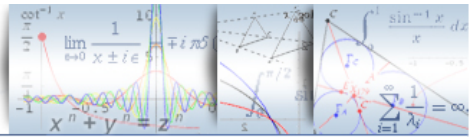
# Um breve estudo sobre o algoritmo *K-means*

Diogo Henriques Freitas Nunes

DEPARTAMENTO DE MATEMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



# Um breve estudo sobre o algoritmo *K-means*

Diogo Henriques Freitas Nunes

Dissertação para a obtenção do Grau de **Mestre em Matemática**

Área de Especialização em **Estatística, Otimização e Matemática Financeira**

**Júri**

| | |
|---|---|
| **Presidente:** | João Luís Cardoso Soares |
| **Orientador:** | José Luis Esteves dos Santos |
| **Vogal:** | Gonçalo Nuno Travassos Borges Alves Pena |

**05 de Julho de 2016**

# Resumo

Este trabalho teve como objetivo aprofundar o conhecimento de uma ferramenta de data mining conhecida como o clustering. Optou-se por direcionar este estudo para um algoritmo de clustering clássico, o *K-means*, e prova-se a sua convergência. Complementarmente apresentam-se outros dois algoritmos, o *Expetation-Maximization* e o *Kernel K-means*, e realizam-se testes de desempenho entre os três. No final aplicaram-se estes algoritmos a vários tipos de problemas nomeadamente no campo da segmentação de imagens.

**Palavras Chave:** Clustering,Clusters,*K-means*,*Expectation-Maximization*,*Kernel K-means*

# Abstract

The objective of this work was to understand a data mining technique known as clustering. Particulary, this work was focused on a classic algorithm known as *K-means* and demonstrating its convergence. To understand its potential two more classic algorithms were studied, named *Expetation-Maximization* and *Kernel K-means* and their respective performances were tested and compared. Finally, these algorithms were applied on image segmentation.

**Keywords:** Clustering,Clusters,*K-means*,*Expectation-Maximization*,*Kernel K-means*

# Agradecimentos

*Quero agradecer ao meu orientador Professor José Luís Esteves dos Santos, pela disponibilidade, atenção dispensada e paciência.*

*À minha mãe, devo-lhe tudo.*

*Aos meus colegas de curso, aos meus amigos pelo seu apoio e momentos de folia.*

*E à Leonor, pela ajuda , compreensão e carinho.*

# Conteúdo

# Capítulo 1

# Introdução

Nos dias de hoje muitas aplicações fornecem e/ou guardam grandes quantidades de dados para analisar. Essa informação, quando analisada individualmente, não é útil sendo necessária uma análise conjunta de todos os dados. Neste contexto, surge a necessidade de criar ferramentas de análise de dados que permitam lidar com esse novo paradigma da informação. Um dos principais meios encontrados para lidar com esta quantidade enorme de informação foi a sua categorização em grupos ou clusters, isto é, o clustering.

O problema do clustering é definido através da função de semelhança pela qual o clustering divide um conjunto de informação em pequenos grupos. Essa função aglomera os dados que têm semelhanças entre si, ou seja, dentro de cada cluster os dados são semelhantes e entre os clusters os dados são diferentes. A análise de clusters pode ser abordada através de diferentes perspectivas [7] e neste trabalho focam-se três delas: a distância ao cluster, a probabilidade de pertencer ao cluster e a coesão entre clusters.

A análise de clusters foi primeiramente impulsionada pela monografia de Sokal and Sneath [9]. Entre 1963 e 1975 surgem os primeiros algoritmos de clusters, como por exemplo, o *K-means* e o *Hierárquico* [8]. Com o aumento de dimensão dos conjuntos de dados, aliado a uma maior quantidade e diversidade de atributos, surge por volta de 2002 a necessidade de evoluir os métodos de clusters [10]. Estes novos métodos de aplicacão de clusters vão influenciar as áreas de estatística, computação e machine learning [7]. O clustering como instrumento de segmentação e indexação de texto é mencionado em 2010 por *Aggarwal* e *Zhai* [11]. A sua aplicabilidade no reconhecimento de entidades num texto permite organizar documentos e classificá-los por temas. Atualmente, o clustering surge aplicado na análise de dados que são gerados constantemente [12] com o objetivo de retirar informação de dados contínuos online não estacionários. O clustering é útil em diversas áreas como, por exemplo, na comparação de hábitos de compras de amostras de uma população de modo a

determinar o grupo ao qual se deverá aplicar uma campanha de saldos.

Esta dissertação divide-se em quatro capítulos. No primeiro capítulo, dedicado à introdução, pretende-se situar o tema do clustering ao longo da história e salientar sua aplicabilidade. No segundo capítulo, são apresentados três algoritmos que a literatura considera base fundamental para uma introdução ao estudo do clustering. Estes algoritmos são o K-means, o Expectation-Maximization e o Kernel K-means, sendo apresentada uma descrição de cada algoritmo e o estudo da sua ordem de complexidade. No caso do K-means é feito ainda um estudo sobre a sua convergência. Ainda neste capítulo faz-se referência a outros algoritmos importantes tais como o *Hierárquico*, o *DBSCAN* e o *K Nearest Neighbors*. No terceiro capítulo apresentam-se os resultados computacionais que incluem os diferentes tipos de avaliações efetuadas em vários casos de testes, estando um deles associado à segmentação de imagens. Por último, no quarto capítulo, apresenta-se uma conclusão onde se discutem os três algoritmos estudados em relação aos resultados obtidos.

# Capítulo 2

# Algoritmos

Em primeiro lugar serão apresentados alguns conceitos básicos sobre o clustering e a sua notação. Um clustering $C = \{C_1, C_2, \ldots, C_k\}$ consiste numa partição de um conjunto de dados em subconjuntos $C_i$ (chamados clusters), onde os elementos de cada cluster apresentam alguma semelhança entre si e diferem dos elementos que se encontram nos outros clusters.

Considere-se um conjunto de dados $\boldsymbol{D} = \{\boldsymbol{x}\}_{i=i}^{n}$ com $n$ pontos num espaço de dimensão $d$. Seja $C = \{C_1, C_2, \ldots, C_k\}$ um clustering do conjunto de dados. Para cada cluster $C_i$ existe um ponto $\boldsymbol{z}_i$ que o representa e que é designado por centroide. Neste trabalho iremos defini-lo como a média de todos os elementos dentro do cluster, isto é:

$$\boldsymbol{z}_i = \frac{1}{n_i} \sum_{x_j \in C_i} \boldsymbol{x}_j$$

onde $n_i$ é o número de elementos de cada cluster.

Na aplicação dos algoritmos de clusters é necessário avaliar iterativamente a qualidade do clustering e, para tal, recorreu-se a uma função de semelhança. Esta denomina-se de soma dos erros quadrados (Sum of Squared Errors) $SSE$ e mede a semelhança de cada elemento através da sua distância aos centroides:

$$SSE(C) = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} D(\boldsymbol{x}_j, \boldsymbol{z}_i)$$

Esta função também pode ser vista como uma avaliação de dispersão dos elementos dentro de um cluster.

Após estas notações e conceitos serão abordados 3 tipos de algoritmos clássicos: o *K-means*, o *Expectation–Maximization* e o *Kernel K-means*.

## 2.1. K-means

### 2.1.1. Descrição do algoritmo

O K-means aplica uma abordagem gananciosa para encontrar o clustering que minimiza o $SSE$, convergindo para uma solução local em vez de um clustering ótimo global. Este algoritmo começa por distribuir aleatoriamente os pontos do conjunto $\boldsymbol{D}$ em $k$ clusters e calcula os centroides através da média dos pontos do cluster $C_i$. De seguida aplicam-se duas fases iterativamente: a atribuição de clusters e a atualização dos centroides.

Na atribuição de clusters, cada ponto $\boldsymbol{x}_j \in \boldsymbol{D}$ é associado ao cluster que tem o centroide $\boldsymbol{z}_i$ mais próximo do ponto, isto é, $\boldsymbol{x}_j$ é atribuído ao cluster $C_{j^*}$ quando:

$$j^* = \underset{i=1,\ldots,k}{\operatorname{argmin}} \left\{ \|\boldsymbol{x}_j - \boldsymbol{z}_i\|_2^2 \right\}.$$

De seguida atualizam-se os centroides através da média de todos os pontos que se encontram no cluster $C_i$. Estas duas fases são realizadas iterativamente até alcançarem um mínimo local, isto é, o *K-means* converge se os centroides não mudarem após uma iteração. Note-se que se pode impor ao algoritmo uma condição de paragem, como por exemplo, $\sum_{i=1}^{k} \|\boldsymbol{z}_i^t - \boldsymbol{z}_i^{t-1}\|_2^2 \leq \epsilon$, onde $\epsilon > 0$ é o limite de convergência, $t$ é a iteração corrente e $\boldsymbol{z}_i^t$ é o centroide do cluster $C_i$ na iteração $t$.

O pseudo-código deste método encontra-se no apêndice B algoritmo 1. Este é um exemplo de atribuição única, o que significa que cada ponto só pertence a um cluster. No apêndice A, na figura A.1, encontra-se um exemplo do resultado obtido com este algoritmo.

### 2.1.2. Convergência

Nesta secção, estudamos a convergência do *K-means* e, para o efeito, apresenta-se um problema de otimização de onde o algoritmo *K-means* pode ser deduzido: seja $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ um número finito de elementos pertencentes a $\boldsymbol{D}$, que pretendem ser agrupados em $k$ subconjuntos, de acordo com a semelhança entre eles.

Uma formulação deste problema é a seguinte:

$$
\begin{aligned}
(P): \min \qquad & f(\boldsymbol{W}, \boldsymbol{Z}) = \sum_{i=1}^{k} \sum_{j=1}^{n} \boldsymbol{w}_{ij} D(\boldsymbol{x}_j, \boldsymbol{z}_i) \\
\text{sujeito a} \quad & \sum_{i=1}^{k} \boldsymbol{w}_{ij} = 1, j = 1, 2, \ldots, n \\
& \boldsymbol{w}_{ij} = 1 \text{ ou } \boldsymbol{w}_{ij} = 0, \quad i = 1, 2, \ldots, k \quad j = 1, 2, \ldots, n
\end{aligned}
$$

tal que $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k] \in R^{d \times k}$, $\boldsymbol{z}_i$ é o centroide do cluster $i$ e $\boldsymbol{W} = [w_{ij}]$ sendo uma matriz real de dimensão $k \times n$. Se $w_{ij}$ tomar o valor 1 significa que o ponto $\boldsymbol{x}_j$ pertence ao cluster $i$ e verifica-se o contrário quando o seu valor é 0. A função $D$ mede a semelhança entre o ponto $\boldsymbol{x}_j$ e o centroide $\boldsymbol{z}_i$ que, no *K-means*, será da seguinte forma:

$$D(\boldsymbol{x}_j, \boldsymbol{z}_i) = \|\boldsymbol{x}_j - \boldsymbol{z}_i\|_2^2 = \sum_{l=1}^{d} (\boldsymbol{x}_{lj} - \boldsymbol{z}_{li})^2.$$

De seguida, por uma questão de simplificação, considera-se a representação da linha $i$ de $\boldsymbol{W}$ como $\boldsymbol{W}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]$ e a influência do cluster $i$ (elementos e centroide) na função $f$ como $f_i(\boldsymbol{W}_i, \boldsymbol{Z}_i) = \sum_{j=1}^{n} w_{ij} D(\boldsymbol{x}_j, \boldsymbol{z}_i)$ e, por consequência, $f(\boldsymbol{W}, \boldsymbol{Z}) = \sum_{i=1}^{k} f_i(\boldsymbol{W}_i, \boldsymbol{Z}_i)$.

Para deduzir o algoritmo *K-means* a partir do problema (P) introduzem-se ainda mais algumas notações que simplificam a exposição. Dado $\hat{\boldsymbol{W}}$ fixo define-se $f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}) = f(\hat{\boldsymbol{W}}, \boldsymbol{Z})$ que goza da seguinte propriedade:

**Propriedade 1** *Se $D(\boldsymbol{x}, \boldsymbol{Z})$ é uma função convexa em $\boldsymbol{Z}$, então $f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z})$ também o é.*

**Prova** Considerem-se duas matrizes $\boldsymbol{Z}^1, \boldsymbol{Z}^2 \in \mathbb{R}^{d \times k}$ *e* $\gamma \in [0, 1]$. Uma vez que a função $D(\boldsymbol{x}, \boldsymbol{Z})$ é convexa então verifica a seguinte condição:

$$D(\boldsymbol{x}, \gamma \boldsymbol{Z}^1 + (1 - \gamma)\boldsymbol{Z}^2) \leq \gamma D(\boldsymbol{x}, \boldsymbol{Z}^1) + (1 - \gamma) D(\boldsymbol{x}, \boldsymbol{Z}^2). \tag{2.1}$$

Deste modo

$$f_{\hat{\boldsymbol{W}}}(\gamma \boldsymbol{Z}^1 + (1 - \gamma)\boldsymbol{Z}^2) = \sum_{i=1}^{k} \sum_{j=1}^{n} w_{ij} D(\boldsymbol{x}_j, \gamma \boldsymbol{Z}_i^1 + (1 - \gamma)\boldsymbol{Z}_i^2)$$

Usando a equação (2.1), deduz-se o seguinte:

$$\begin{aligned} f_{\hat{\boldsymbol{W}}}(\gamma \boldsymbol{Z}^1 + (1-\gamma)\boldsymbol{Z}^2) &\leq \sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij}\left(\gamma D(\boldsymbol{x}_j, \boldsymbol{Z}_i^1) + (1-\gamma)D(\boldsymbol{x}_j, \boldsymbol{Z}_i^2)\right) \\ &= \gamma \sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij} D(\boldsymbol{x}_j, \boldsymbol{Z}_i^1) + (1-\gamma)\sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij} D(\boldsymbol{x}_j, \boldsymbol{Z}_i^2) \\ &= \gamma f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}^1) + (1-\gamma) f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}^2) \end{aligned}$$

Conclui-se assim a prova. □

Esta secção foca-se na convergência do algoritmo *K-means*, que usa a função de semelhança $D(\boldsymbol{x}, \boldsymbol{z}_i) = \|\boldsymbol{x} - \boldsymbol{z}_i\|_2^2$, por isso interessa provar que esta é convexa. Sejam $\boldsymbol{Z}^1, \boldsymbol{Z}^2 \in \mathbb{R}^{d\times k}$ duas matrizes e $\gamma \in [0,1]$ , então a função verifica o seguinte:

$$\begin{aligned} D(\boldsymbol{x}, (\gamma \boldsymbol{Z}^1 + (1-\gamma)\boldsymbol{Z}^2)) &= \|\boldsymbol{x} - \gamma \boldsymbol{Z}^1 - (1-\gamma)\boldsymbol{Z}^2\|_2^2 \\ &= \|\gamma \boldsymbol{x} + (1-\gamma)\boldsymbol{x} - \boldsymbol{Z}^1 - (1-\gamma)\boldsymbol{Z}^2\|_2^2 \\ &= \|\gamma(\boldsymbol{x} - \boldsymbol{Z}^1) + (1-\gamma)(\boldsymbol{x} - \boldsymbol{Z}^2)\|_2^2. \end{aligned}$$

Aplicando a desigualdade triangular à última expressão, obtém-se o seguinte:

$$\begin{aligned} \|\gamma(\boldsymbol{x} - \boldsymbol{Z}^1) + (1-\gamma)(\boldsymbol{x} - \boldsymbol{Z}^2)\|_2^2 &\leq \|\gamma(\boldsymbol{x} - \boldsymbol{Z}^1)\|_2^2 + \|(1-\gamma)(\boldsymbol{x} - \boldsymbol{Z}^2)\|_2^2 \\ &= \gamma D(\boldsymbol{x}, \boldsymbol{Z}^1) + (1-\gamma) D(\boldsymbol{x}, \boldsymbol{Z}^2), \end{aligned}$$

uma vez que $\gamma \geq 0$ e $1 - \gamma \geq 0$.

