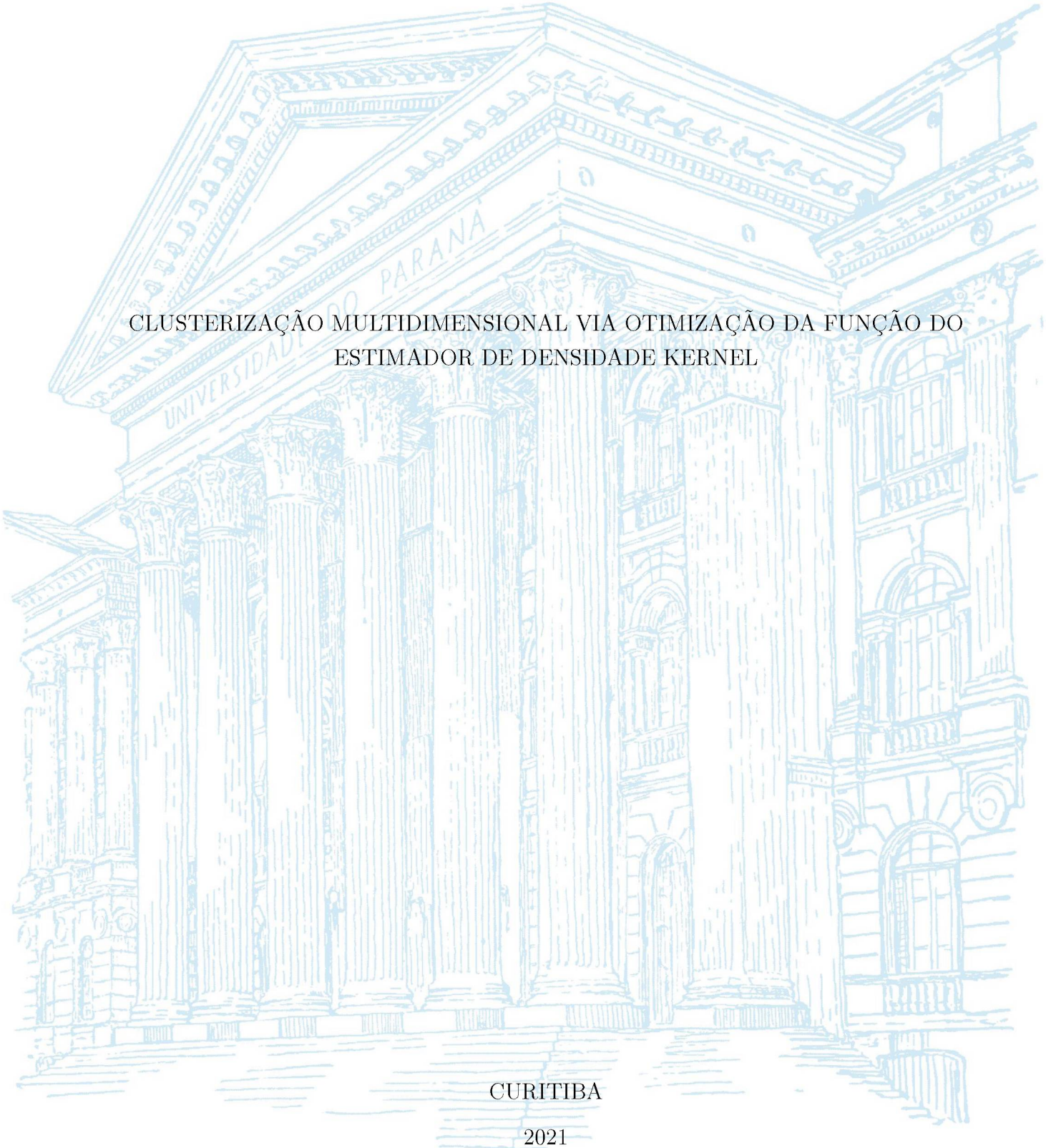
UNIVERSIDADE FEDERAL DO PARANÁ

DIRCEU SCALDELAI



CLUSTERIZAÇÃO MULTIDIMENSIONAL VIA OTIMIZAÇÃO DA FUNÇÃO DO ESTIMADOR DE DENSIDADE KERNEL

CURITIBA

2021

DIRCEU SCALDELAI

CLUSTERIZAÇÃO MULTIDIMENSIONAL VIA OTIMIZAÇÃO DA FUNÇÃO DO ESTIMADOR DE DENSIDADE KERNEL

Tese apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática, dos Setores de Tecnologia e de Ciências Exatas da Universidade Federal do Paraná, como requisito parcial à obtenção do título de Doutor em Ciências.

