ax$.

Proposition 2. Let k_1 be a kernel and $p(x)$ a polynomial with positive coefficients. The following functions
1. $k(x,y) = p(k(x,y))$
2. $k(x,y) = exp(k(x,y))$
3. $k(x,y) = exp(-\frac{\|x-y\|^2}{2\sigma^2})$
are also kernels.

Proposition 1 provides several simple rules to create new kernels functions with simple basic kernel functions while proposition 2 gives more complicated kernel construction methods. A sketch of the proof of proposition 2 could be found in [71]. However both propositions follow the same rule: the corresponding kernel matrix after kernel operations should be always positive semi-definite.

2.3.1.1 Embedding effects corresponding to kernel constructions

The last subsection shows that new kernels can be obtained from existing kernels using a series of operations. New functions are verified to be valid kernels if we can demonstrate that the corresponding matrix is positive semi-definite, which means that data can be embedded in a feature space where inner product can be implemented by the new kernel function in the original space. Normally it is not necessary to figure out the explicit embedding function and in most of cases, it is impossible to represent the feature space. However, sometimes it is helpful to know how data are distributed in the feature space if we can understand the effect of kernel constructions on the feature space.

In this part, we give several example analyses of the embedding effect on kernel constructions that we have introduced in the last section.

1. $k(x, y) = k_1(x, y) + k_2(x, y)$.
   The feature vector corresponding to the addition of two kernels is given by

$$\phi(x) = [\phi_1(x), \phi_2(x)]$$

   The proof is given as:

$$\begin{aligned} k(x, y) &= \langle \phi(x), \phi(y) \rangle \\ &= \langle [\phi_1(x), \phi_2(x)], [\phi_1(y), \phi_2(y)] \rangle \\ &= \langle \phi_1(x), \phi_1(y) \rangle + \langle \phi_2(x), \phi_2(y) \rangle \end{aligned}$$

2. $k(x, y) = ak_1(x, y), a > 0$
   The embedding effect of this construction is very simple which is only a re-scaling of the feature space

$$\phi(x) = \sqrt{a}\phi_1(x)$$

3. $k(x, y) = f(x)f(y)$
   This kernel construction itself is also a feature construction with the embedding function

$$\phi(x) = f(x)$$

2.3.2 Operation on kernel matrix

The feature space can also be transformed by changing the kernel matrix, as mentioned before, which should always be positive semi-definite. Questions arise like how to calculate the kernel matrix when new data points arrive. Another problem that should be considered is that sometimes, if we focus on adapting kernels on a particular kernel matrix, we may be able to adjust kernels well on training data while performances on new data are not satisfactory.

Taloy et al. [71] give detailed presentations of operations on kernel matrices and their corresponding effects in the feature space. In general, the following operations could be implemented on the kernel matrices so as to realize feature space transformations.

- Simple transformation
  Simple transformations include adding a constant to all the entries or just to the diagonal of the kernel matrix. These simple transformations always have practice significance. For example the former operation corresponds to adding a constant feature and the latter one corresponds to adding different features to different inputs. Further transformation like normalizing data set in feature space can also be implemented by a short sequence of operations.

- Centering data
  By performing operations on kernel matrix, this transform can also be realized even though it is more complicated than the simple transformation above. The key technique is how to choose the data center. Normally we consider to take the origin where the sum of the norms of the elements is minimal. As we know, the sum of the norms equals to the trace of the kernel matrix which makes the operations more straightforward.

- Subspace projection
  This transformation is in fact a low-rank approximation of a kernel matrix. The number of non-zero eigenvalues corresponds to the space dimension if data points are constrained in a low-dimensional subspace which provides a better model of the data [71]. There are several ways to create low-rank approximations, for example projecting data into the space which is spanned by the first eigenvectors or implementing a partial Cholesky decomposition of the kernel matrix. In both of these methods, only the original kernel matrix is necessary to calculate inner product for new data.

Conclusion : Operation on kernel matrices can be considered as the first phase in data learning which tries to provide the most appropriate feature space. It is not difficult to understand that a good feature space will greatly improve the performance of kernel algorithms. All these operations correspond to moving data points in the feature space, sculpting the inner product matrix [71]. The new feature space defines a better topology which should consider the prior knowledge of data points in the input space.

2.4 Basic kernels

Two important properties should be considered when we try to apply a kernel function in applications: Firstly the kernel function enables us to measure data similarities in the feature space while to avoid giving the explicit embedding function ϕ. Secondly the computation complexity should be less than which would be needed in the case of providing an explicit embedding function ϕ. Kernel functions which meet the above conditions are not only vectorial inputs. There exists also kernels on graphs and kernels on real numbers etc [71]. However in this section, we only focus on kernels on vectors and give examples of most popular kernel functions.

2.4.1 Polynomial kernels

Definition 10. A **Polynomial kernel** is defined as

$$k_d(x, y) = (\langle x, y \rangle + R)^d \tag{2.4}$$

where R and d are parameters.

Notice that we could expand k_d by using the binomial formula

$$k_d(x, y) = \sum_{s=0}^{d} \begin{pmatrix} d \\ s \end{pmatrix} R^{d-s} (\langle x, y \rangle)^s \tag{2.5}$$

In section. 2.3.1.1, it was demonstrated that the feature corresponding to a sum of kernels is the concatenation of the features corresponding to each kernel. Given a general term of each polynomial kernel

$$\hat{k}_s(x, y) = \langle x, y \rangle^s, \text{for} \quad s = 0, ..., d \tag{2.6}$$

Eq. 2.5 can be considered as the sum of weighted polynomial kernels and the corresponding feature is also weighed.

Remark. The parameter R controls the weight of the different degree monomials. Eq. 2.5 could be written as

$$k_d(x, y) = \sum_{s=0}^{d} a_s \hat{k}_s(x, y) \tag{2.7}$$

where

$$a_s = \begin{pmatrix} d \\ s \end{pmatrix} R^{d-s} \tag{2.8}$$

By increasing R, we can decrease the weight of higher order monomials.

The following kernels are also very popular in applications which are in fact special cases of polynomial kernels.

- Linear kernels

$$k(x, y) = \langle x, y \rangle \tag{2.9}$$

  The linear kernel is a special case of polynomial kernels with $R = 0$ and $d = 1$. The mapping function ϕ is the identity function.

- Quadratic kernels

$$k_2(x, y) = \langle x, y \rangle^2 \tag{2.10}$$

2.4.2 Gaussian kernels

Gaussian kernels are the most widely used kernels in machine learning area. The definition of a gaussian kernel is given as follows:

Definition 11.

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

In section. 2.3.1 we have demonstrated that a gaussian kernel is a valid kernel function. It can be illustrated in another way that the gaussian kernel is obtained by normalizing the kernel function. Given a kernel k, with feature mapping ϕ, we normalize ϕ

$$\phi(x) \mapsto \frac{\phi(x)}{\|\phi(x)\|}$$

Then the *normalized kernel* k corresponding to the feature map is given

$$\hat{k}(x,y) = \langle \frac{\phi(x)}{\|\phi(x)\|}, \frac{\phi(y)}{\|\phi(y)\|} \rangle = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$$

Now, given the valid kernel $exp(\langle x,y\rangle/\sigma^2)$, we normalize this kernel and then we have

$$\begin{aligned} \frac{exp(\langle x,y\rangle/\sigma^2)}{\sqrt{exp(\|x\|^2/\sigma^2)exp(\|y\|^2/\sigma^2)}} &= exp(\frac{\langle x,y\rangle}{\sigma^2} - \frac{\langle x,x\rangle}{2\sigma^2} - \frac{\langle y,y\rangle}{2\sigma^2}) \\ &= exp(-\frac{\|x-y\|^2}{2\sigma^2}) \end{aligned}$$

We can consider the gaussian kernel as the result of the normalization of $exp(\langle x,y\rangle/\sigma^2)$.

Gaussian kernel is also referred to as the radial basis function kernel and hence could be used in radial basis networks. Gaussian kernel has the following properties

- All the points have a norm equal to 1 in the feature space induced by a gaussian kernel since
$$\|\phi(x)\|^2 = k(x,x) = 1$$
- All the points lie in a single orthant since $\langle \phi(x), \phi(y) \rangle = k(x,y)$ is always positive.
- The feature space induced by a gaussian kernel is infinite-dimensional. Which means that the mapping function ϕ can not been given explicitly.

Remark. The parameter σ controls the flexibility of gaussian kernels which is often referred to as *kernel width*. Similar to the parameter d in polynomial kernels, σ greatly influences different order-feature spaces. With a small values of σ, the kernel matrix is close to the identity matrix. Algorithms almost fit any labels which leads to a risk of overfitting. When σ is large, the kernel matrix is close to a constant matrix with all the elements equals to 1. All points in the feature space seem identical.

Gaussian kernel has been demonstrated to outperform polynomial kernel in many fields like kernel Principal Component Analysis (KPCA). In the following chapters, our work focus on the gaussian kernel. 'Kernel' will refer to the gaussian kernel if not specified.

Conclusion In this chapter, we have provided fundamental kernel theory. The positive semi-definite property of kernel function and kernel matrix has been highlighted. It plays a central role in kernel validation. Finally we introduced the most commonly used kernel: polynomial kernels and Gaussian kernels. Notice that we will only consider Gaussian kernels in our following clustering algorithms evaluations if not specified.

Chapter 3

Maximum Mean Discrepancy and Hierarchical Agglomerative Clustering via Ward linkage

Contents

Maximum Mean Discrepancy (MMD) is a *distance* measure between two probability density functions. It has been used to compare distributions. Maximum

Mean Discrepancy (MMD) firstly owned its prominence in comparison of distributions problems. It can be used to compare different distributions. As density functions of distributions can be embedded in a Reproducing Kernel Hilbert Space (RKHS), which means that, in a RKHS, it is possible as well to compare different distributions where the corresponding Maximum Mean Discrepancy is straightforward to estimate. Considering that the square of Maximum Mean Discrepancy is strongly connected to the Ward criterion, which is widely used in Hierarchical Agglomerative Classification, we propose to consider the Kernel-based Hierarchical Agglomerative algorithms as a tool to perform clustering.

In this chapter, we firstly provide the reader with some fundamentals of Maximum Mean Discrepancy and its estimation in a RKHS. Then we show the limitations of squared MMD when it is considered as a measure of quality of a binary partition and we introduce weighted Squared MMD (WSMMD). In the next, we recall some elements of HAC and finally we introduce its kernelized version (KHAC).

3.1 Maximum Mean Discrepancy (MMD)

3.1.1 Theoretical Foundations of MMD

A lot of relevant work of MMD has been done by Gretton et al. [29]. We recall some fundamentals of it.

Definition 12. ***Maximum Mean Discrepancy*** [23]: Suppose that $\mathcal{F}$ is a class of functions f: $\mathcal{X} \to \mathbb{R}$ with p and q the Borel probabilistic measures. The Maximum Mean Discrepancy (MMD) is defined as:

$$MMD[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} (\mathbf{E}_p[f(x)] - \mathbf{E}_q[f(y)]) \tag{3.1}$$

MMD characterizes identical distributions but also allows to measure some *distance* between different distributions. Unfortunately, as we generally do not know p and q, we even cannot imagine to compute the corresponding MMD. We can estimate MMD by the sample mean of Eq.A.4 (we suppose that X=(x_1,...,x_m) and Y=(y_1,...,y_n) are two ensembles of iid observations drawn from distributions

p and q respectively) and we have:

$$\widehat{MMD}[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{i=1}^{n} f(y_i) \right) \tag{3.2}$$

where m and n are the number of observations drawn from p and q respectively.

Theorem 1. [20] Suppose that ($\mathcal{X}$,d) is a metric space and p,q two ensembles of Borel probabilistic measures defined on $\mathcal{X}$, $p = q$ if and only if $\mathbf{E}_p[f(x)] = \mathbf{E}_q[f(y)]$ for any function $f \in C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of continuous bounded functions and x, y are random variables drawn from distribution p and q respectively.

This theorem could help the practitioner to develop a test of equality of two distributions if we could estimate MMD, which can be performed in a universal Reproducing Kernel Hilbert Space.

3.1.2 MMD embedded in RKHS

Being inspired by the pioneering work of Fortet and Mourier [23] on Maximum Mean Discrepancy (MMD), Smola [73] has shown that it is possible to perform a two sample test of equality of distributions by embedding distributions in a RKHS, where a distribution is represented by $E_x(\phi(x))$. In this space, estimation of the distance between the distributions is straightforward and the decision threshold (or equivalently the $p-value$) is easily determined using the asymptotic normality of the decision statistic.

Smola [73] and Gretton et al. [29] have demonstrated that MMD can be calculated in a RKHS ($\mathcal{H}$) which provides us linear solutions for higher order statistics problems [74] owing to its reproducing properties. For any $f \in \mathcal{H}$, we have:

$$\begin{aligned} \langle k(x, .), f(.) \rangle_{\mathcal{H}} &= f(x) \\ \langle k(x, .), k(x', .) \rangle_{\mathcal{H}} &= k(x, x') \end{aligned} \tag{3.3}$$

More details properties about RKHS can be found in Chapter. 2.

Theorem 2. [77], [73]$MMD[\mathcal{F}, p, q] = 0$ iff $p = q$ when $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$ provided that $\mathcal{H}$ is universal (which means that $k(x, .)$ is continuous for all x and the RKHS induced by k is dense in $C(\mathcal{X})$)).

According to Eq. A.4 and Eq. 3.3

$$\begin{aligned} MMD[\mathcal{F}, p, q] &= \sup_{f:\|f\|\leq 1} (\mathbf{E}_p[f(x)] - \mathbf{E}_q[f(y)]) \\ &= \sup_{f:\|f\|\leq 1} (\mathbf{E}_p[\langle k(x,.), f(.)\rangle_{\mathcal{H}}] - \mathbf{E}_q[\langle k(y,.), f(.)\rangle_{\mathcal{H}}]) \\ &= \sup_{f:\|f\|\leq 1} (\langle \mathbf{E}_p[k(x,.)], f(.)\rangle_{\mathcal{H}} - \langle \mathbf{E}_q[k(y,.)], f(.)\rangle_{\mathcal{H}} \\ &= \sup_{f:\|f\|\leq 1} (\langle \mu_p, f\rangle_{\mathcal{H}} - \langle \mu_q, f\rangle_{\mathcal{H}}) \\ &= \sup_{f:\|f\|\leq 1} (\langle \mu_p - \mu_q, f\rangle_{\mathcal{H}}) \end{aligned}$$

whose maximum is reached (with $\|f\| \leq 1$) when $f = \frac{\mu_p - \mu_q}{\|\mu_p - \mu_q\|_{\mathcal{H}}}$, which gives:

$$MMD[\mathcal{F}, p, q] = \|\mu_p - \mu_q\|_{\mathcal{H}}$$

Theorem 3. [29]Given a universal RKHS, the squared MMD can be expressed as follows:

$$\begin{aligned} MMD^2[\mathcal{F}, p, q] &= \|\mu_p - \mu_q\|^2_{\mathcal{H}} \\ &= \mathbf{E}_{x,x'}[k(x,x')] - 2\mathbf{E}_{x,y}[k(x,y)] + \mathbf{E}_{y,y'}[k(y,y')] \end{aligned} \tag{3.4}$$

where x and x' are independent observations from distribution p, y and y' are independent observations from distribution q. Its corresponding unbiased ([70]) finite sample estimate is given by:

$$\begin{aligned} \widehat{MMD}^2_u[\mathcal{F}, X, Y] = &\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j\neq i}^{m} k(x_i, x_j) \\ &+ \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j\neq i}^{n} k(y_i, y_j) \\ &- \frac{2}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, y_j) \end{aligned} \tag{3.5}$$

where $x_i, i = 1, \ldots, m$ and $y_i, i = 1, \ldots, n$ are iid examples drawn from p and q respectively. This approach has been validated by Smola [73] and Gretton [29] et al. to test equality of two distributions.

3.1.3 MMD based clustering

As presented before, MMD can be used as a dissimilarity measure between two distributions. It can also be easily expressed after embedding the distributions in a RKHS with the help of a kernel function. In a RKHS, MMD can be estimated using any two labeled class data sets. This inspires us to consider MMD as a criterion to be optimized to perform a clustering task. In this case, the unknown data labels would result from the optimization of the MMD between the distributions. Obviously, we must propose an optimization method that avoids the computational burden which derives from combinatorial aspects (for a binary clustering problem of an unlabeled data set composed of N observations, there are 2^N possible label solutions).

3.1.3.1 Squared MMD based clustering

We consider Squared MMD (SMMD) as a separability measure. We try to evaluate MMD in a (universal) RKHS, expecting that after some nonlinear mapping of the data we can improve cluster separability. We also expect that the best clustering result will be obtained when SMMD is maximized. In the following, as specified before, we only consider the gaussian kernel with parameter σ, which leads to a universal RKHS [58].

To check this assumption, we take the data represented in Figure 3.1 and calculate the Squared MMD (SMMD) for different partitions, which is a function of the kernel parameter σ ($\sigma \in [0.01, 3.0]$). Looking at the different curves obtained on Fig. 1a, we can see that when the SMMD is maximal w.r.t the partitions considered (see in particular Fig. 1c), we obtain uneven-sized clusters for which the obtained labels are far from the initial ones provided in this simulation. Among the partitions considered, a good one (see Fig. 1b) does not correspond to a high MMD, and this remains true whatever σ is.

Remark. The example above shows that using SMMD to be a criterion to perform clustering sometimes leads to imbalanced clusters.

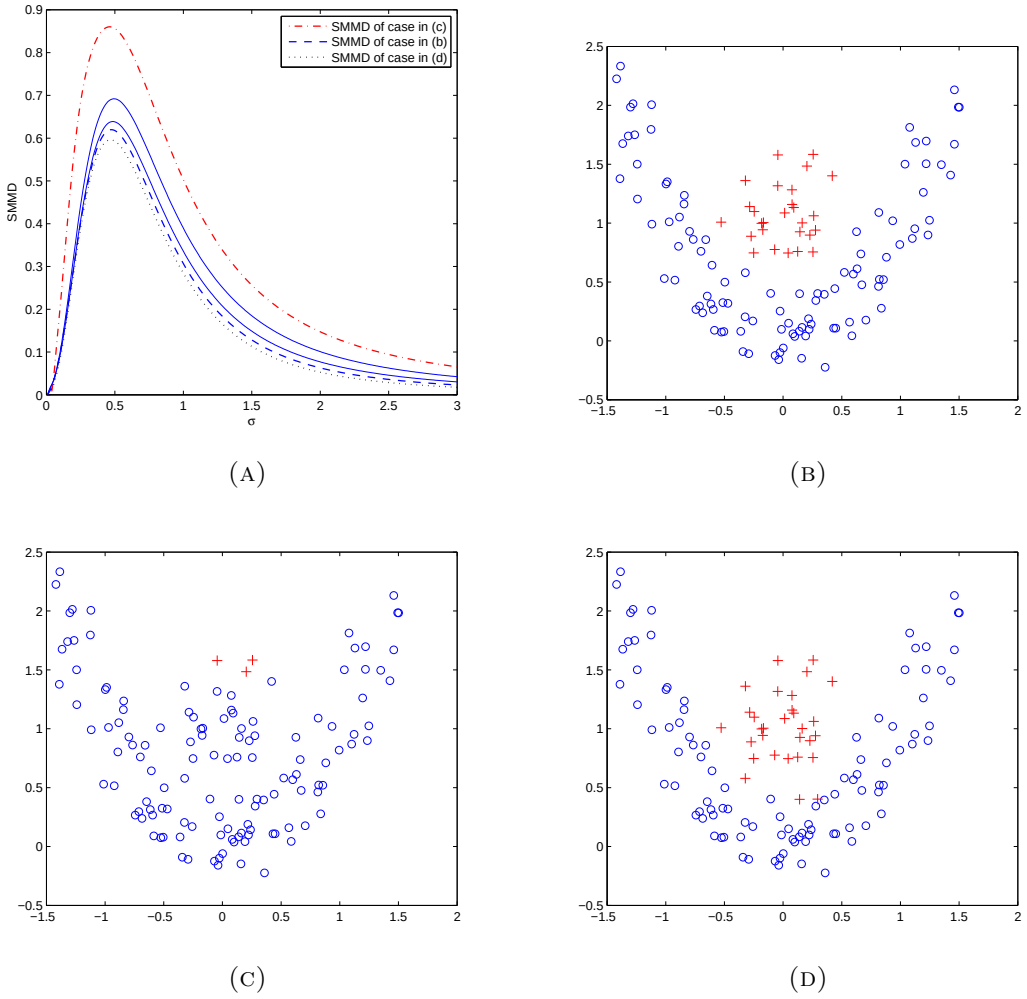


FIGURE 3.1: SMMD evaluated for different Partitions. a: SMMD as a function of the kernel parameter σ for 5 different partitions. b: Intermediate case. c: Best-SMMD based partition among partitions considered. d: Worst-SMMD based partition among partitions considered (original clusters)

3.1.3.2 Weighted MMD based clustering

To obtain more adequate solutions, we considered to take into account the number of observations of the obtained clusters. To do so, we scale the SMMD by $mn/(m+n)$ (where m and n are the number of samples in classes p and q respectively)

$$WMMD^2[\mathcal{F}, p, q] = \frac{mn}{m+n}\|\mu_p - \mu_q\|^2_{\mathcal{H}} \tag{3.6}$$

This criterion is noted as Weighted Squared MMD (WSMMD). Experiments with the same toy dataset as in Figure 3.1 show that maximizing WSMMD corresponds to our assumption: satisfactory clustering is obtained when WSMMD reaches its

maximum (see Fig. 1b). We observe that WSMMD favors balanced clustering, unlike the squared MMD.

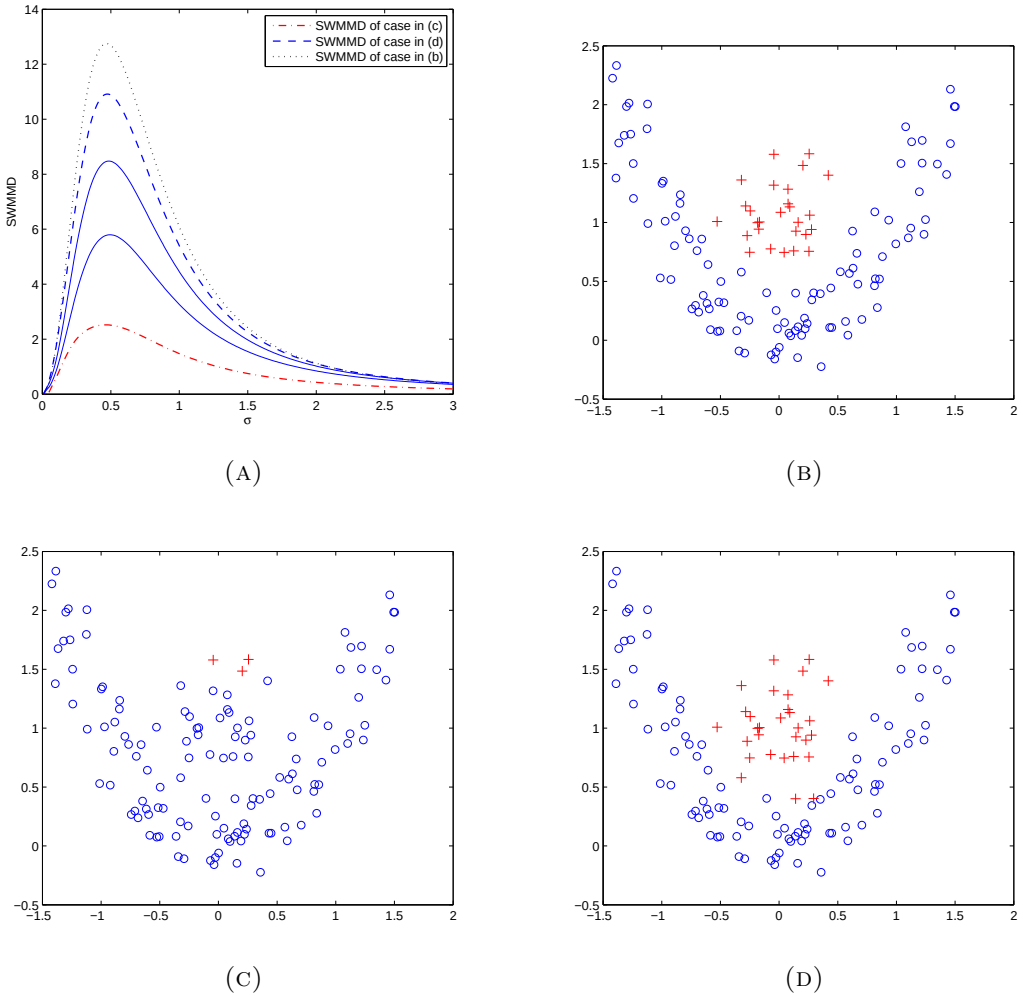


FIGURE 3.2: WSMMD evaluated for the same partitions as in Fig. 3.1. a: WSMMD as a function of the kernel parameter σ. b: Best-WSMMD based partition among partitions considered. c: Worst-WSMMD based partition among partitions considered. d: Intermediate case (original clusters)

Conclusion This result, compared with SMMD, is not surprising since WSMMD is equivalent to the Ward criterion [82] which corresponds to the loss of between-class dispersion when merging two clusters. The Ward criterion is a frequently used dissimilarity measure in Hierarchical Agglomerative Clustering. Considering that WSMMD (equivalent to Ward criterion), like standard squared MMD, can be easily calculated in a RKHS, we propose to do further research on Kernel-based HAC (KHAC) via Ward linkage.

3.2 Hierarchical Agglomerative Clustering

3.2.1 Introduction

Hierarchical Agglomerative Classification (HAC) methods have owned prominent success in some automatic classification problems because of their abilities of providing an accurate description of the data structure by using a *dendrogram* which illustrates the arrangement of the classes produced by HAC. In some empirical fields, investigators are not only interested in finding out the homogenous groups but also exploring more interior *connections* between objects in the whole data set. A lot of research has been done in the earlier years and a general framework of a hierarchical classification scheme has been proposed and evaluated in [51].

In general, hierarchical classification can be divided into two categories:

- Hierarchical agglomerative classification
- Hierarchical divisive classification

with the previous being a bottom-up approach and the latter being a top-down approach.

Hierarchical classification solutions have been primarily obtained by using agglomerative schemes [31]. In an agglomerative scheme, each data point is firstly considered to be a singleton, then pairs of most similar (according to a given similarity measure) objects are merged to form one larger group. This procedure will iteratively continue until at last only one group is left, including all the data points.

A general agglomerative algorithm for hierarchical classification scheme may be given as follows:

- *Step 1* Assign each object to be a singleton.
- *Step 2* Search the the most similar pair of objects/groups using an appropriate similarity/disimilarity measure.
- *Step 3* Merge these two objects/groups into one and update the similarity/-disimilarity measure.

- *Step 4* Go back to *step 2* until only one group is left.

3.2.2 Similarity or Dissimilarity measure

The key to the previous process is to define and to update the similarity/dissimilarity measure. Different classification strategies differ in the manner of defining this measure. A general formula has been proposed in [51]. In *step 2*, we assume G_i and G_j being the two groups to be fused and G_k being the group after the fusion. d_{ij} is the dissimilarity between G_i and G_j, where d_{ij} is the smallest value in the dissimilarity matrix. Then the updated dissimilarity between G_k and any other group G_h is defined as:

$$d_{hk} = \alpha_i d_{hi} + \alpha_j d_{hj} + \beta d_{ij} + \gamma |d_{hi} - d_{hj}| \tag{3.7}$$

where α_i, α_j, β and γ are parameters that determinate the different strategies.

The advantage of this general formula is the recursive update of the dissimilarity without retaining the initial pairwise dissimilarity. A more general formula has been proposed by [43]. Milligan [59] also demonstrates that hierarchical classification strategy is monotonic, i.e $d_{hk} \geqslant d_{ij}$ iff

$$\gamma \geq 0 \cup (\gamma < 0 \cap |\gamma| \leq min(\alpha_i, \alpha_j)) \tag{3.8}$$

$$min(\alpha_i, \alpha_j) \geq 0 \tag{3.9}$$

$$\alpha_i + \alpha_j + \beta \geq 1 \tag{3.10}$$

3.2.2.1 Ultrametric inequality

By using formula 3.7 which holds the previous conditions, hierarchical methods can always produce monotonic hierarchies. A simple proof has been given by [44] and we have

$$d_{ij} \leq max(d_{ih}, d_{jh}) \tag{3.11}$$

This equation is clearly stronger than the normal triangular inequality (Eq. 3.12) and satisfies the ultrametric inequality.

$$d_{ij} \leq d_{ih} + d_{jh} \tag{3.12}$$

So the metric that we use to construct the hierarchy (Ultrametric) is much stronger than the metric by which the data set is evaluated (Euclidean metric).

Remark. The solution of a hierarchical method is actually a transform from the dissimilarity matrix to an ultrametric structure.

3.2.2.2 Hierarchical strategies of HAC

Table 3.1 shows several well known hierarchical strategies and associated parameters which satisfy the general formula as given by William [15].

TABLE 3.1: Parameter values for different Hierarchical strategies

Name	α_i	α_j	β	γ
Single linkage (Nearest neighbor)	1/2	1/2	0	-1/2
Complete linkage (Furthest neighbor)	1/2	1/2	0	1/2
Group Average (UPGMA)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Minimum Variance (Ward)	$\frac{n_i+n_h}{n_i+n_j+n_h}$	$\frac{n_j+n_h}{n_i+n_j+n_h}$	$\frac{-n_k}{n_i+n_j+n_h}$	0

*n_i is the number of objects in group i

- Single linkage
  $d(r,s) = min(d(x_{ri}, x_{sj})), \quad x_{ri} \in r, \quad x_{sj} \in s$
- Complete linkage
  $d(r,s) = max(d(x_{ri}, x_{sj})), \quad x_{ri} \in r, \quad x_{sj} \in s$
- Average linkage
  $d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} d(x_{ri}, x_{sj})$
- Ward's linkage
  $d^2(r,s) = n_r n_s \frac{\|\bar{x}_r - \bar{x}_s\|_2^2}{(n_r+n_s)}$, where $\bar{x}_r = \frac{1}{n_r} \sum_{i=1}^{n_r} x_{ri}$

The single linkage strategy, which is also mentioned as the nearest-neighbor strategy is one of the earliest conventional strategies. The dissimilarity measure between two groups is defined by the distance between the closest elements in any two different groups. The complete linkage uses the distance between the furthest elements. Average linkage take into account all the elements in the considered two groups and the dissimilarity is defined by weighted sum of all the pairwise distances between two groups. Each strategy has advantages and drawbacks depending on the data structure. In this document we consider Ward's linkage. Ward's linkage was first proposed by [82] and has been widely applied in HAC since then. At each step of agglomeration, the two groups where the increase of within-class variance is minimum are merged.

Theorem 4. Recall the definition of Weighted Squared MMD (WSMMD)

$$WMMD^2[\mathcal{F}, p, q] = \frac{mn}{m+n}\|\mu_p - \mu_q\|^2_{\mathcal{H}} \tag{3.13}$$

(where m and n are the number of samples in classes p and q respectively) is equivalent to ward linkage in monotonic hierarchical procedure.

Remark. Another advantage of HAC methods lies in the tree diagram often called *dendrogram* which gives a intuitive description of the data set. In a *dendrogram*, a height h_{ij} is associated with the amalgamation of two groups i, j. To obtain a valid *dendrogram*, a necessary and sufficient condition is that h_{ij} satisfies the ultrametric inequality:

$$h_{ij} \leq max(h_{ik}, h_{jk}) \ \forall i, j, k \in \Omega \tag{3.14}$$

3.3 Kernel-based Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Classification methods have been widely used in many fields. However not much work concerns research on kernelized HAC methods.

$\phi : X \to F$ performs the mapping from the original space onto the (high dimensional) feature space. Inner product calculations in the feature space can be computed by a kernel function in the original space, without explicitly specifying

the mapping function Φ. Computation of Euclidean distances in the feature space F benefits from this idea.

$$d^2(\Phi(x_i), \Phi(x_j)) = \|\Phi(x_i) - \Phi(x_j)\|^2 \tag{3.15}$$
$$= k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j) \tag{3.16}$$

Several commonly used Mercer kernels are listed in [80]. In this paper, we only consider the gaussian kernel defined as follows:

$$k(x_i, x_j) = \exp\left(-\frac{\| x_i - x_j \|^2}{2\sigma^2}\right) \tag{3.17}$$

The main idea used here can be applied to any universal kernel based HAC for which the kernel is of the form

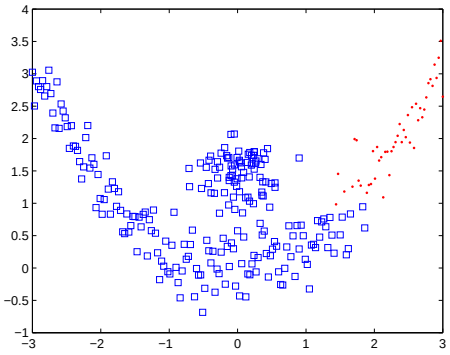
$$k(x, y) = f(\| x - y \|) \tag{3.18}$$

The aim of introducing the kernel trick in HAC is to easily explore wider classes of dissimilarity measures. Fig. 3.3 shows an example of a 2-classes data set that cannot be clustered (by cutting the dendrogram in such a way to obtain 2 clusters) by standard HAC (Fig. 3.3a) while Kernel HAC allows perfect clustering (Fig. 3.3c).