Conclui-se assim que a função de semelhança usada pelo *K-means* é convexa. De seguida, para auxiliar raciocínios futuros, serão apresentadas outras notações. Dado $\hat{\boldsymbol{Z}} \in \mathbb{R}^{d\times k}$, seja $f_{\hat{\boldsymbol{z}}}(\boldsymbol{W}) = f(\boldsymbol{W}, \hat{\boldsymbol{Z}})$. Esta função $f$ verifica a seguinte propriedade:

**Propriedade 2** *Seja $\hat{\boldsymbol{Z}} \in \mathbb{R}^{d\times k}$. A função $f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W})$ é linear.*

**Prova** Seja $\hat{\boldsymbol{Z}} \in \mathbb{R}^{d\times k}$ e considere-se $\boldsymbol{W}^1$ e $\boldsymbol{W}^2 \in \mathbb{R}^{k\times n}$. Então:

$$\begin{aligned} f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W^1} + \boldsymbol{W^2}) &= \sum_{i=1}^{k}\sum_{j=1}^{n} \left(w_{ij}^1 + w_{ij}^2\right) D(\boldsymbol{x}_j, \hat{\boldsymbol{Z}}_i) \\ &= \sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij}^1 D(\boldsymbol{x}_j, \hat{\boldsymbol{Z}}_i) + \sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij}^2 D(\boldsymbol{x}_j, \hat{\boldsymbol{Z}}_i) \\ &= f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W^1}) + f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W^2}). \end{aligned}$$

Conlui-se então que a função é linear. □

De seguida, define-se a função reduzida $F$ através da função objetivo do problema $(P)$ e prova-se que é côncava.

**Definição 1 (Função objetivo $F$ reduzida)** *A forma reduzida da função $f(W)$ do problema (P) é definida por:*

$$F(\boldsymbol{W}) = \min\left\{f_{\boldsymbol{Z}}(\boldsymbol{W}) : \boldsymbol{Z} \in \mathbb{R}^{d\times k}\right\},$$

*e $\boldsymbol{W}$ é uma matriz real de dimensão $k \times n$.*

**Lema 1** *A função objetivo $F$ reduzida é côncava.*

**Prova** Considere-se duas matrizes $\boldsymbol{W}^1$, $\boldsymbol{W}^2 \in \mathbb{R}^{k\times n}$ e seja $\gamma$ um escalar tal que, $0 \leq \gamma \leq 1$. Assim, para mostrar que $F$ é côncava, tem de se verificar o seguinte: $F(\gamma\boldsymbol{W}^1 + (1-\gamma)\boldsymbol{W}^2) \geq \gamma F(\boldsymbol{W}^1) + (1-\gamma)F(\boldsymbol{W}^2), \forall\gamma \in [0,1]$. De facto,

$$\begin{aligned}
F(\gamma\boldsymbol{W}^1 + (1-\gamma)\boldsymbol{W}^2) &= \min\left\{f_{\boldsymbol{Z}}(\gamma\boldsymbol{W}^1 + (1-\gamma)\boldsymbol{W}^2) : \boldsymbol{Z} \in \mathbb{R}^{d\times k}\right\} \\
&= \min\left\{\gamma f_{\boldsymbol{Z}}(\boldsymbol{W}^1) + (1-\gamma) f_{\boldsymbol{Z}}(\boldsymbol{W}^2) : \boldsymbol{Z} \in \mathbb{R}^{d\times k}\right\}, \text{ pela linearidade de } f_{\boldsymbol{Z}}. \\
&\geq \gamma \min\left\{f_{\boldsymbol{Z}}(\boldsymbol{W}^1) : \boldsymbol{Z} \in \mathbb{R}^{d\times k}\right\} + (1-\gamma)\min\left\{f_{\boldsymbol{Z}}(\boldsymbol{W}^2) : Z \in \mathbb{R}^{d\times k}\right\} \\
&= \gamma F(\boldsymbol{W}^1) + (1-\gamma)F(\boldsymbol{W}^2).
\end{aligned}$$

□

Desta forma conclui-se que $F$ é uma função côncava e o seguinte exemplo ilustra esse facto onde se considerou $\boldsymbol{W}^1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ e $\boldsymbol{W}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$.

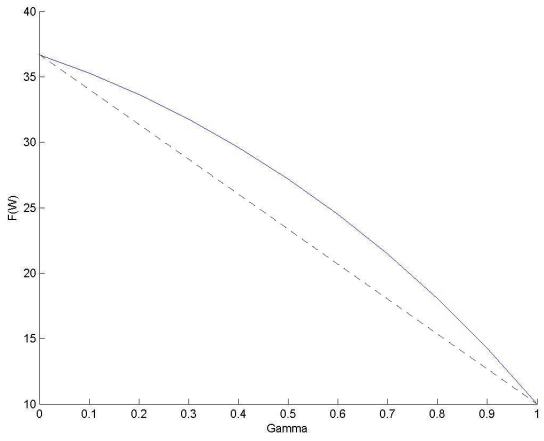



Figura 2.1: Exemplo de um gráfico da função $F(\boldsymbol{W})$, onde $\boldsymbol{W} = \gamma\boldsymbol{W}^1 + (1-\gamma)\boldsymbol{W}^2$.

Observe-se na figura 2.1 a variação de $\gamma$ no eixo das abcissas e o resultado de $F$ no eixo das ordenadas.

De seguida define-se o conjunto $S$ e interessa provar que é um poliedro convexo. Um poliedro diz-se convexo se um segmento de reta que une quaisquer dois pontos do conjunto estiver contido no poliedro. Assim, define-se o conjunto $S = S_1 \times S_2 \times \ldots \times S_n$.

**Proposição 1** *O conjunto $S$ é um poliedro convexo.*

**Prova** Sabendo que $\boldsymbol{W}_j \in S_j$, é equivalente a escrever o seguinte:

$$\begin{cases} \boldsymbol{A}\boldsymbol{W}_j = \boldsymbol{b}_j \\ \boldsymbol{W}_j \geq 0 \end{cases},$$

sendo $\boldsymbol{A} = [1 \ldots 1]$ e $\boldsymbol{b}_j = [1]$.

Prove-se em primeiro lugar que $S_j, j \in \{1, \ldots, n\}$ é convexo. Seja $\boldsymbol{W}_j^1$ e $\boldsymbol{W}_j^2$ dois elementos de $S_j$, logo $\sum_{i=1}^k w_{ij}^1 = 1, w_{ij}^1 \geq 0$ e $\sum_{i=1}^k w_{ij}^2 = 1,\ w_{ij}^2 \geq 0$. Quer-se provar que $\alpha \boldsymbol{W}_j^1 + (1-\alpha)\boldsymbol{W}_j^2 \in S_j$. De facto:

$$\begin{aligned} \sum_{i=1}^{k} \left(\alpha w_{ij}^1 + (1-\alpha) w_{ij}^2\right) &= \sum_{i=1}^{k} \alpha w_{ij}^1 + (1-\alpha) \sum_{i=1}^{k} w_{ij}^2 \\ &= \alpha + 1 - \alpha \\ &= 1. \end{aligned}$$

Além disso $\alpha w_j^1 + (1-\alpha) w_j^2 \geq 0$. Como a reta se encontra contida no poliedro, então $S_j$ é um poliedro convexo. Como $S$ é um produto cartesiano de convexos, logo é um convexo. Além disso, é um poliedro porque $\boldsymbol{W} \in S \Leftrightarrow \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, onde $\boldsymbol{A} = \begin{bmatrix} 1 \ldots 1 & 0 \ldots 0 & \ldots & 0 \ldots 0 \\ 0 \ldots 0 & 1 \ldots 1 & \ldots & 0 \ldots 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 \ldots 0 & 0 \ldots 0 & \ldots & 1 \ldots 1 \end{bmatrix}$, $x = \begin{bmatrix} \boldsymbol{W}_1 \\ \vdots \\ \boldsymbol{W}_n \end{bmatrix}$ e $b = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$, com $\boldsymbol{x} \geq 0$. □

Após introduzido o conjunto prova-se que as restrições do problema $(P)$ podem ser obtidas do conjunto $S_j$ através do seguinte teorema:

**Teorema 1** *O conjunto dos pontos extremos de $S$ corresponde às soluções admissíveis de $(P)$.*

**Prova** Seja $\boldsymbol{W}$ um ponto extremo de $S$. Como $\boldsymbol{W} \in S$ então $\sum_{i=1}^k w_{ij} = 1,\ \forall j \in \{1, \ldots, n\}$ e $w_{ij} \geq 0$. Como $S$ é um poliedro convexo resta apenas provar $w_{ij} \in \{0, 1\}$.

Como $\boldsymbol{W}$ é um ponto extremo de $S$, é possível decompor $\boldsymbol{A}$ na forma $\boldsymbol{A} = [\boldsymbol{B}|\boldsymbol{N}]$, onde $\boldsymbol{B}_{k \times k} = I$ é uma base formada pelas colunas de $\boldsymbol{A}$ associadas às $k$ variáveis

básicas de $\boldsymbol{W}$. Já $\boldsymbol{N}$ contém as restantes colunas associadas às variáveis não básicas (que têm o valor zero). Sendo assim obtém-se o seguinte:

$$\boldsymbol{Ax} = \boldsymbol{b} \Leftrightarrow \begin{bmatrix} \boldsymbol{B} & | & \boldsymbol{N} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_B} \\ \boldsymbol{x_N} \end{bmatrix} = \boldsymbol{b} \Leftrightarrow \boldsymbol{Bx_B} + \boldsymbol{Nx_N} = \boldsymbol{b} \Leftrightarrow \boldsymbol{x_B} = \boldsymbol{b} = \mathbb{1}.$$

Este cálculo confirma que todas as variáveis básicas terão o valor de 1, concluindo-se que $w_{ij} \in \{0, 1\}$.

Para provar a implicação contrária, seja $\boldsymbol{W}$ uma solução admissível de $(P)$, então $w_{ij} \in \{0, 1\}$ e $\sum_{i=1}^{k} w_{ij} = 1,\ \forall j = 1, \ldots, n$, logo $\boldsymbol{W} \in S$, restando apenas provar que $\boldsymbol{W}$ é um ponto extremo de $S$. Considera-se a submatriz $\boldsymbol{B}$ de $\boldsymbol{W}$ formada pelas colunas, onde $w_{ij} = 1$. Esta matriz corresponde à matriz $I$ uma vez que em cada grupo de variáveis $w_{ij}, j \in \{1, \ldots, n\}$ só existe uma que é igual a 1 sendo as restantes iguais a 0, isto porque $\sum_{i=1}^{k} w_{ij} = 1$ e $w_{ij} \in \{0, 1\}$. Sendo assim, $\boldsymbol{B}$ é uma base, logo a solução é básica o que implica que $\boldsymbol{W}$ seja um ponto extremo de $S$. □

O problema $(P)$ é um problema discreto e, por isso, os algoritmos tornam-se computacionalmente mais exigentes. Desta forma interessa relaxar o problema $(P)$ substituindo as restrições $w_{ij} \in \{0, 1\}$ por $w_{ij} \geq 0$. Esta relaxação permite definir o problema reduzido $(RP)$.

**Definição 2 (Problema reduzido $(RP)$)** *O problema reduzido (RP) do problema (P) é definido pelo seguinte:*

$$(RP): \ \textit{min } F(W), \ \textit{sujeito a } W \in S.$$

**Propriedade 3** *Se $F$ é uma função côncava num poliedro convexo $S$, então $F$ atinge o mínimo num ponto extremo de $S$.*

**Prova** Seja $x$ um minimizante de $F$ e suponha-se que $x$ não é um ponto extremo de $S$. Então $x$ pode escrever-se como uma combinação linear de pontos extremos de $S$, isto é, $x$ é da forma $x = \sum_{i=1}^{n} \alpha_i x_i$ tal que $\sum_{i=1}^{n} \alpha_i = 1$, com $x_i$ pontos extremos de $S$ e $\alpha_i \geq 0$. Como $F$ é côncava, então verifica o seguinte:

$$F(\sum_{i=1}^{n} \alpha_i x_i) \geq \sum_{i=1}^{n} \alpha_i F(x_i)$$

$$F(x) - \sum_{i=1}^{n} \alpha_i F(x_i) \geq 0$$

$$\sum_{i=1}^{n} \alpha_i F(x) - \sum_{i=1}^{n} \alpha_i F(x_i) \geq 0$$

$$\sum_{i=1}^{n} \alpha_i (F(x) - F(x_i)) \geq 0$$

Mas $F(x)$ é um mínimo, logo $F(x) \leq F(x_i)$ o que implica que $F(x) = F(x_i)$. Assim conclui-se que os pontos extremos $x_i$ também são pontos onde $F$ atinge o mínimo. □

Após esta propriedade pode deduzir-se o lema seguinte:

**Lema 2** *O problema (RP) e (P) são equivalentes.*

Este lema deduz-se a partir da propriedade 3 e do teorema 1. A resolução direta do problema $(RP)$ é difícil e, por esse motivo, introduz-se o conceito de solução ótima parcial.

**Definição 3 (Solução Ótima Parcial)** *Uma solução ótima parcial* $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ *do problema (P) é uma solução que satisfaz as seguintes condições:*

$$f_{\boldsymbol{Z}^*}(\boldsymbol{W}^*) \leq f_{\boldsymbol{Z}^*}(\boldsymbol{W}), \quad \forall \boldsymbol{W} \in S,$$

$$f_{\boldsymbol{W}^*}(\boldsymbol{Z}^*) \leq f_{\boldsymbol{W}^*}(\boldsymbol{Z}), \quad \forall \boldsymbol{Z} \in \mathbb{R}^{d \times k}.$$

No sentido da sua resolução minimiza-se alternadamente a função $f$ em $\boldsymbol{W}$ e $\boldsymbol{Z}$ através de dois tipos de problemas:

- Problema $P_{\hat{Z}}$: Dado um $\hat{Z} \in R^{d \times k}$, pretende-se minimizar $\{f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W}) : \boldsymbol{W} \in S\}$.

  Neste problema $P_{\hat{Z}}$ é feita a atribuição dos pontos aos clusters.

- Problema $P_{\hat{W}}$: Dado $\hat{W} \in S$, pretende-se minimizar $\{f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}) : \boldsymbol{Z} \in \mathbb{R}^{d \times k}\}$.

  Neste problema $P_{\hat{W}}$ atualizam-se os centroides.

A solução para o $P_{\hat{Z}}$ é simples de obter quando $D(\boldsymbol{x}_j, \boldsymbol{z}_i) = \|\boldsymbol{x}_j - \boldsymbol{z}_i\|_2^2$, como se pode observar através do próximo lema.

**Lema 3** *Seja* $\hat{\boldsymbol{Z}} \in \mathbb{R}^{d\times k}$ *fixo, define-se* $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n\times k}$ *do seguinte modo: para cada* $j \in \{1,\dots,n\}$*, tem-se o seguinte:*

$$\tilde{w}_{ij} = \begin{cases} 1 & se \quad \|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_i\|_2^2 \leq \|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2, \forall l \in \{1,\dots,k\} \\ 0 & se \quad \textit{caso contrário.} \end{cases}$$

*Então* $\boldsymbol{W}$ *é uma solução ótima de* $P_{\hat{\boldsymbol{Z}}}$.

