UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

José Nataniel Andrade de Sá

Modelos de Co-clustering Fuzzy Baseados em Kernel no Espaço de Características com Ponderação Automática das Variáveis

Recife

2023

José Nataniel Andrade de Sá

Modelos de Co-clustering Fuzzy Baseados em Kernel no Espaço de Características com Ponderação Automática das Variáveis

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador: Prof. Dr. Francisco de Assis Tenório de Carvalho

Coorientador: Prof. Dr. Marcelo Rodrigo Portela Ferreira

Recife

2023

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S111m Sá, José Nataniel Andrade de
Modelos de co-clustering fuzzy baseados em kernel no espaço de características com ponderação automática das variáveis / José Nataniel Andrade de Sá. – 2023.
121 f.: fig., tab.

Orientador: Francisco de Assis Tenório de Carvalho.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências e apêndice.

1. Inteligência computacional. 2. Co-clustering. I. Carvalho, Francisco de Assis Tenório de (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2023-42

José Nataniel Andrade de Sá

“Modelos de Co-clustering Fuzzy Baseados em Kernel no Espaço de Características com Ponderação Automática das Variáveis”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 24/02/2023

BANCA EXAMINADORA

_____

Prof. Dr. Sérgio Ricardo de Melo Queiroz
Centro de Informática/UFPE

_____

Prof. Dr. Luciano Demétrio Santos Pacífico
Departamento de Computação / UFRPE

_____

Prof. Dr. Francisco de Assis Tenório de Carvalho
Centro de Informática / UFPE
(**Orientador**)

Este trabalho é dedicado a Deus, por tudo; aos meus pais, Francilene e José; a minha tia, Ana; aos demais familiares e amigos, que direta ou indiretamente contribuíram para o meu progresso até aqui.

AGRADECIMENTOS

Agradeço primeiramente ao senhor Deus por me conceder tudo que eu precisava para concluir essa etapa da minha vida.

De modo especial, agradeço aos meus pais, Francilene e José, e a titia, Ana. Obrigado pela paciência, compreensão e incentivo. Agradeço também as minhas irmãs, Ana Beatriz e Ana Cláudia, e a minha tia, Joaquina. Sou grato a todos os meus familiares e amigos, por direta ou indiretamente estarem presentes em minha vida e me motivarem a concluir essa jornada.

Eu quero expressar a minha sincera gratidão ao professor Francisco Carvalho por toda sua orientação, paciência, motivação, confiança e apoio. Agradeço ao professor Marcelo Ferreira por todo suporte e incentivo durante esse período. Obrigado por serem muito mais que orientadores.

Um agradecimento especial aos professores Sérgio Queiroz e Luciano Pacífico pela disposição em participar da banca avaliadora e pelas valiosas críticas e sugestões.

Sou grato ao Centro de Informática (CIn-UFPE) por todo o suporte, em especial gostaria de agradecer aos meus colegas, e aos professores pelas excelentes aulas e ensinamentos. Agradeço à Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) por todo suporte financeiro concedido durante essa jornada.

RESUMO

Nesta dissertação, propomos métodos de co-clustering fuzzy baseados em kernel com ponderação automática das variáveis via distâncias adaptativas. Algoritmos de co-clustering realizam o agrupamento simultâneo de objetos e variáveis. Esses métodos organizam a matriz de dados em blocos homogêneos, conseguindo fornecer insights a respeito da associação entre objetos e variáveis. Funções kernel têm sido usadas com sucesso nos algoritmos de agrupamento convencional para mitigar o problema da separabilidade dos grupos, que também pode afetar os algoritmos de co-clustering. O uso de distâncias adaptativas nos permite aprender dinamicamente os pesos das variáveis durante o processo de otimização, levando a uma melhora no desempenho dos algoritmos. Distâncias adaptativas mudam a cada iteração do algoritmo e podem ser a mesma para todos os grupos (distância adaptativa global) ou diferentes de um grupo para outro (distância adaptativa local). Nesse sentido, três algoritmos foram propostos: (i) O *Gaussian Kernel Fuzzy Double K-Means* (*GKFDK*) é o algoritmo base, o qual combina co-clustering com o kernel gaussiano, mas considera que todas as variáveis são igualmente importantes na formação dos grupos de objetos. (ii) O *Gaussian Kernel Fuzzy Double K-Means Based on Global Adaptive Distance* (*GKFDK-GP*) é uma extensão do *GKFDK* que realiza a ponderação de variáveis. O *GKFDK-GP* considera que as variáveis têm o mesmo peso para todos os grupos de objetos. (iii) O *Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance* (*GKFDK-LP*) também é uma extensão do *GKFDK* que realiza a ponderação de variáveis. Contudo, o *GKFDK-LP* considera que as variáveis têm pesos diferentes para cada grupo de objetos. Experimentos realizados com dados sintéticos e reais, em comparação com algoritmos de co-clustering e de agrupamento convencional do estado da arte, mostraram a eficácia dos algoritmos propostos. Os modelos com ponderação das variáveis apresentaram os melhores resultados no geral, onde o *GKFDK-LP* obteve o melhor desempenho entre todos, seguido do *GKFDK-GP*. O algoritmo base *GKFDK* obteve o terceiro melhor desempenho, em relação à maioria das métricas de avaliação.

Palavras-chaves: co-clustering; funções kernel; ponderação automática das variáveis; distâncias adaptativas; agrupamento fuzzy.

ABSTRACT

In this dissertation, we propose kernel-based fuzzy co-clustering methods with automatic variable weighting via adaptive distances. Co-clustering algorithms perform simultaneous clustering of objects and variables. These methods organize the data matrix into homogeneous blocks, being able to provide insights about the association between objects and variables. Kernel functions have been successfully used in conventional clustering algorithms to mitigate the problem of cluster separability, which can also affect co-clustering algorithms. The use of adaptive distances allows us to dynamically learn the weights of the variables during the optimization process, leading to improved performance of the algorithms. Adaptive distances change with each iteration of the algorithm and can be the same for all clusters (global adaptive distance) or different from one cluster to another (local adaptive distance). In this sense, three algorithms were proposed: (i) The *Gaussian Kernel Fuzzy Double K-Means* (*GKFDK*) is the base algorithm, which combines co-clustering with the Gaussian kernel, but considers that all variables are equally important in the formation of the clusters of objects. (ii) The *Gaussian Kernel Fuzzy Double K-Means Based on Global Adaptive Distance* (*GKFDK-GP*) is an extension of the *GKFDK* that performs variable weighting. The *GKFDK-GP* considers variables to have the same weight for all clusters of objects. (iii) The *Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance* (*GKFDK-LP*) is also an extension of the *GKFDK* that performs variable weighting. However, *GKFDK-LP* considers that variables have different weights for each cluster of objects. Experiments performed with both synthetic and real datasets, in comparison with state-of-the-art co-clustering and conventional clustering algorithms, showed the effectiveness of the proposed algorithms. The models with variable weighting showed the best results overall, where *GKFDK-LP* performed best among all, followed by *GKFDK-GP*. The base algorithm *GKFDK* obtained the third best performance, with respect to most of the evaluation metrics.

Keywords: co-clustering; kernel functions; automatic variable weighting; adaptive distances; fuzzy clustering.

LISTA DE FIGURAS

Figura 1 – (a) Conjunto de dados original; (b) Conjunto de dados reorganizado de acordo com a partição dos objetos; (c) Conjunto de dados reorganizado de acordo com as partições dos objetos e das variáveis. 19

Figura 2 – (a) Estrutura de dados simples: grupos com formato hiperesférico e linearmente separáveis entre si; (b) Estrutura de dados complexa: grupos com formato não-hiperesférico e linearmente não-separáveis entre si. 21

Figura 3 – Taxonomia de agrupamento. 26

Figura 4 – (a) Conjunto de dados original. (b) Dendrograma com dois cortes (linha preta e linha vermelha) que produzem partições com dois e três grupos, respectivamente. 28

Figura 5 – Agrupamento obtido a partir do conjunto de dados (Ver Figura 4(a)). (a) Agrupamento hard. (b) Agrupamento fuzzy. 30

Figura 6 – Partições obtidas do conjunto de dados (Ver Figura 4(a)). 31

Figura 7 – Visualização de diferentes subespaços de variáveis. 41

Figura 8 – Dados reorganizados de acordo com co-clustering. 46

Figura 9 – Fluxograma dos algoritmos de co-clustering propostos. 70

Figura 10 – Visualização dos conjuntos de dados simulados. (a) Conjunto de dados sintéticos 1; (b) Conjunto de dados sintéticos 2; (c) Conjunto de dados sintéticos 3; (d) Conjunto de dados sintéticos 4. 83

Figura 11 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 2. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas. 87

Figura 12 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 3. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas. 88

Figura 13 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 4. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas. 89

Figura 14 – Pesos de relevância obtidos no conjunto de dados simulados 3 usando os algoritmos *GKFDK-GP* e *GKFDK-LP*. 91

Figura 15 – Pesos de relevância obtidos no conjunto de dados simulados 4 usando os algoritmos *GKFDK-GP* e *GKFDK-LP*. 92
Figura 16 – Distribuição das variáveis no conjunto de dados Algerian Forest. 96
Figura 17 – Valores do critério de adequação $\mathcal{J}$ obtidos no conjunto de dados Algerian Forest através dos modelos propostos com ponderação das variáveis. 97

LISTA DE TABELAS

Tabela 1 – Exemplos de funções kernel. 36
Tabela 2 – Regras de alocação para a partição fuzzy dos objetos de acordo com cada algoritmo proposto de Co-clustering. 67
Tabela 3 – Regras de alocação para a partição fuzzy das variáveis de acordo com cada algoritmo proposto de Co-clustering. 68
Tabela 4 – Funções objetivos, parâmetros e categorias dos algoritmos. 73
Tabela 5 – Matriz de confusão 74
Tabela 6 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 1. 80
Tabela 7 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 2. 81
Tabela 8 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 3. 81
Tabela 9 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 4. 82
Tabela 10 – Valores dos parâmetros de cada algoritmo nos conjuntos de dados simulados. 82
Tabela 11 – Desempenho dos algoritmos nos conjuntos de dados simulados. 86
Tabela 12 – p-valor do teste de Friedman nas aplicações aos dados sintéticos. 86
Tabela 13 – Descrição dos conjuntos de dados reais. 93
Tabela 14 – Valores dos parâmetros dos algoritmos usados nas aplicações aos conjuntos de dados reais. 94
Tabela 15 – Pesos de relevância obtidos no conjunto de dados Algerian Forest através dos algoritmos *GKFDK-GP* e *GKFDK-LP*. 97
Tabela 16 – Métricas de avaliação fuzzy obtidas nas aplicações aos dados reais. O valor na tabela representa o resultado do melhor caso entre as 50 repetições, de acordo com a função objetivo. O valor entre parênteses refere-se ao ranking entre os algoritmos. 99

Tabela 17 – Métricas de avaliação crisp obtidas nas aplicações aos dados reais. O valor na tabela representa o resultado do melhor caso entre as 50 repetições, de acordo com a função objetivo. O valor entre parênteses refere-se ao ranking entre os algoritmos. 100
Tabela 18 – Ranking médio de desempenho dos algoritmos nos conjuntos de dados reais. 101
Tabela 19 – Média $\pm$ desvio padrão das métricas fuzzy nas 50 repetições. Valores em negrito indicam diferenças significativas entre o modelo do estado da arte e o modelo proposto com melhores desempenhos (p-valor do teste de Wilcoxon abaixo de 0.05). 103
Tabela 20 – Média $\pm$ desvio padrão das métricas crisp nas 50 repetições. Valores em negrito indicam diferenças significativas entre o modelo do estado da arte e o modelo proposto com melhores desempenhos (p-valor do teste de Wilcoxon abaixo de 0.05). 104
Tabela 21 – Desempenho médio geral considerando todas as repetições e todos os conjuntos de dados reais (média $\pm$ desvio padrão). Valores em negrito indicam o melhor resultado. 105

LISTA DE ABREVIATURAS E SIGLAS

ARI	*Adjusted Rand Index*
DK	*Double K-Means*
FCM	*Fuzzy C-Means*
FDK	*Fuzzy Double K-Means*
FMS	*F-Measure*
FRG	*Frigui Index*
FWC	*Feature Weighting Clustering*
GKFDK-GP	*Gaussian Kernel Fuzzy Double K-Means Based on Global Adaptive Distance*
GKFDK-LP	*Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance*
GKFDK	*Gaussian Kernel Fuzzy Double K-Means*
HSC	*Hard Subspace Clustering*
HUL	*Hullermeier Index*
KFCM	*Kernel Fuzzy C-Means*
KM	*K-Means*
MPC	*Modified Partition Coefficient*
NMI	*Normalized Mutual Information*
OAR	*Overall Average Ranking*
SC	*Subspace Clustering*
SSC	*Soft Subspace Clustering*
WFDK	*Weighted Fuzzy Double K-Means*

LISTA DE SÍMBOLOS

$\mathbf{X}$	Conjunto de dados
$\mathbf{X}_{kh}$	Bloco de dados formado pelo k-ésimo grupo de objetos e o h-ésimo grupo de variáveis
$\mathbf{x}_i$	i-ésimo objeto do conjunto de dados
x_{ij}	descreve o i-ésimo objeto na j-ésima variável
N	Número de objetos (instâncias)
P	Número de variáveis (atributos)
K	Número de grupos dos objetos
H	Número de grupos das variáveis
$\mathbf{C}$	Matriz de protótipos dos grupos nos algoritmos de agrupamento convencional
$\mathbf{c}_k$	Protótipo do k-ésimo grupo nos algoritmos de agrupamento convencional
c_{kj}	Protótipo do k-ésimo grupo na j-ésima variável nos algoritmos de agrupamento convencional
$\mathbf{G}$	Matriz de protótipos dos blocos (co-clusters) nos algoritmos de co-clustering
g_{kh}	Protótipo do bloco $\mathbf{X}_{kh}$ nos algoritmos de co-clustering
$\mathbf{U}$	Matriz de pertencimento (pertinência) fuzzy dos objetos
u_{ik}	Grau de pertencimento do i-ésimo objeto no k-ésimo grupo de objetos
$\mathbf{V}$	Matriz de pertencimento (pertinência) fuzzy das variáveis
v_{jh}	Grau de pertencimento da j-ésima variável no h-ésimo grupo de variáveis
$\mathbf{\Lambda}$	Matriz ou vetor de pesos de relevância das variáveis
λ_j	Peso da j-ésima variável

λ_{kj}	Peso da j-ésima variável no k-ésimo grupo de objetos
m	Parâmetro que controla a imprecisão dos objetos
n	Parâmetro que controla a imprecisão das variáveis
γ	Parâmetro que controla a influência dos pesos no algoritmo *WFDK*
σ^2	Hiperparâmetro do kernel gaussiano
$\mathcal{J}$	Função objetivo
ε	Melhoria mínima na função objetivo entre duas iterações consecutivas
T	Número máximo de iterações
$\mathcal{L}$	Função lagrangiana
θ, η, δ	Multiplicadores de Lagrange
∂	Derivada parcial

SUMÁRIO

1 INTRODUÇÃO 17
1.1 MOTIVAÇÃO 20
1.2 OBJETIVOS E CONTRIBUIÇÕES 22
1.3 PUBLICAÇÕES 23
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO 23
2 ALGORITMOS DE AGRUPAMENTO 25
2.1 INTRODUÇÃO 25
2.2 TIPOS DE AGRUPAMENTO 26
2.3 MEDIDAS DE DISTÂNCIA 31
2.4 *FUZZY C-MEANS (FCM)* 33
2.5 AGRUPAMENTO BASEADO EM KERNEL 35
2.5.1 *Kernel Fuzzy C-Means (KFCM)* 38
2.6 AGRUPAMENTO DE DADOS COM PONDERAÇÃO DAS VARIÁVEIS . . 40
2.7 SÍNTESE DO CAPÍTULO 42
3 ALGORITMOS DE CO-CLUSTERING 44
3.1 INTRODUÇÃO 44
3.2 CO-CLUSTERING BASEADO EM MÉTRICAS 46
3.2.1 *Fuzzy Double K-Means (FDK)* 47
3.3 ALGORITMOS DE CO-CLUSTERING COM PONDERAÇÃO DAS VARIÁVEIS 50
3.3.1 *Weighted Fuzzy Double K-Means (WFDK)* 51
3.4 SÍNTESE DO CAPÍTULO 55
4 ALGORITMOS DE CO-CLUSTERING FUZZY BASEADOS EM KERNEL COM DISTÂNCIAS ADAPTATIVAS 56
4.1 MEDIDAS DE DISTÂNCIA BASEADAS EM KERNEL 56
4.2 MODELOS DE CO-CLUSTERING FUZZY BASEADOS EM KERNEL COM PONDERAÇÃO AUTOMÁTICA DAS VARIÁVEIS 59
4.2.1 Passos de otimização 61
4.2.1.1 Etapa de representação 61
4.2.1.2 Etapa de ponderação 63

4.2.1.3 *Etapa de alocação dos objetos* . . . *65*
4.2.1.4 *Etapa de alocação das variáveis* . . . *67*
4.2.2 **Análise de complexidade** . . . **68**
4.2.3 **Algoritmos de co-clustering fuzzy propostos** . . . **68**
4.3 SÍNTESE DO CAPÍTULO . . . 71
5 **AVALIAÇÃO EXPERIMENTAL** . . . **72**
5.1 CONFIGURAÇÃO EXPERIMENTAL . . . 72
5.1.1 **Métricas de avaliação** . . . **73**
5.1.1.1 *Adjusted Rand Index* . . . *74*
5.1.1.2 *F-Measure* . . . *75*
5.1.1.3 *Normalized Mutual Information* . . . *75*
5.1.1.4 *Modified Partition Coefficient* . . . *76*
5.1.1.5 *Frigui Index* . . . *77*
5.1.1.6 *Hullermeier Index* . . . *78*
5.1.2 **Configuração dos hiperparâmetros** . . . **78**
5.2 APLICAÇÕES COM DADOS SIMULADOS . . . 80
5.2.1 **Resultados e discussão** . . . **82**
5.3 APLICAÇÕES COM DADOS REAIS . . . 93
5.3.1 **Análise do melhor caso** . . . **97**
5.3.2 **Análise do caso médio** . . . **101**
5.4 SÍNTESE DO CAPÍTULO . . . 105
6 **CONCLUSÕES** . . . **106**
6.1 CONTRIBUIÇÕES . . . 106
6.2 RESUMO DOS RESULTADOS . . . 107
6.3 LIMITAÇÕES . . . 109
6.4 TRABALHOS FUTUROS . . . 109
REFERÊNCIAS . . . **110**
APÊNDICE A - ARTIGO FUZZ-IEEE 2022 . . . **119**

1 INTRODUÇÃO

Agrupamento é uma das técnicas mais utilizadas em aprendizado de máquina não-supervisionado e desempenha um papel importante em diversas áreas do conhecimento, como reconhecimento de padrões, processamento de imagens, mineração de dados, recuperação de informação, entre outras (GAN; MA; WU, 2020; TANG; REN; PEDRYCZ, 2020). A ideia básica dos algoritmos de agrupamento é dividir dados em grupos, de modo que em um mesmo grupo exista um alto grau de similaridade entre os padrões, enquanto que em grupos diferentes os padrões sejam relativamente dissimilares. Por ser uma técnica de longa história, esses algoritmos têm sido desenvolvidos sob diferentes aspectos e aplicados a diversos cenários de aplicações. Apesar de toda essa evolução, a pesquisa em agrupamento continua em aberto, principalmente pela complexidade dos dados, que atualmente, tem se evidenciado de diversas maneiras, como tamanho dos conjuntos de dados, características das variáveis (atributos), escalabilidade e multiplicidade dos dados (JIANG et al., 2014).

A maioria dos algoritmos de agrupamento são divididos em duas categorias: hierárquico e particional. Os modelos hierárquicos fornecem uma sequência aninhada dos dados, resultando em uma estrutura de dependência entre os grupos, sendo normalmente representada por um dendrograma. Em contraste, os métodos particionais fornecem uma partição única dos dados de entrada em um número fixo de grupos, através da otimização (geralmente local) de uma função objetivo, criando superfícies de separação entre os grupos. Os modelos de agrupamento particional podem ser executados de duas maneiras diferentes: hard (crisp) e fuzzy (soft). Na abordagem hard, os padrões pertencem a um e, somente um, grupo. Nesse caso os grupos são disjuntos e não sobrepostos. Por outro lado, na abordagem fuzzy os objetos podem pertencer a todos os grupos, com um certo grau de pertencimento. A abordagem fuzzy é particularmente eficaz quando há sobreposição dos dados e os limites entre os grupos de dados são ambíguos. Estudos mais detalhados sobre diversos algoritmos de agrupamento podem ser encontrados em (XU; TIAN, 2015; SAXENA et al., 2017; EZUGWU et al., 2022).

Uns dos mais conhecidos algoritmos de agrupamento são o *K-Means* (*KM*) (MACQUEEN, 1967) e *Fuzzy C-Means* (*FCM*) (BEZDEK, 1981), que compreendem as abordagens hard e fuzzy, respectivamente. O *FCM* é uma extensão do *KM* para o framework de agrupamento fuzzy. Nos últimos anos, os métodos fuzzy tem ganhado destaque, por serem mais naturais que os algoritmos hard, em diversas situações. Por exemplo, ao se perguntar se está quente em um

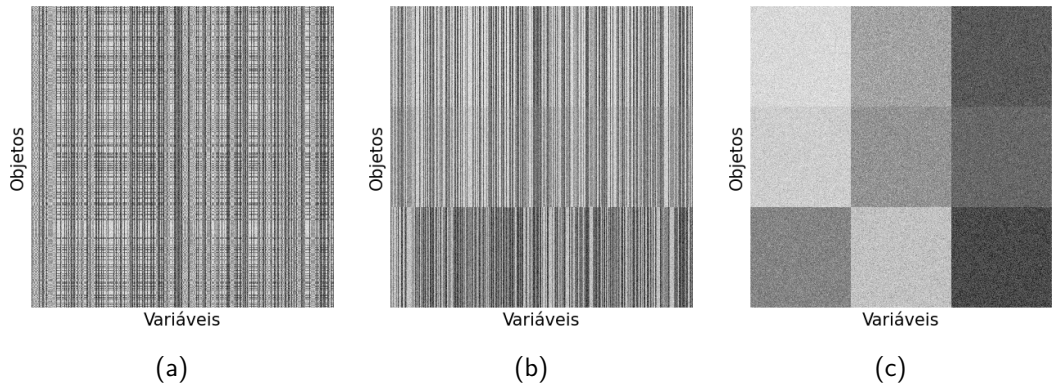
determinado ambiente, a resposta pode ser simplesmente sim ou não. Contudo, essa resposta exclusiva pode não ser apropriada, pois, no ambiente, pode estar muito quente, quente, moderado, pouco quente ou frio. Nesse caso a lógica fuzzy faz mais sentido. No agrupamento fuzzy os objetos nas fronteiras das várias classes não são forçados a pertencer totalmente a uma das classes, mas recebem graus de pertinência entre 0 e 1, indicando sua pertinência parcial (BORA; GUPTA; KUMAR, 2014). Diversos algoritmos de agrupamento fuzzy foram propostos nos últimos tempos. Chen e Zang (CHEN; ZHANG, 2004) propuseram algoritmos *FCM* robustos baseados em medidas de distância induzida por kernel para segmentação de imagem. Geweniger *et al.* (GEWENIGER et al., 2010) introduziram um algoritmo *FCM* baseado em mediana para dados de dissimilaridade. Mais recentemente, Chiu e Hsu (CHIU; HSU, 2017) propuseram um algoritmo fuzzy baseado em uma integração de dados psicológicos e fisiológicos para design de música terapêutica. Mahmoudi *et al.* (MAHMOUDI et al., 2020) usaram um algoritmo de agrupamento fuzzy para comparar a taxa de propagação da Covid-19 em países de alto risco. Algumas revisões acerca de agrupamento fuzzy e suas aplicações podem ser encontradas em (NAYAK; NAIK; BEHERA, 2015) e (RUSPINI; BEZDEK; KELLER, 2019).

A grande maioria dos métodos tradicionais de agrupamento realizam um processamento unilateral, ou seja, agrupa apenas os objetos (agrupamento mais comum) ou variáveis em um conjunto de dados. No entanto, em muitos problemas do mundo real pode existir uma relação intrínseca entre objetos e variáveis. Por exemplo, na análise de documentos, muitos conjuntos de dados são formulados como uma matriz termo-documento, ou seja, os objetos (linhas) são documentos, as variáveis (colunas) são palavras e cada valor da tabela representa a frequência de uma palavra em um documento ou uma medida estatística que indica a importância de uma palavra em um documento. Nesse caso, em uma determinada classe de documentos as palavras que a compõem devem ser correlacionadas, pois em teoria esses documentos estão relacionados ao mesmo tema, enquanto que para outra classe de documentos, outro conjunto de palavras deve ser mais similar. Assim, é razoável supor que os documentos são agrupados de acordo com suas relações com os grupos de palavras e o mesmo pode ser assumido com o agrupamento das palavras. Outro exemplo está relacionado à análise biomédica, quando o objetivo é agrupar uma tabela de sintomas de doenças. Alguns sintomas estão intimamente relacionados a algumas doenças. Por exemplo, existe uma estreita relação entre o aumento da pressão de pulso e tipos de doenças metabólicas. Nesse caso, não é científico negligenciar as correlações entre os sintomas (LIU et al., 2017). Na análise de expressão genética, essa relação dupla entre objetos (pacientes ou condições) e variáveis (genes) também é observada.

Em todos os casos citados anteriormente, os métodos tradicionais de agrupamento podem encontrar estruturas não representativas e obter um desempenho ruim, pois negligenciam a relação entre objetos e variáveis. No entanto, nos últimos anos surgiram outros métodos de agrupamento, chamados de co-clustering (biclustering, double clustering ou block-clustering), que ganharam destaque ao agrupar simultaneamente objetos e variáveis com base nas informações de co-ocorrência. Esses métodos superam várias limitações associadas aos métodos tradicionais de agrupamento, permitindo a descoberta automática de similaridade com base em um subconjunto de atributos (HUANG; XU; LV, 2018).

Os métodos de co-clustering conseguem reorganizar a matriz de dados inicial em blocos homogêneos (co-clusters). Esses blocos podem ser vistos como subconjuntos da matriz de dados caracterizados por um conjunto de objetos e um conjunto de variáveis cujos elementos são similares. Ao reorganizar a matriz de dados, os algoritmos de co-clustering podem fornecer insights sobre a associação entre objetos e variáveis, principalmente em dados de alta dimensão. Os métodos de co-clustering oferecem algumas vantagens sobre os algoritmos de agrupamento tradicional: eles são capazes de reduzir a matriz inicial em uma forma mais simples com a mesma estrutura básica; intrinsecamente, realizam seleção de variáveis de acordo com a capacidade de discriminar as variáveis que são relevantes para os grupos de objetos; e exigem menos computação quando comparados com o processo de agrupar objetos e variáveis separadamente (LACLAU; NADIF, 2016). A Figura 1 mostra um exemplo de quando os algoritmos de co-clustering podem fornecer resultados mais interessantes que os algoritmos de agrupamento tradicional.

Figura 1 – (a) Conjunto de dados original; (b) Conjunto de dados reorganizado de acordo com a partição dos objetos; (c) Conjunto de dados reorganizado de acordo com as partições dos objetos e das variáveis.



(a) (b) (c)

Fonte: Autor (2023)

Por essas razões, métodos de co-clustering têm sido desenvolvidos sob diferentes aborda-

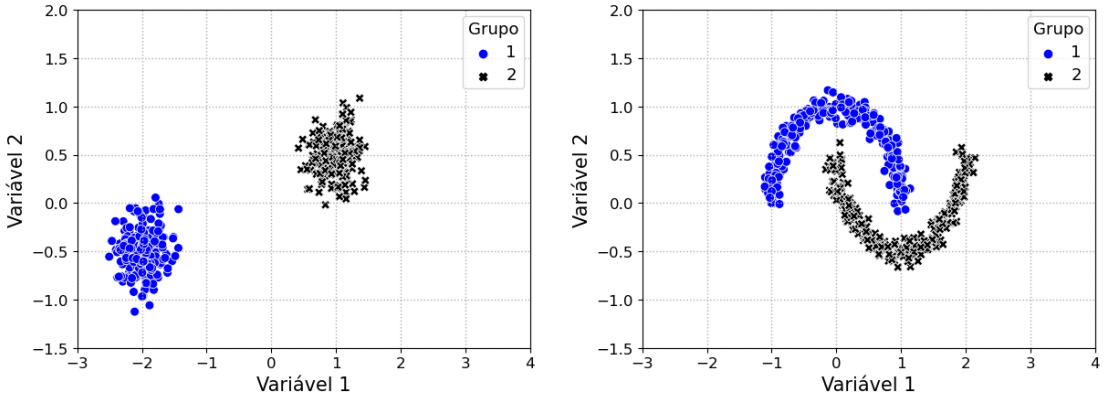
gens, como, por exemplo: co-clustering baseado em grafos, principalmente os modelos espectrais (HUANG et al., 2020; WU; BENSON; GLEICH, 2016), co-clustering baseado na fatoração de matrizes (NIE; SHI; LI, 2020; DENG et al., 2021), co-clustering baseado em mistura de modelos (GOVAERT; NADIF, 2008; UBUKATA; MOURI; HONDA, 2022), co-clustering bayesiano (DU et al., 2019; WANG et al., 2022), co-clustering fuzzy (PHAM et al., 2021; VERMA; HOODA, 2020), entre outras. Em relação aos campos de aplicação, esses algoritmos têm uma vasta gama. Nos dados de expressão genética, esses modelos conseguem identificar grupos de genes (variáveis) que se comportam da mesma maneira sob um subconjunto de condições (objetos) (HOUARI; AYADI; YAHIA, 2018; MAÂTOUK et al., 2021). Na mineração de texto, vários métodos exploram a dualidade entre documentos e palavras (AFFELDT; LABIOD; NADIF, 2021; HUANG; XU; LV, 2018; LACLAU; NADIF, 2016). Em sistemas de recomendação, o uso de métodos de co-clustering tem ganhado destaque (AIREN; AGRAWAL, 2023; LI; WEN; CHEN, 2021; PEREIRA; HRUSCHKA, 2015). Em Análise de Dados Simbólicos (do inglês, Symbolic Data Analysis (SDA)) algoritmos de co-clustering também foram desenvolvidos (CARVALHO et al., 2021). Uma exposição detalhada sobre algoritmos de co-clustering pode ser encontrada em (GOVAERT; NADIF, 2013).

1.1 MOTIVAÇÃO

Um problema comum em algoritmos de agrupamento está relacionado à separabilidade dos grupos, principalmente quando esses modelos são baseados na distância euclidiana. Os algoritmos que tem a distância euclidiana como medida de similaridade, geralmente, apresentam bons desempenhos em conjuntos de dados com grupos aproximadamente hiperesféricos e linearmente separáveis (ver Figura 2 (a)). Em dados onde os grupos têm formato não hiperesférico e/ou padrões linearmente não-separáveis (ver Figura 2 (b)), esses algoritmos podem apresentar desempenho abaixo do desejado para o problema em estudo, conduzindo a uma estrutura de grupos que não é representativa. Devido a essa limitação, vários métodos que conseguem lidar com dados complexos têm sido propostos, como, por exemplo: algoritmos de agrupamento espectral, algoritmos de particionamento de grafos e algoritmos de agrupamento baseados em kernel, de modo que muitos pesquisadores têm demonstrado interesse em desenvolver algoritmos nessa última linha (BEN-HUR et al., 2001; FILIPPONE et al., 2008). Os algoritmos baseados em kernel produzem hipersuperfícies com separação não-linear entre os grupos. Em outras palavras, esses algoritmos projetam os dados em um espaço de alta dimensão (chamaremos de espaço de kernel) com a esperança de que os grupos no espaço de

entrada se tornem bem separados nesse novo espaço, e dessa forma, o processo de agrupar seja mais simples.

Figura 2 – (a) Estrutura de dados simples: grupos com formato hiperesférico e linearmente separáveis entre si; (b) Estrutura de dados complexa: grupos com formato não-hiperesférico e linearmente não-separáveis entre si.



Fonte: Autor (2023)

Desde o trabalho seminal de (GIROLAMI, 2002), uma variedade de algoritmos de agrupamento baseados em kernel foram propostos, como agrupamento fuzzy baseado em kernel (RODRIGUES; OSPINA; FERREIRA, 2021; CHEN, 2002), Mapas Auto-Organizáveis (SOM) baseados em kernel (INOKUCHI; MIYAMOTO, 2004), (KIM et al., 2005), algoritmos para segmentação de imagens(CHEN; CHEN; LU, 2011), dentre outros. Os modelos de agrupamento baseados em kernel são desenvolvidos sob duas abordagens principais. A primeira abordagem, denominada agrupamento no espaço de características, considera que os protótipos (protótipos são os representantes dos grupos) dos grupos estão no espaço de entrada dos dados (chamaremos de espaço de características), e a distância entre os objetos e os protótipos é computada através de funções kernel (ZHANG; CHEN, 2003; GRAVES; PEDRYCZ, 2010; SHEN et al., 2006). A segunda abordagem, agrupamento no espaço de kernel, assume que os protótipos estão no espaço de kernel (GRAVES; PEDRYCZ, 2010; MULLER et al., 2001; SCHÖLKOPF; SMOLA; MÜLLER, 1998; FERREIRA; CARVALHO, 2014). Vários trabalhos mostraram a eficiência dos algoritmos de agrupamento baseados em kernel em uma variedade de problemas do mundo real (DING; FU, 2016; XIANG et al., 2013; YANG et al., 2016).

Assim como nos algoritmos de agrupamento convencionais, o problema de separabilidade dos grupos também pode afetar os algoritmos de co-clustering, de uma forma até mais severa, haja vista que esses últimos realizam o processo em dois sentidos. No entanto, até onde

sabemos, nenhum trabalho estudou o uso de funções kernel em algoritmos de co-clustering, visando superar essa limitação. Outra deficiência dos principais algoritmos de co-clustering, até mesmo dos algoritmos de agrupamento convencional, é que eles tratam as variáveis de forma igualitária, no sentido de que elas possuem a mesma relevância para definir os grupos de objetos. Essa desvantagem não é desejável em algumas aplicações, como agrupamento de dados esparsos de alta dimensão, onde a estrutura de grupos é limitada a um subconjunto de variáveis em vez do conjunto de todas as variáveis (RODRÍGUEZ, 2018).

Diversos algoritmos de agrupamento convencional foram propostos de modo a diferenciar os pesos de relevâncias das variáveis. Esses métodos que integram os chamados Soft Subspace Clustering (SSC) (DENG et al., 2016) atribuem pesos a todas as variáveis, de modo que aquelas com pesos maiores formam o subespaço significativo, isto é, o subconjunto de variáveis na qual um determinado grupo estar melhor definido. Keller and Klawonn (KELLER; KLAWONN, 2000) propuseram um algoritmo *FCM* ponderado atribuindo um parâmetro de influência a cada variável em cada grupo de objetos. Jing, Ng e Huang propuseram um algoritmo K-Means ponderado para dados esparsos (JING; NG; HUANG, 2007). Mais recentemente, Rodrígues e Carvalho (RODRÍGUEZ; CARVALHO, 2017) apresentaram um algoritmo fuzzy baseado na distância Euclidiana e entropia de regularização distinguindo os pesos das variáveis em cada grupo de dados. Muitos algoritmos que distinguem o peso de relevância das variáveis foram desenvolvidos sob o conceito de distâncias adaptativas (CARVALHO; TENÓRIO; JUNIOR, 2006; CARVALHO; LECHEVALLIER, 2009; FERREIRA; CARVALHO, 2014). Esse tipo de distância permite computar os pesos das variáveis para todo o conjunto de dados (distância adaptativa global) ou para cada grupo de dados (distância adaptativa local). No tocante aos algoritmos de co-clustering, Laclau, Carvalho e Nadif (LACLAU; CARVALHO; NADIF, 2015) propuseram um algoritmo fuzzy que computa o peso das variáveis, sendo capaz de realizar seleção de variáveis. Recentemente De Carvalho *et al* (CARVALHO et al., 2021) propuseram algoritmos para dados simbólicos que distinguem os pesos das variáveis.

1.2 OBJETIVOS E CONTRIBUIÇÕES

O principal objetivo deste trabalho é desenvolver algoritmos de co-clustering fuzzy baseados em kernel, considerando a abordagem de agrupamento no espaço de características. De forma mais específica, nós propusemos três algoritmos. Um dos modelos considera apenas a função kernel como medida de similaridade, assumindo que todas as variáveis possuem a

mesma importância para a construção dos grupos de objetos. Para distinguir a relevância das variáveis, visando lidar com a limitação citada anteriormente, usamos distâncias adaptativas, que nos permite aprender os pesos de relevância das variáveis, de forma automática. Existem dois tipos principais de distâncias adaptativas: global e local. Ao usar a distância adaptativa global, assumimos que o conjunto de variáveis relevantes é o mesmo para todos os grupos de objetos, enquanto no caso da distância adaptativa local, assumimos que o conjunto de variáveis relevantes é diferente para todos os grupos de objetos. Vale destacar que em algumas situações as distâncias adaptativas locais não são apropriadas porque podem levar o algoritmo a cair em mínimos locais, fornecendo soluções sub-ótimas (FERREIRA; CARVALHO, 2014). Por esta razão, desenvolvemos os métodos de co-clustering fuzzy baseados kernel considerando ambos os tipos de distâncias adaptativas, global e local. Os objetivos específicos deste trabalho são:

- Estudar os algoritmos de agrupamento tradicional e co-clustering do estado da arte, especialmente os principais modelos fuzzy e baseados em kernel;
- Derivar e implementar os novos modelos de co-clustering fuzzy baseados em kernel;
- Aplicar os algoritmos a problemas de reconhecimento de padrões;
- Avaliar os resultados obtidos pelos algoritmos propostos e comparar com os resultados dos algoritmos do estado da arte.

1.3 PUBLICAÇÕES

Parte desse trabalho resultou, como produção bibliográfica, no artigo ***Kernel-based Fuzzy Co-clustering in Feature Space with Automated Variable Weighting***, aceito e publicado na ***2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)***, onde algoritmos de co-clustering fuzzy baseados em kernel foram propostos.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Além deste capítulo introdutório, este trabalho é organizado em mais cinco capítulos. Os principais conteúdos desses capítulos estão resumidos abaixo.

Capítulo 2: nesse capítulo, nós inicialmente descrevemos a diferença entre os algoritmos particionais de agrupamento hard e fuzzy. Os modelos de agrupamento tradicionais fuzzy

são revisados, onde o algoritmo *Fuzzy C-Means* (*FCM*) é detalhado. A teoria básica sobre funções kernel é apresentada. Mostramos as diferenças entre as duas abordagens de kernel, agrupamento no espaço de características e agrupamento no espaço de kernel. Detalhamos o algoritmo *Kernel Fuzzy C-Means* (*KFCM*). E por fim, foi feita uma revisão a respeito dos algoritmos do estado da arte que realizam ponderação das variáveis.

Capítulo 3: nesse capítulo, nós apresentamos definições e conceitos a respeito de co-clustering. Discutimos brevemente a respeito de diferentes tipos de algoritmos que realizam agrupamento simultâneo e descrevemos sobre os algoritmos mais relacionados com esse trabalho: o *Fuzzy Double K-Means* (*FDK*) e o *Weighted Fuzzy Double K-Means* (*WFDK*).

Capítulo 4: esse capítulo apresenta a principal contribuição desse trabalho: os algoritmos de co-clustering fuzzy baseados em kernel com ponderação automática das variáveis, considerando a abordagem de agrupamento baseado em kernel no espaço de características.

Capítulo 5: esse capítulo apresenta os experimentos realizados com dados simulados e dados reais, para demonstrar a efetividade dos algoritmos propostos. Os resultados obtidos mostraram a superioridade desses algoritmos em relação aos algoritmos do estado da arte.

Capítulo 6: por fim, esse capítulo apresenta as conclusões a respeito dos algoritmos propostos, limitações e possíveis futuras perspectivas de trabalho.

2 ALGORITMOS DE AGRUPAMENTO

2.1 INTRODUÇÃO

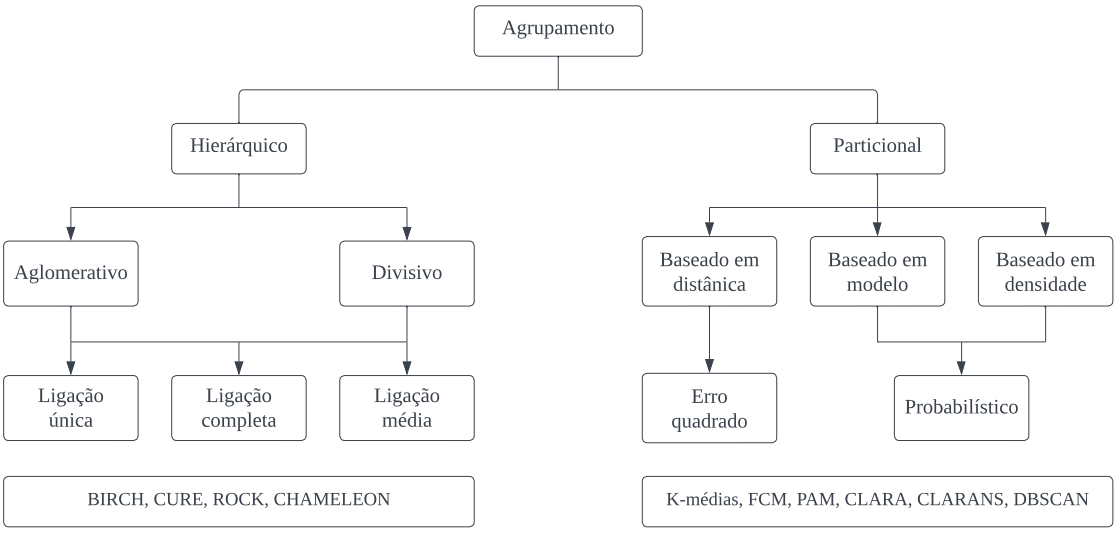
Agrupar é uma tarefa comum aos seres humanos. Corriqueiramente, buscamos organizar itens de acordo com alguma (as) característica (as) que nos permite (em) dizer se eles possuem alguma semelhança ou não. Geralmente, essa organização é feita com base em alguma característica previamente definida como, por exemplo, tamanho, cor, função, entre outras. Na grande maioria das vezes, a tarefa de agrupar é simples do ponto de vista humano, sendo algo que exige mais esforço físico do que reflexão.

Em se tratando de dados, considerando os casos em que essa tarefa é executada por um computador ou por um ser humano, agrupar é um processo desafiador como, por exemplo, quando existe sobrecarga de informação. Nesse caso, essa tarefa se torna humanamente impraticável, sendo necessário o uso de computadores, especialmente, os chamados algoritmos de aprendizado de máquina. Ao ter algum conhecimento prévio a respeito dos dados, isto é, algum rótulo que os classifique em classes, esses algoritmos são geralmente chamados de classificadores, e são enquadrados no aprendizado de máquina supervisionado. Nesse caso, o objetivo principal é treinar o algoritmo, de modo que ele possa "aprender" com os dados existentes e seja capaz de predizer futuras observações não rotuladas. Por outro lado, quando não existe nenhum conhecimento prévio a respeito dos dados, a etapa de treinamento, realizada na abordagem supervisionada, não pode ser feita, pois nenhum rótulo é informado. Nesse caso, o objetivo é agrupar os dados, baseado na similaridade intrínseca e natural existente entre eles. Esses métodos fazem parte do aprendizado de máquina não supervisionado e seus principais representantes são os algoritmos de agrupamento.