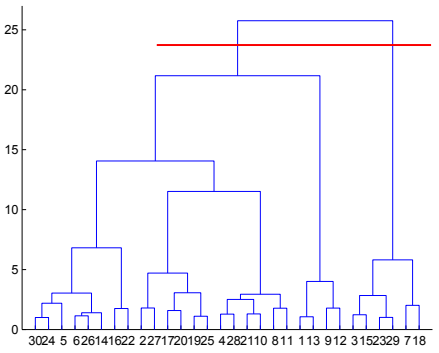
In this example, we suppose the number of class $K = 2$ is known, so the algorithm cuts the dendrogram at the level where two parts are obtained with each part corresponding to one class. Comparing the dendrogram achieved by HAC (Fig. 3.3a) and Kernel HAC (Fig. 3.3c), we observed that by choosing the proper kernel parameter, the dendrogram of Kernel HAC shows a more interesting result in dividing the data set into two classes. This results from the nonlinear transformation of the input space induced by the gaussian kernel $k(x, .)$.

3.4 Conclusion

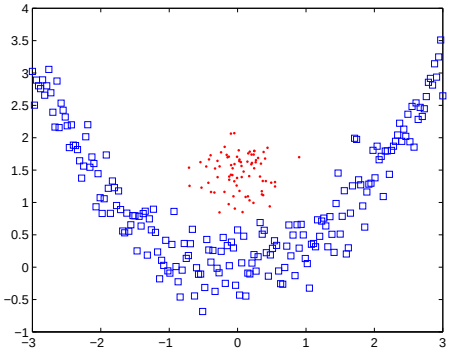
In this chapter, we took the Maximum Mean Discrepancy, estimated in a RKHS, as a distance between two distributions to perform clustering. On some toy data sets,



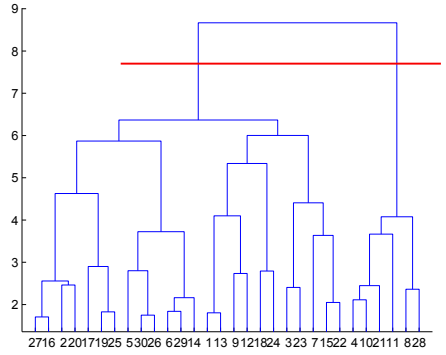
(A) 2 clusters obtained with HAC using Ward criterion



(B) Dendrogram of HAC



(C) 2 clusters obtained with KHAC using Ward criterion



(D) Dendrogram of Kernel HAC

FIGURE 3.3: Illustration of HAC and Kernel HAC for a non-linearly separable data set

we showed that a high value of MMD does not correspond to a good partitioning of the data because MMD generally leads to uneven groups. We then considered to take into account the number of observations of the clusters to obtain a new distance whose maximization provides more satisfactory groups. The equivalence of this distance and the Ward criterion [82] frequently used in HAC has been demonstrated. With this distance being estimated in a RKHS, we introduced Kernel Hierarchical Agglomerative Classification (KHAC).

In the following chapters, we will present the fruits of our research on kernel parameter selection methods based on KHAC, detection of number of clusters and KHAC algorithms etc.

Chapter 4

Kernel parameter selection for KHAC

Contents

In this chapter, our goal is to find a method to choose a proper kernel for KHAC. We try to explore a criterion for kernel parameter selection adapted to KHAC based on the kernel research above. *Alignment* has firstly been evaluated as a kernel selection criterion and *centered alignment* is proposed after some

limitations of *Alignment* being demonstrated. We also propose *inter cluster distance* as a criterion for kernel parameter selection. Experiments have been done with all these criteria and other aspects, like proportion of 2 classes, have also been considered.

4.1 Introduction

Kernel-based clustering has drawn much attention since it has been confirmed to be a good technique in dealing with non linearly separable clusters. One of the challenges when we consider to apply kernel-based clustering is how to choose a proper kernel parameter. According to the literature, choosing the proper kernel function has often been a trial-and-error task. Up to now, not enough methods have been proposed to solve this problem and the determination of the kernel parameter is still an open research.

Determination of kernel parameters began with SVM since when the kernel trick owned its reputation. Min and Lee [62] used a grid-search technique using 5-fold cross-validation to find out the optimal parameter values of kernel function of SVM, which is a conventional method in the earlier years but a little expensive [16]. Then several model selection methods have been proposed and among them margin bound has given good performances [11]. In later research, Wu and Wang [83] proposed a method to choose the kernel parameter of SVM by inter-cluster distance in the feature space, which provides competitive results with the grid-search technique but with a simple distance calculation and a non-iterative process. This method will also be considered in clustering for kernel parameter selection in our later work.

In the last few years, a few methods to determinate the kernel parameters in clustering have been proposed and most of them are somewhat ad-hoc under a certain

kind of clustering algorithms. Unlike in SVM, clustering algorithms differ a lot in similarity measure and optimized function, so the selection of kernel parameter is problem dependent. A general idea is to work on the Gram Matrix K which is the pairwise similarity matrix between all pairs of points in the data set. This matrix can straightforwardly specify kernel features.

Early research on kernel learning investigated the idea of learning the kernel from data ([1],[45],[12],[52]). Research focused on specifying a kernel family rather than a specified kernel. The most widely use kernel family is a convex combination of a finite set of base kernels. Even though much work has been done and different methods have been tried, to our knowledge, the most approved and successful one is still *uniform combination* solution which concentrate on learning a hypothesis out of the RKHS associated to a uniform combination of base kernels. Theoretical results provides favorable guarantees of kernel choosing and related algorithms have been proposed.

Kandola et al. [46] proposed a method to measure the degree of agreement between a kernel and a learning task which is called *Alignment*. A higher agreement is expected when the kernel is well chosen, so this provides a method to choose a proper kernel function where the proper kernel parameter is determined. This indicator can not only be used to assess the coherence between the clustering induced by a kernel and that by the labels of the data points themselves, but also be capable to measure the similarity between two kernels. Two kernels are equivalent if they induce the same Gram matrix so they provide the same clustering.

Theoretically, *Alignment* gives satisfactory results. However according to our experimental testing, this is not always the case. Cortes et al. [13] confirmed the same conclusion and in their paper they proposed a better criterion: *centered alignment* whose performances even outperforms the *uniform combination* solution.

4.2 Alignment

Alignment, a criterion proposed by Cristianini et al. [46], provides a method to evaluate the Gram matrix which contains information of pairwise points in the feature space. Gram matrix is the key notion in kernel method as most of the information about data set in feature space is given by it.

Alignment is in fact a Frobenius inner product between kernel matrixes or between a kernel matrix and a learning task (associated labels). According to research of Kandola et al. [46], the calculation of *Alignment* is independent of the clustering algorithm because it is based only on kernel matrix which is obtained by the kernel function and data information. It is demonstrated in their paper that *Alignment* is sharply concentrated around its expected value and hence the empirical value is stable whatever the clustering result is.

In the following, we give some basic notions about *Alignment* defined in the paper of Kandola et al. [46].

4.2.1 Alignment between two kernels

Definition 13. The empirical *Alignment* between two kernels K_1 K_2 is given by:

$$A = \frac{< K_1, K_2 >_F}{\sqrt{< K_1, K_1 >_F}\sqrt{< K_2, K_2 >_F}} \tag{4.1}$$

where the Frobenius product between two matrices is denoted as $< X, Y >_F = \sum_{ij} x_{ij} y_{ij} = Tr(X'Y)$. In this definition, *Alignment* is in fact the cosine angle between two Gram Matrices. It reaches its maximum when two kernel matrix are identical and equals to zero when they are orthogonal. The Cauchy-Schwarz inequality states that $-1 \leq A \leq 1$.

4.2.2 Alignment between a kernel and a learning task

In the same way, *Alignment* could be used to measure the *Alignment* between a kernel and a learning task. The kernel is better aligned with the given data set if they have a higher degree of agreement. Given a kernel $K : X^2 \to [-1, +1]$, a labeled data set from $X \times \{-1, +1\}$ and the vector of labels $y \in \{-1, +1\}^p$, we consider $K_2 = yy'$, the *Alignment* between the Gram matrix K and the matrix yy' obtained by the label vector can be written as:

Definition 14. The empirical *Alignment* between K and yy' is given by:

$$A = \frac{< K, yy^{'} >_F}{\sqrt{< K, K >_F < yy^{'}, yy^{'} >_F}} \tag{4.2}$$

A high *Alignment* indicates a better kernel for the learning task.

4.2.3 Alignment applications

Alignment has several interesting practical applications.

- It provides a kernel combination validation rule: by combining two kernels which have a hight Alignment with the target respectively but a low Alignment with each other, split result is improved.
- A better kernel choice could be made by improving the Alignment with a certain task data set and on the contrary an optimal label combination could be chosen with the kernel function fixed by maximizing Alignment.
- Clustering algorithms have been proposed by maximizing Alignment.

In the second application mentioned above, with a label vector y fixed, by maximizing *Alignment*, an optimal kernel could be obtained. Our methods of kernel parameter selection used in the following part are based on this idea.

Fig. 4.1 shows an example how *Alignment* varies when the kernel parameter σ changes. The value of *Alignment* indicates the agreement between a kernel function and the learning task. We expect that the 'best' kernel function will provide a maximum *Alignment*.

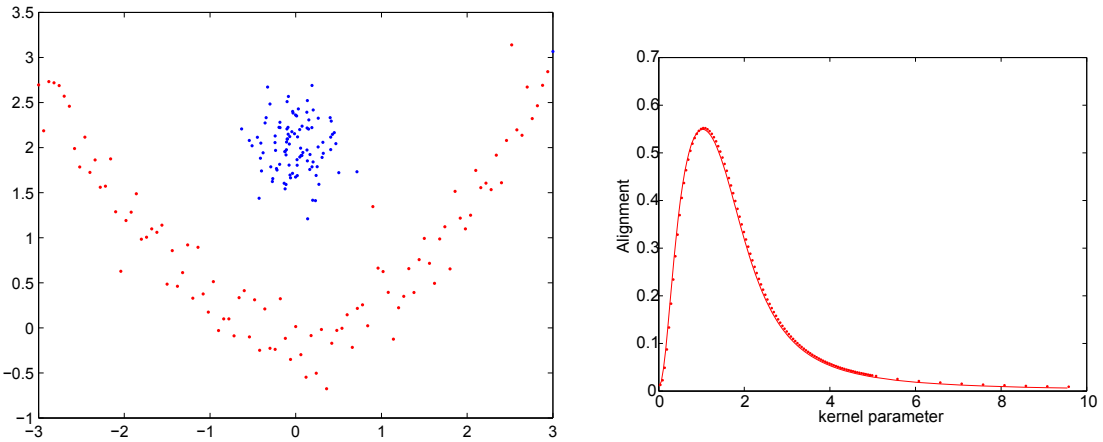


FIGURE 4.1: Alignment evaluation as a function of the kernel parameter σ

4.3 Gaussian Kernel parameter selection by Alignment

In supervised or semi-supervised classification, the matrix yy' is known so it can be used to maximize *Alignment*. However in clustering problems, the label vector y is unknown. So we can not calculate the *Alignment* before the clustering algorithm being carried out. Most of kernel parameter selection methods are based on trial and error: we carry out the clustering algorithm with different kernel parameters and then we compare the split results.

Here our idea is to maximize *Alignment* with $y(\sigma)$ which is in fact a function of the kernel parameter σ. In our experiment, we vary the kernel parameter to obtain the associated y, we calculate the function $A(\sigma)$,

$$A(\sigma) = \frac{< K, y(\sigma)y(\sigma)' >_F}{\sqrt{< K, K >_F < y(\sigma)y(\sigma)', y(\sigma)y(\sigma)' >_F}} \tag{4.3}$$

the optimal σ^* locates at the point where the *Alignment* is maximum. We then have

$$\sigma^* = \arg\max_{\sigma} A(\sigma) = \arg\max_{\sigma} \frac{< K, y(\sigma)y(\sigma)' >_F}{\sqrt{< K, K >_F < y(\sigma)y(\sigma)', y(\sigma)y(\sigma)' >_F}} \tag{4.4}$$

4.3.1 Criteria for kernel parameter selection

In this part, we propose several criteria to evaluate the kernel parameter selection, and compar them with *Alignment* [46]. According to Cristianini [46], *Alignment* is also a measure of clusterization. Seeing in Eq.A.10, the numerator of *Alignment*

$$< K, yy' >_F = \Sigma_{y_i=y_j} K(x_i, x_j) - \Sigma_{y_i \neq y_j} K(x_i, x_j) \tag{4.5}$$

is related to within class distances and between classes distances. The term $\Sigma_{y_i=y_j} K(x_i, x_j)$ concerns pairwise similarities of data in the same cluster and similarly $\Sigma_{y_i \neq y_j} K(x_i, x_j)$ concerns pairwise similarities of data in different clusters. As we can see in the literature, inter-class dispersion has been proposed to be a good criterion of class generalization. So in our experiments, we consider also using within and between class dispersions to estimate the optimal kernel parameters.

Here we list the criteria that we have considered for selecting the kernel parameters:

- Alignment (theoretical)
  The definition of this criterion is asin Eq.A.10, with label vector y containing real labels supposed known. This *Alignment* is used to be compared with the other criteria as y is the desirable output of clustering algorithms.
- Alignment (experimental)
  Similar to *Alignment* (theoretical), the definition of this criterion is as Eq.A.11 where y is the estimated label vector after applying the clustering algorithm.
- Inter-class dispersion

$$\text{Inter} = \Sigma_{r=1}^{k} n_r \|\bar{x}_r - \bar{x}_0\|_{\Phi}^2 \tag{4.6}$$

- Similitude

$$\text{Similitude} = \frac{\text{number of well classified data points}}{\text{total number of data points}} \tag{4.7}$$

All these criteria are calculated in the feature space $\mathcal{F}$. In our experiments, to evaluate these criteria, we consider Kernel HAC using Ward criterion as the clustering algorithm. We vary the kernel parameter σ and we observe the evaluation of these criteria.

Criteria analysis

For the calculation of *Alignment* (theoretical), the label vector y is invariable and the only element which changes is the Gram Matrix K. So this criterion could clearly illustrate how K could influence *Alignment*. However for *Alignment* (experimental), not only K changes but also y varies when the kernel parameter σ changes. As a quantity who has been widely used in cluster validation index [42], inter cluster distance has been demonstrated to be a satisfactory criterion in clustering. It measures cluster 'compactness'. As for Similitude, it measures the good classification rate. This criterion can not be calculated in real problem because the true labels are unknown and we can not identify the 'number of well classified data points'. The meaning of proposing this criterion here is to help us in evaluating the effectiveness of other criteria.

4.3.2 Kernel parameter selection evaluation

In this experiment, we test the criteria above on 3 common used toy data sets which are 2-class problems. We vary the kernel parameter σ from 0.01 to 10. Inter-cluster distance here is normalized by its maximum. Experiment results are shown in Fig. A.4.

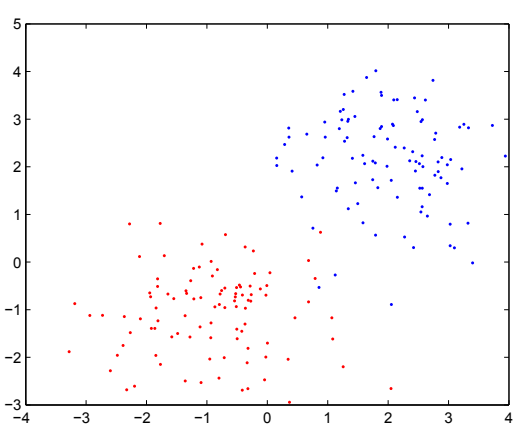
Looking at the results, a hight *Alignment* (theoretical) always corresponds to a good clustering result (a high similitude value) but conversely, that is not true. This is not expected as in the research of Kandola et al. [46]. Relevant research will be given later. Inter-cluster distance shows very competitive performance comparing with *Alignment* (theoretical), which uses the true labels. However, *Alignment* (experimental) is not as good as we expect. The curve of *Alignment* is not always continuous. Sometimes there is a 'jump' on the curve, where the *Alignment* rises but similitude goes down.

4.3.3 Limitations of alignment as kernel parameter selection criterion

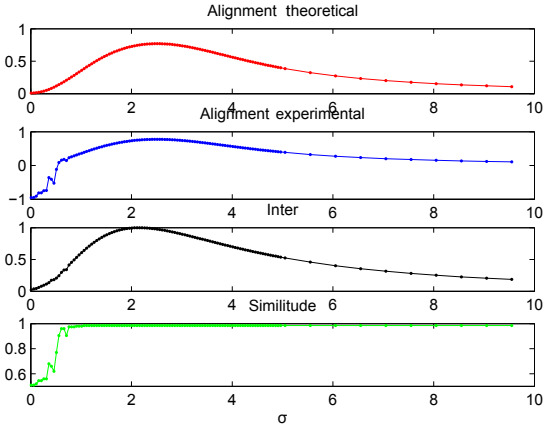
The experimental results shown in Fig. A.4 indicate some limitations of *Alignment* as a kernel parameter selection criterion. Discontinuity of the *Alignment* curve makes the *Alignment* not robust enough to select the optical kernel parameter. Fig. 4.3 shows an example of data split results before and after the 'Jump'. With kernel parameter σ before the 'Jump', we obtain the expected split result with a smaller value of *Alignment* than this of the case just after the 'Jump', where the split result is not satisfactory. This is against the anticipatory idea proposed above for kernel parameter selection. In the following part, we analyze the 'Jump' using both theoretical and experimental methods.

4.3.3.1 Theoretical analysis

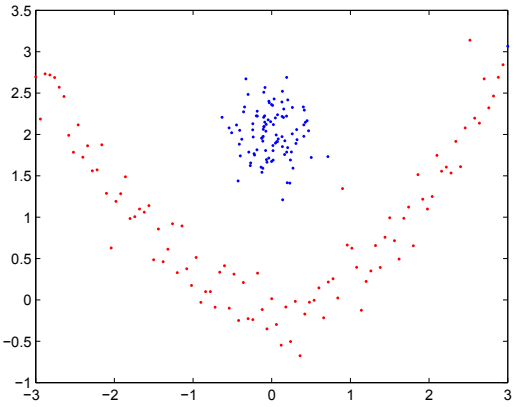
As shown in Fig. A.4, *Alignment* (experimental) is not as good as *Alignment* (Theoretical). However the calculation of them only differ in y. So we infer that when we vary the kernel parameter, the change in y will greatly influence the value of *Alignment*. Even with a little change of σ, the clustering results change a lot,



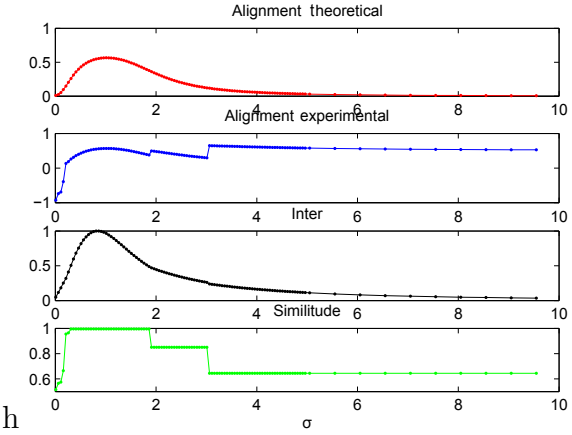
(A) Data set



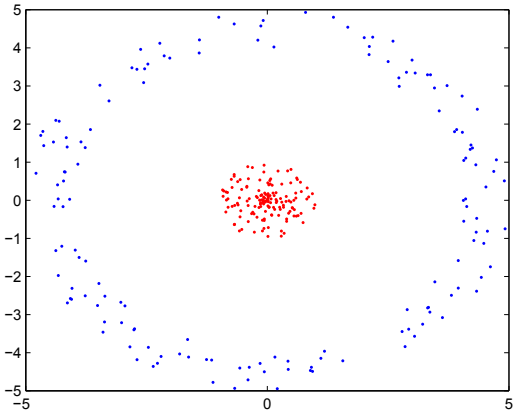
(B) Criteria evaluation



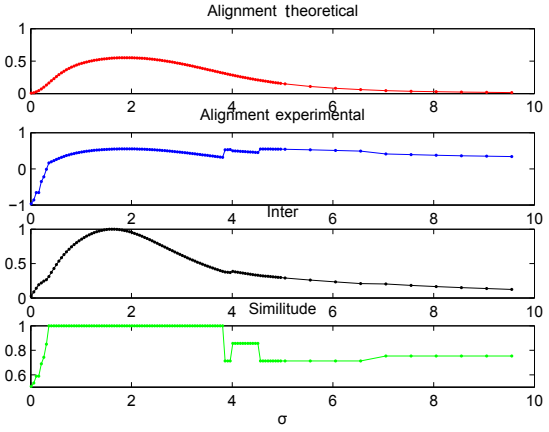
(C) Data set



(D) Criteria evaluation



(E) Data set

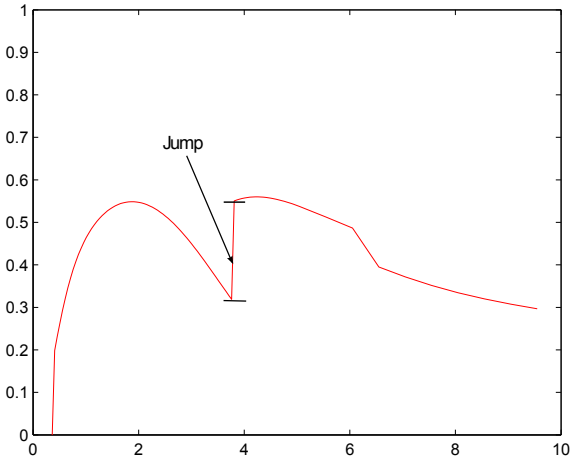


(F) Criteria evaluation

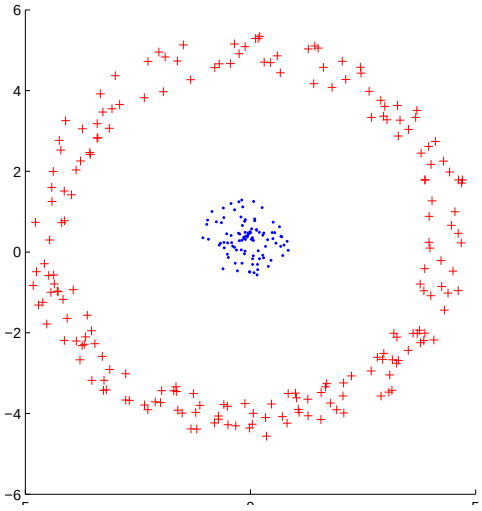
FIGURE 4.2: Evaluation of kernel selection criteria

which means that the label vector y changes. Here we recall the definition of *Alignment*

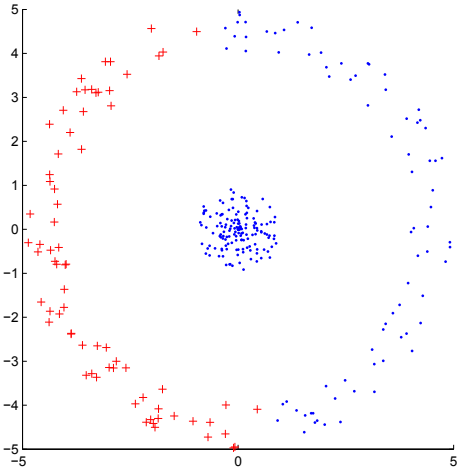
$$A = \frac{< K, yy^{'} >_F}{\sqrt{< K, K >_F < yy^{'}, yy^{'} >_F}} \tag{4.8}$$



(A) Alignment (experimental) evaluation in case of a 'Jump'



(B) Clustering results before 'jump'



(C) Clustering results after 'jump'

FIGURE 4.3: 'Jump' analysis.

With a little change of σ, K is changed smoothly since the gaussian kernel function is continuous. $< yy^{'}, yy^{'} >_F$ is a constant which equals N which is the number of observations in data set. So we have the denominator of the above equation almost invariable. We suppose that there is only one point (x_{i_0}) of which the label is changed, from class A to class B par example. The numerator of *Alignment*, as shown in Eq. 4.5, will be different. The difference of the numerator is $2\Sigma_{j\in B}K(x_{i_0}, x_j) - 2\Sigma_{j\in A}K(x_{i_0}, x_j)$. So we have difference of *Alignment*

$$\Delta A = \frac{2\Sigma_{j\in B}K(x_{i_0}, x_j) - 2\Sigma_{j\in A}K(x_{i_0}, x_j)}{\sqrt{< K, K >_F < yy^{'}, yy^{'} >_F}} \tag{4.9}$$

Looking at Fig. 4.4, observations in cluster B have changed their labels, moving

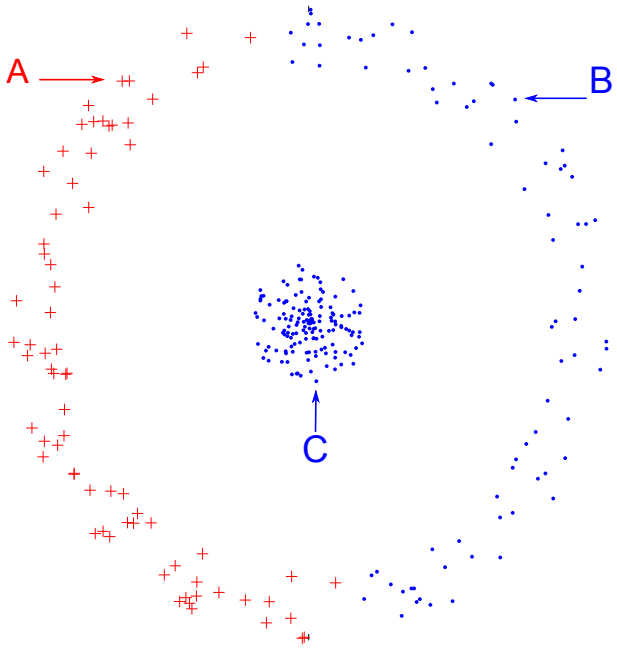


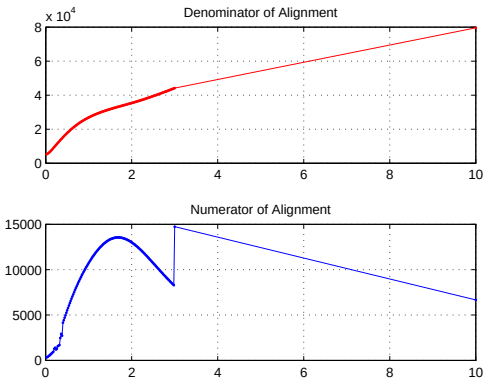
FIGURE 4.4: 'Jump analysis'

from cluster A to cluster B. According to Eq. A.10 and Eq. 4.5, we have

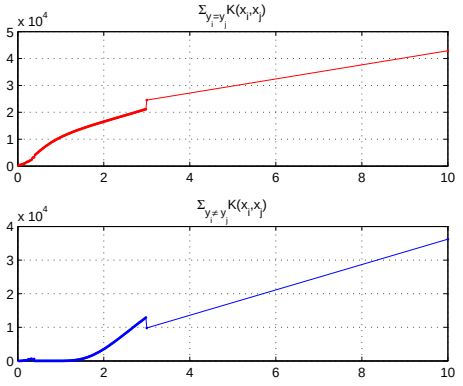
$$\Delta A = \frac{2\Sigma_{i,j\in A,B}K(x_i,x_j) - 2\Sigma_{i,j\in B,C}K(x_i,x_j)}{\sqrt{< K,K >_F < yy', yy' >_F}} \tag{4.10}$$

4.3.3.2 Experimental analysis

Experimental results are shown in Fig. 4.5. We consider the same case as in Fig. 4.3. The simulation results correlate well with the theoretical analysis. For one value of σ, there is a sudden change in the numerator of *Alignment* which leads to a 'Jump' of *Alignment*. The denominator keeps continuous, as analysed previously. We decompose the numerator into two parts: $\Sigma_{y_i=y_j}K(x_i,x_j)$ contains pairwise similarity information of points from the same cluster and $\Sigma_{y_i\neq y_j}K(x_i,x_j)$ contains pairwise similarity information of points from different clusters. From case of Fig. A.5b to case of Fig. A.5c, there is a increase of within class similarity and a decrease of between class similarity. This is reasonable in this case. As shown in Fig. 4.5, in the original space, moving cluster B from A to C leads a higher within class distance and smaller between class distance, which is expected in clustering problem. However the result is not satisfactory.



(A) Numerator and denominator of Alignment



(B) Numerator positive part and negative part

FIGURE 4.5: Experimental analysis of 'Jump'.

Conclusion Both theoretical and experimental analyses indicate that *Alignment* is not a good indicator to measure the agreement of a kernel function and a learning task. It is more sensitive to the label vector y than to the kernel Gram Matrix K, so it can not be used to determinate accurately the kernel parameter. This makes the *Alignment* very dependent to the clustering algorithm.

4.4 Centered Alignment

Cortes et al. [13] has figured out that the uncentered *Alignment* of Kandola et al. [46] doesn't correlate well with performance. We have also observed that in our experiments. The definition of *centered alignment* is similar to this of [46] however with centering and normalization techniques added. The difference in these two definitions seem tiny however the performances have significant differences. Cortes et al. [13] gave both theoretical and empirical validation of the importance of centering. Another key notion of normalization having been used by other investigators like Gretton et al. [30]. Kim et al. [49] guarantees the efficiency of *centered alignment*.

A high *centered alignment* predicts good kernels aligned for a given task. Algorithms based on centered alignment have been proposed ([13]) and have outperformed many other previous algorithms both in classification ([52]) and regression ([12]). In this part, we use *centred alignment* as a criterion for kernel selection in

KHAC. The same experiments as those presented for studying *Alignment* will be performed to test the effectiveness of this criterion.

To introduce the notion of *centered alignment*, we firstly give the definition of 'Centered Kernel Functions'.

4.4.1 Centered Kernel Functions

Let D be the distribution where we draw training and test points. Centering a feature mapping $\Phi : X \to \mathcal{F}$ is subtracting from it its expectation

$$\Phi_c = \Phi - E_x[\Phi] \tag{4.11}$$

where $E_x[\Phi]$ denotes the expected value of Φ when x is drawn from distribution D. Centering a positive definite symmetric (PDS) kernel function $K : X \times X \to R$ means centering any feature mapping Φ associated to K. Thus, for all x,$y \in X$, the centered kernel K_c associated to K is given as:

$$\begin{aligned} K_c(x,y) &= (\Phi(x) - E_x[\Phi])^T(\Phi(y) - E_y[\Phi]) \\ &= K(x,y) - E_x[K(x,y)] - E_y[K(x,y)] + E_{x,y}[K(x,y)] \end{aligned} \tag{4.12}$$

Note also that $E_{x,y}[K_c(x,y)] = 0$, which means that centering the feature mapping equalscentering kernel function.

4.4.2 Centered Kernel Matrix

Suppose a finite sample $X = (x_1, ...x_n)$ which is drawn from the distribution d. We center the vector $\Phi(x_i)$ in the feature space in the way

$$\Phi_c(x_i) = \Phi(x_i) - \overline{\Phi} \tag{4.13}$$

where $\overline{\Phi} = \frac{1}{n}\Sigma_{i=1}^{n}\Phi(x_i)$. According to Eq. 4.12, the centered kernel matrix K_c for all $i, j \in [1, n]$ is given as:

$$[K_c]_{ij} = K_{ij} - \frac{1}{n}\Sigma_{i=1}^{n}K_{ij} - \frac{1}{n}\Sigma_{j=1}^{n}K_{ij} + \frac{1}{n^2}\Sigma_{i,j=1}^{n}K_{ij} \tag{4.14}$$

We have the following properties of centered kernel matrices.

Lemma 1 For any kernel matrix K, the centered kernel matrix K_c can be expressed as follows:

$$K_c = (\mathbf{I} - \frac{1}{n}\mathbf{1}_n)K(\mathbf{I} - \frac{1}{n}\mathbf{1}_n) \tag{4.15}$$

I_n is a $n \times n$ all-one matrix where all the elements equal one. **Lemma 2** For any two kernel matrices K and K',

$$\langle K_c, K'_c\rangle_F = \langle K, K'_c\rangle_F = \langle K_c, K'\rangle_F \tag{4.16}$$

The proof of these two lemmas is given by [13].

4.4.3 Centered alignment

The definition of centered kernel alignment of two kernel matrices is similar to the uncentered *Alignment* (Eq. A.9) originally proposed by [46]

$$A_c = \frac{< K_c, K'_c >_F}{\sqrt{< K_c, K_c >_F}\sqrt{< K'_c, K'_c >_F}} \tag{4.17}$$

Like *Alignment*, *centered alignment* can also measure the agreement of a kernel and task data set. We consider $K'_c = Y_c$ where Y_c is the centered matrix of $Y = yy'$. The empirical *centered alignment* of a kernel and learning data is given:

$$A_c = \frac{< K_c, (yy')_c >_F}{\sqrt{< K_c, K_c >_F}\sqrt{< (yy')_c, (yy')_c >_F}} \tag{4.18}$$

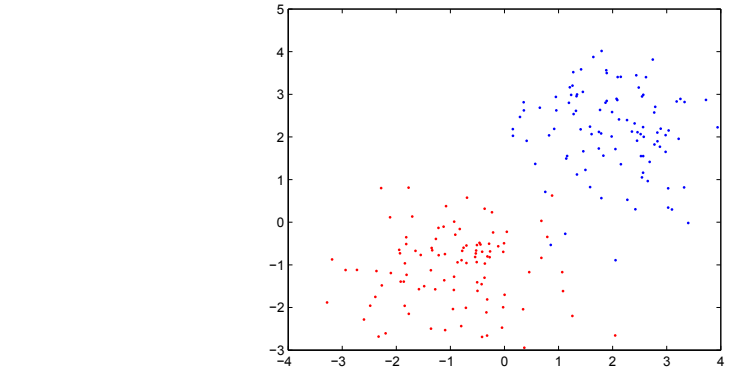
4.5 Simulations of kernel parameter selection

We complete the criterion list for kernel parameter selection, adding the *centered alignment* into the list and the same experiments as performed in part A.4.3 are done. Here we give the criterion list:

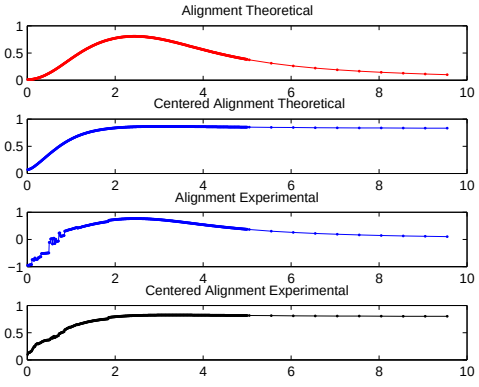
- Alignment (theoretical)
- Centered Alignment (theoretical)
- Alignment (experimental)
- Centered Alignment (experimental)
- Inter cluster distance
- Similitude

Simulation results are given in Fig. 4.6,A.5,4.8.