**Prova** Primeiro prova-se que $\tilde{\boldsymbol{W}} \in S$. Pela definição de $\tilde{w}_{ij}$, temos $\tilde{w}_{ij} \in \{0,1\}$ e portanto $\tilde{w}_{ij} \geq 0$. Para cada $j \in \{1,\dots,n\}$, existe apenas um $i \in \{1,\dots,k\}$, onde $\tilde{w}_{ij} = 1$, o índice onde $\|\boldsymbol{x}_j - \boldsymbol{z}_l\|_2^2$ é mínimo. Para os restantes $i$ tem-se que $\tilde{w}_{ij} = 0$. Desta forma verifica-se que $\sum_{i=1}^{k} \tilde{w}_{ij} = 1$, logo $\tilde{\boldsymbol{W}} \in S$. Falta provar que $f_{\hat{\boldsymbol{Z}}}(\tilde{\boldsymbol{W}}) < f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W})$, $\forall\, \boldsymbol{W} \in \mathbb{R}^{n\times k}$.

Dado $j \in \{1,\dots,n\}$ e seja $i^* \in \{1,\dots,k\}$ tal que $\tilde{w}_{i^*j} = 1$, se $\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_{i^*}\|_2^2 \leq \|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2$, $\forall l \in \{1,\dots,k\}$. Para $\forall \boldsymbol{W} \in S$ temos que:

$$\begin{aligned} & w_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_{i^*}\|_2^2 \leq w_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2, \ \forall l \\ \Leftrightarrow & \sum_{l=1}^{k} w_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_{i^*}\|_2^2 \leq \sum_{l=1}^{k} w_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2 \\ \Leftrightarrow & \|\boldsymbol{x}_j - \boldsymbol{z}_{i^*}\|_2^2 \leq \sum_{l=1}^{k} w_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2. \end{aligned} \tag{2.2}$$

O termo esquerdo da inequação 2.2 é igual a ter-se o seguinte:

$$\begin{aligned} \|\boldsymbol{x}_j - \boldsymbol{z}_{i^*}\|^2 &= \tilde{w}_{i^*j}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_{i^*}\|_2^2 \\ &= \tilde{w}_{i^*j}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_{i^*}\|_2^2 + \sum_{l\neq i^*} \tilde{w}_{ij}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2 \\ &= \sum_{l=1}^{k} \tilde{w}_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2 \end{aligned}$$

Substituindo em (2.2) obtém-se $\sum_{l=1}^{k} \tilde{w}_{lj}\|\boldsymbol{x}_j - \hat{\boldsymbol{z}}_l\|_2^2 \leq \sum_{l=1}^{k} w_{lj}\|\boldsymbol{x}_j - \boldsymbol{z}_l\|_2^2$. Resta apenas aplicar em ambos os termos o somatório em $j$:

$$\sum_{j=1}^{n}\sum_{i=1}^{k} \tilde{w}_{ij}\|\boldsymbol{x}_j - z_i\|_2^2 \leq \sum_{j=1}^{n}\sum_{l=1}^{k} w_{lj}\|\boldsymbol{x}_j - z_l\|_2^2$$

Permutando os somatórios, conclui-se o seguinte:

$$f_{\hat{\boldsymbol{Z}}}(\tilde{\boldsymbol{W}}) \leq f_{\hat{\boldsymbol{Z}}}(\boldsymbol{W}).$$

Logo $\tilde{\boldsymbol{W}}$ é uma solução ótima de $P_{\hat{\boldsymbol{Z}}}$. □

Já a solução $P_{\hat{\boldsymbol{W}}}$ em geral, não é tão simples de obter. No entanto, no caso do algoritmo *K-means* através da função de semelhança $D(\boldsymbol{x}, \boldsymbol{Z}) = \|\boldsymbol{x} - \boldsymbol{Z}\|_2^2$, consegue deduzir-se uma expressão direta para a solução ótima de $P_{\hat{\boldsymbol{W}}}$.

**Lema 4** *Para um $\hat{\boldsymbol{W}} \in \mathbb{R}^{n\times k}$ fixo, define-se $\boldsymbol{Z} \in \mathbb{R}^{d\times k}$ da seguinte forma:*

$$\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k], \text{ onde } \boldsymbol{z}_i = \frac{\sum_{j=1}^n w_{ij}\boldsymbol{x}_j}{\sum_{j=1}^n w_{ij}},\ i = 1, \ldots, k.$$

*Então $\boldsymbol{Z}$ é uma solução ótima de $P_{\hat{\boldsymbol{W}}}$.*

**Prova** Seja $\hat{\boldsymbol{W}}$ fixo. Note-se que $f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}) = \sum_{i=1}^k \sum_{j=1}^n \hat{w}_{ij}\|\boldsymbol{x}_j - \boldsymbol{z}_i\|_2^2$ é uma função convexa, o que garante que o mínimo é único. Basta, então, encontrar o zero da derivada da função $f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z})$, em ordem a $\boldsymbol{z}_i$ , $i \in \{1, \ldots, k\}$, para qualquer $\boldsymbol{z}_i$. Derivando $f_{\hat{\boldsymbol{W}}}$ obtém-se o seguinte:

$$\begin{aligned}\nabla_{\boldsymbol{z}_i} f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}) &= \nabla_{\boldsymbol{z}_i} \sum_{i=1}^k \sum_{j=1}^n \hat{w}_{ij}\|\boldsymbol{x}_j - \boldsymbol{z}_i\|_2^2 \\ &= \sum_{j=1}^n \hat{w}_{ij}(-2(\boldsymbol{x}_j - \boldsymbol{z}_i)).\end{aligned}$$

A solução de $P_{\hat{\boldsymbol{W}}}$ obtém-se da seguinte forma:

$$\begin{aligned}\nabla_{\boldsymbol{z}_i} f_{\hat{\boldsymbol{W}}}(\boldsymbol{Z}) = 0 &\Leftrightarrow \sum_{j=1}^n \hat{w}_{ij}(-2(\boldsymbol{x}_j - \boldsymbol{z}_i)) = 0 \\ &\Leftrightarrow \boldsymbol{z}_i = \frac{\sum_{j=1}^n \hat{w}_{ij}\boldsymbol{x}_j}{\sum_{j=1}^n \hat{w}_{ij}}, i = 1, \ldots, k\end{aligned}$$

Encontra-se assim a solução de $P_{\hat{\boldsymbol{W}}}$. □

Estas soluções são importantes porque satisfazem as condições necessárias de otimalidade de primeira ordem, também conhecidas por condições **Karush-Kuhn-Tucker** $(KKT)$[3], para o problema $(RP)$, e por este facto, são candidatos a mínimos locais de $F$. Para se abordar este resultado, é necessário reformular o problema $(RP)$ e introduzir algumas restrições. Note-se que o problema $(RP)$ é equivalente a:

$$\begin{aligned} min \quad & f(\boldsymbol{W}, \boldsymbol{Z}) \\ s.a. \quad & \boldsymbol{W} \in S \\ & \nabla_{\boldsymbol{Z}} f_{\boldsymbol{W}}(\boldsymbol{W}, \boldsymbol{Z}) = 0 \end{aligned} \quad ,$$

uma vez que $f_{\boldsymbol{W}}$ é convexa e, portanto, o minimizante de $f_{\boldsymbol{W}}$ verifica $\nabla_{\boldsymbol{Z}} f(\boldsymbol{W}, \boldsymbol{Z}) = \nabla_{\boldsymbol{Z}} f_{\boldsymbol{W}}(\boldsymbol{Z}) = 0$.

Tendo como base o lema 4 e tendo em conta a função $D(\boldsymbol{x}, \boldsymbol{Z})$ do *K-means*, então $\nabla_{z_i} f_{\boldsymbol{W}}(\boldsymbol{Z}) = 0 \Leftrightarrow \sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_i) = 0$. Seja $c_i(\boldsymbol{W}, \boldsymbol{Z}) = \sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_i)$, o que torna o problema anterior equivalente a:

$$\begin{aligned} min \quad & f(\boldsymbol{W}, \boldsymbol{Z}) \\ s.a. \quad & g_j(\boldsymbol{W}) \geq 0, \quad j = 1, \ldots, n \\ & h_j(\boldsymbol{W}) = 0, \quad j = 1, \ldots, n \\ & c_i(\boldsymbol{W}, \boldsymbol{Z}) = 0, \quad i = 1, \ldots, k \end{aligned}$$

De seguida, define-se o lagrangeano, tendo em conta que $\mu_j \in \mathbb{R}^k, \mu_j \leq 0, \lambda_j \in \mathbb{R}$ e $\alpha_i \in \mathbb{R}^d$ são os multiplicadores de lagrange:

$$L(\boldsymbol{W}, \boldsymbol{Z}, \mu, \lambda, \alpha) = f(\boldsymbol{W}, \boldsymbol{Z}) + \sum_{l=1}^{n} \mu_l^T g_l(\boldsymbol{W}) + \sum_{l=1}^{n} \lambda_l h_l(\boldsymbol{W}) + \sum_{l=1}^{k} \lambda_l^T c_l(\boldsymbol{W}, \boldsymbol{Z})$$

Após a definição do lagrangeano estamos em condições de enunciar as condições $KKT$ para este problema. As condições são as seguintes:

$$\nabla_{\boldsymbol{W}_j} L(\boldsymbol{W}, \boldsymbol{Z}, \mu, \lambda, \alpha) = 0, \forall j \tag{2.3}$$

$$\nabla_{\boldsymbol{Z}_i} L(\boldsymbol{W}, \boldsymbol{Z}, \mu, \lambda, \alpha) = 0, \forall i \tag{2.4}$$

$$g_j(\boldsymbol{W}) \geq 0, \forall j \tag{2.5}$$

$$h_j(\boldsymbol{W}) = 0, \forall j \tag{2.6}$$

$$c_i(\boldsymbol{W}, \boldsymbol{Z}) = 0, \forall i \tag{2.7}$$

$$\mu_j \leq 0, \forall j \tag{2.8}$$

$$g_{ij}(\boldsymbol{W})\mu_{ij} = 0, \forall i, \forall j \tag{2.9}$$

A equação 2.3 é equivalente a:

$$\begin{aligned} & \nabla_{\boldsymbol{W}_j} f(\boldsymbol{W}, \boldsymbol{Z}) + \nabla_{\boldsymbol{W}_j}^T g_j(\boldsymbol{W})\mu_j + \lambda_j \nabla_{\boldsymbol{W}_j} h_j(\boldsymbol{W}) + \sum_{l=1}^{k} \nabla_{\boldsymbol{W}_j}^T c_l(\boldsymbol{W}, \boldsymbol{Z})\alpha_l = 0 \\ \Leftrightarrow \quad & \nabla_{\boldsymbol{W}_j} f(\boldsymbol{W}, \boldsymbol{Z}) + \mu_j + \lambda_j \mathbb{1} + \sum_{l=1}^{k} \nabla_{\boldsymbol{W}_j}^T c_l(\boldsymbol{W}, \boldsymbol{Z})\alpha_l = 0 \end{aligned} \tag{2.10}$$

A equação 2.4 é equivalente a:

$$\nabla_{\boldsymbol{z}_i} f(\boldsymbol{W}, \boldsymbol{Z}) + \nabla_{\boldsymbol{z}_i}^T c_i(\boldsymbol{W}, \boldsymbol{Z})\alpha_i = 0,$$

com base na condição (2.7), $\nabla_{\boldsymbol{z}_i} f(\boldsymbol{W}, \boldsymbol{Z}) = 0$ obtém-se o seguinte:

$$0 - \left( \sum_{j=1}^{n} w_{ij} \right) \times I \times \alpha_i = 0.$$

Assumindo que $\sum_{j=1}^{n} w_{ij} \neq 0$, isto indica que o cluster $i$ não tinha pontos associados e, por isso, não seria utilizado. Daí que, $\forall i \in \{1, \ldots, k\}$, então $\alpha_i = 0$. Substituindo em 2.10 $\alpha_i = 0$ obtém-se o seguinte:

$$\nabla_{\boldsymbol{W}_j} f(\boldsymbol{W}, \boldsymbol{Z}) + \mu_j + \lambda_j \mathbb{1} = 0 \tag{2.11}$$

As condições para o problema $(RP)$ são sete. As duas primeiras são as derivadas do lagrangeano em ordem a $\boldsymbol{W}$ e a $\boldsymbol{Z}$, que verificam as condições de $P_{\hat{\boldsymbol{W}}}$ e $P_{\hat{\boldsymbol{Z}}}$. Após reformulado o problema $(RP)$ e introduzidas as condições $KKT$, estão reunidas todas as condições para se abordar o teorema seguinte.

**Teorema 2** *$(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ é um ponto $KKT$ de $(RP)$ se e só se $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ for uma solução ótima parcial de $(P)$.*

**Prova** Suponhamos que $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ é um ponto $KKT$ de $(RP)$. Então existem $\mu_j \in \mathbb{R}^k, \lambda_j \in \mathbb{R}$ e $\alpha_i \in \mathbb{R}^d$ tais que as condições 2.3 a 2.9 são verificadas. Quer-se provar que $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ é uma solução ótima parcial. Deste modo, $\boldsymbol{Z}^*$ deve ser uma solução ótima de $P_{\boldsymbol{W}^*}$, e $\boldsymbol{W}^*$ deve ser uma solução ótima de $P_{\boldsymbol{Z}^*}$. Portanto devem-se verificar as condições de otimalidade de primeira ordem para $P_{\boldsymbol{W}^*}$, isto é:

$$\nabla_{\boldsymbol{Z}} f_{\boldsymbol{W}^*}(\boldsymbol{Z}^*) = 0, \tag{2.12}$$

o que se verifica pela equação (2.7). Além disso, sendo $f_{\boldsymbol{W}^*}(\boldsymbol{Z}^*)$ convexa, então $\boldsymbol{Z}^*$ é solução ótima de $P_W^*$. Por outro lado devem-se verificar as condições de otimilidade de primeira ordem para $P_{\boldsymbol{Z}^*}$, isto é:

$$\nabla_{\boldsymbol{W}_j} f_{\boldsymbol{Z}^*}(\boldsymbol{W}^*) + \sum_{l=1}^{n} \nabla_{\boldsymbol{W}_j}^T g_l(\boldsymbol{W}^*)\mu_l + \sum_{l=1}^{n} \lambda_l \nabla_{\boldsymbol{W}_j} h_j(\boldsymbol{W}^*) = 0 \tag{2.13}$$

$$g_j(\boldsymbol{W}^*) \geq 0,\ h_j(\boldsymbol{W}^*) = 0,\ \mu_j \leq 0,\ \forall j \in \{1, \ldots n\} \tag{2.14}$$

$$g_{ij}(\boldsymbol{W}^*)\mu_{ij} = 0,\ \forall i \in \{1, \ldots, k\}\ ,\ \forall j \in \{1, \ldots, n\} \tag{2.15}$$

estas condições são satisfeitas pelas equações (2.5), (2.6), (2.8), (2.9) e (2.11). Sendo $f_{\boldsymbol{Z}^*}$ uma função linear, então $\boldsymbol{W}^*$ é solução ótima de $(P)$.
Reciprocamente, se $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ é uma solução ótima parcial de $(P)$ então existem $\mu_j \in \mathbb{R}^k$ e $\lambda_j \in \mathbb{R}$ que verificam as equações (2.12) a (2.15). Considerando que $\alpha_i = 0$, então verifica-se que as restrições (2.3) a (2.9) são válidas e, por isso, o ponto $(\boldsymbol{W}^*, \boldsymbol{Z}^*)$ é $KKT$ para o problema $(RP)$.

□

De seguida apresentam-se dois lemas que auxiliam na conclusão do estudo sobre a convergência do algoritmo. Considere-se que $\boldsymbol{W}^r$ são as atribuições dos pontos em relação à iteração $r$ e $\boldsymbol{Z}^r$ os centroides na iteração $r$. No lema 5 prova-se que a sequência $f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^r)$ é decrescente e no lema 6 prova-se que todos os $\boldsymbol{W}^r$ são distintos entre si.