Com o desenvolvimento tecnológico dos últimos anos, a capacidade de armazenamento e geração de dados aumentou exponencialmente. Com grandes bases de dados disponíveis, torna-se quase impossível rotulá-las manualmente usando o conhecimento humano. Outro ponto muito importante é que um dos maiores problemas na rotulagem de dados é a inconsistência dos rótulos, resultante de erros humanos, suposições não claras, ambiguidade (no caso de imagens, esse ponto é mais relevante) ou da natureza subjetiva da tarefa. Dessa forma, se os dados apresentam rotulagem incorreta, os algoritmos terão problemas para encontrar o verdadeiro relacionamento entre as observações. Nesse sentido, os algoritmos de agrupamento levam vantagem sobre os algoritmos de aprendizado supervisionado.

O principal objetivo do agrupamento é dividir o conjunto de dados em grupos homogêneos, de modo que em um mesmo grupo os objetos tenham alta similaridade, enquanto que em grupos diferentes os objetos tenham baixa similaridade. Esses métodos são classificados conforme a abordagem usada para gerar os grupos e apresentar os resultados (JAIN; MURTY; FLYNN, 1999). A divisão mais usada em agrupamento consiste em duas classes de algoritmos: hierárquicos e particionais. Alguns autores adicionam outras classes além dessas, como, por exemplo: algoritmos baseados em densidade, algoritmos baseados em modelos e algoritmos baseados em grid (HAN; PEI; TONG, 2022; WEGMANN et al., 2021). Iremos adotar a primeira divisão, pois é mais amplamente utilizada (JAIN, 2010), e os algoritmos do tipo baseados em densidade e baseados em modelos podem ser considerados métodos particionais (SAXENA et al., 2017) (Ver Figura 3).

Figura 3 – Taxonomia de agrupamento.



Fonte: (SAXENA et al., 2017)

2.2 TIPOS DE AGRUPAMENTO

No agrupamento hierárquico, os grupos são formados pela divisão iterativa dos padrões usando uma abordagem de cima para baixo ou de baixo para cima, de acordo com uma matriz de similaridade. Existem duas categorias de modelos hierárquicos, considerando como a hierarquia final é obtida: agrupamento hierárquico aglomerativo e divisivo (MURTAGH, 1983). Os algoritmos aglomerativos seguem a abordagem de baixo para cima, que constrói grupos come-

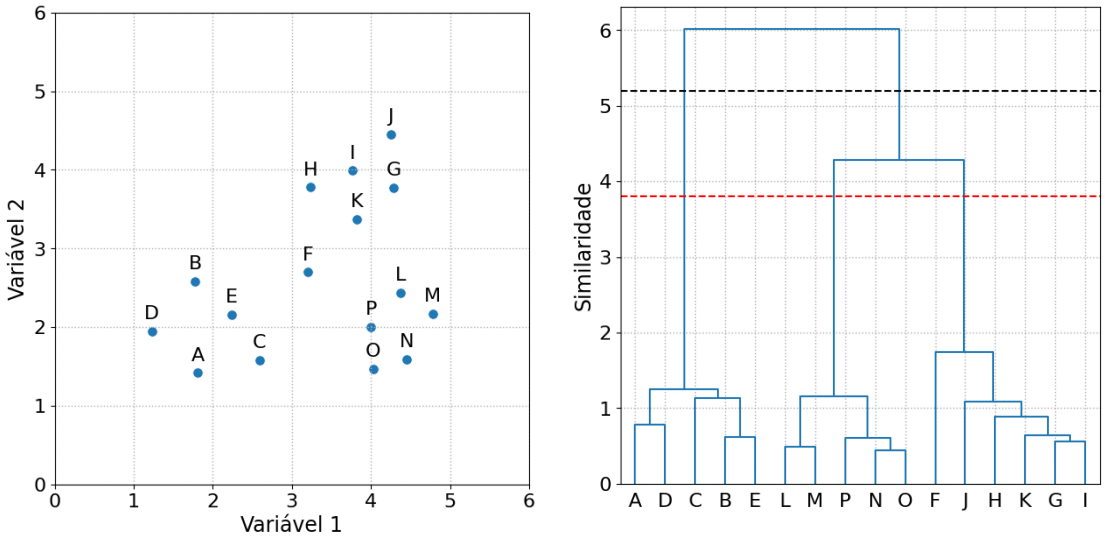
çando com um único objeto em cada um, mesclando-os em outros grupos cada vez maiores, até que todos os objetos estejam finalmente em um único grupo ou até que certas condições de terminação sejam satisfeitas. O agrupamento hierárquico divisivo segue a abordagem de cima para baixo, que se inicia com um grupo contendo todos os objetos e sucessivamente divide-o em grupos menores, até que cada objeto forme um grupo, ou até que satisfaça certas condições de terminação. Os métodos de agrupamento hierárquico podem ser ainda agrupados *de acordo com a medida de similaridade ou ligação* (JAIN; MURTY; FLYNN, 1999):

- **Agrupamento hierárquico de ligação única**: a ligação entre dois grupos é feita por apenas um par de elementos, ou seja, aqueles dois elementos (um em cada grupo) que estão mais próximos um do outro;
- **Agrupamento hierárquico de ligação completa**: a distância entre dois grupos é determinada pela maior distância de qualquer objeto de um grupo para qualquer objeto do outro grupo.
- **Agrupamento hierárquico de ligação média**: a distância entre dois grupos é determinada pela distância média entre os objetos de um grupo e os objetos do outro grupo.

Os métodos hierárquicos fornecem uma saída representada por uma estrutura de hierarquia de grupos representada por um dendrograma (Figura 4 (b)). O nó raiz do dendrograma representa todo o conjunto de dados e cada nó folha é considerado um objeto de dados. Os nós intermediários descrevem a extensão em que os objetos são próximos uns dos outros. A altura de um dendrograma geralmente expressa a similaridade entre cada par de objetos ou aglomerados, ou um objeto e um grupo (RODRÍGUEZ, 2022). Uma vantagem dos algoritmos hierárquicos é que, uma vez que o algoritmo é executado e o dendrograma é construído, pode-se escolher automaticamente um determinado número de grupos dividindo a árvore em diferentes níveis para obter diferentes soluções de agrupamento para o mesmo conjunto de dados sem executar novamente o algoritmo. Por exemplo, na Figura 4 (b) a linha preta pontilhada representa uma corte que fornece uma partição com dois grupos, enquanto a linha vermelha fornece uma partição com três grupos.

A principal deficiência do agrupamento hierárquico está ligada a incapacidade de corrigir decisões erradas (NAGPAL; JATAIN; GAUR, 2013), no sentido de que após dois pontos estarem ligados entre si, eles não podem se mover em outros grupos na hierarquia. Outra grande des-

Figura 4 – (a) Conjunto de dados original. (b) Dendrograma com dois cortes (linha preta e linha vermelha) que produzem partições com dois e três grupos, respectivamente.



(a) (b)

Fonte: Autor (2023)

vantagem desses algoritmos é que eles não são escaláveis. Para grandes conjuntos de dados, os métodos hierárquicos tornam-se impraticáveis, a menos que outras técnicas sejam incorporadas, porque geralmente os métodos hierárquicos são $O(N^2)$ para espaço de memória e $O(N^3)$ para tempo de CPU (ZAIT; MESSATFA, 1997; MURTAGH, 1983), onde N é o número de objetos no conjunto de dados. Alguns algoritmos de destaque, que usam algum aprimoramento, são: *Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH)* (ZHANG; RAMAKRISHNAN; LIVNY, 1996), *Clustering Using Representatives (CURE)* (GUHA; RASTOGI; SHIM, 1998), *ROCK* (GUHA; RASTOGI; SHIM, 2000) e *CHAMELEON* (KARYPIS; HAN; KUMAR, 1999).

No agrupamento particional, os dados são divididos em um número fixo de grupos sem qualquer estrutura hierárquica, tipicamente otimizando algum critério. O objetivo geral é obter uma partição dos dados que forneça o melhor valor para este critério. Os algoritmos particionais têm a vantagem em aplicações que envolvem grandes conjuntos de dados para os quais a construção de um dendrograma é computacionalmente muito complexa. Além disso, como os métodos hierárquicos consideram apenas os vizinhos locais em cada etapa, eles não podem incorporar conhecimento a priori sobre a forma ou tamanho global dos grupos (FRIGUI; KRISHNAPURAM, 1999).

Os algoritmos particionais podem ser divididos pela *forma como encontram os grupos*. Existem diversos tipos, mas aqui, iremos destacar os três mais importantes:

– **Agrupamento baseado em distância** é o caso onde os grupos são formados segundo a distância entre os padrões. A grande maioria dos algoritmos tentam encontrar centroides (protótipos) de dados, que são representantes de cada grupo, e agrupar instâncias com base em sua proximidade a eles. O algoritmo de agrupamento mais conhecido e amplamente utilizado é baseado em distância, *K-Means* (*KM*) (MACQUEEN, 1967). Outros algoritmos de destaque são: *Fuzzy C-Means* (*FCM*) (BEZDEK, 2013), *Partitioning Around Medoids (PAM)* (KAUFMANN, 1987), *Clustering LARge Applications (CLARA)* (KAUFMAN; ROUSSEEUW, 2009) e *Clustering Large Applications based on RANdomized Search (CLARANS)* (NG; HAN, 2002).

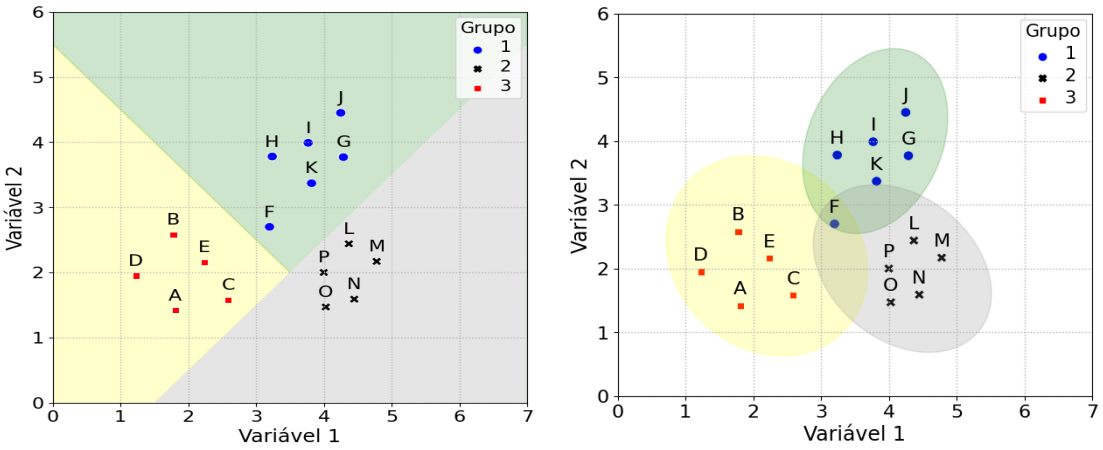
– **Agrupamento baseado em modelo** otimiza o ajuste entre um conjunto de dados e um modelo específico para cada grupo. A ideia básica por trás desses algoritmos é que os dados são gerados por uma mistura de distribuições de probabilidade subjacentes. Existem dois tipos de algoritmos de agrupamento baseados em modelo: um baseado em aprendizado estatístico e outro baseado em aprendizado de rede neural. Abordagens estatísticas recorrem a medidas de probabilidade na determinação dos grupos, enquanto abordagens de rede neural usam um conjunto de neurônios interconectados, onde cada conexão tem um peso associado (WEGMANN et al., 2021). O objetivo é estimar os parâmetros associados a essas distribuições ou os pesos da rede neural e alocar os objetos aos grupos. *MCLUST*, *EM* e *COBWED* são exemplos populares de algoritmos baseados em modelo.

– A ideia básica do **agrupamento baseado em densidade** é que os objetos são particionados em regiões de alta densidade, de modo que cada região com alta densidade representa um grupo de dados, e cada grupo é separado por regiões de baixa densidade. A principal vantagem dessa abordagem em particular é que ela não exige que o número de grupos seja definido antecipadamente. Além disso, algoritmos baseados em densidade são geralmente capazes de descobrir grupos de formas arbitrárias e conseguem detectar discrepâncias e ruídos. Contudo, eles podem falhar na identificação correta dos grupos se os dados não possuírem regiões de baixa densidade devido à presença de variáveis não espaciais (LACLAU, 2016). Alguns algoritmos populares baseados em densidade incluem *DBSCAN* (ESTER et al., 1996), *DENCLUE* (HINNEBURG; KEIM et al., 1998) e *OPTICS* (ANKERST et al., 1999).

Esses algoritmos podem ser ainda divididos pela *forma como atribuem os objetos aos grupos*. Nesse caso, podemos ter algoritmos de agrupamento hard (crisp) e algoritmos de agrupamento fuzzy (soft). Na abordagem de agrupamento hard, os objetos pertencem a um único grupo. Consequentemente, os grupos são disjuntos e não sobrepostos (ver Figura 5 (a)). No agrupamento fuzzy, os padrões podem pertencer a todos os grupos, com um certo grau

de pertencimento. Nesse caso, os grupos podem ser sobrepostos (ver Figura 5 (b)).

Figura 5 – Agrupamento obtido a partir do conjunto de dados (Ver Figura 4(a)). (a) Agrupamento hard. (b) Agrupamento fuzzy.



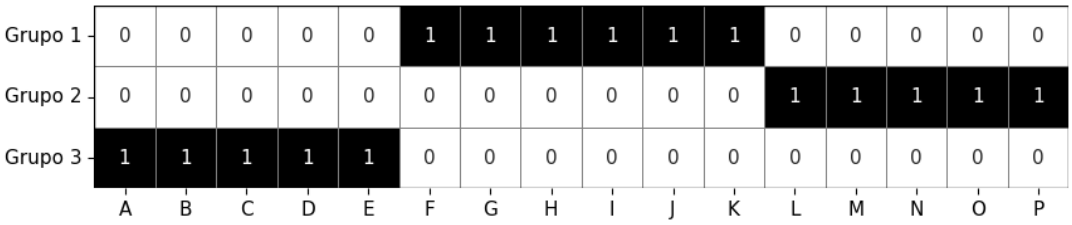
(a) (b)

Fonte: Autor (2023)

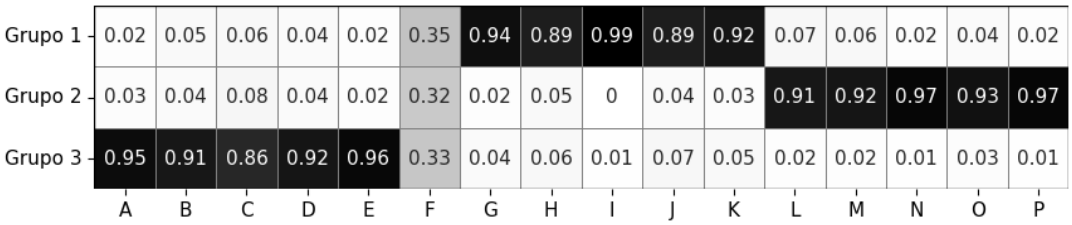
Em aplicações reais, onde diversos conjuntos de dados apresentam milhares de objetos e variáveis, a abordagem fuzzy pode ser mais apropriada, pois os grupos podem se sobrepor e algumas observações podem estar situadas em regiões de limites. Para exemplificar, considere o conjunto de dados com 16 objetos apresentado na Figura 4 (a). Podemos ver claramente a existência de três grupos: o primeiro formado pelos objetos A, B, C, D e E, o segundo formado pelos elementos G, H, I, J e K, e o último formado pelos elementos L, M, N, O e P. Também podemos observar que o objeto F está situado entre os três grupos. Como na abordagem hard os elementos são atribuídos a exatamente um único grupo, o objeto F é alocado no grupo 1 (Figura 5 (a)). A matriz de partição obtida a partir desse exemplo está na Figura 6 (a).

A partir desse exemplo podemos notar que a abordagem hard pode ser muito restritiva para capturar uma representação real dos dados. Ao lidar com as incertezas inerentes aos dados, o agrupamento fuzzy consegue obter uma melhor representação em casos como esse. Esses algoritmos não forçam os objetos a pertencerem a apenas um grupo, sendo vantajoso em situações como a do elemento F, que está, de certa forma, equidistante dos três grupos. Para esse exemplo, a matriz de pertencimento fuzzy obtida a partir da execução de um algoritmo fuzzy está mostrada na Figura 6 (b). Observe que o elemento F agora pertence a todos os grupos, com graus de pertencimento muito próximos, refletindo corretamente a sua posição no meio dos grupos.

Figura 6 – Partições obtidas do conjunto de dados (Ver Figura 4(a)).



(a) Partição hard



(b) Partição fuzzy

Fonte: Autor (2023)

2.3 MEDIDAS DE DISTÂNCIA

Como mencionado anteriormente, o principal objetivo em análise de agrupamento é dividir os dados em grupos similares. Por similaridade, entende-se como uma medida que quantifica quão parecidos dois objetos são. Se tratando especificamente da classe de algoritmos baseado em distância (foco principal desse trabalho), a noção de similaridade (ou dissimilaridade) entre os dados é introduzida através de medidas de distância. Nesse sentido, essas medidas desempenham um papel importante nos algoritmos de agrupamento.

Formalmente, uma medida de distância em um conjunto X é uma função d definida como:

$$d : X \times X \longrightarrow \mathbb{R}^+,$$

onde $\mathbb{R}^+$ refere-se ao conjunto dos números reais não-negativos. Além disso, para três objetos $\mathbf{x}, \mathbf{y}, \mathbf{z}$ em X, uma medida de distância deve satisfazer as seguintes condições:

- $d(\mathbf{x}, \mathbf{y}) \geq 0$ (não-negatividade);
- $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$ (distância é nula se, e somente se, dois objetos são iguais);
- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (simetria);
- $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ (desigualdade triangular).

Abaixo estão listadas as definições de algumas distâncias ou dissimilaridades comumente utilizadas no contexto de agrupamento. Para isso considere dois objetos $\mathbf{x} = (x_1, ..., x_d)$ e $\mathbf{y} = (y_1, ..., y_d) \in \mathbb{R}^d$.

1. **Distância de Minkowski**:

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{\frac{1}{p}}$$

A distância de Minkowski é provavelmente a medida de distância mais utilizada em algoritmos de agrupamento. Isso porque quando $p = 2$ a distância de Minkowski se torna a distância euclidiana, a mais comumente utilizada em agrupamento, e quando $p = 1$ obtém-se a distância city-block (também chamada de Manhattan). Quando $p \to \infty$, obtém-se a distância de Tchebychev, que mede a máxima magnitude em qualquer dimensão.

2. **Distância de Mahalanobis**:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y})},$$

onde $\mathbf{S}^{-1}$ é a inversa da matriz de covariância entre $\mathbf{x}$ e $\mathbf{y}$. Se $\mathbf{S}$ é a matriz identidade, a distância de mahalanobis coincide com a distância euclidiana. A ideia da distância de mahalanobis é considerar a covariância entre os objetos, o que é vantajoso em casos onde os vetores de dados não sejam independentes.

3. **Similaridade por cosseno**

$$1 - \cos(\theta) = 1 - \frac{\sum_{j=1}^{d} x_j y_j}{\left(\sqrt{\sum_{j=1}^{d} x_j^2}\right)\left(\sqrt{\sum_{j=1}^{d} y_j^2}\right)} \tag{2.1}$$

A métrica dada na Equação (2.1) não se trata de uma medida de distância, pois não satisfaz todas as condições citadas acima. Contudo, a similaridade por cosseno é muito popular em aplicações de dados esparsos, como, por exemplo, em agrupamento de documentos. A similaridade por cosseno varia entre -1, que significa que os objetos são completamente diferentes, a 1, que significa que os objetos são iguais.

Uma medida de similaridade amplamente utilizada nos algoritmos de agrupamento particional é a distância euclidiana quadrada, isto é

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||^2 = \sum_{j=1}^{d} (x_j - y_j)^2. \tag{2.2}$$

Embora seja sensível a outliers, a distância euclidiana quadrada é bem empregada na literatura, por questões de facilidade na manipulação matemática para a obtenção das equações dos

algoritmos. Os algoritmos particionais mais populares, o *K-Means* (*KM*) e o *Fuzzy C-Means* (*FCM*), são baseados na distância euclidiana quadrada. A seguir descreveremos o algoritmo *FCM*.

2.4 *FUZZY C-MEANS (FCM)*

Fuzzy C-Means (*FCM*) é uma extensão do algoritmo *K-Means* (*KM*) (MACQUEEN, 1967) para abordagem fuzzy que permite que os objetos pertençam a mais de um grupo simultaneamente. Esse algoritmo foi desenvolvido por Dunn (DUNN, 1973) e posteriormente aprimorado por Bezdek (BEZDEK, 1981).

Seja $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_i..., \mathbf{x}_N) = [x_{ij}]_{\substack{1 \leq i \leq N \\ 1 \leq j \leq P}}$ uma matriz de dados com N objetos (instâncias) descrita por P variáveis (atributos) contínuas. O objetivo do *FCM* é fornecer uma partição fuzzy dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ em K grupos e uma correspondente matriz de protótipos $\mathbf{C} = (\mathbf{c}_1, ..., \mathbf{c}_k, ..., \mathbf{c}_K) = [c_{kj}]_{\substack{1 \leq k \leq K \\ 1 \leq j \leq P}}$ de modo que o critério $\mathcal{J}_{FCM}$, que mede o ajuste entre os elementos e os protótipos de cada grupo, seja localmente minimizado. O critério $\mathcal{J}_{FCM}$ é definido da seguinte forma:

$$\mathcal{J}_{FCM}(\mathbf{C}, \mathbf{U}) = \sum_{k=1}^{K} \sum_{i=1}^{N} (u_{ik})^m ||\mathbf{x}_i - \mathbf{c}_k||^2 = \sum_{k=1}^{K} \sum_{i=1}^{N} (u_{ik})^m \sum_{j=1}^{P} (x_{ij} - c_{kj})^2 , \tag{2.3}$$

onde

$$||\mathbf{x}_i - \mathbf{c}_k||^2 = \sum_{j=1}^{P} (x_{ij} - c_{kj})^2 ,$$

é a distância euclidiana quadrada que mede a similaridade entre um objeto $\mathbf{x}_i = (x_{ij}, ..., x_{Nj})$ $(j = 1, ..., P)$ e um protótipo $\mathbf{c}_k = (c_{kj}, ..., c_{Kj})$ $(j = 1, ..., P)$, u_{ik} representa o grau de pertencimento do i-ésimo objeto no k-ésimo grupo, sujeito a $\sum_{k=1}^{K} u_{ik} = 1 \ \forall i$ e $u_{ik} \geq 0$, e $m \in (1, \infty)$ é o parâmetro que controla a imprecisão de cada objeto i, isto é, se eles têm pertinência significativa em mais de um grupo. Conforme $m \to \infty$, a partição se torna completamente fuzzy $\left(u_{ik} = \frac{1}{K}\right)$. Por outro lado, conforme $m \to 1$ a partição tende a ser hard.

O *FCM* define uma solução inicial dos graus de pertencimento $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ sujeito a $\sum_{k=1}^{K} u_{ik} = 1 \ \forall i$ e $u_{ik} \geq 0$, e alterna entre uma etapa de representação e uma etapa de alocação até a convergência, que ocorre quando a função objetivo $\mathcal{J}_{FCM}$ alcança um valor estacionário ou quando o algoritmo atinge um número máximo de iterações T.

Na etapa de representação os graus de pertencimento dos objetos u_{ik} $(i = 1, ..., N$ e $k = 1, ..., K)$ são mantidos fixos. O objetivo aqui é encontrar os protótipos $\mathbf{c}_k$ $(k = 1, ..., K)$

que minimizam o critério $\mathcal{J}_{FCM}$. Tomando a derivada de $\mathcal{J}_{FCM}$ em relação a $\mathbf{c}_k$ e igualando a zero, temos:

$$\frac{\partial \mathcal{J}_{FCM}}{\partial \mathbf{c}_k} = \sum_{i=1}^{N}(u_{ik})^m (\mathbf{x}_i - \mathbf{c}_k) = \sum_{i=1}^{N}(u_{ik})^m \mathbf{x}_i - \mathbf{c}_k \sum_{i=1}^{N}(u_{ik})^m = 0 \tag{2.4}$$

Resolvendo a Equação(2.4) obtém-se que o protótipo $\mathbf{c}_k$ $(k = 1, .., K)$ que minimizam o critério $\mathcal{J}_{FCM}$ é dado por:

$$\mathbf{c}_k = \frac{\sum_{i=1}^{N}(u_{ik})^m \mathbf{x}_i}{\sum_{i=1}^{N}(u_{ik})^m} \tag{2.5}$$

Na etapa de alocação, os protótipos $\mathbf{c}_k$ $(k = 1, .., K)$ são mantidos fixos e o objetivo é encontrar os graus de pertinência u_{ik} da matriz $\mathbf{U}$ que minimizam a função objetivo $\mathcal{J}_{FCM}$. Para minimizar o critério $\mathcal{J}_{FCM}$ definido na Equação (2.3) com respeito às condições impostas a u_{ik}, usamos o método dos multiplicadores de Lagrange e obtemos a seguinte função Lagrangiana:

$$\mathcal{L}_{FCM} = \sum_{k=1}^{K}\sum_{i=1}^{N}(u_{ik})^m ||\mathbf{x}_i - \mathbf{c}_k||^2 - \sum_{i=1}^{N}\theta_i \left(\sum_{k=1}^{K}(u_{ik}) - 1\right), \tag{2.6}$$

onde θ_i é o multiplicador de Lagrange.

Tomando a derivada parcial de $\mathcal{L}_{FCM}$ em relação a u_{ik} e igualando a zero, após algumas manipulações matemáticas, chegamos a:

$$u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{||\mathbf{x}_i - \mathbf{c}_k||^2}{||\mathbf{x}_i - \mathbf{c}_r||^2}\right)^{\frac{1}{m-1}}\right]^{-1} = \left[\sum_{r=1}^{K}\left(\frac{\sum_{j=1}^{P}(x_{ij} - c_{kj})^2}{\sum_{j=1}^{P}(x_{ij} - c_{rj})^2}\right)^{\frac{1}{m-1}}\right]^{-1} \quad \forall i, k \tag{2.7}$$

O algoritmo *FCM* converge para um mínimo local (BEZDEK, 1981) e, consequentemente, o algoritmo fica sensível à escolha dos valores iniciais, de modo que diferentes inicializações podem levar a resultados completamente diferentes. A complexidade de tempo do *FCM* é $O(NPK^2T)$ (HORE; HALL; GOLDGOF, 2007), sendo possível reduzi-la para $O(NPKT)$ (KOLEN; HUTCHESON, 2002), onde T é o número de iterações requeridas para estabilizar o algoritmo e alcançar a convergência. O Algoritmo 1 resume os passos do *FCM*.

Algoritmo 1 *Fuzzy C-Means* (*FCM*)

Entrada: O conjunto de dados $\mathbf{X}$, o número de grupos K, o parâmetro fuzzificador m, o número máximo de iterações T e o limite de erro ε, com $0 < \varepsilon << 1$.

Saída: A matriz de protótipos $\mathbf{C}$ e a matriz de pertinência fuzzy dos objetos $\mathbf{U}$.

Inicialização:

$\mathbf{U}$: inicializar aleatoriamente os graus de pertencimentos dos objetos u_{ik} $\forall i, k$, sujeito a $u_{ik} \in [0, 1]$ e $\sum_{k=1}^{K} u_{ik} = 1$;

$t \leftarrow 0$;

repita

$t \leftarrow t + 1$;

Etapa 1: Representação .

Atualize os protótipos $\mathbf{c}_k$ $(k = 1, ..., K)$ de acordo com a Equação (2.5).

Etapa 2: Alocação

Atualize os graus de pertinência u_{ik} da matriz de pertencimento dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de acordo com a Equação (2.7.

até $|\mathcal{J}_{FCM}^{(t)} - \mathcal{J}_{FCM}^{(t-1)}| \leq \varepsilon$ ou $t > T$.

2.5 AGRUPAMENTO BASEADO EM KERNEL

Desde o início do século XXI funções kernel têm sido amplamente utilizadas em análise de agrupamento, de modo que diversos pesquisadores têm demonstrado interesse em desenvolver métodos baseados em kernel (BEN-HUR et al., 2001; FILIPPONE et al., 2008). A principal ideia acerca desses algoritmos é a realização de um mapeamento não-linear, Φ, do espaço de entrada dos dados para outro espaço de mais alta dimensão (possivelmente infinita), conhecido como espaço de kernel. Na prática, se os grupos não são facilmente identificados (por exemplo, se um par de grupos não é linearmente separável no espaço de entrada dos dados), então os dados podem ser mapeados do espaço de entrada para o espaço de kernel, com a esperança de que os grupos estejam bem definidos nesse novo espaço, e que encontrá-los seria um trabalho mais fácil.

Seja $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ um conjunto não-vazio de objetos no espaço P-dimensional $\mathbb{R}^P$. Uma função $\mathcal{K}$: $X \times X \rightarrow \mathbb{R}$ é dita ser um kernel positivo-definido ou kernel de Mercer se, e somente se, $\mathcal{K}$ é simétrica ($\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i)$) e a seguinte inequação é válida (MERCER,

1909):

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \mathcal{K}\left(\mathbf{x}_i, \mathbf{x}_j\right) \geq 0,\ \forall n \geq 2,$$

onde $c_r \in \mathbb{R}\ \forall r = 1, \ldots, n$.

O uso de funções kernel é embasado no teorema de Cover (HAYKIN, 1994), que garante que um conjunto de dados com padrões não-linearmente separáveis pode tornar-se separável de forma linear através de uma transformação do espaço dos dados de entrada, desde que essa transformação seja não-linear e que a dimensão do novo espaço seja suficientemente alta (possivelmente infinita).

Seja $\Phi : X \rightarrow \mathcal{F}$ um mapeamento não-linear do espaço de entrada X para um espaço de mais alta dimensão, $\mathcal{F}$, conhecido como espaço de kernel. Ao aplicar o mapeamento Φ, o produto interno $\mathbf{x}_i^\top \mathbf{x}_j$ é mapeado para $\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ no espaço de kernel. A ideia chave por trás dos algoritmos de agrupamento baseados em kernel é que o mapeamento não-linear Φ não precisa ser explicitamente especificado porque todo kernel de Mercer pode ser escrito como

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j), \tag{2.8}$$

que é comumente conhecido como *kernel trick* (MULLER et al., 2001).

Baseado no *kernel trick*, distâncias euclidianas são calculadas no espaço de kernel $\mathcal{F}$ como:

$$\begin{aligned} ||\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)||^2 &= [\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)]^\top [\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)] \\ &= \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_i) - 2\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j) + \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_j) \\ &= \mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) - 2\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) + \mathcal{K}(\mathbf{x}_j, \mathbf{x}_j) \end{aligned} \tag{2.9}$$

Alguns exemplos de funções kernel estão listados na Tabela 1.

Tabela 1 – Exemplos de funções kernel.

Kernel	Expressão	Restrições dos parâmetros
Gaussiano	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\|\mathbf{x}_i - \mathbf{x}_j\|\|^2}{2\sigma^2}\right)$	$\sigma > 0$
Polinomial	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\lambda \mathbf{x}_i^\top \mathbf{x}_j + \theta)^d$	$\lambda > 0,\ \theta \geq 0,\ d \in \mathbb{N}$
Linear	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	—
Laplaciano	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\lambda \|\|\mathbf{x}_i - \mathbf{x}_j\|\|\right)$	$\lambda > 0$
Sigmoidal	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\lambda \mathbf{x}_i^\top \mathbf{x}_j + \theta)$	$\lambda > 0,\ \theta \geq 0,$

Os algoritmos baseados em kernel têm sido desenvolvidos sob duas principais abordagens: agrupamento no espaço de características (alguns autores nomeiam essa abordagem como kernelização da métrica (FERREIRA; CARVALHO, 2014; FERREIRA; CARVALHO; SIMÕES, 2016)) e agrupamento no espaço de kernel (alguns autores denotam essa variação como agrupamento no espaço de características (GIROLAMI, 2002; FERREIRA; CARVALHO, 2014)). Os algoritmos de agrupamento no espaço de características buscam por centroides no espaço original dos dados (ZHANG; CHEN, 2003; GRAVES; PEDRYCZ, 2010; SHEN et al., 2006) e os mapeiam implicitamente para o espaço do kernel através de uma função kernel. Dessa forma, a distância entre um objeto $\mathbf{x}_i$ e um centroide $\mathbf{c}_k$ no espaço de kernel pode ser calculada como:

$$||\Phi(\mathbf{x}_i) - \Phi(\mathbf{c}_k)||^2 = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) - 2\mathcal{K}(\mathbf{x}_i, \mathbf{c}_k) + \mathcal{K}(\mathbf{c}_k, \mathbf{c}_k)$$

Por outro lado, os métodos de agrupamento no espaço de kernel realizam um mapeamento dos dados de entrada através de uma função não-linear e busca pelos centroides no espaço de kernel (GRAVES; PEDRYCZ, 2010; MULLER et al., 2001; SCHÖLKOPF; SMOLA; MÜLLER, 1998; FERREIRA; CARVALHO, 2014). No espaço de kernel, os centroides $\mathbf{c}_k^{\Phi}$ $(k = 1, ..., K)$ são computados da seguinte forma (GRAVES; PEDRYCZ, 2010; PAL; SARKAR, 2013):

$$\mathbf{c}_k^{\Phi} = \frac{\sum_{i=1}^{N} (u_{ik})^m \Phi(\mathbf{x}_i)}{\sum_{i=1}^{N} (u_{ik})^m} \quad \forall k \tag{2.10}$$

Como o mapeamento Φ não é conhecido, os protótipos $\mathbf{c}_k^{\Phi}$ $(k = 1, ..., K)$ não podem ser calculados diretamente de acordo com a Equação (2.10). Contudo, através do *kernel trick* sabemos que

$$||\Phi(\mathbf{x}_i) - \mathbf{c}_k^{\Phi}||^2 = \Phi(\mathbf{x}_i)^{\top} \Phi(\mathbf{x}_i) - 2\Phi(\mathbf{x}_i)^{\top} \mathbf{c}_k^{\Phi} + \left(\mathbf{c}_k^{\Phi}\right)^{\top} \mathbf{c}_k^{\Phi} \tag{2.11}$$

Substituindo a Equação (2.10) na Equação (2.11) (GRAVES; PEDRYCZ, 2010; PAL; SARKAR, 2013), chegamos a:

$$||\Phi(\mathbf{x}_i) - \mathbf{c}_k^{\Phi}||^2 = \mathcal{K}(\mathbf{x}_k, \mathbf{x}_k) - 2\frac{\sum_{r=1}^{N} (u_{ik})^m \mathcal{K}(\mathbf{x}_k, \mathbf{x}_r)}{\sum_{r=1}^{N} (u_{ir})^m} + \frac{\sum_{r=1}^{N} \sum_{s=1}^{N} (u_{ir})^m (u_{is})^m \mathcal{K}(\mathbf{x}_r, \mathbf{x}_s)}{\left(\sum_{r=1}^{N} (u_{ik})^m\right)^2} \tag{2.12}$$

Note que na Equação (2.12) a distância entre um objeto $\mathbf{x}_i$ e um centroide $\mathbf{c}_k^{\Phi}$ no espaço de kernel é calculado em função dos dados de entrada. Uma característica dos algoritmos no espaço de kernel é que para obter uma partição dos dados, seja ela hard ou fuzzy, não é necessário computar os protótipos dos grupos, pois a similaridade entre os objetos e os centroides no espaço de kernel é medida apenas baseada nos dados de entrada, segundo a Equação (2.12).

Uma vantagem dos algoritmos no espaço de kernel é que eles não restringem os protótipos a estarem no espaço de entrada, como procedem os algoritmos no espaço de característica. Contudo, uma desvantagem imediata está relacionado ao fato de que os protótipos estão implicitamente localizados no espaço do kernel e precisam ser aproximados por um mapeamento inverso ao espaço de entrada dos dados (GRAVES; PEDRYCZ, 2010). Um método para estimar iterativamente os protótipos no espaço de kernel foi proposto em (ZHOU; GAN, 2004).

Os algoritmos de agrupamento baseados em kernel mais populares são as versões kernelizadas do *K-Means* e do *Fuzzy C-Means*. São eles: Kernel K-Means no espaço de características, Kernel K-Means no espaço de kernel, Kernel Fuzzy C-Means no espaço de características e Kernel Fuzzy C-Means no espaço de kernel. Como o foco principal desse trabalho são os algoritmos fuzzy baseados em kernel no espaço de características, a seguir iremos detalhar o algoritmo *Kernel Fuzzy C-Means* no espaço de características. Como nossa abordagem aqui é apenas os algoritmos no espaço de características, iremos nos referir ao Kernel Fuzzy C-Means no espaço de características apenas como o *Kernel Fuzzy C-Means* (*KFCM*).

2.5.1 *Kernel Fuzzy C-Means (KFCM)*

O *Kernel Fuzzy C-Means* (*KFCM*) segue a ideia básica do *FCM* descrito na Seção 2.4. O *KFCM* busca minimizar a seguinte função objetivo:

$$\begin{aligned} \mathcal{J}_{KFCM}(\mathbf{C},\mathbf{U}) &= \sum_{k=1}^{K}\sum_{i=1}^{N}(u_{ik})^m||\Phi(\mathbf{x}_i)-\Phi(\mathbf{c}_k)||^2 \\ &= \sum_{k=1}^{K}\sum_{i=1}^{N}(u_{ik})^m\mathcal{K}(\mathbf{x}_i,\mathbf{x}_i) - 2\mathcal{K}(\mathbf{x}_i,\mathbf{c}_k) + \mathcal{K}(\mathbf{c}_k,\mathbf{c}_k), \end{aligned} \tag{2.13}$$

sujeito a $\sum_{k=1}^{K} u_{ik} = 1 \ \forall i$ e $u_{ik} \geq 0 \ \forall i,k$. m e u_{ik} são definidos da mesma forma que anteriormente (Seção 2.4).

Assim como *FCM*, o *KFCM* alterna entre uma etapa de representação e uma etapa de alocação até a convergência do critério $\mathcal{J}_{KFCM}$. Na etapa de inicialização, além dos graus de pertencimento $\mathbf{U} = [u_{ik}]_{\substack{1\leq i\leq N \\ 1\leq k\leq K}}$ sujeito a $\sum_{k=1}^{K} u_{ik} = 1 \ \forall i$ e $u_{ik} \geq 0$, o KFCM também inicializa aleatoriamente a matriz de protótipos $\mathbf{C}$.

Na etapa de representação o objetivo é encontrar os protótipos $\mathbf{c}_k$ $(k = 1, ..., K)$ que minimizam o critério $\mathcal{J}_{KFCM}$. Nessa etapa, os graus de pertencimento dos objetos u_{ik} $(i = 1, ..., N$ e $k = 1, ..., K)$ são mantidos fixos. A derivação dos protótipos depende da função de

kernel. Se nos restringirmos ao kernel gaussiano, isto é

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right), \quad \sigma > 0,$$

que é comumente utilizado na literatura, então $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) = 1$. Dessa forma, o critério $\mathcal{J}_{KFCM}$ dado na Equação (2.13) pode ser expresso como:

$$\mathcal{J}_{KFCM}(\mathbf{C}, \mathbf{U}) = 2\sum_{k=1}^{K}\sum_{i=1}^{N}(u_{ik})^m\left(1 - \mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)\right) \tag{2.14}$$

Tomando a derivada de $\mathcal{J}_{KFCM}$, dada na Equação (2.14), em relação a $\mathbf{c}_k$ e igualando a zero, temos:

$$\frac{\partial \mathcal{J}_{KFCM}}{\partial \mathbf{c}_k} = \sum_{i=1}^{N}(u_{ik})^m\mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)\mathbf{x}_i - \mathbf{c}_k\sum_{i=1}^{N}(u_{ik})^m\mathcal{K}(\mathbf{x}_i, \mathbf{c}_k) = 0 \tag{2.15}$$

Resolvendo a Equação(2.15), obtém-se que os protótipos $\mathbf{c}_k$ $(k = 1, .., K)$, que minimizam o critério $\mathcal{J}_{KFCM}$, são computados como:

$$\mathbf{c}_k = \frac{\sum_{i=1}^{N}(u_{ik})^m\mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)\mathbf{x}_i}{\sum_{i=1}^{N}(u_{ik})^m\mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)} \tag{2.16}$$

Na etapa de alocação, os protótipos $\mathbf{c}_k$ $(k = 1, .., K)$ são mantidos fixos e o objetivo é encontrar os graus de pertinência u_{ik} da matriz $\mathbf{U}$ que minimizam a função objetivo $\mathcal{J}_{KFCM}$. Para minimizar o critério $\mathcal{J}_{KFCM}$ definido na Equação (2.14) com respeito às condições impostas a u_{ik}, usamos o método dos multiplicadores de Lagrange e obtemos a seguinte função lagrangiana:

$$\mathcal{L}_{KFCM} = 2\sum_{k=1}^{K}\sum_{i=1}^{N}(u_{ik})^m\left(1 - \mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)\right) - \sum_{i=1}^{N}\theta_i\left(\sum_{k=1}^{K}(u_{ik}) - 1\right), \tag{2.17}$$

onde θ_i é o multiplicador de Lagrange.

Tomando a derivada parcial de $\mathcal{L}_{KFCM}$ em relação a u_{ik} e igualando a zero, após algumas manipulações matemáticas, chegamos a:

$$u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{1 - \mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)}{1 - \mathcal{K}(\mathbf{x}_i, \mathbf{c}_r)}\right)^{\frac{1}{m-1}}\right]^{-1} \forall i, k \tag{2.18}$$

O Algoritmo 2 apresenta os passos para a execução do *KFCM*. O *KFCM* converge para um mínimo local e, assim, como o *FCM*, é sensível à escolha dos valores iniciais.

Algoritmo 2 *Kernel Fuzzy C-Means* (*KFCM*)

Entrada: O conjunto de dados $\mathbf{X}$, o número de grupos K, o parâmetro fuzzificador m, o hiperparâmetro do kernel gaussiano σ^2,o número máximo de iterações T e o limite de erro ε, com $0 < \varepsilon << 1$.

Saída: A matriz de protótipos $\mathbf{C}$ e a matriz de pertinência fuzzy dos objetos $\mathbf{U}$.

Inicialização:

$\mathbf{U}$: inicializar aleatoriamente os graus de pertencimentos dos objetos u_{ik} $\forall i, k$, sujeito a $u_{ik} \in [0.1]$ e $\sum_{k=1}^{K} u_{ik} = 1$;
$\mathbf{C}$: inicializar aleatoriamente os protótipos $\mathbf{c}_k$ $(k = 1, ..., K)$. Escolhe-se aleatoriamente K objetos do conjunto de dados como sendo os centroides;

$t \leftarrow 0$;

repita
$t \leftarrow t + 1$;
Etapa 1: Representação .