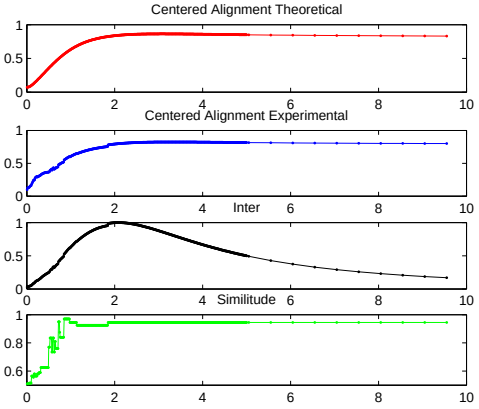
We firstly compare *Alignment* and *centered alignment* with both real y and $y(\sigma)$ estimated by algorithm (Fig.(B) in each case). Looking at the result, the curve of centered alignment is more smooth than that of *Alignment*. There is no 'Jump' for *centered alignment* which makes the kernel parameter selection more feasible.



(A)

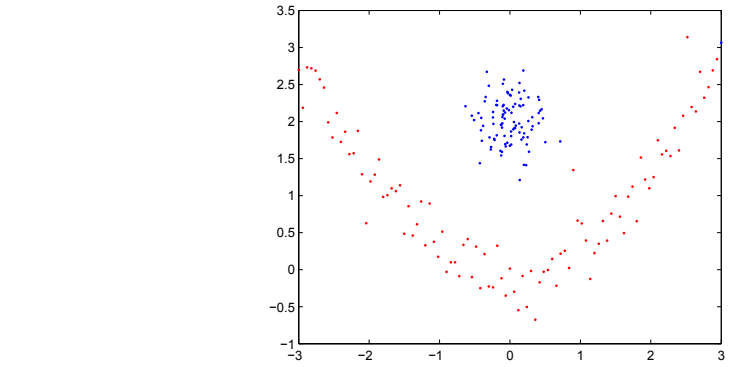


(B)

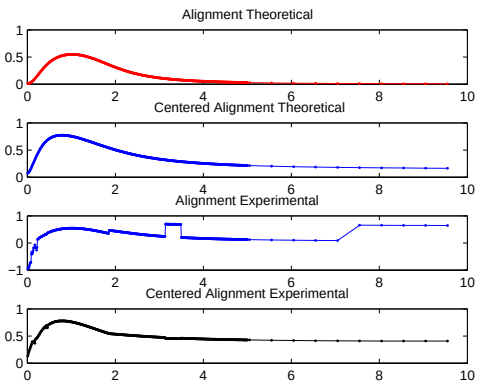


(C)

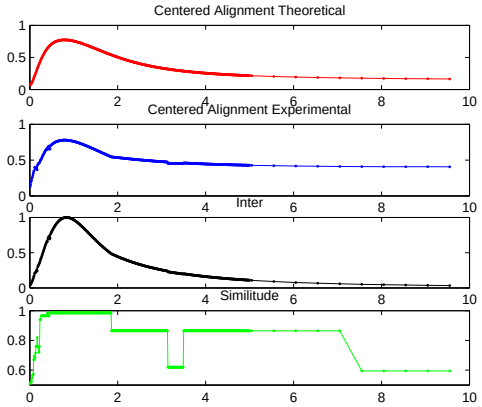
FIGURE 4.6: a.Data set b.Evaluation of Alignment and centered alignment c.Evaluation of different criteria



(A)

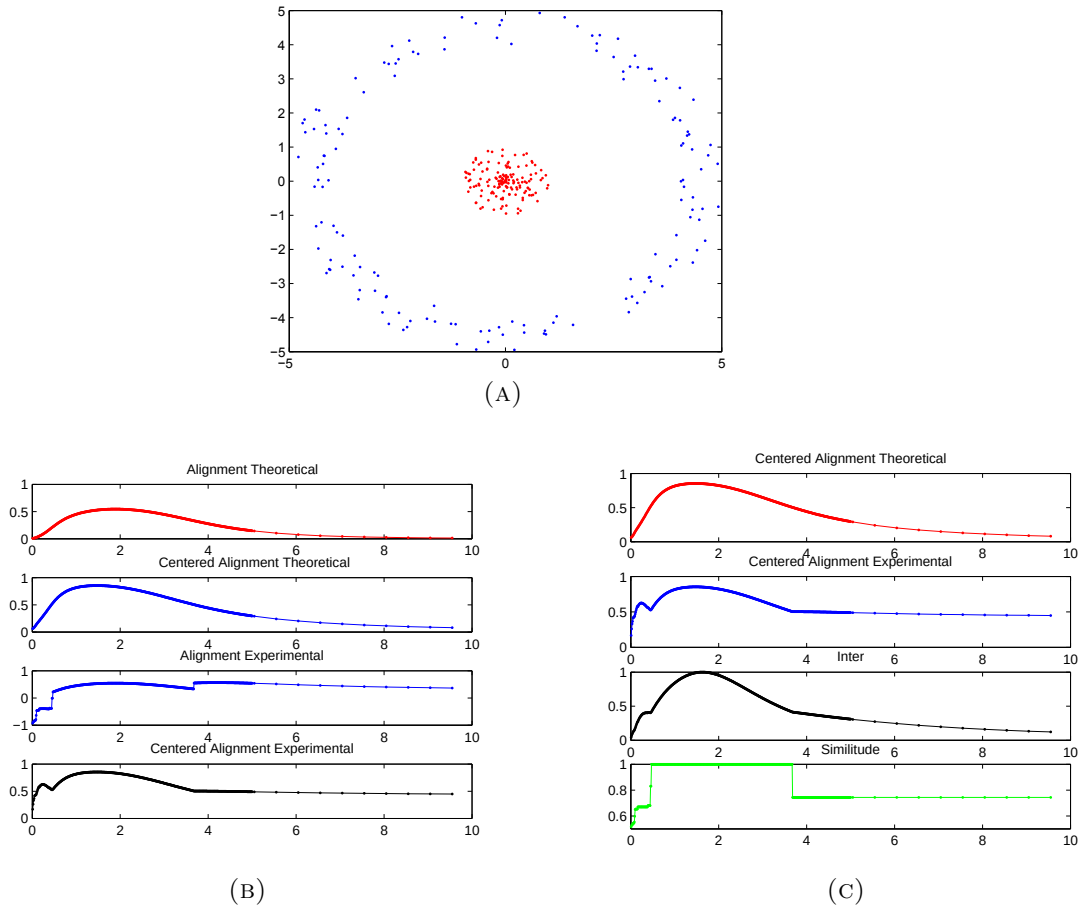


(B)



(C)

FIGURE 4.7: a.Data set b.Evaluation of Alignment and centered alignment c.Evaluation of different criteria



(A) (B) (C)

FIGURE 4.8: a.Data set b.Evaluation of Alignment and centered alignment c.Evaluation of different criteria

So we abandon *Alignment* for kernel selection and we keep *centered alignment* and *inter cluster distance* as criteria. Simulation results are given in Fig.(C) in each case. It is not surprising that *Inter cluster distance* correlates well with the performance and sometimes even better than *centered alignment*. We have demonstrated in chapter 3 that there is a close relationship between *Inter cluster distance* and Ward criterion. $y(\sigma)$ here is the output of KHAC with Ward criterion where the dendrogram is constructed in the way that in each iteration the change of inter cluster distance is maximum. So even we vary σ, there is no sudden change in inter cluster distance and so there is no 'Jump'. And also, there is a close relationship between the numerator of *Alignment* and the Fisher criterion which concerns between cluster distance and within cluster distance.

Conclusion Both *centered alignment* and *inter cluster distance* could give satisfactory performances. So in the following work, our emphases focus on *Centered Alignment* and *inter cluster distance*.

4.6 Study of the stability of criteria w.r.t the proportion of classes

In practice, most of the artificial data sets and real data sets are unbalanced. So it is necessary to test the stability of *Centered Alignment* and *Inter cluster distance* when the proportion of the two classes changes. We consider a simple case as shown in Fig. 4.9 where a fraction $\alpha \in [0, 1]$ of all points are centered at $(-1, 0)$ with a variance of $\rho^2 I_2$ and labeled with -1, and the remaining points centered at $(1, 0)$ with a variance of $\rho^2 I_2$ and labeled with $+1$. Several data distribution examples are given in Fig. A.6 with different α and ρ.

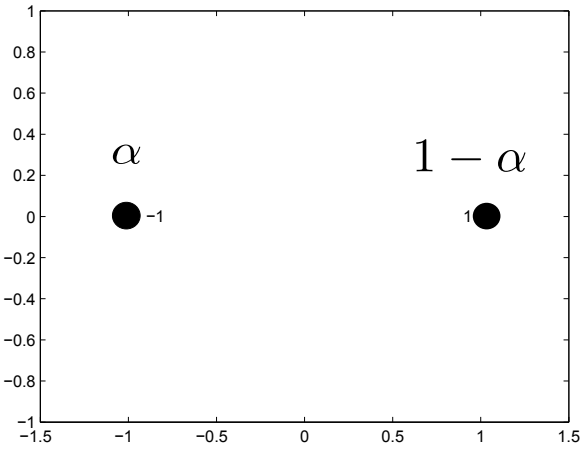


FIGURE 4.9: Data distribution

Criteria which we will test in the following are given as bellow:

- Alignment
- Centered Alignment
- Normalized inter cluster distance

Firstly we consider to test the stability with a simple example of kernel function: $K = x \cdot x' + 1$ and we vary α. Several value of $\rho \in [0, 1]$ have been taken and the results are given in Fig. 4.11:

When $\rho = 0$, seeing the distribution in Fig. A.6, for any value of $\alpha \in [0, 1]$, the problem is separable and one would expect that the criteria to be close to 1. Looking at the first figure in Fig. 4.11, *centered alignment* correlates very well with the

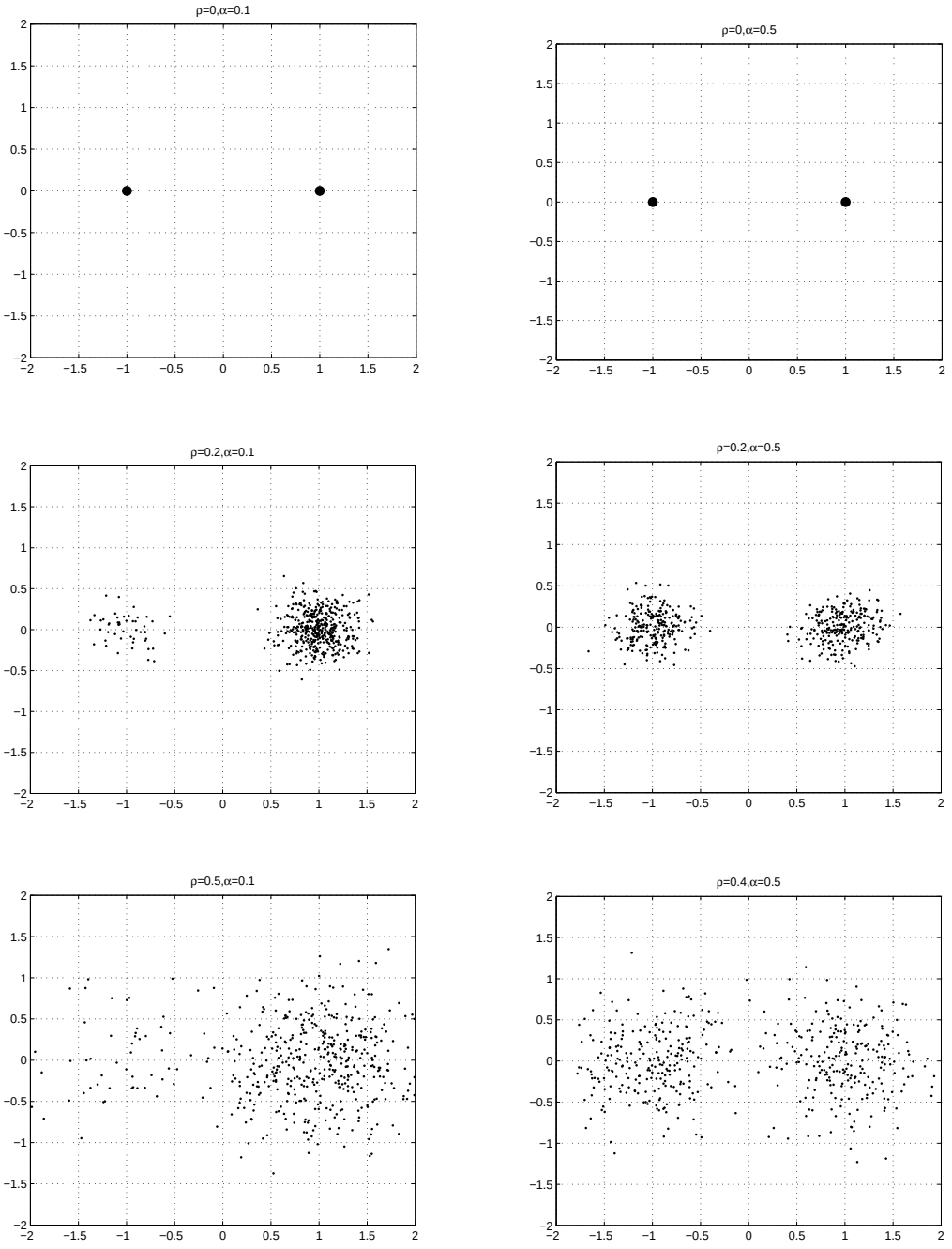


FIGURE 4.10: Distributions with different α and ρ

expectation and *inter cluster distance* gives good performance when we have identical number of points in each class. Results here show again that *Alignment* is not a good predictor for kernel learning.

As in our research, in most of the cases we consider to use a gaussian kernel, so criteria have been tested under the Gaussian kernel condition. Results are shown

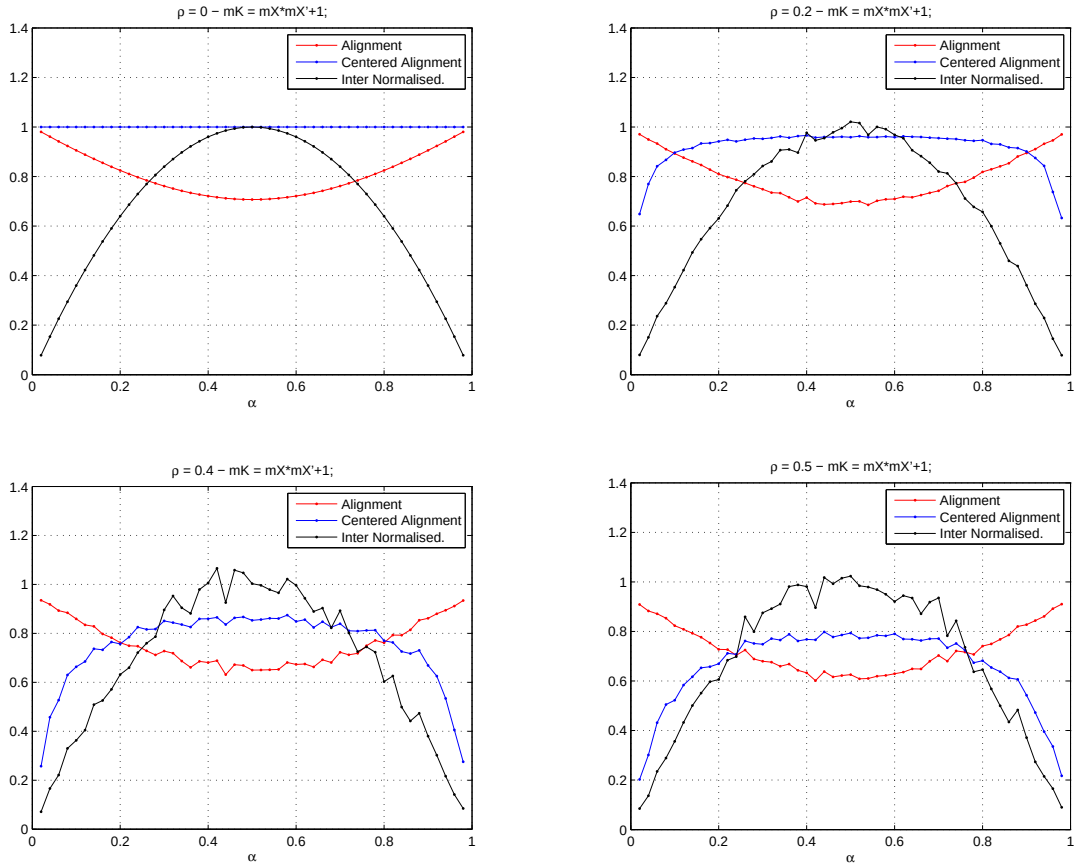


FIGURE 4.11: Criteria stability evaluation using the kernel $K = X * X' + 1$

in Fig. A.8 and the same conclusions are made as in the previous case.

4.7 Centered alignment and inter cluster distance

Both *centered alignment* and *inter cluster distance* have been demonstrated to correlate well with cluster performances using KHAC algorithms. We then try to estimate the efficiency of these two criteria by varying the proportion rate α and the kernel parameter σ. We consider $y(\sigma)$ in criteria calculations which means that these two criteria measure a degree of agreement between a kernel function and a learning task.

Fig. 4.13 shows experiments results in the case of a two-circle data distribution.

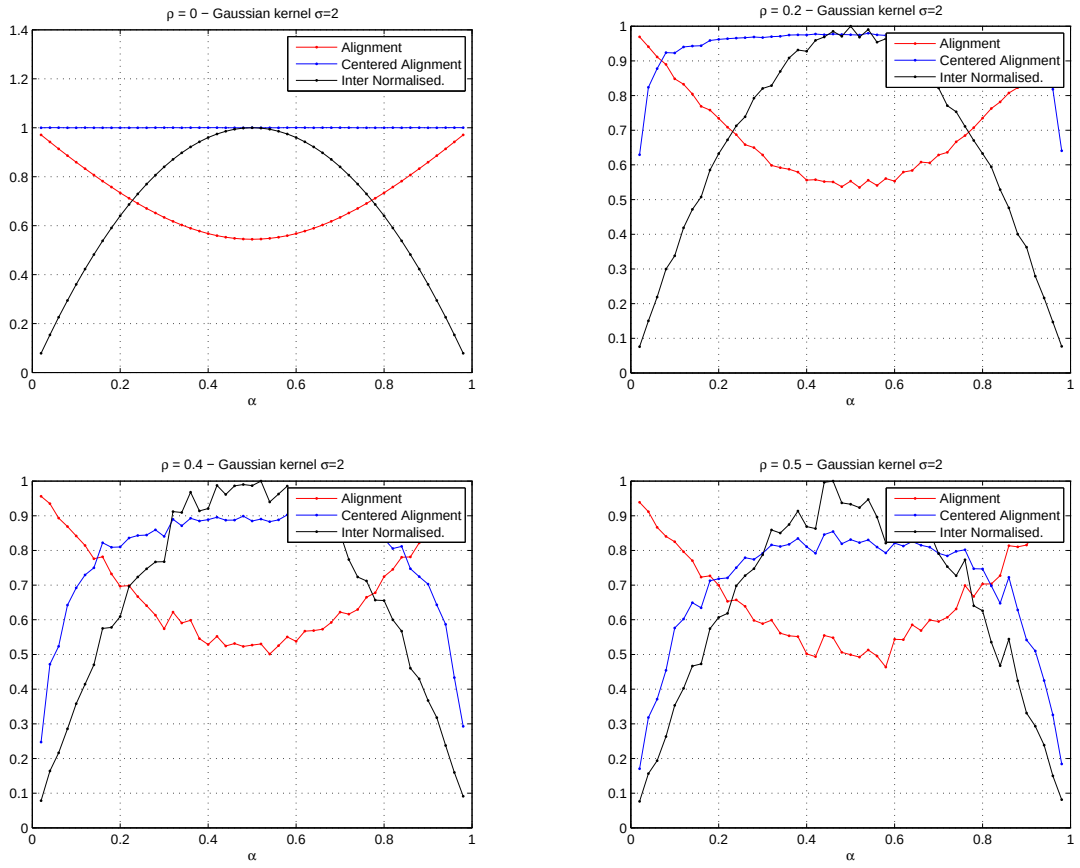


FIGURE 4.12: Criteria stability evaluation using the Gaussian kernel

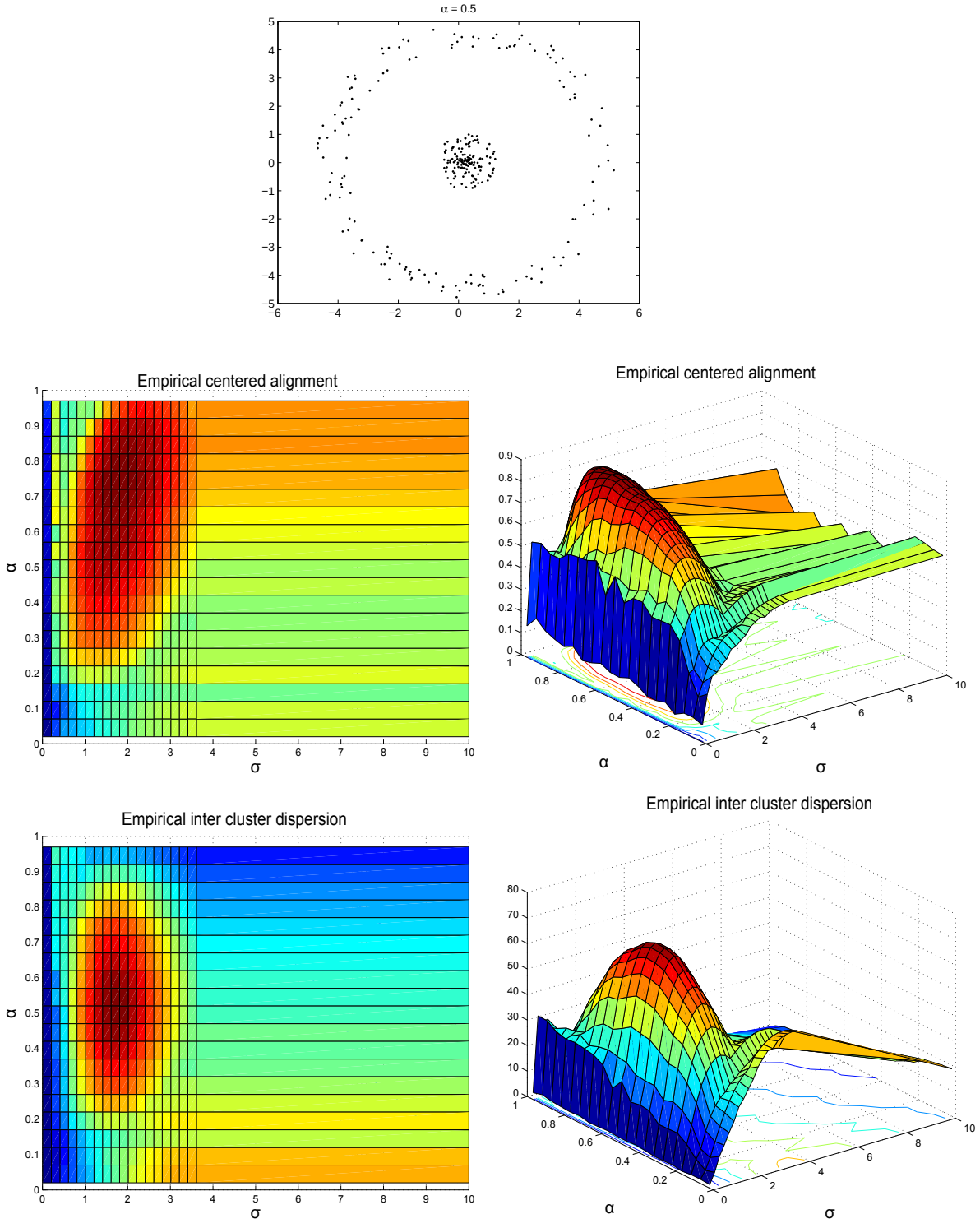


FIGURE 4.13: Empirical centered alignment and inter cluster distance evaluation by varying α and σ-data case of two circles

We run 100 times the same experiments as shown in Fig. 4.13 and results are given in Fig. 4.14

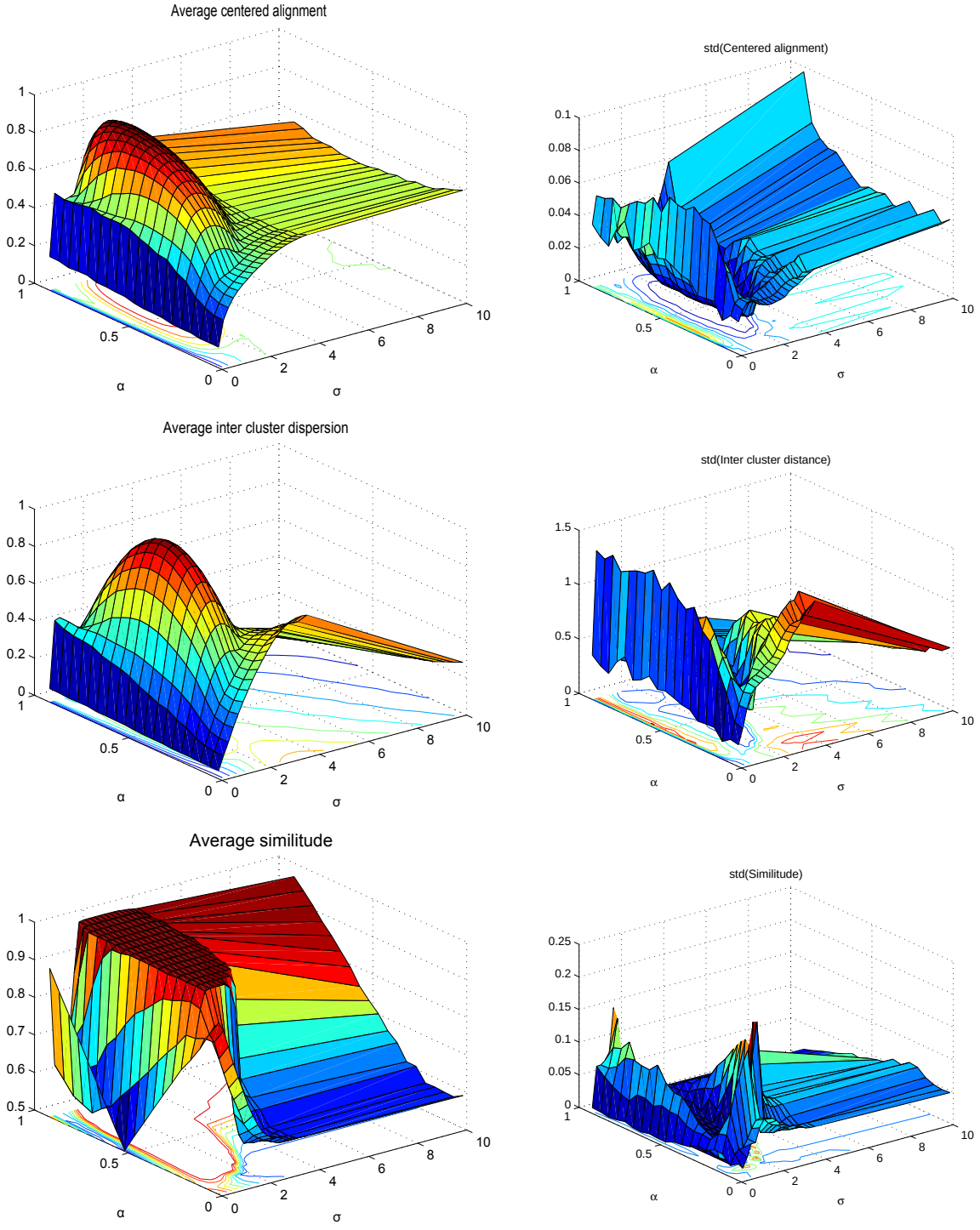


FIGURE 4.14: Average centered alignment and inter cluster distance evaluation by varying α and σ(100 times)-data case of two circles

4.8 Conclusion

Shortcomings of alignment are given in this chapter. It seems that centered alignment is the best criterion among those we considered. Inter-cluster dispersion gives also competitive performance and as we can see in Fig. 4.14, in a large interval of σ, clustering result is satisfactory. In experiments, both of them can be considered.

As there is a tight relationship between optimal kernel parameter and qualified dendrogram which represent the data structure, we also consider these two criteria in our following work to determine the number of clusters etc.

Chapter 5

Determination of the number of clusters

Contents

5.1 Introduction

Determining the number of clusters is one of the most difficult and necessary problems in cluster analysis. For some algorithms like k-means, the number of clusters needs to be provided in advance. Knowing the number of clusters leads the clustering procedure to be more easier and may help in finding better criteria for clustering validation problems. In fact the notion *number of clusters* is quite

indistinct as in clustering problems no prior information is available and finding the number of clusters is in fact a task of discovering the hidden cluster structures. Research on solving this problem goes through a difficult time and most of the methods that investigators have proposed are often very problem dependent. Until now, there is no conclusion that there exists a method which outperforms others most of the time.

In the earlier times, Milligan and Copper [61] have done a research on criteria for estimating the number of clusters, reporting the results of a simulation experiment designed to determine the validity of 30 criteria proposed in the literature. These criteria have been tested under a condition of hierarchical algorithms. They have also been considered as stopping rules in hierarchical algorithms and often can be extended in nonhierarchical procedures as well. In this research, Milligan and Copper have generated either 2,3,4 or 5 distinct nonoverlapping clusters to test the efficiency of these rules and it seems that Calinski and Harabasz index [9] and the Je(2)/Je(1) rule proposed by [19] give excellent results. Detailed simulation results of these 30 rules are given using a clear tabular summary presentation in [61].

After that, several improved criteria emerged like Hartigan's rule [35], Krzanowski and Lai's index [50], the silhouette statistic suggested by Kaufman and Rousseeuw [47] and Calinski and Harabasz's index [9], which has demonstrated better performance under most of the situations considered in Milligan and Copper's study. More recent methods for determining the number of clusters include an model-based approach using approximate Bayes factors in a Gaussian mixture distribution proposed by [24] and a jump method by Sugar and James [78].

Unfortunately, most of these methods are somewhat ad hoc or model-based and hence sometimes require parametric assumptions which lead to a lack of generality. However, the Gap Statistics proposed by Tibshirani et al. [79] is designed to be applicable to virtually any clustering method like the commonly used k-means and hierarchical procedures. The principle of this method is to compare the within cluster dispersion obtained by the considered clustering algorithm with that we would obtain under a null hypothesis. Among all the methods that have emerged, the Gap Statistic is probably one of the most promising approach.

5.2 Gap Statistics

Consider a data set $x_{ij}, i = 1, 2, ..., n, j = 1, 2, ..., p$, consisting of p features, all of them being measured on n independent observations. $d_{ii'}$ denotes some dissimilarity between observations i and i'. The most common used dissimilarity measure is the squared Euclidean distance ($\sum_j (x_{ij} - x_{i'j})^2$). Suppose that the data set is composed of k clusters and that C_r denotes the indices of observations in cluster r and $n_r = |C_r|$.

According to Tibshirani, and using his notations, we define:

$$D_r = \sum_{i,i' \in C_r} d_{ii'} \tag{5.1}$$

as the sum of all the distances between any two observations in cluster r. So

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \tag{5.2}$$

is the within-class dispersion. W_k monotonically decreases as the number of clusters k increases but, according to Tibshirani [79], from some k onwards, the rate of decrease is dramatically reduced. It has been shown that the location of such an *elbow* indicates the appropriate number of clusters.

The main idea of Gap Statistics is to compare the graph of $log(W_k)$ with its expectation we could observe under a single cluster hypothesis (H_0). Unlike the case of task data set (H_1), when the number of cluster k increases, W_k decreases monotonically but smoothly, there is no sharp change in W_k under null hypothesis (see Fig. 5.1 upper curve).

We define the Gap Statistic as:

$$Gap(k) = E_n\{log(W_k/H_0)\} - log(W_k) \tag{5.3}$$

Here $E_n\{log(W_k/H_0)\}$ denotes the expectation of $log(W_k)$ under the null reference distribution H_0. The estimated number of clusters $\hat{k}$ falls at the point where $Gap_n(k)$ is maximum. Expectation is estimated by averaging the results obtained from different realizations of the data set under the null reference distribution.