**Lema 5** *Para quaisquer iterações r se* $\boldsymbol{Z}^{r-1} \neq \boldsymbol{Z}^r$*, então* $f(\boldsymbol{W}^{r-2}, \boldsymbol{Z}^{r-1}) > f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^r)$.

**Prova** Atendendo que $\boldsymbol{W}^{r-1}$ é solução de $P_{\boldsymbol{Z}^{r-1}}$, então $f(\boldsymbol{W}^{r-2}, \boldsymbol{Z}^{r-1}) \geq f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^{r-1})$. Por outro lado, $\boldsymbol{Z}^r$ é solução de $P_{\boldsymbol{W}^{r-1}}$, então $f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^{r-1}) \geq f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^r)$. Além disso, se $\boldsymbol{Z}^{r-1} \neq \boldsymbol{Z}^r$, então $f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^{r-1}) > f(\boldsymbol{W}^{r-1}, \boldsymbol{Z}^r)$. Uma vez que $f_{\boldsymbol{W}^{r-1}}$ é convexa, logo o mínimo será único. □

**Lema 6** *Admita-se que o algoritmo K-means realizou r iterações. Então* $\boldsymbol{W}^0 \dots \boldsymbol{W}^r$ *são distintos entre si.*

**Prova** Mostra-se por absurdo que os elementos da sequência de atribuições $\boldsymbol{W}^0 \dots \boldsymbol{W}^r$ são todos distintos. Se assim não fosse, então existira $r_1$ e $r_2$, tal que $r_1 < r_2 < r$, e $\boldsymbol{W}^{r_2} = \boldsymbol{W}^{r_1}$. Por $(P_{\boldsymbol{W}^{r_1}})$,$(P_{\boldsymbol{W}^{r_2}})$ obtêm-se as soluções $\boldsymbol{Z}^{r_1+1}$ e $\boldsymbol{Z}^{r_2+1}$ e, como $\boldsymbol{W}^{r_2} = \boldsymbol{W}^{r_1}$, então $\boldsymbol{Z}^{r_1+1} = \boldsymbol{Z}^{r_2+1}$ logo $f(\boldsymbol{W}^{r_1}, \boldsymbol{Z}^{r_1+1}) = f(\boldsymbol{W}^{r_2}, \boldsymbol{Z}^{r_2+1})$. Como $f$ é uma sequência não crescente, obteria-se $\boldsymbol{Z}^{r_2} = \boldsymbol{Z}^{r_1+1} \dots \boldsymbol{Z}^{r_1} = \boldsymbol{Z}^{r_2+1}$, pelo que o algoritmo deveria ter parado na iteração $r_1 + 1$ e não na iteração $r$, o que faz com que todos os $\boldsymbol{W}^r$ sejam distintos. □

Aborde-se agora o teorema que prova que o *K-means* converge num número finito de iterações, sendo este o objetivo desta secção.

**Teorema 3** *O algoritmo K-means converge para uma solução ótima parcial do problema (P) num número finito de iterações.*

**Prova** Como todos os $\boldsymbol{W}^r$ obtidos pelo algoritmo são distintos e correspondem a pontos extremos de $S$, que são finitos, então no máximo o algoritmo precisa de um número finito de iterações até convergir para uma solução. □

Conclui-se, deste modo, o estudo da convergência do algoritmo *K-means* e, de seguida, estuda-se a ordem de complexidade do mesmo.

### 2.1.3. Ordem de complexidade

Seja $\boldsymbol{D}$ o conjunto de informação que contém, no total, $n$ pontos e cada ponto com uma dimensão $d$. Ao ser aplicado um método de clustering a $\boldsymbol{D}$, este fica dividido em $k$ cluster, $C = \{C_1, C_2, \ldots, C_k\}$, sendo cada cluster representado pelo respetivo centroide. Denote-se por $t$ o número de iterações que o algoritmo leva até convergir, cada uma delas composta por duas fases: a fase de atribuição dos pontos aos clusters e a fase da atualização do centroide de cada cluster. Será importante considerar esforço computacional envolvido em cada uma delas. A fase de atribuição dos $n$ pontos aos $k$ clusters necessita de calcular as $n$ distâncias dos pontos de dimensão $d$ aos $k$ centroides o que leva um tempo de $O(nkd)$. Denotando por $n_i$ a dimensão do cluster $C_i$, e tendo em conta que $\sum_{i=1}^{k} n_i = n$, pode avaliar-se a ordem de complexidade da fase de atualização dos clusters. O centroide do cluster $C_i$ é atualizado com o cálculo da média dos seus $n_i$ pontos (de dimensão $d$) originando um tempo de $O(n_i d)$. Deste modo, o tempo total desta fase é $O(nd)$. Tendo em conta que cada iteração $t$ necessita de executar cada uma destas fases, conclui-se que este algoritmo tem uma complexidade computacional de $O(tnkd)$.

## 2.2. Expectation-Maximization

Nesta secção, aborda-se um exemplo de atribuição múltipla onde é atribuída uma probabilidade de um ponto poder pertencer a um outro cluster. Para uma melhor compreensão deste algoritmo é necessário introduzir primeiramente a caracterização dos clusters através do modelo misto gaussiano, cujos parâmetros serão estimados pelo método da máxima verosimilhança.

### 2.2.1. Modelo Misto Gaussiano

Neste algoritmo assume-se que cada cluster $C_i$ é caracterizado através da distribuição normal multivariada, isto é:

$$f_i(\boldsymbol{x}) = f(\boldsymbol{x}|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{z}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{z}_i)}{2}\right\}$$

onde $\boldsymbol{z_i} \in \mathbb{R}^d$ é a média do cluster $C_i$ e $\boldsymbol{\Sigma_i} \in \mathbb{R}^{d\times d}$ a matriz covariância, sendo ambos desconhecidos.

Devido a esta caracterização dos clusters, a função densidade para todos os pontos $f(\boldsymbol{x})$ é vista como um modelo misto gaussiano. Esta função representa-se da seguinte forma:

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} f(\boldsymbol{x}|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)P(C_i)$$

onde $P(C_i)$ são os parâmetros de mistura (também desconhecidos) que satisfazem a condição $\sum_{i=1}^{k} P(C_i) = 1$, e que são designados pelas probabilidades à priori de cada cluster. Assim o modelo é definido através da média $\boldsymbol{z}_i$, da matriz covariâncias $\boldsymbol{\Sigma}_i$ e das probabilidades a priori $P(C_i)$.

### 2.2.2. Estimação da máxima verosimilhança

Nesta secção, apresenta-se um breve resumo das fórmulas para estimar os parâmetros do modelo misto Gaussiano utilizando o método da máxima verosimilhança. A dedução completa das fórmulas pode ser consultada em [13].

Por uma questão de simplificação, considera-se:

$$\boldsymbol{\theta} = \{\boldsymbol{z}_1, \boldsymbol{\Sigma}_1, P(C_1), \ldots, \boldsymbol{z}_k, \boldsymbol{\Sigma}_k, P(C_k)\}\,.$$

Dado um conjunto de pontos $\boldsymbol{D}$, define-se a função de verosimilhança de $\boldsymbol{\theta}$ como a probabilidade condicional de $\boldsymbol{D}$ dado os parâmetros do modelo $\boldsymbol{\theta}$. Deste modo, supondo que os pontos correspondem a observações de uma amostra aleatória do vector $X$ com função de densidade $f$ do modelo misto gaussiano, obtém-se a seguinte função de verosimilhança:

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) = \Pi_{j=1}^{n} f(\boldsymbol{x}_j)$$

O objetivo desta estimação é encontrar os parâmetros $\boldsymbol{\theta}$ que maximizam a função de verosimilhança. Como o valor máximo do $ln(\mathcal{L}(\theta|\boldsymbol{D}))$ é atingido no mesmo ponto que no máximo do $\mathcal{L}(\theta|\boldsymbol{D})$, opta-se pelo uso da primeira função. Através das propriedades dos logaritmos pode obter-se um somatório a partir de um produtório.

Deste modo, os parâmetros que maximizam a função de máxima verosimilhança são dados por:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \{ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}))\}$$

A função do logaritmo da verosimilhança é dada pela seguinte expressão:

$$ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})) = \sum_{j=1}^{n} ln(f(\boldsymbol{x}_j)) = \sum_{j=1}^{n} ln(\sum_{i=1}^{k} f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)P(C_i)).$$

O cálculo analítico (e mesmo numérico) do máximo desta função é muito difícil. Por esse motivo, utiliza-se a abordagem expetativas-maximização para encontrar os parâmetros que maximizam $ln(\mathcal{L}(\theta|\boldsymbol{D}))$. Esta abordagem inicializa os parâmetros $\boldsymbol{\theta}$ e introduz variáveis de latência que dão informação sobre a probabilidade de um ponto $\boldsymbol{x}_j$ pertencer a um determinado cluster. O valor destas variáveis não é conhecido mas, dado o conjunto $\boldsymbol{D}$ e os parâmetros $\boldsymbol{\theta}$, o seu valor esperado será $P(C_i|\boldsymbol{x}_j)$. Este valor designa-se por probabilidades posteriores e pode ser interpretado como a contribuição do ponto $\boldsymbol{x}_j$ no cluster $C_i$. O cálculo deste valor pode ser feito através do teorema de *Bayes*:

$$P(C_i|\boldsymbol{x}_j) = \frac{f_i(\boldsymbol{x}_j)P(C_i)}{\sum_{a=1}^{k} f_a(\boldsymbol{x}_j)P(C_a)} = \frac{f_i(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)P(C_i)}{f(\boldsymbol{x}_j)}. \tag{2.16}$$

Note-se que, a partir deste momento, sempre que for utilizada a notação $w_{ij}$, esta refere-se à contribuição do ponto $\boldsymbol{x}_j$ no cluster $C_i$.

Deste modo, pode definir-se uma matriz dos pesos de todos os pontos para cada cluster onde o elemento genérico é $w_{ij} = P(C_i|\boldsymbol{x}_j)$.

As fórmulas para estimação dos parâmetros deste modelo serão obtidas anulando as derivadas parciais da função $\ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}))$. A derivada parcial em função de um parâmetro genérico $\boldsymbol{\theta}_i$ para o cluster $C_i$, é dada pelo seguinte:

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} ln\left(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})\right) = \sum_{j=1}^{n} \left[ \frac{1}{f(\boldsymbol{x}_j)} \frac{\partial}{\partial \boldsymbol{\theta}_i} \left(f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)P(C_i)\right) \right]$$

Tendo em conta que $\boldsymbol{\Sigma}_i$ é uma matriz invertível, pode reescrever-se a derivada parcial da seguinte forma:

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} ln\left(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})\right) = \sum_{j=1}^{n} \left[ \frac{1}{f(\boldsymbol{x}_j)} \frac{\partial}{\partial \boldsymbol{\theta}_i} \left( (2\pi)^{-\frac{d}{2}} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\left\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\right\} P(C_i) \right) \right] \tag{2.17}$$

onde se substitui a densidade normal multivariada $f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)$ e se tem em conta que $g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i) = -\frac{1}{2}(\boldsymbol{x}_j - \boldsymbol{z}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_j - \boldsymbol{z}_i)$. Note-se também que a derivada da

exponencial de $g$, é dada pelo seguinte:

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} = \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} \frac{\partial}{\partial \boldsymbol{\theta}_i} g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i) \tag{2.18}$$

Derivando $\ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}))$ em ordem a $\boldsymbol{z}_i$, obtém-se uma fórmula para recalcular a média $\boldsymbol{z}_i$. Tendo em conta que:

$$\frac{\partial}{\partial \boldsymbol{z}_i} g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i) = \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}_j - \boldsymbol{z}_i) \tag{2.19}$$

obtemos:

$$\frac{\partial}{\partial \boldsymbol{z}_i} ln\left(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})\right) = 0 \Leftrightarrow \sum_{j=1}^{n} w_{ij} \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}_j - \boldsymbol{z}_i) = 0$$

$$\boldsymbol{z}_i = \frac{\sum_{j=1}^{n} w_{ij} \boldsymbol{x}_j}{\sum_{j=1}^{n} w_{ij}}.$$

Para se recalcular a matriz das covariâncias $\boldsymbol{\Sigma}_i$, faz-se um raciocínio análogo. No entanto, o cálculo da derivada parcial de $\ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}))$ é mais complexo. Recorre-se, por isso, à regra do produto para derivar o termo $|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\}$ na fórmula (2.17),

- $\frac{\partial |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}}}{\partial \boldsymbol{\Sigma}_i^{-1}}$

  Para abordar esta derivada é necessário ter em conta que, para todas as matrizes quadradas $A$, se obtém $\frac{\partial |\boldsymbol{A}|}{\partial \boldsymbol{A}} = |\boldsymbol{A}|.(\boldsymbol{A}^{-1})^T$. Dado este resultado, reescreve-se a derivada em questão da seguinte forma:

  $$\frac{\partial |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}}}{\partial \boldsymbol{\Sigma}_i^{-1}} = \frac{1}{2} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \boldsymbol{\Sigma}_i \tag{2.20}$$

- $\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\}$

  Nesta derivada, note-se que, para todas as matrizes quadradas $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ e vetores $a, b \in \mathbb{R}^d$, temos $\frac{\partial}{\partial A} \boldsymbol{a}\boldsymbol{A}\boldsymbol{b}^T = \boldsymbol{a}\boldsymbol{b}^T$. Através da equação (2.18), reescreve-se a derivada da exponencial de $g$ em ordem a $\boldsymbol{\Sigma}_i^{-1}$ da seguinte forma:

  $$\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} = -\frac{1}{2} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} (\boldsymbol{x}_j - \boldsymbol{z}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T \tag{2.21}$$

Após o cálculo destas duas derivadas, pode agora aplicar-se a regra do produto a $|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\}$. Tendo em conta as equações (2.20) e (2.21), obtém-se o seguinte:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} = \\
=&\frac{1}{2}|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \boldsymbol{\Sigma}_i \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} - \frac{1}{2}|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} (\boldsymbol{x}_j - \boldsymbol{z}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T \quad (2.22)\\
=&\frac{1}{2}|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\{g(\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)\} \left(\boldsymbol{\Sigma}_i - (\boldsymbol{x}_j - \boldsymbol{z}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T\right)
\end{aligned}$$

Neste momento, pode aplicar-se a derivada parcial ao logaritmo da função de verosimilhança e igualar a uma matriz $\mathbf{0}_{d\times d}$, com o objetivo de encontrar uma fórmula para recalcular a matriz das covariâncias. Mergulhando a equação (2.22) na equação (2.17) obtém-se o seguinte:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} ln(\mathcal{L}(\boldsymbol{D}|\boldsymbol{\theta})) = 0 \Leftrightarrow \frac{1}{2}\sum_{j=1}^{n} w_{ij} \left(\boldsymbol{\Sigma}_i - (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T\right) = 0 \\
\Leftrightarrow \boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T}{\sum_{j=1}^{n} w_{ij}}
\end{aligned}$$

Finalmente, para se encontrar a fórmula que recalcula as probabilidades à priori, é necessário utilizar a derivada parcial do logaritmo da função de verosimilhança em ordem a $P(C_i)$. Neste cálculo é preciso introduzir um multiplicador de Lagrange $\alpha$ para a seguinte restrição $\sum_{a=1}^{k} P(C_a) = 1$, de modo a reescrever a derivada da seguinte forma:

$$\frac{\partial}{\partial P(C_i)} \left( ln(\mathcal{L}(\boldsymbol{D}|\boldsymbol{\theta})) + \alpha(\sum_{a=1}^{k} P(C_a) - 1) \right) \quad (2.23)$$