Orientador: Prof. Dr. Luiz Carlos Matioli
Coorientadora: Profa. Dra. Solange Regina dos Santos

CURITIBA

2021

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

S281c Scaldelai, Dirceu
Clusterização multidimensional via otimização da função do estimador de densidade Kernel [recurso eletrônico] / Dirceu Scaldelai – Curitiba, 2021.

Tese - Universidade Federal do Paraná, Setores de Tecnologia e de Ciências Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática.

Orientador: Prof. Dr. Luiz Carlos Matioli
Coorientadora: Profa. Dra. Solange Regina dos Santos

1. Algorítimos. 2. Estimador de densidade Kernel. 3. Otimização. I. Universidade Federal do Paraná. II. Matioli, Luiz Carlos. III. Santos, Solange Regina dos. IV. Título.

CDD: 005.1

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO MÉTODOS NUMÉRICOS EM ENGENHARIA - 40001016030P0

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **DIRCEU SCALDELAI** intitulada: **CLUSTERIZAÇÃO MULTIDIMENSIONAL VIA OTIMIZAÇÃO DA FUNÇÃO DO ESTIMADOR DE DENSIDADE KERNEL**, sob orientação dos professores SOLANGE REGINA DOS SANTOS e LUIZ CARLOS MATIOLI, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa. A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 09 de Dezembro de 2021.

LUIZ CARLOS MATIOLI
Presidente da Banca Examinadora

CESAR AUGUSTO TACONELI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

LUCAS GARCIA PEDROSO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

SISSY DA SILVA SOUZA
Avaliador Externo (UNIVERSIDADE FEDERAL DO PIAUÍ)

ADEMIR ALVES RIBEIRO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

AGRADECIMENTOS

A Deus em primeiro lugar, pois sem Ele nada seria possível.

À minha esposa, Raquel, por todo amor, compreensão e apoio na conquista desta etapa tão importante em minha vida.

Às minhas filhas, Samara e Laura, por compreenderem os muitos momentos de ausência afetiva, proveniente do ardo tempo dedicado a essa pesquisa.

Aos meus pais, pelo apoio e presença física e emocional dado à minha esposa e filhas nos momentos de minha ausência.

Ao meus orientadores, professores Matioli e Solange, que muito mais que orientadores, foram grandes amigos, que me apoiaram, me motivaram, me incentivaram por todo o percurso.

À Universidade Federal do Paraná e ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia, pela oportunidade de cursar o doutorado.

Aos professores do Programa de Pós-Graduação em Métodos Numéricos em Engenharia, pelos valiosos ensinamentos transmitidos.

À Universidade Estadual do Paraná, Campus Campo Mourão, por me possibilitar condições necessárias para a conclusão deste curso.

"Descobrir consiste em olhar para o que todo mundo está vendo e pensar uma coisa diferente".