Atualize os protótipos $\mathbf{c}_k$ $(k = 1, ..., K)$ de acordo com a Equação (2.16).

Etapa 2: Alocação

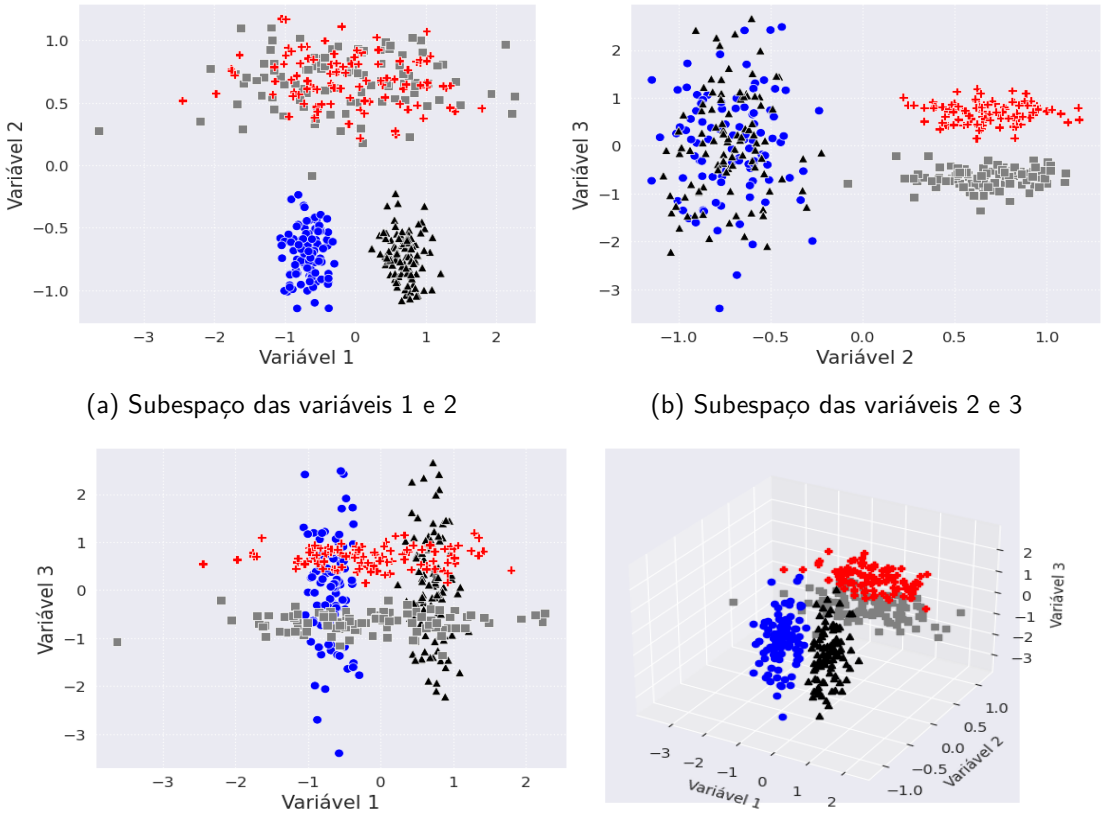
Atualize os graus de pertinência u_{ik} da matriz de pertencimento dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de acordo com a Equação (2.18.

até $|\mathcal{J}_{KFCM}^{(t)} - \mathcal{J}_{KFCM}^{(t-1)}| \leq \varepsilon$ ou $t > T$.

2.6 AGRUPAMENTO DE DADOS COM PONDERAÇÃO DAS VARIÁVEIS

Os algoritmos de agrupamento tradicional, como *KM* e *FCM*, consideram todas as variáveis de um conjunto de dados de entrada na tentativa de aprender o máximo possível sobre a estrutura de grupos inerente a esses dados. No entanto, em dados de alta dimensão muitas das variáveis são frequentemente irrelevantes, no sentido de que os grupos não estão bem definidos para esses atributos. Essas variáveis não informativas podem confundir os algoritmos de agrupamento, ocultando grupos em dados ruidosos. Em dimensões muito altas, é comum que todos os objetos em um conjunto de dados estejam quase equidistantes uns dos outros, mascarando completamente os grupos. Problema esse conhecido como maldição da dimensionalidade (BELLMAN, 1966). Nesse sentido, pode haver grupos melhores definidos em apenas algumas dimensões do espaço de alta dimensão (Ver Figura 7). Veja que apenas dois grupos estão bem definidos no subespaço das variáveis 1 e 2 (Figura 7 (a)), enquanto os outros dois grupos estão separados no subespaço das variáveis 2 e 3 (Figura 7 (b)).

Figura 7 – Visualização de diferentes subespaços de variáveis.



(a) Subespaço das variáveis 1 e 2

(b) Subespaço das variáveis 2 e 3

(c) Subespaço das variáveis 1 e 3

(d) Subespaço das variáveis 1, 2 e 3

Fonte: Autor (2023)

Diversos métodos de agrupamento de dados em alta dimensão incluem técnicas de transformação e seleção de variáveis. No primeiro caso, os dados passam por uma transformação, na qual compreende uma etapa de pré-processamento, e os algoritmos são aplicados aos novos atributos gerados. A ideia é resumir o conjunto de dados em menos dimensões, criando combinações das variáveis originais. Esses métodos incluem técnicas como *Principal Component Analisys (PCA)* (JOLLIFFE, 2002). No entanto, como preservam as distâncias relativas entre objetos, eles são menos eficazes quando muitos atributos são irrelevantes. Além disso, as novas variáveis são combinações das originais, tornando-as difíceis de interpretar. Por outro lado, as técnicas de seleção de variáveis selecionam apenas as variáveis mais relevantes, aquelas nas quais os grupos estão mais bem definidos. Apesar de sua utilidade em muitos conjuntos de dados, os algoritmos de seleção de variáveis têm dificuldade em encontrar grupos em diferentes subespaços, o que motivou a evolução dos algoritmos no contexto de *Subspace Clustering* (*SC*).

Subspace Clustering (*SC*) é uma extensão da seleção de variáveis na qual tenta identificar

grupos em diferentes subespaços ou em um subespaço específico do conjunto de dados de entrada. Esses métodos são divididos em duas categorias: *Hard Subspace Clustering* (*HSC*) e *Soft Subspace Clustering* (*SSC*). Na primeira categoria, os algoritmos determinam os subconjuntos de variáveis onde os grupos são descobertos. Por outro lado, os algoritmos em *SSC* agrupam os dados atribuindo peso a cada variável para quantificar a contribuição de cada uma na formação dos grupos. A abordagem de *Soft Subspace Clustering* pode ser vista como uma extensão do framework *Feature Weighting Clustering* (*FWC*) (HUANG et al., 2005; TSAI; CHIU, 2008). Enquanto os algoritmos em *SSC* consideram um vetor de pesos de variáveis para cada grupo de dados, os métodos em *FWC* consideram um único vetor de pesos para todo o conjunto de dados.

Vários algoritmos em *SSC* foram propostos (KELLER; KLAWONN, 2000; JING; NG; HUANG, 2007; WANG et al., 2016; RODRÍGUEZ; CARVALHO, 2017). Chitsaz e Jahromi (CHITSAZ; JAHROMI, 2016) propuseram um algoritmo capaz de encontrar grupos em diferentes subespaços e detectar ruídos nos dados, simultaneamente. Jia e Cheung (JIA; CHEUNG, 2017) propuseram um algoritmo com ponderação das variáveis para dados mistos, i.e. dados numéricos e categóricos. Mais recentemente, Rodriguez e de Carvalho (RODRIGUEZ; CARVALHO, 2021) propuseram alguns algoritmos baseados na distância Euclidiana e distância City-block que combinavam regularização de entropia com *SSC*. Muitos algoritmos que distinguem os pesos de relevância das variáveis foram desenvolvidos sob o conceito de distâncias adaptativas (CARVALHO; TENÓRIO; JUNIOR, 2006; CARVALHO; LECHEVALLIER, 2009; FERREIRA; CARVALHO, 2014). Esse tipo de distância permite computar os pesos das variáveis para todo o conjunto de dados (distância adaptativa global) ou para cada grupo de dados (distância adaptativa local). Nesse caso, os algoritmos baseados na distância adaptativa global se enquadram na abordagem de *Feature Weighting Clustering*, enquanto os algoritmos com distância adaptativa local se enquadram na abordagem de *Soft Subspace Clustering*.

2.7 SÍNTESE DO CAPÍTULO

Esse capítulo apresenta uma revisão acerca dos algoritmos tradicionais de agrupamento. Foram abordadas as diferenças dos tipos de agrupamento, quais os métodos mais populares em cada tipo e as medidas de distâncias usadas como similaridade (ou dissimilaridade) nos algoritmos de agrupamento. Além disso, foi apresentada a teoria básica acerca de funções kernel, quais as abordagens de agrupamento baseado em kernel tem sido desenvolvidas na

literatura e quais os algoritmos baseados em kernel mais populares. Também foi apresentado uma revisão bibliográfica acerca dos algoritmos de agrupamento com ponderação das variáveis. Os algoritmos de agrupamento *Fuzzy C-Means* (*FCM*) e *Kernel Fuzzy C-Means* (*KFCM*), por estarem ligados com o tema central do trabalho, foram apresentados em detalhes.

3 ALGORITMOS DE CO-CLUSTERING

3.1 INTRODUÇÃO

No capítulo anterior, algoritmos de agrupamento convencional foram discutidos. Os tipos de agrupamento, as diferenças entre eles, os principais métodos em cada abordagem, bem como os principais algoritmos relacionados a esse trabalho foram descritos para demonstrar a essência de cada um. Embora os métodos de agrupamento convencional tenham sido aplicados com sucesso em diversas áreas, esses algoritmos não são capazes de explorar a relação intrínseca que pode existir entre objetos e variáveis.

Essa característica é comum em alguns campos do conhecimento como, por exemplo, mineração de texto, análise de genes e sistema de recomendações. Para exemplificar, na mineração de texto, onde o objetivo é encontrar a co-ocorrência de palavras (colunas) e documentos (linha), é racional pensar que um grupo de palavras está ligado a um grupo de documentos relacionados a um mesmo tema. Na expressão genética, onde os dados são representados por uma matriz de expressão de genes, cujas linhas representam os genes e as colunas representam as condições experimentais, o objetivo pode ser encontrar grupos de genes que apresentam as mesmas respostas em relação às condições experimentais. Por fim, em sistemas de recomendação, o objetivo pode ser agrupar usuários com base nas suas preferências e vínculos sociais com outros usuários. Embora a relação intrínseca entre objetos e variáveis exista de forma mais comum em áreas específicas, como citado anteriormente, essa característica também pode estar presente em diversos outros campos de aplicação. Nesse sentido, a fim de cobrir esse cenário de aprendizagem em aplicações de algoritmos de agrupamento, foram propostos os chamados algoritmos de co-clustering.

A ideia básica nos métodos de co-clustering é agrupar simultaneamente objetos e variáveis. Ao fornecer uma partição de objetos e outra de variáveis, em um número fixo de grupos, esses métodos organizam o conjunto de dados em blocos homogêneos, formados por um conjunto de objetos e variáveis cujos elementos são similares. Esses métodos conseguem fornecer insights a respeito da associação entre objetos e variáveis, o que pode ser útil para melhor agrupar os objetos (agrupamento mais comum), principalmente em aplicações a dados em alta dimensão.

Existem muitas estratégias pelas quais os métodos de co-clustering se baseiam para encontrar as partições de grupos e objetos e consequentemente formar os blocos de dados. Em relação às abordagens particionais hard e fuzzy, os algoritmos de co-clustering também po-

dem ser divididos nessas duas categorias. No caso hard, objetos e variáveis pertencem a um e, somente um, grupo. Sendo assim, as observações pertencem a um único bloco de dados (co-cluster). Na abordagem fuzzy, os objetos e variáveis podem pertencer a todos os grupos, com um certo grau de pertinência, de modo que as observações podem pertencer a todos os blocos de dados.

Os algoritmos de co-clustering podem ainda ser divididos nas seguintes categorias:

– **Mistura de modelos:** esses algoritmos partem da ideia de que a população é composta de vários blocos de dados gerados por diferentes distribuições de probabilidades, ou no caso mais simples e mais estudado, os blocos de dados são gerados a partir da mesma distribuição de probabilidade com diferentes parâmetros. O objetivo final é estimar os parâmetros das distribuições subjacentes aos blocos de dados. Alguns estudos podem ser encontrados em (GOVAERT; NADIF, 2003; GOVAERT; NADIF, 2008).

– **Fatoração matricial:** esses métodos decompõem uma matriz de dados em um produto de três matrizes, onde duas delas representam as partições de objetos e variáveis enquanto a terceira representa as ligações entre elas. Esses algoritmos incluem duas famílias principais: algoritmos baseados na fatoração de matriz não-negativa, do inglês Non-Negative Matrix Factorization (NMF) (LEE; SEUNG, 2000) e algoritmos baseados na decomposição dos valores dos blocos – Block Value Decomposition (LONG; ZHANG; YU, 2005).

– **Baseado em grafo:** esses algoritmos transformam os dados em uma representação de grafos bipartidos e usam uma função objetivo adequada para escolher como cortar as arestas dos grafos e formar os grupos de objetos e variáveis. Tais métodos têm sido desenvolvidos, principalmente, baseado na abordagem de Spectral Co-clustering (SpCO) (DHILLON, 2001). Outra abordagem envolvendo esses algoritmos é utilizar a otimização da modularidade dos grafos (LABIOD; NADIF, 2011; AILEM; ROLE; NADIF, 2015).

– **Baseado na teoria da informação:** esses algoritmos usam a teoria da informação como forma de dissimilaridade entre os dados. Por exemplo, o algoritmo Information-Theoretic Co-clustering (ITCC) (DHILLON; MALLELA; KUMAR, 2003) minimiza uma função objetivo que considera a informação mútua nos grupos. Outra abordagem foi proposta em (BANERJEE et al., 2004), na qual a informação mútua é substituída pela divergência de Bregman.

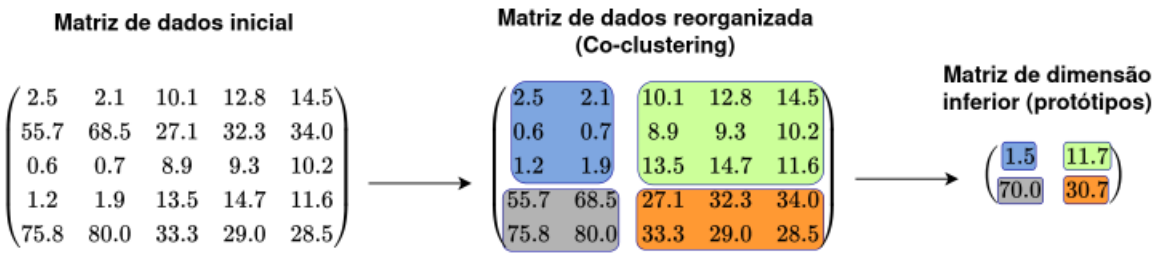
Segundo Govaert e Nadif (GOVAERT; NADIF, 2013) os métodos de co-clustering podem ainda ser divididos em duas principais abordagens: modelos baseados em métricas e modelos baseados em probabilidade. Os primeiros modelos definem um critério de adequação e otimizam esse critério através de alguma heurística adequada. Algoritmos baseados em fatoração

matricial, em grafos e na teoria da informação são exemplos de algoritmos baseados em métricas. Por outro lado, na abordagem probabilística os algoritmos se baseiam nos modelos de mistura generativos. Esses modelos assumem que o conjunto de dados é formado por diferentes blocos homogêneos, gerados por alguma distribuição de probabilidade (por exemplo, distribuição gaussiana e distribuição Bernoulli). Um algoritmo que se tornou bastante popular é o Latent Block Model (LBM) proposto em (GOVAERT; NADIF, 2003). A popularidade deste algoritmo se dá pelos seguintes motivos: (1) ao usar diferentes distribuições de probabilidades, consegue lidar com diferentes estruturas de dados, especificamente com dados contínuos, dados binários e matrizes de contingências; (2) é fortemente ligado aos algoritmos baseados em métricas, mas exige menos esforço computacional.

3.2 CO-CLUSTERING BASEADO EM MÉTRICAS

O foco principal deste trabalho são os algoritmos baseados em métricas, especificamente aqueles baseados em distâncias. Os algoritmos de co-clustering dessa abordagem otimizam uma função objetivo, fornecendo duas partições ótimas relacionadas aos objetos e variáveis. Esses métodos são capazes de organizar a matriz inicial de dados em blocos homogêneos (diferentemente dos algoritmos tradicionais de agrupamento, que organizam a matriz de dados em grupos de objetos), fornecendo uma matriz resumo (de dimensão inferior), geralmente chamada de matriz de protótipos, que consiste em valores representativos de cada bloco, onde esses blocos são definidos pelo par de partições relacionadas a objetos e variáveis. A Figura 8 exemplifica o resumo da matriz de dados inicial fornecida por um algoritmo de co-clustering considerando dois grupos de objetos e dois grupos de variáveis.

Figura 8 – Dados reorganizados de acordo com co-clustering.



Fonte: Autor (2023)

A seguir, serão descritos os dois algoritmos de co-clustering que estão mais relacionados com esse trabalho e foram usados para efeito comparativo com os algoritmos propostos: *Fuzzy*

Double K-Means (*FDK*) e *Weighted Fuzzy Double K-Means* (*WFDK*). Além desses métodos, alguns outros algoritmos baseados em distâncias mostraram sua eficiência em diversos casos. Por exemplo, os chamados "Métodos CRO" (GOVAERT, 1983). Essa família de métodos é formada por três métodos: *CROEUC*, *CROBIN* e *CROK12*, adequados para dados contínuos, binários e tabelas de contingências, respectivamente. Além desses, outro algoritmo bastante conhecido é o *Double K-Means* (*DK*) (ROCCI; VICHI, 2008), sendo a versão do *K-Means* (*KM*) na abordagem de co-clustering. Existem também os algoritmos *Diagonal Double K-Means (DDKM)* e *Fuzzy Diagonal Double K-Means (F-DDKM)* propostos em (LACLAU; NADIF, 2016), que consideram uma estrutura diagonal de blocos inerente a problemas relacionados ao agrupamento de dados textuais.

Notação. Para simplificar a notação relacionada aos somatórios, as somas de linhas, colunas, grupos de linhas e grupos de colunas serão indicadas apenas pelos índices relacionados, cujos intervalos de variação são dados respectivamente por $i = 1, ..., N$, $j = 1, ..., P$, $k = 1, ..., K$ e $h = 1, ..., H$, sem indicar os limites de variação. Por exemplo, $\sum_i$ substitui $\sum_{i=1}^{N}$, $\sum_{k,i}$ substitui $\sum_{k=1}^{K}\sum_{i=1}^{N}$ e $\sum_{k,h,i,j}$ substitui $\sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{P}$.

3.2.1 *Fuzzy Double K-Means (FDK)*

O algoritmo *Fuzzy Double K-Means* (*FDK*) (LACLAU; CARVALHO; NADIF, 2015) é uma extensão do algoritmo *DK* (ROCCI; VICHI, 2008) para a abordagem fuzzy. Antes de descrever o algoritmo *FDK*, iremos apresentar brevemente o algoritmo *DK*.

O objetivo do algoritmo *DK* é minimizar uma função objetivo $\mathcal{J}_{DK}(\mathcal{Z}, \mathcal{W}, \mathbf{G})$, onde $\mathcal{Z} = \{\mathcal{Z}_1, ..., \mathcal{Z}_k, ..., \mathcal{Z}_K\}$ é uma partição crisp de N objetos em K grupos, $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_h, ..., \mathcal{W}_H\}$ é uma partição crisp de P variáveis em H grupos e $\mathbf{G} = [g_{kh}]_{\substack{1\leq k\leq K\\ 1\leq h\leq H}}$ é a matriz de protótipos dos blocos de dados, vista como uma matriz resumo da matriz de dados inicial $\mathbf{X} = [x_{ij}]_{\substack{1\leq i\leq N\\ 1\leq j\leq P}}$, onde cada elemento g_{kh} é conhecido como protótipo do bloco $\mathbf{X}_{kh} = [x_{ij}]_{\substack{i\in\mathcal{Z}_k\\ j\in\mathcal{W}_h}}$. O algoritmo *DK* adota a distância Euclidiana quadrada para medir a similaridade entre os dados x_{ij} e os protótipos g_{kh}, de modo que o critério $\mathcal{J}_{DK}$ é definido como:

$$\mathcal{J}_{DK}(\mathcal{Z}, \mathcal{W}, \mathbf{G}) = \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i\in\mathcal{Z}_k}\sum_{j\in\mathcal{W}_h}(x_{ij} - g_{kh})^2 \tag{3.1}$$

Para valores fixos de Z e W, os valores ótimos de $\mathbf{G}$, que minimizam o critério $\mathcal{J}_{DK}$, dado

na Equação (3.1), são as médias dos blocos $\mathbf{X}_{kh}$, isto é:

$$g_{kh} = \frac{\sum_{i \in \mathcal{Z}_k} \sum_{j \in \mathcal{W}_h} x_{ij}}{|\mathcal{Z}_k| \times |\mathcal{W}_h|}, \quad \forall k, h$$

onde $|.|$ denota a cardinalidade da partição. Após essa etapa de representação, os grupos de objetos $\mathcal{Z}_k$ $(k = 1, ..., K)$, que compõem a partição ótima Z, são atualizados como:

$$\mathcal{Z}_k = \left\{ i \in \{1, ..., N\} : \sum_{h=1}^{H} \sum_{j \in \mathcal{W}_h} (x_{ij} - g_{kh})^2 = \min_{1 \leq r \leq K} \sum_{h=1}^{H} \sum_{j \in \mathcal{W}_h} (x_{ij} - g_{rh})^2 \right\}$$

No último passo do algoritmo, os grupos de variáveis $\mathcal{W}_h$ $(h = 1, ..., H)$, que compõem a partição ótima W, são atualizados como:

$$\mathcal{W}_h = \left\{ j \in \{1, ..., P\} : \sum_{k=1}^{K} \sum_{i \in \mathcal{Z}_k} (x_{ij} - g_{kh})^2 = \min_{1 \leq s \leq H} \sum_{k=1}^{K} \sum_{i \in \mathcal{Z}_k} (x_{ij} - g_{ks})^2 \right\}$$

A partir de uma solução inicial das partições Z e W o algoritmo realiza um processo iterativo de atualização dos protótipos, partição de objetos e partição de variáveis até que o critério $\mathcal{J}_{DK}$ alcance um valor estacionário.

Tendo em vista que considerar que os objetos e variáveis pertencem a um único grupo pode ser muito restritivo em aplicações práticas do mundo real, Laclau, Carvalho e Nadif (LACLAU; CARVALHO; NADIF, 2015) introduziram o conceito de partições fuzzy no critério $\mathcal{J}_{DK}$, assumindo que pode existir sobreposição entre os blocos de dados. Desse modo, eles consideraram uma partição fuzzy $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de N objetos em K grupos, onde u_{ik} representa o grau de pertencimento do objeto i no k-ésimo grupo de objetos, sujeito a $\sum_{k=1}^{K} u_{ik} = 1 \ \forall i$ e $u_{ik} \in [0, 1]$, e uma partição fuzzy $\mathbf{V} = [v_{jh}]_{\substack{1 \leq j \leq P \\ 1 \leq h \leq H}}$ de P variáveis em H grupos, onde v_{jh} representa o grau de pertencimento da variável j no h-ésimo grupo de variáveis, sujeito a $\sum_{h=1}^{H} v_{jh} = 1 \ \forall j$ e $v_{jh} \in [0, 1]$. O critério de adequação do algoritmo *FDK* é dado por:

$$\mathcal{J}_{FDK}(\mathbf{U}, \mathbf{V}, \mathbf{G}) = \sum_{k,h,i,j} (u_{ik})^m (v_{jh})^n \left(x_{ij} - g_{kh}\right)^2, \tag{3.2}$$

onde $m \in (1, \infty)$ é o parâmetro que controla a imprecisão de cada objeto i e $n \in (1, \infty)$ é o parâmetro que controla a imprecisão de cada variável j. Em outras palavras, esses parâmetros decidem se objetos e variáveis tem pertinência significativa em mais de um grupo.

O *FDK* define uma solução inicial das matrizes de pertinência $\mathbf{U}$ e $\mathbf{V}$, dos objetos e variáveis, respectivamente, sujeito às restrições impostas, e alterna entre três etapas: representação, alocação dos objetos e alocação das variáveis, até a convergência, que ocorre quando o critério $\mathcal{J}_{FDK}$ alcança um valor estacionário ou quando o algoritmo atinge um número máximo de iterações T.

Na etapa de representação as matrizes $\mathbf{U}$ e $\mathbf{V}$ são mantidas fixas. O objetivo é encontrar os protótipos g_{kh} ($k = 1, ..., K,\ h = 1, ..., H$) que minimizam o critério $\mathcal{J}_{FDK}$. Tomando a derivada de $\mathcal{J}_{FDK}$ em relação a g_{kh} e igualando a zero, temos:

$$\frac{\partial \mathcal{J}_{FDK}}{\partial g_{kh}} = \sum_{i,j} (u_{ik})^m (v_{jh})^n (x_{ij} - g_{kh}) = \sum_{i,j} (u_{ik})^m (v_{jh})^n x_{ij} - g_{kh} \sum_{i,j} (u_{ik})^m (v_{jh})^n = 0 \tag{3.3}$$

Resolvendo a Equação (3.3) obtém-se que a matriz de protótipos $\mathbf{G}$, que minimiza o critério $\mathcal{J}_{FDK}$, dado pela Equação (3.2), tem suas componentes g_{kh} atualizadas como:

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n} \quad \forall k, h \tag{3.4}$$

Nas etapas 2 e 3 o objetivo é encontrar os graus de pertinência dos objetos e variáveis que minimizam o critério $\mathcal{J}_{FDK}$. Para minimizar $\mathcal{J}_{FDK}$ definido na Equação (3.2) com respeito às condições impostas a u_{ik} e v_{jh}, usamos o método dos multiplicadores de Lagrange e obtemos a seguinte função lagrangiana:

$$\mathcal{L}_{FDK} = \sum_{k,h,i,j} (u_{ik})^m (v_{jh})^n (x_{ij} - g_{kh})^2 - \sum_i \theta_i \left(\sum_k (u_{ik}) - 1 \right) - \sum_j \eta_j \left(\sum_h (v_{jh}) - 1 \right), \tag{3.5}$$

onde θ_i e η_j são multiplicadores de Lagrange.

Na segunda etapa, de alocação de objetos, a matriz de protótipos $\mathbf{G}$ e a matriz de pertinência das variáveis $\mathbf{V}$ são mantidas fixas. O objetivo é encontrar os graus de pertinência dos objetos u_{ik} da matriz $\mathbf{U}$ que minimizam o critério $\mathcal{J}_{FDK}$. Tomando a derivada parcial de $\mathcal{L}_{FDK}$ em relação a u_{ik} e igualando a zero, após algumas manipulações matemáticas, chegamos a:

$$u_{ik} = \left[\sum_{r=1}^{K} \left(\frac{\sum_{j,h} (v_{jh})^n (x_{ij} - g_{kh})^2}{\sum_{j,h} (v_{jh})^n (x_{ij} - g_{rh})^2} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i, k. \tag{3.6}$$

Na última etapa ocorre a alocação das variáveis. A matriz de protótipos $\mathbf{G}$ e a matriz de pertinência dos objetos $\mathbf{U}$ são mantidas fixas. O objetivo é encontrar os graus de pertinência das variáveis v_{jh} da matriz $\mathbf{V}$ que minimizam o critério $\mathcal{J}_{FDK}$. Tomando a derivada parcial de $\mathcal{L}_{FDK}$ em relação a v_{jh} e igualando a zero, chegamos a:

$$v_{jh} = \left[\sum_{w=1}^{H} \left(\frac{\sum_{i,k} (u_{ik})^m (x_{ij} - g_{kh})^2}{\sum_{i,k} (u_{ik})^m (x_{ij} - g_{kw})^2} \right)^{\frac{1}{n-1}} \right]^{-1} \quad \forall j, h. \tag{3.7}$$

Esses três passos são repetidos iterativamente até a convergência, que ocorre quando o critério $\mathcal{J}_{FDK}$ alcança um valor estacionário ou atinge um número máximo de iterações T. Esses passos estão resumidos no Algoritmo 3.

Algoritmo 3 *Fuzzy Double K-Means* (*FDK*)

Entrada: O conjunto de dados $\mathbf{X}$, o número de grupos de objetos K, o número dos grupos de variáveis H, os parâmetros fuzzificadores m e n, o número máximo de iterações T e o limite de erro ε, com $0 < \varepsilon << 1$.

Saída: A matriz de protótipos $\mathbf{G}$, a matriz de pertinência fuzzy dos objetos $\mathbf{U}$, a matriz de pertinência das variáveis $\mathbf{V}$.

Inicialização:

$\mathbf{U}$: inicializar aleatoriamente os graus de pertencimentos dos objetos u_{ik} $\forall i, k$, sujeito a $u_{ik} \in [0.1]$ e $\sum_{k=1}^{K} u_{ik} = 1$;
$\mathbf{V}$: inicializar aleatoriamente os graus de pertencimentos das variáveis v_{jh} $\forall j, h$, sujeito a $v_{jh} \in [0.1]$ e $\sum_{h=1}^{H} v_{jh} = 1$.

$t \leftarrow 0$;

repita

$t \leftarrow t + 1$;

Etapa 1: Representação .

Calcule os componentes g_{kh} da matriz de protótipos $\mathbf{G}$ de acordo com a Equação 3.4.

Etapa 2: Alocação dos objetos

Atualize os graus de pertencimento u_{ik} da matriz de pertencimento dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de acordo com a Equação (3.6) .

Etapa 3: Alocação das variáveis

Atualize os graus de pertencimento v_{jh} da matriz de pertencimento das variáveis $\mathbf{V} = [v_{jh}]_{\substack{1 \leq j \leq P \\ 1 \leq h \leq H}}$ de acordo com a Equação (3.7)

até $|\mathcal{J}_{FDK}^{(t)} - \mathcal{J}_{FDK}^{(t-1)}| \leq \varepsilon$ ou $t > T$.

3.3 ALGORITMOS DE CO-CLUSTERING COM PONDERAÇÃO DAS VARIÁVEIS

Assim como no agrupamento convencional, o problema da maldição da dimensionalidade também afeta os algoritmos de co-clustering. Apesar desses modelos realizarem o agrupamento das variáveis, os grupos de objetos podem estar bem definidos em apenas algumas dimensões, ao invés de todo o conjunto de variáveis. Além disso, pode haver variáveis que não são informativas para os grupos de objetos. Nesse sentido, o desenvolvimento de modelos capazes de encontrar subespaços significativos para os grupos é um campo de pesquisa promissor, principalmente pelo sucesso desses modelos nos algoritmos de agrupamento convencional.

O desenvolvimento de algoritmos de co-clustering que são capazes de computar os pesos

das variáveis não é tão estudado como no agrupamento tradicional, embora alguns algoritmos tenham sido propostos com essa finalidade. Tjhi e Chen (TJHI; CHEN, 2006) propuseram o algoritmo Flexible Fuzzy Co-clustering with Feature-cluster Weighting (FFCFW), que distingue os pesos das variáveis para cada grupo de objetos e permite que o número de grupo de objetos seja diferente do número de grupos das variáveis. Laclau, Carvalho e Nadif (LACLAU; CARVALHO; NADIF, 2015) propuseram o algoritmo *Weighted Fuzzy Double K-Means* (*WFDK*) que computa o peso das variáveis, conseguindo realizar seleção de variáveis. O método *WFDK* será descrito em detalhes posteriormente. Chen *et al.* (CHEN et al., 2018) propuseram um método de co-clustering ponderado que computa os pesos das variáveis para os grupos de objetos e computa os pesos dos objetos para os grupos de variáveis. Foi utilizado um algoritmo iterativo para otimizar a função objetivo, que calculou automaticamente os dois tipos de pesos. Pham *et al.* (PHAM et al., 2021) propuseram um novo esquema de aprendizado, incorporando a entropia ponderada de variáveis na função objetivo do algoritmo, chamado Feature-Reduction Fuzzy Co-Clustering (FRFCoC). Recentemente De Carvalho *et al* (CARVALHO et al., 2021) propuseram algoritmos para dados simbólicos que distinguem os pesos das variáveis.

3.3.1 *Weighted Fuzzy Double K-Means (WFDK)*

O algoritmo *Weighted Fuzzy Double K-Means* (*WFDK*) (LACLAU; CARVALHO; NADIF, 2015) é uma extensão do algoritmo *FDK* descrito anteriormente. O *FDK* não leva em consideração os pesos de relevância das variáveis, i.e., considera que todas as variáveis são igualmente importantes para o processo de agrupamento, no sentido que todas elas têm o mesmo peso. Por outro lado, o *WFDK* computa de forma automática os pesos das variáveis, conseguindo detectar quais são irrelevantes, sendo pertinente no contexto de dados em alta dimensão. O *WFDK* minimiza o critério dado na Equação (3.8).

$$\mathcal{J}_{WFDK}(\mathbf{U},\mathbf{V},\mathbf{\Lambda},\mathbf{G}) = \sum_{k,h,i,j} (u_{ik})^m (v_{jh})^n (\lambda_j)^\gamma \left(x_{ij} - g_{kh}\right)^2, \tag{3.8}$$

onde $\mathbf{\Lambda} := [\lambda_j]_{1\leq j\leq P}$, sujeito a $\sum_{j=1}^{P} \lambda_j = 1$ e $\lambda_j \in [0,1]$, é o vetor de pesos de relevância, de modo que λ_j representa o peso da variável j, $\gamma \in (1,\infty)$ é o parâmetro que controla o grau de influência do peso de cada variável, de modo que se γ é grande o suficiente, as variáveis terão a mesma importância, enquanto que se $\gamma \to 1$ as variáveis terão pesos máximos e m e n são definidos como antes.

O *WFDK* define uma solução inicial das matrizes de pertinência $\mathbf{U}$ e $\mathbf{V}$ e do vetor de pesos $\mathbf{\Lambda}$, sujeito às restrições impostas, e alterna entre quatro etapas: representação, ponderação das variáveis, alocação dos objetos e alocação das variáveis, até a convergência, que ocorre quando o critério $\mathcal{J}_{WFDK}$ alcança um valor estacionário ou quando o algoritmo atinge um número máximo de iterações T.

Na etapa de representação as matrizes de pertinência $\mathbf{U}$ e $\mathbf{V}$ e os pesos $\mathbf{\Lambda}$ são mantidos fixos. O objetivo é encontrar os protótipos g_{kh} ($k = 1, ..., K$, $h = 1, ..., H$) que minimizam o critério $\mathcal{J}_{WFDK}$. Tomando a derivada de $\mathcal{J}_{WFDK}$ em relação a g_{kh} e igualando a zero, temos:

$$\begin{aligned}\frac{\partial \mathcal{J}_{WFDK}}{\partial g_{kh}} &= \sum_{i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma\,(x_{ij} - g_{kh}) \\ &= \sum_{i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma x_{ij} - g_{kh}\sum_{i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma = 0 \end{aligned} \tag{3.9}$$

Resolvendo a Equação (3.9) obtém-se que a matriz de protótipos $\mathbf{G}$, que minimiza o critério $\mathcal{J}_{WFDK}$, dado pela Equação (3.8), tem suas componentes g_{kh} atualizadas como:

$$g_{kh} = \frac{\sum_{i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma x_{ij}}{\sum_{i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma} \quad \forall k, h \tag{3.10}$$

A minimização de $\mathcal{J}_{WFDK}$ com respeito a $\mathbf{U}$, $\mathbf{V}$ e $\mathbf{\Lambda}$, sujeito as restrições impostas, é realizada via multiplicadores de Lagrange, de modo que a função lagrangiana é definida da seguinte forma

$$\begin{aligned}\mathcal{L}_{WFDK} = &\sum_{k,h,i,j}(u_{ik})^m(v_{jh})^n(\lambda_j)^\gamma\,(x_{ij} - g_{kh})^2 - \sum_i \theta_i\left(\sum_k (u_{ik}) - 1\right) \\ &- \sum_j \eta_j\left(\sum_h (v_{jh}) - 1\right) - \delta_j\left(\sum_j (\lambda_j - 1)\right),\end{aligned} \tag{3.11}$$

onde θ_i, η_j e δ_j são os multiplicadores de Lagrange.

Na segunda etapa, de ponderação das variáveis, a matriz de protótipos $\mathbf{G}$, a matriz de pertinência dos objetos $\mathbf{U}$ e a matriz de pertinência das variáveis $\mathbf{V}$ são mantidas fixas. O objetivo é encontrar os pesos de relevância das variáveis λ_j, sujeito a $\sum_{j=1}^{P}\lambda_j = 1$ e $\lambda_j \in [0, 1]$, que minimizam o critério $\mathcal{J}_{WFDK}$. Tomando a derivada parcial de $\mathcal{L}_{WFDK}$ em relação a λ_j e igualando a zero, após algumas manipulações matemáticas, chegamos a:

$$\lambda_j = \left[\sum_{l=1}^{P}\left(\frac{\sum_{k,h,i}(u_{ik})^m(v_{jh})^n(x_{ij} - g_{kh})^2}{\sum_{k,h,i}(u_{ik})^m(v_{lh})^n(x_{il} - g_{kh})^2}\right)^{\frac{1}{\gamma-1}}\right]^{-1} \quad \forall i, k. \tag{3.12}$$

Na terceira etapa, de alocação dos objetos, a matriz de protótipos $\mathbf{G}$, o vetor de pesos das variáveis $\mathbf{\Lambda}$ e a matriz de pertinência das variáveis $\mathbf{V}$ são mantidos fixos. O objetivo

é encontrar os graus de pertinência dos objetos u_{ik} da matriz $\mathbf{U}$ que minimizam o critério $\mathcal{J}_{WFDK}$. Tomando a derivada parcial de $\mathcal{L}_{WFDK}$ em relação a u_{ik} e igualando a zero, após algumas manipulações matemáticas, chegamos a:

$$u_{ik} = \left[\sum_{r=1}^{K} \left(\frac{\sum_{j,h} (v_{jh})^n (\lambda_j)^\gamma (x_{ij} - g_{kh})^2}{\sum_{j,h} (v_{jh})^n (\lambda_j)^\gamma (x_{ij} - g_{rh})^2} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i, k. \tag{3.13}$$

Na quarta e última etapa ocorre a alocação das variáveis. A matriz de protótipos $\mathbf{G}$, o vetor de pesos das variáveis $\mathbf{\Lambda}$ e a matriz de pertinência dos objetos $\mathbf{U}$ são mantidos fixos. O objetivo é encontrar os graus de pertinência das variáveis v_{jh} da matriz $\mathbf{V}$ que minimizam o critério $\mathcal{J}_{WFDK}$. Tomando a derivada parcial de $\mathcal{L}_{WFDK}$ em relação a v_{jh} e igualando a zero, chegamos a:

$$v_{jh} = \left[\sum_{w=1}^{H} \left(\frac{\sum_{i,k} (u_{ik})^m (x_{ij} - g_{kh})^2}{\sum_{i,k} (u_{ik})^m (x_{ij} - g_{kw})^2} \right)^{\frac{1}{n-1}} \right]^{-1} \quad \forall j, h. \tag{3.14}$$

Esses quatro passos são repetidos iterativamente até a convergência, que ocorre quando o critério $\mathcal{J}_{WFDK}$ alcança um valor estacionário ou atinge um número máximo de iterações T.

O algoritmo 4 resume o passo a passo para a execução do *Weighted Fuzzy Double K-Means* (*WFDK*).

Algoritmo 4 *Weighted Fuzzy Double K-Means* (*WFDK*)

Entrada: O conjunto de dados $\mathbf{X}$, o número de grupos de objetos K, o número dos grupos de variáveis H, os parâmetros m, n e γ, o número máximo de iterações T e o limite de erro ε, com $0 < \varepsilon << 1$.

Saída: A matriz de protótipos $\mathbf{G}$, a matriz de pertinência fuzzy dos objetos $\mathbf{U}$, a matriz de pertinência das variáveis $\mathbf{V}$ e os pesos de relevância das variáveis $\mathbf{\Lambda}$.

Inicialização:

$\mathbf{U}$: inicializar aleatoriamente os graus de pertencimentos dos objetos u_{ik} $\forall i, k$, sujeito a $u_{ik} \in [0.1]$ e $\sum_{k=1}^{K} u_{ik} = 1$;
$\mathbf{V}$: inicializar aleatoriamente os graus de pertencimentos das variáveis v_{jh} $\forall j, h$, sujeito a $v_{jh} \in [0.1]$ e $\sum_{h=1}^{H} v_{jh} = 1$;
$\mathbf{\Lambda}$: inicializar os pesos das variáveis como $1/P$ ou aleatoriamente, desde que as condições impostas sejam satisfeitas.

$t \leftarrow 0$;

repita

$t \leftarrow t + 1$;

Etapa 1: Representação .

Calcule o componente g_{kh} da matriz de protótipos $\mathbf{G}$ de acordo com a Equação (3.10)

Etapa 2: Ponderação.

Atualize os pesos das variáveis $\mathbf{\Lambda}$ de acordo com a Equação (3.12).

Etapa 3: Alocação dos objetos

Atualize os graus de pertencimento u_{ik} da matriz de pertencimento dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de acordo com a Equação (3.13).

Etapa 4: Alocação das variáveis

Atualize os graus de pertencimento v_{jh} da matriz de pertencimento das variáveis $\mathbf{V} = [v_{jh}]_{\substack{1 \leq j \leq P \\ 1 \leq h \leq H}}$ de acordo com a Equação (3.14).

até $|\mathcal{J}^{(t)} - \mathcal{J}^{(t-1)}| \leq \varepsilon$ ou $t > T$.

3.4 SÍNTESE DO CAPÍTULO

Esse capítulo apresenta uma revisão acerca dos algoritmos de co-clustering. Foram abordadas as principais categorias desses algoritmos. Apresentamos uma breve revisão acerca dos métodos de co-clustering que realizam ponderação das variáveis. Os algoritmos *Fuzzy Double K-Means* (*FDK*) e *Weighted Fuzzy Double K-Means* (*WFDK*), por estarem ligados com o tema central do trabalho, foram apresentados em detalhes.

4 ALGORITMOS DE CO-CLUSTERING FUZZY BASEADOS EM KERNEL COM DISTÂNCIAS ADAPTATIVAS

Os modelos clássicos de agrupamento baseados em kernel apresentaram bons desempenhos em comparação com suas versões não-kernelizadas. Os resultados obtidos mostraram que tais algoritmos conseguem lidar com estrutura de dados complexa, i.e., grupos não-linearmente separáveis e/ou não-hiperesféricos, sendo capazes de encontrar grupos representativos dos dados nesse cenário. Embora seja muito empregada com os algoritmos de agrupamento tradicional, as funções kernel não têm sido combinadas com os métodos de co-clustering.