Deve-se ter em conta que a derivada parcial do logaritmo da função verosimilhança em ordem a $P(C_i)$ é dada pelo seguinte:

$$\frac{\partial}{\partial P(C_i)} \ln\left(\mathcal{L}(\boldsymbol{D}|\boldsymbol{\theta})\right) = \sum_{j=1}^{n} \frac{f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)}{f(\boldsymbol{x}_j)}$$

De seguida, torna-se à equação (2.23) para utilizar o resultado anterior e igualar a 0, e obtém-se o seguinte:

$$\left( \sum_{j=1}^{n} \frac{f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i)}{f(\boldsymbol{x}_j)} \right) + \alpha = 0$$

Multiplicando ambos os membros por $P(C_i)$, alcança-se o seguinte resultado:

$$\left( \sum_{j=1}^{n} \frac{f(\boldsymbol{x}_j|\boldsymbol{z}_i, \boldsymbol{\Sigma}_i) P(C_i)}{f(\boldsymbol{x}_j)} \right) = -\alpha P(C_i).$$

Sendo o multiplicador de Lagrange $\alpha = -n$, obtém-se a fórmula para recalcular as probabilidades à priori:

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

### 2.2.3. Descrição do algoritmo

Este algoritmo aplica iterativamente a fase de expetativas e a fase da maximização até convergir. A fase das expectativas calcula as probabilidades posteriores do cluster $C_i$ de um dado ponto $\boldsymbol{x}_j$, usando a equação (2.16) e a fase da maximização estima os parâmetros $\boldsymbol{z}_i, \boldsymbol{\Sigma}_i$ e $P(C_i)$ através das equações seguintes :

- $\boldsymbol{z}_i = \frac{\sum_{j=1}^{n} w_{ij}\boldsymbol{x}_j}{\sum_{j=1}^{n} w_{ij}}$
- $\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_i)(\boldsymbol{x}_j - \boldsymbol{z}_i)^T}{\sum_{j=1}^{n} w_{ij}}$
- $P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n}$

O Expectation–Maximization é uma aproximação iterativa e, como tal, é necessário definir os parâmetros iniciais. No caso deste trabalho, foram atribuídos aleatoriamente os elementos do conjunto aos $k$ clusters. Com essa distribuição de pontos foi calculada a média dos pontos de cada um deles de modo a atribuir um valor inicial aos respetivos centroides. Já a matriz de covariâncias assume-se como a matriz identidade, uma vez que não se possui informação sobre possíveis dependências entre as componentes do vetor. Em relação às probabilidades a priori, assume-se que os pontos se encontram distribuídos do mesmo modo entre os clusters e, como tal, cada cluster terá um peso igual a $\frac{1}{k}$ no modelo misto gaussiano.

O algoritmo converge quando se verifica $\max_{1 \leq i \leq k} \left\{ \|\boldsymbol{z}_i^t - \boldsymbol{z}_i^{t-1}\|_2^2 \right\} \leq \epsilon$, onde $\epsilon \geq 0$ é o limite de convergência e $t$ é a iteração onde se encontra.

O pseudo-código deste método encontra-se apresentado no apêndice B no algoritmo 2. No apêndice A na figura A.2 encontra-se um exemplo do resultado obtido com este algoritmo.

### 2.2.4. Ordem de complexidade

No algoritmo *Expectations-Maximization* cada iteração $t$ é composta por duas fases: a fase expectativas e a fase maximização. Na fase de expectativas é necessário inverter a matriz das covariâncias $\boldsymbol{\Sigma}_i$ e, para isso, é preciso calcular o determinante $|\boldsymbol{\Sigma}_i|$ operação essa que leva um tempo de $O(d^3)$. Esta inversão será executada para

todos os $k$ clusters, logo a fase demora um total de $O(kd^3)$. Também nesta fase é necessário avaliar a função densidade $f(x|z_k, \Sigma_k)$ o que leva um tempo de $O(d^2)$. Esta função vai ser realizada para todos os $n$ pontos e para todos os $k$ clusters e no total necessita de $O(knd^2)$. Na fase de maximização, o tempo necessário é dominado pela atualização da matriz das covariâncias $\boldsymbol{\Sigma}_i$ que necessita de um tempo $O(nkd^2)$ para todos os $k$ clusters. Como cada iteração $t$ é composta por estas duas fases, então a complexidade computacional deste algoritmo é de $O(tnkd^2)$, isto porque, $n > k$.

## 2.3. Kernel K-means

O algoritmo *K-means* faz uma separação rígida dos elementos e, por essa razão, na presença de conjuntos não convexos, perde a sua eficiência. Surge assim a necessidade de modificar o algoritmo de forma a resolver este problema. Ao aplicar o Kernel no *K-means*, este algoritmo torna-se capaz de separar os pontos em clusters de forma não linear.

### 2.3.1. Introdução aos métodos de Kernel

A partir de um conjunto de dados $\boldsymbol{D}$, interessa encontrar o mapeamento $\phi$ para mapear o elemento $\boldsymbol{x}_j$ do espaço de entrada $\mathcal{I}$ para o espaço característico $\mathcal{F}$, sendo $\phi(\boldsymbol{x}_j)$ a sua representação.

Como a transformação dos elementos do conjunto $\boldsymbol{D}$ do espaço de entrada para o espaço característico é um processo demorado, usam-se os métodos de kernel que evitam a transformação direta dos elementos. Este processo dá-se através dos $n \times n$ valores de semelhança dos pares $(\boldsymbol{x}_j, \boldsymbol{x}_i)$, e para medir a semelhança de pares usa-se uma função denominada Kernel. Esta define-se tendo em conta que $\mathcal{I}$ é o espaço de entrada e o conjunto $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{I}$.

**Definição 4 (Função Kernel $K$)** *Uma função Kernel é uma função $K : \mathcal{I} \times \mathcal{I} \longrightarrow \mathbb{R}$ que satisfaz o seguinte:*

$$K(\boldsymbol{x}_j, \boldsymbol{x}_i) = \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_i) \tag{2.24}$$

*onde $\phi : \mathcal{I} \to \mathcal{F}$ é a função que realiza o mapeamento dos elementos de $\boldsymbol{D}$.*

Dado que esta função apenas utiliza produtos internos torna-se simples reconstruir $\phi(\boldsymbol{x}_j)$ sem precisar de passar do espaço de entrada para o espaço característico todos os elementos de $\boldsymbol{D}$. As semelhanças entre os pontos de $\boldsymbol{K}$ são representadas por uma matriz Kernel de dimensão $n \times n$ definida pelo o seguinte:

$$\boldsymbol{K} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

onde $K$ é a função kernel responsável por atribuir um valor à semelhança entre o par $(\boldsymbol{x}_j, \boldsymbol{x}_i)$.

É preciso ter em atenção que a função Kernel não pode ser escolhida arbitrariamente. Esta tem de verificar as seguintes condições de forma a que a equação (2.24) continue válida:

- A função Kernel deve ser simétrica : $K(\boldsymbol{x}_j, \boldsymbol{x}_i) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
- A matriz Kernel proveniente é semidefinida positiva para qualquer subconjunto $\boldsymbol{D} \in \mathcal{I}$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0,\ a_i, a_j \in \mathbb{R},\ i, j = 1, \ldots, n$$

Neste trabalho foi escolhida a seguinte função Kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right\}$ que se denomina por Kernel Gaussiano. Após esta breve introdução aos Kernels, vai-se abordar a descrição do algoritmo *Kernel K-means*.

### 2.3.2. Descrição do algoritmo

Seja $\boldsymbol{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$ a matriz kernel de dimensão $n \times n$ simétrica, onde $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. Seja $\{C_1, \ldots, C_k\}$ a partição de $n$ pontos em $k$ clusters e as $k$ médias de cada cluster $\left\{\boldsymbol{z}_1^{\phi}, \ldots, \boldsymbol{z}_k^{\phi}\right\}$, onde:

$$\boldsymbol{z}_i^{\phi} = \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in C_i} \phi(\boldsymbol{x}_j)$$

é a média do cluster $C_i$ no espaço característico com $n_i = |C_i|$.

No *Kernel K-means* não é necessário conhecer os centroides como no *K-means*, porque no espaço característico é possível obter uma função objetivo do *Kernel K-means*($SSE$) através do uso de funções Kernels, sendo esta equivalente à função $SSE$ do *K-means*. Observe-se então a função objetivo do *Kernel K-means*:

$$\min_{\mathcal{C}} SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \|\phi(\boldsymbol{x}_j) - \boldsymbol{z}_i^{\phi}\|^2$$

Desenvolvendo a equação anterior através de funções Kernel, tem-se o seguinte:

$$\begin{aligned}
SSE(\mathcal{C}) &= \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \|\phi(\boldsymbol{x}_j) - \boldsymbol{z}_i^{\phi}\|^2 \\
&= \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \left( \|\phi(\boldsymbol{x}_j)\|^2 - 2\phi(\boldsymbol{x}_j)^T \boldsymbol{z}_i^{\phi} + \|\boldsymbol{z}_i^{\phi}\|^2 \right) \\
&= \sum_{i=1}^{k} \left( \left( \sum_{\boldsymbol{x}_j \in C_i} \|\phi(\boldsymbol{x}_j)\|^2 \right) - 2n_i \left( \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in C_i} \phi(\boldsymbol{x}_j) \right)^T \boldsymbol{z}_i^{\phi} + n_i \|\boldsymbol{z}_i^{\phi}\|^2 \right) \\
&= \left( \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_j) \right) - \left( \sum_{i=1}^{k} n_i \|\boldsymbol{z}_i^{\phi}\|^2 \right) \\
&= \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b) \\
&= \sum_{j=1}^{n} K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b).
\end{aligned}$$

A função $SSE$ anterior mede a distância no espaço característico entre um ponto $\boldsymbol{x}_j$ e um cluster $C_i$, com o objetivo de descobrir o cluster mais próximo para se lhe atribuir $\boldsymbol{x}_j$. Para tal, é necessário calcular o centroide de cada cluster no espaço característico através dos Kernels.

De seguida, apresenta-se o cálculo da distância de um ponto ao centroide no espaço característico usando as funções Kernels:

$$\begin{aligned}
\|\phi(\boldsymbol{x}_j) - \boldsymbol{z}_i^{\phi}\|^2 &= \|\phi(\boldsymbol{x}_j)\|^2 - 2\phi(\boldsymbol{x}_j)^T \boldsymbol{z}_i^{\phi} + \|\boldsymbol{z}_i^{\phi}\|^2 \\
&= \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in C_i} \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_a) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} \phi(\boldsymbol{x}_a)^T \phi(\boldsymbol{x}_b) \\
&= K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)
\end{aligned}$$

Para a atribuição $C_i^*(\boldsymbol{x}_j)$ do ponto $\boldsymbol{x}_j$ ao cluster $C_i$ escolhe-se o menor valor da equação anterior, isto é:

$$\begin{aligned}
C_i^*(\boldsymbol{x}_j) &= \underset{i}{\operatorname{argmin}} \left\{ \|\phi(\boldsymbol{x}_j) - \boldsymbol{z}_i^{\phi}\|^2 \right\} \\
&= \underset{i}{\operatorname{argmin}} \left\{ K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b) \right\} \\
&= \underset{i}{\operatorname{argmin}} \left\{ -\frac{2}{n_i} \sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b) \right\}
\end{aligned} \tag{2.25}$$

O termo $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ é igual em todos os clusters, logo não tem influência na sua atribuição e desaparece. O último termo é a norma quadrada da média de cada cluster $C_i$ ($sqnorm_i$) e o primeiro termo é duas vezes a média do valor do kernel ($avg$) para cada $\boldsymbol{x}_j$ em cada $C_i$.

Este algoritmo inicia-se a partir de uma atribuição aleatória dos elementos pelos $k$ clusters. Posteriormente reatribuiu-se iterativamente cada ponto à média mais próxima no espaço característico através da equação (2.25) sendo feito o cálculo separado do $sqnorm$ e do $avg$. Esta reatribuição é usada numa nova repartição de pontos que irá formar novos clusters. Este processo é repetido até convergir e, para que tal aconteça, é necessário que os pontos nos clusters não mudem de uma iteração para a outra. Para se poder obter uma convergência mais rápida, opta-se pela aplicação de um erro $\epsilon > 0$ na comparação dos cluster de diferentes iterações. O *Kernel K-means* converge quando $1 - \frac{1}{n}\sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}| \leq \epsilon$ para uma iteração $t$.

O pseudo-código deste método está apresentado no apêndice B no algoritmo 3. No apêndice A, na figura A.3, encontra-se um exemplo do resultado obtido com este algoritmo.

### 2.3.3. Ordem de complexidade

O cálculo da matriz Kernel para todos os $k$ clusters $C_i$ leva um tempo de $O(n^2)$. Para calcular o $sqnorm_i$ e o $avg$ para todos os $k$ clusters também é necessário um tempo de $O(n^2)$ em ambos. Por último, é necessário calcular a distância da média de cada cluster em relação a cada ponto com o objetivo de o atribuir ao cluster com a média mais próxima. Logo, o total para a ordem de complexidade deste algoritmo é de $O(tn^2)$ onde $t$ é o número de iterações até convergir.

## 2.4. Outros algoritmos

Nesta secção referem-se outros algoritmos importantes na área do clustering. O seu estudo não foi aprofundado e surge apenas como uma informação complementar.

### 2.4.1. Hierárquico

O algoritmo Hierárquico gera partições sequenciais através de duas estratégias, uma por aglomeração de elementos e outra por divisão de elementos. Tendo por base um conjunto $\boldsymbol{D} = \{\boldsymbol{x}\}_{i=1}^{n} \in \mathbb{R}^d$, este algoritmo cria clusters sequenciais que, por norma, serão representados por uma árvore de clusters.

Na abordagem por aglomeração, percorre-se a árvore de clusters desde a sua base até ao topo. Inicialmente cada um dos $n$ pontos pertence a um cluster diferente e o algoritmo irá recursivamente aglomerar quais os clusters semelhantes até existir apenas um, que contém todos os elementos.

Na abordagem por divisão, percorre-se a árvore de clusters do topo até à sua base. Inicialmente todos os elementos estão contidos no mesmo cluster e o algoritmo irá recursivamente dividir o cluster inicial através da sua dissemelhança até todos os $n$ pontos estarem contidos em $n$ clusters diferentes.

### 2.4.2. DBSCAN

Este algoritmo separa os elementos $\boldsymbol{x}_j \in \mathbb{R}^d,\ j = 1, \ldots, n$ de um conjunto de dados $\boldsymbol{D}$ em função da sua densidade. Para isso, define-se como sendo o conjunto de elementos de $\boldsymbol{D}$ que estão numa bola centrada no ponto $\boldsymbol{x}$, com um raio $\epsilon$, que se representa da seguinte forma:

$$N_\epsilon(\boldsymbol{x}) = \{\boldsymbol{y} | \delta(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\}$$

onde $\delta(\boldsymbol{x}, \boldsymbol{y})$ mede a distância entre o ponto $\boldsymbol{x}$ e o ponto $\boldsymbol{y}$.

O *DBSCAN* inicialmente calcula para os $n$ pontos $\boldsymbol{x}$ os respetivos $N_\epsilon(\boldsymbol{x})$, com o objetivo de identificar os pontos cores. Neste algoritmo, os pontos do conjunto $\boldsymbol{D}$ podem ser classificados de três formas:

1. Ponto core:

   Denomina-se de ponto core o ponto $\boldsymbol{x}$, onde $|N_\epsilon(\boldsymbol{x})| \geq MinimodePontos$. $MinimodePontos$ é uma quantidade estabelecida a priori de um número mínimo de pontos que se encontram na bola centrada em $\boldsymbol{x}$.

2. Ponto de fronteira:

   Denomina-se de ponto de fronteira o ponto que verifica $1 < |N_\epsilon(\boldsymbol{x})| < MinimoPontos$.