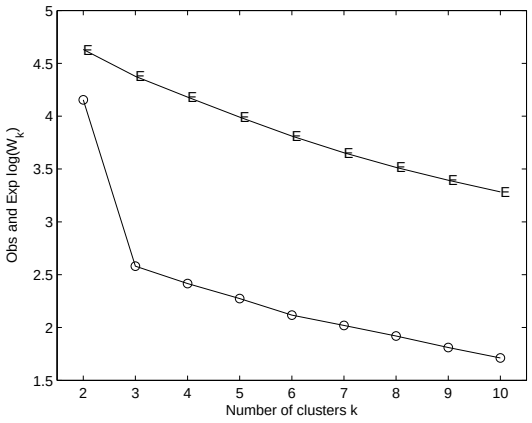


FIGURE 5.1: Graphs of $E_n\{log(W_k/H_0)\}$ (upper curve) and $log(W_k)$ (lower curve).

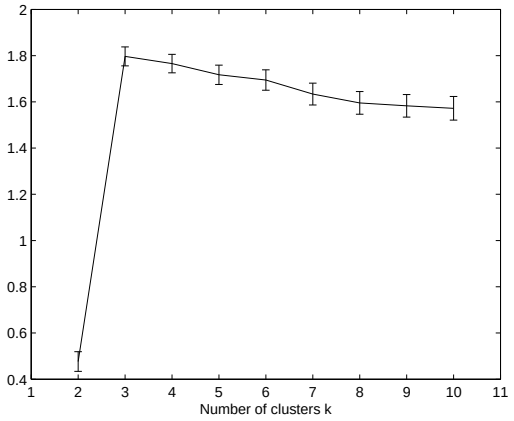


FIGURE 5.2: Gap statistic as a function of the number of clusters.

Figures 5.1 and 5.2 show an example using kernel HAC. Data are composed of three distinct bi-dimensional Gaussian clusters centred on $(0, 0)$, $(0, 1.3)$, $(1.4, -1)$ respectively, with unit covariance matrix I_2 and 100 observations per class. The functions $log(W_k)$ and the estimate of $E_n\{log(W_k/H_0)\}$ are shown in Figure 5.1. The Gap Statistic is shown in Figure 5.2. In this example, $E_n\{log(W_k/H_0)\}$ was estimated using 150 independent realizations of the null data set. We also estimated the standard deviation $sd(k)$ of $log(W_k/H_0)$. Let $s_k = \sqrt{1 + \frac{1}{B}}sd(k)$, which is represented by vertical bars in Figure 5.2, then, according to Tibshirani, the estimated number of cluster $\hat{k}$ is the smallest k such that:

$$Gap(k) \geq Gap(k + 1) - s_{k+1} \tag{5.4}$$

Fig. 5.2 shows that, for the considered data set, $\hat{k} = 3$, which is correct.

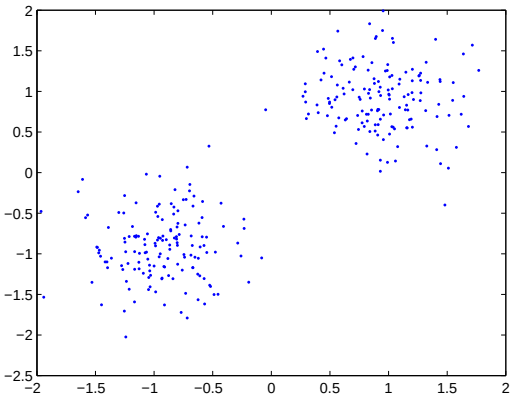
5.3 Null reference distribution

The importance of the choice of an appropriate null reference distribution of the data has been studied in [28] by Gordon. In this paper, Gordon firstly introduced some standard null models like Poisson model ([86]), Unimodal model ([66]) and Random dissimilarity matrix model ([54]). Then some data-influenced null models where the form of the null model is greatly influenced by the data characteristics have been given like random permutation of pattern matrix, convex hull and ellipsoidal models etc. U-statistic method ([55]) was used to assess clustering results under these null models. A small value of U-statistic indicates a good cluster validation. In our following work, we also noticed that the choice of a good null reference is quite important for determining the number of clusters.

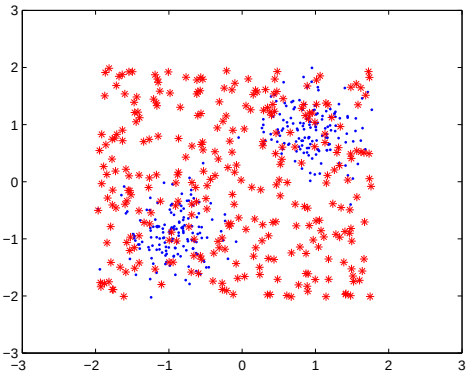
According to Tibshirani et al. [79], the assumed null model of the data set must be a single cluster model. The most common considered reference distributions are:

- an uniform distribution over the range of the observed data set.
- an uniform distribution over an area which is aligned with the principal components of the data set.

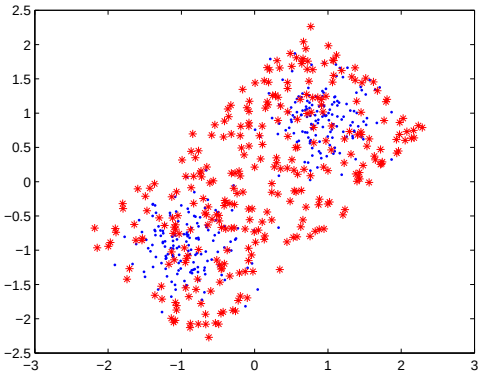
The first method has the advantage of simplicity while the second is more accurate in terms of consistency because it takes into account the shape of the data distribution. Fig. 5.3 shows an example of samples drawn from these two null distributions for a 2-class gaussian distribution case.



(A) Data set



(B) Uniform null model



(C) Uniform null model aligned with PCA

FIGURE 5.3: Two common used null reference distributions - case of two gaussian classes

5.4 New Criteria to Estimate the Number of Clusters in Kernel HAC

The main idea of Gap Statistics was to compare the within-class dispersion obtained on the data with that of an appropriate reference distribution. Inspired by this idea, we propose to extend it to other criteria which are suitable for HAC to estimate the number of clusters.

These criteria are :

- Modified Gap Statistic

  In the Gap Statistic proposed by Tibshirani, the estimated number of clusters $\hat{k}$ is chosen according to equation A.18. In this modified Gap Statistic, we define $\hat{k}$ as the number of clusters where $Gap(k)$ (equation A.17) is maximum.

As will be shown later in this paper, this allows to potentially improve the estimate, at least for standard HAC.

- Centered Alignment Gap
  A good guess for the nonlinear function $\Phi(x_i)$ should be to directly produce the expected result y_i for all observations. In this case, the *best* Gram matrix K whose general term is $K_{ij} = K(x_i, x_j)$ becomes yy' where y is the column vector (or matrix, depending on the selected code book) of all the data labels. Shawe-Taylor et al. ([72]) have proposed a function that depends on both the labels and the Gram matrix, called *alignment*, to measure the degree of agreement between a kernel function and the clustering task. As presented before, the *Alignment* is defined by:

$$Alignment = \frac{< K, yy' >_F}{\sqrt{< K, K >_F < yy', yy' >_F}} \tag{5.5}$$

  where the subscript F denotes Frobenius norm. In this paper, y is estimated after the clustering process. The Centered Alignment (CA) is defined as:

$$CA = \frac{< K_c, Y_c >_F}{\sqrt{< K_c, K_c >_F < Y_c, Y_c >_F}} \tag{5.6}$$

  Here the $Y = yy'$ and the centered matrix K_c associated to the matrix K is defined by:

$$\begin{aligned} K_c(x, y) &= \langle \Phi(x) - E_x[\Phi], \Phi(x') - E_{x'}[\Phi] \rangle \\ &= K(x, y) - E_x[K(x, y)] - E_y[K(x, y)] \\ &\quad + E_{x,y}[K(x, y)] \end{aligned}$$

  where the expectation operator is evaluated by averaging over the data. Y_c is defined by:

$$Y_c = Y - E[Y]$$

  So the Centered Alignment Gap is defined by:

$$Gap_{CA}(k) = E_n\{CA/H_0\} - CA \tag{5.7}$$

  The estimated number of clusters $\hat{k}$ is the k such that $Gap_{CA}(k)$ is maximum.

- Delta Level Gap
  In the dendrogram, we consider each level of the similarity measure $h_i, i =$

$1...n-1$ where h_1 is the highest level. Then we define $\Delta h(k) = h_i - h_{i+1}, k = 2...n$. We define the criterion Delta Level Gap:

$$Gap_{\Delta h(k)}(k) = \Delta h(k) - E_n\{\Delta h(k)/H_0\} \tag{5.8}$$

The estimated number of clusters $\hat{k}$ is the value of k where $Gap_{\Delta h(k)}(k)$ is maximum.

- Weighted Delta Level Gap
  This criterion is related to the previous one. We define the Weighted Delta Level (denoted by $\Delta h_W(k)$) by: $\Delta h_W(2) = \Delta h(2), k = 2$ and $\Delta h_W(k) = \frac{\Delta h(k)}{\sum_{i=2}^{k-1} \Delta h(i)}, k \geq 2$. We define the criterion Weighted Delta Level Gap as:

$$Gap_{\Delta h_W(k)}(k) = \Delta h_W(k) - E_n\{\Delta h_W(k)/H_0\} \tag{5.9}$$

  The estimated number of clusters $\hat{k}$ is the value of k where $Gap_{\Delta h_W(k)}(k)$ is maximum.

5.4.1 Determinate the number of clusters using different statistics

The procedure of our algorithm (seeing Fig. A.9) is given as follows:

- ***Step1*** Apply KHAC on the task data set to obtain the above criteria.
- ***Step2*** Apply KHAC on null reference data set and repeat N times(500 par example) to obtain expectations of criteria in *step1*.
- ***Step3*** Compare the criteria obtain in *step1* with their expectations obtained in *step2* and finally calculate the indexes to determinate the number of clusters using different strategies as described above.

5.4.2 Multiclass codebook

To evaluate the centered kernel alignment, labels must be known. Notice that it's no longer only a 2-class problem anymore, it may also be multiclass problem. Several well-known labelbooks for multiclass problem are given in Table. 5.1 [38].

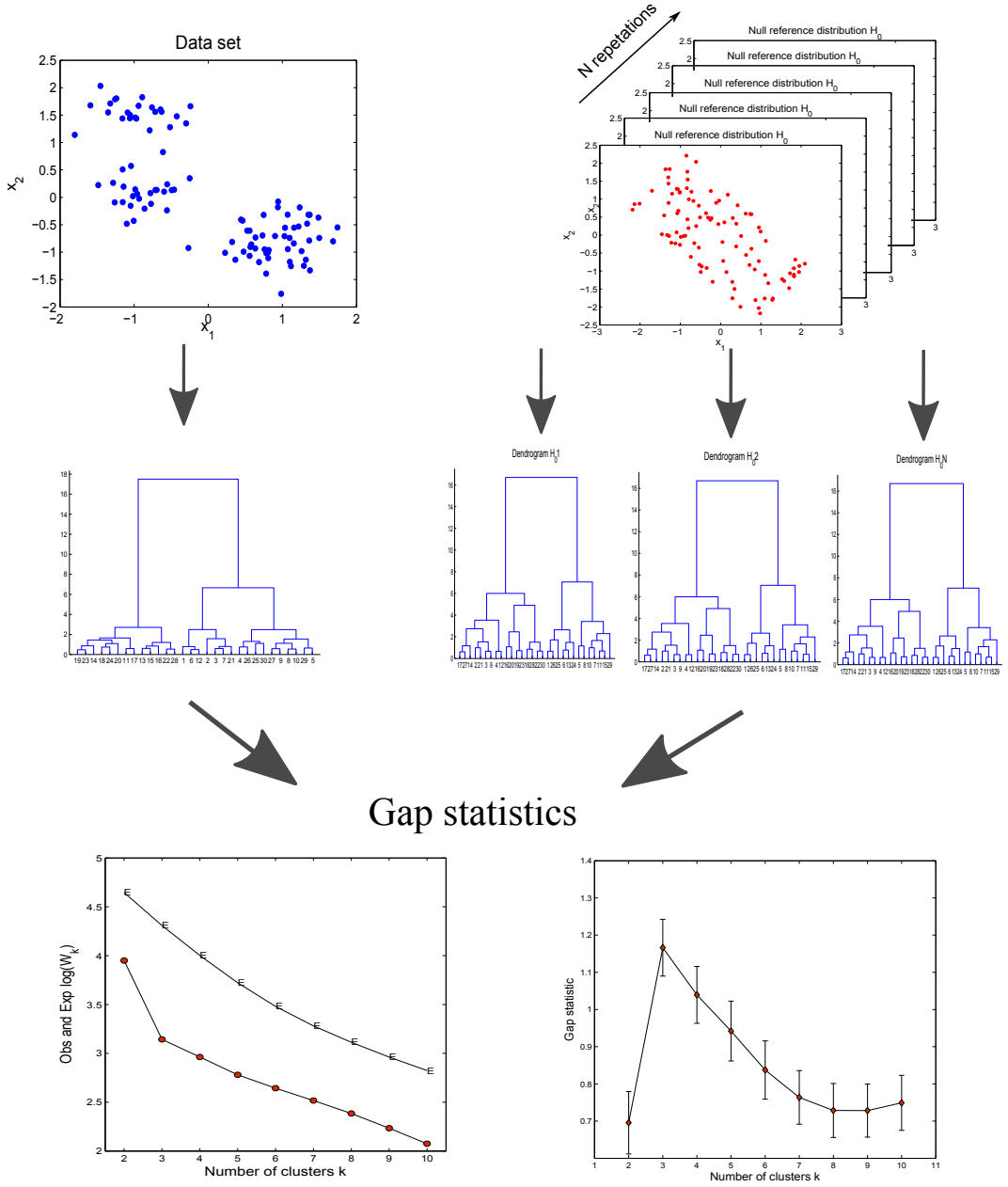


FIGURE 5.4: Procedure to determinate the number of clusters.

In our experiments, coding Y should performed in such a way that the centered kernel alignment is invariant to the (arbitrary) cluster number. In this part, the vector codebook is, for an observation x_i belonging to cluster $m, m = 1, \ldots, k$:

$$\begin{cases} y_{ij} = 1, \text{ if } j = m, \\ y_{ij} = -1, \text{ if } j \neq m. \end{cases} \tag{5.10}$$

which is in fact the (± 1)label codebook.

Labelbook	$[y^{(k)}]_i = [y_i]_k$
(± 1)label	$+1$, if $x_i belongstoclass k$, -1, otherwise.
Standard basis	1, if $x_i belongstoclass k$, 0, otherwise.
Consistency-based	1, if $x_i belongstoclass k$, $\frac{-1}{l-1}$, otherwise.

TABLE 5.1: Well-known labelbooks for multiclass problem.(here l is the number of clusters)

5.4.3 SIMULATIONS

We have generated 6 different data sets (see Fig. 5.5) to compare the proposed criteria with that from Tibshirani:

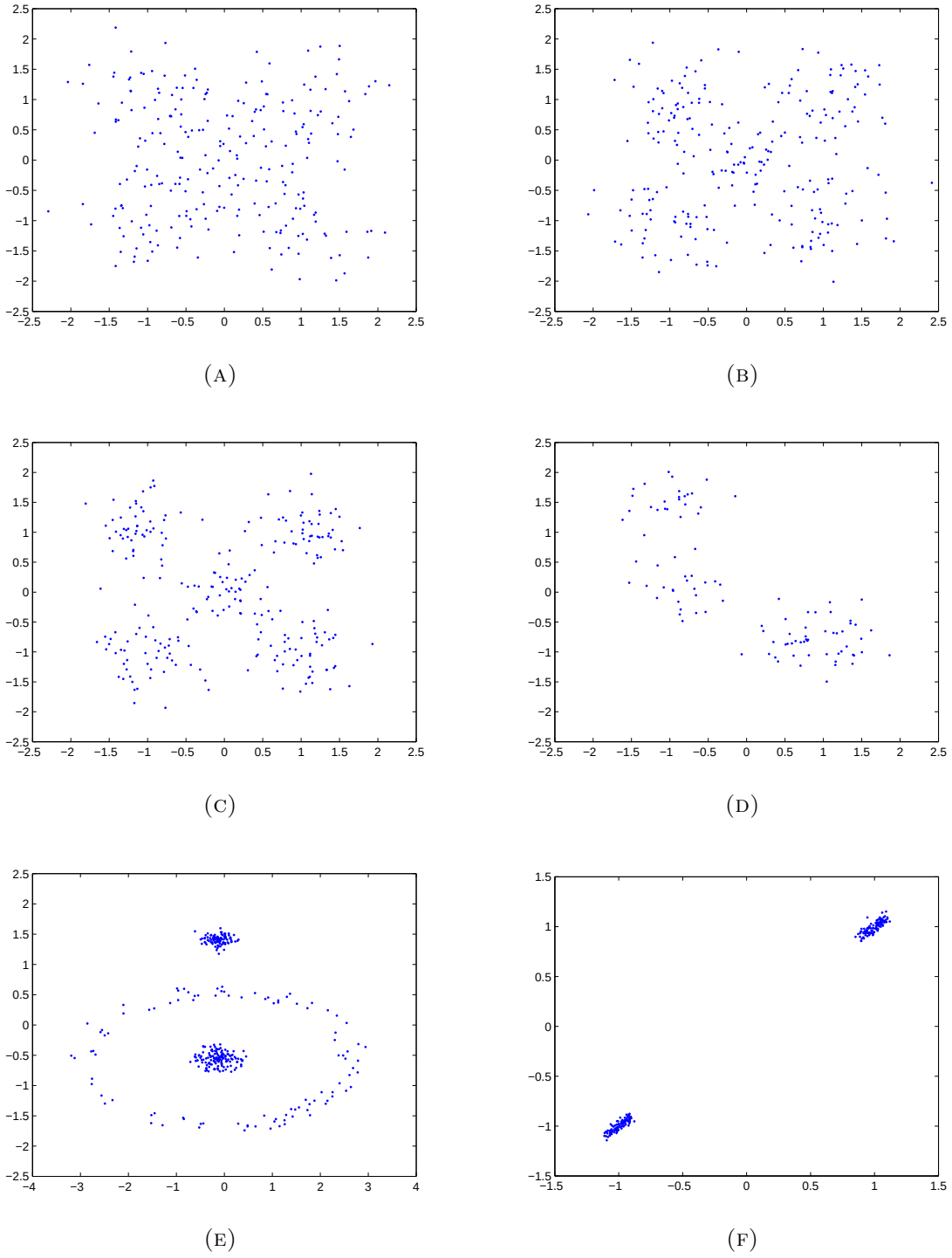
1. Five clusters in two dimensions [(1)]
   The clusters consist of gaussian bidimensional distributions $N(0, 1.5^2)$ centered at (0,0), (-3,3), (3,-3), (3,3) and (-3,-3). Each cluster is composed of 50 observations. Clusters strongly overlap. The kernel parameter is $\sigma = 0.90$.

2. Five clusters in two dimensions [(2)]
   The data are generated in the same way as in the previous case but the variance of the clusters is now 1.25^2). Clusters slightly overlap. $\sigma = 0.85$.

3. Five clusters in two dimensions [(3)]
   Same case but with variance equal to 1.0^2. There is no overlap. $\sigma = 0.80$.

4. Three clusters in two dimensions
   This is a data set used by Tibshirani [79]. The clusters are unit variance gaussian bidimensional distributions with $25, 25, 50$ observations, centered at $(0, 0)$, $(0, 5)$ and $(5, -3)$, respectively. $\sigma = 0.80$.

5. Two nested circles and one outside isolated disk in two dimensions
   These three circles are centered at $(0, 0)$, $(0, 0)$ and $(0, 8)$ with $150, 100, 100$ observations. The respective radii are uniformly distributed over $[0, 1]$, $[4, 5]$ and $[0, 1]$. $\sigma = 0.55$.



(A) (B) (C) (D) (E) (F)

FIGURE 5.5: 6 data sets for simulations of determination of number of clusters

6. Two elongated clusters in three dimensions

   This is also a data set used by Tibshirani [79] to show the interest of performing PCA to define the null distribution. Clusters are aligned with the first diagonal of a cube. We have $x_1 = x_2 = x_3 = x$ with x composed of

100 equally spaced values between -0.5 and 0.5. To each component of x a Gaussian perturbation with standard deviation 0.1 is added. The second cluster is generated similarly, except for a constant value of 10 which is added to each component. The result is two elongated clusters, stretching out along the main diagonal of a three dimensional cube. $\sigma = 1.00$.

Simulation results with kernel HAC are shown in Table 5.2. 50 realizations were generated for each case and we used principal component analysis to define the distributions used as the null reference samples. A special scenario using a uniform distribution over the initial area covered by the data (without PCA) is provided in case 7. For every simulation, expectation appearing in equation A.17 was estimated over 100 independent realizations of the null distribution.

The kernel parameter σ has been selected as the value which maximizes the centered kernel alignment defined by equation 5.6.

Experiment results clearly indicate that one of the proposed criteria Delta Level Gap outperforms other criteria (including the Gap Statistic by Tibshirani) in almost all cases.

Tibshirani [79] mainly focused on well-separated clusters. Our simulations also show that the Gap Statistic estimation is not good at identifying the number of clusters when they highly overlap. For data sets $1, 2$ and 3: the lower the overlap, the better the performance. Looking at the example of data set 1, all criteria do not give the expected results. The number of clusters estimated by Delta Level Gap mostly give 4 and 5 clusters, which is better. However, all the criteria suffer from overlap, which is not surprising.

Another important conclusion is the importance of the choice of an appropriate null reference distribution. Looking at scenarios 6 and 7, results vary a lot between the uniform distribution and the uniform distribution aligned with principal components. This gives some room to improve our framework by choosing a more appropriate null reference distribution.

We have also done simulations using standard HAC on some of these data sets. Comparing the results obtained show that kernel HAC generally improves the performance. We can observe that Centered Alignment Gap, Gap Statistic and Weighted Delta Level Gap do not perform well on examples $1, 2$ and 3 for standard

TABLE 5.2: Simulation results using Kernel HAC. Each number represents the number of times each criterion gives the number of clusters indicated in the corresponding column, out of the 50 realizations. The column corresponding to the right number of clusters is indicated in boldface. Numbers between parentheses indicate the results obtained using standard HAC, when of some interest. NF stands for Not Found.

Number of clusters	2	3	4	5	6	7	8	9	10	**NF**
1. Five clusters in two dimensions[(1)]										
Gap Statistic	0 (48)	0 (2)	4 (0)	**28 (0)**	15 (0)	3 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Modified Gap Statistic	0 (2)	0 (0)	0 (10)	**9 (36)**	9 (1)	5 (0)	5 (0)	3 (0)	19 (0)	0 (0)
Delta Level Gap	0 (0)	0 (0)	18 (9)	**30 (38)**	1 (1)	1 (1)	0 (0)	0 (0)	0 (1)	0 (0)
Weighted Delta Level Gap	0 (47)	7 (0)	20 (1)	**22 (1)**	1 (1)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Centered Alignment Gap	6 (1)	4 (0)	11 (1)	**23 (6)**	6 (10)	0 (9)	0 (8)	0 (9)	0 (6)	0 (0)
2. Five clusters in two dimensions [(2)]										
Gap Statistic	0 (48)	0 (0)	0 (0)	**43 (2)**	6 (0)	1 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Modified Gap Statistic	0 (0)	0 (0)	0 (1)	**19 (45)**	10 (3)	6 (1)	5 (0)	2 (0)	8 (0)	0 (0)
Delta Level Gap	0 (0)	0 (0)	2 (1)	**48 (48)**	0 (1)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Weighted Delta Level Gap	0 (45)	3 (0)	12 (0)	**35 (5)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Centered Alignment Gap	0 (0)	1 (0)	2 (0)	**40 (10)**	5 (15)	2 (9)	0 (7)	0 (4)	0 (5)	0 (0)
3. Five clusters in two dimensions [(3)]										
Gap Statistic	0 (39)	0 (0)	0 (0)	**48 (11)**	2 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Modified Gap Statistic	0 (0)	0 (0)	0 (0)	**26 (47)**	12 (3)	2 (0)	0 (0)	1 (0)	9 (0)	0 (0)
Delta Level Gap	0 (0)	0 (0)	0 (0)	**50 (50)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Weighted Delta Level Gap	0 (30)	1 (0)	3 (0)	**46 (20)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Centered Alignment Gap	0 (0)	0 (0)	0 (0)	**50 (10)**	0 (18)	0 (12)	0 (6)	0 (3)	0 (1)	0 (0)
4. Three clusters in two dimensions										
Gap Statistic	0	**38**	12	0	0	0	0	0	0	0
Modified Gap Statistic	0	**14**	18	3	0	5	4	3	3	0
Delta Level Gap	5	**45**	0	0	0	0	0	0	0	0
Weighted Delta Level Gap	0	**50**	0	0	0	0	0	0	0	0
Centered Alignment Gap	35	**15**	0	0	0	0	0	0	0	0
5. Two nested circles and one outside isolated disk in two dimensions										
Gap Statistic	0 (0)	**0 (0)**	0 (38)	0 (6)	0 (1)	0 (3)	0 (1)	1 (0)	0 (0)	49 (1)
Modified Gap Statistic	0 (0)	**0 (0)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (1)	0 (0)	50 (49)	0 (0)
Delta Level Gap	0 (0)	**50 (3)**	0 (42)	0 (2)	0 (0)	0 (1)	0 (2)	0 (0)	0 (0)	0 (0)
Weighted Delta Level Gap	0 (48)	**7 (0)**	0 (2)	0 (0)	2 (0)	8 (0)	6 (0)	13 (0)	14 (0)	0 (0)
Centered Alignment Gap	50 (12)	**0 (23)**	0 (0)	0 (0)	0 (1)	0 (1)	0 (2)	0 (3)	0 (8)	0 (0)
6. Two elongated clusters in three dimensions										
Gap Statistic	**50**	0	0	0	0	0	0	0	0	0
Modified Gap Statistic	**45**	0	5	0	0	0	0	0	0	0
Delta Level Gap	**50**	0	0	0	0	0	0	0	0	0
Weighted Delta Level Gap	**0**	0	0	0	0	0	4	10	36	0
Centered Alignment Gap	**50**	0	0	0	0	0	0	0	0	0
7. Two elongated clusters in three dimensions (without PCA to define the null reference)										
Gap Statistic	**0**	0	1	0	18	13	15	3	0	0
Modified Gap Statistic	**0**	0	0	0	1	2	6	7	34	0
Delta Level Gap	**50**	0	0	0	0	0	0	0	0	0
Weighted Delta Level Gap	**0**	0	0	0	0	0	0	3	47	0
Centered Alignment Gap	**50**	0	0	0	0	0	0	0	0	0

HAC. Evolution of these statistics as functions of the number of clusters, that can be seen in Figures 5.6, 5.7 and 5.8, explain these results. Figure 5.6 shows the evolution of the Gap Statistic as a function of the number of clusters k for a realization of the second data set. As can be seen, the criterion initially proposed gives $k = 2$. Figure 5.7 represents the evolution of Centered Alignment Gap. The

behavior is rather erratic and explains the variations in the estimated number of clusters. This can also be observed for Weighted Delta Level Gap shown in Figure 5.8.

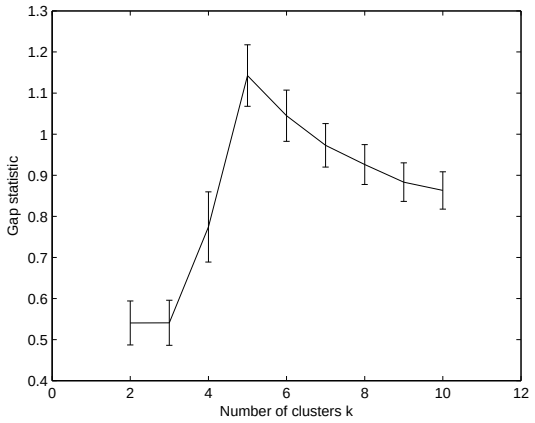


FIGURE 5.6: Number of clusters using Gap Statistic.

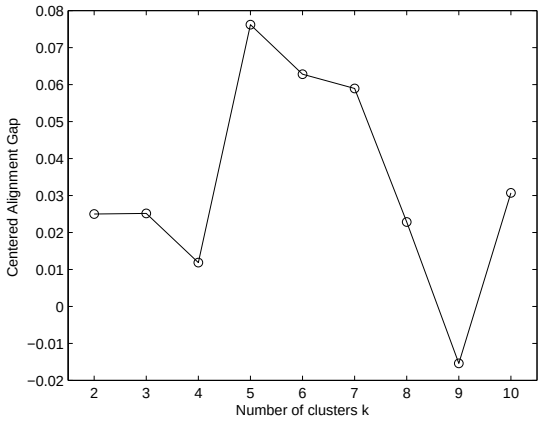


FIGURE 5.7: Number of clusters using Centered Alignment Gap.

Furthermore, Delta Level Gap shows excellent performances on data set 5, which cannot be classified using non kernel HAC while easily separated by kernel HAC, as can be seen from experimental results.

5.5 Conclusion

According to all the experiments we have done (not all of them are presented here), we have observed that the initial Gap Statistic of Tibshirani is not always efficient in estimating the right number of clusters. Looking for the maximum

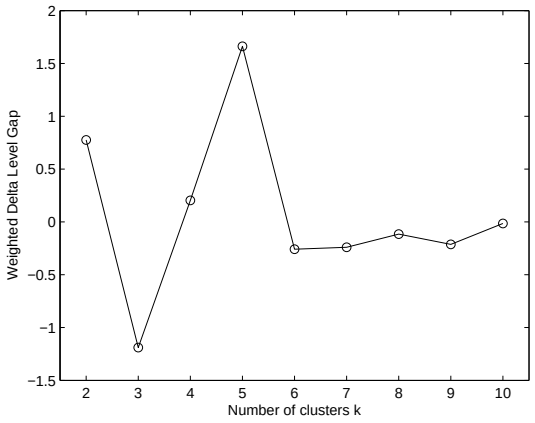


FIGURE 5.8: Number of clusters using Weighted Delta Level Gap.

value instead of selecting the value proposed as in equation A.18, appears to be a possible alternative. However, the Delta Level Gap seems to be the most reliable estimate among those we have studied and potentially one of the less sensitive to the choice of the null distribution of the data.

Chapter 6

Iterative Kernel-based Hierarchical Agglomerative Clustering

Contents

In real kernel-based clustering algorithms applications, selection of the kernel function and determination of the number of clusters are two issues of crucial importance. In chapter 3 and chapter 4, we have studied these two aspects respectively. In this chapter, we propose an iterative KHAC algorithm and try to find out a method to determine the gaussian kernel parameter (our method can be generalized to more complex kernels) and the number of clusters automatically. Our work begins with the evaluation of the effectiveness of the iterative KHAC when the number of clusters is known then we try to introduce several criteria to help finding the number of clusters when the number is unknown. As we know, there is no universal criterion to determine the number of clusters. It is still an open question and needs a further study. Of course there are strong relationships between determining the number of clusters and decision between a single cluster and multiple clusters hypotheses. In this chapter, we propose several methods to evaluate the effectiveness of the previous hypothesis tests, giving their deficiencies and potentials. Finally perspectives will be given.

6.1 Iterative KHAC when the number of clusters is known

6.1.1 Introduction of the proposed Iterative KHAC algorithm

In this section, we propose a kernel HAC based clustering algorithm whose basic idea is to iteratively cut the dendrogram top-down. At the first iteration, we apply Kernel HAC to the whole data set and we try to partition the data set into several sub-clusters by cutting the dendrogram. All the sub-clusters will be partitioned iteratively. We call any of these clusters a *Waiting Task*. A criterion is proposed to stop partitioning a *Waiting Task* and once no *Waiting Task* could be partitioned anymore, or when the desired number of clusters K, when known, is reached, the algorithm stops.

By construction, HAC (and KHAC) generates nested partitions of the data in k clusters, $1 \leq k \leq N$ where N is the number of observations. So the choice of

the number of clusters, when unknown, can be performed *a posteriori*. However, at each iteration of the agglomerative process used to construct the hierarchy, we are guaranteed to obtain the best (lowest) value of the dissimilarity. Of course, this local property does not guarantee that the final hierarchy will give an optimal partition for every value of k. We can only expect that successive optimal local solutions will not be very different from the global optimal one. This heuristic avoids the computational burden that would appear when trying to estimate the labels of observations, which is the aim of clustering. The algorithm is shown in Algorithm.2

Algorithm 1: Kernel HAC based Clustering Algorithm

step 1.Apply Kernel HAC to the original data set, obtaining the initial dendrogram

step 2.Determine the optimal kernel parameter σ^* and the optimal number of sub-clusters K^*

step 3.Calculate a Gap Statistic ([79]) to make the decision to cut or not the dendrogram to obtain K^* sub-clusters

while *at* ***step 3*** *it is decided to cut and while the number of clusters obtained is still smaller than K (when known)* **do**

  Cut the dendrogram into K^* parts to obtain these K^* sub-clusters and put these sub-clusters into the existing *Waiting tasks* list;

  Take any *Waiting task* and perform **step 2** and **step 3**;

end

The criterion used in step 3 is inspired by Gap Statistics of [79] which we have introduced in Chapter 5. The principle is to compare some separability criteria on the result obtained by the considered clustering algorithm with that under a single cluster hypothesis H_0 (null reference distribution, see Fig. A.10). In our previous work [53], instead of the within cluster dispersion, several other new criteria have been studied to estimate the number of clusters like Centered Alignment Gap, Weighted Delta Level Gap *etc.* The choice of a data distribution under H_0 (single cluster hypothesis) has been shown to be rather challenging [28]. Here we consider a uniform distribution over an area which is aligned with the principal components of the data set as the null reference distribution H_0.