Também vale destacar que os algoritmos de co-clustering do estado da arte, como o *Double K-Means* e *Fuzzy Double K-Means*, consideram que todas as variáveis têm a mesma importância para a construção dos grupos de objetos que compõem os blocos de dados, no sentido de todas elas possuem o mesmo peso. Não obstante, esses algoritmos têm sido aplicados, em sua grande maioria, a conjuntos de dados de alta dimensão. Nesse sentido, algumas variáveis podem apresentar pesos pequenos, ou até mesmo desprezíveis. Dentre as variáveis relevantes, algumas podem ter pesos maiores que as demais. Além disso, pode existir variáveis com o mesmo peso para todos os grupos de objetos, como também pode haver variáveis com pesos diferentes para cada grupo. Seguindo esse raciocínio, ao computar os pesos de relevância das variáveis, a performance final dos algoritmos pode ser melhorada.

Este capítulo trata da principal contribuição deste trabalho. Aqui, nós introduzimos os algoritmos de co-clustering fuzzy baseados em kernel com ponderação automática das variáveis, considerando apenas a abordagem de agrupamento baseada em kernel no espaço de características. Tal ponderação foi feita através de distâncias adaptativas, calculadas usando funções kernel. A seguir, nós definimos as medidas de distâncias baseadas no kernel gaussiano, sendo duas delas combinadas com distâncias adaptativas, que nos permite computar os pesos de relevância das variáveis automaticamente, e por fim, nós introduzimos os modelos de co-clustering fuzzy baseados em kernel com ponderação automática das variáveis.

4.1 MEDIDAS DE DISTÂNCIA BASEADAS EM KERNEL

Uma das componentes mais importantes em análise de agrupamento é a medida de similaridade (ou dissimilaridade). Elas são responsáveis por mensurar o quão próximo os dados estão/são, e consequentemente são a base para a formação dos grupos ou blocos em um

conjunto de dados. Um dos exemplos básicos de medidas de dissimilaridade são as medidas de distâncias. Diversas distâncias foram usadas em algoritmos de agrupamento, como, por exemplo: distância Euclidiana (a mais comumente utilizada) (STEINHAUS, 1956; BEZDEK, 1981), city-block (JAJUGA, 1991; RODRÍGUEZ; CARVALHO, 2019b), distância de Mahalanobis (GUSTAFSON; KESSEL, 1979), dentre outras. Essas métricas têm influência considerável em algumas características dos algoritmos, como convergência e derivação das equações, bem como podem influenciar no desempenho final destes métodos (BORA et al., 2014). Cada medida de distância é utilizada com uma finalidade, seja ela facilidade nas obtenções das equações, robustez a outliers ou até mesmo com intuito de capturar a correlação entre os dados, que é o caso da distância de Mahalanobis.

Se tratando de funções kernel, o ponto chave é a tentativa de obter grupos ou blocos bem definidos no espaço do kernel, através de um mapeamento não-linear. Outro ponto importante, é que os algoritmos de agrupamento baseados em kernel são mais robustos a ruídos e outliers (GONZÁLEZ et al., 2012). Normalmente para encontrar grupos, o *kernel trick* é empregado aproveitando que o produto interno no espaço do kernel pode ser representado por um kernel de Mercer (Eq. 2.8). Portanto, a distância no espaço do kernel pode ser substituída pelo kernel Mercer, de acordo com a Eq. (2.9). Levando em consideração o kernel gaussiano, i.e. $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$, $\sigma > 0$, temos que $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) = 1 \ \forall i$, o que leva a:

$$||\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)||^2 = \underbrace{\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i)}_{=1} - 2\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) + \underbrace{\mathcal{K}(\mathbf{x}_j, \mathbf{x}_j)}_{=1} = 2 - 2\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \tag{4.1}$$

Em relação à ponderação automática das variáveis, surge a ideia de distâncias adaptativas, que permite computar o peso de cada variável, como um passo adicional nos algoritmos de agrupamento. Distâncias adaptativas são, na verdade, distâncias ponderadas que associam pesos positivos às variáveis de acordo com alguma restrição adequada. Na literatura existem dois tipos de restrições que são comumente usadas nos métodos de agrupamento com distâncias adaptativas: a restrição de que a soma dos pesos das variáveis deve ser igual a um (HUANG et al., 2005; FERREIRA; CARVALHO, 2014) e a restrição de que o produto dos pesos das variáveis deve ser igual a um (CARVALHO et al., 2021; FERREIRA; CARVALHO, 2014). Embora seja mais intuitivo pensar na ponderação com a restrição da soma, na prática, os métodos que se baseiam nessa restrição são mais custosos computacionalmente, pois, por questões matemáticas, no processo de derivação aparece um novo hiperparâmetro, que precisa de um passo de pré-processamento para ajustá-lo, haja vista que este exerce influência considerável no desempenho final dos algoritmos (FERREIRA; CARVALHO; SIMÕES, 2016). Além disso,

esse hiperparâmetro não tem uma interpretação real. Por outro lado, quando a restrição do produto é utilizada, nenhum hiperparâmetro aparece, e consequentemente não é necessário nenhum passo adicional. Outra característica importante dessa restrição, é que ela tende a atenuar os pesos menos expressivos e destacar aqueles mais expressivos. Por esses motivos, nesse trabalho, apenas a restrição do produto é considerada.

Existem dois tipos principais de distâncias adaptativas, relacionados com a forma como os pesos são atribuídos às variáveis:

- *Distância adaptativa global*: quando os pesos são atribuídos a cada variável para o conjunto de dados inteiro, i.e., os pesos das variáveis são os mesmos para todos os grupos de objetos;
- *Distância adaptativa local*: quando os pesos são atribuídos a cada variável para cada grupo de objetos, individualmente, i.e., os pesos das variáveis diferem de um grupo para o outro.

Um aspecto importante em relação a essas duas abordagens, é que, embora possam existir variáveis que sejam relevantes para grupos diferentes, a distância adaptativa local pode levar os algoritmos a mínimos locais, produzindo soluções sub-ótimas, não sendo apropriada em algumas situações, como, por exemplo, quando as dispersões internas dos grupos de objetos são quase as mesmas (RODRÍGUEZ; CARVALHO, 2019a).

Seja $d^2(x_{ij}, g_{kh})$ uma medida de distância baseada em kernel entre um ponto genérico x_{ij} da matriz de dados e o protótipo de um bloco g_{kh}. Diferentes medidas de distância d^2 produzem diferentes algoritmos de co-clustering. Nesse trabalho, nós definimos três medidas de distâncias baseadas em kernel, onde duas delas combinam as utilidades de funções kernel (nos baseamos no kernel gaussiano) e distâncias adaptativas. As distâncias propostas nesta dissertação são:

1. Distância não-adaptativa kernelizada:

$$d_N^2(x_{ij}, g_{kh}) = 2 - 2\mathcal{K}(x_{ij}, g_{kh}), \tag{4.2}$$

2. Distância adaptativa global kernelizada:

$$d_G^2(x_{ij}, g_{kh}|\mathbf{\Lambda}) = \lambda_j(2 - 2\mathcal{K}(x_{ij}, g_{kh})), \tag{4.3}$$

onde $\mathbf{\Lambda} = (\lambda_j) \in (\mathbb{R}^+)^P$ é o vetor de pesos que pondera a distância entre os pontos e protótipos dos blocos para todos os grupos de objetos, sujeito à $\prod_{j=1}^{P} \lambda_j = 1$ e $\lambda_j > 0$.

3. Distância adaptativa local kernelizada:

$$d_L^2(x_{ij}, g_{kh}|\mathbf{\Lambda}) = \lambda_{kj}(2 - 2\mathcal{K}(x_{ij}, g_{kh})), \tag{4.4}$$

onde $\mathbf{\Lambda} = (\lambda_{kj}) \in (\mathbb{R}^+)^{(K\times P)}$ é uma matriz de pesos que pondera a distância entre os pontos e protótipos dos blocos individualmente para cada grupo de objetos, sujeito à $\prod_{j=1}^{P} \lambda_{kj} = 1$ e $\lambda_{kj} > 0, \forall k$.

Note que, no caso da distância adaptativa global (Eq. (4.3)), assume-se que o conjunto de variáveis importantes é o mesmo para todos os grupos de objetos, enquanto que, no caso da distância adaptativa local (Eq. 4.4)), supõe-se que o conjunto de variáveis importantes pode não ser o mesmo para todos os grupos de objetos.

4.2 MODELOS DE CO-CLUSTERING FUZZY BASEADOS EM KERNEL COM PONDERAÇÃO AUTOMÁTICA DAS VARIÁVEIS

Nesta seção, nós apresentamos os algoritmos de co-clustering fuzzy baseados em kernel com distâncias adaptativas, que permitem computar os pesos das variáveis de forma automática. Os modelos propostos medem a heterogeneidade de uma partição fuzzy de objetos e uma partição fuzzy de variáveis, simultaneamente, como a soma da heterogeneidade em cada bloco de dados, formados por um conjunto de objetos e um conjunto de variáveis correlacionados. De forma geral, esses algoritmos minimizam a função objetivo definida na Equação (4.5), sujeito a $u_{ik} \in [0, 1]$, $\sum_{k=1}^{K} u_{ik} = 1$, $v_{jh} \in [0, 1]$ e $\sum_{h=1}^{H} v_{jh} = 1$.

$$\mathcal{J} = \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m \; v_{\bullet h}^n} \, (u_{ik})^m \, (v_{jh})^n \, d^2(x_{ij}, g_{kh}), \tag{4.5}$$

onde $u_{\bullet k}^m = \sum_i (u_{ik})^m$, $v_{\bullet h}^n = \sum_j (v_{jk})^n$ e $d^2(x_{ij}, g_{kh})$ é uma medida de distância baseada em kernel, definida na Seção 4.1. Os termos $u_{\bullet k}^m$ e $v_{\bullet h}^n$ estão relacionados com a quantidade de pertencimento fuzzy dos objetos no grupo k e quantidade de pertencimento fuzzy das variáveis no grupo h, respectivamente. Na abordagem hard, esses termos seriam relacionados com número de objetos no grupo k e o número de variáveis no grupo h.

Em uma tentativa inicial, nós desenvolvemos os modelos sem considerar os termos $u_{\bullet k}^m$ e $v_{\bullet h}^n$ na função objetivo $\mathcal{J}$. Após realizar alguns experimentos, notamos que a proximidade entre os protótipos dos blocos g_{kh} era facilmente afetada pelos valores de m e n. Em decorrência desse problema, em alguns casos, a grande maioria dos objetos ou das variáveis era alocada

em um único grupo. Para tornar os métodos mais robustos e, consequentemente, mitigar esses problemas indesejáveis, introduzimos os referidos termos no critério $\mathcal{J}$, como feito de maneira similar na referência (LACLAU; NADIF, 2016).

De acordo com a medida de distância utilizada temos diferentes algoritmos de co-clustering fuzzy baseados em kernel, na qual otimizam diferentes critérios de adequação.

a) A primeira proposta é baseada na distância não-adaptativa kernelizada, dada pela Equação 4.2. Esse método considera que todas as variáveis são igualmente importantes para o processo de otimização e leva em consideração apenas as características de funções kernel. Nesse caso, o algoritmo é nomeado como *Gaussian Kernel Fuzzy Double K-Means* (*GKFDK*) e minimiza a função objetivo definida na Equação 4.6.

$$\begin{aligned}\mathcal{J}_N(\mathbf{G},\mathbf{U},\mathbf{V}) &= \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n d_N^2(x_{ij}, g_{kh}) \\ &= 2\sum_{k,h} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} \sum_{i,j} (u_{ik})^m (v_{jh})^n (1 - \mathcal{K}(x_{ij}, g_{kh})) \end{aligned} \tag{4.6}$$

b) Considerando a distância adaptativa global kernelizada (Equação (4.3)), o algoritmo proposto *Gaussian Kernel Fuzzy Double K-Means Based on Global Adaptive Distance* (*GKFDK-GP*) assume que as variáveis têm pesos de relevância diferentes, mas são os mesmos para todos os grupos de objetos. O algoritmo *GKFDK-GP* minimiza a Equação (4.7) sujeito a $\lambda_j > 0$ e $\prod_{j=1}^{P} \lambda_j = 1$.

$$\begin{aligned}\mathcal{J}_G(\mathbf{G},\mathbf{\Lambda},\mathbf{U},\mathbf{V}) &= \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n d_G^2(x_{ij}, g_{kh}) \\ &= 2\sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n \lambda_j (1 - \mathcal{K}(x_{ij}, g_{kh})) \end{aligned} \tag{4.7}$$

c) Quando a distância adaptativa local kernelizada é utilizada, o conjunto de variáveis relevantes é diferente para cada grupo de objetos, no sentido de que existem variáveis com pesos diferentes em cada um deles. Nesse caso, o algoritmo proposto *Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance* (*GKFDK-LP*) minimiza a Equação(4.8) sujeito a $\lambda_{kj} > 0$ e $\prod_{j=1}^{P} \lambda_{kj} = 1\ \forall k = 1, ..., K$.

$$\begin{aligned}\mathcal{J}_L(\mathbf{G},\mathbf{\Lambda},\mathbf{U},\mathbf{V}) &= \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n d_L^2(x_{ij}, g_{kh}) \\ &= 2\sum_{k,h,i,j} \frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n \lambda_{kj} (1 - \mathcal{K}(x_{ij}, g_{kh})) \end{aligned} \tag{4.8}$$

4.2.1 Passos de otimização

Essa subseção apresenta os passos de otimização dos algoritmos propostos, visando calcular os protótipos dos blocos, os pesos de relevância das variáveis, a partição fuzzy dos objetos e a partição fuzzy das variáveis.

Os algoritmos se iniciam a partir de uma solução inicial das partições $\mathbf{U}$ e $\mathbf{V}$, da matriz de protótipos $\mathbf{G}$ e dos pesos $\mathbf{\Lambda}$, no caso de distâncias adaptativas, e alterna entre três ou quatro etapas, conforme a medida de distância utilizada, até a convergência do algoritmo, que ocorre quando a função objetivo alcança um valor estacionário representando um mínimo local ou quando o algoritmo atinge um número máximo de iterações. As etapas de representação, atribuição dos objetos e atribuição das variáveis são comuns a todos os algoritmos. Para os algoritmos baseados em distâncias adaptativas (*GKFDK-GP* e *GKFDK-LP*) existe uma etapa adicional de ponderação. Nesse sentido, a função objetivo é minimizada iterativamente ao longo desses passos.

4.2.1.1 Etapa de representação

Durante a etapa de representação, a matriz fuzzy de pertencimento dos objetos $\mathbf{U}$, a matriz fuzzy de pertencimento das variáveis $\mathbf{V}$, e os pesos das variáveis $\mathbf{\Lambda}$ são mantidos fixos. Essa etapa fornece a solução ótima para a matriz de protótipos $\mathbf{G}$. Dessa forma, as funções objetivos dos algoritmos são minimizadas em relação aos protótipos.

Proposition 4.1. *Os protótipos dos blocos que minimizam as funções objetivo propostas são calculados segundo a medida de distância utilizada.*

(I) Se o algoritmo é baseado na distância não-adaptativa kernelizada (Equação (4.2)), o protótipo g_{kh} do bloco $\mathbf{X}_{kh}$ que minimiza o critério $\mathcal{J}_N$ (Equação 4.6), é atualizado de acordo com a seguinte equação:

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh})} \qquad \forall k, h \tag{4.9}$$

(II) Se o algoritmo é baseado na distância adaptativa global kernelizada (Equação (4.3)), o protótipo g_{kh} do bloco $\mathbf{X}_{kh}$ que minimiza o critério $\mathcal{J}_G$ (Equação (4.7)), é atualizado de acordo

com a seguinte equação :

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j \mathcal{K}(x_{ij}, g_{kh})} \qquad \forall k, h \tag{4.10}$$

(III) Se o algoritmo é baseado na distância adaptativa local kernelizada (Equação (4.4)), o protótipo g_{kh} do bloco $\mathbf{X}_{kh}$ que minimiza o critério $\mathcal{J}_L$ (Equação (4.8)), é atualizado de acordo com a seguinte equação :

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_{kj} \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_{kj} \mathcal{K}(x_{ij}, g_{kh})} \qquad \forall k, h \tag{4.11}$$

Demonstração. (I) Queremos encontrar o protótipo g_{kh} que minimiza o critério $\mathcal{J}_N$, dado na Equação (4.6). Levando em consideração o kernel gaussiano, temos que $0 \leq \mathcal{K}(x_{ij}, g_{kh}) \leq 1$. Nesse sentido, minimizar $\mathcal{J}_G$ com relação a g_{kh} equivale a maximizar o seguinte termo, $\forall k, h$:

$$\mathcal{T}(x_{ij}, g_{kh}) = \sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh})$$

Derivando $\mathcal{T}(x_{ij}, g_{kh})$ e igualando a zero, temos para $k = 1, ..., K$ e $h = 1, ..., H$:

$$\frac{\partial \mathcal{T}}{\partial g_{kh}} = \sum_{i,j} (u_{ik})^m (v_{jh})^n \frac{\partial \left(e^{\left(-(x_{ij} - g_{kh})^2 / 2\sigma^2\right)} \right)}{\partial g_{kh}} = 0$$

$$-\sum_{i,j} (u_{ik})^m (v_{jh})^n e^{\left(-(x_{ij} - g_{kh})^2 / 2\sigma^2\right)} \frac{(x_{ij} - g_{kh})}{\sigma^2} = 0$$

$$\sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh}) (x_{ij} - g_{kh}) = 0$$

$$g_{kh} \sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh}) = \sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh}) x_{ij}$$

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \mathcal{K}(x_{ij}, g_{kh})}$$

(II) Definindo $\mathcal{T}(x_{ij}, g_{kh}) = \sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j \mathcal{K}(x_{ij}, g_{kh})$ e seguindo os mesmos passos da parte (I) chegamos a:

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j \mathcal{K}(x_{ij}, g_{kh})}$$

(III) Definindo $\mathcal{T}(x_{ij}, g_{kh}) = \sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_{kj} \mathcal{K}(x_{ij}, g_{kh})$ e seguindo os mesmos passos da parte (I) chegamos a:

$$g_{kh} = \frac{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_{kj} \mathcal{K}(x_{ij}, g_{kh}) x_{ij}}{\sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_{kj} \mathcal{K}(x_{ij}, g_{kh})}$$

□

4.2.1.2 Etapa de ponderação

Esse passo fornece a solução ótima para computar os pesos de relevância das variáveis para os algoritmos *GKFDK-GP* e *GKFDK-LP*. Durante o passo de ponderação, a matriz de protótipos $\mathbf{G}$, a partição fuzzy dos objetos $\mathbf{U}$ e a partição fuzzy das variáveis $\mathbf{V}$ são mantidas fixas.

Proposition 4.2. *Os pesos das variáveis que minimizam as funções objetivo propostas, são calculados de acordo com a medida de distância adaptativa utilizada.*

(I) Se a distância adaptativa global, dada pela Eq. (4.3), é utilizada, o vetor de pesos $\mathbf{\Lambda} := [\lambda_j]_{1 \leq j \leq P}$ *que minimiza o critério* $\mathcal{J}_G$ *(Equação (4.7)), sujeito a* $\lambda_j > 0$ *e* $\prod_{j=1}^{P} \lambda_j = 1$*, tem seus componentes* λ_j *computados da seguinte forma:*

$$\lambda_j = \frac{\left[\prod_{r=1}^{P} \left(\sum_{k,h,i} \frac{1}{u_{\bullet k}^m \; v_{\bullet h}^n} (u_{ik})^m (v_{rh})^n (1 - \mathcal{K}(x_{ir}, g_{kh}))\right)\right]^{\frac{1}{P}}}{\sum_{k,h,i} \frac{1}{u_{\bullet k}^m \; v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n (1 - \mathcal{K}(x_{ij}, g_{kh}))} \qquad \forall j \qquad (4.12)$$

(II) Se a distância adaptativa local, dada pela Eq. (4.4), é utilizada, a matriz de pesos $\mathbf{\Lambda} := [\lambda_{kj}]_{\substack{1 \leq k \leq K \\ 1 \leq j \leq P}}$ *que minimiza o critério* $\mathcal{J}_L$ *(Equação (4.8)), sujeito a* $\lambda_{kj} > 0$ *e* $\prod_{j=1}^{P} \lambda_{kj} = 1, \; \forall k$*, tem suas componentes* λ_{kj} *computadas da seguinte forma:*

$$\lambda_{kj} = \frac{\left[\prod_{r=1}^{P} \left(\sum_{h,i} \frac{1}{v_{\bullet h}^n} (u_{ik})^m (v_{rh})^n (1 - \mathcal{K}(x_{ir}, g_{kh}))\right)\right]^{\frac{1}{P}}}{\sum_{h,i} \frac{1}{v_{\bullet h}^n} (u_{ik})^m (v_{jh})^n (1 - \mathcal{K}(x_{ij}, g_{kh}))} \qquad \forall k, j \qquad (4.13)$$

Demonstração. (I) Queremos minimizar $\mathcal{J}_G = 2 \sum_{k,h} \frac{1}{u_{\bullet k}^m \; v_{\bullet h}^n} \sum_{i,j} (u_{ik})^m (v_{jh})^n \lambda_j (1 - \mathcal{K}(x_{ij}, g_{kh}))$ em relação a λ_j ($j = 1, ..., p$) sob as restrições $\lambda_j > 0 \; \forall j$ e $\prod_{j=1}^{p} \lambda_j = 1$. Nós usamos

multiplicadores de Lagrange para resolver esse problema de minimização. Seja $\mathcal{L}_1$ a função Lagrangiana.

$$\begin{aligned}\mathcal{L}_1 &= 2\sum_{k,h,i,j}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}\left(u_{ik}\right)^m\left(v_{jh}\right)^n\lambda_j(1-\mathcal{K}(x_{ij},g_{kh}))-\theta\left[\prod_{j=1}^{P}\lambda_j-1\right]\\ &= 2\sum_{k,h,i,j}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}\left(u_{ik}\right)^m\left(v_{jh}\right)^n\lambda_j(1-\mathcal{K}(x_{ij},g_{kh}))-\theta\left[\sum_{j=1}^{P}\ln\left(\lambda_j\right)\right]\end{aligned}$$

Tomando a derivada parcial de $\mathcal{L}_1$ em relação a λ_j e igualando o gradiente a zero, temos

$$\frac{\partial\mathcal{L}_1}{\partial\lambda_j}=2\underbrace{\sum_{k,h,i}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}\left(u_{ik}\right)^m\left(v_{jh}\right)^n\left(1-\mathcal{K}(x_{ij},g_{kh})\right)}_{D_j}-\frac{\theta}{\lambda_j}=0\Longrightarrow\lambda_j=\frac{\theta}{D_j}\tag{4.14}$$

Sabendo que $\prod_{r=1}^{P}\lambda_r=1$, temos:

$$\prod_{r=1}^{P}\lambda_r=1\Longrightarrow\prod_{r=1}^{P}\frac{\theta}{D_r}=1\Longrightarrow\theta^P\prod_{r=1}^{P}\frac{1}{D_r}=1\Longrightarrow\theta=\left[\prod_{r=1}^{P}D_r\right]^{\frac{1}{P}}\tag{4.15}$$

Substituindo a Equação (4.15) na Equação (4.14) obtemos

$$\lambda_j=\frac{\left[\prod_{r=1}^{P}D_r\right]^{\frac{1}{P}}}{D_j}=\frac{\left[\prod_{r=1}^{P}\left(\sum_{k,h,i}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}(u_{ik})^m(v_{rh})^n(1-\mathcal{K}(x_{ir},g_{kh}))\right)\right]^{\frac{1}{P}}}{\sum_{k,h,i}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}(u_{ik})^m(v_{jh})^n(1-\mathcal{K}(x_{ij},g_{kh}))}$$

Podemos reescrever $\mathcal{J}_G$ como $\mathcal{J}(\lambda_1,...,\lambda_P)=2\sum_j\lambda_jD_j$. Note que $\mathcal{J}(\lambda_1,...,\lambda_P)=\sum_j\lambda_jD_j=\sum_{j=1}^{P}\frac{\left(\prod_{r=1}^{P}D_r\right)^{\frac{1}{P}}}{D_j}D_j=P\left\{D_1\times...\times D_P\right\}^{\frac{1}{P}}$ é um extremo. Como $\mathcal{J}(1,...,1)=\sum_j D_j=D_1+...+D_P$, e sabendo que a média geométrica é sempre menor que a média aritmética, isto é, $\left\{D_1\times...\times D_P\right\}^{\frac{1}{P}}<\frac{1}{P}\left\{D_1+...+D_P\right\}$ (a igualdade vale se $D_1=...=D_P$), podemos concluir que esse extremo é um ponto de mínimo.

(II) Definindo a função lagrangiana como

$$\begin{aligned}\mathcal{L}_2 &= 2\sum_{k,h,i,j}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}\left(u_{ik}\right)^m\left(v_{jh}\right)^n\lambda_j(1-\mathcal{K}(x_{ij},g_{kh}))-\sum_k\theta_k\left[\prod_{j=1}^{P}\lambda_{kj}-1\right]\\ &= 2\sum_{k,h,i,j}\frac{1}{u_{\bullet k}^m\ v_{\bullet h}^n}\left(u_{ik}\right)^m\left(v_{jh}\right)^n\lambda_j(1-\mathcal{K}(x_{ij},g_{kh}))-\sum_k\theta_k\left[\sum_{j=1}^{P}\ln\left(\lambda_{kj}\right)\right]\end{aligned}$$

e seguindo um raciocínio similar ao da parte (I), concluímos que

$$\lambda_{kj}=\frac{\left[\prod_{r=1}^{P}\left(\sum_{h,i}\frac{1}{v_{\bullet h}^n}(u_{ik})^m(v_{rh})^n(1-\mathcal{K}(x_{ir},g_{kh}))\right)\right]^{\frac{1}{P}}}{\sum_{h,i}\frac{1}{v_{\bullet h}^n}(u_{ik})^m(v_{jh})^n(1-\mathcal{K}(x_{ij},g_{kh}))}$$

□

4.2.1.3 *Etapa de alocação dos objetos*

A etapa de alocação dos objetos fornece a solução para calcular a matriz $\mathbf{U}$ de pertencimento fuzzy dos objetos em K grupos de objetos. Nessa etapa, a matriz de protótipos $\mathbf{G}$, os pesos de relevância $\mathbf{\Lambda}$ (para os algoritmos *GKFDK-GP* e *GKFDK-LP*) e a matriz $\mathbf{V}$ de pertencimento das variáveis são mantidos fixos.

Queremos minimizar $\mathcal{J}$ (Equação (4.5)) em relação a u_{ik} sob $u_{ik} \in [0,1]$ e $\sum_{k=1}^{K} u_{ik} = 1$. Seja $\mathcal{L}_3$ a função Lagrangiana.

$$\mathcal{L}_3 = \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^{m}\, v_{\bullet h}^{n}} \left(u_{ik}\right)^{m} \left(v_{jh}\right)^{n} d^2(x_{ij}, g_{kh}) - \sum_{i} \theta_i \left(\sum_{k} u_{ik} - 1 \right),$$

onde $d^2(x_{ij}, g_{kh})$ é uma das medidas de distâncias definidas nas Equações (4.2), (4.3) ou (4.4).

Tomando a derivada parcial de $\mathcal{L}_3$ com respeito a u_{ik} e igualando o gradiente a zero, temos:

$$\frac{\partial \mathcal{L}_3}{\partial u_{ik}} = \underbrace{\left(\sum_{h,j} \frac{1}{u_{\bullet k}^{m}\, v_{\bullet h}^{n}} (v_{jh})^{n} d^2(x_{ij}, g_{kh}) \right)}_{D_{ik}} \left(m \times \left(u_{ik}\right)^{m-1} \right) - \theta_i = 0 \tag{4.16}$$

A partir da Equação (4.16) obtemos

$$u_{ik} = \left(\frac{\theta_i}{m} \right)^{\frac{1}{m-1}} \left(\frac{1}{D_{ik}} \right)^{\frac{1}{m-1}} \tag{4.17}$$

Como $\sum_{r=1}^{K} u_{ir} = 1$, temos

$$\begin{aligned} \sum_{r=1}^{K} \left(\frac{\theta_i}{m} \right)^{\frac{1}{m-1}} \left(\frac{1}{D_{ir}} \right)^{\frac{1}{m-1}} &= 1 \\ \left(\frac{\theta_i}{m} \right)^{\frac{1}{m-1}} \sum_{r=1}^{K} \left(\frac{1}{D_{ir}} \right)^{\frac{1}{m-1}} &= 1 \\ \left(\frac{\theta_i}{m} \right)^{\frac{1}{m-1}} &= \left[\sum_{r=1}^{K} \left(\frac{1}{D_{ir}} \right)^{\frac{1}{m-1}} \right]^{-1} \end{aligned} \tag{4.18}$$

Substituindo a Equação (4.18) na Equação (4.17) chegamos a

$$
\begin{aligned}
u_{ik} &= \frac{\left(1/D_{ik}\right)^{\frac{1}{m-1}}}{\sum_{r=1}^{K}\left(1/D_{ir}\right)^{\frac{1}{m-1}}} \\
u_{ik} &= \left[\sum_{r=1}^{K}\left(\frac{D_{ik}}{D_{ir}}\right)^{\frac{1}{m-1}}\right]^{-1} \\
u_{ik} &= \left[\sum_{r=1}^{K}\left(\frac{\sum_{h,j}\frac{1}{u_{\bullet k}^{m}\ v_{\bullet h}^{n}}(v_{jh})^{n}d^{2}(x_{ij},g_{kh})}{\sum_{h,j}\frac{1}{u_{\bullet r}^{m}\ v_{\bullet h}^{n}}(v_{jh})^{n}d^{2}(x_{ij},g_{rh})}\right)^{\frac{1}{m-1}}\right]^{-1}
\end{aligned}
\tag{4.19}
$$

Sabemos que $\frac{\partial \mathcal{J}}{\partial u_{ik}} = m(u_{ik})^{m-1}D_{ik}$, $\frac{\partial^2 \mathcal{J}}{\partial u_{ik}^2} = m(m-1)(u_{ik})^{m-2}D_{ik}$ e $\frac{\partial^2 \mathcal{J}}{\partial u_{ik}\partial u_{ir}} = 0,\ \forall r \neq k$. A matriz Hessiana de $\mathcal{J}$ de acordo com $\mathbf{U}$ é

$$
\partial^2\mathcal{J}(\mathbf{U}) = \begin{pmatrix}
m(m-1)(u_{11})^{m-2}D_{11} & \cdots & 0 & 0 \\
\vdots & & \vdots\ \ddots & \vdots \\
0 & & 0\ \cdots & m(m-1)(u_{NK})^{m-2}D_{NK}
\end{pmatrix}
$$

de modo que $\partial^2\mathcal{J}(\mathbf{U})$ é positiva-definida, e consequentemente, podemos concluir que o extremo encontrado é um ponto de mínimo.

Importante: nós realizamos alguns experimentos e notamos que calcular o grau de pertencimento dos objetos segundo a Equação (4.19) aloca todos os objetos em uma única classe, o que é totalmente ilógico, haja vista que a tarefa de agrupar faz sentido quando existe mais de um grupo. Para contornar esse problema, nós adotamos uma heurística, como foi feito em (LACLAU; NADIF, 2016), que faz uma pequena mudança na Equação (4.19). Basicamente, a mudança consiste em remover os termos $1/u_{\bullet k}^{m}$ e $1/u_{\bullet r}^{m}$. Dessa forma, a formulação usada para computar as componentes da matriz $\mathbf{U}$ de pertencimento dos objetos é dada pela Equação (4.20). A Tabela 2 apresenta as regras de alocação dos objetos de acordo com as funções de distâncias kernelizadas, e consequentemente, de acordo com cada algoritmo de co-clustering proposto.

$$
u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}d^{2}(x_{ij},g_{kh})}{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}d^{2}(x_{ij},g_{rh})}\right)^{\frac{1}{m-1}}\right]^{-1}
\tag{4.20}
$$

Tabela 2 – Regras de alocação para a partição fuzzy dos objetos de acordo com cada algoritmo proposto de Co-clustering.

Algoritmo	Função objetivo	Regra para u_{ik}
GKFDK	Equação (4.6)	$u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}(1-\mathcal{K}(x_{ij},g_{kh}))}{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}(1-\mathcal{K}(x_{ij},g_{rh}))}\right)^{\frac{1}{m-1}}\right]^{-1}$
GKFDK-GP	Equação (4.7)	$u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}\lambda_{j}(1-\mathcal{K}(x_{ij},g_{kh}))}{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}\lambda_{j}(1-\mathcal{K}(x_{ij},g_{rh}))}\right)^{\frac{1}{m-1}}\right]^{-1}$
GKFDK-LP	Equação (4.8)	$u_{ik} = \left[\sum_{r=1}^{K}\left(\frac{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}\lambda_{kj}(1-\mathcal{K}(x_{ij},g_{kh}))}{\sum_{h,j}\frac{1}{v_{\bullet h}^{n}}(v_{jh})^{n}\lambda_{rj}(1-\mathcal{K}(x_{ij},g_{rh}))}\right)^{\frac{1}{m-1}}\right]^{-1}$

4.2.1.4 Etapa de alocação das variáveis

Essa etapa fornece a solução para calcular a matriz $\mathbf{V}$ de pertencimento fuzzy das variáveis em H grupos de variáveis. Na etapa de alocação das variáveis, a matriz de protótipos $\mathbf{G}$, os pesos de relevância $\mathbf{\Lambda}$ (para os algoritmos *GKFDK-GP* e *GKFDK-LP*) e a matriz $\mathbf{U}$ de pertencimento dos objetos são mantidos fixos.

Queremos minimizar $\mathcal{J}$ (Equação (4.5)) em relação a v_{jh} sob $v_{jh} \in [0,1]$ e $\sum_{h=1}^{H} v_{jh} = 1$. Definindo a função Lagrangiana como

$$\mathcal{L}_4 = \sum_{k,h,i,j} \frac{1}{u_{\bullet k}^{m}\ v_{\bullet h}^{n}} (u_{ik})^{m} (v_{jh})^{n} d^2(x_{ij}, g_{kh}) - \sum_{j} \theta_j \left(\sum_{h} v_{jh} - 1\right),$$

e seguindo os mesmos passos da Seção 4.2.1.3 (Etapa de alocação dos objetos) chegamos a

$$v_{jh} = \left[\sum_{w=1}^{H}\left(\frac{\sum\limits_{k,i}\frac{1}{u_{\bullet k}^{m}\ v_{\bullet h}^{n}}(u_{ik})^{m}d^2(x_{ij},g_{kh})}{\sum\limits_{k,i}\frac{1}{u_{\bullet k}^{m}\ v_{\bullet w}^{n}}(u_{ik})^{m}d^2(x_{ij},g_{kw})}\right)^{\frac{1}{n-1}}\right]^{-1} \tag{4.21}$$

Importante: Na etapa de alocação das variáveis acontece o mesmo problema da etapa de alocação dos objetos (ver Seção 4.2.1.3). Seguindo as mesmas justificativas, a matriz $\mathbf{V}$ de pertencimento das variáveis é atualizada de acordo com a Equação (4.22). A Tabela 3 apresenta

as regras de alocação das variáveis de acordo com as funções de distâncias kernelizadas, e consequentemente, de acordo com cada algoritmo fuzzy de co-clustering.

$$v_{jh} = \left[\sum_{w=1}^{H} \left(\frac{\sum\limits_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} d^{2}(x_{ij}, g_{kh})}{\sum\limits_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} d^{2}(x_{ij}, g_{kw})} \right)^{\frac{1}{n-1}} \right]^{-1} \tag{4.22}$$

Tabela 3 – Regras de alocação para a partição fuzzy das variáveis de acordo com cada algoritmo proposto de Co-clustering.

Algoritmo	Função Objetivo	Regra para v_{jh}
GKFDK e *GKFDK-GP*	Equação (4.6) e Equação (4.7)	$v_{jh} = \left[\sum_{w=1}^{H} \left(\frac{\sum_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} (1 - \mathcal{K}(x_{ij}, g_{kh}))}{\sum_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} (1 - \mathcal{K}(x_{ij}, g_{kw}))} \right)^{\frac{1}{n-1}} \right]^{-1}$
GKFDK-LP	Equação (4.8)	$v_{jh} = \left[\sum_{w=1}^{H} \left(\frac{\sum_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} \lambda_{kj} (1 - \mathcal{K}(x_{ij}, g_{kh}))}{\sum_{k,i} \frac{1}{u_{\bullet k}^{m}} (u_{ik})^{m} \lambda_{kj} (1 - \mathcal{K}(x_{ij}, g_{kw}))} \right)^{\frac{1}{n-1}} \right]^{-1}$

4.2.2 Análise de complexidade

Na etapa de representação, a complexidade de tempo para atualizar os protótipos é $O(NPKH)$, onde N é o número objetos, P é o número de variáveis, K é o número de grupos de objetos e H é o número de grupos de variáveis. Para obter os pesos de relevância das variáveis (algoritmos *GKFDK-GP* e *GKFDK-LP*), a complexidade computacional é $O(NPKH)$. A atualização dos graus de pertencimento fuzzy dos objetos e variáveis requer uma complexidade de tempo de $O(NPKH)$ em cada etapa. Dessa forma, assumindo que T é o número de iterações necessárias para estabilizar o algoritmo e alcançar a convergência, a complexidade de tempo de execução dos algoritmos propostos é $O(TNPKH)$.

4.2.3 Algoritmos de co-clustering fuzzy propostos

O Algoritmo 5 resume o passo a passo para executar os métodos de co-clustering fuzzy propostos. A Figura 9 mostra o fluxograma de cada algoritmo individualmente.

Algoritmo 5 Algoritmos propostos

Entrada: O conjunto de dados $\mathbf{X}$, o número de grupos de objetos K, o número dos grupos de variáveis H, os parâmetros fuzzificadores m e n, o hiperparâmetro do kernel gaussiano σ^2, o número máximo de iterações T e o limite de erro ε, com $0 < \varepsilon << 1$.

Saída: A matriz de protótipos $\mathbf{G}$, a matriz de pertinência fuzzy dos objetos $\mathbf{U}$, a matriz de pertinência das variáveis $\mathbf{V}$ e, para os algoritmos *GKFDK-GP* e *GKFDK-LP*, os pesos de relevância das variáveis $\mathbf{\Lambda}$.

Inicialização:

$\mathbf{U}$: inicializar aleatoriamente os graus de pertencimentos dos objetos u_{ik} $\forall i, k$, sujeito a $u_{ik} \in [0.1]$ e $\sum_{k=1}^{K} u_{ik} = 1$;
$\mathbf{V}$: inicializar aleatoriamente os graus de pertencimentos das variáveis v_{jh} $\forall j, h$, sujeito a $v_{jh} \in [0.1]$ e $\sum_{h=1}^{H} v_{jh} = 1$;
$\mathbf{G}$: inicializar os protótipos g_{kh} $\forall$ k, h, como:

$$g_{kh} = \frac{\sum_{i,j}(u_{ik})^m (v_{jh})^n x_{ij}}{\sum_{i,j}(u_{ik})^m (v_{jh})^n}$$

$\mathbf{\Lambda} = 1$: inicializar os pesos das variáveis como 1 (para os Algoritmos *GKFDK-GP* e *GKFDK-LP*).

$t \leftarrow 0$;

repita

$t \leftarrow t + 1$;

Etapa 1: Representação .

Calcule o componente g_{kh} da matriz de protótipos $\mathbf{G}$ de acordo com a Seção 4.2.1.1.

Etapa 2: Ponderação.

Se a distância considerada for a não-adaptativa (Equação (4.2)), então vá para a Etapa 3 (Etapa de alocação dos objetos). Caso contrário, atualize os pesos das variáveis $\mathbf{\Lambda}$ de acordo com a Seção 4.2.1.2.

Etapa 3: Alocação dos objetos

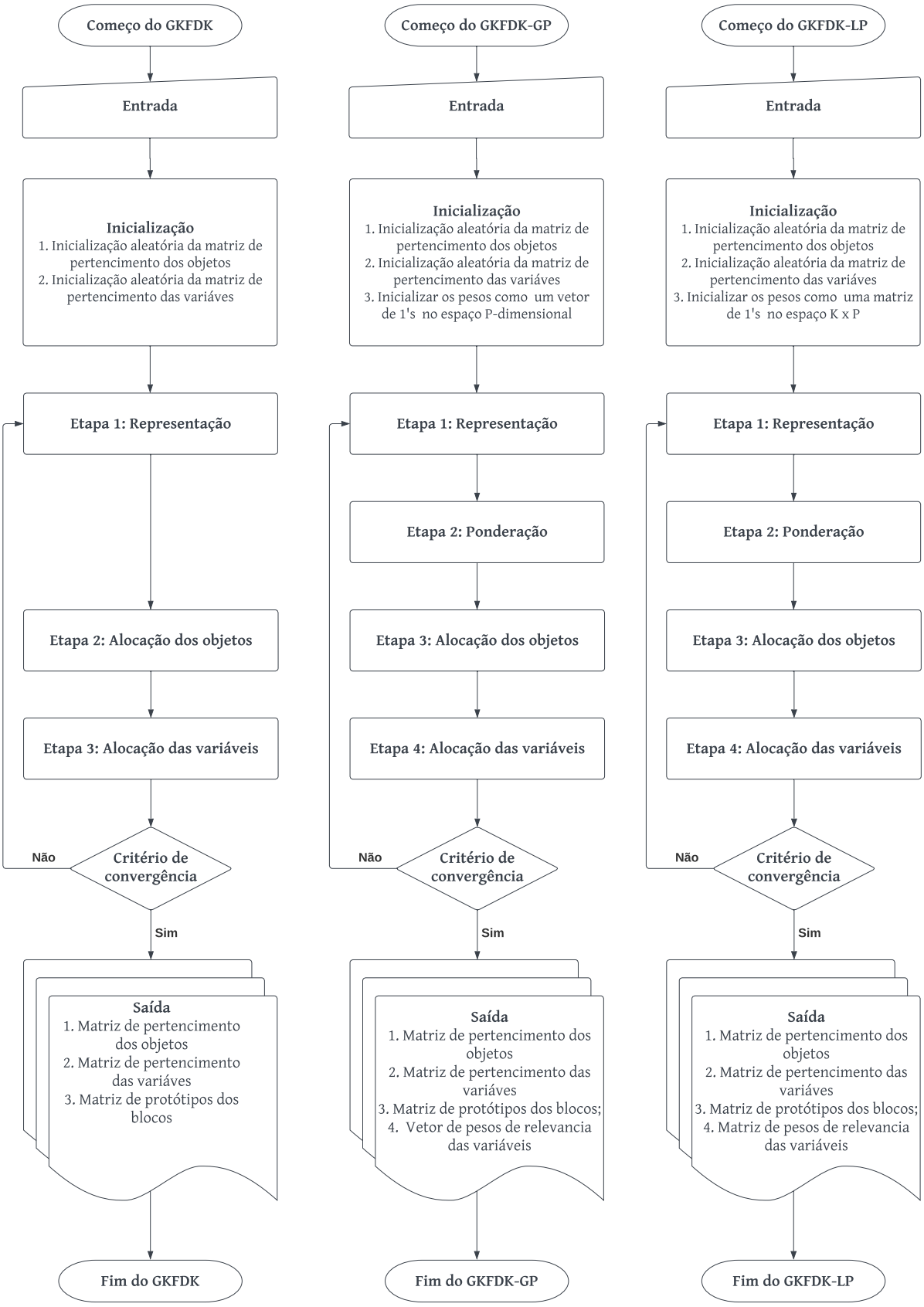
Atualize os graus de pertencimento u_{ik} da matriz de pertencimento dos objetos $\mathbf{U} = [u_{ik}]_{\substack{1 \leq i \leq N \\ 1 \leq k \leq K}}$ de acordo com a Equação (4.20) (ver a Tabela 2 para melhor entendimento).