3. Ponto isolado:

   Denomina-se de ponto isolado todo o ponto que não seja um ponto core ou um ponto de fronteira, isto é, pontos que se encontrem isolados no conjunto $\boldsymbol{D}$.

Ao contrário dos outros algoritmos, este não atribui a priori os $n$ pontos aos clusters. Para a construção dos clusters, este algoritmo apresenta duas abordagens:

1. Alcance de densidade direta

   Diz-se que o ponto $\boldsymbol{x}$ alcança $\boldsymbol{y}$ por densidade direta se $y \in N_\epsilon(\boldsymbol{x})$.

2. Alcance de densidade

   Diz-se que $\boldsymbol{x}$ alcança $\boldsymbol{y}$ por densidade se existe uma ligação de pontos cores $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l = \boldsymbol{y}$, onde o ponto $\boldsymbol{x}_{j-1}$ alcança diretamente o ponto $\boldsymbol{x}_j$ para todo o $j = 1, \ldots, l$.

Nesta fase, o algoritmo escolhe um ponto core e verifica recursivamente quais são os pontos ligados pelo alcance de densidade e pela densidade direta atribuindo-lhes um cluster. De seguida percorrem-se os pontos core que não estão atribuídos a um cluster verificando recursivamente aqueles que estão ligados entre si por densidade. Desta forma recursiva, identificam-se os cluster até se esgotarem os pontos core não atribuídos a um cluster. Por fim, os pontos isolados que não foram abrangidos por densidade constituem um novo cluster entre si.

### 2.4.3. K Nearest Neighbors

O *K Nearest Neighbors*($KNN$) utiliza um conjunto auxiliar chamado de conjunto de aprendizagem (training data set), que permitirá identificar à partida os elementos que definem cada cluster. Esta identificação é feita através de diferentes níveis de semelhança entre os elementos dos dois conjuntos, dependendo do conjunto que se pretende estudar.

Inicialmente, o algoritmo classifica todos os elementos do conjunto a ser estudado, que sejam semelhantes aos elementos do conjunto de aprendizagem. Após este processo, haverá ainda elementos que não foram classificados por não serem considerados semelhantes aos elementos do conjunto de aprendizagem. Nestes elementos não classificados, estudam-se os seus $k$ vizinhos que estão a uma determinada distância $\epsilon$ a priori e atribui-se a classificação verificada na maioria dos $k$ vizinhos mais próximos. Desta forma, são separados os elementos do conjunto a ser estudado, criando-se clusters através dos elementos que contêm a mesma classificação.

# Capítulo 3

# Resultados computacionais

Este capítulo pretende ser um breve estudo do comportamento dos algoritmos através de uma avaliação experimental comparativa de desempenho. Tem-se por objetivo comparar o algoritmo *K-means* com a sua variante *Kernel K-means* e com o algoritmo *Expectation-Maximization*. Tanto os conjuntos de dados como a avaliação experimental dos algoritmos e os próprios algoritmos foram implementados em matlab.

## 3.1. Medidas de avaliação dos clusters

O objetivo da avaliação dos clusters é determinar a qualidade e a estabilidade do clustering. Neste trabalho, consideram-se dois tipos de avaliações: avaliações externas e avaliações internas.

### 3.1.1. Avaliações externas

As avaliações externas partem do pressuposto que se conhece o particionamento dos dados. Portanto, através destes métodos, é possível validar o clustering de um conjunto de dados obtido através de um dos algoritmos estudados anteriormente. Estas avaliações usam uma tabela de contingência $\boldsymbol{N}$(de dimensão $k \times r$), que contém o número de interseções entre cada cluster e de cada partição permitindo assim avaliar a sua semelhança.

Em [13] estudaram-se várias avaliações deste tipo tais como a pureza, o coeficiente de Jaccard, a estatística de rand, $SSE$ e a medida de Fowlkes-Mallows, sendo a estatística de Rand a que revelou melhores resultados.

A estatística de Rand tem como objetivo avaliar a semelhança do clustering com uma partição conhecida a priori. Nesta avaliação deve ter-se em conta todos os pares de pontos que se encontram bem classificados, isto é, os pares que estão no mesmo cluster na mesma partição ou os pares que não estão no mesmo cluster nem na mesma partição. Desta forma, cria-se um rácio entre os pares de pontos corretamente

colocados e todos os pares de pontos do conjunto. Este rácio varia entre zero e um, onde um significa que o cluster coincide com a partição e zero significa que os clusters obtidos não estão de acordo com a partição conhecida.

### 3.1.2. Avaliações internas

Ao contrário das avaliações externas, estas avaliações não necessitam de conhecer um particionamento a priori, que é uma situação mais corrente no mundo real. Para este tipo de avaliações, usam-se medidas de semelhança entre os clusters e dentro dos clusters. Para estas avaliações, usam-se matrizes de distância com dimensão $n \times n$ entre os $n$ pontos. Devido à simetria destas matrizes, apenas se tem em conta os elementos acima da diagonal principal para o cálculo das avaliações internas.

De novo, estudaram-se várias avaliações deste tipo, tais como, o betacv, o coeficiente de silhueta e a medida $SSE$. Aqui, decidiu-se também usar apenas o coeficiente de silhueta, porque em [13] foi a avaliação que demonstrou ter os melhores resultados. O coeficiente de silhueta avalia a coesão dos clusters através da proximidade de um ponto $x_i$ aos pontos $x_j$ do cluster $C_i$ onde pertence e a proximidade do mesmo ponto aos pontos do cluster mais próximo $C_j$. O valor deste coeficiente varia entre $[-1, 1]$: o valor 1 significa que $x_i$ está mais próximo dos pontos do seu cluster do que dos pontos do cluster mais próximo e $-1$ representa a situação contrária. Por fim, de modo a perceber se os cluster estão bem coesos, faz-se uma média dos coeficientes de silhueta de todos os pontos.

## 3.2. Desempenho dos algoritmos

Nesta secção, estuda-se o comportamento do algoritmo *K-Means* em relação ao *Expetation-Maximization* e a variante *Kernel K-Means*. Os testes são elaborados com as seguintes classes de conjuntos:

- Classe 1: O conjunto de pontos encontra-se distribuído em três nuvens espalhadas aleatoriamente à volta do respetivo centro. Nesta classe de problemas existem três variantes: nuvens afastadas (A.4), nuvens aproximadas (A.5) e nuvens intersectadas (A.6). Neste tipo de problemas, pretende-se avaliar o efeito do aumento do número de clusters nos resultados dos algoritmos.

- Classe 2: Nesta classe de problemas, os pontos estão distribuídos em vários núcleos densos e um anel exterior envolvendo os núcleos. Foram considerados

conjuntos com um núcleo no centro (A.7), dois núcleos no centro (A.8) e três núcleos no centro (A.9). Neste tipo de problemas, mantém-se o número de clusters igual ao número de partições e pretende-se avaliar o desempenho dos algoritmos com o aumento do número de partições.

- Classe 3: Neste tipo de classe de problemas pretendem-se aplicar os algoritmos à segmentação de imagens, que poderão ser a preto e branco ou a cores. Neste trabalho optaram-se por imagens de dimensões reduzidas (a maior tem $129 \times 79$ pixeis) de modo a ter um tempo de calculo reduzido. Aqui as partições não são conhecidas a priori e pretende-se avaliar o desempenho dos algoritmos em função do aumento do número de clusters, o que se traduz numa separação de um maior número de tonalidades.

### Classe 1

Para analisar o comportamento dos algoritmos e a qualidade dos clusters daí provenientes, optou-se neste trabalho por gerar cinco vezes o mesmo tipo de conjunto, para que os resultados não dependam da distribuição inicial dos pontos no conjunto. Em cada conjunto gera-se três nuvens de pontos em que a primeira é gerada com cem pontos, a segunda com 150 pontos e a terceira com 250 pontos. Para cada um dos conjuntos no estudo variou-se o número de clusters entre dois e cinco. Para cada variação de clusters, foram geradas trinta repetições, cada uma delas inicializada de forma diferente, e calculou-se a média dos resultados obtidos para cada uma das variações de clusters. Estas repetições têm como objetivo estimar os valores do tempo de execução, do número de iterações, da estatística de Rand e do coeficiente de silhueta. No final, obteve-se a média das 150 repetições de cada uma das avaliações dos algoritmos para cada variação de clusters para os diferentes conjuntos gerados.

### Nuvens Afastadas

No conjunto de gráficos da figura 3.1 podem comparar-se as quatro avaliações mencionadas anteriormente. No primeiro gráfico, referente ao número de iterações que cada algoritmo necessita até convergir, nota-se que os algoritmos *EM* e *Kernel K-means*, a partir dos três clusters, necessitam de um maior numero de iterações. Isto acontece porque o conjunto em estudo contém apenas três partições. Os resultados apresentados no segundo gráfico relativos ao tempo de execução de cada algoritmo são expectáveis tendo em conta o resultado obtido no primeiro gráfico. No gráfico

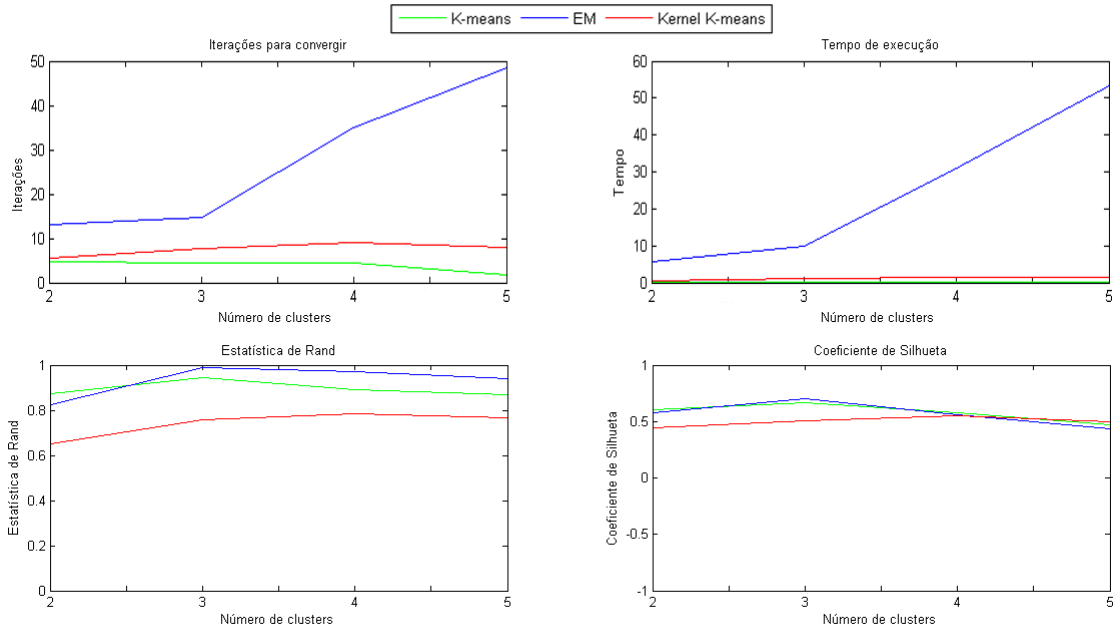



Figura 3.1: Gráfico com as avaliações em relação a um conjunto de Nuvens afastadas

referente à estatística de Rand, observa-se que a melhor aproximação à partição conhecida a priori é alcançada na combinação com três clusters. Isso acontece porque a partição conhecida a priori contém também ela três clusters. Por último, o gráfico relativo ao coeficiente de silhueta mostra que, mais uma vez, a combinação com três clusters é a mais coesa.

Através da observação destes gráficos, conclui-se que o algoritmo *K-means* será a melhor escolha a aplicar neste tipo de conjuntos. Isto porque, abdicando um pouco da semelhança em relação à partição a priori, pode-se obter uma convergência muito mais rápida. Já o *Kernel K-means* não apresentou um bom desempenho neste tipo de classes. É também de salientar que, nas avaliações sobre a qualidade do clustering, se obteve o melhor resultado com três clusters como seria de esperar dadas as características do conjunto utilizado.

### Nuvens aproximadas

Nesta variação de classe de problemas, obtiveram-se gráficos que revelam comportamentos semelhantes ao caso anterior. Salienta-se, no entanto, que o número de iterações e o tempo de execução aumentou quase para o dobro. Este aumento explica-se pela posição periférica de alguns pontos de diferentes nuvens que, por se encontrarem próximos entre si, dificultam a separação realizada pelos algoritmos.

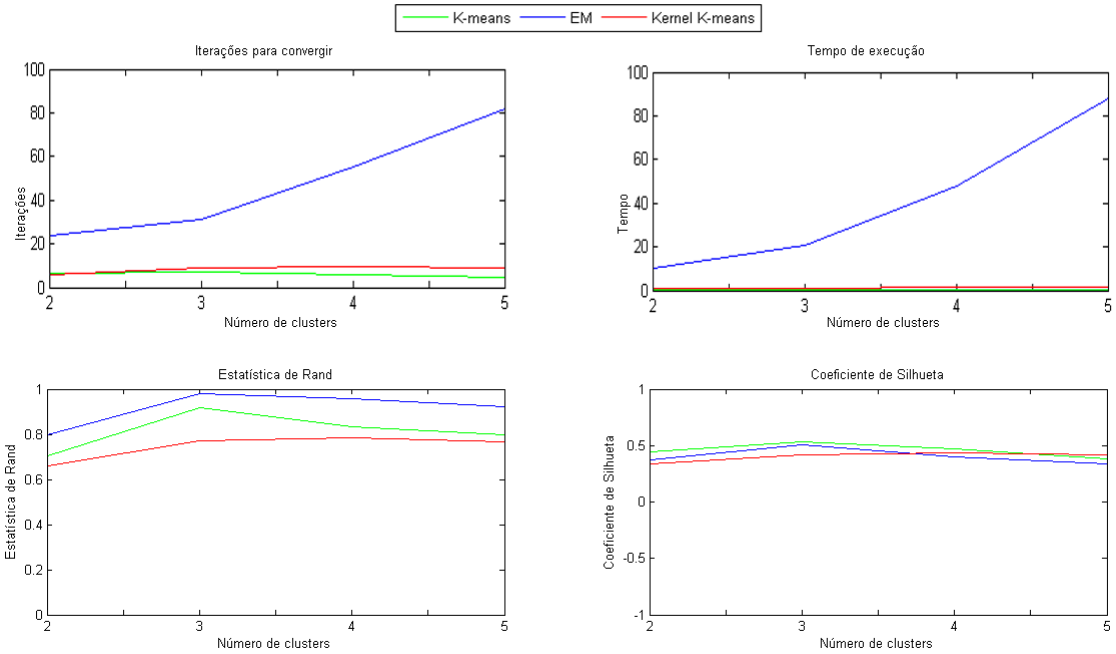



Figura 3.2: Gráfico com as avaliações em relação a um conjunto de Nuvens aproximadas

Além disso, o valor da estatística de Rand no algoritmo *K-means* diminui já que existem alguns pontos misturados e também porque este executa uma separação rígida através de hiperplanos. Como o *EM* aplica uma separação não rígida consegue manter o clustering próximo à partição conhecida à priori.

### Nuvens intersetadas

Nos dois primeiros gráficos houve um decréscimo de iterações, o que consequentemente resultou num tempo de execução menor. Isto deve-se ao facto de, neste conjunto, existirem três nuvens intersectadas cujos limites são pouco definidos. Apesar desta má separação, estes algoritmos recorrem a menos iterações. Os resultados na estatística de Rand não foram bons porque, como se pode ver na imagem A.6, os elementos das diferentes partições estão sobrepostos, logo os algoritmos não foram capazes de recuperar clusters semelhantes à partição conhecida a priori. Já o coeficiente de silhueta é baixo porque todos os clusters se encontram muito próximos. Deste modo, existem pontos que estão menos distantes dos elementos do cluster mais próximo do que dos elementos do próprio cluster.