To perform step 2, we firstly evaluate the maximum difference between two consecutive levels of the dendrogram by

$$\Delta h_i(\sigma, K) = h_i(\sigma, K) - h_{i-1}(\sigma, K)$$

and the optimal kernel parameter σ^* and the optimal number of clusters K^* are defined as:

$$[\sigma^*, K^*] = \arg max_{\sigma,K} \left(\max_i (\Delta h_i(\sigma, K)) \right)$$

Secondly, we define

$$\Delta h_{\max} \equiv \max_i \Delta h_i(\sigma^*, K^*)$$

We decide in favour of H_{K^*} (K^* clusters hypothesis) if $\Delta h_{max} > s_{1-\alpha}$.

To estimate the probability density function of Δh_{max} under the null hypothesis, we perform $N = 500$ independent estimations of Δh_{max} under the null hypothesis (single cluster) with the obtained values of σ^* and K^*. We define

$$s_{1-\alpha} = \mathrm{s} \mid \hat{P}(\Delta h_{max} \leq s \mid H_0) = 1 - \alpha$$

If Δh_{max} is larger than $s_{0.95}$, we partition the *Waiting Task* into K^* sub-clusters which are added to the *Waiting Task* list, else the *Waiting Task* is not partitioned and removed from the *Waiting Task* list.

6.1.2 Illustration of the iterative clustering procedure

The whole clustering process is depicted in Fig. A.11, A.12, A.13, **??**. This toy data set contains 400 2-dimensional data points. 3 iterations have been executed to obtain the $K = 5$ desired clusters. In the first iteration, we process all the data set (Fig. A.11a) and we obtain the corresponding dendrogram in Fig. A.11b. We then execute the procedure shown in Fig. A.10 and we have $\Delta h_{max} > s_{0.95}$ (Fig A.11c), which means that this *Waiting Task* should be partitioned. The clustering result after the first iteration is given in Fig. A.11d. In the second iteration we process the center part of the data set (first *Waiting Task*, Fig. A.12a) which is not partitioned because $\Delta h_{max} < s_{0.95}$ (Fig. A.12c). In the third iteration we take the second (and last) *Waiting Task* (Fig. A.13a) with a clustering result shown in Fig. A.13d. We stop the algorithm when the desired number of clusters $K = 5$ is reached. The final clustering result is given in Fig. A.14 which is satisfactory. Different optimal kernel parameters have been used at the different iterations. This allows the algorithm to focus on the specific data structure of every *Waiting Task*.

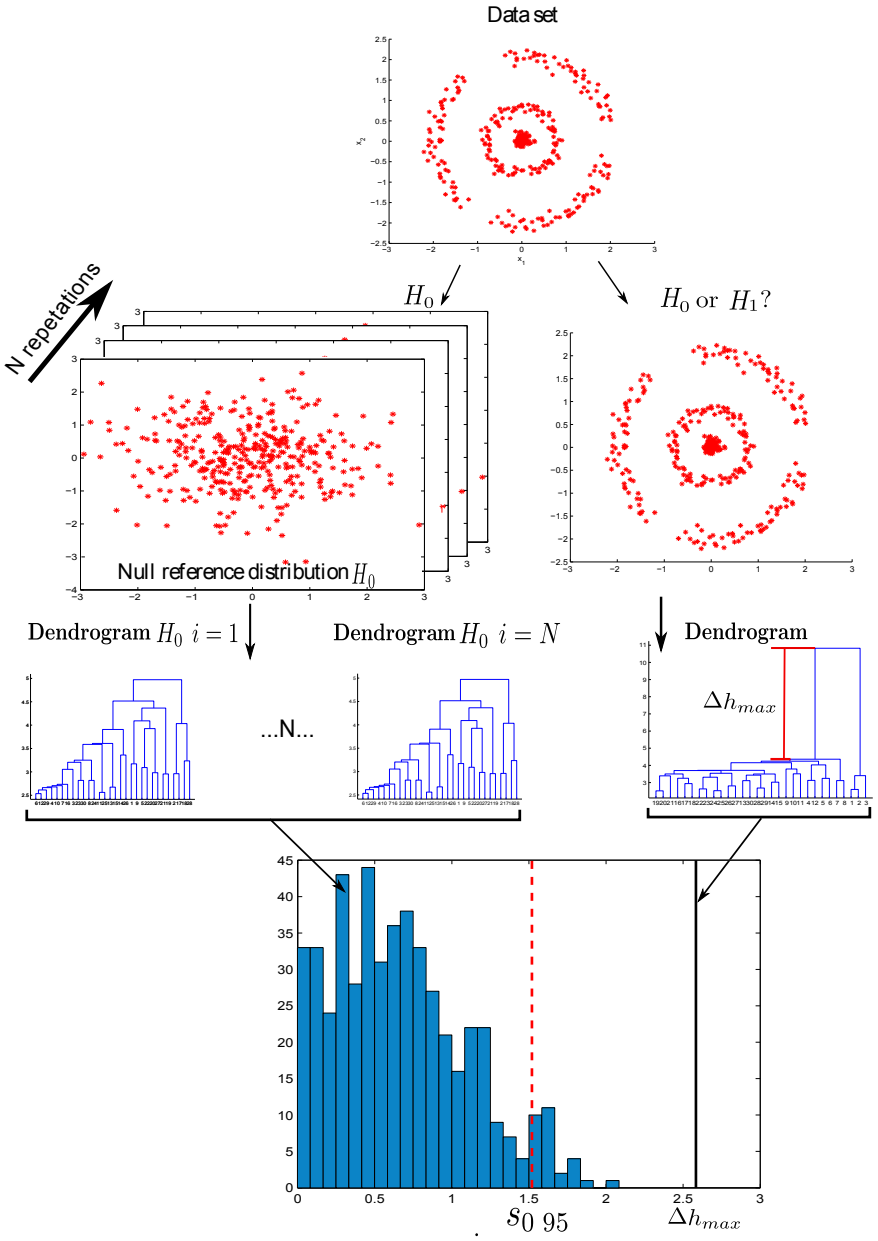


FIGURE 6.1: Illustration of the decision between a single cluster hypothesis (H_0) and a multiple cluster hypothesis (H_1) inspired by Gap Statistics

6.1.3 Experimental results

To benchmark our method, we present the results obtained on several synthetic data sets shown in Fig. A.15. In Fig. A.15a, we consider a data set from [81]. 3 iterations have been performed with respective kernel parameter $\sigma = 0.72, 0.82, 0.52$. The result is comparable to this of the original article. In Fig. A.15b and Fig. A.15c, we generate 2 2-dimensional distributions where the conventional kernel based

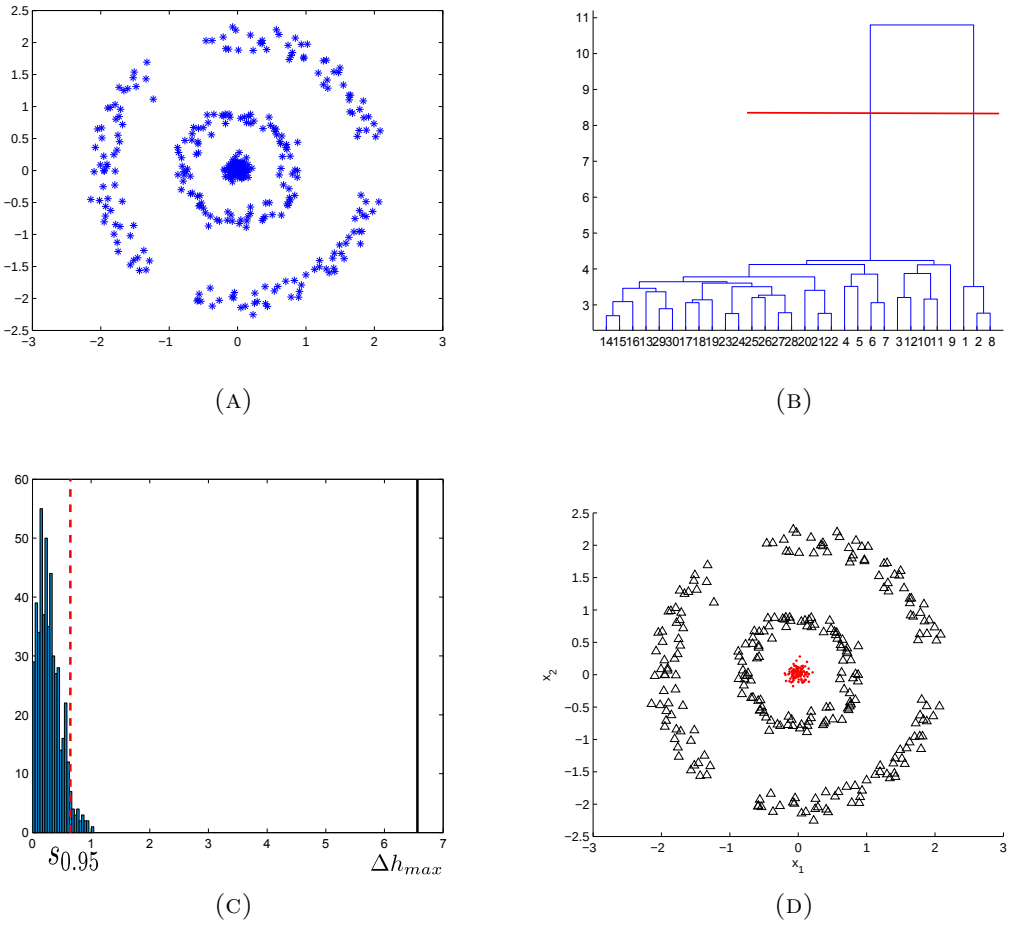


FIGURE 6.2: Illustration of the clustering procedure on a synthetic data set (1^{st} iteration). a: *Waiting Task* (initial data set). b: Obtained dendrogram $\sigma^* = 0.22$,$K^* = 2$. c: Histogram of the distribution of Δh_{max} under H_0, we have here $\Delta h_{max} > s_{0.95}$. d: Clustering result after the 1^{st} iteration.

methods (kernel k-means) often fail due to the difficulties in finding a single optimal kernel parameter σ. In Fig. A.15b, 2 iterations have been executed with respective kernel parameter $\sigma^* = 0.52, 1.2$. In Fig. A.15c, also 2 iterations have been executed with respective kernel parameter $\sigma^* = 0.22, 0.72$. In both cases, results are satisfactory. In Fig. A.15d, only 1 iteration is needed with $\sigma^* = 0.62$ and we succeed to partition the data as desired.

The method proposed in this paper provides satisfactory results for some complicated non-linearly separable clustering tasks when the number of clusters is supposed to be known. However if the number of clusters is not given, we apply the iterative KHAC on data set from [25] which has been often used to benchmark clustering algorithms because it contains features that are known to create difficulties for algorithms like narrow bridges between clusters, uneven clusters *etc.*

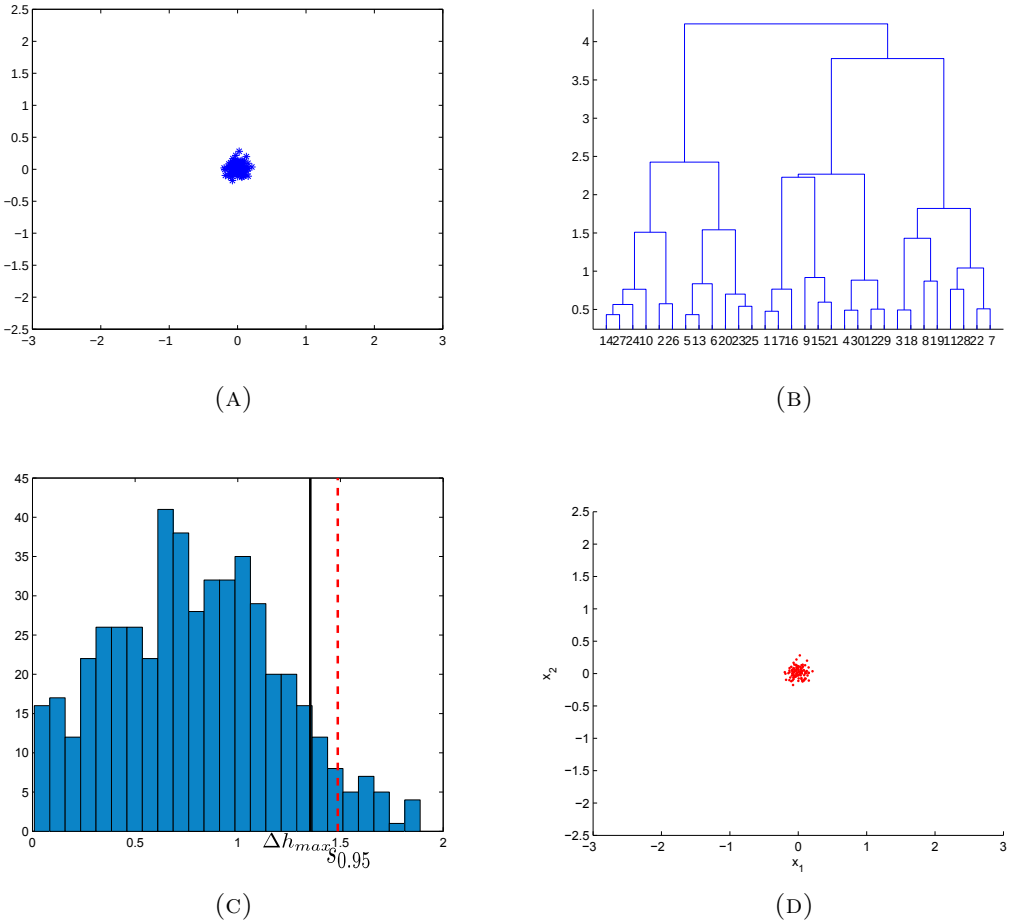


FIGURE 6.3: Illustration of the clustering procedure on a synthetic data set (2^{nd} iteration). a: *Waiting Task*. b: Obtained dendrogram $\sigma^* = 1.52$, $K^* = 3$. c: Histogram of the distribution of Δh_{max} under H_0, we have here $\Delta h_{max} < s_{0.95}$. d: Clustering result after the 2^{nd} iteration.

As shown in Fig. 6.7, the part in the rectangular area is desired to be partitioned into 3 clusters but we over-divide this part. We find out the 3 real clusters but we also further partition one of them into 4 parts. When we try to determine the number of clusters from the data set itself we observe that the choice of the data distribution under the null hypothesis can strongly impact the result obtained (see for example Fig. 6c in which the observed value of $\Delta h_{\max}$ is close to the threshold).

Now we consider if the number of clusters is known, how we can stop breaking the dendrogram. The kernel-based iterative HAC when the number of clusters is unknown is the task of the nest section.

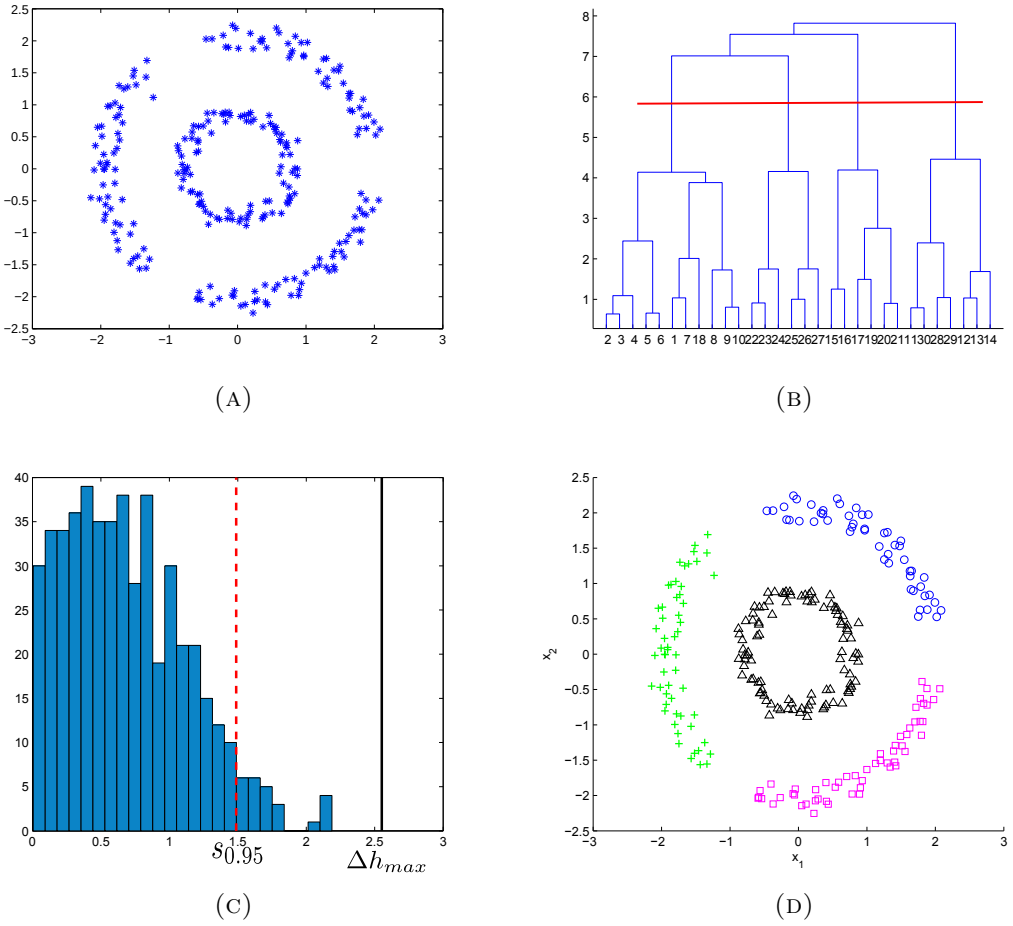


FIGURE 6.4: Illustration of the clustering procedure on a synthetic data set (3^{rd} iteration). a: *Waiting Task*. b: Obtained dendrogram $\sigma^* = 1.12$, $K^* = 4$. c: Histogram of the distribution of Δh_{max} under H_0, we have here $\Delta h_{max} > s_{0.95}$. d: Clustering result after the 3^{rd} iteration.

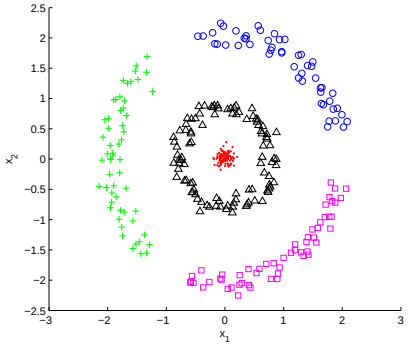


FIGURE 6.5: Illustration of the clustering procedure on a synthetic data set: final result

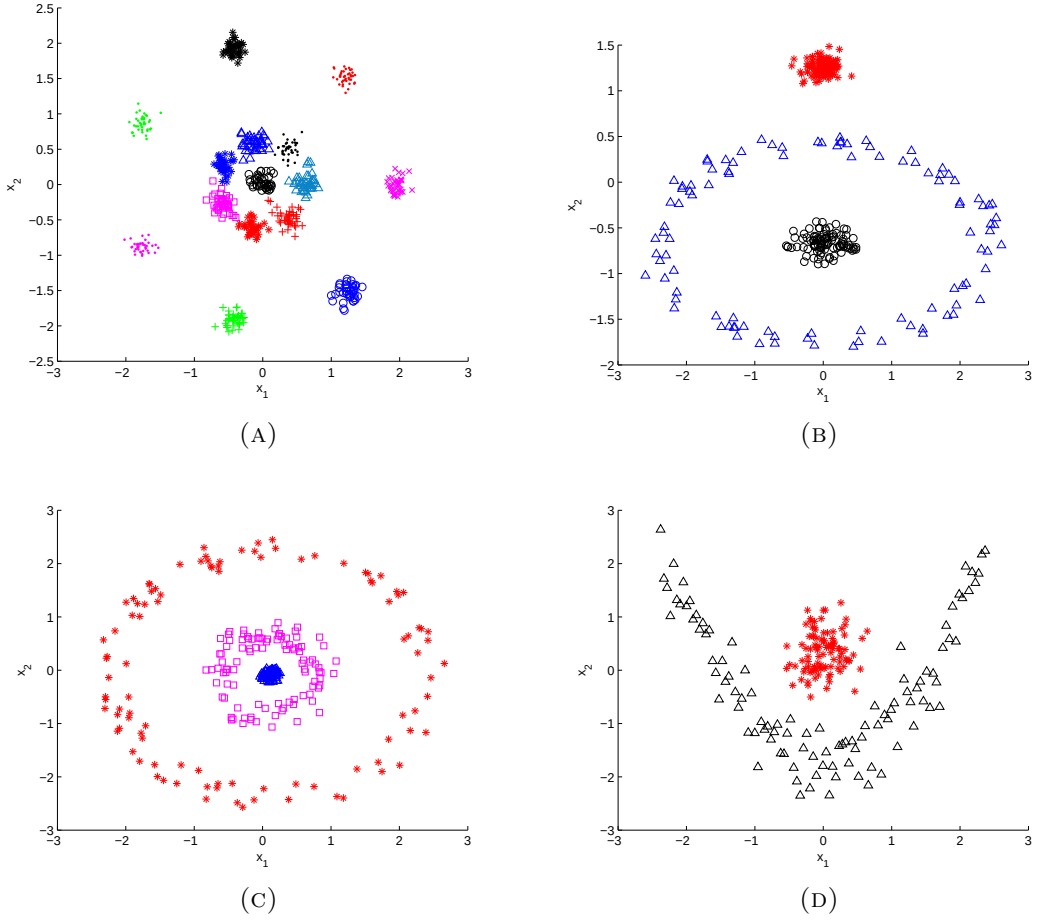


FIGURE 6.6: Results on synthetic distributions

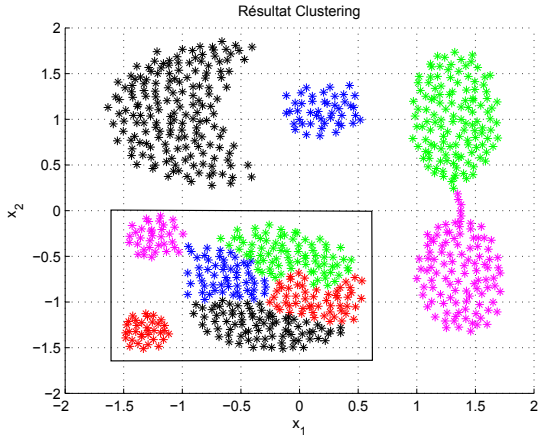


FIGURE 6.7: Illustration of wrong number of clusters determination

6.2 Iterative KHAC when the number of clusters is unknown

In the previous section, we proposed an algorithm based on KHAC when the number of clusters is known. The main idea is to cut the dendrogram iteratively and the key point is how to stop partitioning clusters. When the number of clusters is unknown, obviously we can not use it as a stopping criterion anymore. As we see in the last example, Δh_{max} may not be the best statistic to decide to cut the dendrogram. In this section, we consider to do a further study on the influence of the choice of null reference distribution and try to define a better criterion to stop partitioning clusters.

6.2.1 Centered Alignment and inter-cluster dispersion criteria

In Chapter 4 and Chapter 5, we have shown that *centered alignment* and *inter-cluster dispersion* play an important role in kernel parameter selection and number of clusters determination. Besides, clustering algorithms based on these two criteria have shown to give good performances [84]. We suppose that they could well interpret the similarity/disimilarity between clusters. Hence it is well worth studying them in the case of one cluster (H_0) and compare with the case where there are two clusters or more (H_1).

6.2.1.1 Criterion evaluation based on perturbation of between-cluster distance

Given a criterion C, consider two cases: criterion under null reference distribution (H_0) and under the case where there are two or more clusters (H_1). We introduce a tiny perturbation of inter-clusters distance Δd and we expect that under H_0, the criterion evolves rapidly however under H_1, the criterion stays relatively stable. We propose a method to verify the effectiveness of this idea as follows:

We measure the variation of the considered criterion with the change of the distance d (as in Fig. A.16) between two clusters in the feature space. In the case of two clusters, let $m_{i\mathcal{H}}$, $i = 1, 2$ be the center of each cluster, $m_{0\mathcal{H}}$ be the center of

the whole data set in feature space. For a given inter-cluster distance d, we calculate a criterion, noted as $C(d)$ and then we sightly elongate the distance between the two clusters in feature space in the following way:

$$\phi(x)_\alpha = \phi(x) + \alpha(m_{i\mathcal{H}} - m_{0\mathcal{H}}), \alpha > 0, \alpha \to 0 \tag{6.1}$$

Then the corresponding estimates variation of criterion w.r.t α is

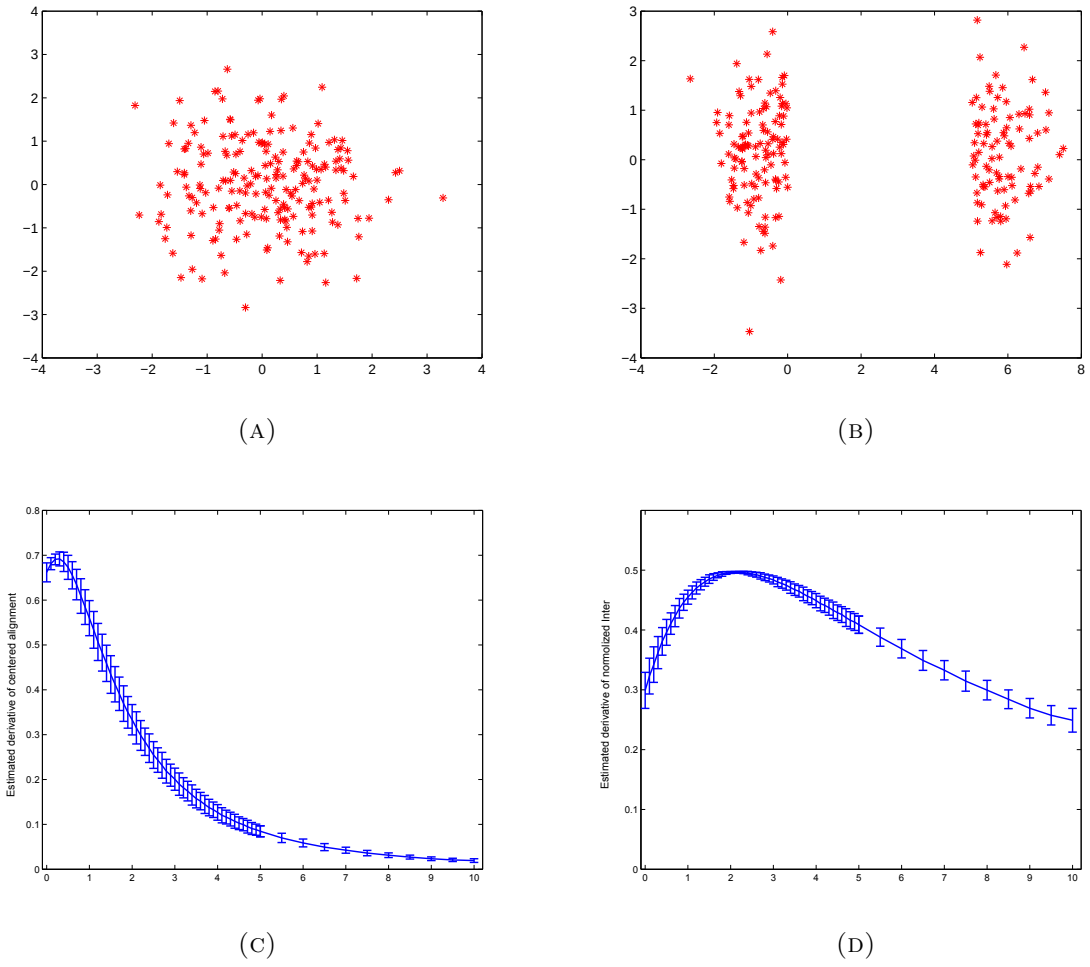
$$S = \frac{\partial C(d,\alpha)}{\partial \alpha} \mid_{\alpha=0} = \frac{C(d,\alpha) - C(d,0)}{\alpha} \tag{6.2}$$

where C depends on the Gram matrix of the perturbed data, which can be easily calculated. The idea is that the estimated partial deviative of the criterion will be small if two clusters are well separated (where d is relatively large). On the contrary, if its variation is important, we consider that there is only one cluster(H_0) (which corresponds to a small value of d).

Remark. When a point x is moved in the feature space according to Eq. A.19, the new point no longer satisfies $\parallel \phi(x)_\alpha \parallel = 1$. Hence the modified space is no longer a Gaussian kernel induced RKHS. However this does not impact our method as the new kernel matrix can be constructed by the old kernel matrix according to Chapter 2.

We now present the results (Fig. A.16) obtained from simulations by selecting different values of d to evaluate the sensibility of the criterion to detect H_1. We increase the value of distance d gradually as shown in Fig. A.16a and Fig.A.16b. For each value of d, we do 100 simulations. Fig. A.16c and Fig. A.16d gives the curve of the variation of *centered alignment* and *normalized inter-cluster dispersion* as defined in Eq. **??** respectively and also their standard deviations.

Results correspond well to our assumption. As distance d increases, the criterion deviation is smaller. It means that both criteria we considered here could be good indicators to know whether there are one or two clusters. As we can see, the *centered alignment* seems to be a good indicator of the 'distance' between clusters which allows us to decide between H_0 or H_1.



(A) (B)

(C) (D)

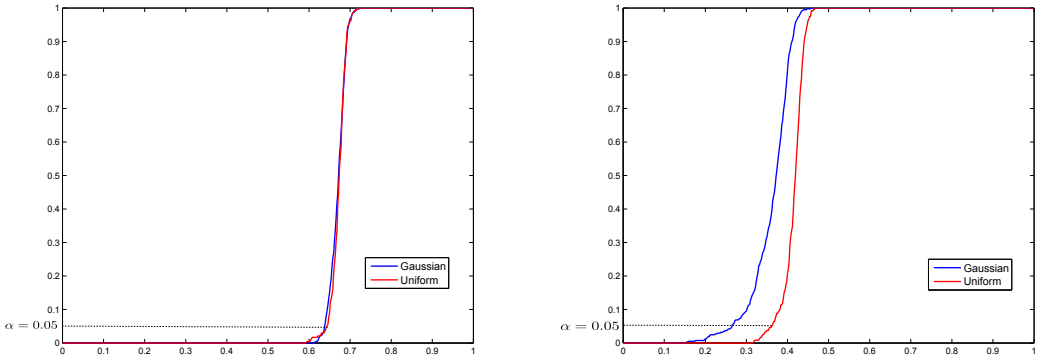
FIGURE 6.8: Illustration of deviation of the criteria as a function of *d*. A: Data set with a small d (d=0). B:Data set with a relative large d (d=4). C: Derivative of centered alignment and its standard deviation as a function of d. D: Derivative of normalized inter-cluster dispersion and its standard deviation as a function of d.

6.2.1.2 Determination of thresholds decision

The threshold decision is defined by

$$P(S < s \mid H_0) = \alpha \tag{6.3}$$

where S is defined in Eq. A.20. s can be computed using the cumulative distribution function of S under H_0. This can be done by simulations for any value of the number of observations and the dimension of the input space. Of course this makes sense if the cumulative distribution function of S under H_0 is independent from the data distribution under H_0.



(A) Cumulative distribution functions of centered alignment

(B) Cumulative distribution functions of normalized inter-cluster dispersion

FIGURE 6.9: Cumulative distribution functions of centered alignment and inter-cluster dispersion under gaussian and uniform null hypothesis (case of number of clusters is 2).

For the *centered alignment*, S has very similar cumulative distribution function when the data is gaussian or uniform. This is not the case for normalized inter-cluster dispersion.