— Roger Von Oech

RESUMO

Neste trabalho, apresentamos três novos algoritmos para clusterização de dados multidimensionais baseados na múltipla otimização da função do estimador de densidade kernel Gaussiano, o MulticlusterKDE, o AdditiveclusterKDE e o TreeKDE. Os algoritmos propostos possuem a principal vantagem de não exigirem, a priori, o número de clusters, além de serem simples de implementar, bem definidos e sempre convergirem em um número finito de etapas, independentemente do conjunto de dados a ser agrupado. Além dos algoritmos, propomos ainda uma nova métrica de validação interna de clusterização, o índice de Densidade da Clusterização (índice CD), baseado na máxima razão entre a dispersão interna dos clusters e a separação entre centroides. Visando facilitar a compreensão dos algoritmos propostos, os quais foram implementados no *Software R*, descrevemos detalhadamente suas metodologias e procedimentos, exemplificando cada um dos seus passos por meio de um problema simples, bidimensional, com um número reduzido de observações e uma estrutura de grupos bem definida. Na sequência, realizamos experimentos numéricos comparando os três algoritmos propostos a alguns outros já consagrados na literatura, sendo: K-means, K-medoids, CLARA, GMM, DIANA, CLINK, PdfCluster e DBSCAN. Os experimentos conduzidos tiveram três vertentes: explorar e analisar as diferentes características dos algoritmos de clusterização aplicados em problemas bidimensionais; medir a qualidade das clusterizações propiciadas pelos algoritmos em problemas de dimensões variadas; avaliar o comportamento dos algoritmos de clusterização aplicados a problemas de classificação. Os dados utilizados consistem em problemas práticos provenientes da literatura e de pacotes do *Software R*, além de problemas cujos dados foram gerados randomicamente. Os resultados numéricos evidenciaram que os algoritmos MulticlusterKDE, AdditiveclusterKDE e TreeKDE são promissores e competitivos para a clusterização de dados multidimensionais, quando comparados aos outros algoritmos da literatura supra citados, uma vez que esses apresentam bons desempenhos, não necessitam da especificação do número de clusters e conseguem identificar grupos de alta densidade. Quanto ao índice CD, resultados preliminares revelaram que quando comparado com métricas já consagradas na literatura, tais como o índice DB e o coeficiente de silhueta, é eficiente para avaliar clusterização de dados multidimensionais uma vez que apresentou uma concordância substancial com o índice DB a um custo de execução similar, e uma concordância significativa com o coeficiente de silhueta a um custo execução consideravelmente menor, comprovando sua qualidade como métrica de validação interna para clusterização de dados multidimensionais.

Palavras-chave: Clusterização; Estimador de Densidade Kernel; Otimização.

ABSTRACT

In this thesis we present three new algorithms for multidimensional data clustering based on multiple optimization of the Gaussian kernel density estimator function, the MulticlusterKDE, AdditiveclusterKDE and TreeKDE. The suggested algorithms have the main advantage of not requiring, a priori, the number of clusters, besides the fact that they are simple to implement, are well defined and always converge in a finite number of steps, regardless of the dataset to be clustered. In addition to the algorithms, we also propose a new internal clustering validation metric, the Clustering Density index (CD index), based on the maximum ratio between the internal dispersion of clusters and the separation between centroids. In order to facilitate the understanding of the suggested algorithms, which were implemented in the software R, we describe in detail its methodologies and procedures, exemplifying each of its steps by means of a simple two-dimensional problem, with a reduced number of observations and a well-defined structure of groups. After this, we performed numerical tests by comparing the three proposed algorithms with some others already established in the literature, as follows: K-means, K-medoids, CLARA, GMM, DIANA, CLINK, PdfCluster and DBSCAN. The experiments conducted had three main aspects: to explore and analyze the different characteristics of clustering algorithms applied to two-dimensional problems; to measure the quality of clustering provided by the algorithms in problems of different dimensions; to evaluate the performance of clustering algorithms applied to classification problems. The input data consist of practice problems sourced from the literature and from Software R packages, as well as problems whose data were randomly generated. The numerical results showed that MulticlusterKDE, AdditiveclusterKDE and TreeKDE algorithms are promising and competitive for multidimensional data clustering when compared with other algorithms in the abovementioned literature, since they have good performance, do not require specification of the number of clusters and can identify high-density clusters. Regarding the CD index, preliminary results showed that when compared to metrics already established in the literature, such as the DB index and the silhouette coefficient, it is an accurate way to evaluate the clustering of multidimensional data since it showed substantial agreement with the DB index at a similar execution cost, and significant agreement with the silhouette coefficient at a considerably lower execution cost, hence proving its quality as an internal validation metric for multidimensional data clustering.