Etapa 4: Alocação das variáveis

Atualize os graus de pertencimento v_{jh} da matriz de pertencimento das variáveis $\mathbf{V} = [v_{jh}]_{\substack{1 \leq j \leq P \\ 1 \leq h \leq H}}$ de acordo com a Equação (4.22) (ver a Tabela 3 para melhor entendimento).

até $|\mathcal{J}^{(t)} - \mathcal{J}^{(t-1)}| \leq \varepsilon$ ou $t > T$.

Figura 9 – Fluxograma dos algoritmos de co-clustering propostos.



Fonte: Autor (2023)

4.3 SÍNTESE DO CAPÍTULO

Este capítulo apresentou a principal contribuição desta dissertação: os algoritmos de co-clustering fuzzy baseados em kernel no espaço de características com ponderação automática das variáveis. Para calcular os pesos de relevância das variáveis foram considerados dois casos com a restrição produto: no primeiro caso, baseado na distância adaptativa global kernelizada, os pesos das variáveis eram os mesmos para todos os grupos de objetos, e nesse caso o produto dos pesos das variáveis deve ser igual a um; no segundo caso, baseado na distância adaptativa local kernelizada, os pesos das variáveis diferiam para cada grupo de objetos, e aqui, o produto dos pesos das variáveis em cada grupo deve ser igual a um.

As novas funções objetivo foram formuladas e foram encontradas as equações para a atualização dos protótipos dos blocos, dos pesos das variáveis, da matriz de pertencimento dos objetos e da matriz de pertencimento das variáveis.

5 AVALIAÇÃO EXPERIMENTAL

Esse capítulo apresenta a avaliação de desempenho dos algoritmos propostos em comparação com alguns métodos da literatura. O estudo comparativo foi feito através de uma série de experimentos realizados com dados sintéticos e dados reais. Nós realizamos esses experimentos em uma máquina modelo Dell Inc. Inspiron 5567 equipada com um processador de sétima geração Intel Core i5-7200U CPU @ 2.50GHz (frequência base), 4 cores, 8 GB de memória RAM, um armazenamento SSD de 512 GB e um sistema operacional Ubuntu 22.04.1 LTS de 64-bit com o Linux Kernel versão 5.15.0-56-generic. Todos os algoritmos (inclusive os métodos do estado da arte) foram implementados pelor autor usando a linguagem de programação Python, e os códigos estão disponíveis em: <https://github.com/Natandradesa/Kernel_Fuzzy_Co-clustering>. Vale destacar que a semente (*seed point*) utilizada para gerar os números aleatórios foi a mesma para todos os algoritmos, em cada experimento.

5.1 CONFIGURAÇÃO EXPERIMENTAL

Para avaliar a performance dos algoritmos propostos, quatro métodos do tipo fuzzy foram escolhidos para análise comparativa, sendo dois deles algoritmos de agrupamento tradicional: *Fuzzy C-Means* (*FCM*) (BEZDEK, 1981) e *Kernel Fuzzy C-Means* (*KFCM*) (GRAVES; PEDRYCZ, 2010), e os outros dois algoritmos de co-clustering: *Fuzzy Double K-Means* (*FDK*) (LACLAU; CARVALHO; NADIF, 2015) e *Weighted Fuzzy Double K-Means* (*WFDK*) (LACLAU; CARVALHO; NADIF, 2015). A Tabela 4 resume as diferentes características de cada algoritmo, incluindo o número de parâmetros que eles envolvem e a natureza de suas partições

Para estimar a qualidade dos resultados obtidos, utilizamos diferentes métricas de avaliação. Tendo em vista que o melhor desempenho de agrupamento indicado por algum índice nem sempre é consistente com outros índices, comparar os algoritmos baseados em várias medidas traz robustez para a nossa análise. As métricas usadas nesse trabalho são detalhadas na Seção 5.1.1.

Tabela 4 – Funções objetivos, parâmetros e categorias dos algoritmos.

Métodos	Características				
	Critério	Parâmetros	Co-clustering	Kernel	Ponderação
FCM	$\sum_{k,i,j}(u_{ik})^m \left(x_{ij} - c_{kj}\right)^2$	m			
KFCM	$2\sum_{k,i}(u_{ik})^m \left[1 - \mathcal{K}(\mathbf{x}_i, \mathbf{c}_k)\right]$	m, σ^2		$\times$	
FDK	$\sum_{k,h,i,j}(u_{ik})^m (v_{jh})^n \left(x_{ij} - g_{kh}\right)^2$	m, n	$\times$		
WFDK	$\sum_{k,h,i,j}(u_{ik})^m (v_{jh})^n \lambda_j^\gamma \left(x_{ij} - g_{kh}\right)^2$	m, n, γ	$\times$		$\times$
GKFDK	$2\sum_{k,h,i,j}(u_{ik})^m (v_{jh})^n \left[1 - \mathcal{K}(x_{ij}, g_{kh})\right]$	m, n, σ^2	$\times$	$\times$	
GKFDK-LP	$2\sum_{k,h,i,j}(u_{ik})^m (v_{jh})^n \lambda_{kj} \left[1 - \mathcal{K}(x_{ij}, g_{kh})\right]$	m, n, σ^2	$\times$	$\times$	$\times$
GKFDK-GP	$2\sum_{k,h,i,j}(u_{ik})^m (v_{jh})^n \lambda_j \left[1 - \mathcal{K}(x_{ij}, g_{kh})\right]$	m, n, σ^2	$\times$	$\times$	$\times$

5.1.1 Métricas de avaliação

O desempenho dos algoritmos pode ser computado por métricas adequadas para métodos de agrupamento. De modo geral, essas medidas estimam a qualidade da partição fornecida por um algoritmo, seja ela uma partição fuzzy ou uma partição crisp (hard). Ainda, essas métricas de avaliação podem ser divididas em índices externos e índices internos. Os primeiros são baseados em critérios externos. Isso implica que os algoritmos são avaliados de acordo com uma estrutura previamente especificada imposta a um conjunto de dados e reflete nossa intuição a respeito da estrutura de agrupamento do conjunto de dados. Os índices internos não dependem de informações externas (conhecimento prévio). Pelo contrário, eles examinam a estrutura de agrupamento diretamente dos dados originais.

Nesta dissertação, a performance dos algoritmos foi computada através de seis métricas: *Adjusted Rand Index* (*ARI*), *F-Measure* (*FMS*), *Normalized Mutual Information* (*NMI*), *Modified Partition Coefficient* (*MPC*), *Frigui Index* (*FRG*) e *Hullermeier Index* (*HUL*). Das seis, apenas o *MPC* é um critério interno, i.e., usa apenas a informação fornecida pelo algoritmo para estimar a qualidade do resultado obtido. Além disso, as métricas *ARI*, *FMS* e *NMI* são índices para avaliar uma partição crisp, enquanto *MPC*, *FRG* e *HUL* estimam a qualidade de uma partição fuzzy.

Observação: na grande maioria dos casos, apenas as informações a priori a respeito dos objetos estão disponíveis. Por esse motivo, nesse trabalho, nós calculamos os valores das métricas apenas para a partição dos objetos.

A partir de uma partição fuzzy $\mathbf{U} = [u_{ik}]_{\substack{1\leq i\leq N \\ 1\leq k\leq K}}$, uma partição crisp $\mathcal{Z} = \{\mathcal{Z}_1, ..., \mathcal{Z}_K\}$ pode ser obtida da seguinte forma:

$$\mathcal{Z}_k = \{i \in \{1, ..., N\} : u_{ik} \geq u_{ir}, \forall r \in \{1, ..., K\}\} \quad (1 \leq k \leq K)$$

Considere $\mathcal{Q} = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_r, \ldots, \mathcal{Q}_c\}$ uma partição a priori de N objetos em c classes e seja $\mathcal{Z} = \{\mathcal{Z}_1, \ldots, \mathcal{Z}_k, \ldots, \mathcal{Z}_K\}$ uma partição de N objetos em K grupos fornecida por um algoritmo de agrupamento. Considerando que as quantidades N_{rk} ($r = 1, ..., c$ e $k = 1, ..., K$) representam o número de observações que pertence à classe $\mathcal{Q}_r$ e ao grupo Z_k ($N_{rk} = |\mathcal{Q}_r \bigcap \mathcal{Z}_k|$), é possível representá-las através de uma matriz de confusão (Tabela 5).

Tabela 5 – Matriz de confusão

	Grupos					
Classes	$\mathcal{Z}_1$	$\cdots$	$\mathcal{Z}_k$	$\cdots$	$\mathcal{Z}_K$	$\sum$
$\mathcal{Q}_1$	N_{11}	$\cdots$	N_{1k}	$\cdots$	N_{1K}	$N_{1\bullet} = \sum_{k=1}^{K} N_{1k}$
$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$
$\mathcal{Q}_r$	N_{r1}	$\cdots$	N_{rk}	$\cdots$	N_{rK}	$N_{r\bullet} = \sum_{k=1}^{K} N_{rk}$
$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$
$\mathcal{Q}_c$	N_{c1}	$\cdots$	N_{ck}	$\cdots$	N_{cK}	$N_{c\bullet} = \sum_{k=1}^{K} N_{ck}$
$\sum$	$N_{\bullet 1} = \sum_{r=1}^{c} N_{r1}$	$\cdots$	$N_{\bullet k} = \sum_{r=1}^{c} n_{rk}$	$\cdots$	$N_{\bullet K} = \sum_{r=1}^{c} N_{rK}$	$N = \sum_{r=1}^{c}\sum_{k=1}^{K} Nrk$

5.1.1.1 Adjusted Rand Index

O *Adjusted Rand Index* (*ARI*) (HUBERT; ARABIE, 1985) é um índice de validação externo que mede a similaridade entre duas partições de um mesmo conjunto de dados. O *ARI* assume valores no intervalo [-1,1], de modo que 1 indica concordância perfeita entre as partições, enquanto valores negativos ou próximos de zero indicam similaridade ao acaso.

O *ARI* é uma correção do Rand Index (RI) (RAND, 1971). O RI mede a concordância entre duas partições como a proporção de atribuições corretas. Um problema relacionado ao Rand Index é que para duas partições aleatórias, podemos obter diferentes valores maiores que zero. O *ARI* foi proposto para superar essa limitação. A ideia do *Adjusted Rand Index* é comparar o RI com o valor esperado de RI assumindo que as duas partições são independentes. A forma ajustada do Rand Index é definida como:

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{E(RI)}}{\max(\mathrm{RI}) - \mathrm{E(RI)}},$$

onde E(RI) é o Rand Index esperado. Usando uma distribuição hipergeométrica generalizada como modelo de aleatoriedade, o *Adjusted Rand Index* é definido como:

$$\mathrm{ARI} = \frac{\sum_{r=1}^{c}\sum_{k=1}^{K}\binom{N_{rk}}{2} - \binom{N}{2}^{-1}\sum_{r=1}^{c}\binom{N_{r\bullet}}{2}\sum_{k=1}^{K}\binom{N_{\bullet k}}{2}}{\frac{1}{2}\left[\sum_{r=1}^{c}\binom{N_{r\bullet}}{2} + \sum_{k=1}^{K}\binom{N_{\bullet k}}{2}\right] - \binom{N}{2}^{-1}\sum_{r=1}^{c}\binom{N_{r\bullet}}{2}\sum_{k=1}^{K}\binom{N_{\bullet k}}{2}},$$

onde $N_{r\bullet}$ representa o número de objetos na classe $\mathcal{Q}_r$, $N_{\bullet k}$ representa o número de objetos no grupo $\mathcal{Z}_k$ e N é o número total de objetos no conjunto de dados.

5.1.1.2 F-Measure

A tradicional *F-Measure* (*FMS*) entre uma classe $\mathcal{Q}_r$ $(r = 1, ..., c)$ e um grupo $\mathcal{Z}_k(k = 1, ..., K)$ é definida como sendo a média harmônica entre a precisão e a revocação, isto é:

$$\mathrm{FMS}(\mathcal{Q}_r, \mathcal{Z}_k) = 2\frac{\mathrm{Precisão}(\mathcal{Q}_r, \mathcal{Z}_k) \times \mathrm{Revocação}(\mathcal{Q}_r, \mathcal{Z}_k)}{\mathrm{Precisão}(\mathcal{Q}_r, \mathcal{Z}_k) + \mathrm{Revocação}(\mathcal{Q}_r, \mathcal{Z}_k)},$$

A precisão entre uma classe $\mathcal{Q}_r$ e um grupo $\mathcal{Z}_k$ é a fração entre o número de objetos que estão simultaneamente na classe $\mathcal{Q}_r$ e no grupo $\mathcal{Z}_k$ e o número de objetos no grupo $\mathcal{Z}_k$, isto é:

$$\mathrm{Precisão}(\mathcal{Q}_r, \mathcal{Z}_k) = \frac{N_{rk}}{N_{\bullet k}} = \frac{N_{rk}}{\sum_{r=1}^{c} N_{rk}}$$

A revocação entre uma classe $\mathcal{Q}_r$ e um grupo $\mathcal{Z}_k$ é a fração entre o número de objetos que estão simultaneamente na classe $\mathcal{Q}_r$ e no grupo $\mathcal{Z}_k$ e o número de objetos na classe $\mathcal{Q}_r$, isto é:

$$\mathrm{Revocação}(\mathcal{Q}_r, \mathcal{Z}_k) = \frac{N_{rk}}{N_{r\bullet}} = \frac{N_{rk}}{\sum_{k=1}^{K} N_{rk}}$$

Dessa forma, a *FMS* (BAEZA-YATES; RIBEIRO-NETO et al., 1999) entre uma partição a priori $\mathcal{Q} = \{\mathcal{Q}_1, ..., \mathcal{Q}_r, ..., \mathcal{Q}_c\}$ e uma partição crisp $\mathcal{Z} = \{\mathcal{Z}_1, ..., \mathcal{Z}_k, ..., \mathcal{Z}_K\}$ fornecida por uma algoritmo de agrupamento, é dada por:

$$\mathrm{FMS}(\mathcal{Q}, \mathcal{Z}) = \frac{1}{N}\sum_{r=1}^{c} N_{r\bullet}\ \max_{1\leq k\leq K}\mathrm{FMS}(\mathcal{Q}_r, \mathcal{Z}_k),$$

onde $N_{\bullet k}$, $N_{r\bullet}$ e N são definidos como antes. A *F-Measure* assume valores no intervalo (0,1) no qual o valor 1 indica concordância perfeita entre as partições.

5.1.1.3 Normalized Mutual Information

O *Normalized Mutual Information* (*NMI*) (STREHL; GHOSH, 2002) é baseado na Informação Mútua, do inglês Mutual Information (MI), sendo um critério que mede a correlação entre a

repartição de objetos nos grupos e a repartição de objetos nas classes verdadeiras. O MI é definido como:

$$\mathrm{MI}(\mathcal{Z},\mathcal{Q}) = \sum_{k=1}^{K}\sum_{r=1}^{c} p(\mathcal{Z}_k,\mathcal{Q}_r)\log\left(\frac{p(\mathcal{Z}_k,\mathcal{Q}_r)}{p(\mathcal{Z}_k)p(\mathcal{Q}_r)}\right),$$

onde $p(\mathcal{Z}_k,\mathcal{Q}_r)$ é a função densidade de probabilidade conjunta de $\mathcal{Z}$ e $\mathcal{Q}$, enquanto $p(\mathcal{Z}_k)$ e $p(\mathcal{Q}_r)$ são as funções densidade de probabilidade marginais de $\mathcal{Z}$ e $\mathcal{Q}$, respectivamente.

Para aumentar a interpretabilidade do MI, que não tem um limite superior, o *NMI* foi proposto. A normalização do MI pode ser feita de algumas maneiras, como, por exemplo, baseado na média geométrica das entropias F de $\mathcal{Z}$ e $\mathcal{Q}$:

$$\mathrm{NMI}(\mathcal{Z},\mathcal{Q}) = \frac{\mathrm{MI}(\mathcal{Z},\mathcal{Q})}{\sqrt{\mathrm{F}(\mathcal{Z})\mathrm{F}(\mathcal{Q})}}$$

Do ponto de vista estatístico, esta normalização deriva do primeiro pensamento da informação mútua como um análogo da covariância e seu cálculo está relacionado ao coeficiente de correlação de Pearson (LACLAU, 2016). Finalmente temos:

$$\mathrm{NMI} = \frac{\sum_{k=1}^{K}\sum_{r=1}^{c} N_{rk}\log\left(\frac{N_{rk}}{N_{\bullet k}N_{r\bullet}}\right)}{\sqrt{\left(\sum_{k=1}^{K} N_{\bullet k}\log\left(\frac{N_{\bullet k}}{N}\right)\right)\left(\sum_{r=1}^{c} N_{l\bullet}\log\left(\frac{N_{r\bullet}}{N}\right)\right)}},$$

onde $N_{\bullet k}$, $N_{r\bullet}$ e N são definidos como antes. O *NMI* é uma métrica que mede a concordância entre duas partições e toma seus valores no intervalo [0,1]. Quanto mais próximos seus valores estiverem de 1, mais concordantes serão as partições.

5.1.1.4 Modified Partition Coefficient

Dada uma partição $\mathbf{U}$ de N objetos em K grupos, Bezdek (BEZDEK, 2013) definiu uma medida baseada na minimização do conteúdo geral da partição $\mathbf{U}$. Ele propôs um índice de validação para agrupamento fuzzy chamado de Partition Coefficient (PC), definido como

$$\mathrm{PC} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}(u_{ik})^2$$

O índice PC indica a quantidade relativa média de compartilhamento de pertinência feito entre pares de subconjuntos fuzzy em $\mathbf{U}$. O PC toma seus valores no intervalo $[\frac{1}{K}, 1]$, de modo que PC $= 1$ indica que $\mathbf{U}$ é uma partição crisp.

Segundo o autor, o índice PC apresenta uma tendência monotônica com o número de grupos K. Para eliminar essa limitação, nós usamos o *Modified Partition Coefficient* (*MPC*)

(DAVE, 1996) que aplica uma transformação linear no índice PC. O *MPC* é definido como:

$$\mathrm{MPC} = 1 - \frac{K}{K-1}(1 - \mathrm{PC})$$

O *MPC* varia no intervalo [0,1], de modo que valores próximos a zero indicam alta imprecisão, no sentido de que os grupos apresentam alto grau de sobreposição, enquanto valores próximos a um indicam que os grupos estão bem definidos e que a partição $\mathbf{U}$ é próxima a uma partição crisp.

5.1.1.5 Frigui Index

Métricas como *ARI*, *FMS* e *NMI* não fazem o uso explicito da partição fuzzy fornecida pelos algoritmos de clustering. Consequentemente, podem não conseguir discriminar entre grupos de dados sobrepostos, uma vez que negligenciam as informações contidas na matriz de partição fuzzy. O *Frigui Index* (*FRG*) (FRIGUI; HWANG; RHEE, 2007) é uma versão do Rand Index (RI) para partição fuzzy. Seja $\mathbf{U}^{(T)} = [u_{ik}^{(T)}]_{\substack{1\leq i\leq N \\ 1\leq k\leq K}}$, onde $u_{ik}^{(T)} \in \{0,1\}$ com $\sum_{k=1}^{K} u_{ik}^{(T)} = 1$ uma matriz de partição crisp construída a partir da partição a priori $\mathcal{Q}$. A ideia é comparar a partição fuzzy $\mathbf{U}$, fornecida por um algoritmo de clustering, com $\mathbf{U}^{(T)}$ construída a partir de uma partição a priori.

Frigui et al (FRIGUI; HWANG; RHEE, 2007) fizeram essa comparação com base em matrizes de coincidência, também chamadas de matrizes de conectividade de grupos (LEVINE; DOMANY, 2001). Para comparar duas matrizes de partição $\mathbf{U}^{(1)}$ e $\mathbf{U}^{(2)}$, primeiro calcula-se a suas matrizes de coincidência $\mathbf{\Psi}^{(1)}$ e $\mathbf{\Psi}^{(2)}$, onde $\mathbf{\Psi} = [\psi_{il}]_{1\leq i,l\leq N}$ e $\psi_{il} = \sum_{k=1}^{K} u_{ik}u_{lk}$. Dessa forma, as seguintes quantidades são calculadas:

$$\begin{aligned}
N_{SS}\left(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}\right) &= \sum_{i=2}^{N}\sum_{l=1}^{i-1} \psi_{il}^{(1)}\psi_{il}^{(2)}, \\
N_{SD}\left(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}\right) &= \sum_{i=2}^{N}\sum_{l=1}^{i-1} \psi_{il}^{(1)}\left(1 - \psi_{il}^{(2)}\right), \\
N_{DS}\left(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}\right) &= \sum_{i=2}^{N}\sum_{l=1}^{i-1} \left(1 - \psi_{il}^{(1)}\right)\psi_{il}^{(2)}, \\
N_{DD}\left(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}\right) &= \sum_{i=2}^{N}\sum_{l=1}^{i-1} \left(1 - \psi_{il}^{(1)}\right)\left(1 - \psi_{il}^{(2)}\right),
\end{aligned}$$

onde S denota "mesmo grupo" e D "denota grupos diferentes".

Baseado nessas quantidades, diversas medidas podem ser calculadas (FRIGUI; HWANG; RHEE, 2007). Nessa dissertação utilizamos o índice baseado no Rand Index, que pode ser

computado como:

$$\mathrm{FRG} = \frac{N_{SS} + N_{DD}}{N_{SS} + N_{SD} + N_{DS} + N_{DD}}$$

Este índice assume valores no intervalo [0, 1], no qual o valor 1 indica concordância perfeita entre as partições, enquanto valores menores ou próximos a $1/2$ indicam que a concordância entre as partições foi encontrada ao acaso.

5.1.1.6 Hullermeier Index

Hullermeier *et al* (HULLERMEIER et al., 2011) argumenta que medidas como o índice de Frigui (FRIGUI; HWANG; RHEE, 2007) é, de certa forma, defeituosa, pois não é uma métrica, no sentido de que não satisfaz algumas propriedades. Em vez disso, a métrica proposta por eles é guiada pelo fato de que o Rand Index (RI) conta o número de atribuições corretas dividido pelo número total de pares possíveis, e isso os leva a uma generalização direta do Rand Index:

$$\mathrm{HUL}(\mathbf{U}, \mathbf{U}^{(T)}) = 1 - \left[\frac{\sum_{i=2}^{N-1} \sum_{l=1}^{i-1} |||\mathbf{U}_i^{(T)} - \mathbf{U}_l^{(T)}|| - ||\mathbf{U}_i - \mathbf{U}_l|||}{\binom{N}{2}} \right],$$

onde $||.||$ é uma métrica adequada definida em $[0, 1]^K$. Nessa dissertação usamos $||.||$ como sendo a norma L_1 dividia por 2, conforme a referência (HULLERMEIER et al., 2011). O índice HUL varia no intervalo [0, 1], onde o valor 1 indica concordância perfeita entre as partições fuzzy, enquanto valores próximos a 0 correspondem à concordância encontrada ao acaso.

5.1.2 Configuração dos hiperparâmetros

Um importante e complicado processo em algoritmos de clustering é a seleção dos hiperparâmetros dos modelos. Primeiro, porque pequenas mudanças podem, consideravelmente, melhorar ou piorar a performance dos algoritmos. Segundo, como estamos trabalhando na abordagem não supervisionada, não existem rótulos dos dados e, dessa forma, não podemos proceder como na abordagem supervisionada, dificultando a escolha desses valores. Ainda assim, muitos trabalhos na literatura ajustam os valores dos parâmetros usando os rótulos dos objetos, o que é impraticável em aplicações reais de agrupamento. Neste trabalho, fizemos a escolha dos hiperparâmetros baseados em uma abordagem não supervisionada, isto é, sem usar os rótulos dos dados.

Todos os parâmetros foram escolhidos baseado na estratégia de grid. Dado um conjunto de valores pré-fixados para cada parâmetro, os algoritmos foram executados individualmente em

cada conjunto de dados, para todas as possíveis combinações dos valores dos hiperparâmetros. Para cada combinação, cada algoritmo foi executado cinco vezes. Em cada uma das execuções, foi computado a distância mínima entre os protótipos dos grupos, no caso de algoritmos de agrupamento tradicional, e a distância mínima entre os protótipos dos blocos, no caso dos algoritmos de co-clustering. Para cada combinação, tomamos como valor final o valor mínimo entre as cinco execuções. Por fim, a combinação de parâmetros escolhida era aquela cujo valor final era o máximo entre todas as combinações.

Uma limitação técnica encontrada na seleção dos parâmetros foi em relação ao intervalo de busca de m e n nos algoritmos de co-clustering. Para valores muito pequenos (valores inferiores a 1.01) os termos $(D_{ik})^{1/1-m}$ e $(D_{jh})^{1/1-n}$, dos cálculos dos graus de pertinência dos objetos e variáveis, respectivamente, "explodiam" para o infinito (computacionalmente falando), prejudicando o cálculo de u_{ik} e v_{jh} na grande maioria dos conjuntos de dados. Por outro lado, valores altos para m e n tendem a deixar os protótipos dos blocos mais próximos. Isso é uma característica de todos os algoritmos fuzzy, mas é mais acentuado nos métodos de co-clustering, em comparação com os algoritmos de agrupamento tradicional. Dessa forma, nós procedemos da seguinte maneira: inicialmente a busca era feita considerando valores menores que 1.01. Caso, o problema fosse observado, nós restringiríamos a busca para valores no intervalo [1.01, 2.2]. Limitamos o intervalo em 2.2 devido ao segundo problema.

Em relação ao parâmetro γ do algoritmo *WFDK* fizemos uma busca de valores no conjunto: $\{1.1,\ 1.5,\ 2.0,\ 2.5,\ 3.0\}$. Quanto a σ^2, parâmetro de largura do kernel gaussiano, nós definimos o conjunto de busca a partir de um valor inicial, estimado segundo Caputo (CAPUTO et al., 2002). Considere um conjunto de dados $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N) = [x_{ij}]_{\substack{1 \leq i \leq N \\ 1 \leq j \leq P}}$. Para o algoritmo *KFCM*, o valor inicial de σ^2 foi estimado como sendo $\sigma_0^2 = (Q_{10} + Q_{90})/4$, onde Q_{10} e Q_{90} são, respectivamente, os quantis 0.1 e 0.9 de $||\mathbf{x}_l - \mathbf{x}_r||^2$, $l \neq r$. Para os algoritmos propostos, nós estimamos o valor inicial de σ^2 como sendo $\sigma_0^2 = (Q_{10} + Q_{90})/4$, onde Q_{10} e Q_{90} são, respectivamente, os quantis 0.1 e 0.9 de $(x_{ab} - x_{cd})^2$, $a \neq c$ e $b \neq d$. Feito isso, para a aplicação com dados simulados, onde se tem controle sobre a geração dos dados, utilizamos σ_0^2 como valor estimado para σ^2. Para aplicações com dados reais, o conjunto de busca para σ^2 foi definido como $\{0.1\sigma_0^2,\ \sigma_0^2,\ 10\sigma_0^2\}$. Por fim, o parâmetro ε foi definido como 10^{-10} e o número máximo de iterações T como 100.

5.2 APLICAÇÕES COM DADOS SIMULADOS

Para mostrar a robustez dos algoritmos propostos, fizemos aplicações a dados sintéticos. Neste caso temos controle sobre o mecanismo de geração de dados e podemos gerar diferentes cenários conforme as características dos modelos. Para avaliar o desempenho dos algoritmos, nós consideramos quatro configurações de dados sintéticos assumindo uma dupla partição dos dados em $K = 5$ grupos de objetos e $H = 3$ grupos de variáveis. Em todas as configurações os conjuntos de dados eram constituídos de $N = 50$ objetos e $P = 30$ variáveis, de modo que cada grupo de objetos e cada grupo de variáveis era composto por 10 observações. Cada bloco de dados $\mathbf{X}_{kh}$ foi gerado a partir de uma distribuição Normal (Gaussiana) com média μ_{kh} e variância σ^2_{kh}. Os experimentos foram realizados a partir de simulações de Monte Carlo com 100 réplicas. Em cada réplica, os algoritmos foram executados (até a convergência para um valor estacionário do critério de adequação) 30 vezes e o melhor resultado foi selecionado segundo o critério de adequação.

Conjunto de dados sintéticos 1: essa configuração é o cenário básico. Os grupos de objetos e os grupos de variáveis são bem definidos, de modo que os blocos de dados também são bem definidos. Nesse cenário, espera-se que todos os algoritmos apresentem bons resultados. Os parâmetros μ e σ usados para gerar os blocos de dados estão dispostos na Tabela 6. A Figura 10 (a) mostra uma visualização geral desse conjunto de dados.

Tabela 6 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 1.

		Variáveis					
		1-10		11-20		21-30	
		μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}
Objetos	1-10	1	0.4	2	0.5	11	0.6
	11-20	3	0.4	4	0.5	12	0.6
	21-30	5	0.4	6	0.5	13	0.6
	31-40	7	0.4	8	0.5	14	0.6
	41-50	9	0.4	10	0.5	15	0.6

Conjunto de dados sintéticos 2: nesse cenário todas as classes de objetos foram bem definidas nas duas primeiras classes de variáveis (20 primeiras variáveis). A última classe de variáveis foi considerada sendo não informativa, isto é, para esse conjunto de variáveis as classes de objetos não são bem definidas. Esse tipo de variável afeta consideravelmente os algoritmos de agrupamento tradicional. Dessa forma, espera-se que os algoritmos de co-

clustering apresentem melhores resultados, tendo em vista que estes também agrupam as variáveis. Os parâmetros μ e σ usados para gerar os blocos de dados nessa configuração estão dispostos na Tabela 7. A Figura 10 (b) mostra uma visualização geral desse conjunto de dados.

Tabela 7 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 2.

		Variáveis					
		1-10		11-20		21-30	
		μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}
Objetos	1-10	1	0.4	2	0.5	0	5.0
	11-20	3	0.4	4	0.5	0	5.0
	21-30	5	0.4	6	0.5	0	5.0
	31-40	7	0.4	8	0.5	0	5.0
	41-50	9	0.4	10	0.5	0	5.0

Conjunto de dados sintéticos 3: nessa configuração o número de variáveis irrelevantes é maior do que o número de variáveis relevantes. As classes de objetos estão bem definidas apenas na primeira classe de variáveis. As variáveis que compõem os outros dois blocos são não informativas para as classes de objetos. O objetivo aqui é destacar as características dos algoritmos com ponderação global, ou seja, *WFDK* e *GKFDK-GP*. Espera-se que o algoritmo com ponderação local *GKFDK-LP* também apresente um bom desempenho, haja vista que o *GKFDK-GP* pode ser considerado um caso particular quando as variáveis apresentam o mesmo peso para todos os grupos de objetos. Os parâmetros μ e σ usados para gerar os blocos de dados nessa configuração estão dispostos na Tabela 8. A Figura 10 (c) mostra uma visualização geral desse conjunto de dados.

Tabela 8 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 3.

		Variáveis					
		1-10		11-20		21-30	
		μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}
Objetos	1-10	1	1.0	20.01	5.0	0	5.0
	11-20	3	1.0	20.02	5.0	0	5.0
	21-30	5	1.0	20.03	5.0	0	5.0
	31-40	7	1.0	20.04	5.0	0	5.0
	41-50	9	1.0	20.05	5.0	0	5.0

Conjunto de dados sintéticos 4: essa configuração foi gerada para fornecer um cenário na qual o algoritmo *GKFDK-LP* deve ser o melhor candidato para encontrar a estrutura de grupos

dos objetos. Para isso, nós definimos diferentes classes de variáveis informativas para classes particulares de objetos. Especificamente, a primeira classe de variáveis (as dez primeiras) é informativa apenas para as duas primeiras classes de objetos. A segunda classe de variáveis (10-20) é informativa apenas para as três últimas classes de variáveis, enquanto a terceira classe de variáveis (as dez últimas) é não informativa para as classes de objetos. Os parâmetros μ e σ usados para gerar os blocos de dados nessa configuração estão dispostos na Tabela 9. A Figura 10 (d) mostra uma visualização geral desse conjunto de dados.

Tabela 9 – Parâmetros μ e σ da distribuição gaussiana usados para gerar cada bloco do conjunto de dados sintéticos 4.

		Variáveis					
		1-10		11-20		21-30	
		μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}	μ_{kh}	σ_{kh}
Objetos	1-10	1	0.4	0	5.0	0	5.0
	11-20	3	0.4	0	5.0	0	5.0
	21-30	0	5.0	6	0.5	0	5.0
	31-40	0	5.0	8	0.5	0	5.0
	41-50	0	5.0	10	0.5	0	5.0

5.2.1 Resultados e discussão

Os valores dos hiperparâmetros dos algoritmos usados nos dados simulados estão dispostos na Tabela 10. Esses valores foram obtidos segundo a estratégia descrita na Seção 5.1.2.

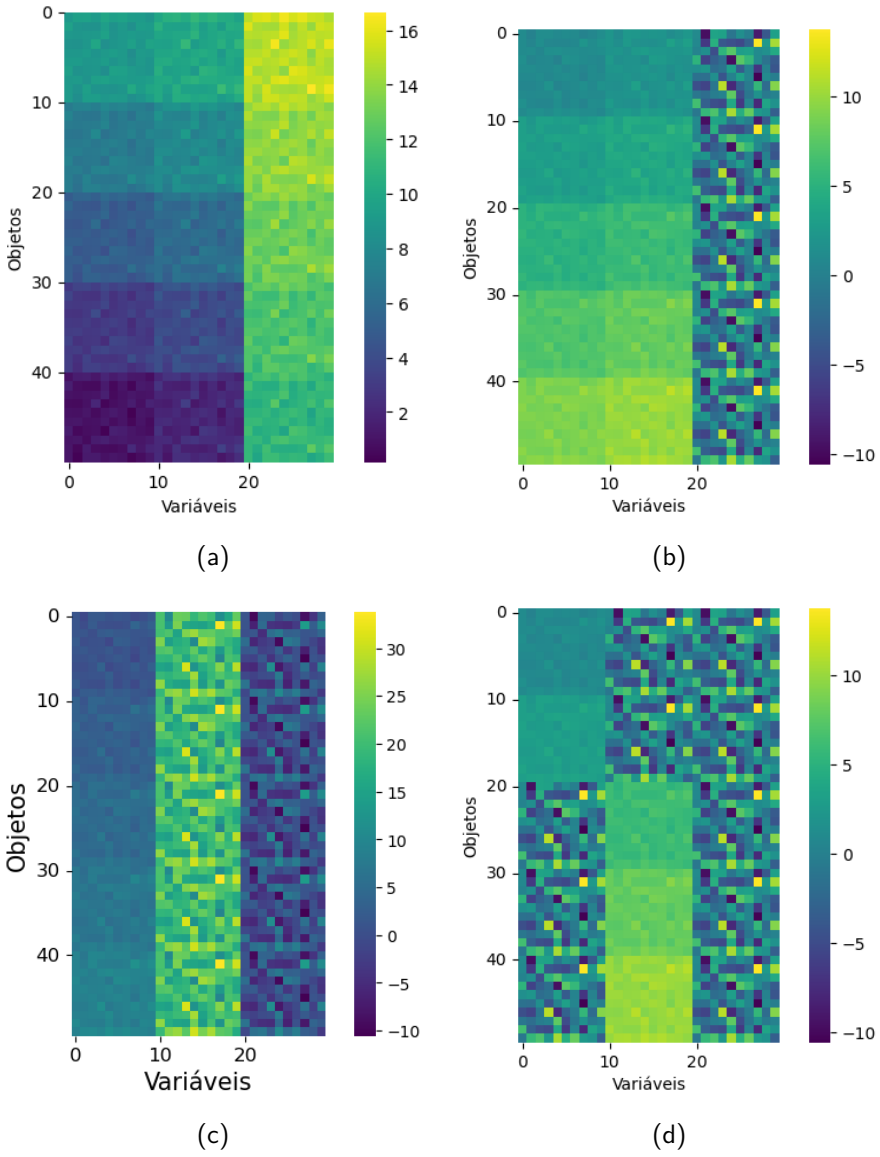
Tabela 10 – Valores dos parâmetros de cada algoritmo nos conjuntos de dados simulados.

Algoritmo	Simulação 1	Simulação 2	Simulação 3	Simulação 4
FCM (m)	1.6	1.1	1.1	1.01
KFCM (m)	1.5	1.1	1.01	1.1
FDK (m; n)	(1.9; 1.4)	(1.1; 1.4)	(1.01; 1.1)	(1.01; 1.01)
WFDK (m; n; γ)	(1.9; 1.2; 2.0)	(1.4; 1.01; 2.0)	(1.3; 1.2; 2.5)	(1.1; 1.01; 2.5)
GKFDK (m; n)	(1.9; 1.2)	(1.1; 1.01)	(1.1; 1.9)	(1.1; 1.01)
GKFDK-GP (m; n)	(1.9; 1.1)	(1.9; 1.1)	(1.1; 1.3)	(1.1; 1.01)
GKFDK-LP (m; n)	(1.2; 1.6)	(1.5; 1.1)	(1.2; 1.6)	(1.6; 1.01)

O parâmetro σ^2 dos algoritmos KFCM, GKFDK, GKFDK-GP e GKFDK-LP foi computado em cada réplica de acordo com Caputo (CAPUTO et al., 2002), como discutido na Seção 5.1.2

A Tabela 11 mostra a média e o desvio padrão obtidos nas 100 réplicas de Monte Carlo para todas as métricas de avaliação. Nós realizamos o teste de hipóteses de Friedman (FRIEDMAN,

Figura 10 – Visualização dos conjuntos de dados simulados. (a) Conjunto de dados sintéticos 1; (b) Conjunto de dados sintéticos 2; (c) Conjunto de dados sintéticos 3; (d) Conjunto de dados sintéticos 4.



Fonte: Autor (2023)

1937) para explorar a significância estatística dos resultados obtidos. Esse teste analisa a diferença entre mais de dois algoritmos, simultaneamente, através do ranqueamento baseado nos desempenhos dos algoritmos. Mais precisamente, o teste compara o ranking médio dos algoritmos nas 100 réplicas de Monte Carlo. A hipótese nula, de que todos os algoritmos apresentam o mesmo desempenho, é testada contra a hipótese alternativa, de que pelo menos um dos algoritmos apresenta desempenho diferente. Em caso de rejeição da hipótese nula, nós realizamos o teste post-hoc de Nemenyi (NEMENYI, 1963) que compara os algoritmos entre si e determina quais deles apresentam desempenhos estatisticamente diferentes.

A Tabela 12 mostra o p-valor do teste de Friedman obtido em cada aplicação aos dados

simulados. Tomando 5% como nível de significância, o teste rejeitou a hipótese nula (todos os algoritmos apresentam desempenhos equivalentes) em todos os conjunto de dados e para todos os índices de avaliação, exceto para os índices *ARI*, *FMS* e *NMI* no conjunto de dados simulados 1. Para as simulações 2, 3 e 4, o teste de Nemenyi está disposto nas Figuras 11, 12 e 13, respectivamente.

Em relação ao conjunto de dados simulados 1, todos os algoritmos apresentaram bons desempenhos, de modo que, de acordo com as métricas crisp, todos eles conseguiram capturar a verdadeira estrutura de dados perfeitamente. Em relação às métricas fuzzy, os métodos obtiveram bons resultados, com destaque para o algoritmo proposto *GKFDK-LP* que se sobressaiu em relação aos demais e, segundo os testes estatísticos, esse desempenho superior é significativo.

No conjunto de dados simulados 2 foram introduzidas variáveis irrelevantes para as classes de objetos. Nesse caso, como esperado, os algoritmos convencionais de agrupamento, i.e. *FCM* e *KFCM* apresentaram desempenho inferior aos demais algoritmos, exceto para o índice *MPC*. Os resultados obtidos mostram que os algoritmos de co-clustering podem ser uma alternativa viável para encontrar grupos de objetos em conjuntos de dados com variáveis irrelevantes. Cenário esse que é mais compatível com as aplicações atuais do mundo real. Os resultados do teste de Nemenyi dispostos na Figura 11 mostram que a performance desses algoritmos é estatisticamente diferente da performance dos algoritmos de co-clustering, exceto para o *MPC*. Por fim, todos os algoritmos de co-clustering apresentaram resultados excelentes e estatisticamente equivalentes em relação às métricas crisp.

O conjunto de dados sintéticos 3 simula um cenário mais complicado em relação aos cenários anteriores, haja vista que dois dos três blocos de variáveis não são informativas para as classes de objetos. Como esperado, o modelo proposto baseado na distância local *GKFDK-GP* e o modelo do estado da arte *WFDK* apresentaram os melhores desempenhos em relação a todas os índices de avaliação, exceto para o *MPC*, no qual o *KFCM* obteve o melhor resultado. Em relação aos índices *FRG* e *HUL* o algoritmo *GKFDK-GP* apresentou a melhor performance. Quanto as métricas crisp, o algoritmo *WFDK* foi melhor em relação aos índices *ARI* e *FMS*, enquanto o método *GKFDK-GP* se sobressaiu em relação ao *NMI*. A Figura 12 mostra o resultado dos testes de Nemenyi para esse cenário. Em relação às métricas fuzzy *FRG* e *HUL*, os algoritmos *GKFDK-GP* e *WFDK* apresentaram os melhores resultados, de modo que o teste não indicou diferença significativa entre os desempenhos desses modelos, embora o *GKFDK-GP* tenha sido levemente melhor. O teste também indicou que o desempenho

desses algoritmos é estatisticamente superior em relação aos demais. A respeito do *MPC*, a performance do *KFCM* foi estatisticamente diferente. Quanto aos índices crisp, o teste indicou que o desempenho dos algoritmos *WFDK*, *GKFDK-GP* e *GKFDK-LP* eram estatisticamente equivalentes e diferentes do desempenho dos algoritmos que não computa o peso das variáveis. Esse resultado mostra a importância de diferenciar os pesos das variáveis, principalmente em cenários com variáveis irrelevantes.

No conjunto de dados simulados 4 as variáveis são relevantes para conjuntos diferentes de classes de objetos. Como esperado, o algoritmo proposto *GKFDK-LP* obteve os melhores resultados em relação a todas as métricas, com exceção do *MPC*, no qual o algoritmo *FCM* se sobressaiu. Em relação às métricas crisp, o algoritmo *GKFDK-GP* obteve o segundo melhor desempenho. O resultado do teste de Nemenyi mostrado na Figura 13 indica que o desempenho do *GKFDK-LP* nesse cenário é estatisticamente diferente dos demais algoritmos. Em relação ao índice *MPC* o desempenho do *FCM* foi estatisticamente diferente dos demais algoritmos. Embora o algoritmo com ponderação global *GKFDK-GP* tenha apresentado bons desempenhos, como esperado, os resultados obtidos indica a importância de considerar conjuntos distintos de variáveis relevantes para as classes de objetos, mesmo em cenários onde existe variáveis que são irrelevantes.