6.2.2 Criterion of Q Nearest Neighbours

In classification, most of the controversial points lie on the boundary of the clusters. So we try to propose a criterion which focus on the points close to the boundary(ies) and ignore the points far away from the boundary(ies). This idea also makes the criterion to be less dependent on the data distribution. We illustrate our approach on the following figure:

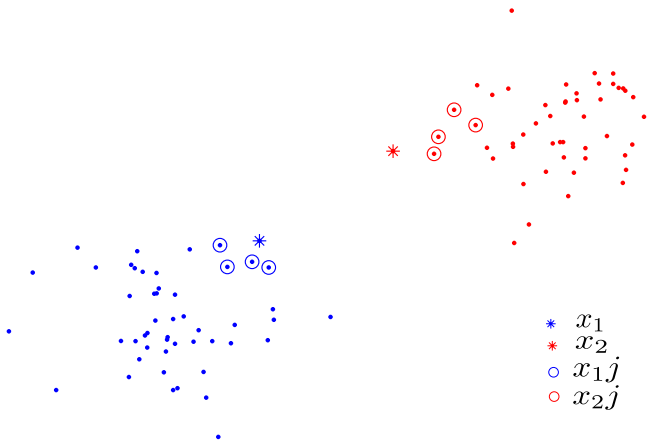


FIGURE 6.10: Q Nearest Neighbors (QNN) criterion in the case of two clusters

As shown in Fig. 6.10, x_1 and x_2 (noted by $*$) are the nearest pair of points between the two clusters. Then we find out the Q (here $Q = 5$) points (x_1 and x_2 included) of each cluster which are the nearest neighbors of x_1 and x_2 respectively in each cluster, noted as x_{1j} and x_{2j}, $j = 1, ..., 4$. Then we define

$$\overline{\text{Intra}} = \frac{\sum_{j=1}^{Q} \|x_{1j} - x_1\|^2 + \sum_{j=1}^{Q} \|x_{2j} - x_2\|^2}{2(Q-1)}$$

$$\overline{\text{Inter}} = \frac{\sum_{i=1}^{Q} \sum_{j=1}^{Q} \|x_{1i} - x_2 j\|^2}{Q^2}$$

The Q Nearest Neighbors criterion (QNN)is given by:

$$\text{QNN} = \frac{\overline{\text{Intra}}}{\overline{\text{Inter}}}$$

Definition 15. Given a data set where we have K clusters and let x_m, $m = 1, ..., K$ be the points from nearest pairs of points between clusters, we define

$$\overline{\text{Intra}_K} = \frac{\sum_{m=1}^{K} \sum_{j=1}^{Q} \|x_{mj} - x_m\|^2}{2(Q-1)\frac{k(k-1)}{2}}$$

$$\overline{\text{Inter}_K} = \frac{\sum_{m=1}^{K} \sum_{n=m+1}^{K} (\sum_{i=1}^{Q} \sum_{j=1}^{Q} \|x_{mi} - x_n j\|^2)}{Q^2 \frac{k(k-1)}{2}}$$

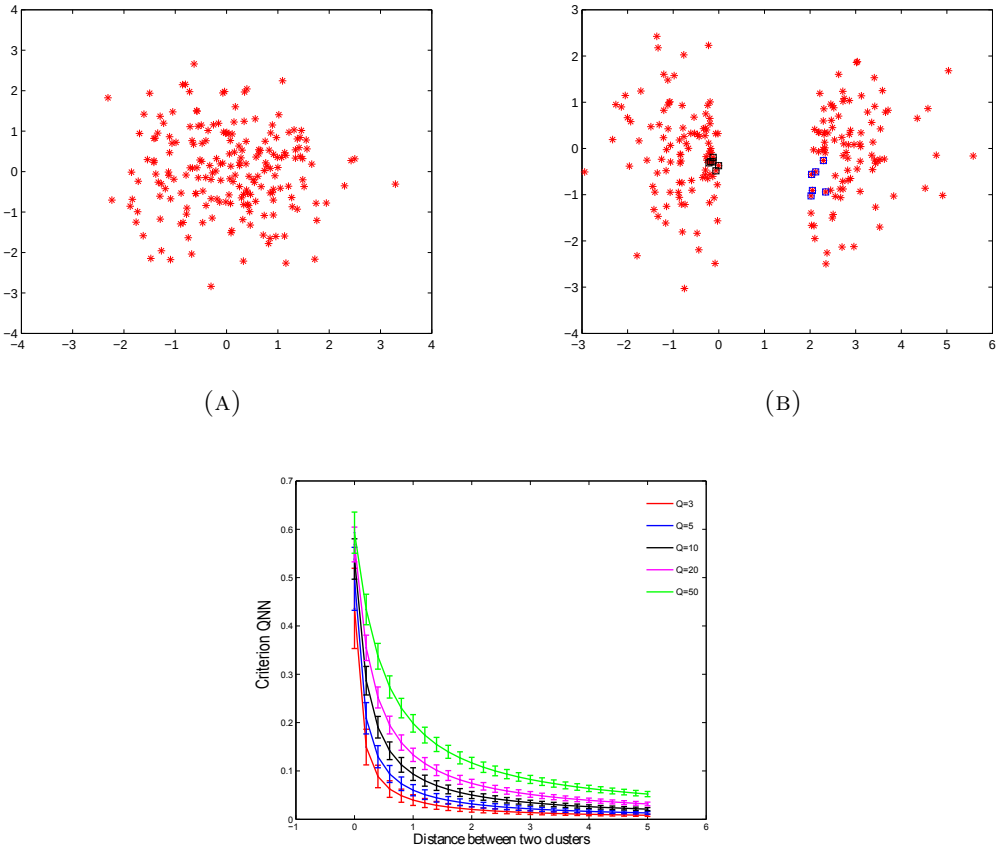
Then the QNN of these K clusters is given by:

$$\text{QNN} = \frac{\overline{\text{Intra}_K}}{\overline{\text{Inter}_K}} = \frac{Q^2}{2(Q-1)} \frac{\sum_{m=1}^{K} \sum_{j=1}^{Q} \|x_{mj} - x_m\|^2}{\sum_{m=1}^{K} \sum_{n=m+1}^{K} (\sum_{i=1}^{Q} \sum_{j=1}^{Q} \|x_{mi} - x_n j\|^2)} \tag{6.4}$$

6.2.2.1 Study of QNN as a function of distance between clusters

As shown in Fig. 6.11, we increase the distance d between clusters and obtain the curve of QNN which is a function of d.

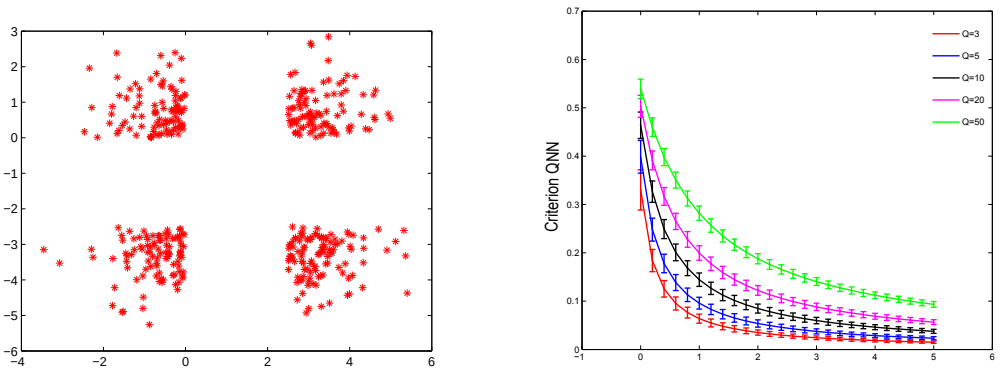
500 simulations have been done for each value of d. Each cluster contains 100 points and according to the definition of QNN, it also depends on the value of Q. Fig. 6.11c provides the results obtained with different values of Q. In general, the value of QNN and its deviation w.r.t d decreases as the inter-cluster distance increases. Hence a threshold could be found to determine if we should partition the clusters or not. If the value of QNN is smaller than this threshold, we consider that there should be 2 clusters. We also notice that, in the case where the data set is drawn from a gaussian distribution, for a given inter-cluster distance d, when the value of Q increases, QNN increase as well. Fig 6.12 presents results in the case of 4 classes.



(A) (B)

(C)

FIGURE 6.11: Illustration of variation of QNN as a function of the distance between clusters. A,B: Data set where inter-cluster distance increases. D: Curves of the mean value of QNN for different values of Q as a function of d and their standard deviations.



(A) (B)

FIGURE 6.12: Illustration of variation of QNN as a function of the distance between clusters (case of 4 clusters)

Q=2			Q=3		
nbc	Counter	Threshold	nbc	Counter	Threshold
2	**430**	**0.0693**	**2**	**458**	**0.1552**
3	477	0.0490	3	448	0.0879
4	93	0.0176	4	94	0.0387
Q=5			Q=10		
nbc	Counter	Threshold	nbc	Counter	Threshold
2	**456**	**0.2584**	**2**	**463**	**0.4181**
3	475	0.1399	3	449	0.2089
4	69	0.0664	4	88	0.1132
Q=15			Q=20		
nbc	Counter	Threshold	nbc	Counter	Threshold
2	**473**	**0.5092**	**2**	**529**	**0.5686**
3	454	0.2574	3	447	0.2982
4	73	0.1548	4	24	0.1921

TABLE 6.1: Results of 1000 simulations for QNN under null hypothesis. nbc stands for number of clusters. Counter indicate the number of times out of 1000 that we find the corresponding number of clusters. $\alpha = 0.05$ for the definition of the threshold. Boldface number will be used later

6.2.2.2 Determination of the threshold

S is the observed value of criterion QNN. If $S < s$, we assume that there are more than one cluster (D_1).

$$s : P(S < s \mid H_0) = P(D_1 \mid H_0) = \alpha$$

We implement 1000 simulations under the null hypothesis where there is only one cluster by applying KHAC. Results are given in Table. 6.1:

Remark. As KHAC provides results with different numbers of clusters, the threshold is a function of the number of clusters proposed by KHAC. For example, when $Q = 5$, 456 times out of 1000 provide a result of 2 clusters and $P(D_1 : 2 \text{ clusters} \mid H_0) = P(S < \text{threshold}(2 \text{ clusters})) = 0.05$, where $threshold = 0.2584$.

Fig. 6.13 provides the cumulative distribution functions of QNN in the case of 2 clusters for different values of Q. Experiments, as before, have been done under both gaussian and uniform null hypotheses. In this case, according to the curves shown in the figure, the cumulative distribution function of QNN appears to be

rather independent of the distribution. Given $\alpha = 0.05$, the threshold s for each Q could be determined using the corresponding curve.

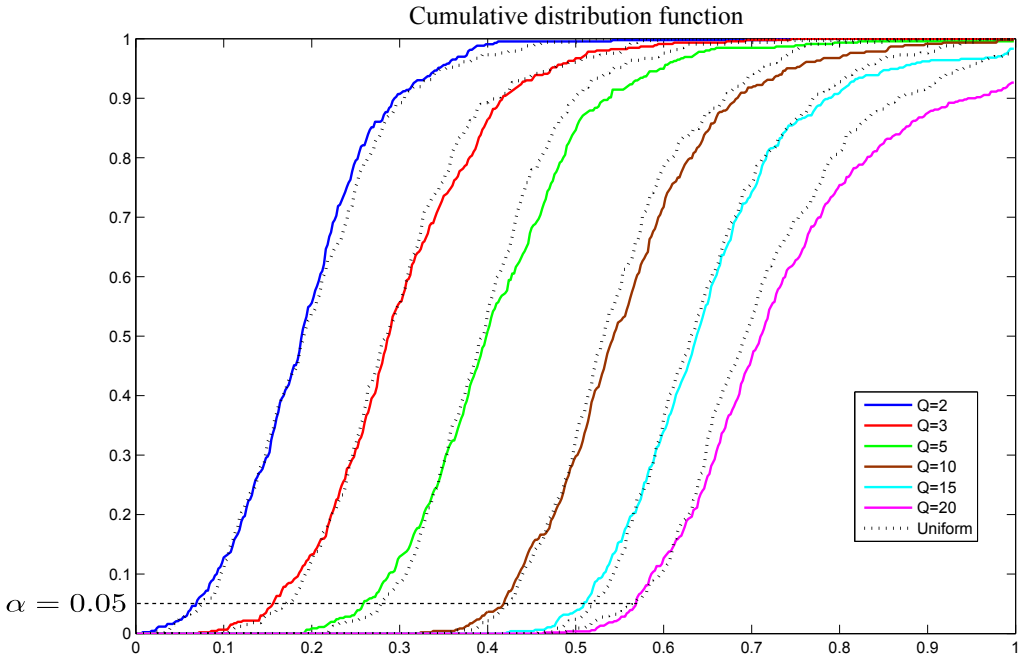


FIGURE 6.13: Cumulative distribution functions of QNN in the case of 2 clusters.

Still considering the case where the number of clusters is 2, we test the ability of QNN to determine the right number of clusters when the distance between clusters changes. We consider the data set shown in Fig. 6.11. Fig. 6.14 shows the estimated *power* function $P(D_1 \mid H_1)$ where 2 clusters are found for different values of Q. Obviously, when the distance d between clusters is 0,

$$\text{Power}(0) = \lim_{H_1 \to H_0} P(D_1 \mid H_1) = P(D_1 \mid H_0) = \alpha$$

As the distance d increases, the *power* function increases and when $d = 1$, the *power* reaches about 0.8. In this case (100 points in the data set), $Q = 3, 5, 10, 15$ have very similar performances. We now take $Q = 5$ in all experiments.

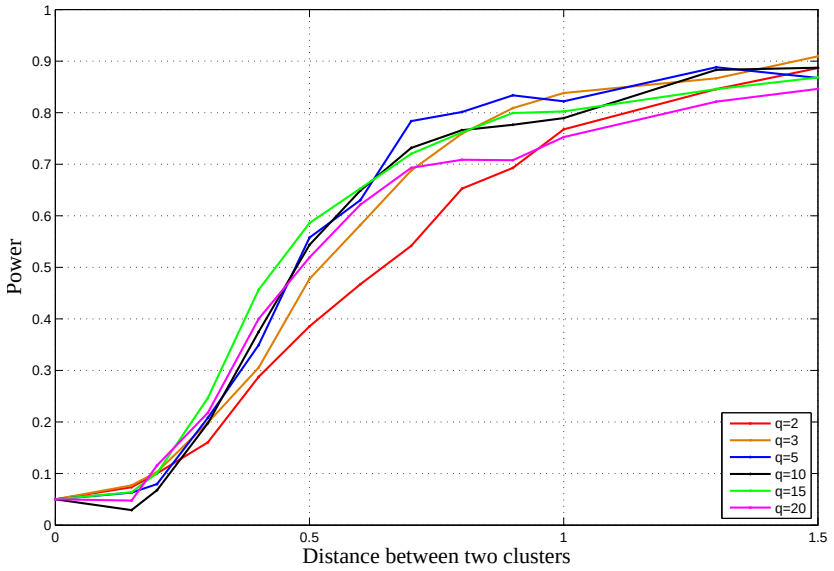


FIGURE 6.14: Power curve of QNN with change of distance between clusters (case of number of cluster is 2)

6.3 Conclusion

As demonstrated before, *centered alignment* and *inter-cluster dispersion* criteria are not robust enough for different data distributions to stop cutting the dendrogram. QNN seems to be a good criterion and data sets like in Fig. 6.15 could be partitioned automatically but the results are not always satisfactory. Many deficiencies and limitations exist. QNN is sensitive to outliers (Fig. 6.16) which greatly influence the final result. A better criterion which could avoid the outlier problem will help a lot in improving the performance. Besides, QNN relies on the number of clusters founded at each iteration. Hence an improper number of cluster will lead to a dissatisfactory result.

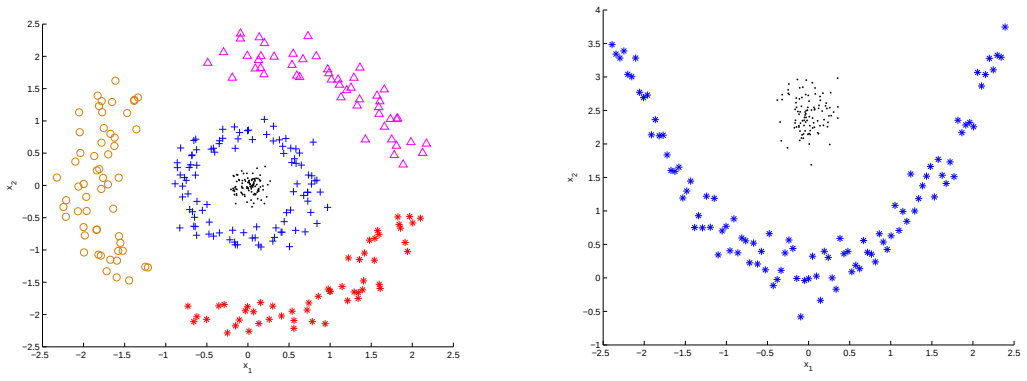
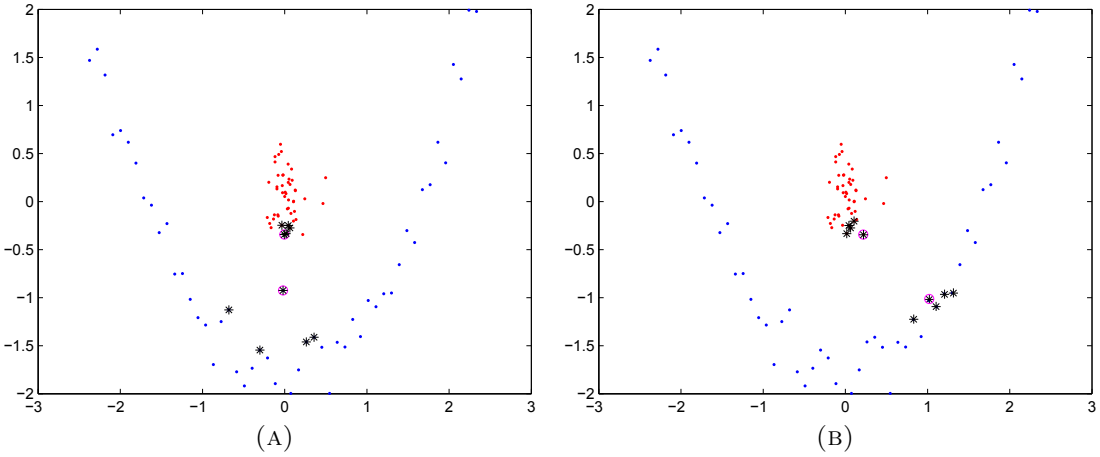


FIGURE 6.15: Data set



(A) (B)

FIGURE 6.16: Limitation induced by outliers between the two clusters which are associated to a high variance of the QNN statistic

Chapter 7

Conclusion

Identifying significative groups in a given data set owns great importance in this world where data exists everywhere around us. Classification methods, especially unsupervised classification techniques are necessary in development of a large number of disciplines. Clustering, as a useful tool of unsupervised classification, attracts researcher's attention since a long time. However, it's not a easy problem where dilemmas exist all the time. A mass of clustering algorithms have been proposed, however there is no universal method who is capable to deal with all the problems. Most of the clustering methods are very independent of the problems and the user's needs.

In this document, we focus on hierarchical agglomerative clustering using Ward linkage, which is proved to be an interesting method by another recently arisen criterion Maximum Mean Discrepancy (MMD). MMD has firstly been proposed to measure difference between different distributions and can easily be embedded in to a RKHS. Thus kernel based HAC using Ward linkage has been studied in our research.

Kernel based clustering algorithms shown competitive performance compared with the conventional methods and in most cases, they provides better performance owing to their ability of transforming the nonlinear problem into a linear one in a higher dimensional feature space. We proposed to use *centered alignment* and *inter-cluster dispersion* for kernel parameter selection in KHAC and use gap statistic theory to determine the number of clusters. We then induced a method to solve both of these previous problem at the same time and identify clusters automatically.

Centered alignment and *inter-cluster dispersion* provides satisfactory kernel parameter selections which enable us to obtain a proper dendrogram. A good dendrogram means a qualified data structure which make the clustering algorithm to be meaningful.

The Gap statistic using Δh_{max} provides satisfactory result in determination of number of clusters. However it is still not robust enough for iterative KHAC algorithm that we proposed later. The iterative KHAC shows interesting potentials but still needs further work.

Better technique to determine the number of clusters will help a lot in improving performance of iterative KHAC. For example, in place of breaking the dendrogram at the level which is decided by a criterion, we always break the dendrogram into 2 clusters at each iteration. This seems to be a good idea as we avoid the effects of the criterion of number of clusters determination to the iterative KAC. Only the criterion to stop cutting dendrogram should be considered. A better choice of null reference distribution should be considered as well. Finally, there is still much works to do about the criteria we proposed to stop breaking the dendrogram. How to solve the outlier problem and how to find a better method to determine the threshold of derivative of centered alignment are worth of work.

Appendix A

Résumé en Français

A.1 Introduction

Au cours des dernières années, les méthodes à noyau, dont les éléments les plus connus sont les Support Vector Machines, ont démontré une grande efficacité en apprentissage supervisé. Elles ont démontré leur capacité à engendrer des solutions généralisantes et à analyser de grands ensembles de données. Cependant, dans les applications du monde réel, il peut être utile de caractériser la structure des données sans connaître leur appartenance aux classes. Cela rend le problème non supervisé. Dans le cas non supervisé, l'objectif est de trouver la combinaison optimale des étiquettes afin que les données de même étiquette soient plus semblables entre elles qu'avec celles d'étiquettes différentes. Essayer toutes les combinaisons possibles des étiquettes est un problème NP-difficile. En conséquence, il faut proposer des heuristiques permettant de contourner la difficulté précédente.

Il parait intéressant d'étendre les méthodes à noyau au cas de la classification (binaire ou multiple) non supervisée, de formuler le problème d'optimisation correspondant et de proposer des heuristiques permettant de résoudre le problème en temps raisonnable.

A.1.1 Classification non supervisée

Il n'existe aucun algorithme universel de classification non supervisée. La plupart des algorithmes sont dépendant du problème à résoudre. La recherche d'un algorithme adapté à une tâche donnée nécessite de procéder à une étape d'inférence en essayant de répondre aux questions suivantes :

- Qu'est-ce qu'un groupe homogène d'observations (cluster) ?
- Comment extraire et sélectionner les caractéristiques de l'espace de représentation ?
- Quelle est la bonne métrique pour un ensemble de données?
- Quelle mesure de similitude utiliser ?
- Quel algorithme de classification utiliser ?
- Combien de classes avons-nous ?

- Comment valider un groupe ?
- Comment traiter un grand volume de données ?

A.1.1.1 Clustering algorithms

La Fig. A.1 présente une taxonomie des algorithmes de clustering. Celle-ci n'est bien évidemment pas unique. En général, les algorithmes de classification peuvent être divisés en deux classes: les algorithmes hiérarchiques et algorithmes de regroupement ou de partitionnement.

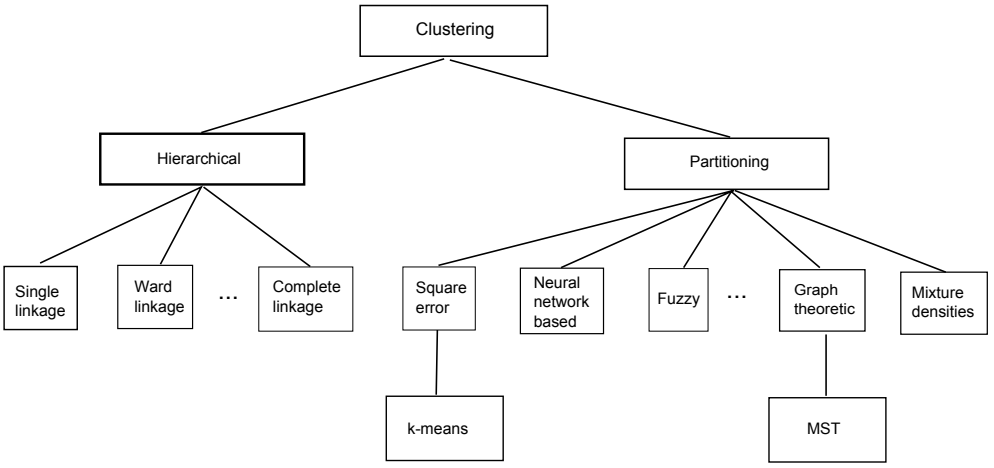


FIGURE A.1: Une taxonomie de la classification.

Les algorithmes hiérarchiques fournissent une représentation de la structure de données par un dendrogramme qui représente les données sous la forme de partitions imbriqués leur associe un niveau de similitude.

Les Algorithmes de partitionnement. Dans les algorithmes de partitionnement, les clusters sont déterminés par l'optimisation d'une fonction objectif.

A.1.2 Classification et méthodes à noyau

Les méthodes à noyau ont fait l'objet d'un intérêt majeur par leur capacité à réaliser des tâches non linéaires dans l'espace initial par des opérations linéaires réalisées après une transformation non linéaire $\phi(.)$ des données initiales. Dans le domaine de l'apprentissage à partir d'exemples, l'astuce du noyau, qui consiste à

calculer un produit scalaire dans l'espace transformé par l'évaluation d'une fonction dans l'espace initial, a été introduite par Aizerman [1]. Elle est devenue célèbre grâce aux Support Vector Machines (SVM) initialement proposées par Cortes et Vapnik [14]. Les SVM ont montré de très bonnes performances dans de nombreux problèmes et ce succès a engendré le développement de l'utilisation de l'astuce du noyau dans d'autres algorithmes comme l'Analyse en Composantes Principales à noyau (KPCA) [69], le filtrage adaptatif non linéaire [65] etc. Les méthodes à noyau ont été extensivement utilisés dans les problèmes de classification supervisée et ont été étendus à la classification non supervisée.

Un grand nombre d'algorithmes de clustering à noyau ont émergé en raison de l'utilisation massive de produits scalaires dans les méthodes linéaires. La plupart de ces algorithmes sont des versions *kernelisées* des algorithmes classiques correspondants. Une revue des méthodes à noyau pour le clustering est proposée dans [22, 48, 63]. En comparaison avec les algorithmes classiques correspondants, leurs versions *kernelisées* ont démontré de meilleures performances pour les ensembles de données non-linéairement séparables dans l'espace initial

.

A.1.3 Motivation de la recherche

La discrépance maximale de la moyenne (Maximum Mean Discrepancy, MMD) est une *distance* mesure entre deux densités de probabilité. Elle est utilisée pour comparer deux distributions. La MMD est aisée à évaluer dans un espace de Hilbert à noyau reproduisant (RKHS). On peut trouver une revue des mesures *kernelisées* de distance entre lois de probabilité dans [88].

Dans nos travaux, nous montrons que le carré de la MMD est fortement relié au critère de Ward, qui est largement utilisé en Classification Ascendante Hiérarchique (CAH). La CAH dépend de calculs de distance qui reposent sur des produits scalaires. On peut donc aisément effectuer une CAH après transformation des données d'entrée dans un espace de dimension plus élevée en utilisant une transformation non linéaire, les produits scalaires correspondants pouvant être évalués à l'aide de l'astuce du noyau [1].

Compte tenu des liens entre le critère de Ward et la MMD (mesure de distance entre lois de probabilité), nous proposons de réaliser la classification (détermination

des étiquettes) par maximisation du critère de Ward dans le cadre d'une procédure hiérarchique en introduisant la Classification Ascendante Hiérarchique à Noyau (KHAC).

A.2 Eléments fondamentaux des noyaux

A.2.1 Introduction

Les méthodes à noyau reposent sur une transformation non linéaire des données initiales qui conduit à des traitements linéaires dans l'espace d'arrivée $\mathcal{H}$, en général de dimension importante voire infinie. Le calcul des produits scalaires dans cet espace pourra être réalisé dans l'espace initial à l'aide de l'astuce du noyau.

Un ***Noyau*** est une fonction k qui pour tout $x, y \in X$ satisfait

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{A.1}$$

où ϕ est une application de X à dans $\mathcal{H}$ appelé espace transformé.

$$\phi : x \longmapsto \phi(x) \in \mathcal{H} \tag{A.2}$$

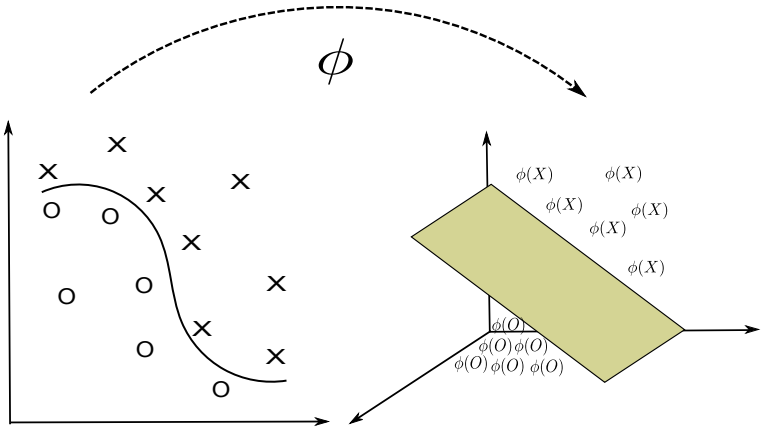


FIGURE A.2: Les données initiales sont transformées de l'espace d'entrée, par la transformation ϕ, vers un espace de grande dimension où le problème non linéaire devient linéaire.

A.2.2 Eléments fondamentaux de la théorie des noyaux

Cette section présente les définitions et propriétés essentielles à la compréhension de la théorie des noyaux.

A.2.2.1 Espace de Hilbert

Definition 16. Un espace associé à un produit scalaire est un ***espace de Hilbert*** s'il est complet.

Le caractère complet signifie que chaque séquence de Cauchy $\{h\}_{n\geq 1}$ converge dans $\mathcal{H}$.

A.2.2.2 Espace de Hilbert à noyau reproduisant

Definition 17. $k(.,.)$ est un ***Noyau reproduisant*** [27] d'un espace de Hilbert $\mathcal{H}$ si

$$f(x) = \langle k(x,.), f(.)\rangle_{\mathcal{H}}, \forall f \in \mathcal{H}$$

Definition 18. ***Espace de Hilbert à noyau reproduisant*** [36] (RKHS) est un espace de Hilbert possédant un noyau reproduisant et dans lequel les fonctions d'évaluation $f(.)$ sont bornés, c.-à-

$$\exists M > 0 \quad : |\delta_x(f)| = |f(x)| \leq M\|f\|_{\mathcal{H}}$$

A.2.2.3 Caractérisation des noyaux

Caractérisation des noyaux [67] Une fonction $k : X \times X \to \mathbb{R}$ peut être écrite comme

$$k(x,y) = \langle \phi(x), \phi(y)\rangle_{\mathcal{H}}$$

où x, y sont injectées dans un espace de Hilbert par une transformation ϕ, si et seulement si elle satisfait la propriété de semi-définie positivité.

Le théorème de **Moore-Aronszajn** [3] stipule que pour chaque fonction définie positive $k(.,.)$, il existe un RKHS unique.

A.2.2.4 Matrice de Gram

Definition 19. Etant donné un noyau k et un ensemble de données $X = \{x_1, ...x_n\}$, la matrice qui contient les produits scalaires de toute paire d'éléments est appelée matrice de Gram. Son terme général est noté K_{ij} et on a :

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$$

La matrice de Gram est symétrique car $K_{ij} = K_{ji}$ et est semi-définie positive.

A.2.3 Noyaux élémentaires

A.2.3.1 Polynomial kernels

Definition 20. Le **Noyau Polynômial** est défini par

$$k_d(x, y) = (\langle x, y \rangle + R)^d \tag{A.3}$$

où R et d sont deux paramètres définissant l'élément de la famille.

A.2.3.2 Noyau gaussien

Les noyaux gauissiens sont les noyaux les plus utilisés en apprentissage. La définition d'un noyau gaussien est la suivante :

Definition 21.

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

Il a été montré que le noyau gaussien présente en général de meilleures performances que le noyau polynômial.
Dans les chapitres suivants, nous nous intéressons spécifiquement au noyau gaussien, sauf spécification contraire.

A.3 Discrépance maximale de la moyenne et classification ascendante hiérarchique par critère de Ward

La discrépance maximale de la moyenne (MMD) est une *distance* entre deux densité de probabilité. Elle est utilisée pour comparer deux distributions. Il a été montré que des densité de probabilité pouvaient être injectées dans un espace de Hilbert à noyau reproduisant (RKHS). Dans cet espace, et sous certaines conditions, la discrépance maximale de la moyenne (MMD) est très facile à calculer. Considérant de plus que le carré de la discrépance maximale de la moyenne (MMD) est fortement relié au critère de Ward largement utilisé en classification ascendante hiérarchique, nous introduisons la classification ascendante hiérarchique à noyau (KHAC).

A.3.1 Discrépance maximale de la moyenne (MMD)

A.3.1.1 Fondements de la MMD

Definition 22. ***Discrépance maximale de la moyenne*** [23]: Soit $\mathcal{F}$ une classe de fonctions f: $\mathcal{X} \to \mathbb{R}$ où p et q sont deux densités de probabilité. La discrépance maximale de la moyenne est définie par :

$$MMD[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} (\mathbf{E}_p[f(x)] - \mathbf{E}_q[f(y)]) \tag{A.4}$$

Theorem 5. [20] Supposons que ($\mathcal{X}$,d) est un espace métrique et soit p,q deux densités de probabilité définies sur $\mathcal{X}$. On a $p = q$ si et seulement si $\mathbf{E}_p[f(x)] = \mathbf{E}_q[f(y)]$ quelle que soit $f \in C(\mathcal{X})$, où $C(\mathcal{X})$ est l'espace des fonctions continues bornées x, y sont des variables aléatoires tirées de p and q respectivement.

A.3.1.2 Calcul de la MMD dans un RKHS

Theorem 6. [77], [73]$MMD[\mathcal{F}, p, q] = 0$ si et seulement si $p = q$ quand $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$ pourvu que $\mathcal{H}$ soit universel (ce qui signifie que $k(x, .)$ est continu pour tout x et que le RKHS induit par k est dense dans $C(\mathcal{X})$)).

Il peut alors être montré que :

$$MMD[\mathcal{F}, p, q] = \|\mu_p - \mu_q\|_{\mathcal{H}}$$

A.3.1.3 Utilisation de la MMD en classification

Classification basée sur le carré de la MMD

Nous considérons le carré de la MMD en tant que mesure de (dis)similitude entre groupes. Bien évidemment, cette grandeur sera évaluée dans un RKHS, en espérant que la transformation non linéaire associée au noyau induira une meilleure classification. A ce niveau, nous pouvons également espérer que le meilleur résultat de classification sera obtenu lorsque le carré de la MMD est maximisé.

Cependant, nous pouvons observer que lorsque le carré de la MMD est maximal, les partitions obtenues sont très éloignées de celles fournies lors de nos simulations. par ailleurs, nous constatons que l'algorithme conduit à des classes dont les effectifs sont très disproportionnés.