Conclui-se que, numa distribuição de pontos como a que se apresenta, todos os elementos são demasiadamente semelhantes, o que faz com que os algoritmos não

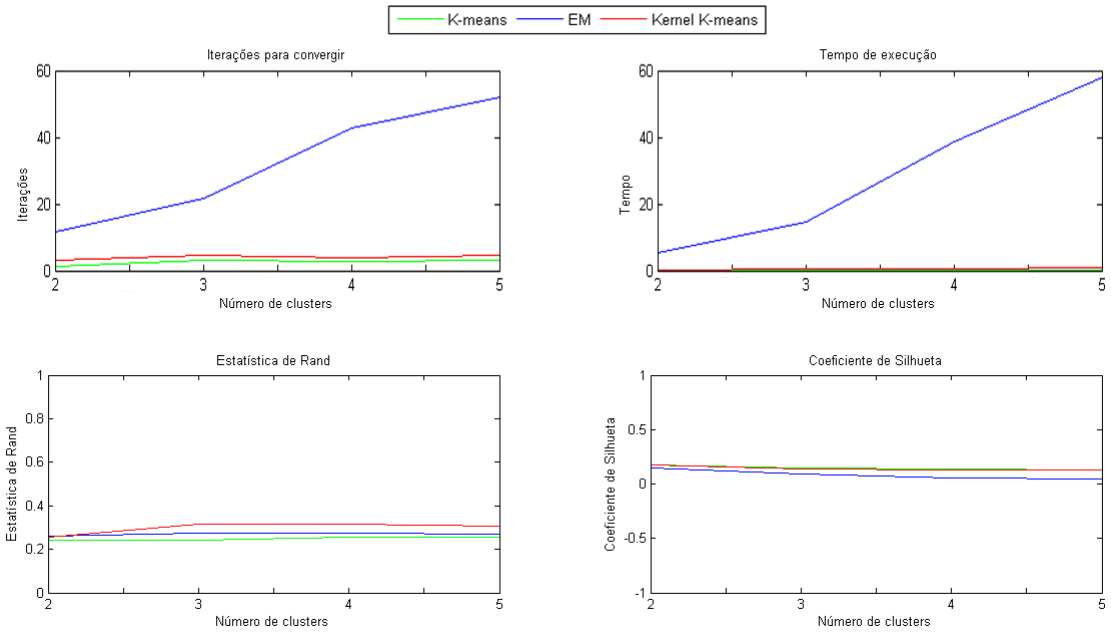



Figura 3.3: Gráfico com as avaliações em relação a um conjunto de Nuvens intersectadas

consigam fazer uma separação eficaz. Neste caso, a melhor abordagem possível seria mesmo considerar apenas um cluster.

### Classe 2

Nesta classe, os problemas foram gerados de uma forma semelhante ao descrito na classe 1 excluindo a variação de clusters no estudo em cada um dos cinco conjuntos gerados. Isto é, o número de clusters para cada um é fixo, sendo este igual ao número de partições do conjunto. No caso do conjunto com um núcleo foram precisos duzentos pontos no núcleo mais quatrocentos no anel exterior. No conjunto com dois núcleos foram precisos duzentos pontos para cada núcleo e oitocentos pontos para a formação do anel exterior. Por último, no conjunto com três núcleos, estes foram gerados com duzentos pontos cada um e o anel exterior foi gerado com mil e cem pontos.

É possível verificar no primeiro gráfico que o *EM* é o algoritmo que necessita de mais iterações até convergir, quando aplicado em conjuntos de dois e três núcleos. À medida que o número dos elementos no conjunto de dados aumenta, os algoritmos necessitam de mais iterações para percorrerem todo o conjunto. Observa-se, então, uma grande diferença no número de iterações necessárias num conjunto com

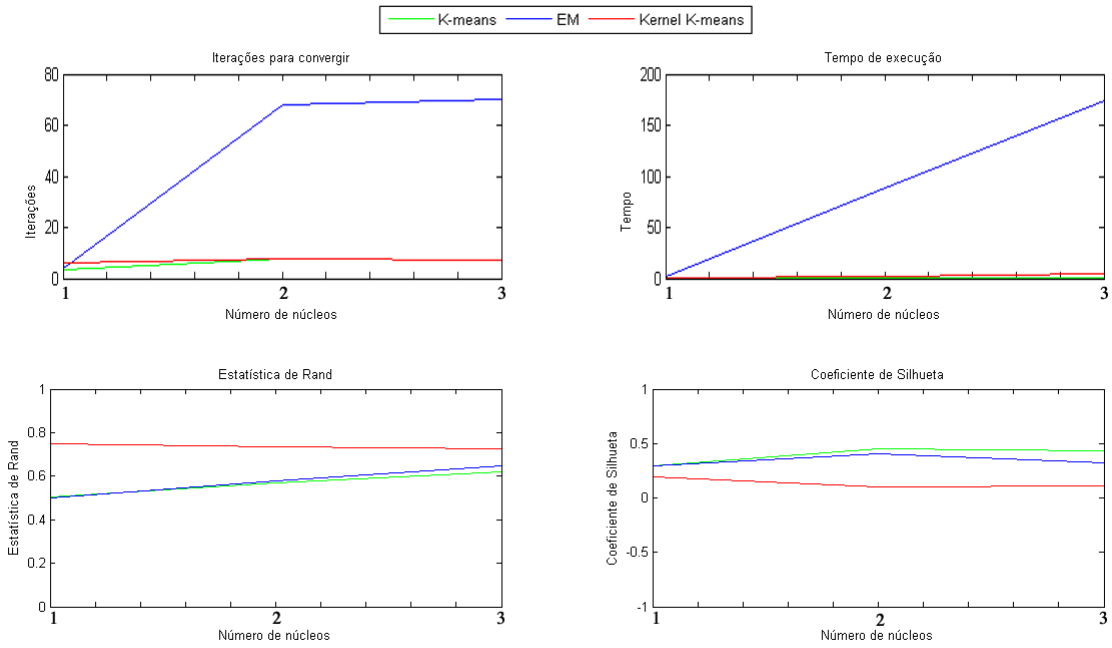



Figura 3.4: Gráfico com as avaliações em relação aos conjuntos do tipo 2

um núcleo face aos conjuntos de dois e três núcleos, dado o acréscimo de dados. Por outro lado a diferença entre os conjuntos de dois e três núcleos não é tão expressiva porque o aumento de dados é menor. Esta diferença explica-se porque foi necessário introduzir novos pontos para a formação dos novos núcleos. Também o redimensionamento do anel exterior exigiu a introdução de mais pontos de modo a aumentar de tamanho e manter a sua coesão. Deste modo, houve um maior aumento de pontos entre o conjunto com um núcleo e o conjunto de dois núcleos. O aumento de pontos verificado entre os conjuntos de dois e três núcleos é relativamente menor e as iterações dos algoritmos refletem isso mesmo. Os resultados do gráfico relativo ao tempo de execução dos algoritmos *K-means* e *Kernel K-means* têm um comportamento semelhante ao do gráfico anterior. Já o *EM* apresenta um comportamento linear devido ao aumento linear de pontos entre conjuntos. No gráfico da estatística de Rand, observa-se que, à medida que são acrescentados núcleos no conjunto, o desempenho do *K-means* e do *EM* aproximam-se ao do *Kernel K-means*. Já nos resultados do coeficiente de silhueta, observa-se que o *Kernel K-means* apresenta um coeficiente pior do que o dos outros algoritmos. De facto, tendo em conta que o coeficiente de silhueta é a média dos coeficientes de silhueta de todos os elementos, é de notar que o anel exterior contém pontos cuja distância aos pontos do cluster mais próximo é muitas vezes menor do que a distância entre os pontos desse mesmo

cluster. Deste modo, é compreensível o baixo valor do coeficiente de silhueta proveniente deste algoritmo e conclui-se que esta avaliação não é a mais adequada para este tipo de conjuntos.

Pode concluir-se que o *Kernel K-means* é o algoritmo mais adequado para este tipo de conjuntos porque consegue separar os núcleos em clusters distintos e o anel exterior num outro. O algoritmo *K-means* não o consegue fazer porque a divisão que opera, através de hiperplanos rectos, não lho permite e o *EM* dificilmente o consegue, porque este algoritmo não é capaz de fazer uma separação não linear.

## 3.3. Aplicação em segmentação de imagens

Nesta secção, foram estudados conjuntos de imagens tanto a preto e branco como a cores. Nas imagens a cores, cada pixel contém 5 componentes sendo os dois primeiros coordenadas $(x,y)$ e os restantes três componentes $RGB$ $(r,g,b)$, que determinam a sua cor. Nas imagens a preto e branco, cada pixel contém apenas três componentes: dois referentes às coordenadas e um terceiro que define a intensidade da cor.

Ao transformar a imagem no conjunto de pontos $\boldsymbol{D}$, utilizou-se uma variável peso $p$ multiplicada pelas componentes da cor para realçar a sua relevância em relação às componentes das coordenadas. As coordenadas $x$ e $y$ variam consoante a largura e o comprimento da figura e as componentes das cores $r$,$g$ e $b$ variam entre zero e duzentos e cinquenta e cinco. Assim, para se dar mais relevância às cores, escolheu-se uma variável $p$ que vai multiplicar pelas componentes $RGB$. Nesse sentido testaram-se alguns valores para $p$ num número fixo de clusters de forma a encontrar o que se ajustava melhor. Variou-se o valor de $p$ numa figura a preto e branco com várias tonalidades de cinza e com um tamanho de $129 \times 79$ que gerou um conjunto $\boldsymbol{D}$ com 10191 pontos.

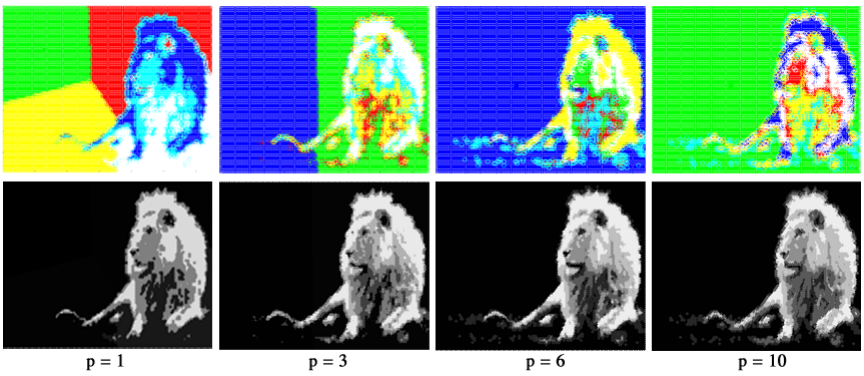



Figura 3.5: Estudo feito à variável $p$.

Como se pode observar na figura 3.5 a melhor separação do conjunto $\boldsymbol{D}$ verificou-se no $p = 10$. Não se escolheu um $p$ maior para não dar o domínio às componentes do $RGB$.

De seguida aplicaram-se diferentes algoritmos à imagem mencionada anteriormente.

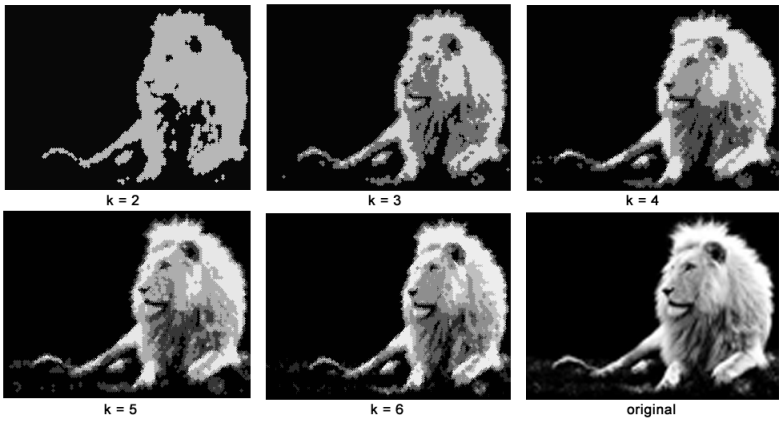



Figura 3.6: Exemplo do Algoritmo *K-means* aplicado ao problema de segmentação de uma imagem a preto e branco.

Nesta figura, atribuiu-se um peso igual a dez, e apenas se obtiveram resultados satisfatórios utilizando o algoritmo *K-means*. Atendendo que as componentes da cor continham muitos gradientes de intensidade, não houve coesão suficiente para o *Kernel K-means* fazer uma boa separação. No caso do *EM*, a dispersão levou-o a colocar tudo num cluster ou em dois clusters de dimensões bastante distintas.

O *Kernel K-means* e o *EM* não geraram bons resultados quando aplicados a uma imagem com muitos gradientes de intensidade de cor. Optou-se então por aplicar estes mesmos algoritmos numa imagem de um desenho animado de tamanho $47 \times 90$ com cores mais sólidas e sem grandes variações de tonalidade gerando um conjunto $\boldsymbol{D}$ com 4230 pontos. Aqui usou-se mais uma vez um peso igual a dez.

Mais uma vez os algoritmos *Kernel K-means* e *EM* não obtiveram bons resultados. Analizando a figura mais pormenorizadamente, observou-se que a imagem não possuía cores claramente definidas. Os contornos da figura continuam com alguns pixeis de diferentes tonalidades, o que fez com que as componentes das cores se tornassem dispersas.

Por fim, testaram-se os algoritmos numa imagem de tamanho $108 \times 73$ gerando um conjunto $\boldsymbol{D}$ com 7884 pontos. Esta figura, mais simples, é constituída somente

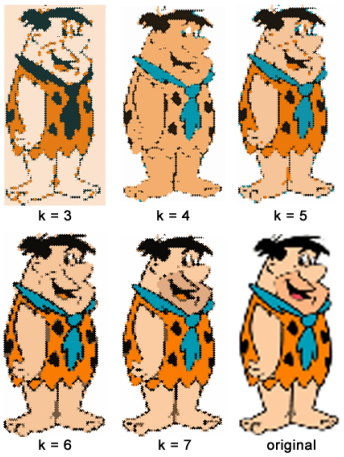



Figura 3.7: Exemplo do Algoritmo *K-means* aplicado ao problema de segmentação de uma imagem a cores.

por três cores sólidas e sem ruído nas zonas de fronteira.

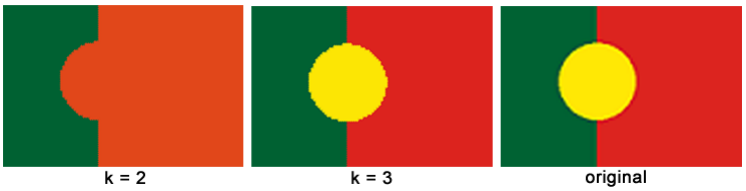



Figura 3.8: Exemplo do Algoritmo *K-means* aplicado ao problema de segmentação de uma imagem a cores.

Mais uma vez foi o *K-means* que gerou melhores resultados. Neste caso, as componentes das cores estavam coesas, mas as componentes das coordenadas não, o que fez com que os dados estivessem suficientemente dispersos para que os outros dois algoritmos não obtivessem bons resultados.

# Capítulo 4

# Conclusão

Neste trabalho estudou-se o comportamento de três algoritmos com abordagens bastante diferentes face à separação de elementos de um conjunto. O algoritmo *K-means* separa os elementos através de hiperplanos tendo em conta a distância dos elementos do conjunto aos seus centroides. O *EM* separa os elementos através da maior probabilidade destes pertencerem a um cluster tendo em conta um modelo misto gaussiano. Finalmente, o *Kernel K-means* diferencia os elementos através da sua coesão em diferentes clusters.