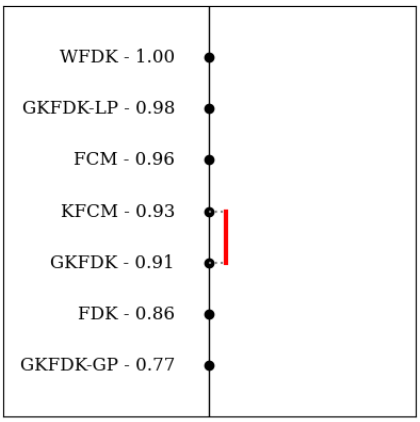
Tabela 11 – Desempenho dos algoritmos nos conjuntos de dados simulados.

Algoritmos	Conjunto de dados sintéticos 1					
	MPC	FRG	HUL	ARI	FMS	NMI
FCM	0.947 ± 0.003	0.984 ± 0.001	0.986 ± 0.001	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
KFCM	0.971 ± 0.002	0.991 ± 0.001	0.992 ± 0.001	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
FDK	0.761 ± 0.008	0.925 ± 0.003	0.925 ± 0.003	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
WFDK	0.839 ± 0.011	0.950 ± 0.003	0.951 ± 0.003	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK	0.749 ± 0.008	0.922 ± 0.003	0.918 ± 0.003	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK-GP	0.790 ± 0.009	0.934 ± 0.003	0.933 ± 0.003	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK-LP	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
Algoritmos	Conjunto de dados sintéticos 2					
	MPC	FRG	HUL	ARI	FMS	NMI
FCM	0.963 ± 0.010	0.827 ± 0.020	0.827 ± 0.020	0.445 ± 0.068	0.666 ± 0.056	0.614 ± 0.051
KFCM	0.928 ± 0.015	0.827 ± 0.019	0.825 ± 0.019	0.455 ± 0.062	0.672 ± 0.054	0.620 ± 0.048
FDK	0.865 ± 0.022	0.950 ± 0.016	0.953 ± 0.017	0.973 ± 0.068	0.987 ± 0.037	0.981 ± 0.046
WFDK	**0.998 ± 0.000**	**0.999 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK	0.905 ± 0.024	0.961 ± 0.028	0.965 ± 0.028	0.966 ± 0.093	0.980 ± 0.055	0.981 ± 0.048
GKFDK-GP	0.772 ± 0.011	0.929 ± 0.003	0.927 ± 0.004	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK-LP	0.985 ± 0.002	0.995 ± 0.001	0.996 ± 0.000	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
Algoritmos	Conjunto de dados sintéticos 3					
	MPC	FRG	HUL	ARI	FMS	NMI
FCM	0.999 ± 0.002	0.648 ± 0.016	0.648 ± 0.016	-0.076 ± 0.015	0.251 ± 0.023	0.002 ± 0.018
KFCM	**1.000 ± 0.000**	0.649 ± 0.017	0.649 ± 0.017	-0.074 ± 0.023	0.253 ± 0.032	0.005 ± 0.030
FDK	0.982 ± 0.013	0.797 ± 0.025	0.797 ± 0.025	0.349 ± 0.084	0.594 ± 0.062	0.546 ± 0.073
WFDK	0.976 ± 0.015	0.986 ± 0.021	0.986 ± 0.021	**0.975 ± 0.074**	**0.990 ± 0.032**	0.980 ± 0.057
GKFDK	0.313 ± 0.131	0.742 ± 0.010	0.645 ± 0.043	0.411 ± 0.174	0.645 ± 0.121	0.588 ± 0.124
GKFDK-GP	0.986 ± 0.030	**0.988 ± 0.030**	**0.989 ± 0.029**	0.970 ± 0.094	0.983 ± 0.057	**0.982 ± 0.056**
GKFDK-LP	0.924 ± 0.038	0.960 ± 0.035	0.961 ± 0.035	0.939 ± 0.125	0.969 ± 0.066	0.959 ± 0.084
Algoritmos	Conjunto de dados sintéticos 4					
	MPC	FRG	HUL	ARI	FMS	NMI
FCM	**1.000 ± 0.000**	0.774 ± 0.016	0.774 ± 0.016	0.287 ± 0.048	0.554 ± 0.042	0.487 ± 0.038
KFCM	0.871 ± 0.022	0.777 ± 0.014	0.772 ± 0.014	0.285 ± 0.059	0.559 ± 0.054	0.488 ± 0.047
FDK	0.993 ± 0.007	0.824 ± 0.020	0.824 ± 0.020	0.444 ± 0.064	0.661 ± 0.051	0.623 ± 0.050
WFDK	0.946 ± 0.068	0.855 ± 0.048	0.856 ± 0.049	0.573 ± 0.152	0.727 ± 0.111	0.695 ± 0.124
GKFDK	0.826 ± 0.096	0.869 ± 0.044	0.868 ± 0.048	0.650 ± 0.138	0.779 ± 0.094	0.766 ± 0.107
GKFDK-GP	0.763 ± 0.084	0.862 ± 0.017	0.865 ± 0.020	0.687 ± 0.063	0.812 ± 0.042	0.817 ± 0.047
GKFDK-LP	0.881 ± 0.018	**0.963 ± 0.006**	**0.969 ± 0.005**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**

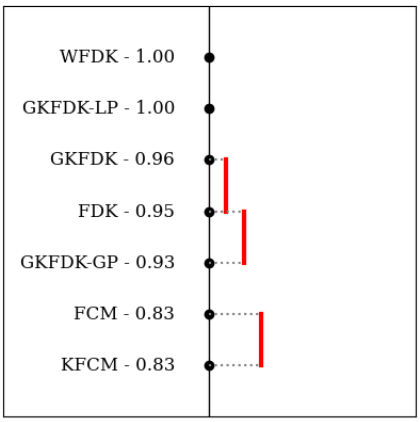
Tabela 12 – p-valor do teste de Friedman nas aplicações aos dados sintéticos.

Métrica	Simulação 1	Simulação 2	Simulação 3	Simulação 4
MPC	2.33×10^{-126}	5.56×10^{-124}	8.36×10^{-115}	5.85×10^{-106}
FRG	2.33×10^{-126}	5.62×10^{-121}	5.96×10^{-123}	4.12×10^{-97}
HUL	2.33×10^{-126}	3.33×10^{-121}	2.82×10^{-115}	2.45×10^{-98}
ARI	—	1.53×10^{-115}	9.28×10^{-121}	2.29×10^{-107}
FMS	—	1.47×10^{-115}	6.48×10^{-121}	4.99×10^{-101}
NMI	—	4.30×10^{-115}	1.53×10^{-121}	1.15×10^{-105}

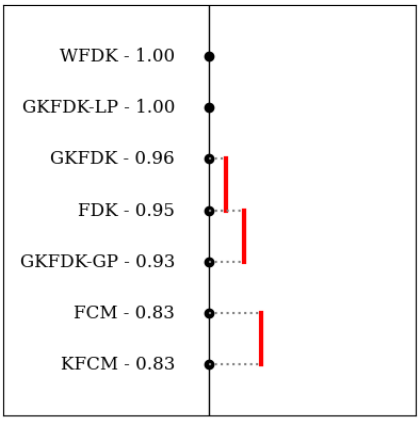
Figura 11 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 2. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas.



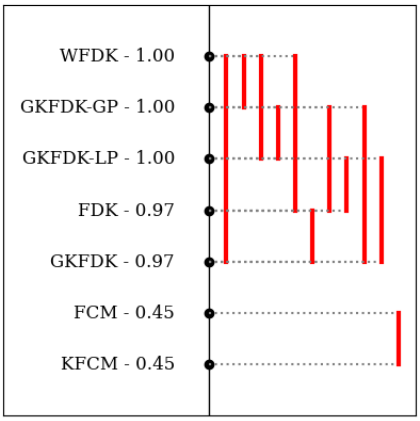
(a) *MPC*



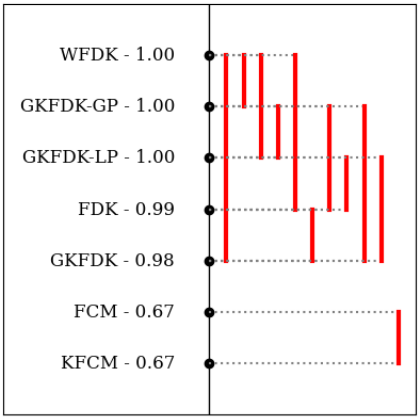
(b) *FRG*



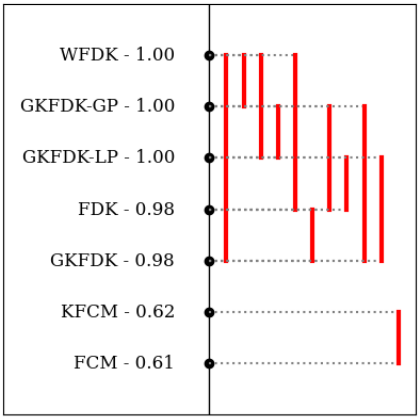
(c) *HUL*



(d) *ARI*



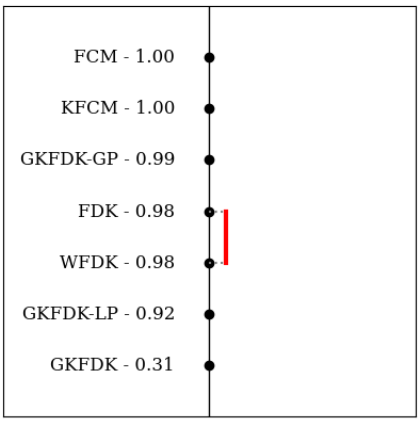
(e) *FMS*



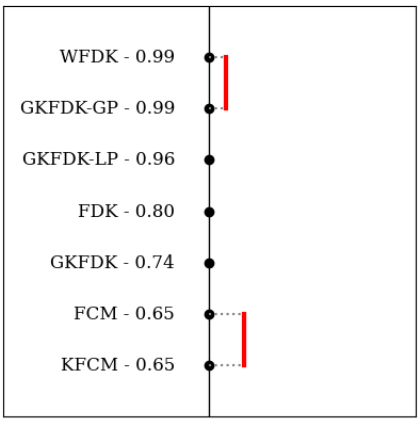
(f) *NMI*

Fonte: Autor (2023)

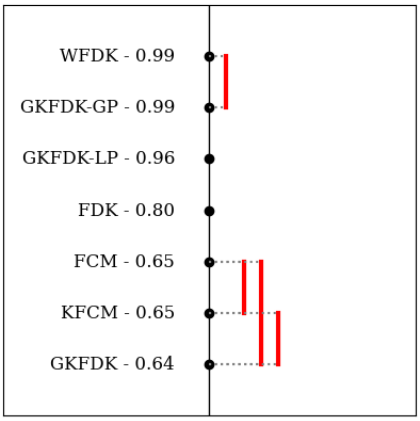
Figura 12 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 3. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas.



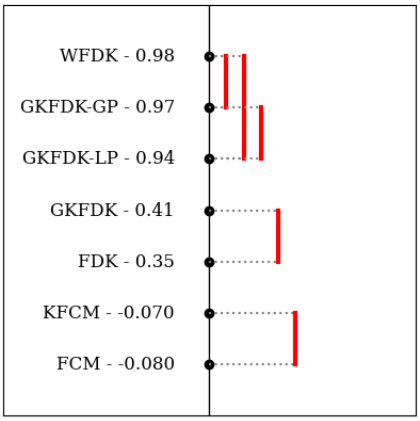
(a) *MPC*



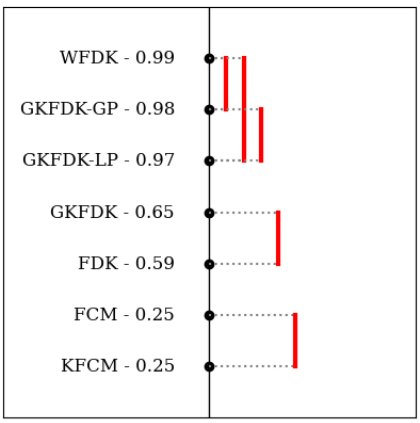
(b) *FRG*



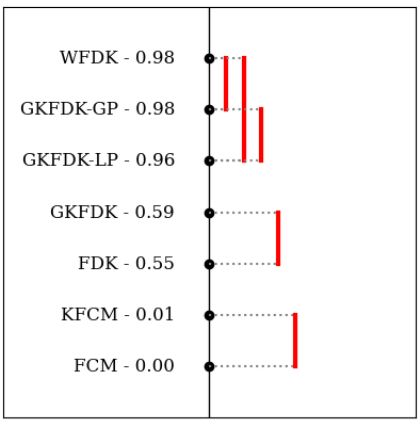
(c) *HUL*



(d) *ARI*



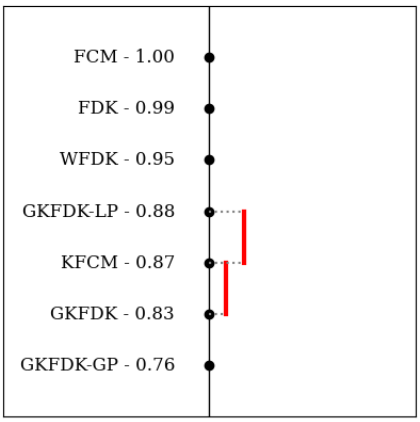
(e) *FMS*



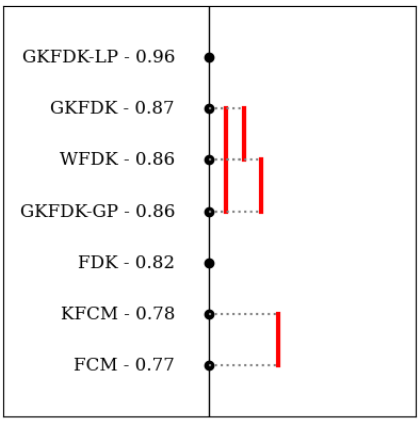
(f) *NMI*

Fonte: Autor (2023)

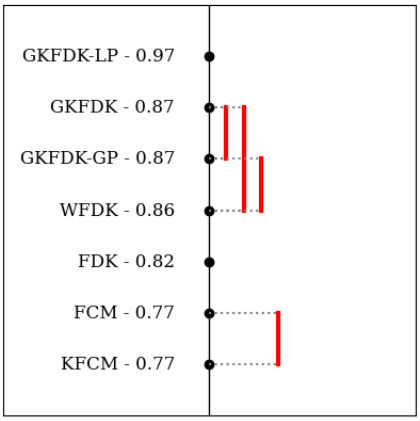
Figura 13 – Comparação dos algoritmos entre si através do teste Nemenyi no conjunto de dados sintéticos 4. Para os modelos unidos pelas linhas horizontais e pela barra vermelha, não há evidências de diferenças estatisticamente significativas.



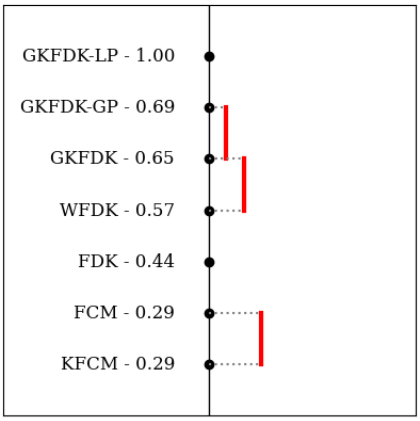
(a) *MPC*



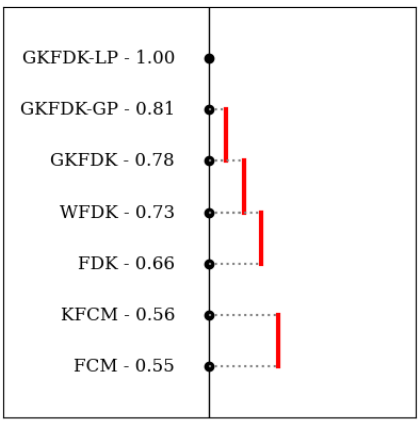
(b) *FRG*



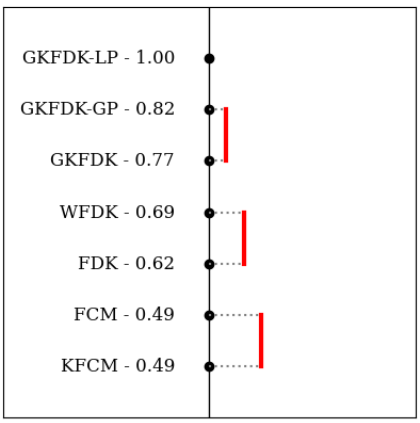
(c) *HUL*



(d) *ARI*



(e) *FMS*



(f) *NMI*

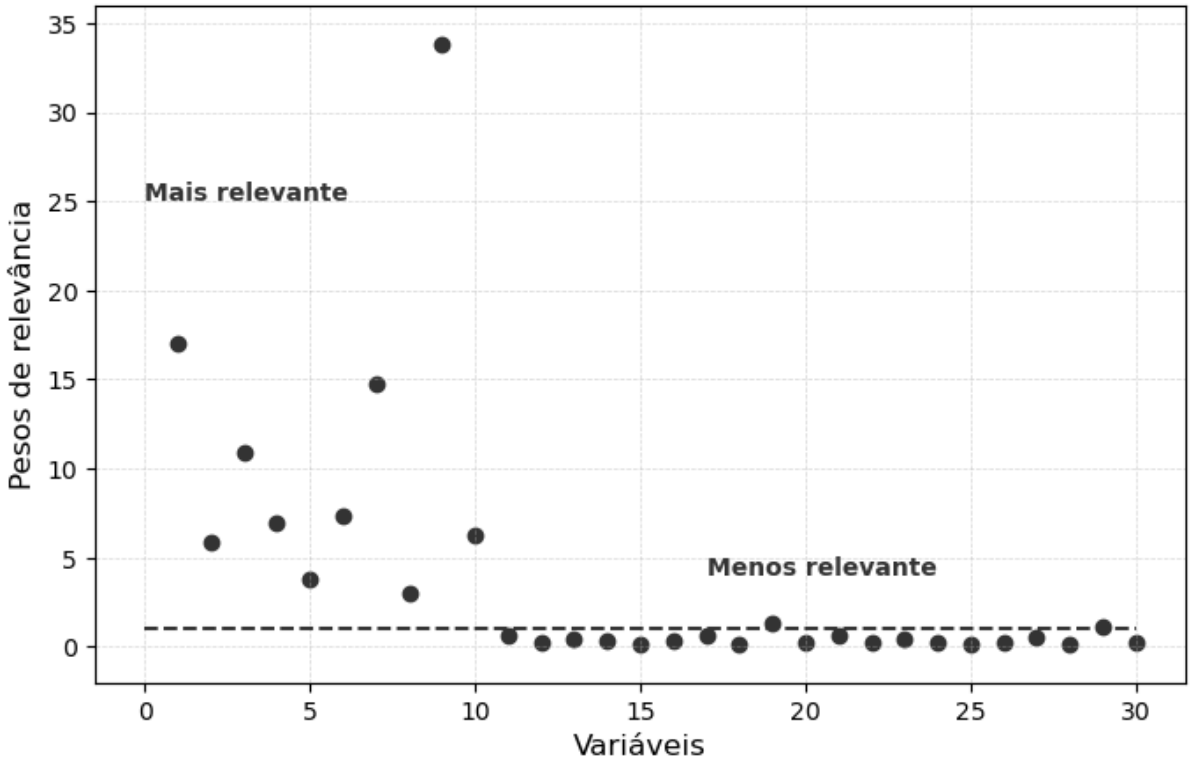
Fonte: Autor (2023)

Para analisar o comportamento dos algoritmos propostos *GKFDK-GP* e *GKFDK-LP* quanto a forma de ponderação das variáveis, nós plotamos os gráficos dos pesos de relevância fornecidos por esses algoritmos na primeira réplica de Monte Carlo nos conjuntos de dados simulados 3 e 4. A Figura 14 mostra os pesos de relevância para o conjunto de dados simulados 3, enquanto a Figura 15 mostra os pesos obtidos no conjunto de dados simulados 4.

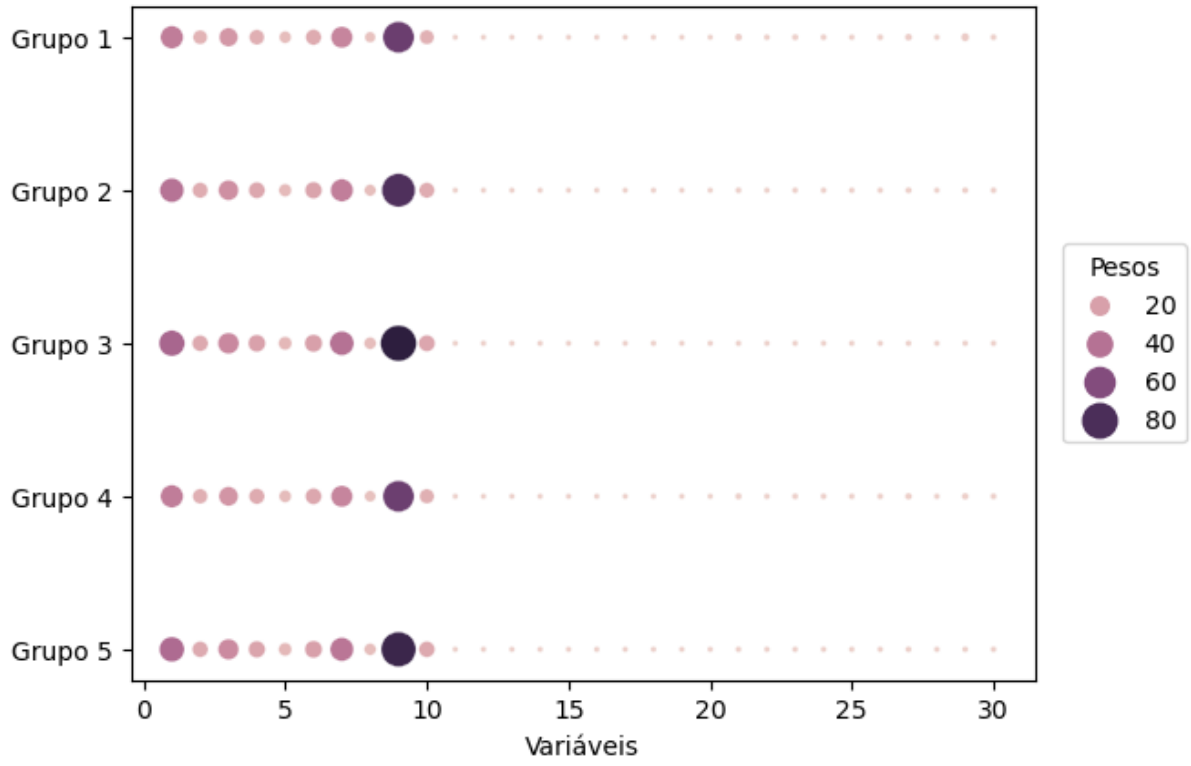
Na configuração 3 as vinte últimas variáveis não são informativas para as classes de objetos, de modo que espera-se que os pesos dessas variáveis sejam pequenos. Observe na Figura 14 (a) que o algoritmo *GKFDK-GP* detectou corretamente essas variáveis irrelevantes, pois os pesos fornecidos a elas foram menores em relação às outras variáveis. Note na Figura 14 (b) que o algoritmo *GKFDK-LP* também detectou as variáveis irrelevantes. Embora o algoritmo *GKFDK-GP* seja mais adequado nessa situação, pois fornece um único vetor de pesos, o algoritmo com ponderação local detectou em cada grupo o mesmo conjunto de variáveis relevantes, refletindo o bom desempenho desse algoritmo nesse cenário. Como dito anteriormente, o algoritmo *GKFDK-GP* pode ser considerado um caso particular do *GKFDK-LP* quando o conjunto de variáveis relevantes é o mesmo para todas as classes, sendo o que justamente acontece nesse cenário.

No cenário 4 as classes de objetos são descritas por conjuntos diferentes de variáveis relevantes. Mais especificamente, as variáveis no intervalo 1-10 são informativas apenas para duas classes de objetos, enquanto as variáveis 11-20 são informativas apenas para as outras três classes de objetos. As dez últimas variáveis são não informativas para todas as classes de objetos. Observe na Figura 15 (a) que o algoritmo *GKFDK-GP* não consegue detectar quais variáveis são relevantes para cada grupo de objetos, fornecendo pesos de relevâncias que não retrata as características dos dados. Até mesmo as variáveis irrelevantes o algoritmo não consegue identificar. Isso acontece, pois o conjunto de variáveis relevantes não é o mesmo para todas as classes. Por outro lado, note na Figura 15 (b) que o algoritmo *GKFDK-LP* identificou corretamente o conjunto de variáveis relevantes para cada classe de objetos. Note que as variáveis 1-10 foram relevantes para os grupos 1 e 4 de objetos, enquanto as variáveis 11-20 foram relevantes para os grupos 2, 3 e 5 de objetos. Além disso, esse método conseguiu detectar quais variáveis eram irrelevantes, atribuindo pesos pequenos para elas. Esses resultados são consistentes com a construção do conjunto de dados simulados 4.

Figura 14 – Pesos de relevância obtidos no conjunto de dados simulados 3 usando os algoritmos *GKFDK-GP* e *GKFDK-LP*.



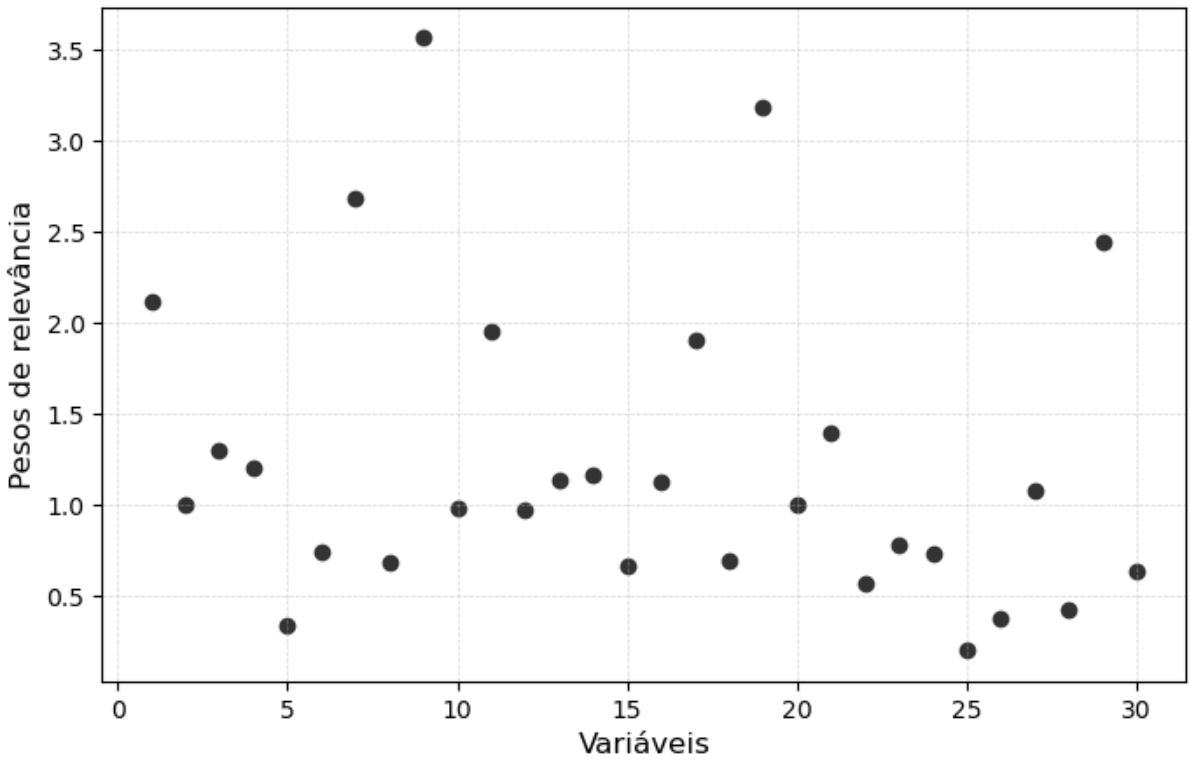
(a) *GKFDK-GP*



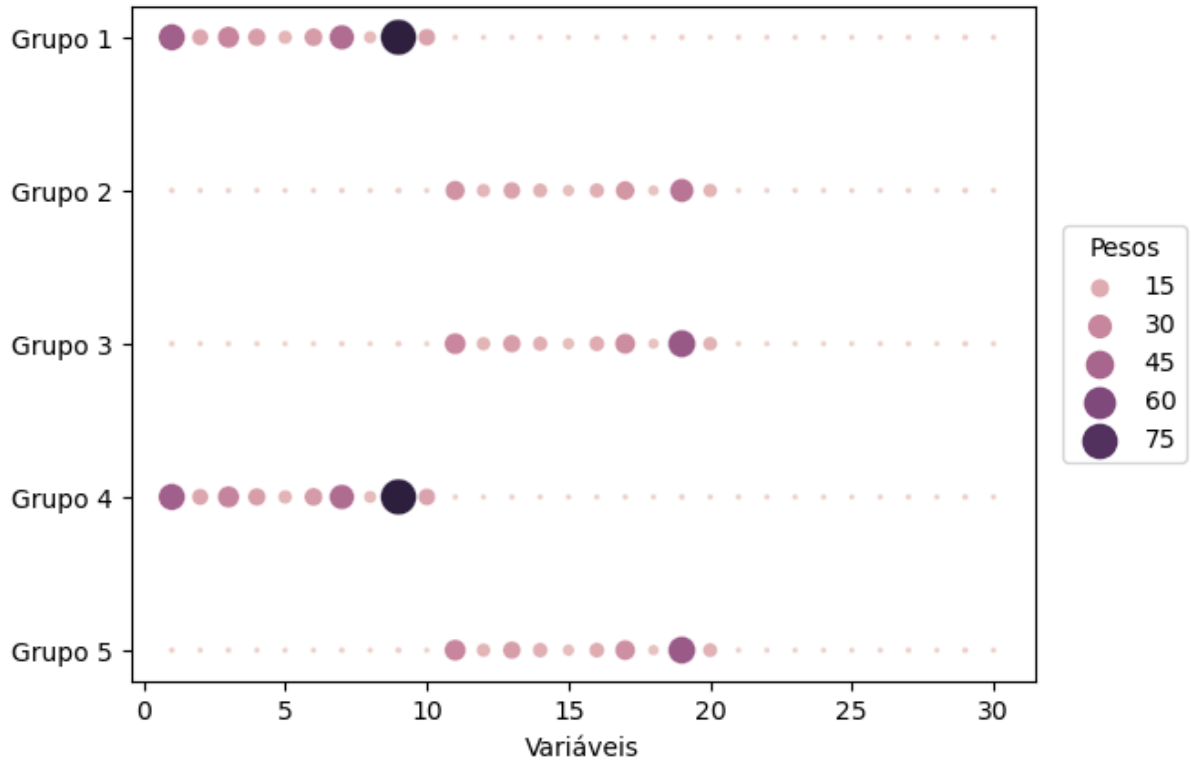
(b) *GKFDK-LP*

Fonte: Autor (2023)

Figura 15 – Pesos de relevância obtidos no conjunto de dados simulados 4 usando os algoritmos *GKFDK-GP* e *GKFDK-LP*.



(a) *GKFDK-GP*



(b) *GKFDK-LP*

Fonte: Autor (2023)

5.3 APLICAÇÕES COM DADOS REAIS

Os algoritmos propostos e os algoritmos do estado da arte também foram aplicados a 14 conjuntos de dados reais, com diferentes números de objetos (N), de variáveis (P) e de classes (c). Nove conjunto de dados foram obtidos do UCI Machine Learning Repository (DUA; GRAFF, 2017), dois do scikit-feature (Feature Selection Repository) (LI et al., 2018), dois do trabalho de Laclau e Nadif (LACLAU; NADIF, 2016) e um conjunto de dados obtido do Kaggle, disponível em <https://www.kaggle.com/datasets/mssmartypants/paris-housing-classification>. A Tabela 13 descreve brevemente os conjuntos de dados reais usados para análise comparativa.

Tabela 13 – Descrição dos conjuntos de dados reais.

Conjunto de dados	N	P	c	Fonte
Algerian Forest	244	10	2	UCI
Breast Cancer	569	30	2	UCI
Cervical Cancer	72	20	2	UCI
Chemical Composition	88	17	2	UCI
Image Segmentation	2310	19	7	UCI
Vehicle	846	18	4	UCI
Paris Housing	10000	17	2	Kaggle
Glioma	50	4434	4	scikit-feature
WebKB4	4199	1000	4	(LACLAU; NADIF, 2016)
Yale	165	1024	15	scikit-feature
CSTR	475	1000	4	(LACLAU; NADIF, 2016)
Vertebral Column	310	6	2	UCI
Breast Tissue	106	9	6	UCI
Abalone	4177	8	3	UCI

A escolha dos parâmetros de cada algoritmo foi feita segundo o procedimento descrito na Seção 5.1.2. A Tabela 14 mostra os valores desses parâmetros para cada conjunto de dados real.

Nós definimos o número de grupos de objetos como sendo o número de classes a priori (c), e definimos o número de grupos de variáveis como sendo o mesmo número de grupos de objetos, i.e. $K = H$. Como em aplicações reais, na grande maioria das vezes, apenas as classes a priori dos objetos são conhecidas, em nossas aplicações calculamos as métricas considerando apenas as partições relacionadas aos objetos. Além disso, todos os conjuntos de dados foram normalizados baseado na média e desvio padrão de cada variável. Para cada conjunto de dados reais, todos os algoritmos foram executados 50 vezes com diferentes inicializações e foi feito dois tipos de análises: análise do melhor caso, descrito na Seção 5.3.1, e análise do caso médio, descrito na Seção 5.3.2.

Tabela 14 – Valores dos parâmetros dos algoritmos usados nas aplicações aos conjuntos de dados reais.

Algoritmo	Algerian Forest	Breast Cancer	Cervical Cancer	Chemical Composition	Image Segmentation
FCM (m)	1.2	1.1	1.5	1.01	1.1
KFCM (m; σ^2)	(1.2; 124.78)	(1.1; 347.44)	(1.01; 2.0)	(1.1; 1.76)	(1.01; 1.98)
FDK (m; n)	(1.2; 1.1)	(1.1; 1.01)	(1.01; 1.1)	(1.01; 1.01)	(1.01; 1.01)
WFDK (m; n; γ)	(1.5; 1.01; 3.0)	(1.1; 1.01; 1.5)	(1.01; 1.2; 2.5)	(1.01; 1.001; 2.0)	(1.01; 1.01; 1.5)
GKFDK (m; n; σ^2)	(1.2; 1.2; 1.291)	(1.01; 1.01; 12.82)	(1.3; 1.1; 1.333)	(1.01; 1.01; 13.52)	(1.01; 1.01; 0.633)
GKFDK-GP (m; n; σ^2)	(1.01; 1.01; 1.291)	(1.1; 1.01; 12.82)	(1.1; 1.3; 1.333)	(1.01; 1.01; 1.352)	(1.01; 1.01; 6.33)
GKFDK-LP (m; n; σ^2)	(1.3; 1.01; 1.291)	(1.1; 1.01; 12.82)	(1.01; 1.01; 13.33)	(1.01; 1.01; 1.352)	(1.01; 1.01; 0.633)

Algoritmo	Vehicle	Paris Housing	Glioma	WebKB4	Yale
FCM (m)	1.01	1.1	1.1	1.01	1.01
KFCM (m; σ^2)	(1.1; 202.06)	(1.1; 172.1)	(1.5; 483.95)	(1.01; 10595.48)	(1.003; 1095.96)
FDK (m; n)	(1.01; 1.1)	(1.1; 1.1)	(1.5; 1.3)	(1.01; 1.003)	(1.01; 1.01)
WFDK (m; n; γ)	(1.01; 1.01; 1.5)	(1.1; 1.1; 1.5)	(1.1; 1.1; 2.0)	(1.01; 1.005; 1.5)	(1.01; 1.003; 1.5)
GKFDK (m; n; σ^2)	(1.01; 1.01; 0.128)	(1.4; 1.1; 0.136)	(1.4; 1.1; 13.31)	(1.01; 1.003; 8.66)	(1.003; 1.01; 13.54)
GKFDK-GP (m; n; σ^2)	(1.1; 1.01; 12.87)	(1.4; 1.1; 0.136)	(1.1; 1.1; 133.1)	(1.005; 1.005; 8.66)	(1.01; 1.005; 0.135)
GKFDK-LP (m; n; σ^2)	(1.1; 1.01; 0.128)	(1.3; 1.1; 0.136)	(1.5; 1.1; 1.331)	(1.01; 1.005; 0.866)	(1.01; 1.01; 0.135)

Algoritmo	CSTR	Vertebral Column	Breast Tissue	Abalone	–
FCM (m)	1.001	1.01	1.01	1.3	–
KFCM (m; σ^2)	(1.003; 104.64)	(1.1; 66.6)	(1.01; 102.6)	(1.3; 102.1)	–
FDK (m; n)	(1.001; 1.001)	(1.7; 1.01)	(1.01; 1.1)	(1.1; 1.01)	–
WFDK (m; n; γ)	(1.01; 1.001; 1.1)	(1.1; 1.01; 2.5)	(1.7; 1.01; 2.5)	(1.01; 1.01; 1.5)	–
GKFDK (m; n; σ^2)	(1.003; 1.001; 0.1467)	(1.1; 1.01; 1.266)	(1.01; 1.1; 13.06)	(1.5; 1.01; 0.13)	–
GKFDK-GP (m; n; σ^2)	(1.003; 1.001; 0.1467)	(1.7; 1.01; 12.66)	(1.01; 1.1; 13.06)	(1.1; 1.01; 0.13)	–
GKFDK-LP (m; n; σ^2)	(1.01; 1.003; 0.0146)	(1.1; 1.7; 0.126)	(1.3; 1.01; 0.13)	(1.01; 1.01; 0.13)	–

Para ter uma noção intuitiva das características da atribuição dos pesos nos dados reais, plotamos a distribuição das 10 variáveis que descrevem o conjunto de dados Algerian Forest na Figura 16. Note que nas variáveis 1, 2 e 3 as classes de objetos não estão bem definidas e, principalmente, estão bem dispersas, enquanto que nas outras 7 variáveis as classes de objetos estão melhor distribuídas e uma delas está menos dispersa. Nas variáveis 4 e 5 a classe de objetos com a cor laranja apresenta menor dispersão, enquanto a classe descrita pelos pontos azuis está mais dispersa. Nas variáveis 6, 7, 8, 9 e 10 é observado exatamente o oposto. Nesse sentido, as variáveis 4 e 5 devem contribuir mais para a construção de um grupo, enquanto as cinco últimas variáveis devem ser importantes para a formação do outro grupo. Além disso, as três primeiras variáveis não devem contribuir muito no processo de agrupamento, de modo que pesos pequenos devem ser atribuídos para elas.

A Tabela 15 mostra os pesos de relevância obtidos no conjunto de dados Algerian Forest através dos algoritmos *GKFDK-GP* e *GKFDK-LP*. Os valores correspondem ao melhor resultado segundo o critério de adequação nas 50 execuções. Note que os algoritmos atribuíram pesos pequenos às três primeiras variáveis. O algoritmo *GKFDK-GP* atribuiu maiores pesos às últimas cinco variáveis, sendo pertinente com as características do conjunto de dados. Além disso, o algoritmo *GKFDK-LP* identificou conjuntos diferentes de variáveis relevantes para cada grupo. Para o Grupo 1 as variáveis 4 e 5 foram mais importantes, enquanto que para o Grupo 2 as variáveis 6,7,8,9 e 10 foram mais relevantes. Resultado este condizente com as características do conjunto de dados, discutido logo acima.

A Figura 17 mostra o comportamento da função objetivo nessa aplicação ao conjunto de dados Algerian Forest. Embora o critério de adequação dos algoritmos propostos não convirjam monotonicamente, pelo fato de que nas etapas de alocação dos objetos e variáveis as equações utilizadas serem heurísticas, o valor da função objetivo decresce para um valor estacionário. Note na Figura 17 (a) que o valor de $\mathcal{J}$ decresce e tem uma leve subida, mas se mantém estacionário até o critério de parada ser satisfeito.

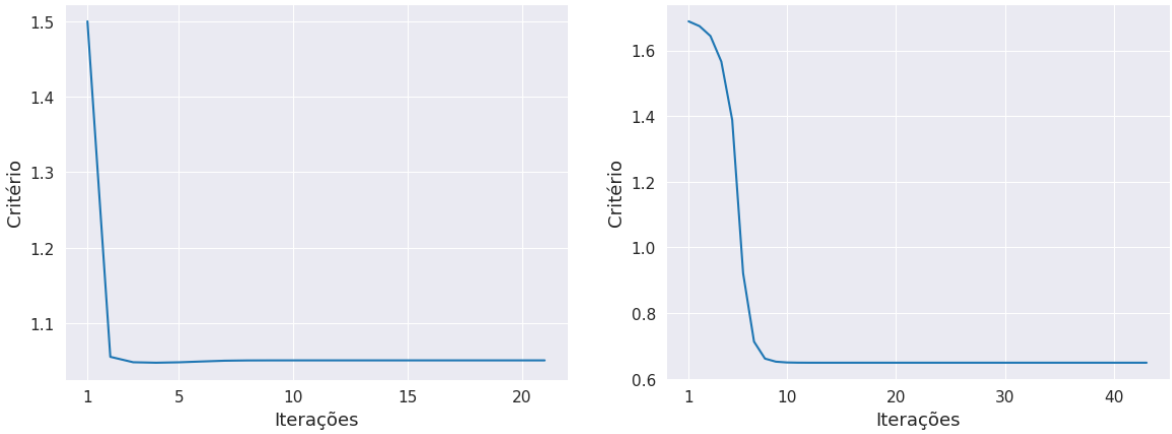
Figura 16 – Distribuição das variáveis no conjunto de dados Algerian Forest.

Fonte: Autor (2023)

Tabela 15 – Pesos de relevância obtidos no conjunto de dados Algerian Forest através dos algoritmos *GKFDK-GP* e *GKFDK-LP*.

GKFDK-GP	Variáveis									
	1	2	3	4	5	6	7	8	9	10
	0.98	0.32	0.34	0.98	**1.27**	**1.58**	**1.25**	**1.48**	**1.53**	**1.66**
GKFDK-LP	Variáveis									
	1	2	3	4	5	6	7	8	9	10
Grupo 1	0.79	0.25	0.27	**42.88**	**5.29**	0.62	0.52	0.67	0.60	0.65
Grupo 2	0.41	0.16	0.15	0.22	0.38	**2.90**	**2.77**	**4.47**	**3.16**	**10.74**

Figura 17 – Valores do critério de adequação $\mathcal{J}$ obtidos no conjunto de dados Algerian Forest através dos modelos propostos com ponderação das variáveis.