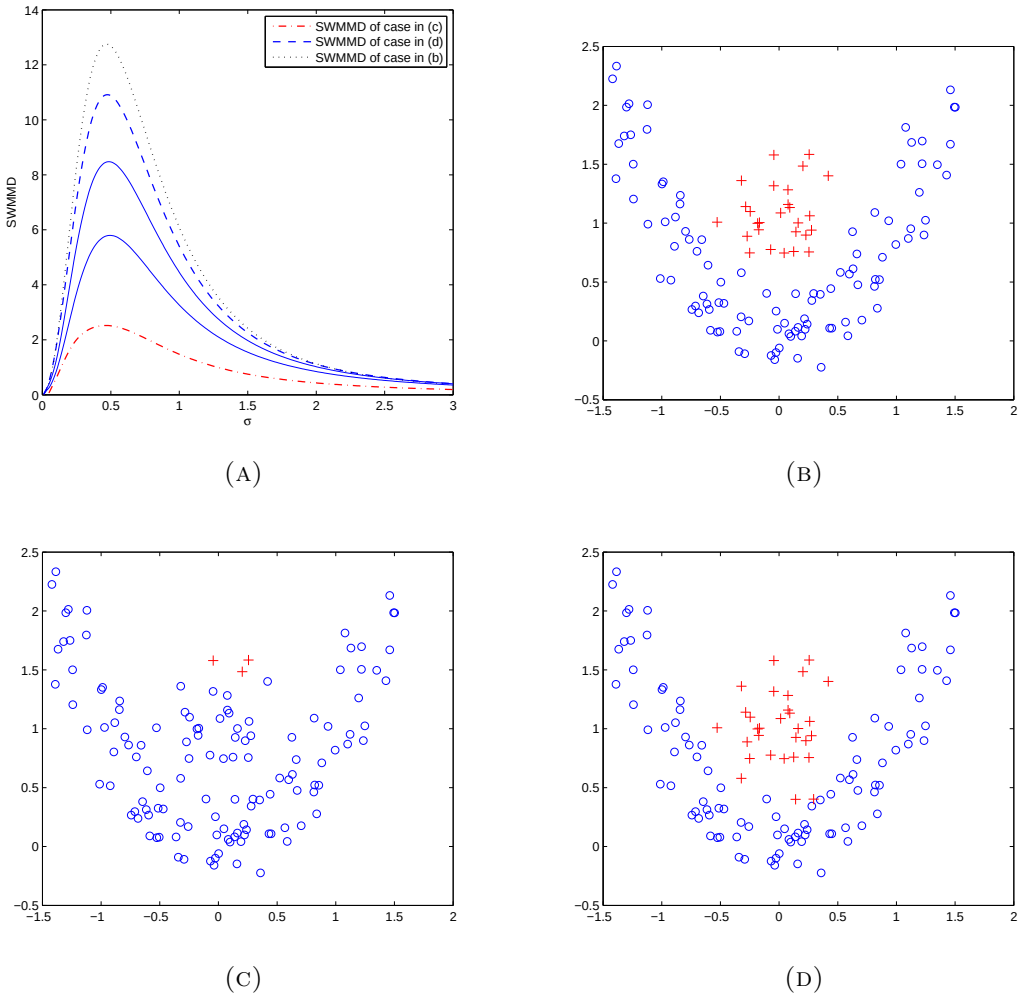
Classification reposant sur une MMD pondérée

Pour obtenir des solutions plus adéquates, nous avons pris en compte le nombre d'observations dans le critère de MMD. Pour ce faire, nous pondérons le carré de la MMD par $mn/(m+n)$ où m et n sont les effectifs des classes obtenues, l'objectif étant de rechercher une partition plus cohérente avec les résultats attendus.

$$WMMD^2[\mathcal{F}, p, q] = \frac{mn}{m+n}\|\mu_p - \mu_q\|^2_{\mathcal{H}} \tag{A.5}$$

Ce critère est appelé *Weighted Squared MMD (WSMMD)*. Les expériences réalisées avec les mêmes ensembles de données que précédemment (Figure A.3) montrent que la maximisation de la WSMMD correspond effectivement à notre hypothèse : la classification obtenue est satisfaisante lorsque WSMMD la atteint son maximum. Nous observons que la WSMMD favorise le regroupement équilibré, contrairement au carré de la MMD.

Conclusion La WSMMD est équivalente au critère Ward [82] évalué dans un RKHS. Le critère de Ward est une mesure de dissimilitude fréquemment utilisée dans la classification ascendante hiérarchique. Nous proposons d'étudier plus en profondeur cette approche.



(A) (B)

(C) (D)

FIGURE A.3: WSMMD évaluée. a: WSMMD en fonction du paramètre de noyau σ. b: Meilleure partition obtenue (correspondant à une valeur élevée de la WSMMD). c: Partition obtenue lorsque le maximum de la WSMMD (par rapport à σ) est minimal parmi les partitions testées. d: Cas intermédiaire (classes initiales)

A.3.2 Classification Ascendante Hiérarchique (CAH)

A.3.2.1 Introduction

Nous décrivons brièvement ici le principe d'un algorithme de classification ascendante hiérarchique :

- *Etape 1* Chaque élément constitue initialement un groupe.
- *Etape 2* On recherche les 2 groupes les plus semblables au sens d'une mesure de similitude.

- *Etape 3* On agrège ces deux groupes et on met à jour la mesure de similitude entre les groupes
- *Etape 4* On retourne à l'étape 2 jusqu'à ce qu'il ne reste qu'un seul groupe.

A.3.2.2 Mesures de (dis)similitude

La clé du processus précédent est de définir et de mettre à jour la mesure de similitude / dissimilitude. Il existe plusieurs stratégies et nous nous focalisons sur le critère de Ward. Comme nous l'avons mentionné précédemment, il existe une relation étroite entre le critère de Ward et la WSMMD.

Le critère de Ward s'écrit:
$d^2(r,s) = n_r n_s \frac{\|\bar{x}_r - \bar{x}_s\|_2^2}{(n_r+n_s)}$, $\bar{x}_r = \frac{1}{n_r}\sum_{i=1}^{n_r} x_{ri}$

La WSMMD s'exprime selon :

$$WSMMD[\mathcal{F}, p, q] = \frac{mn}{m+n}\|\mu_p - \mu_q\|_{\mathcal{H}}^2 \tag{A.6}$$

où m et n sont les nombres d'échantillons dans les classes p et q, respectivement. Elle est équivalente au critère de Ward classiquement utilisé en classification ascendante hiérarchique.

A.3.3 Classification ascendante hiérarchique à noyau

Les méthodes de classification ascendante hiérarchique ont été largement utilisés dans de nombreux domaines. Il existe toutefois assez peu de travaux relatifs à leur version *kernelisée*.

$\phi : X \to F$ effectue une transformation de l'espace original vers un espace (*feature space*) de grande dimension. L'évaluation des produits scalaires dans l'espace d'arrivée peuvent être effectués par une fonction noyau dans l'espace initial, sans avoir à préciser explicitement la transformation Φ. Le calcul des distances euclidiennes dans F utilise cette idée.

$$d^2(\Phi(x_i), \Phi(x_j)) = \|\Phi(x_i) - \Phi(x_j)\|^2 \tag{A.7}$$

$$= k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j) \tag{A.8}$$

A.4 Choix du paramètre du noyau

A.4.1 Introduction

L'objectif de cette partie est de proposer des méthodes permettant de sélectionner automatiquement le paramètre du noyau utilisé. Nous considérons, dans un premier temps, le critère d'alignement.

A.4.2 Alignment

L'alignement mesure la ressemblance entre une matrice de Gram et la matrice de Gram "idéale" ou encore entre deux matrices de Gram. Il repose sur le calcul du produit de Frobenius entre deux matrices de Gram.

A.4.2.1 Alignement entre deux noyaux

Definition 23. L'*alignement empirique* A entre deux noyaux (matrices de Gram) K_1 K_2 est défini par :

$$A = \frac{< K_1, K_2 >_F}{\sqrt{< K_1, K_1 >_F}\sqrt{< K_2, K_2 >_F}} \tag{A.9}$$

où
$< X, Y >_F = \sum_{ij} x_{ij} y_{ij} = Tr(X'Y)$. représente le produit Frobenius entre les matrices X et Y.

A.4.2.2 Alignement entre un noyau et une tâche

Pour les tâches de classification, la "meilleure" fonction Phi doit fournir en sortie l'étiquette du point considéré. La "meilleure" matrice de Gram, en ce sens, est donc yy' où y est une fonction de σ.

Definition 24. L'alignement entre K and yy' est donné par :

$$A = \frac{< K, yy^{'} >_F}{\sqrt{< K, K >_F < yy', yy^{'} >_F}} \tag{A.10}$$

Une meilleure valeur de l'alignement (obtenue par un choix adapté du paramètre du noyau) conduit donc à l'identification d'un meilleur noyau pour la tâche considérée.

A.4.3 Sélection du paramètre du noyau par alignement

Conformément à ce qui a été dit plus haut, nous allons retenir la valeur du paramètre du noyau σ qui maximise l'alignement entre K et yy'. Pour chaque valeur considérée de σ, nous évaluons la sortie de notre algorithme de classification (les étiquettes des données) et nous calculons l'alignement entre la matrice de Gram et yy'.

$$A(\sigma) = \frac{< K, y(\sigma)y(\sigma)^{'} >_F}{\sqrt{< K, K >_F < y(\sigma)y(\sigma)', y(\sigma)y(\sigma)^{'} >_F}} \tag{A.11}$$

La valeur optimale de σ est celle pour laquelle l'alignement est maximum. Nous avons donc :

$$\sigma^* = \arg\max_{\sigma} A(\sigma) = \arg\max_{\sigma} \frac{< K, y(\sigma)y(\sigma)^{'} >_F}{\sqrt{< K, K >_F < y(\sigma)y(\sigma)', y(\sigma)y(\sigma)^{'} >_F}} \tag{A.12}$$

A.4.3.1 Autres critères pour la sélection du paramètre du noyau

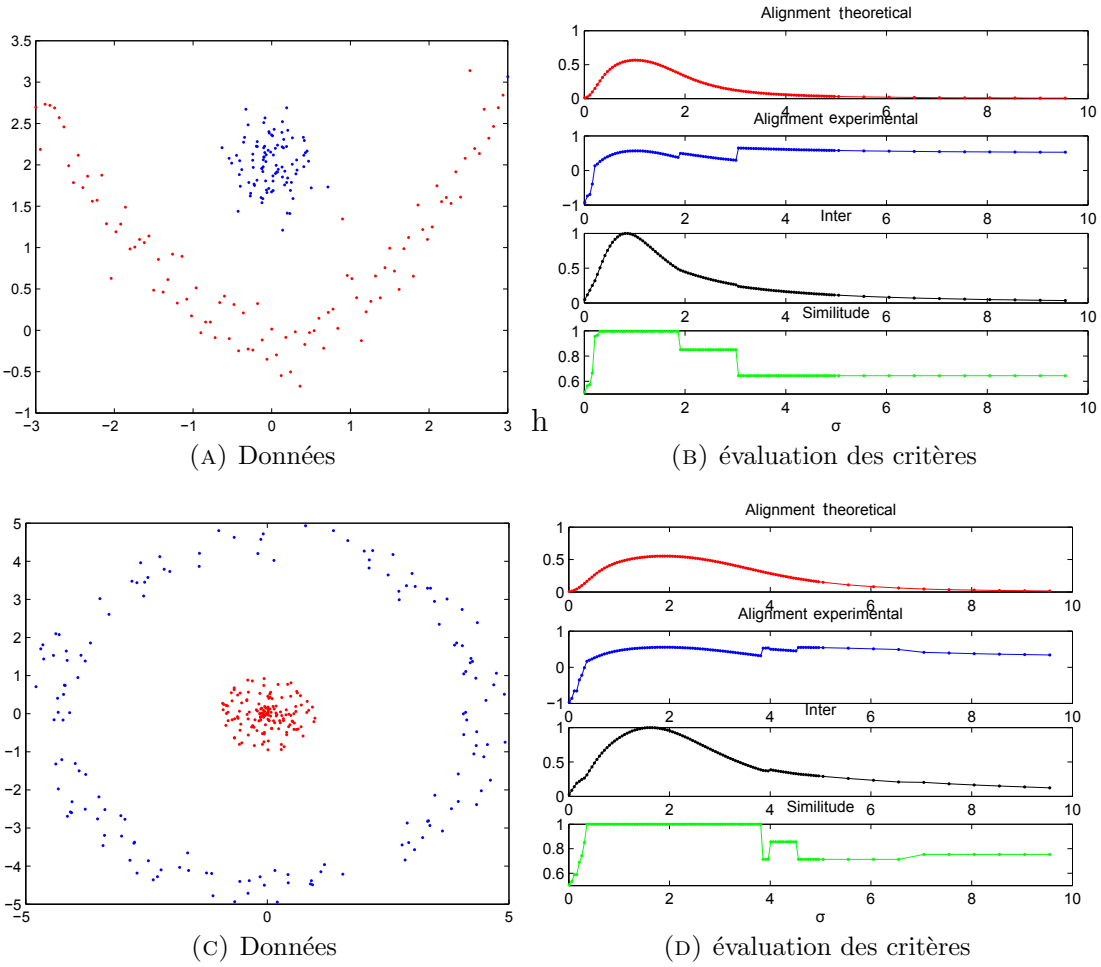
Dans cette partie, nous proposons plusieurs critères pour effectuer la sélection du paramètre du noyau. Nous les comparons avec l'*alignement* [46].

- Alignment (theoretical)
- Alignment (experimental)
- dispersion interclasse
- Similitude

Tous ces critères sont calculés dans l'espace des caractéristiques $\mathcal{F}$.

A.4.3.2 Comparaison des critères

Dans cette expérience, nous évaluaons les critères précédents sur 3 ensembles (simulés) couramment utilisés de données binaires. Nous faisons varier le paramètre du noyau σ dans l'intervalle $[0.01, 10]$.



(A) Données (B) évaluation des critères

(C) Données (D) évaluation des critères

FIGURE A.4: Comparaison des critères de sélection du paramètre du noyau

En étudiant les résultats, une valeur élevée de l'alignement théorique (qui est inaccessible hors simulation car il nécessite la connaissance des véritables étiquettes) correspond toujours à un bon résultat de clustering (une valeur élevée de la similitude), mais l'inverse n'est pas nécessairement vrai.

En outre, la courbe de l'alignement expérimental n'est pas toujours continue.

Conclusion

Les deux analyses théoriques et expérimentales indiquent que l'alignement n'est pas un bon indicateur pour mesurer l'accord entre une fonction noyau et une tâche d'apprentissage. Les étiquettes ne sont pas des fonctions continues de σ et, en conséquence, l'alignement expérimental n'est pas continu. De plus, il ne présente pas une corrélation forte avec la similitude qui mesure le taux d'accord entre les étiquettes fournies en simulation et celles obtenues.

A.4.4 Alignement centré

Nous allons voir que le caractère discontinu de l'alignement expérimental peut être quasiment éliminé en considérant sa version centrée.

A.4.4.1 Centrage d'une matrice de Gram

Lemma 1 Pour toute matrice de Gram K, sa version centrée K_c s'exprime :

$$K_c = (\mathbf{I} - \frac{1}{n}\mathbf{1}_n)K(\mathbf{I} - \frac{1}{n}\mathbf{1}_n) \tag{A.13}$$

où 1_n est la matrice $n \times n$ dont tous les éléments sont égaux à 1.
L'opération de centrage peut être réalisée de différentes manières en utilisant le lemme suivant :

Lemma 2 Soit 2 matrices de Gram K et K', on a :

$$\langle K_c, K'_c \rangle_F = \langle K, K'_c \rangle_F = \langle K_c, K' \rangle_F \tag{A.14}$$

La preuve de ces assertions est donnée dans [13].

A.4.4.2 Alignement centré

L'alignement centré entre deux matrices de Gram est défini par :

$$A_c = \frac{< K_c, K'_c >_F}{\sqrt{< K_c, K_c >_F}\sqrt{< K'_c, K'_c >_F}} \tag{A.15}$$

Comme l'*alignement*, l'*alignement* centré peut mesurer l'adéquation entre une fonction noyau et une tâche. L'*alignement empirique centré* entre une tâche et

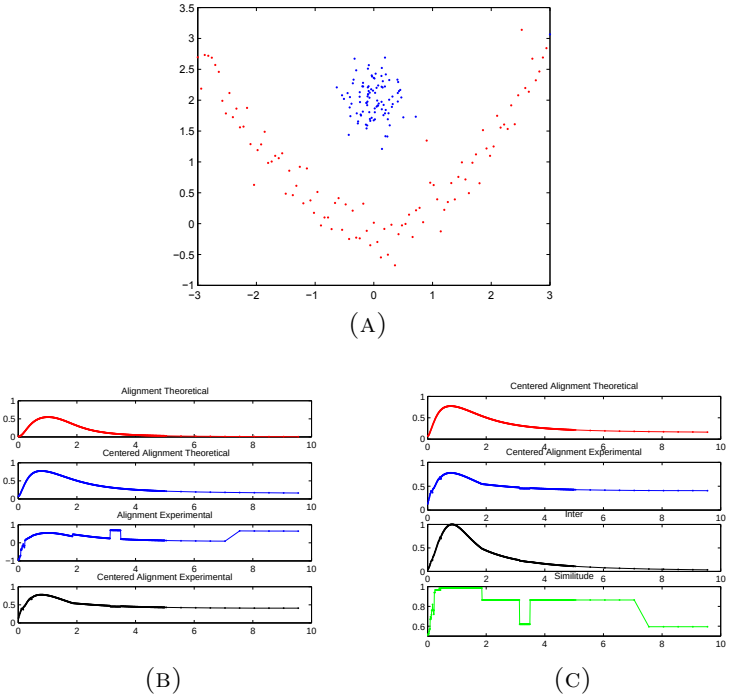
la matrice de Gram associée est donné par :

$$A_c = \frac{< K_c, (yy')_c >_F}{\sqrt{< K_c, K_c >_F}\sqrt{< (yy')_c, (yy')_c >_F}} \tag{A.16}$$

A.4.5 Simulations pour le choix du paramètre du noyau

Les résultats de simulation sont donnés dans la Fig. A.5.

Nous avons d'abord comparé l'alignement et l'alignement centré théorique (en utilisant les véritables étiquettes connues en simulation) et expérimental (qui utilise les étiquettes fournies par la classification) en Fig.(B). En regardant le résultat, la courbe d'alignement centré est plus *lisse* que celle de l'alignement. Il n'y a pas de saut pour l'alignement centré, ce qui rend la sélection du paramètre du noyau réalisable.



(A) (B) (C)

FIGURE A.5: a.Données b. Alignement et alignement centré c.Evaluation des différents critères

Conclusion

L'alignement centré et la distance interclasse donnent des performances satisfaisantes. Dans ce qui suit, nous nous concentrons sur l'alignement centré et la distance interclasse comme critères de sélection du paramètre de noyau.

A.4.6 Etude de la sensibilité des critères à l'effectif des classes

Nous considérons le cas simple où une fraction $\alpha \in [0, 1]$ de tous les points est centrée sur $(-1, 0)$ avec une dispersion de $\rho^2 I_2$. Leur étiquette est -1. Les points restants sont centrés sur $(1, 0)$ avec une dispersion de $\rho^2 I_2$. Leur étiquette est $+1$. Plusieurs exemples de distribution de données sont représentés en Fig. A.6 pour différentes valeurs de α et ρ.

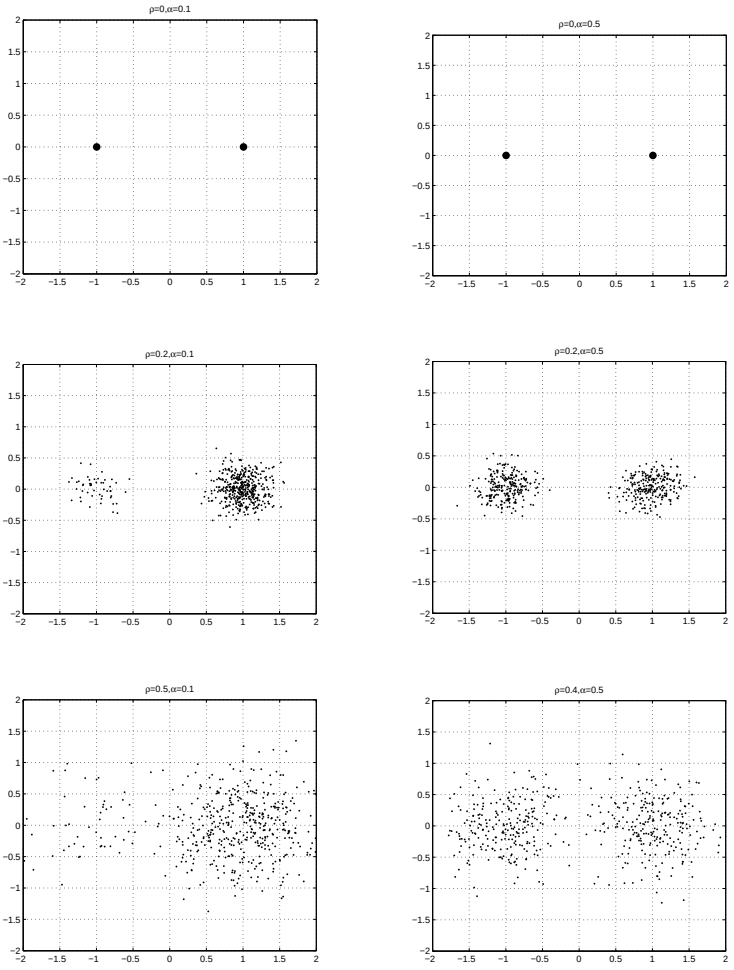


FIGURE A.6: Données correspondant à différentes valeurs de α et ρ

Dans notre travail, nous considérons le noyau gaussien. Les résultats sont présentés en Fig. A.8.

Conclusion

Le critère qui semble le plus robuste par rapport à la proportion des classes est l'alignment centré.

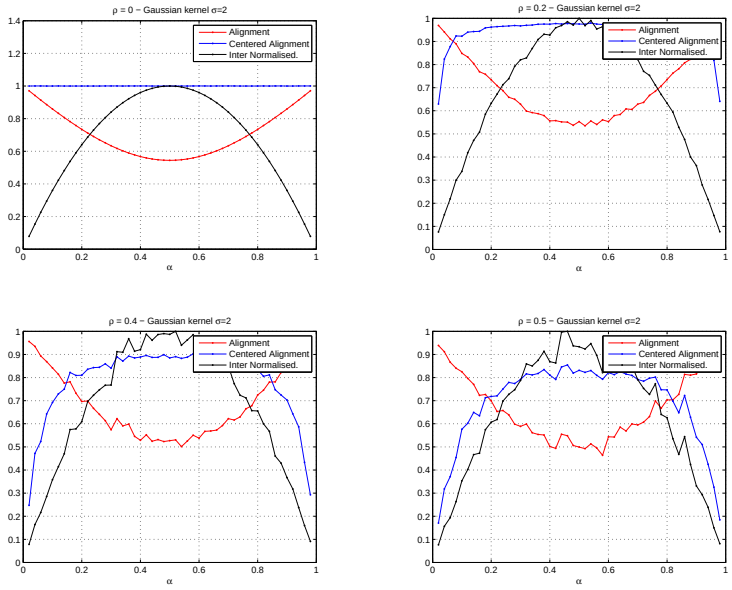


FIGURE A.7: Etude de la sensibilité des critères

A.5 Détermination du nombre de classes

A.5.1 Introduction

Déterminer le nombre de classes est l'un des problèmes les plus difficiles en classification non supervisée. Les *gap statistics* proposées par Tibshirani peuvent être utilisées pour faire face à ce problème.

A.5.2 Gap Statistics

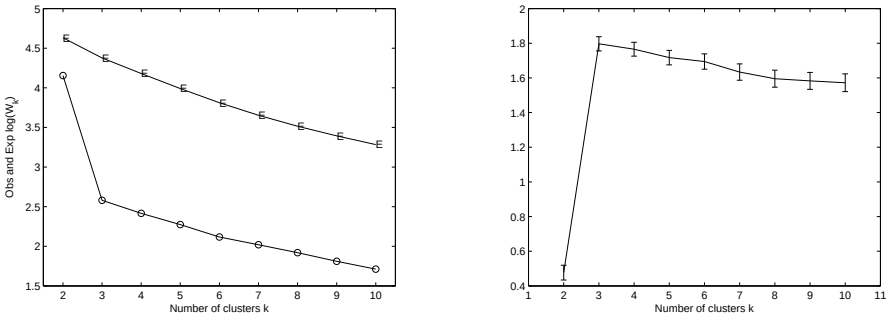
L'idée principale de la *gap statistic* est de comparer un certain critère avec sa valeur moyenne que nous pourrions observer sous l'hypothèse d'un groupe unique (H_0).

Selon Tibshirani :

$$Gap(k) = E_n\{log(W_k/H_0)\} - log(W_k) \tag{A.17}$$

Ici W_k est la dispersion au sein de la classe et $E_n\{log(W_/H_0)\}$ désigne l'espérance de l'estimation de $log(W_k)$ sous l'hypothèse nulle H_0.

Les figures A.8 montrent les résultats obtenus par KHAC.



(A) Graphes de $E_n\{log(W_k/H_0)\}$ (courbe du haut) et de $log(W_k)$ (courbe du bas)

(B) Gap statistic en fonction du nombre de classes

FIGURE A.8: Exemple de détermination du nombre de classes par KHAC.

Selon Tibshirani, le nombre de classes $\hat{k}$ est la plus petite valeur de k telle que :

$$Gap(k) \geq Gap(k+1) - s_{k+1} \tag{A.18}$$

A.5.3 Choix de la distribution sous H_0

L'importance du choix d'une distribution de référence appropriée sous H_0 a été étudiée dans [28] par Gordon.

Les distributions de référence le plus souvent considérées sont les suivantes :

- une distribution uniforme sur l'hyperparallélépipède couvert par l'ensemble des données observées.
- une distribution uniforme sur l'hyperparallélépipède aligné avec les composantes principales de l'ensemble des données.

La première méthode présente l'avantage de la simplicité tandis que la seconde est plus précise en termes de cohérence avec les données, car elle prend en compte la forme de la distribution de celles-ci.

A.5.4 Nouveaux critères pour la détermination du nombre de classes

L'idée principale des *gap statistics* était de comparer un critère obtenu sur les données avec la moyenne que l'on peut observer sur une distribution monoclasse de référence. Inspiré par cette idée, nous proposons d'étendre cette idée à d'autres critères en vue de mieux estimer le nombre de groupes en présence.

Ces critères sont:

- Gap Statistic modifiée
- Gap sur l'alignement centré
- Gap sur la différence entre deux niveaux consécutifs du dendrogramme
- Gap pondéré sur la différence entre deux niveaux consécutifs du dendrogramme

A.5.5 Détermination du nombre de classes à l'aide de différentes statistiques

Notre algorithme est représenté en Fig. A.9.

A.5.5.1 Simulations

Nous avons généré 6 ensembles de données différents afin de comparer les critères proposés avec ceux de Tibshirani:

Les résultats obtenus indiquent clairement que le Gap sur la différence entre deux niveaux consécutifs du dendrogramme surpasse les autres critères (y compris la statistique initiale de Tibshirani) dans presque tous les cas.

En outre, le Gap sur la différence entre deux niveaux consécutifs du dendrogramme montre d'excellentes performances sur des ensembles de données qui ne peuvent être classés en utilisant la HAC alors que facilement traités par KHAC.

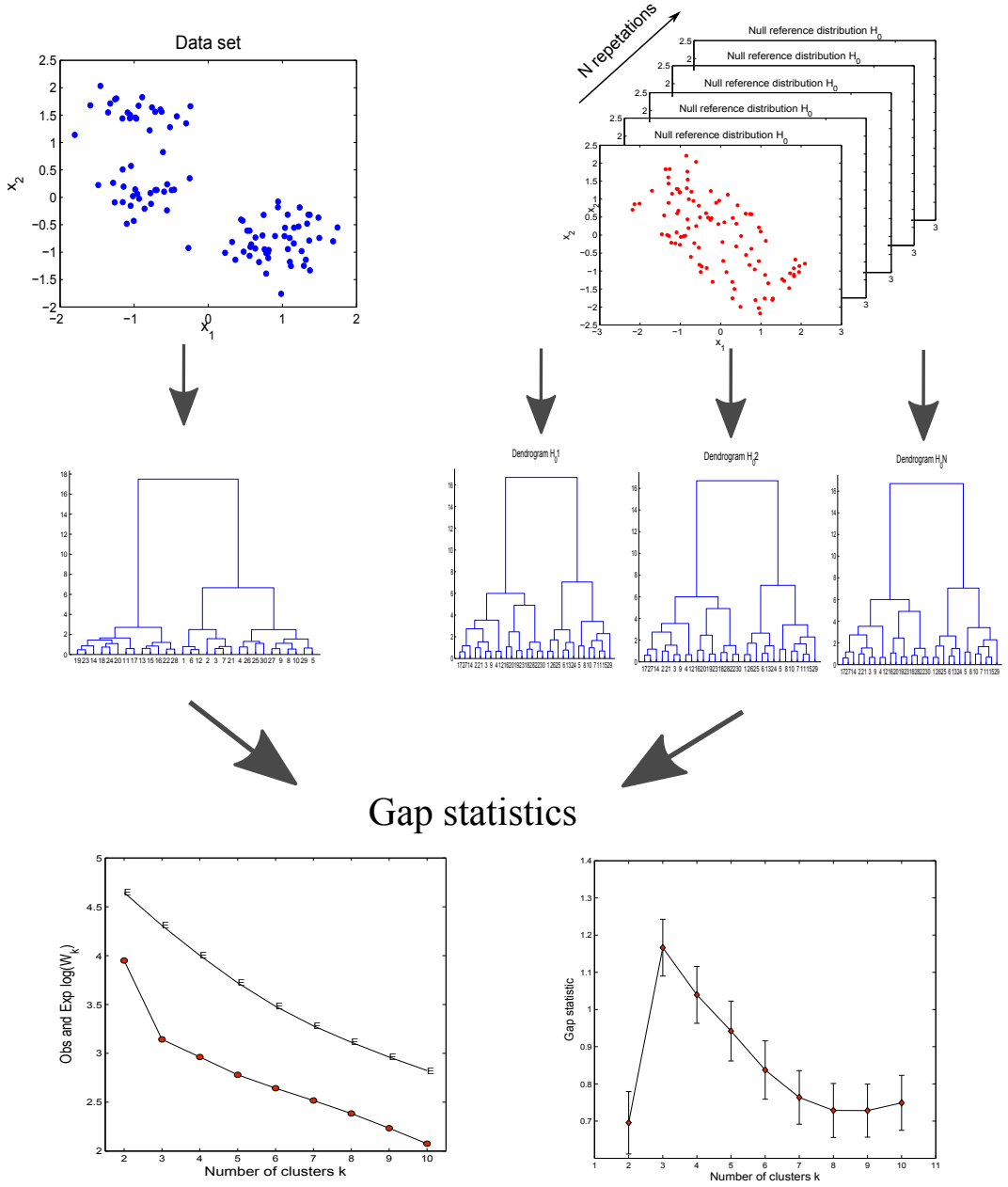
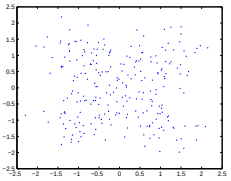


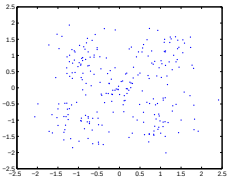
FIGURE A.9: Procédure pour determiner le nombre de classes.

A.5.6 Conclusion

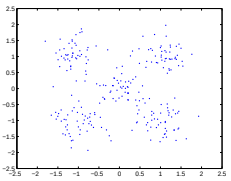
Selon toutes les expériences que nous avons faites, nous avons observé que la Gap statistic initiale de Tibshirani n'est pas toujours efficace pour l'estimation du nombre exact de groupes. Rechercher la valeur maximale du critère au lieu de sélectionner la valeur proposée dans l'équation A.18, semble être une alternative satisfaisante. Parmi les critères pris en compte, le Gap sur la différence entre deux niveaux consécutifs du dendrogramme semble être le plus robuste et potentiellement l'un des moins sensibles au choix de la distribution de référence sous



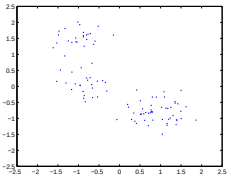
(A) Gap sur la différence entre deux niveaux consécutifs du dendrogramme, 30/50



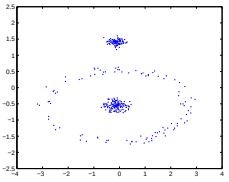
(B) Gap sur la différence entre deux niveaux consécutifs du dendrogramme, 48/50



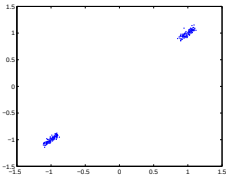
(C) Gap sur la différence entre deux niveaux consécutifs du dendrogramme, 50/50



(D) Gap pondéré sur la différence entre deux niveaux consécutifs du dendrogramme, 50/50; Gap sur la différence entre deux niveaux consécutifs du dendrogramme, 45/50



(E) Delta Level Gap, 50/50



(F) Gap Statistic;Gap sur la différence entre deux niveaux consécutifs du dendrogramme;Gap sur l'alignement centré,50/50. PCA:Gap sur la différence entre deux niveaux consécutifs du dendrogramme,Gap sur l'alignement centré,50/50

H_O.

A.6 Classification itérative par CAH à noyau

Dans ce chapitre, nous proposons un algorithme de KHAC itératif et essayons de trouver une méthode permettant de déterminer automatiquement la valeur optimale du paramètre du noyau gaussien ainsi que le nombre de groupes. Notre travail commence par l'évaluation de l'efficacité de la KHAC itérative lorsque le nombre de groupes est connu. Lorque celui-ci est inconnu, nous proposons plusieurs critères permettant de le déterminer.