Key-words: Clustering; Kernel Density Estimator; Optimization

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

KDE	*Kernel Density Estimation* (Estimador de Densidade Kernel)
f.d.p.	Função densidade de probabilidade
MSE	*Mean Square Error* (Erro quadrático médio)
MISE	*Mean Integrated Squared Error* (Erro quadrático médio integrado)
AMISE	*Asymptotic Mean Integrate Square Error* (Erro quadrático médio integrado assintótico)
PAM	*Partitioning Around Medoids* (k-medoids)
CLARA	*Clustering LARge Applications*
DIANA	*DIvisive ANAlysis Clustering*
SLINK	*Single-LINKage*
CLINK	*Complete LINKage*
STING	*STatistical INformation Grid*
CLIQUE	*CLustering In QUEst*
GMM	*Gaussian Mixture Model*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
OPTICS	*Ordering Points To Identify the Clustering Structure*
DPC	*Density Peak Clustering*
FKNN-DPC	*Fuzzy weighted K-Nearest Neighbors Density Peak Clustering*

SUMÃRIO

1 INTRODUÇÃO

Com o avanço e o uso massivo de tecnologias, a quantidade de dados gerada e disponibilizada tem crescido exponencialmente. No entanto, para que essa grande quantidade de dados forneça informações que possam ser transformadas em conhecimentos úteis e relevantes é preciso realizar análises essenciais para a sua correta interpretação.

O processo de explorar grandes quantidades de dados à procura de padrões, regras ou sequências de informações, para detectar correlações entre as variáveis, faz parte do campo da ciência denominada Ciências de dados, do inglês *Data science*. De acordo com Mount e Zumel (2019) e Said e Torra (2019), a Ciência de Dados se concentra na implementação de decisões fundamentadas em dados e no gerenciamento de suas consequências sendo uma prática interdisciplinar com forte conexão com Estatística, Aprendizado de máquina e *Big data*.

Para Mount e Zumel (2019) e Kassambara (2017) a clusterização, do inglês *clustering*, é um dos campos de estudos da Ciências de Dados que busca identificar padrões ou grupos[1] de objetos semelhantes em um conjunto de dados de interesse, sendo utilizada nos mais variados campos da pesquisa científica, tendo seus resultados frequentemente utilizados para prever, analisar e replicar fenômenos, além de fomentar novas pesquisas.

A quantidade de algoritmos de clusterização é vasta na literatura, com as mais variadas metodologias, apresentando fatores positivos e negativos no processo de agrupamento multidimensional. A maioria desses algoritmos é, de alguma forma, dependente da definição a priori do número de grupos que, em muitas circunstâncias, é desconhecida, não especificada e um dos focos da análise de dados. Outra característica de inúmeros algoritmos de clusterização é a de formação de grupos globulares, isto é, grupos inscritos em hiperesferas, determinadas por raios baseados em distâncias, principalmente a Euclidiana. Por isso, a busca por novas metodologias capazes de suprir as dificuldades em determinar de forma autônoma o número de grupos e/ou com formas arbitrárias, ainda é um campo fértil para a pesquisa sobre clusterização de dados multidimensionais.

Neste trabalho, propomos três novos algoritmos para agrupar dados multidimensionais, baseados na múltipla otimização da função do estimador de densidade kernel, do inglês *Kernel Density Estimation* (KDE). Denominamos o primeiro algoritmo de "MulticlusterKDE", em virtude desse realizar múltiplas otimizações da função KDE, com o intuito de determinar os centroides de cada grupo. Tal algoritmo tem a vantagem de determinar o número de grupos de forma autônoma e ser dependente apenas de um escalar

[1] No decorrer do texto também é adotada a expressão cluster para fazer referência a cada grupo proveniente do agrupamento de dados, da mesma forma que clusterização e agrupamento são considerados sinônimos

como parâmetro de entrada, utilizado para o cálculo da matriz largura de banda da função KDE, além é claro, do próprio conjunto de dados a ser agrupado.

O segundo algoritmo, por ter como base a adição de um novo grupo a cada passo, é denominado "AdditiveclusterKDE". A ideia central desse algoritmo é trabalhar com amostras e, a partir dessas, realizar múltiplas otimizações da função KDE com o objetivo de obter um conjunto de centroides que serão utilizados para clusterização do problema (população). Assim como o Algoritmo MulticlusterKDE, o Algoritmo AdditiveclusterKDE não necessita, a priori, do número de grupos, sendo esse determinado por meio da adição de um novo grupo e pela avaliação da qualidade dessa inserção a cada etapa do algoritmo. O algoritmo é interrompido quando a adição de novos grupos não produz melhoras na clusterização.