(a) *GKFDK-GP* (b) *GKFDK-LP*

Fonte: Autor (2023)

5.3.1 Análise do melhor caso

Na análise do melhor caso, o objetivo é replicar uma situação mais prática, na qual apenas uma execução do algoritmo é utilizada. Aqui, a melhor inicialização, entre as 50, é aquela que fornece o melhor ajuste entre dados e protótipos, ou seja, cujo valor da função objetivo é mínimo.

As Tabelas 16 e 17 mostram os melhores resultados (de acordo com a função objetivo) em relação às métricas fuzzy e crisp, respectivamente. Em ambas as tabelas, os valores em parênteses referem-se ao ranking de desempenho de cada algoritmo segundo os índices de avaliação em cada conjunto de dados.

A Tabela 18 apresenta o ranking médio de desempenho considerando todos os conjuntos de dados e foi construída a partir das Tabelas 16 e 17. Os valores em parênteses mostram o

ranking de performance conforme o ranking médio de desempenho. Por fim, nessa tabela é mostrado o *Overall Average Ranking* (*OAR*), uma medida global que leva em consideração o ranking médio de desempenho de acordo com todas as métricas.

Em relação às métricas fuzzy (Tabela 16), que avalia a qualidade da partição fuzzy fornecida pelos algoritmos, os modelos propostos apresentaram melhores desempenhos em comparação aos algoritmos da literatura com respeito aos índices FRG e HUL. Quanto ao índice MPC, os algoritmos propostos apresentaram bons desempenhos na grande maioria dos conjuntos de dados, mas não forneceram os melhores resultados. Com respeito as métricas crisp (Tabela 17), que avalia uma partição hard obtida a partir da partição fuzzy fornecida pelos algoritmos, os modelos propostos apresentam os melhores resultados, com destaque para os algoritmos com ponderação automática das variáveis, *GKFDK-LP* e *GKFDK-GP*.

Analisando o ranking dos algoritmos (Tabela 18) observamos que o algoritmo proposto *GKFDK-LP* apresentou o melhor desempenho em relação a todos os índices de avaliação, com exceção do MPC, no qual o algoritmo do estado da arte *WFDK* se sobressaiu em relação aos demais métodos. Destaque também para o modelo proposto *GKFDK-GP* que apresentou o segundo melhor desempenho, com exceção do índice MPC. O modelo proposto sem ponderação *GKFDK* apresentou o terceiro melhor ranking médio, atrás apenas dos outros algoritmos propostos, em relação às métricas FRG, HUL, ARI e FMS. Olhando o para o índice *Overall Average Ranking* (*OAR*), notamos que os algoritmos propostos apresentaram os melhores rankings, superando os algoritmos do estado da arte. O método *GKFDK-LP* obteve o melhor ranking médio de desempenho, seguido do *GKFDK-GP* e *GKFDK*.

Tabela 16 – Métricas de avaliação fuzzy obtidas nas aplicações aos dados reais. O valor na tabela representa o resultado do melhor caso entre as 50 repetições, de acordo com a função objetivo. O valor entre parênteses refere-se ao ranking entre os algoritmos.

Algoritmos	Algerian Forest			Algoritmos	Breast Cancer		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.895 (3)	0.677 (5)	0.679 (5)	FCM	0.949 (5)	0.837 (5)	0.837 (5)
KFCM	0.891 (4)	0.676 (6)	0.678 (6)	KFCM	0.948 (6)	0.839 (4)	0.839 (4)
FDK	0.888 (5)	0.687 (4)	0.690 (4)	FDK	0.943 (7)	0.823 (6)	0.823 (6)
WFDK	0.775 (6)	0.619 (7)	0.629 (7)	WFDK	0.979 (2)	0.806 (7)	0.806 (7)
GKFDK	0.769 (7)	0.730 (3)	0.742 (3)	GKFDK	**0.994 (1)**	0.840 (3)	0.840 (3)
GKFDK-GP	**0.999 (1)**	0.759 (2)	0.759 (2)	GKFDK-GP	0.955 (4)	0.851 (2)	0.851 (2)
GKFDK-LP	0.917 (2)	**0.848 (1)**	**0.848 (1)**	GKFDK-LP	0.962 (3)	**0.868 (1)**	**0.868 (1)**
Algoritmos	Cervical Cancer			Algoritmos	Chemical Composition		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.398 (7)	0.554 (6)	0.586 (6)	FCM	**1.000 (1)**	**1.000 (1)**	**1.000 (1)**
KFCM	0.442 (6)	0.497 (7)	0.495 (7)	KFCM	0.331 (7)	0.580 (7)	0.586 (7)
FDK	**1.000 (1)**	0.649 (3)	0.649 (3)	FDK	**1.000 (1)**	0.796 (4)	0.796 (4)
WFDK	**1.000 (1)**	0.634 (4)	0.634 (4)	WFDK	**1.000 (1)**	0.745 (6)	0.745 (6)
GKFDK	0.531 (5)	0.604 (5)	0.628 (5)	GKFDK	**1.000 (1)**	0.796 (4)	0.796 (4)
GKFDK-GP	0.930 (4)	0.652 (2)	0.652 (2)	GKFDK-GP	**1.000 (1)**	**1.000 (1)**	**1.000 (1)**
GKFDK-LP	**1.000 (1)**	**0.665 (1)**	**0.665 (1)**	GKFDK-LP	**1.000 (1)**	0.977 (3)	0.977 (3)
Algoritmos	Image Segmentation			Algoritmos	Vehicle		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.945 (6)	0.885 (3)	0.885 (3)	FCM	0.996 (2)	0.616 (7)	0.616 (7)
KFCM	0.895 (7)	0.890 (2)	0.887 (2)	KFCM	0.916 (7)	0.650 (4)	0.648 (4)
FDK	0.995 (4)	0.858 (5)	0.858 (5)	FDK	0.995 (3)	0.640 (6)	0.640 (6)
WFDK	**0.999 (1)**	0.806 (7)	0.806 (7)	WFDK	**0.999 (1)**	0.652 (3)	0.652 (3)
GKFDK	0.984 (5)	**0.914 (1)**	**0.914 (1)**	GKFDK	0.994 (4)	0.655 (2)	0.655 (2)
GKFDK-GP	0.997 (3)	0.863 (4)	0.863 (4)	GKFDK-GP	0.960 (5)	0.648 (5)	0.648 (4)
GKFDK-LP	**0.999 (1)**	0.838 (6)	0.838 (6)	GKFDK-LP	0.954 (6)	**0.677 (1)**	**0.677 (1)**
Algoritmos	Paris Housing			Algoritmos	Glioma		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.996 (2)	0.616 (7)	0.616 (7)	FCM	**0.974 (1)**	0.732 (2)	**0.732 (1)**
KFCM	0.916 (7)	0.650 (4)	0.648 (4)	KFCM	0.080 (7)	0.626 (7)	0.313 (7)
FDK	0.995 (3)	0.640 (6)	0.640 (6)	FDK	0.162 (6)	0.676 (5)	0.575 (5)
WFDK	**0.999 (1)**	0.652 (3)	0.652 (3)	WFDK	0.855 (2)	0.731 (3)	0.730 (2)
GKFDK	0.994 (4)	0.655 (2)	0.655 (2)	GKFDK	0.206 (4)	0.690 (4)	0.617 (4)
GKFDK-GP	0.960 (5)	0.648 (5)	0.648 (4)	GKFDK-GP	0.768 (3)	**0.735 (1)**	0.729 (3)
GKFDK-LP	0.954 (6)	**0.677 (1)**	**0.677 (1)**	GKFDK-LP	0.164 (5)	0.676 (5)	0.574 (6)
Algoritmos	WebKB4			Algoritmos	Yale		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.820 (4)	0.586 (3)	0.579 (3)	FCM	**1.000 (1)**	0.901 (4)	0.901 (4)
KFCM	0.904 (2)	0.584 (4)	0.581 (2)	KFCM	**1.000 (1)**	0.903 (3)	0.903 (3)
FDK	0.647 (6)	0.520 (7)	0.488 (6)	FDK	0.999 (5)	0.907 (2)	0.907 (2)
WFDK	0.833 (3)	0.567 (5)	0.558 (4)	WFDK	**1.000 (1)**	0.893 (6)	0.893 (6)
GKFDK	0.708 (5)	0.567 (5)	0.541 (5)	GKFDK	**1.000 (1)**	0.899 (5)	0.899 (5)
GKFDK-GP	0.170 (7)	0.599 (2)	0.414 (7)	GKFDK-GP	0.978 (7)	**0.915 (1)**	**0.915 (1)**
GKFDK-LP	**0.996 (1)**	**0.711 (1)**	**0.711 (1)**	GKFDK-LP	0.999 (5)	0.888 (7)	0.888 (7)
Algoritmos	CSTR			Algoritmos	Vertebral Column		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.110 (4)	0.612 (4)	0.419 (5)	FCM	**0.995 (1)**	0.552 (4)	0.552 (6)
KFCM	0.443 (2)	0.611 (7)	0.529 (2)	KFCM	0.941 (3)	0.557 (3)	0.557 (3)
FDK	0.109 (5)	0.612 (4)	0.422 (4)	FDK	0.543 (7)	0.533 (7)	0.557 (3)
WFDK	0.231 (3)	0.616 (2)	0.464 (3)	WFDK	0.960 (2)	0.544 (5)	0.544 (7)
GKFDK	0.001 (7)	0.612 (4)	0.276 (7)	GKFDK	0.892 (5)	0.563 (2)	0.565 (2)
GKFDK-GP	0.047 (6)	0.613 (3)	0.316 (6)	GKFDK-GP	0.579 (6)	0.535 (6)	0.556 (5)
GKFDK-LP	**1.000 (1)**	**0.816 (1)**	**0.816 (1)**	GKFDK-LP	0.924 (4)	**0.627 (1)**	**0.627 (1)**
Algoritmos	Breast Tissue			Algoritmos	Abalone		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	**1.000 (1)**	0.727 (7)	0.727 (7)	FCM	0.868 (5)	0.611 (4)	0.608 (5)
KFCM	**1.000 (1)**	0.728 (6)	0.728 (6)	KFCM	0.866 (6)	0.611 (4)	0.608 (5)
FDK	**1.000 (1)**	0.775 (4)	0.775 (4)	FDK	0.956 (3)	0.612 (3)	0.612 (3)
WFDK	0.816 (6)	0.743 (5)	0.735 (5)	WFDK	**0.998 (1)**	0.611 (4)	0.611 (4)
GKFDK	**1.000 (1)**	0.783 (3)	0.783 (2)	GKFDK	0.453 (7)	0.580 (7)	0.536 (7)
GKFDK-GP	**1.000 (1)**	**0.813 (1)**	**0.813 (1)**	GKFDK-GP	0.895 (4)	0.617 (2)	0.616 (2)
GKFDK-LP	0.748 (7)	0.788 (2)	0.781 (3)	GKFDK-LP	0.992 (2)	**0.623 (1)**	**0.623 (1)**

Tabela 17 – Métricas de avaliação crisp obtidas nas aplicações aos dados reais. O valor na tabela representa o resultado do melhor caso entre as 50 repetições, de acordo com a função objetivo. O valor entre parênteses refere-se ao ranking entre os algoritmos.

Algoritmos	Algerian Forest			Algoritmos	Breast Cancer		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.326 (5)	0.785 (5)	0.332 (5)	FCM	0.671 (5)	0.909 (5)	0.555 (6)
KFCM	0.326 (5)	0.785 (5)	0.332 (5)	KFCM	0.683 (3)	0.913 (3)	0.566 (3)
FDK	0.365 (4)	0.803 (4)	0.358 (4)	FDK	0.659 (6)	0.905 (6)	0.550 (7)
WFDK	0.254 (7)	0.748 (7)	0.322 (7)	WFDK	0.611 (7)	0.888 (7)	0.561 (5)
GKFDK	0.567 (2)	0.878 (2)	0.506 (2)	GKFDK	0.676 (4)	0.911 (4)	0.564 (4)
GKFDK-GP	0.518 (3)	0.861 (3)	0.462 (3)	GKFDK-GP	0.712 (2)	0.921 (2)	0.612 (2)
GKFDK-LP	**0.726 (1)**	**0.926 (1)**	**0.623 (1)**	GKFDK-LP	**0.748 (1)**	**0.932 (1)**	**0.671 (1)**
Algoritmos	Cervical Cancer			Algoritmos	Chemical Composition		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.213 (6)	0.748 (6)	0.203 (7)	FCM	**1.000 (1)**	**1.000 (1)**	**1.000 (1)**
KFCM	0.160 (7)	0.719 (7)	**0.311 (1)**	KFCM	0.955 (3)	0.989 (3)	0.922 (3)
FDK	0.296 (3)	0.786 (3)	0.228 (4)	FDK	0.593 (5)	0.885 (5)	0.587 (5)
WFDK	0.267 (5)	0.773 (5)	0.211 (6)	WFDK	0.491 (7)	0.849 (7)	0.515 (7)
GKFDK	**0.362 (1)**	**0.813 (1)**	0.290 (2)	GKFDK	0.593 (5)	0.885 (5)	0.587 (5)
GKFDK-GP	0.296 (3)	0.786 (3)	0.228 (4)	GKFDK-GP	**1.000 (1)**	**1.000 (1)**	**1.000 (1)**
GKFDK-LP	0.329 (2)	0.800 (2)	0.270 (3)	GKFDK-LP	0.955 (3)	0.989 (3)	0.922 (3)
Algoritmos	Image Segmentation			Algoritmos	Vehicle		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.601 (3)	0.736 (2)	0.697 (3)	FCM	0.076 (5)	0.420 (5)	0.117 (5)
KFCM	0.619 (2)	0.735 (3)	**0.724 (1)**	KFCM	0.067 (7)	0.407 (6)	0.089 (7)
FDK	0.496 (4)	0.653 (4)	0.635 (4)	FDK	0.071 (6)	0.402 (7)	0.098 (6)
WFDK	0.291 (7)	0.484 (7)	0.475 (7)	WFDK	0.122 (3)	0.449 (4)	0.185 (2)
GKFDK	**0.648 (1)**	**0.798 (1)**	0.702 (2)	GKFDK	0.125 (2)	0.456 (2)	**0.187 (1)**
GKFDK-GP	0.473 (5)	0.625 (5)	0.616 (5)	GKFDK-GP	0.114 (4)	0.452 (3)	0.172 (3)
GKFDK-LP	0.405 (6)	0.604 (6)	0.548 (6)	GKFDK-LP	**0.145 (1)**	**0.504 (1)**	0.172 (3)
Algoritmos	Paris Housing			Algoritmos	Glioma		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.000 (6)	0.585 (6)	0.000 (6)	FCM	0.328 (3)	0.606 (2)	0.500 (3)
KFCM	0.000 (6)	0.585 (6)	0.000 (6)	KFCM	0.335 (2)	0.612 (1)	0.476 (5)
FDK	0.015 (4)	0.632 (4)	0.037 (4)	FDK	0.304 (4)	0.587 (4)	0.503 (2)
WFDK	0.014 (5)	0.630 (5)	0.034 (5)	WFDK	0.284 (7)	0.545 (7)	0.429 (7)
GKFDK	0.029 (3)	0.653 (3)	0.078 (3)	GKFDK	0.289 (6)	0.574 (6)	0.475 (6)
GKFDK-GP	0.043 (2)	0.668 (2)	**0.120 (1)**	GKFDK-GP	**0.338 (1)**	0.589 (3)	0.484 (4)
GKFDK-LP	**0.044 (1)**	**0.669 (1)**	0.117 (2)	GKFDK-LP	0.302 (5)	0.578 (5)	**0.507 (1)**
Algoritmos	WebKB4			Algoritmos	Yale		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.094 (2)	0.469 (2)	0.167 (3)	FCM	0.272 (4)	0.522 (5)	0.538 (5)
KFCM	0.085 (3)	0.458 (3)	0.183 (2)	KFCM	0.267 (5)	0.501 (6)	0.527 (6)
FDK	0.026 (6)	0.400 (4)	0.013 (6)	FDK	0.301 (2)	0.543 (3)	0.562 (3)
WFDK	0.042 (4)	0.388 (6)	0.048 (4)	WFDK	0.093 (7)	0.345 (7)	0.376 (7)
GKFDK	0.041 (5)	0.398 (5)	0.046 (5)	GKFDK	0.285 (3)	0.525 (4)	0.549 (4)
GKFDK-GP	0.012 (7)	0.361 (7)	0.005 (7)	GKFDK-GP	**0.357 (1)**	**0.641 (1)**	**0.617 (1)**
GKFDK-LP	**0.315 (1)**	**0.626 (1)**	**0.388 (1)**	GKFDK-LP	0.265 (6)	0.552 (2)	0.564 (2)
Algoritmos	CSTR			Algoritmos	Vertebral Column		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	-0.001 (7)	0.293 (7)	0.006 (7)	FCM	0.102 (6)	0.672 (6)	0.161 (4)
KFCM	0.359 (2)	0.608 (2)	0.373 (2)	KFCM	0.112 (4)	0.679 (4)	0.160 (5)
FDK	0.001 (6)	0.302 (6)	0.009 (6)	FDK	0.111 (5)	0.679 (4)	0.166 (3)
WFDK	0.047 (5)	0.403 (5)	0.052 (5)	WFDK	0.090 (7)	0.663 (7)	0.141 (7)
GKFDK	0.282 (3)	0.588 (3)	0.278 (3)	GKFDK	0.122 (2)	0.687 (2)	0.144 (6)
GKFDK-GP	0.219 (4)	0.556 (4)	0.204 (4)	GKFDK-GP	0.116 (3)	0.682 (3)	0.169 (2)
GKFDK-LP	**0.563 (1)**	**0.746 (1)**	**0.513 (1)**	GKFDK-LP	**0.251 (1)**	**0.760 (1)**	**0.214 (1)**
Algoritmos	Breast Tissue			Algoritmos	Abalone		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.297 (3)	0.61 (2)	0.566 (2)	FCM	0.134 (7)	0.517 (6)	0.163 (5)
KFCM	0.288 (4)	0.585 (3)	0.549 (3)	KFCM	0.137 (4)	0.521 (4)	0.164 (3)
FDK	0.286 (5)	0.545 (6)	0.507 (6)	FDK	0.136 (5)	0.518 (5)	0.163 (5)
WFDK	0.286 (5)	0.556 (5)	0.517 (5)	WFDK	0.135 (6)	0.515 (7)	0.156 (7)
GKFDK	0.244 (7)	0.521 (7)	0.446 (7)	GKFDK	**0.164 (1)**	0.563 (2)	0.164 (3)
GKFDK-GP	**0.365 (1)**	**0.63 (1)**	**0.575 (1)**	GKFDK-GP	0.160 (3)	0.562 (3)	0.169 (2)
GKFDK-LP	0.341 (2)	0.584 (4)	0.538 (4)	GKFDK-LP	0.162 (2)	**0.564 (1)**	**0.172 (1)**

Tabela 18 – Ranking médio de desempenho dos algoritmos nos conjuntos de dados reais.

Algoritmos	MPC	FRG	HUL	ARI	FMS	NMI	OAR
FCM	3.00 (2)	4.36 (4)	4.64 (6)	4.50 (5)	4.29 (5)	4.43 (5)	4.50 (4)
KFCM	4.36 (7)	5.00 (7)	4.57 (5)	4.07 (4)	4.00 (4)	3.71 (3)	5.00 (5)
FDK	4.14 (4)	4.50 (5)	4.21 (4)	4.64 (6)	4.64 (6)	4.64 (6)	5.20 (6)
WFDK	**2.36 (1)**	4.79 (6)	5.00 (7)	5.86 (7)	6.14 (7)	5.79 (7)	5.80 (7)
GKFDK	4.29 (6)	3.79 (3)	3.64 (3)	3.21 (3)	3.36 (3)	3.79 (4)	3.70 (3)
GKFDK-GP	4.14 (4)	2.43 (2)	3.00 (2)	2.86 (2)	2.93 (2)	2.86 (2)	2.30 (2)
GKFDK-LP	3.14 (3)	**2.27 (1)**	**2.57 (1)**	**2.36 (1)**	**2.14 (1)**	**2.14 (1)**	**1.30 (1)**

Os resultados obtidos indicam que os algoritmos propostos conseguem fornecer uma boa qualidade de agrupamento de dados, mostrando a habilidade desses algoritmos em encontrar a verdadeira classe dos objetos. Além disso, a combinação de funções kernel com algoritmos de co-clustering se mostrou eficiente, como também a utilização de distâncias adaptativas, que permitem computar os pesos das variáveis durante o processo de otimização.

5.3.2 Análise do caso médio

Na análise do caso médio o objetivo é verificar a robustez dos algoritmos quanto a inicialização. Assim sendo, a média e o desvio padrão das métricas são computadas considerando as 50 repetições em cada conjunto de dados. Além disso, nessa análise nós aplicamos o teste de hipóteses de Wilcoxon (WILCOXON, 1992) para comparar estatisticamente o melhor algoritmo proposto com o melhor algoritmo do estado da arte. O objetivo é averiguar se os resultados obtidos pelos dois algoritmos diferem estatisticamente ou se a diferença entre eles ocorre ao acaso.

As Tabelas 19 e 20 mostram a média e o desvio padrão das métricas fuzzy e crisp, respectivamente. Em ambas as tabelas os valores em negrito indicam que houve diferença estatística, considerando 5% de significância, entre os resultados do algoritmo proposto com melhor desempenho e do algoritmo do estado da arte com melhor desempenho. A Tabela 21 mostra o desempenho médio geral dos algoritmos considerando as 50 repetições e todos os conjuntos de dados.

Os resultados das métricas fuzzy expressos em termos de média e desvio padrão mostram que os algoritmos propostos, no geral, apresentaram desempenho superior em relação aos índices *FRG* e *HUL*, reforçado pelo fato de que os testes estatísticos indicaram uma diferença

significativa entre os resultados do algoritmo proposto com melhor desempenho e do algoritmo do estado da arte com melhor desempenho em todos os conjuntos de dados, com exceção do conjunto Image Segmentation. Em relação ao *MPC*, os algoritmos propostos apresentaram bons desempenhos no geral, principalmente o algoritmo *GKFDK-LP*. Os testes estatísticos indicaram que não há diferença estatística entre o melhor algoritmo proposto e o melhor algoritmo do estado da arte em quatro conjunto de dados: Cervical Cancer, Chemical Composition, Yale e Breast Tissue.

Em relação às métricas crisp, o resultado médio mostra que os algoritmos propostos apresentaram melhores desempenho em relação aos algoritmos da literatura na grande maioria dos conjuntos de dados, considerando os três índices de avaliação. O teste estatístico de Wilcoxon indicou diferença significativa entre o melhor modelo proposto e o melhor modelo do estado da arte em quase todos os casos. Apenas em relação ao *ARI* e *FMS* no conjunto de dados Image Segmentation e em relação ao *ARI* no conjunto de dados Glioma a diferença não foi estatisticamente significativa. Esse resultado reforça a superioridade dos algoritmos propostos em relação aos algoritmos do estado da arte.

Ao analisar o desempenho médio geral (Tabela 21) observamos que o algoritmo proposto com ponderação local *GKFDK-LP* apresentou o melhor desempenho em relação a todas as métricas de avaliação. O algoritmo proposto com ponderação global *GKFDK-GP* obteve o segundo melhor resultado, com exceção do índice *MPC*, em que o *WFDK* obteve o segundo melhor resultado. O algoritmo proposto sem ponderação *GKFDK* obteve o terceiro melhor resultado em relação às métricas *FRG*, *ARI*, *FMS* e *NMI*.

Os resultados obtidos em relação à média mostram que os algoritmos propostos são robustos a inicialização aleatória inerente a eles. Embora, tanto o *GKFDK* como o *GKFDK-GP* apresentaram bons e satisfatórios resultados, o destaque ficou por conta do método *GKFDK-LP* que se sobressaiu em relação a todos os outros algoritmos com respeito a todas as métricas de avaliação.

Tabela 19 – Média ± desvio padrão das métricas fuzzy nas 50 repetições. Valores em negrito indicam diferenças significativas entre o modelo do estado da arte e o modelo proposto com melhores desempenhos (p-valor do teste de Wilcoxon abaixo de 0.05).

Algoritmos	Algerian Forest			Algoritmos	Breast Cancer		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.895 ± 0.000	0.677 ± 0.000	0.679 ± 0.000	FCM	0.949 ± 0.000	0.837 ± 0.000	0.837 ± 0.000
KFCM	0.891 ± 0.000	0.676 ± 0.000	0.678 ± 0.000	KFCM	0.948 ± 0.000	0.839 ± 0.000	0.839 ± 0.000
FDK	0.888 ± 0.000	0.687 ± 0.000	0.690 ± 0.000	FDK	0.943 ± 0.000	0.823 ± 0.000	0.823 ± 0.000
WFDK	0.775 ± 0.001	0.619 ± 0.003	0.629 ± 0.003	WFDK	0.979 ± 0.000	0.806 ± 0.000	0.806 ± 0.000
GKFDK	0.769 ± 0.000	0.730 ± 0.000	0.742 ± 0.000	GKFDK	**0.994 ± 0.000**	0.840 ± 0.000	0.840 ± 0.000
GKFDK-GP	**0.999 ± 0.000**	0.759 ± 0.000	0.759 ± 0.000	GKFDK-GP	0.955 ± 0.000	0.851 ± 0.000	0.851 ± 0.000
GKFDK-LP	0.917 ± 0.000	**0.848 ± 0.000**	**0.848 ± 0.000**	GKFDK-LP	0.962 ± 0.000	**0.868 ± 0.000**	**0.868 ± 0.000**
Algoritmos	Cervical Cancer			Algoritmos	Chemical Composition		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.398 ± 0.000	0.554 ± 0.000	0.586 ± 0.000	FCM	1.000 ± 0.000	0.949 ± 0.101	0.949 ± 0.101
KFCM	0.303 ± 0.089	0.517 ± 0.015	0.534 ± 0.024	KFCM	0.236 ± 0.068	0.531 ± 0.030	0.548 ± 0.024
FDK	1.000 ± 0.000	0.649 ± 0.000	0.649 ± 0.000	FDK	1.000 ± 0.000	0.799 ± 0.019	0.799 ± 0.019
WFDK	1.000 ± 0.000	0.635 ± 0.002	0.635 ± 0.002	WFDK	0.994 ± 0.010	0.698 ± 0.141	0.698 ± 0.141
GKFDK	0.518 ± 0.011	0.603 ± 0.001	0.629 ± 0.000	GKFDK	1.000 ± 0.000	0.796 ± 0.000	0.796 ± 0.000
GKFDK-GP	0.930 ± 0.000	0.652 ± 0.000	0.652 ± 0.000	GKFDK-GP	1.000 ± 0.000	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK-LP	1.000 ± 0.000	**0.654 ± 0.007**	**0.654 ± 0.007**	GKFDK-LP	1.000 ± 0.000	0.977 ± 0.000	0.977 ± 0.000
Algoritmos	Image Segmentation			Algoritmos	Vehicle		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.931 ± 0.008	0.889 ± 0.022	0.888 ± 0.022	FCM	0.997 ± 0.001	0.644 ± 0.009	0.644 ± 0.009
KFCM	0.836 ± 0.081	0.851 ± 0.034	0.846 ± 0.036	KFCM	0.917 ± 0.004	0.650 ± 0.001	0.648 ± 0.001
FDK	0.992 ± 0.003	0.860 ± 0.010	0.860 ± 0.010	FDK	0.992 ± 0.002	0.650 ± 0.003	0.650 ± 0.003
WFDK	**0.999 ± 0.001**	0.832 ± 0.008	0.832 ± 0.008	WFDK	**0.999 ± 0.000**	0.652 ± 0.000	0.652 ± 0.000
GKFDK	0.989 ± 0.003	0.893 ± 0.023	0.893 ± 0.023	GKFDK	0.980 ± 0.006	0.653 ± 0.004	0.653 ± 0.004
GKFDK-GP	0.997 ± 0.001	0.861 ± 0.006	0.861 ± 0.006	GKFDK-GP	0.959 ± 0.003	0.647 ± 0.000	0.647 ± 0.000
GKFDK-LP	0.998 ± 0.001	0.804 ± 0.046	0.804 ± 0.046	GKFDK-LP	0.953 ± 0.007	**0.677 ± 0.001**	**0.677 ± 0.002**
Algoritmos	Paris Housing			Algoritmos	Glioma		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	**0.690 ± 0.000**	0.500 ± 0.000	0.523 ± 0.000	FCM	**0.97 ± 0.007**	**0.739 ± 0.011**	**0.739 ± 0.011**
KFCM	0.680 ± 0.000	0.500 ± 0.000	0.524 ± 0.000	KFCM	0.080 ± 0.000	0.625 ± 0.001	0.316 ± 0.006
FDK	0.298 ± 0.000	0.504 ± 0.000	0.613 ± 0.000	FDK	0.162 ± 0.000	0.676 ± 0.000	0.575 ± 0.000
WFDK	0.320 ± 0.000	0.504 ± 0.000	0.606 ± 0.000	WFDK	0.816 ± 0.037	0.728 ± 0.003	0.727 ± 0.003
GKFDK	0.067 ± 0.047	0.501 ± 0.001	**0.717 ± 0.038**	GKFDK	0.206 ± 0.004	0.690 ± 0.002	0.616 ± 0.010
GKFDK-GP	0.105 ± 0.062	0.504 ± 0.002	0.705 ± 0.037	GKFDK-GP	0.732 ± 0.031	0.729 ± 0.006	0.727 ± 0.002
GKFDK-LP	0.193 ± 0.078	**0.507 ± 0.003**	0.673 ± 0.043	GKFDK-LP	0.125 ± 0.013	0.663 ± 0.004	0.499 ± 0.025
Algoritmos	WebKB4			Algoritmos	Yale		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.820 ± 0.004	0.585 ± 0.004	0.578 ± 0.004	FCM	0.999 ± 0.002	0.909 ± 0.007	0.909 ± 0.007
KFCM	0.908 ± 0.009	0.591 ± 0.016	0.589 ± 0.016	KFCM	0.999 ± 0.001	0.906 ± 0.009	0.906 ± 0.009
FDK	0.644 ± 0.006	0.519 ± 0.001	0.487 ± 0.001	FDK	0.998 ± 0.001	0.911 ± 0.004	0.911 ± 0.004
WFDK	0.833 ± 0.000	0.567 ± 0.000	0.558 ± 0.000	WFDK	0.999 ± 0.001	0.889 ± 0.003	0.889 ± 0.003
GKFDK	0.715 ± 0.003	0.567 ± 0.000	0.542 ± 0.000	GKFDK	1.000 ± 0.000	0.913 ± 0.007	0.913 ± 0.007
GKFDK-GP	0.177 ± 0.007	0.599 ± 0.000	0.418 ± 0.003	GKFDK-GP	0.980 ± 0.006	**0.920 ± 0.005**	**0.920 ± 0.005**
GKFDK-LP	**0.996 ± 0.002**	**0.733 ± 0.015**	**0.733 ± 0.015**	GKFDK-LP	0.999 ± 0.001	0.905 ± 0.010	0.905 ± 0.010
Algoritmos	CSTR			Algoritmos	Vertebral Column		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	0.104 ± 0.003	0.612 ± 0.000	0.417 ± 0.002	FCM	**0.995 ± 0.000**	0.552 ± 0.000	0.552 ± 0.000
KFCM	0.530 ± 0.082	0.534 ± 0.058	0.469 ± 0.040	KFCM	0.941 ± 0.000	0.557 ± 0.000	0.557 ± 0.000
FDK	0.104 ± 0.003	0.612 ± 0.000	0.417 ± 0.002	FDK	0.541 ± 0.016	0.532 ± 0.003	0.557 ± 0.004
WFDK	0.231 ± 0.000	0.616 ± 0.000	0.464 ± 0.000	WFDK	0.960 ± 0.000	0.544 ± 0.000	0.544 ± 0.000
GKFDK	0.000 ± 0.000	0.612 ± 0.000	0.276 ± 0.000	GKFDK	0.892 ± 0.000	0.563 ± 0.000	0.565 ± 0.000
GKFDK-GP	0.046 ± 0.015	0.613 ± 0.000	0.314 ± 0.006	GKFDK-GP	0.571 ± 0.009	0.530 ± 0.006	0.548 ± 0.008
GKFDK-LP	**0.998 ± 0.002**	**0.826 ± 0.031**	**0.826 ± 0.031**	GKFDK-LP	0.924 ± 0.000	**0.627 ± 0.000**	**0.627 ± 0.000**
Algoritmos	Breast Tissue			Algoritmos	Abalone		
	MPC	FRG	HUL		MPC	FRG	HUL
FCM	1.000 ± 0.000	0.767 ± 0.025	0.767 ± 0.025	FCM	0.868 ± 0.000	0.611 ± 0.000	0.608 ± 0.000
KFCM	1.000 ± 0.000	0.776 ± 0.021	0.776 ± 0.021	KFCM	0.866 ± 0.000	0.611 ± 0.000	0.608 ± 0.000
FDK	1.000 ± 0.000	0.779 ± 0.017	0.779 ± 0.017	FDK	0.956 ± 0.000	0.612 ± 0.001	0.612 ± 0.001
WFDK	0.780 ± 0.026	0.764 ± 0.018	0.752 ± 0.014	WFDK	**0.997 ± 0.000**	0.611 ± 0.001	0.611 ± 0.001
GKFDK	1.000 ± 0.000	0.771 ± 0.009	0.771 ± 0.009	GKFDK	0.381 ± 0.010	0.581 ± 0.000	0.526 ± 0.001
GKFDK-GP	1.000 ± 0.000	**0.804 ± 0.016**	**0.804 ± 0.016**	GKFDK-GP	0.898 ± 0.001	0.617 ± 0.000	0.616 ± 0.000
GKFDK-LP	0.731 ± 0.032	0.78 ± 0.008	0.771 ± 0.010	GKFDK-LP	0.993 ± 0.000	**0.623 ± 0.000**	**0.623 ± 0.000**

Tabela 20 – Média ± desvio padrão das métricas crisp nas 50 repetições. Valores em negrito indicam diferenças significativas entre o modelo do estado da arte e o modelo proposto com melhores desempenhos (p-valor do teste de Wilcoxon abaixo de 0.05).

Algoritmos	Algerian Forest			Algoritmos	Breast Cancer		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.326 ± 0.000	0.785 ± 0.000	0.332 ± 0.000	FCM	0.671 ± 0.000	0.909 ± 0.000	0.555 ± 0.000
KFCM	0.326 ± 0.000	0.785 ± 0.000	0.332 ± 0.000	KFCM	0.683 ± 0.000	0.913 ± 0.000	0.566 ± 0.000
FDK	0.365 ± 0.000	0.803 ± 0.000	0.358 ± 0.000	FDK	0.659 ± 0.000	0.905 ± 0.000	0.550 ± 0.000
WFDK	0.253 ± 0.006	0.748 ± 0.004	0.321 ± 0.004	WFDK	0.611 ± 0.000	0.888 ± 0.000	0.561 ± 0.000
GKFDK	0.567 ± 0.000	0.878 ± 0.000	0.506 ± 0.000	GKFDK	0.676 ± 0.000	0.911 ± 0.000	0.564 ± 0.000
GKFDK-GP	0.518 ± 0.000	0.861 ± 0.000	0.462 ± 0.000	GKFDK-GP	0.712 ± 0.000	0.921 ± 0.000	0.612 ± 0.000
GKFDK-LP	**0.726 ± 0.000**	**0.926 ± 0.000**	**0.623 ± 0.000**	GKFDK-LP	**0.748 ± 0.000**	**0.932 ± 0.000**	**0.671 ± 0.000**
Algoritmos	Cervical Cancer			Algoritmos	Chemical Composition		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.213 ± 0.000	0.748 ± 0.000	0.203 ± 0.000	FCM	0.898 ± 0.201	0.969 ± 0.068	0.895 ± 0.204
KFCM	0.166 ± 0.137	0.725 ± 0.060	0.152 ± 0.117	KFCM	0.453 ± 0.371	0.809 ± 0.145	0.461 ± 0.330
FDK	0.296 ± 0.000	0.786 ± 0.000	0.228 ± 0.000	FDK	0.598 ± 0.038	0.886 ± 0.011	0.592 ± 0.032
WFDK	0.267 ± 0.004	0.774 ± 0.002	0.211 ± 0.002	WFDK	0.396 ± 0.284	0.791 ± 0.122	0.397 ± 0.290
GKFDK	**0.358 ± 0.011**	**0.811 ± 0.004**	**0.290 ± 0.009**	GKFDK	0.593 ± 0.000	0.885 ± 0.000	0.587 ± 0.000
GKFDK-GP	0.296 ± 0.000	0.786 ± 0.000	0.228 ± 0.000	GKFDK-GP	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
GKFDK-LP	0.306 ± 0.015	0.790 ± 0.006	0.241 ± 0.019	GKFDK-LP	0.955 ± 0.000	0.989 ± 0.000	0.922 ± 0.000
Algoritmos	Image Segmentation			Algoritmos	Vehicle		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.600 ± 0.077	0.730 ± 0.062	**0.698 ± 0.049**	FCM	0.066 ± 0.011	0.401 ± 0.013	0.095 ± 0.015
KFCM	0.500 ± 0.102	0.656 ± 0.081	0.638 ± 0.075	KFCM	0.069 ± 0.008	0.408 ± 0.006	0.092 ± 0.015
FDK	0.497 ± 0.031	0.658 ± 0.029	0.638 ± 0.023	FDK	0.075 ± 0.008	0.418 ± 0.011	0.100 ± 0.010
WFDK	0.351 ± 0.018	0.540 ± 0.017	0.489 ± 0.004	WFDK	0.121 ± 0.000	0.456 ± 0.003	**0.184 ± 0.000**
GKFDK	0.570 ± 0.093	0.734 ± 0.075	0.659 ± 0.078	GKFDK	0.084 ± 0.015	0.429 ± 0.015	0.110 ± 0.020
GKFDK-GP	0.470 ± 0.019	0.625 ± 0.015	0.618 ± 0.006	GKFDK-GP	0.115 ± 0.000	0.452 ± 0.002	0.173 ± 0.002
GKFDK-LP	0.341 ± 0.087	0.541 ± 0.067	0.485 ± 0.088	GKFDK-LP	**0.146 ± 0.001**	**0.504 ± 0.006**	0.174 ± 0.005
Algoritmos	Paris Housing			Algoritmos	Glioma		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.000 ± 0.000	0.585 ± 0.000	0.000 ± 0.000	FCM	0.340 ± 0.013	**0.615 ± 0.012**	0.501 ± 0.006
KFCM	0.000 ± 0.000	0.585 ± 0.000	0.000 ± 0.000	KFCM	0.265 ± 0.122	0.574 ± 0.071	0.413 ± 0.126
FDK	0.015 ± 0.000	0.632 ± 0.000	0.037 ± 0.000	FDK	0.304 ± 0.000	0.587 ± 0.000	0.503 ± 0.000
WFDK	0.014 ± 0.000	0.630 ± 0.000	0.034 ± 0.000	WFDK	0.297 ± 0.012	0.558 ± 0.012	0.453 ± 0.023
GKFDK	0.017 ± 0.014	0.625 ± 0.033	0.045 ± 0.038	GKFDK	0.290 ± 0.007	0.574 ± 0.001	0.475 ± 0.002
GKFDK-GP	0.030 ± 0.020	0.644 ± 0.037	0.081 ± 0.053	GKFDK-GP	0.311 ± 0.023	0.574 ± 0.013	0.473 ± 0.010
GKFDK-LP	**0.039 ± 0.014**	**0.661 ± 0.020**	**0.103 ± 0.036**	GKFDK-LP	0.348 ± 0.015	0.589 ± 0.004	**0.507 ± 0.00**
Algoritmos	WebKB4			Algoritmos	Yale		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.092 ± 0.005	0.470 ± 0.008	0.161 ± 0.007	FCM	0.289 ± 0.039	0.533 ± 0.043	0.544 ± 0.029
KFCM	0.106 ± 0.046	0.477 ± 0.041	0.202 ± 0.040	KFCM	0.284 ± 0.043	0.530 ± 0.046	0.543 ± 0.032
FDK	0.027 ± 0.001	0.400 ± 0.001	0.012 ± 0.002	FDK	0.299 ± 0.026	0.552 ± 0.028	0.558 ± 0.020
WFDK	0.042 ± 0.000	0.388 ± 0.000	0.048 ± 0.000	WFDK	0.096 ± 0.020	0.331 ± 0.018	0.368 ± 0.011
GKFDK	0.041 ± 0.000	0.399 ± 0.001	0.047 ± 0.000	GKFDK	0.314 ± 0.039	0.554 ± 0.038	0.571 ± 0.029
GKFDK-GP	0.012 ± 0.000	0.359 ± 0.003	0.006 ± 0.000	GKFDK-GP	**0.368 ± 0.021**	**0.637 ± 0.023**	**0.616 ± 0.014**
GKFDK-LP	**0.351 ± 0.031**	**0.657 ± 0.034**	**0.373 ± 0.022**	GKFDK-LP	0.289 ± 0.038	0.543 ± 0.041	0.554 ± 0.028
Algoritmos	CSTR			Algoritmos	Vertebral Column		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.000 ± 0.003	0.297 ± 0.009	0.007 ± 0.003	FCM	0.102 ± 0.000	0.672 ± 0.000	0.161 ± 0.000
KFCM	0.123 ± 0.169	0.492 ± 0.087	0.161 ± 0.160	KFCM	0.112 ± 0.000	0.679 ± 0.000	0.160 ± 0.000
FDK	0.000 ± 0.003	0.297 ± 0.009	0.007 ± 0.003	FDK	0.109 ± 0.012	0.677 ± 0.011	0.165 ± 0.009
WFDK	0.047 ± 0.000	0.403 ± 0.000	0.052 ± 0.000	WFDK	0.090 ± 0.000	0.663 ± 0.000	0.141 ± 0.000
GKFDK	0.279 ± 0.008	0.588 ± 0.003	0.278 ± 0.010	GKFDK	0.122 ± 0.000	0.687 ± 0.000	0.144 ± 0.000
GKFDK-GP	0.226 ± 0.005	0.558 ± 0.002	0.209 ± 0.005	GKFDK-GP	0.100 ± 0.018	0.670 ± 0.013	0.141 ± 0.032
GKFDK-LP	**0.573 ± 0.070**	**0.738 ± 0.032**	**0.533 ± 0.052**	GKFDK-LP	**0.251 ± 0.000**	**0.760± 0.000**	**0.214 ± 0.000**
Algoritmos	Breast Tissue			Algoritmos	Abalone		
	ARI	FMS	NMI		ARI	FMS	NMI
FCM	0.279 ± 0.022	0.545 ± 0.028	0.512 ± 0.023	FCM	0.134 ± 0.000	0.517 ± 0.000	0.163 ± 0.000
KFCM	0.285 ± 0.022	0.550 ± 0.027	0.516 ± 0.023	KFCM	0.137 ± 0.000	0.521 ± 0.000	0.164 ± 0.000
FDK	0.279 ± 0.018	0.547 ± 0.022	0.511 ± 0.016	FDK	0.136 ± 0.001	0.518 ± 0.001	0.163 ± 0.000
WFDK	0.305 ± 0.011	0.560 ± 0.003	0.530 ± 0.006	WFDK	0.135 ± 0.002	0.519 ± 0.003	0.158 ± 0.002
GKFDK	0.250 ± 0.018	0.514 ± 0.008	0.489 ± 0.014	GKFDK	0.160 ± 0.001	0.560 ± 0.001	0.167 ± 0.001
GKFDK-GP	**0.366 ± 0.017**	**0.623 ± 0.014**	**0.587 ± 0.024**	GKFDK-GP	0.160 ± 0.000	0.560 ± 0.001	0.170 ± 0.000
GKFDK-LP	0.318 ± 0.033	0.570 ± 0.029	0.515 ± 0.039	GKFDK-LP	**0.161 ± 0.000**	**0.562 ± 0.001**	**0.171 ± 0.001**

Tabela 21 – Desempenho médio geral considerando todas as repetições e todos os conjuntos de dados reais (média ± desvio padrão). Valores em negrito indicam o melhor resultado.