A.6.1 KHAC itérative avec un nombre de classes connu

A.6.1.1 Présentation de l'algorithme itératif

Dans cette section, nous proposons un algorithme de classification basé sur la classification ascendante hiérarchique dont l'idée de base est d'explorer le dendrogramme obtenu en partant de la classe finale constituée de l'ensemble des données.

L'algorithme est présenté dans l'algorithme 2.

Algorithm 2: Principe de l'algorithme de classification par KHAC

Etape 1. Appliquer la KHAC pour obtenir le dendrogramme initial
Etape 2. Déterminer le paramètre optimal σ^* et le nombre optimal de classes K^* selon une des statistiques présentées précédemment
Etape 3. Calculer la Gap Statistic ([79]) pour prendre (ou non) la décision de partitionner les données en K^* groupes
while *at* ***Etape 3*** *il est décidé de partitionner et tant que le nombre de classes K (connu) n'est pas atteint, alors* **do**
    Couper le dendrogramme pour obtenir les K^* groupes et mettre ces groupes dans la liste des tâches *en attente*;
    Sélectionner une *tâche en attente* et effectuer sur celle-ci les **Etape 2** et **Etape 3**;
end

A.6.1.2 Illustration de la procédure itérative

L'ensemble du processus de classification automatique est représenté en Fig. A.11, Fig. A.12, Fig. A.13, Fig. A.14.

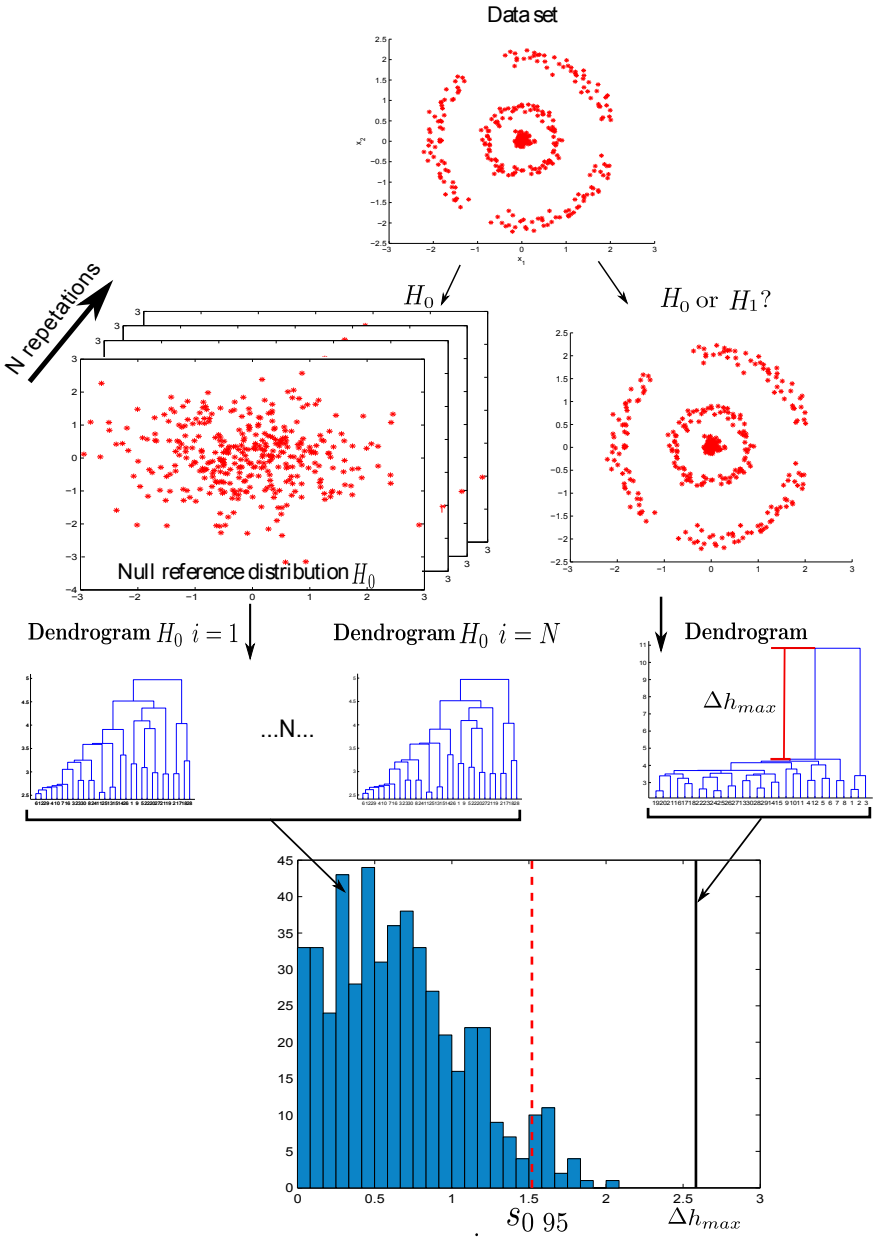


FIGURE A.10: Illustration de la décision entre l'hypothèse d'un groupe unique (H_0) et celle de groupes multiples (H_1) inspirée par les Gap Statistics

A.6.1.3 Résultats complémentaires

Pour valider notre méthode, nous présentons les résultats obtenus sur plusieurs ensembles de données synthétiques représentés en Fig. A.15. Considérant le nombre de groupes correspondant à chacun de ces exemples, les résultats sont satisfaisants.

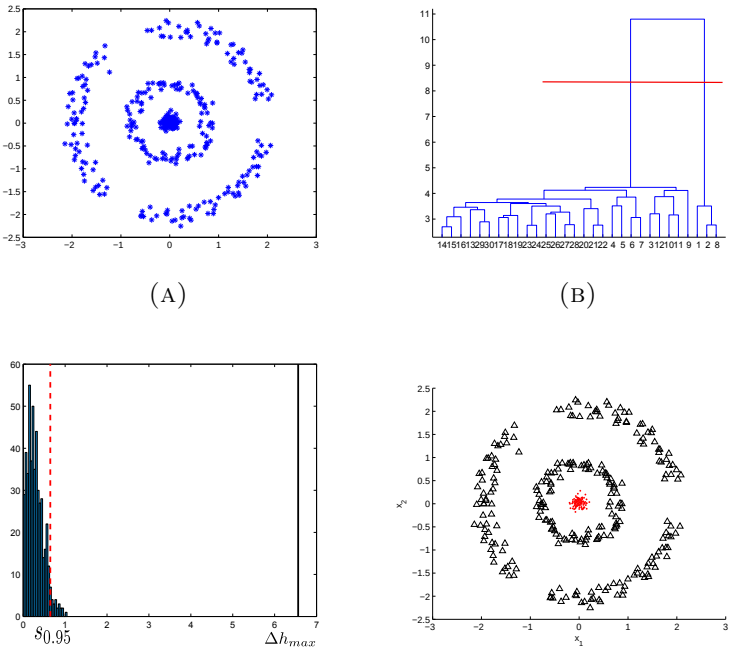


FIGURE A.11: Illustration sur un jeu de données synthétiques (1 itération). a: *tâche en attente* (jeu de données initial). b: Dendrogramme obtenu $\sigma^* = 0.22$,$K^* = 2$. c: Histogramme de la distribution de Δh_{max} sous H_0, nous avons ici $\Delta h_{max} > s_{0.95}$. d: résultat après la 1 itération.

A.6.2 KHAC itérative avec un nombre de classes inconnu

Dans la section précédente, nous avons proposé un algorithme basé sur la KHAC lorsque le nombre de groupes est connu en avance. L'idée principale est de couper le dendrogramme itérativement. Le point clé est de savoir arrêter le partitionnement itératif lorsque le "bon" nombre de classes est obtenu. Lorsque ce nombre de groupes est inconnu et nous avons pu l'observer dans l'exemple précédent, Δh_{max} peut ne pas être la meilleure statistique pour décider de couper le dendrogramme. Dans cette section, nous considérons l'alignement centré et la dispersion interclasse.

A.6.2.1 Alignement centré et dispersion interclasse

Décision de segmenter basée sur la sensibilité du critère étudié à l'éloignement des groupes

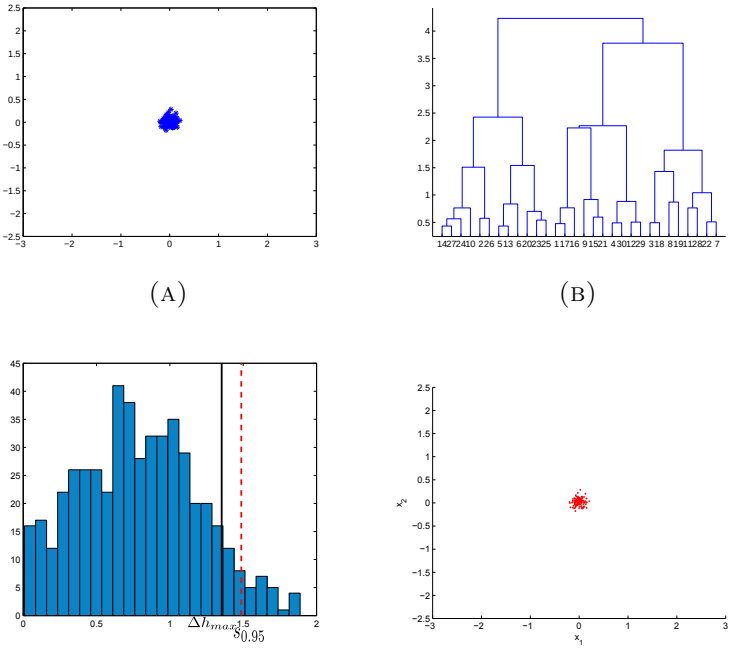


FIGURE A.12: Illustration sur un jeu de données synthétiques (2^{nd} itération). a: *tâche en attente*. b: Dendrogramme obtenu $\sigma^* = 1.52$, $K^* = 3$. c: Histogramme de la distribution de Δh_{max} sous H_0, nous avons ici $\Delta h_{max} < s_{0.95}$. d: résultat après la 2^{nde} itération.

L'idée est de mesurer la variation du critère considéré avec le changement de la distance d (Fig. A.16) entre deux groupes dans l'espace des caractéristiques. Dans le cas de deux groupes, soit $m_{i\mathcal{H}}$, $i = 1, 2$ le centre de chaque groupe, $m_{0\mathcal{H}}$ le centre de masse global de l'ensemble des données dans $\mathcal{H}$. Pour une distance interclasse d donnée, nous calculons un critère, noté $C(d)$ et ensuite nous augmentons légèrement la distance entre les deux groupes dans $mathcal H$ de la manière suivante :

$$\phi(x)_\alpha = \phi(x) + \alpha(m_{i\mathcal{H}} - m_{0\mathcal{H}}), \alpha > 0, \alpha \to 0 \tag{A.19}$$

Ensuite, nous évaluons la variation du critère par rapport à α :

$$S = \frac{\partial C(d,\alpha)}{\partial \alpha} \simeq \frac{C(d,\alpha) - C(d,0)}{\alpha} \mid \alpha \to 0 \tag{A.20}$$

L'idée est que la dérivée partielle estimée du critère sera faible si les deux groupes sont bien séparés (d est relativement importante). Au contraire, si sa variation est

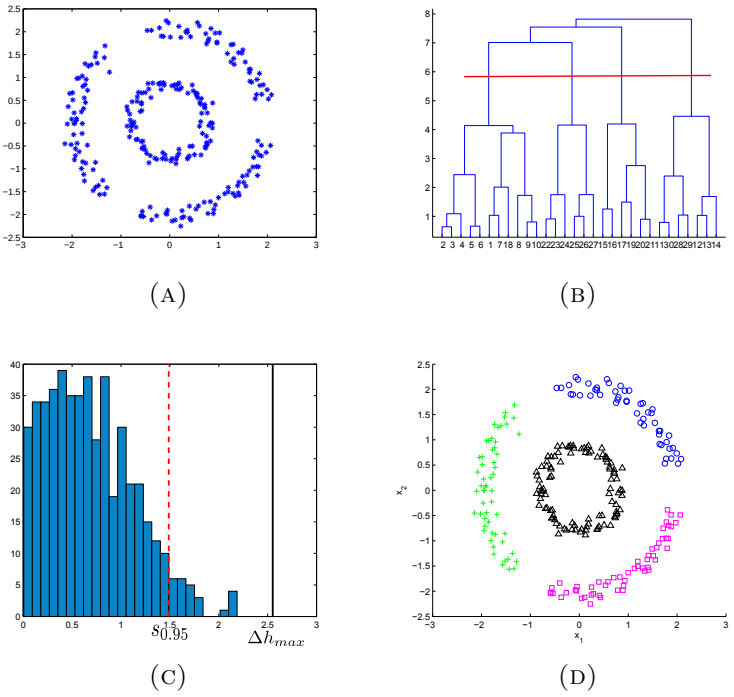


FIGURE A.13: Illustration sur un jeu de données synthétiques (3^{rd} itération). a: *tâche en attente*. b: Dendrogramme obtenu $\sigma^* = 1.12$, $K^* = 4$. c: Histogramme de la distribution de Δh_{max} sous H_0, nous avons ici $\Delta h_{max} > s_{0.95}$. d: résultat après la 3^{rd} iteration.

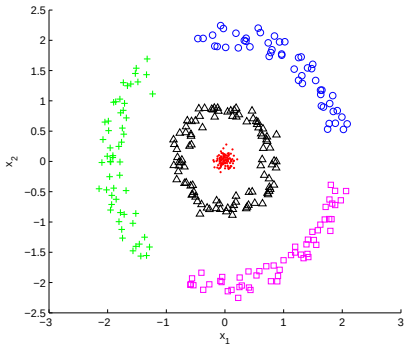


FIGURE A.14: Illustration sur un jeu de données synthétiques: Résultat final

importante, nous considérons qu'il n'y a qu'un seul cluster (H_0) (ce qui correspond à une petite valeur de d).

Nous présentons maintenant les résultats (Fig. A.16) obtenus à partir de simulations en sélectionnant différentes valeurs de d pour évaluer la capacité de notre test à détecter H_1.

Le résultat obtenu avec l'alignement centré confirme notre hypothèse.

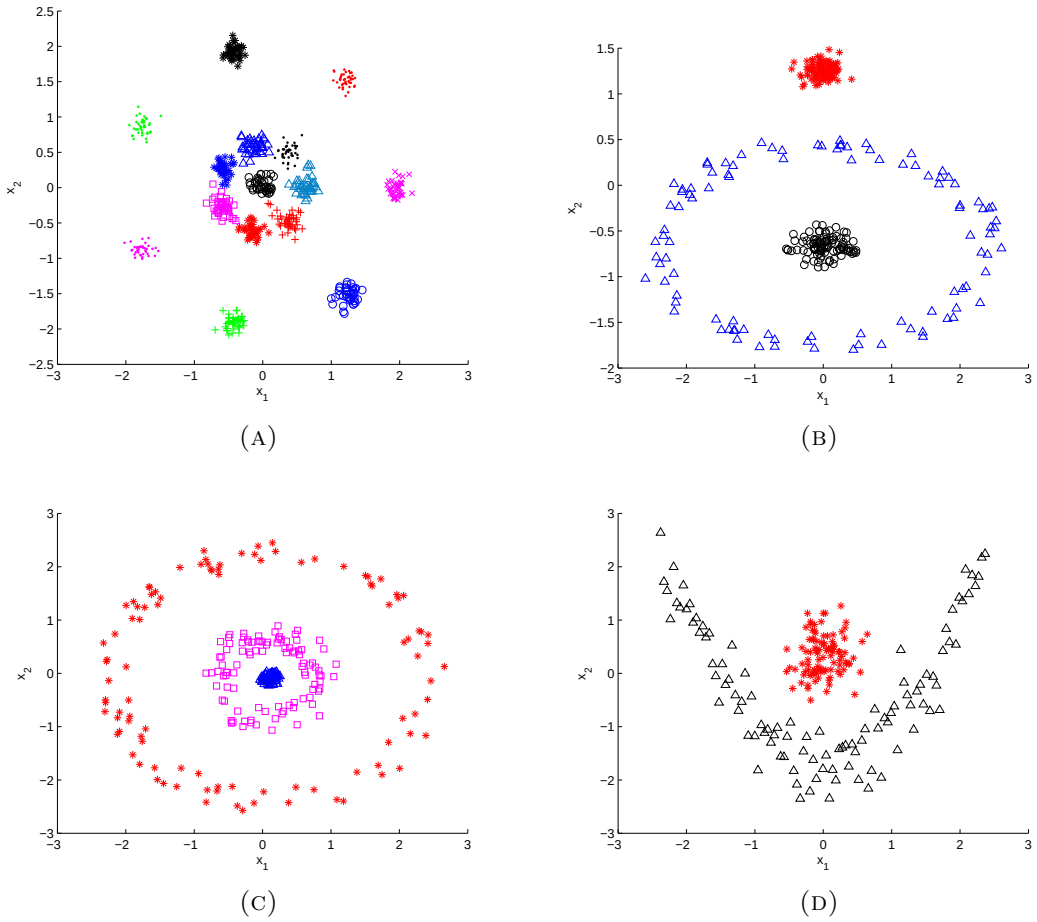


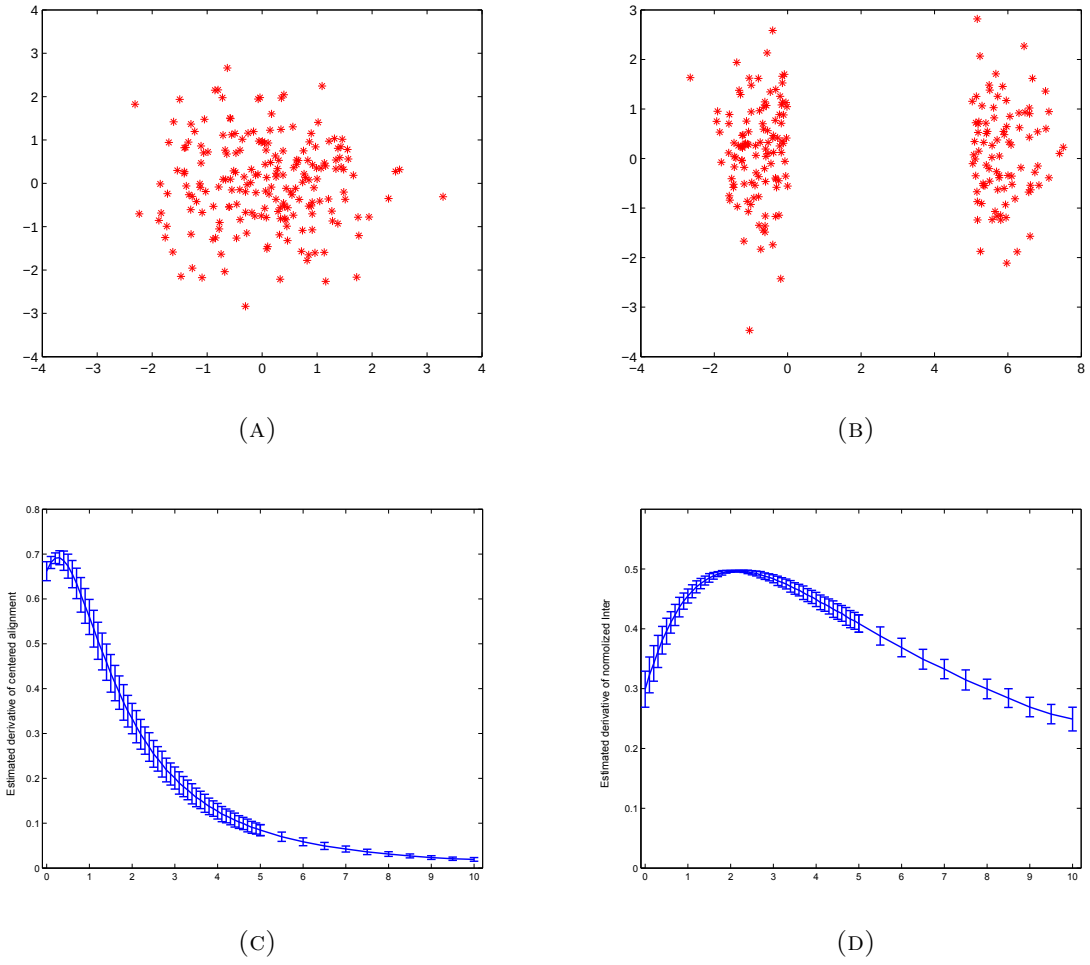
FIGURE A.15: Résultats sur des jeux de données synthétiques

Malheureusement ces critères dépendent de la distribution des données, ce qui signifie que nous ne pouvons pas trouver de seuil unique pour toute distribution des données.

A.7 Conclusion

La classification non supervisée, problème mal posé, fait l'objet d'une attention soutenue de la part de la communauté scientifique depuis de nombreuses années. Un grand nombre d'algorithmes de classification ont été proposés, mais il n'existe pas de méthode universelle qui soit capable de traiter tous les problèmes. La plupart des méthodes de classification efficaces sont très dépendantes du problème considéré.

Dans ce document, nous nous sommes concentrés sur la classification hiérarchique à noyau en utilisant le critère de Ward. Pour aboutir à cette méthode, nous avons



(A) (B)

(C) (D)

FIGURE A.16: Illustration de la variation des critères en fonction de d. A : Jeu de données avec ($d = 0$). B : Jeux de données associé à une valeur importante de d ($d = 4$). C: Variation de l'alignement centré en fonction de d (et écart type). D: Variation de la dispersion interclasse en fonction de d (et écart type).

introduit la discrépance maximale de la moyenne (MMD). La MMD a tout d'abord été proposée pour mesurer une différence entre deux distributions. Celle-ci peut facilement être évaluée dans un RKHS. Nous avons fait le lien entre la MMD (évaluée dans un RKHS) et la version *kernelisée* du critère de Ward.

Nous avons proposé d'utiliser l'alignement centré et la dispersion interclasse pour sélectionner les paramètres du noyau et les Gap statistics pour déterminer le nombre de classes. Nous avons ensuite introduit une méthode pour résoudre les deux problèmes précédents en même temps et à identifier automatiquement des groupes. L'alignement Centré et la dispersion interclasse fournissent des résultats satisfaisants pour la sélection du paramètre du noyau.

Une des statistiques proposées fournit des résultat satisfaisants pour la détermination du nombre de classes. Toutefois, elle n'est pas assez robuste. Notre algorithme itératif de classification montre un potentiel intéressants, mais nécessite de poursuivre les travaux engagés. Une technique plus fiable pour déterminer le nombre de classes aidera beaucoup dans l'amélioration de la performance de notre méthode.

Bibliography

[1] Aizerman, A., Braverman, E. M., and Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837.

[2] Alpert, C. J. and Kahng, A. B. (1994). Multi-way partitioning via spacefilling curves and dynamic programming. In *Proceedings of the 31st annual Design Automation Conference*, pages 652–657. ACM.

[3] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404.

[4] Bankes, S. C., Lempert, R. J., and Popper, S. W. (2001). Computer-assisted reasoning. *Computing in Science & Engineering*, 3(2):71–77.

[5] Baraldi, A. and Blonda, P. (1999). A survey of fuzzy clustering algorithms for pattern recognition. i. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(6):778–785.

[6] Barbará, D. and Jajodia, S. (2002). *Applications of data mining in computer security*, volume 6. Springer.

[7] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[8] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

[9] Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

[10] Chen, C.-h., Pau, L.-F., and Wang, P. S.-p. (2010). *Handbook of pattern recognition and computer vision*, volume 27. World Scientific.

[11] Chung, K.-M., Kao, W.-C., Sun, C.-L., Wang, L.-L., and Lin, C.-J. (2003). Radius margin bounds for support vector machines with the rbf kernel. *Neural Computation*, 15(11):2643–2681.

[12] Cortes, C., Mohri, M., and Rostamizadeh, A. (2009). Learning non-linear combinations of kernels. In *Advances in neural information processing systems*, pages 396–404.

[13] Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828.

[14] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[15] Day, W. H. and Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

[16] Debnath, R. and Takahashi, H. (2004). An efficient method for tuning kernel parameter of the support vector machine. In *Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on*, volume 2, pages 1023–1028. IEEE.

[17] Driver, H. E. and Kroeber, A. L. (1932). *Quantitative expression of cultural relationships*. University of California Press.

[18] Dubes, R. and Jain, A. K. (1976). Clustering techniques: the user's dilemma. *Pattern Recognition*, 8(4):247–260.

[19] Duda, R. O., Hart, P. E., et al. (1973). *Pattern classification and scene analysis*, volume 3. Wiley New York.

[20] Dudley, R. M. (1984). A course on empirical processes. In *Ecole d'été de Probabilités de Saint-Flour XII-1982*, pages 1–142. Springer.

[21] Fasulo, D. (1999). An analysis of recent work on clustering algorithms. *Department of Computer Science & Engineering, University of Washington*.

[22] Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. (2008). A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190.

[23] Fortet, R. and Mourier, E. (1953). Convergence de la répartition empirique vers la réparation théorique. *Ann. Scient. École Norm. Sup.*, pages 266–285.

[24] Fraley, C. and Raftery, A. E. (1998). How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588.

[25] Gionis, V., Strzelecka, H., Veber, M., Kormann, R., and Zuppiroli, L. (1986). Liquid crystalline organic conductors: studies in crystalline and mesomorphic phase. *Molecular Crystals and Liquid Crystals*, 137(1):365–372.

[26] Girolami, M. (2002). Mercer kernel-based clustering in feature space. *Neural Networks, IEEE Transactions on*, 13(3):780–784.

[27] Girosi, F., Jones, M. B., and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269.

[28] Gordon, A. D. (1996). Null models in cluster validation. In *From data to knowledge*, pages 32–44. Springer.

[29] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773.

[30] Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic learning theory*, pages 63–77. Springer.

[31] Guha, S., Rastogi, R., and Shim, K. (1998). Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM.

[32] Guha, S., Rastogi, R., and Shim, K. (1999). Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE.

[33] Hansen, P. and Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical programming*, 79(1-3):191–215.

[34] Harman, H. H. (1960). Modern factor analysis.

[35] Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition.

[36] Haussler, D. (1999). Convolution kernels on discrete structures. Technical report, Citeseer.

[37] Herbrich, R. (2002). *Learning kernel classifiers*. MIT Press, Cambridge.

[38] Honeine, P., Noumir, Z., and Richard, C. (2013). Multiclass classification machines with the complexity of a single binary classifier. *Signal Processing*, 93(5):1013–1026.

[39] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

[40] Jain, A. K., Duin, R. P., and Mao, J. (2000). Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37.

[41] Jain, A. K. and Flynn, P. J. (1996). *Image segmentation using clustering*. IEEE Press, Piscataway, NJ.

[42] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

[43] Jambu, M. and Lebeaux, M.-O. (1983). *Cluster analysis and data analysis*. North-Holland Amsterdam etc.

[44] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

[45] Kandola, J., Shawe-Taylor, J., and Cristianini, N. (2002a). Optimizing kernel alignment over combinations of kernel.

[46] Kandola, J., Shawe-Taylor, J., and Cristianini, N. (2002b). Optimizing kernel alignment over combinations of kernel.

[47] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. Wiley-Interscience.

[48] Kim, D.-W., Lee, K. Y., Lee, D., and Lee, K. H. (2005). Evaluation of the performance of clustering algorithms in kernel-induced feature space. *Pattern Recognition*, 38(4):607–611.

[49] Kim, S.-J., Magnani, A., and Boyd, S. (2006). Optimal kernel selection in kernel fisher discriminant analysis. In *Proceedings of the 23rd international conference on Machine learning*, pages 465–472. ACM.

[50] Krzanowski, W. J. and Lai, Y. (1988). A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, pages 23–34.

[51] Lance, G. N. and Williams, W. T. (1967). A general theory of classificatory sorting strategies ii. clustering systems. *The computer journal*, 10(3):271–277.

[52] Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72.

[53] Li, N., Lefebvre, N., and Lengellé, R. (2014). Kernel hierarchical agglomerative clustering. comparison of different gap statistics to estimate the number of clusters. In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods, ICPRAM 2014*.

[54] Ling, R. F. (1973). A probability theory of cluster analysis. *Journal of the American Statistical Association*, 68(341):159–164.

[55] Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

[56] Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. Siam.

[57] Micchelli, C. A. (1984). *Interpolation of scattered data: distance matrices and conditionally positive definite functions*. Springer.

[58] Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *The Journal of Machine Learning Research*, 7:2651–2667.

[59] Milligan, G. W. (1979). Ultrametric hierarchical clustering algorithms. *Psychometrika*, 44(3):343–346.

[60] Milligan, G. W. and Cooper, M. C. (1985a). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

[61] Milligan, G. W. and Cooper, M. C. (1985b). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

[62] Min, J. H. and Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert systems with applications*, 28(4):603–614.

[63] Muller, K.-R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B. (2001). An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*, 12(2):181–201.

[64] Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359.

[65] Príncipe, J. C., Liu, W., and Haykin, S. (2011). *Kernel Adaptive Filtering: A Comprehensive Introduction*, volume 57. John Wiley & Sons.

[66] Rohlf, F. J. and Fisher, D. R. (1968). Tests for hierarchical structure in random data sets. *Systematic Zoology*, pages 407–412.

[67] Saitoh, S. (1988). *Theory of reproducing kernels and its applications*, volume 189. Longman.

[68] Salton, G. (1991). Developments in automatic text retrieval. *Science*, 253(5023):974.

[69] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

[70] Serfling, R. J. (2009). *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons.

[71] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[72] Shawe-Taylor, N. and Kandola, A. (2002). On kernel target alignment. *Advances in neural information processing systems*, 14:367.

[73] Smola, A. (2006). Maximum mean discrepancy. In *13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006: Proceedings*.

[74] Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer.

[75] Sneath, P. H., Sokal, R. R., et al. (1973). *Numerical taxonomy. The principles and practice of numerical classification.*

[76] Spath, H. (1980). *Cluster analysis algorithms for data reduction and classification of objects*. Ellis Horwood, Ltd. Chichester, England.

[77] Steinwart, I. (2002). On the influence of the kernel on the consistency of support vector machines. *The Journal of Machine Learning Research*, 2:67–93.

[78] Sugar, C. A. and James, G. M. (2003). Finding the number of clusters in a dataset. *Journal of the American Statistical Association*, 98(463).

[79] 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 (Statistical Methodology)*, 63(2):411–423.

[80] Vapnik, V. (2000). *The nature of statistical learning theory*. springer.

[81] Veenman, C. J., Reinders, M. J. T., and Backer, E. (2002). A maximum variance cluster algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(9):1273–1280.

[82] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

[83] Wu, K.-P. and Wang, S.-D. (2009). Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, 42(5):710–717.

[84] Wu, S. and Chow, T. W. (2004). Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density. *Pattern Recognition*, 37(2):175–188.

[85] Xu, R., Wunsch, D., et al. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

[86] Zeng, G. and Dubes, R. C. (1985). A comparison of tests for randomness. *Pattern recognition*, 18(2):191–198.

[87] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM.

[88] Zhou, S. K. and Chellappa, R. (2006). From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel hilbert space. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(6):917–929.

Na LI

Doctorat : Optimisation et Sûreté des Systèmes

Année 2015

Maximum Mean Discrepancy et critère de Ward dans un RKHS. Application à la classification hiérarchique à noyau

La classification non supervisée consiste à regrouper des objets afin de former des groupes homogènes au sens d'une mesure de similitude. C'est un outil utile pour explorer la structure d'un ensemble de données non étiquetées. Par ailleurs, les méthodes à noyau, introduites initialement dans le cadre supervisé, ont démontré leur intérêt par leur capacité à réaliser des traitements non linéaires des données en limitant la complexité algorithmique. En effet, elles permettent de transformer un problème non linéaire en un problème linéaire dans un espace de plus grande dimension. Dans ce travail, nous proposons un algorithme de classification hiérarchique ascendante utilisant le formalisme des méthodes à noyau. Nous avons tout d'abord recherché des mesures de similitude entre des distributions de probabilité aisément calculables à l'aide de noyaux. Parmi celles-ci, la *maximum mean discrepancy* a retenu notre attention. Afin de pallier les limites inhérentes à son usage, nous avons proposé une modification qui conduit au critère de Ward, bien connu en classification hiérarchique. Nous avons enfin proposé un algorithme itératif de clustering reposant sur la classification hiérarchique à noyau et permettant d'optimiser le noyau et de déterminer le nombre de classes en présence.

Mots clés : classification automatique (statistique) - reconnaissance des formes (informatique) - apprentissage automatique - tests d'hypothèses (statistique).

MMD and Ward Criterion in a RKHS. Application to Kernel based Hierarchical Agglomerative Clustering

Clustering, as a useful tool for unsupervised classification, is the task of grouping objects according to some measured or perceived characteristics of them and it has owned great success in exploring the hidden structure of unlabeled data sets. Kernel-based clustering algorithms have shown great prominence. They provide competitive performance compared with conventional methods owing to their ability of transforming nonlinear problem into linear ones in a higher dimensional feature space. In this work, we propose a Kernel-based Hierarchical Agglomerative Clustering algorithms (KHAC) using Ward's criterion. Our method is induced by a recently arisen criterion called Maximum Mean Discrepancy (MMD). This criterion has firstly been proposed to measure difference between different distributions and can easily be embedded into a RKHS. Close relationships have been proved between MMD and Ward's criterion. In our KHAC method, selection of the kernel parameter and determination of the number of clusters have been studied, which provide satisfactory performance. Finally an iterative KHAC algorithm is proposed which aims at determining the optimal kernel parameter, giving a meaningful number of clusters and partitioning the data set automatically.

Keywords: cluster analysis - pattern recognition systems - machine learning - statistical hypothesis testing.

Thèse réalisée en partenariat entre :

CHINA SCHOLARSHIP COUNCIL

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"