Ficou demonstrado que o *EM* obtém bons resultados em conjuntos cujos elementos se organizam em torno de um centro, formando nuvens com alguma dispersão. Neste tipo de conjuntos, o *K-means* conseguiu obter resultados satisfatórios a nível da qualidade dos clusters e foi mais rápido que o *EM*. O *Kernel K-means* revelou bastante dificuldade em separar este tipo de conjuntos. No entanto, em conjuntos onde existiam núcleos coesos de elementos, este foi o algoritmo que teve a melhor prestação, necessitando do mesmo tempo que o *K-means*. Nesta classe de conjuntos, o *K-means* e o *EM* obtêm uma separação razoável e, à medida que o número de núcleos aumenta, o seu comportamento aproxima-se do *Kernel K-means*. Salienta-se que o *EM* leva muito mais tempo do que os outros dois algoritmos até convergir. Por fim, na segmentação de imagens, o *K-means* foi o único dos três algoritmos que conseguiu obter resultados consistentes. Desta forma, o *K-means* demonstrou ser o algoritmo mais robusto, sendo o algoritmo de referência no caso de não haver informação à partida sobre a composição dos elementos do conjunto.

Num trabalho futuro, será interessante estudar os outros três algoritmos que não foram aprofundados e aplicá-los a vários tipos de problemas. Nomeadamente, a aplicação do *Hierárquico* nas soluções proveniente dum problema multicritério, o *DBSCAN* na segmentação de imagens e o *KNN* aplicado ao text mining. Além disso seria igualmente importante estudar algoritmos mais vocacionados para a separação de grandes volumes de informação.

# Apêndice A

# Inclusão de imagens

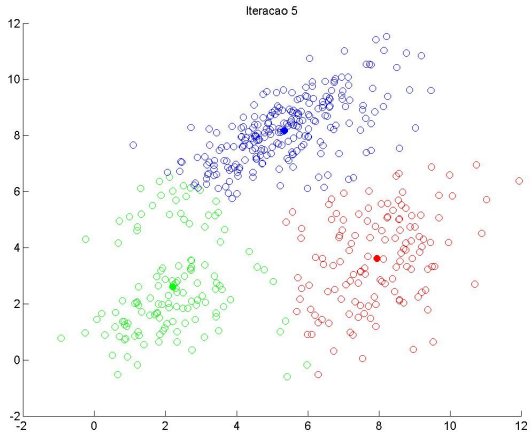



Figura A.1: Exemplo do Algoritmo *K-Means*

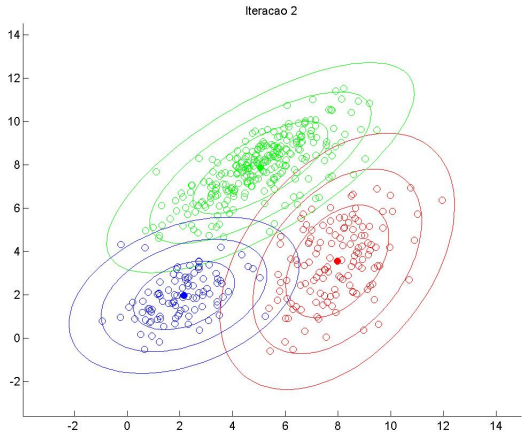



Figura A.2: Exemplo do Algoritmo *Expetation-Maximization*.

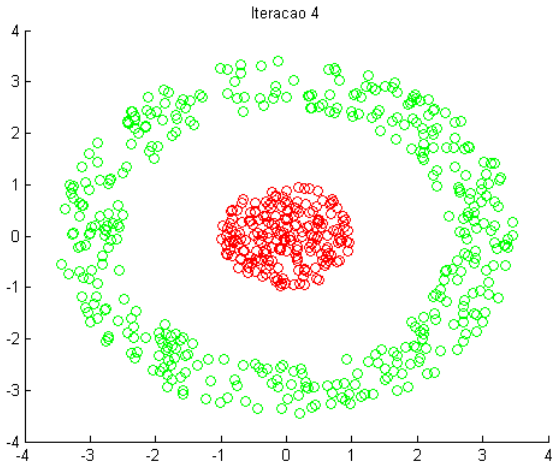



Figura A.3: Exemplo do Algoritmo *Kernel K-means*.

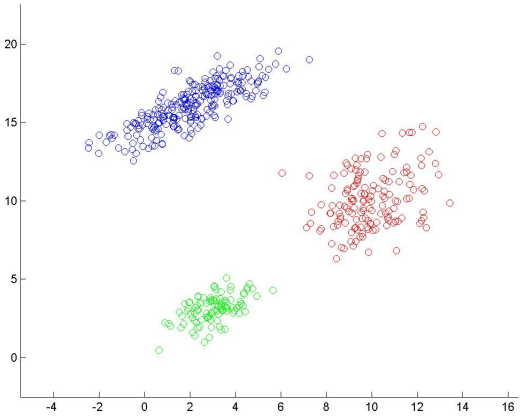


Figura A.4: Conjunto de nuvens afastadas.

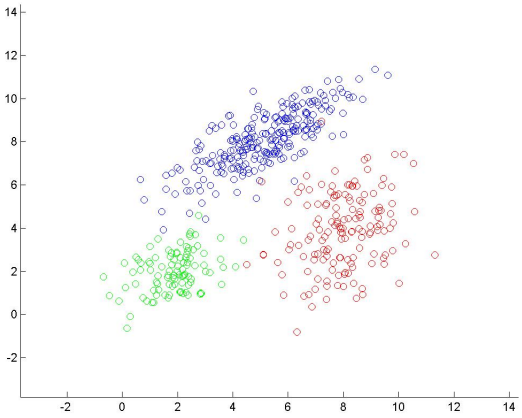


Figura A.5: Conjunto de nuvens Aproximadas.

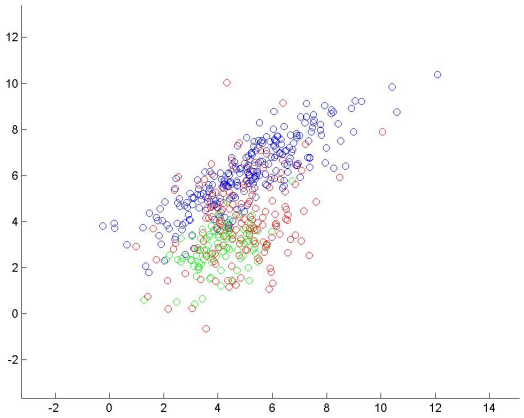



Figura A.6: Conjunto de nuvens intersectadas.

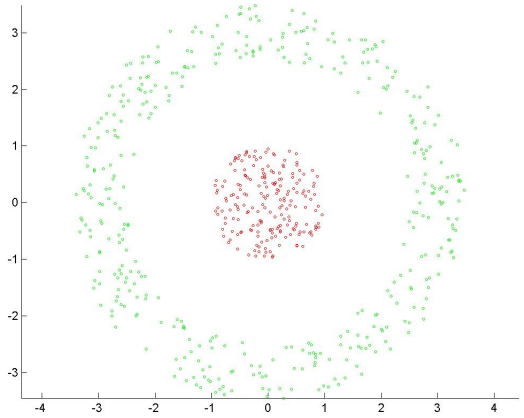



Figura A.7: Conjunto com um núcleo.

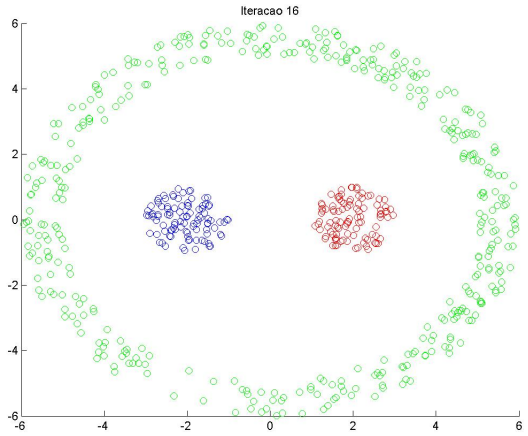



Figura A.8: Conjunto com dois núcleos.

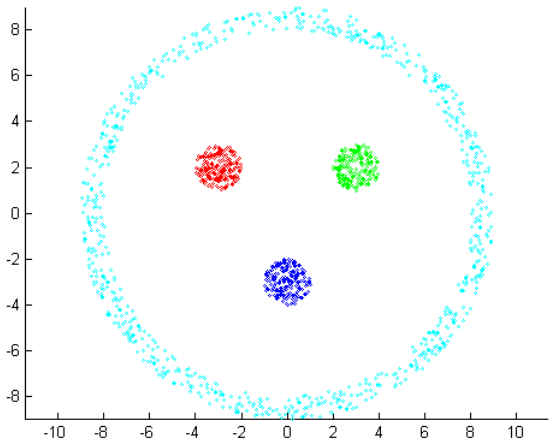



Figura A.9: Conjunto com três núcleos.

# Apêndice B

# Pseudocódigo de algoritmos

**Algorithm 1** K-means

**procedure** K-MEANS($\boldsymbol{D}, k, \epsilon$)▷ $\boldsymbol{D}$ conjunto dos pontos, $k$ número de clusters, $\epsilon$ erro
  $iteracao \leftarrow 0$
  Inicializar os $\boldsymbol{\mu}_i$, $i = \{1, \ldots, k\}$ (Com pontos aleatórios de $\boldsymbol{D}$)
  **repeat**
    $C_i \leftarrow \emptyset, \quad \forall j = 1 \ldots k$
    **for** $\boldsymbol{x}_j \in \boldsymbol{D}$ **do**
      $j^* \leftarrow \underset{i}{\operatorname{argmin}} \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_i^{iteracao}\|^2 \right\}$
      $C_{j^*} = C_{j^*} \cup \{\boldsymbol{x}_j\}$
    **end for**
    **for** $i = 1$ até $k$ **do**
      $\boldsymbol{\mu}_i^{iteracao} = \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$
    **end for**
  **until** $\sum_{i=1}^{k} \|\boldsymbol{\mu}_i^{iteracao} - \boldsymbol{\mu}_i^{iteracao-1}\|^2 < \epsilon$
**end procedure**

**Algorithm 2** Expetaction-Maximization(E-M)

**procedure** E-M($\boldsymbol{D}, k, \epsilon$)▷ $\boldsymbol{D}$ conjunto dos pontos, $k$ número de clusters, $\epsilon$ erro
  $iteracao \leftarrow 0$
  Atribuir um ponto aleatório de $\boldsymbol{D}$ a cada $\boldsymbol{\mu}_1^0, \ldots, \boldsymbol{\mu}_k^0$ ▷ Inicialização
  $\boldsymbol{\Sigma}_i^0 \leftarrow \boldsymbol{I}$
  $P^0(C_i) \leftarrow \frac{1}{k}$, para $i = 1, \ldots, k$
  **repeat**
    $iteracao \leftarrow iteracao + 1$
    **for** $i = 1, \ldots, k$ e $j = 1, \ldots, n$ **do** ▷ Fase das expectativas
      $w_{ij} = \frac{f(\boldsymbol{x}_j|\boldsymbol{\mu}_i, \Sigma_i)P(C_i)}{\sum_{a=1}^{k} f(\boldsymbol{x}_j|\boldsymbol{\mu}_a, \Sigma_a)P(C_a)}$
    **end for**
    **for** $i = 1, \ldots, k$ **do** ▷ Fase da maximização
      $\boldsymbol{\mu}_i^{iteracao} \leftarrow \frac{\sum_{j=1}^{n} w_{ij}\boldsymbol{x}_j}{\sum_{j=1}^{n} w_{ij}}$
      $\boldsymbol{\Sigma}_i^{iteracao} \leftarrow \frac{\sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n}}$
      $P^{iteracao}(C_i) \leftarrow \frac{\sum_{i}^{n} w_{ij}}{n}$
    **end for**
  **until** $\max_{1 \leq i \leq k} \left\{ \|\boldsymbol{z}_i^t - \boldsymbol{z}_i^{t-1}\|_2^2 \right\} \leq \epsilon$
**end procedure**

**Algorithm 3** Kernel K-means

**procedure** KERNEL K-MEANS($\boldsymbol{D}, k, \epsilon$) ▷ $\boldsymbol{D}$ conjunto dos pontos, $k$ número de clusters, $\epsilon$ erro
  $t \leftarrow 0$
  $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$ ▷ Atribuição aleatório dos pontos do conjunto $\boldsymbol{D}$ aos clusters
  **repeat**
    $t \leftarrow t + 1$
    **for** $C_i \in \mathcal{C}^{t-1}$ **do** ▷ Calculo da norma quadrada da média de cada cluster
      $sqnorm_i \leftarrow \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)$
    **end for**
    **for** $\boldsymbol{x}_j \in \boldsymbol{D}$ **do** ▷ Calculo da média do valor da função kernel entre $\boldsymbol{x}_a \in C_i$ e $\boldsymbol{x}_j$
      **for** $C_i \in \mathcal{C}^{t-1}$ **do**
        $avg_{ij} \leftarrow \frac{1}{n_i} \sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j)$
      **end for**
    **end for**
    **for** $\boldsymbol{x}_j \in \boldsymbol{D}$ **do** ▷ atribuição do ponto ao cluster mais próximo
      **for** $C_i \in \mathcal{C}^{t-1}$ **do**
        $d(\boldsymbol{x}_j, C_i) \leftarrow sqnorm_i - 2avg_{ij}$
      **end for**
      $j^* \leftarrow \text{argmin}_i \{d(\boldsymbol{x}_j, C_i)\}$
      $C_j^t \leftarrow C_j^t \cup \{\boldsymbol{x}_j\}$
    **end for**
    $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$
  **until** $1 - \frac{1}{n} \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}| \leq \epsilon$
**end procedure**

# Bibliografia


[1] Zaki, Mohammmed J.; Jr, Wagner Meira.*Data Mining and Analysis: Fundamental Concepts and Algorithms.* Cambridge University Press, New York, 2014.

[2] Selim, Shokri Z.; Ismail, M. A.*K-means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality.* IEEE TRANSACTIONS ON PATTFRN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-6, NO. 1, JANUARY 1984.

[3] Nocedal,Jorge; Wright, Stephen J.*Numerical Optimization.* Second edition, Springer. United States of America, 2006.

[4] Liu,Bing.*Web Data Mining.* Springer. United States of America , 2006.

[5] Brito, P; Bertrand, P; Gucumel, G; Carvalho, F. de. *Selected Contribution in Data Analysis and Classification.* Springer. September 2007

[6] Jain, Anil K.*Data clustering: 50 years beyond K-means.* Elsevier. 2009

[7] Xu, Rui; Wunsch II, Donald.*Survey of Clustering Algorithms.* IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL.16,NO.3, Maio 2005.

[8] Hartigan, J.A.*Clustering algorithms.* John Wiley and Sons, New York City, 1975.

[9] SOKAL, R.R.; SNEATH, P. H.*Principles of numerical taxonomy.* Freeman, San Francisco - London, 1963.

[10] Berkhin,P.*A Survey of Clustering Data Mining Techniques.* Technical report, Accrue Software, 2002.

[11] Aggarwall,Charu C.; Zhai, ChengXiang.*A SURVEY OF TEXT CLUSTERING ALGORITHMS.*

[12] Silva, Jonathan A.; Faria, Elaine R.; Barros, Rodrigo C.; Hruschka, Eduardo R.; CARVALHO, André C. P. L. F.; Gama,J.*Data Stream Clustering: A Survey.* ACM Computing Surveys, Vol. 46, No. 1, Article 13, Outubro , 2013.

[13] Nunes,Diogo. *Uma introdução aos métodos de agrupamento(clustering)* Coimbra, 29 de Outubro, 2016.