Algoritmos	MPC	FRG	HUL	ARI	FMS	NMI
FCM	0.830 ± 0.257	0.702 ± 0.142	0.691 ± 0.155	0.286 ± 0.259	0.627 ± 0.182	0.345 ± 0.265
KFCM	0.724 ± 0.299	0.655 ± 0.131	0.631 ± 0.158	0.251 ± 0.183	0.622 ± 0.14	0.314 ± 0.198
FDK	0.751 ± 0.326	0.687 ± 0.125	0.673 ± 0.140	0.261 ± 0.207	0.619 ± 0.176	0.316 ± 0.229
WFDK	0.834 ± 0.244	0.676 ± 0.110	0.672 ± 0.115	0.216 ± 0.163	0.589 ± 0.162	0.282 ± 0.180
GKFDK	0.679 ± 0.361	0.694 ± 0.127	0.677 ± 0.164	0.309 ± 0.211	0.654 ± 0.162	0.352 ± 0.213
GKFDK-GP	0.739 ± 0.350	0.720 ± 0.145	0.702 ± 0.182	0.335 ± 0.265	0.662 ± 0.170	0.384 ± 0.272
GKFDK-LP	**0.842 ± 0.287**	**0.749 ± 0.126**	**0.749 ± 0.126**	**0.397 ± 0.25**	**0.697 ± 0.156**	**0.435 ± 0.224**

5.4 SÍNTESE DO CAPÍTULO

A efetividade dos algoritmos propostos foi avaliada através de diversos experimentos com dados sintéticos (simulação de Monte Carlo) e dados reais. Os resultados numéricos obtidos mostraram a superioridade dos métodos propostos, principalmente em cenários com variáveis irrelevantes, destacando principalmente os algoritmos que diferenciam os pesos de relevância das variáveis. As características de atribuição dos pesos segundo o tipo de distância adaptativa foi exemplificada nas aplicações com dados simulados e dados reais. Os experimentos com dados reais mostram a utilidade dos algoritmos propostos em diversas situações práticas.

6 CONCLUSÕES

Nesse capítulo nós apresentamos as conclusões desse trabalho, na qual introduzimos novos algoritmos de co-clustering que se baseiam em funções kernel. Parte desse trabalho foi apresentado e publicado nos anais da 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). A seguir é apresentado as contribuições, limitações e trabalhos futuros relacionados a essa dissertação.

6.1 CONTRIBUIÇÕES

Nós propusemos três novos algoritmos de co-clustering, na qual utilizam funções kernel como forma de medida de similaridade. Os novos modelos preenchem uma lacuna na literatura envolvendo co-clustering e métodos baseados em kernel. Funções kernel nos permitem modelar cenários complexos não lineares projetando os dados em um espaço de mais alta dimensão onde grupos de dados possam ser definidos de forma mais clara, com a esperança de que a tarefa de agrupar seja mais fácil. Nesse trabalho nos baseamos no kernel Gaussiano, pois é o mais amplamente utilizado na literatura e é facilmente tratável, analiticamente.

Um dos modelos propostos, na qual denominamos *Gaussian Kernel Fuzzy Double K-Means* (*GKFDK*), é uma versão semelhante do algoritmo *Fuzzy Double K-Means* (*FDK*) para abordagem de kernel. Os outros dois algoritmos introduzem o conceito de distâncias adaptativas, na qual atribuem pesos positivos as variáveis de acordo com alguma restrição adequada. Esse tipo de distância é conveniente para aprender os pesos das variáveis durante o processo de agrupamento, levando a uma melhora no desempenho dos algoritmos. Distâncias adaptativas mudam a cada iteração do algoritmo e podem ser a mesma para todos os grupos de objetos (distância adaptativa global) ou diferentes para cada grupo de objetos (distância adaptativa local). Considerando a distância adaptativa global, propusemos o algoritmo *Gaussian Kernel Fuzzy Double K-Means Based on Global Adaptive Distance* (*GKFDK-GP*), enquanto o algoritmo *Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance* (*GKFDK-LP*) incorpora a ideia da distância adaptativa local.

Os algoritmos propostos partem de uma solução inicial aleatória das partições fuzzy de objetos e variáveis e alterna em três etapas (Representação, Alocação dos objetos e Alocação das variáveis. Os métodos baseados em distâncias adaptativas, *GKFDK-GP* e *GKFDK-LP*,

tem uma etapa adicional de ponderação, na qual os pesos das variáveis são computados) até a convergência que ocorre quando as funções objetivo alcançam um valor estacionário. Para computar os pesos das variáveis nós utilizamos a restrição produto. No caso do *GKFDK-GP* (distância adaptativa global) consideramos que o produto dos pesos das variáveis deve ser igual a um, enquanto que para o *GKFDK-LP* (distância adaptativa local) consideramos que o produto dos pesos das variáveis em cada grupo deve ser igual a um.

6.2 RESUMO DOS RESULTADOS

A efetividade dos algoritmos de co-clustering baseados em kernel foi mostrada através de experimentos realizados com dados simulados e dados reais. Nas aplicações a dados sintéticos foram simuladas quatro configurações de dados segundo as características dos algoritmos. No cenário básico, onde os grupos de objetos e variáveis são bem definidos, todos os algoritmos apresentaram bons desempenhos. No cenário 2 foram introduzidas variáveis que são irrelevantes em relação aos grupos de objetos. Nesse caso, os modelos de agrupamento tradicional tiveram uma degradação no desempenho, principalmente em relação aos índices de avaliação crisp. Os modelos de co-clustering mantiveram uma boa performance, mostrando que esses algoritmos conseguem mitigar a presença de variáveis irrelevantes no conjunto de dados, encontrando uma estrutura representativa dos grupos. Na terceira configuração de dados sintéticos, o número de variáveis irrelevantes era superior ao número de variáveis relevantes. Nesse caso, todos os modelos que não computam os pesos de relevância das variáveis apresentaram resultados insatisfatórios. Os modelos com ponderação global, i.e. *WFDK* (estado da arte) e *GKFDK-GP* (proposto), apresentaram os melhores desempenhos em relação a todas as métricas de avaliação, exceto para o *MPC*. Esse resultado é condizente com a característica desses algoritmos, que consideram um peso global para as variáveis. O algoritmo proposto com ponderação local *GKFDK-LP* também mostrou bons resultados nesse cenário. No quarto e último cenário, as classes de objetos eram descritas por conjuntos diferentes de variáveis relevantes. Nesse caso, o modelo proposto *GKFDK-LP* apresentou os melhores resultados em relação a todas as métricas de avaliação, exceto para o *MPC*. Esse resultado condiz com a característica do algoritmo que, ao usar distância adaptativa local, consegue identificar grupos distintos de variáveis relevantes para os grupos de objetos. Nesse cenário, os algoritmos com ponderação global (*WFDK* e *GKFDK-GP*) apresentaram resultados medianos, pois consideram que o conjunto de variáveis relevantes é o mesmo para todos os grupos de objetos.

Os testes estatísticos de Friedman e de Nemenyi realizados em todos os cenários mostram a significância dos resultados obtidos.

As aplicações aos dados reais mostraram a superioridade dos algoritmos propostos. Os modelos *GKFDK-LP* e *GKFDK-GP* apresentaram o primeiro e segundo melhor ranking médio de desempenho, respectivamente, em relação a todas as métricas de avaliação, exceto para o índice *MPC*, no qual o método *WFDK* se sobressaiu. O algoritmo proposto sem ponderação *GKFDK* apresentou o terceiro melhor ranking médio, exceto para as métricas *MPC* e *NMI*. Em relação à métrica *Overall Average Ranking* (*OAR*), que agrupa as informações do ranking de todas as métricas em uma única medida, os algoritmos propostos foram os melhores, na qual o algoritmo *GKFDK-LP* foi o melhor ranqueado, seguido do *GKFDK-GP* e *GKFDK*. Os resultados expressos em termos de média e desvio padrão provaram a robustez dos algoritmos propostos. O teste estatístico de Wilcoxon usado para comparar o algoritmo do estado da arte com melhor desempenho com o algoritmo proposto com melhor desempenho, mostra a efetividade da abordagem proposta em comparação com a literatura. O algoritmo *GKFDK-LP* apresentou o melhor desempenho médio geral (média dos resultados considerando todos os conjuntos de dados) em relação a todas as métricas de avaliação fuzzy e crisp. O modelo *GKFDK-GP* obteve o segundo melhor desempenho médio, exceto para o índice *MPC*. O método *GKFDK* obteve o terceiro melhor desempenho médio, exceto para os índices *MPC* e *HUL*.

Em geral, os modelos propostos obtiveram os melhores resultados, principalmente em relação às métricas crisp. Embora os modelos propostos não tenham sido os melhores em relação ao *MPC*, no geral, apresentaram bons resultados. Os desempenhos computados através de seis métricas de avaliação mostram a robustez dos resultados obtidos. O grande destaque dos algoritmos propostos foi o algoritmo *Gaussian Kernel Fuzzy Double K-Means Based on Local Adaptive Distance* (*GKFDK-LP*) que considera que as variáveis tem pesos diferentes em relação a cada grupos de objetos. Por fim, os resultados obtidos indicam que co-clustering e funções kernel é uma maneira efetiva de melhorar o desempenho dos algoritmos, sendo capaz de encontrar estruturas de grupos de objetos mais representativas, em comparação com modelos relacionados de co-clustering e agrupamento convencional.

6.3 LIMITAÇÕES

Uma das principais limitações dos algoritmos propostos é que a função objetivo não converge monotonicamente ao longo das iterações do algoritmo, pelo fato dos métodos usarem uma heurística na etapa de alocação dos objetos e variáveis. Isso causa oscilações nos valores da função objetivo, fazendo com o que os algoritmos levem mais iterações para alcançar o critério de parada. Apesar disso, o critério decresce até um valor estacionário. Outra limitação desses algoritmos está na sensibilidade em relação aos parâmetros m e n. Pequenas variações nos valores desses parâmetros degradam ou melhoram consideravelmente o desempenho dos algoritmos. Essa sensibilidade é mais aguda em comparação com os algoritmos de agrupamento tradicional. Talvez o fato de ter um parâmetro a mais influencie nesse quesito. Valores muito pequenos de m e n podem causar problemas técnicos, como descrito na Seção 5.1.2. Valores muito grandes de m e n deixam os protótipos dos blocos de dados muito próximos.

6.4 TRABALHOS FUTUROS

Algumas pesquisas relacionadas aos temas desenvolvidos nessa dissertação podem ser realizadas, dentre elas, podemos citar:

- Extensão dos métodos propostos nessa dissertação para a abordagem de agrupamento baseado em kernel no espaço de kernel;

- Extensão dos métodos propostos nesse trabalho para a abordagem hard de agrupamento. Embora a abordagem fuzzy seja mais condizente com as aplicações reais atuais, os modelos hard são menos custosos computacionalmente, e nesse caso, especificamente, pode se evitar os problemas encontrados com os parâmetros m e n;

- Considerar uma estrutura de diagonal blocos, pertinente a conjunto de dados extraídos de textos e incorporar funções kernel e distâncias adaptativas no cálculo das funções objetivo.

REFERÊNCIAS

AFFELDT, S.; LABIOD, L.; NADIF, M. Regularized bi-directional co-clustering. *Statistics and Computing*, Springer, v. 31, n. 3, p. 32, 2021.

AILEM, M.; ROLE, F.; NADIF, M. Co-clustering document-term matrices by direct maximization of graph modularity. In: *Proceedings of the 24th ACM international on conference on information and knowledge management*. [S.l.: s.n.], 2015. p. 1807–1810.

AIREN, S.; AGRAWAL, J. Movie recommender system using parameter tuning of user and movie neighbourhood via co-clustering. *Procedia Computer Science*, Elsevier, v. 218, p. 1176–1183, 2023.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, ACM New York, NY, USA, v. 28, n. 2, p. 49–60, 1999.

BAEZA-YATES, R.; RIBEIRO-NETO, B. et al. *Modern information retrieval*. [S.l.]: ACM press New York, 1999. v. 463.

BANERJEE, A.; DHILLON, I.; GHOSH, J.; MERUGU, S.; MODHA, D. S. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2004. p. 509–514.

BELLMAN, R. Dynamic programming. *Science*, American Association for the Advancement of Science, v. 153, n. 3731, p. 34–37, 1966.

BEN-HUR, A.; HORN, D.; SIEGELMANN, H. T.; VAPNIK, V. Support vector clustering. *Journal of machine learning research*, v. 2, n. Dec, p. 125–137, 2001.

BEZDEK, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. [S.l.]: Kluwer Academic Publishers, 1981.

BEZDEK, J. C. *Pattern recognition with fuzzy objective function algorithms*. [S.l.]: Springer Science & Business Media, 2013.

BORA, D. J.; GUPTA, D.; KUMAR, A. A comparative study between fuzzy clustering algorithm and hard clustering algorithm. *arXiv preprint arXiv:1404.6059*, 2014.

BORA, M.; JYOTI, D.; GUPTA, D.; KUMAR, A. Effect of different distance measures on the performance of k-means algorithm: an experimental study in matlab. *arXiv preprint arXiv:1405.7471*, 2014.

CAPUTO, B.; SIM, K.; FURESJO, F.; SMOLA, A. Appearance-based object recognition using svms: which kernel should i use? In: *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*. [S.l.: s.n.], 2002. v. 2002.

CARVALHO, F. d. A. D.; BALZANELLA, A.; IRPINO, A.; VERDE, R. Co-clustering algorithms for distributional data with automated variable weighting. *Information Sciences*, Elsevier, v. 549, p. 87–115, 2021.

CARVALHO, F. d. A. de; LECHEVALLIER, Y. Dynamic clustering of interval-valued data based on adaptive quadratic distances. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, IEEE, v. 39, n. 6, p. 1295–1306, 2009.

CARVALHO, F. d. A. de; TENÓRIO, C. P.; JUNIOR, N. L. C. Partitional fuzzy clustering methods based on adaptive quadratic distances. *Fuzzy Sets and Systems*, Elsevier, v. 157, n. 21, p. 2833–2857, 2006.

CHEN, D. Z. S. Fuzzy clustering using kernel method. *IEEE, Nanjing, China*, 2002.

CHEN, L.; CHEN, C. P.; LU, M. A multiple-kernel fuzzy c-means algorithm for image segmentation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 41, n. 5, p. 1263–1274, 2011.

CHEN, S.; ZHANG, D. Robust image segmentation using fcm with spatial constraints based on new kernel-induced distance measure. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 34, n. 4, p. 1907–1916, 2004.

CHEN, X.; YANG, M.; HUANG, J. Z.; MING, Z. Twcc: Automated two-way subspace weighting partitional co-clustering. *Pattern Recognition*, Elsevier, v. 76, p. 404–415, 2018.

CHITSAZ, E.; JAHROMI, M. Z. A novel soft subspace clustering algorithm with noise detection for high dimensional datasets. *Soft Computing*, Springer, v. 20, n. 11, p. 4463–4472, 2016.

CHIU, M.-C.; HSU, Y.-W. Using fuzzy c-means clustering based on integration of psychological and physiological data for therapeutic music design. *Journal of Industrial and Production Engineering*, Taylor & Francis, v. 34, n. 5, p. 382–397, 2017.

DAVE, R. N. Validating fuzzy partitions obtained through c-shells clustering. *Pattern recognition letters*, Elsevier, v. 17, n. 6, p. 613–623, 1996.

DENG, P.; LI, T.; WANG, H.; HORNG, S.-J.; YU, Z.; WANG, X. Tri-regularized nonnegative matrix tri-factorization for co-clustering. *Knowledge-Based Systems*, Elsevier, v. 226, p. 107101, 2021.

DENG, Z.; CHOI, K.-S.; JIANG, Y.; WANG, J.; WANG, S. A survey on soft subspace clustering. *Information sciences*, Elsevier, v. 348, p. 84–106, 2016.

DHILLON, I. S. Co-clustering documents and words using bipartite spectral graph partitioning. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2001. p. 269–274.

DHILLON, I. S.; MALLELA, S.; KUMAR, R. A divisive information theoretic feature clustering algorithm for text classification. *The Journal of machine learning research*, JMLR. org, v. 3, p. 1265–1287, 2003.

DING, Y.; FU, X. Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm. *Neurocomputing*, Elsevier, v. 188, p. 233–238, 2016.

DU, Y.; SUN, Y.-E.; HUANG, H.; HUANG, L.; XU, H.; BAO, Y.; GUO, H. Bayesian co-clustering truth discovery for mobile crowd sensing systems. *IEEE Transactions on Industrial Informatics*, IEEE, v. 16, n. 2, p. 1045–1057, 2019.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

EZUGWU, A. E.; IKOTUN, A. M.; OYELADE, O. O.; ABUALIGAH, L.; AGUSHAKA, J. O.; EKE, C. I.; AKINYELU, A. A. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 110, p. 104743, 2022.

FERREIRA, M. R.; CARVALHO, F. d. A. de. Kernel-based hard clustering methods in the feature space with automatic variable weighting. *Pattern recognition*, Elsevier, v. 47, n. 9, p. 3082–3095, 2014.

FERREIRA, M. R.; CARVALHO, F. d. A. de; SIMÕES, E. C. Kernel-based hard clustering methods with kernelization of the metric and automatic weighting of the variables. *Pattern Recognition*, Elsevier, v. 51, p. 310–321, 2016.

FERREIRA, M. R. P.; CARVALHO, F. A. T. de. Kernel fuzzy c-means with automatic variable weighting. *Fuzzy Sets and Systems*, Elsevier, v. 237, p. 1–46, 2014.

FILIPPONE, M.; CAMASTRA, F.; MASULLI, F.; ROVETTA, S. A survey of kernel and spectral methods for clustering. *Pattern recognition*, Elsevier, v. 41, n. 1, p. 176–190, 2008.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

FRIGUI, H.; HWANG, C.; RHEE, F. C.-H. Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, Elsevier, v. 40, n. 11, p. 3053–3068, 2007.

FRIGUI, H.; KRISHNAPURAM, R. A robust competitive clustering algorithm with applications in computer vision. *Ieee transactions on pattern analysis and machine intelligence*, IEEE, v. 21, n. 5, p. 450–465, 1999.

GAN, G.; MA, C.; WU, J. *Data clustering: theory, algorithms, and applications*. [S.l.]: SIAM, 2020.

GEWENIGER, T.; ZÜLKE, D.; HAMMER, B.; VILLMANN, T. Median fuzzy c-means for clustering dissimilarity data. *Neurocomputing*, Elsevier, v. 73, n. 7-9, p. 1109–1116, 2010.

GIROLAMI, M. Mercer kernel-based clustering in feature space. *IEEE transactions on neural networks*, IEEE, v. 13, n. 3, p. 780–784, 2002.

GONZÁLEZ, F. A.; BERMEO, D.; RAMOS, L.; NASRAOUI, O. On the robustness of kernel-based clustering. In: SPRINGER. *Iberoamerican Congress on Pattern Recognition*. [S.l.], 2012. p. 122–129.

GOVAERT, G. La classification croisée. Thèse d'état, Université Paris 6, France, 1983.

GOVAERT, G.; NADIF, M. Clustering with block mixture models. *Pattern Recognition*, Elsevier, v. 36, n. 2, p. 463–473, 2003.

GOVAERT, G.; NADIF, M. Block clustering with bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis*, Elsevier, v. 52, n. 6, p. 3233–3245, 2008.

GOVAERT, G.; NADIF, M. *Co-clustering: models, algorithms and applications*. [S.l.]: John Wiley & Sons, 2013.

GRAVES, D.; PEDRYCZ, W. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. *Fuzzy sets and systems*, Elsevier, v. 161, n. 4, p. 522–543, 2010.

GUHA, S.; RASTOGI, R.; SHIM, K. Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, ACM New York, NY, USA, v. 27, n. 2, p. 73–84, 1998.

GUHA, S.; RASTOGI, R.; SHIM, K. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, Elsevier, v. 25, n. 5, p. 345–366, 2000.

GUSTAFSON, D. E.; KESSEL, W. C. Fuzzy clustering with a fuzzy covariance matrix. In: IEEE. *1978 IEEE conference on decision and control including the 17th symposium on adaptive processes*. [S.l.], 1979. p. 761–766.

HAN, J.; PEI, J.; TONG, H. *Data mining: concepts and techniques*. [S.l.]: Morgan kaufmann, 2022.

HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Prentice Hall PTR, 1994.

HINNEBURG, A.; KEIM, D. A. et al. *An efficient approach to clustering in large multimedia databases with noise*. [S.l.]: Bibliothek der Universität Konstanz, 1998. v. 98.

HORE, P.; HALL, L. O.; GOLDGOF, D. B. Single pass fuzzy c means. In: IEEE. *2007 IEEE International Fuzzy Systems Conference*. [S.l.], 2007. p. 1–7.

HOUARI, A.; AYADI, W.; YAHIA, S. B. A new fca-based method for identifying biclusters in gene expression data. *International Journal of Machine Learning and Cybernetics*, Springer, v. 9, n. 11, p. 1879–1893, 2018.

HUANG, J. Z.; NG, M. K.; RONG, H.; LI, Z. Automated variable weighting in k-means type clustering. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 27, n. 5, p. 657–668, 2005.

HUANG, S.; XU, Z.; LV, J. Adaptive local structure learning for document co-clustering. *Knowledge-Based Systems*, Elsevier, v. 148, p. 74–84, 2018.

HUANG, S.; XU, Z.; TSANG, I. W.; KANG, Z. Auto-weighted multi-view co-clustering with bipartite graphs. *Information Sciences*, Elsevier, v. 512, p. 18–30, 2020.

HUBERT, L.; ARABIE, P. Comparing partitions. *Journal of classification*, Springer, v. 2, n. 1, p. 193–218, 1985.

HULLERMEIER, E.; RIFQI, M.; HENZGEN, S.; SENGE, R. Comparing fuzzy partitions: A generalization of the rand index and related measures. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 20, n. 3, p. 546–556, 2011.

INOKUCHI, R.; MIYAMOTO, S. Lvq clustering and som using a kernel function. In: IEEE. *2004 IEEE international conference on fuzzy systems (IEEE cat. no. 04CH37542)*. [S.l.], 2004. v. 3, p. 1497–1500.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, Elsevier, v. 31, n. 8, p. 651–666, 2010.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, Acm New York, NY, USA, v. 31, n. 3, p. 264–323, 1999.

JAJUGA, K. L1-norm based fuzzy clustering. *Fuzzy Sets and Systems*, Elsevier, v. 39, n. 1, p. 43–50, 1991.

JIA, H.; CHEUNG, Y.-M. Subspace clustering of categorical and numerical data with an unknown number of clusters. *IEEE transactions on neural networks and learning systems*, IEEE, v. 29, n. 8, p. 3308–3325, 2017.

JIANG, Y.; CHUNG, F.-L.; WANG, S.; DENG, Z.; WANG, J.; QIAN, P. Collaborative fuzzy clustering from multiple weighted views. *IEEE transactions on cybernetics*, IEEE, v. 45, n. 4, p. 688–701, 2014.

JING, L.; NG, M. K.; HUANG, J. Z. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 19, n. 8, p. 1026–1041, 2007.

JOLLIFFE, I. T. *Principal component analysis for special types of data*. [S.l.]: Springer, 2002.

KARYPIS, G.; HAN, E.-H.; KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, IEEE, v. 32, n. 8, p. 68–75, 1999.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. [S.l.]: John Wiley & Sons, 2009.

KAUFMANN, L. Clustering by means of medoids. In: *Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987*. [S.l.: s.n.], 1987. p. 405–416.

KELLER, A.; KLAWONN, F. Fuzzy clustering with weighting of data variables. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, World Scientific, v. 8, n. 06, p. 735–746, 2000.

KIM, D.-W.; LEE, K.; LEE, D.; LEE, K. H. A kernel-based subtractive clustering method. *Pattern Recognition Letters*, Elsevier, v. 26, n. 7, p. 879–891, 2005.

KOLEN, J. F.; HUTCHESON, T. Reducing the time complexity of the fuzzy c-means algorithm. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 10, n. 2, p. 263–267, 2002.

LABIOD, L.; NADIF, M. Co-clustering for binary and categorical data with maximum modularity. In: IEEE. *2011 IEEE 11th international conference on data mining*. [S.l.], 2011. p. 1140–1145.

LACLAU, C. *Hard and fuzzy block clustering algorithms for high dimensional data*. Tese (Doutorado) — Sorbonne Paris Cité, 2016.

LACLAU, C.; CARVALHO, F. A. T. de; NADIF, M. Fuzzy co-clustering with automated variable weighting. In: IEEE. *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2015. p. 1–8.

LACLAU, C.; NADIF, M. Hard and fuzzy diagonal co-clustering for document-term partitioning. *Neurocomputing*, Elsevier, v. 193, p. 133–147, 2016.

LEE, D.; SEUNG, H. S. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, v. 13, 2000.

LEVINE, E.; DOMANY, E. Unsupervised estimation of cluster validity using resampling. *Neural Computation*, v. 13, p. 2573–2593, 2001.

LI, J.; CHENG, K.; WANG, S.; MORSTATTER, F.; TREVINO, R. P.; TANG, J.; LIU, H. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 6, p. 94, 2018.

LI, M.; WEN, L.; CHEN, F. A novel collaborative filtering recommendation approach based on soft co-clustering. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 561, p. 125140, 2021.

LIU, Y.; WU, S.; LIU, Z.; CHAO, H. A fuzzy co-clustering algorithm for biomedical data. *Plos one*, Public Library of Science San Francisco, CA USA, v. 12, n. 4, p. e0176536, 2017.

LONG, B.; ZHANG, Z.; YU, P. S. Co-clustering by block value decomposition. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. [S.l.: s.n.], 2005. p. 635–640.

MAÂTOUK, O.; AYADI, W.; BOUZIRI, H.; DUVAL, B. Evolutionary local search algorithm for the biclustering of gene expression data based on biological knowledge. *Applied Soft Computing*, Elsevier, v. 104, p. 107177, 2021.

MACQUEEN, J. Classification and analysis of multivariate observations. In: *5th Berkeley Symp. Math. Statist. Probability*. [S.l.: s.n.], 1967. p. 281–297.

MAHMOUDI, M. R.; BALEANU, D.; MANSOR, Z.; TUAN, B. A.; PHO, K.-H. Fuzzy clustering method to compare the spread rate of covid-19 in the high risks countries. *Chaos, Solitons & Fractals*, Elsevier, v. 140, p. 110230, 2020.

MERCER, B. J. Functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A*, The Royal Society, v. 209, n. 441-458, p. 415–446, 1909.

MULLER, K.-R.; MIKA, S.; RATSCH, G.; TSUDA, K.; SCHOLKOPF, B. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, IEEE, v. 12, n. 2, p. 181–201, 2001.

MURTAGH, F. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, Oxford University Press, v. 26, n. 4, p. 354–359, 1983.

NAGPAL, A.; JATAIN, A.; GAUR, D. Review based on data clustering algorithms. In: IEEE. *2013 IEEE conference on information & communication technologies*. [S.l.], 2013. p. 298–303.

NAYAK, J.; NAIK, B.; BEHERA, H. Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014. *Computational intelligence in data mining-volume 2*, Springer, p. 133–149, 2015.

NEMENYI, P. B. *Distribution-free multiple comparisons*. [S.l.]: Princeton University, 1963.

NG, R. T.; HAN, J. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, IEEE, v. 14, n. 5, p. 1003–1016, 2002.

NIE, F.; SHI, S.; LI, X. Auto-weighted multi-view co-clustering via fast matrix factorization. *Pattern Recognition*, Elsevier, v. 102, p. 107207, 2020.

PAL, N. R.; SARKAR, K. What and when can we gain from the kernel versions of c-means algorithm? *IEEE Transactions on Fuzzy Systems*, IEEE, v. 22, n. 2, p. 363–379, 2013.

PEREIRA, A. L. V.; HRUSCHKA, E. R. Simultaneous co-clustering and learning to address the cold start problem in recommender systems. *Knowledge-Based Systems*, Elsevier, v. 82, p. 11–19, 2015.

PHAM, N. V.; PHAM, L. T.; PEDRYCZ, W.; NGO, L. T. Feature-reduction fuzzy co-clustering approach for hyper-spectral image analysis. *Knowledge-Based Systems*, Elsevier, v. 216, p. 106549, 2021.

RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, Taylor & Francis, v. 66, n. 336, p. 846–850, 1971.

ROCCI, R.; VICHI, M. Two-mode multi-partitioning. *Computational Statistics & Data Analysis*, Elsevier, v. 52, n. 4, p. 1984–2003, 2008.

RODRIGUES, A. K.; OSPINA, R.; FERREIRA, M. R. Adaptive kernel fuzzy clustering for missing data. *Plos one*, Public Library of Science San Francisco, CA USA, v. 16, n. 11, p. e0259266, 2021.

RODRÍGUEZ, S. I.; CARVALHO, F. d. A. de. Fuzzy clustering algorithm with automatic variable selection and entropy regularization. In: IEEE. *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2017. p. 1–6.

RODRIGUEZ, S. I.; CARVALHO, F. d. A. de. Fuzzy clustering algorithms with distance metric learning and entropy regularization. *Applied Soft Computing*, Elsevier, v. 113, p. 107922, 2021.

RODRÍGUEZ, S. I. R. *A fuzzy partitional clustering algorithm with adaptative euclidean distance and entropy regularization*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2018.

RODRÍGUEZ, S. I. R. Clustering algorithms with new automatic variables weighting. Universidade Federal de Pernambuco, 2022.

RODRÍGUEZ, S. I. R.; CARVALHO, F. d. A. T. de. Clustering interval-valued data with automatic variables weighting. In: IEEE. *2019 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2019. p. 1–8.

RODRÍGUEZ, S. I. R.; CARVALHO, F. d. A. T. de. A new fuzzy clustering algorithm for interval-valued data based on city-block distance. In: IEEE. *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2019. p. 1–6.

RUSPINI, E. H.; BEZDEK, J. C.; KELLER, J. M. Fuzzy clustering: A historical perspective. *IEEE Computational Intelligence Magazine*, IEEE, v. 14, n. 1, p. 45–55, 2019.

SÁ, J. N. A. de; FERREIRA, M. R.; CARVALHO, F. d. A. de. Kernel-based fuzzy co-clustering in feature space with automated variable weighting. In: IEEE. *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2022. p. 1–8.

SAXENA, A.; PRASAD, M.; GUPTA, A.; BHARILL, N.; PATEL, O. P.; TIWARI, A.; ER, M. J.; DING, W.; LIN, C.-T. A review of clustering techniques and developments. *Neurocomputing*, Elsevier, v. 267, p. 664–681, 2017.

SCHÖLKOPF, B.; SMOLA, A.; MÜLLER, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, MIT Press, v. 10, n. 5, p. 1299–1319, 1998.

SHEN, H.; YANG, J.; WANG, S.; LIU, X. Attribute weighted mercer kernel based fuzzy clustering algorithm for general non-spherical datasets. *Soft Computing*, Springer, v. 10, n. 11, p. 1061–1073, 2006.

STEINHAUS, H. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, v. 1, n. 804, p. 801, 1956.

STREHL, A.; GHOSH, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, v. 3, n. Dec, p. 583–617, 2002.

TANG, Y.; REN, F.; PEDRYCZ, W. Fuzzy c-means clustering through ssim and patch for image segmentation. *Applied Soft Computing*, Elsevier, v. 87, p. 105928, 2020.

TJHI, W.-C.; CHEN, L. Flexible fuzzy co-clustering with feature-cluster weighting. In: IEEE. *2006 9th International Conference on Control, Automation, Robotics and Vision*. [S.l.], 2006. p. 1–6.

TSAI, C.-Y.; CHIU, C.-C. Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm. *Computational statistics & data analysis*, Elsevier, v. 52, n. 10, p. 4658–4672, 2008.

UBUKATA, S.; MOURI, K.; HONDA, K. Basic consideration of collaborative filtering based on rough co-clustering induced by multinomial mixture models. In: IEEE. *2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*. [S.l.], 2022. p. 1–6.

VERMA, O. P.; HOODA, H. A novel intuitionistic fuzzy co-clustering algorithm for brain images. *Multimedia Tools and Applications*, Springer, v. 79, n. 41-42, p. 31517–31540, 2020.

WANG, J.; DENG, Z.; CHOI, K.-S.; JIANG, Y.; LUO, X.; CHUNG, F.-L.; WANG, S. Distance metric learning for soft subspace clustering in composite kernel space. *Pattern Recognition*, Elsevier, v. 52, p. 113–134, 2016.

WANG, Z.; YANG, S.; KOGA, Y.; CORBETT, S. E.; SHEA, C. V.; JOHNSON, W. E.; YAJIMA, M.; CAMPBELL, J. D. Celda: A bayesian model to perform co-clustering of genes into modules and cells into subpopulations using single-cell rna-seq data. *NAR Genomics and Bioinformatics*, Oxford University Press, v. 4, n. 3, p. lqac066, 2022.

WEGMANN, M.; ZIPPERLING, D.; HILLENBRAND, J.; FLEISCHER, J. A review of systematic selection of clustering algorithms and their evaluation. *arXiv preprint arXiv:2106.12792*, 2021.

WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 196–202.

WU, T.; BENSON, A. R.; GLEICH, D. F. General tensor spectral co-clustering for higher-order data. *Advances in Neural Information Processing Systems*, v. 29, p. 2559–2567, 2016.

XIANG, D.; TANG, T.; HU, C.; LI, Y.; SU, Y. A kernel clustering algorithm with fuzzy factor: Application to sar image segmentation. *IEEE Geoscience and remote sensing letters*, IEEE, v. 11, n. 7, p. 1290–1294, 2013.

XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science*, Springer, v. 2, n. 2, p. 165–193, 2015.

YANG, D.; FEI, R.; YAO, J.; GONG, M. Two-stage sar image segmentation framework with an efficient union filter and multi-objective kernel clustering. *Applied Soft Computing*, Elsevier, v. 44, p. 30–44, 2016.

ZAIT, M.; MESSATFA, H. A comparative study of clustering methods. *Future Generation Computer Systems*, Elsevier, v. 13, n. 2-3, p. 149–159, 1997.

ZHANG, D.-Q.; CHEN, S.-C. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural processing letters*, Springer, v. 18, n. 3, p. 155–162, 2003.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, ACM New York, NY, USA, v. 25, n. 2, p. 103–114, 1996.

ZHOU, S.; GAN, J. Q. Mercer kernel, fuzzy c-means algorithm, and prototypes of clusters. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2004. p. 613–618.

APÊNDICE A - ARTIGO FUZZ-IEEE 2022

Esse apêndice apresenta a página da web da conferência FUZZ-IEEE onde o artigo *Kernel-based Fuzzy Co-clustering in Feature Space with Automated Variable Weighting* (SÁ; FERREIRA; CARVALHO, 2022) foi publicado e a primeira página do artigo.

IEEE.org | IEEE Xplore | IEEE SA | IEEE Spectrum | More Sites
Cart | Create Account | Personal Sign In
IEEE Xplore® Browse My Settings Help
Access provided by: Universidad Federal de Pernambuco
Sign Out
IEEE
All
ADVANCED SEARCH
Conferences > 2022 IEEE International Confe...

Kernel-based Fuzzy Co-clustering in Feature Space with Automated Variable Weighting

Publisher: IEEE
Cite This
PDF

José Nataniel A. de Sá ; Marcelo R.P. Ferreira ; Francisco de A.T. de ... All Authors

27
Full
Text Views

Abstract
Document Sections
I. Introduction
II. Fuzzy Double Kmeans (FDK)
III. Kernel-Based Fuzzy Co-Clustering
IV. Numerical Experiments
V. Conclusion
Authors
Figures
References
Keywords
Metrics
Footnotes

Abstract:
Kernel functions have been used successfully in clustering algorithms to deal with the separability of clusters efficiently. Bringing this idea to co-clustering, we propose two kernel-based fuzzy co-clustering algorithms based on the fuzzy double Kmeans (FDK). The first proposed algorithm, the Gaussian kernel fuzzy double Kmeans (GKFDK), is based on FDK and computes the cluster prototypes in the original feature space. The second algorithm, the Weighted gaussian kernel fuzzy double Kmeans (WGKFDK), is an extension of the GKFDK with automated variable weighting, that distinguishes the relevance of the variables in each cluster. Experiments performed with both synthetic and real data, in comparison with previous state-of-the-art co-clustering algorithms, showed the effectiveness of the proposed algorithms.

Published in: 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)

Date of Conference: 18-23 July 2022
INSPEC Accession Number: 22050261
Date Added to IEEE *Xplore*: 14 September 2022
DOI: 10.1109/FUZZ-IEEE55066.2022.9882838
ISBN Information:
Publisher: IEEE
ISSN Information:
Conference Location: Padua, Italy
Funding Agency:

SECTION I.
Introduction

Clustering methods are important tools used in several areas such as pattern recognition, image processing, data mining, information retrieval, and so forth. A variety of clustering algorithms as *K*-means, self-organizing maps and hierarchical clustering were developed to find an optimal partition of objects in a dataset. In the last few years, there are other methods, called co-clustering (biclustering,

¡Taller de autoría virtual de IEEE el 16 de noviembre!
Aprenda a publicar en acceso abierto con IEEE asistiendo a este taller gratuito.
¡Regístrese hoy!

More Like This
Study on flood clustering and recognition methods based on fuzzy set theory
2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)
Published: 2011
A Generic Mathematical Model Based on Fuzzy Set Theory for Frequency Reuse in Cellular Networks
IEEE Journal on Selected Areas in Communications
Published: 2013
Show More

IEEE Signal Processing Society | IEEE
Get Published in the *IEEE Open Journal of Signal Processing*
Learn More

PDF
Help
Feedback

Kernel-based Fuzzy Co-clustering in Feature Space with Automated Variable Weighting

José Nataniel A. de Sá
Universidade Federal de Pernambuco
Centro de Informática - CIn/UFPE
Av. Jornalista Aníbal Fernandes, s/n
CEP 50.740-560 - Recife, PE, Brazil
E-mail: jnas@cin.ufpe.br

Marcelo R.P. Ferreira
Universidade Federal da Paraíba
Departamento de Estatística - DE/UFPB
Conj. Pres. Castelo Branco III
CEP 58.050-585 - João Pessoa, PB, Brazil
E-mail: marcelo@de.ufpb.br

Francisco de A.T. de Carvalho
Universidade Federal de Pernambuco
Centro de Informática - CIn/UFPE
Av. Jornalista Aníbal Fernandes, s/n
CEP 50.740-560 - Recife, PE, Brazil
E-mail: fatc@cin.ufpe.br

2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) | 978-1-6654-6710-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/FUZZ-IEEE55066.2022.9882838

***Abstract*—Kernel functions have been used successfully in clustering algorithms to deal with the separability of clusters efficiently. Bringing this idea to co-clustering, we propose two kernel-based fuzzy co-clustering algorithms based on the *fuzzy double Kmeans* (FDK). The first proposed algorithm, the *Gaussian kernel fuzzy double Kmeans* (GKFDK), is based on FDK and computes the cluster prototypes in the original feature space. The second algorithm, the *Weighted gaussian kernel fuzzy double Kmeans* (WGKFDK), is an extension of the GKFDK with automated variable weighting, that distinguishes the relevance of the variables in each cluster. Experiments performed with both synthetic and real data, in comparison with previous state-of-the-art co-clustering algorithms, showed the effectiveness of the proposed algorithms.**

I. Introduction

Clustering methods are important tools used in several areas such as pattern recognition, image processing, data mining, information retrieval, and so forth. A variety of clustering algorithms as K-means, self-organizing maps and hierarchical clustering were developed to find an optimal partition of objects in a dataset. In the last few years, there are other methods, called co-clustering (biclustering, double clustering or block-clustering), that have gained prominence, by simultaneously grouping objects and variables, being able to reorganize the initial data matrix into homogeneous blocks (co-clusters). These co-clusters consists of a set of objects and a set of variables whose elements are similar.

By reorganizing the data matrix, co-clustering methods can provide insights about the association between objects and variables, specifically when the data have high dimensionality. They are also able to perform dimensionality reduction, because they provide a lower-dimensional matrix consisting of representative values (prototypes) of each block. Furthermore, these algorithms intrinsically perform variable selection according to the capability to discriminate the variables that are relevant for the object clusters.

For these reasons, co-clustering methods have been developed under different approaches, such as spectral co-clustering [1], bayesian co-clustering [2], fuzzy co-clustering [3], etc. As for fields of applications, co-clustering has an extensive range. In the gene expression data these models have the ability to identify groups of genes (variables) that behave in the same way under a subset of conditions (objects) [4], [5]. In text mining, several methods exploit the duality between documents and terms [6], [7]. In recommender systems, product and customer relationships have been studied [8], [9].

A common problem in clustering algorithms is related to the separability of classes, especially when they are based on the Euclidean distance. In data where the clusters have non-hyperspherical shapes and/or linearly non-separable patterns, these algorithms can perform poorly and find an unrepresentative cluster structure. Because of this limitation several methods that are able to handle complex data have been proposed, among them, kernel-based clustering algorithms [10]. These algorithms produce hypersurfaces with nonlinear separation between clusters. Since the seminal work of Girolami [11], a variety of kernel-based clustering algorithms have been proposed, such as kernel-based fuzzy clustering [12], kernel-based Self-Organizing Maps (SOM) [13] and Kernel subtrative clustering [14]. These algorithms have been developed under two main approaches: clustering in the feature space, where the cluster prototypes are computed in the original feature space and the distances between the objects and the prototypes are obtained by means of the kernel trick; and clustering in kernel space, in which the centroids cannot be computed because they are in the kernel space but the distances between the objects and the prototypes can be computed by means of the kernel trick. Several works have shown the efficiency of the kernel-based clustering algorithms in a variety of real-world problems [15]–[17].

In the fuzzy approach, objects and variables no longer belong to one particular cluster. Instead, they may belong to all clusters with a certain fuzzy membership degree, being more suitable when there is overlapping in the data. Despite the successful application of fuzzy co-clustering in several areas, to the best of our knowledge, no work has turned attention to addressing the limitation of data separability, specifically using the kernel function approach.

In this article, we propose two distinct kernel-based fuzzy co-clustering algorithms in feature space. The first one is a modified version of the *Fuzzy Double Kmeans* (FDK) [18]. The second algorithm was built upon the first one and introduces the concept of variable weights in each cluster, being

978-1-6654-6710-0/22/$31.00 ©2022 IEEE
Authorized licensed use limited to: Universidad Federal de Pernambuco. Downloaded on November 14,2022 at 11:25:45 UTC from IEEE Xplore. Restrictions apply.