O terceiro algoritmo, denominado "TreeKDE", é baseado numa estrutura de árvore de decisão associada com a otimização da função do estimador de densidade kernel unidimensional. O Algoritmo TreeKDE utiliza a otimização da função KDE unidimensional para criar partições do espaço de forma semelhante a uma estrutura de árvore de decisão. As KDEs unidimensionais são construídas com base na projeção ortogonal do conjunto de dados sobre os eixos coordenados. A principal característica desse algoritmo é a determinação do número de grupos de forma autônoma com formatos arbitrários inscritos em regiões hiper-retangulares paralelas aos eixos coordenados.

Além dos algoritmos mencionados, propomos também uma nova métrica de validação interna para clusterização multidimensional, baseada na máxima razão entre a dissimilaridade intra-grupos e a intergrupos, denominada índice de Densidade de Clusterização ou, simplesmente, "índice CD".

Com o intuito de melhor apresentar nossas contribuições, o presente trabalho está organizado em 5 capítulos. No Capítulo 2, apresentamos os conceitos fundamentais sobre clusterização, estimador de densidade kernel, árvore de decisão e otimização de funções reais, os quais são essenciais para o entendimento dos algoritmos propostos. O Capítulo 3 é destinado exclusivamente para a apresentação do índice CD e dos algoritmos MulticlusterKDE, AdditiveclusterKDE e TreeKDE. Experimentos numéricos são apresentados no Capítulo 4 e, por fim, no Capítulo 5, apresentamos as observações finais e o delineamento de propostas de continuidade da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é destinado a uma breve apresentação dos quatro principais pilares teóricos envolvidos nessa pesquisa. São eles: 1. métodos de clusterização e métricas de validação; 2. estimador de densidade kernel; 3. árvore de decisão e 4. métodos de otimização.

2.1 MÉTODOS DE CLUSTERIZAÇÃO

A realização de um agrupamento (do inglês *clustering*) de um conjunto de dados consiste na identificação de grupos (do inglês *cluster*) onde os elementos pertencentes a um mesmo grupo devem possuir características semelhantes enquanto elementos de grupos distintos devem exibir características diferentes. Logo, a clusterização consiste num processo de partição do conjunto de dados em subconjuntos disjuntos de forma a minimizar a dissimilaridade intra-grupos[1] e maximizar a intergrupos[2].

A clusterização, segundo James et al. (2013) e Kassambara (2017), refere-se ao método de aprendizado de máquina não-supervisionado cuja tarefa é a de inferir uma estrutura oculta a partir de dados não rotulados por uma variável preditora pré-definida. Para Shah, Bhensdadia e Ganatra (2012), os algoritmos de clusterização podem ser classificados, de acordo com a metodologia empregada, em cinco vertentes principais, sendo elas: particionais, hierárquicos, baseados em densidade, em grade e em modelos.

De acordo com Ahuja e Bal (2014), os métodos de particionamento são os mais simples entre os métodos de clusterização. Para Kaufman e Rousseeuw (2009), a ideia básica desses métodos é a de particionar o conjunto de dados em k grupos, atendendo a dois princípios: 1. cada grupo deve conter pelo menos um objeto (aqui entendemos por objeto qualquer observação que possa ser expressa como um vetor $x \in \mathbb{R}^n$); 2. cada objeto deve pertencer a exatamente um único grupo, sendo esse o mais próximo, segundo um critério de dissimilaridade baseada em distância, por exemplo, a distância Euclidiana. De acordo com Rodriguez e Laio (2014), a abordagem de particionamento do conjunto não é capaz de detectar aglomerados não esféricos ou não-globulares. Uma das grandes desvantagens dos algoritmos de particionamento é a necessidade de se conhecer a priori o número de grupos, visto que essa informação é, para a maioria dos problemas pesquisados, uma pergunta a ser respondida.

Um dos métodos de particionamento mais conhecido é o Algoritmo K-means, proposto por MacQueen (1967), o qual consiste em um processo iterativo constituído de duas etapas, em que na primeira o objetivo é determinar um conjunto de k centroides

[1] Mede o quanto os objetos dentro de um grupo estão próximos.
[2] Mede o quanto cada elemento de um grupo está afastado dos elementos de outros grupos.

(centros dos grupos), sendo $k \in \mathbb{Z}_+^*$ um parâmetro pré estabelecido e, na segunda, alocar os objetos do conjunto de dados a estes centroides, pelo critério de mínima distância. Ao fim da designação dos objetos aos grupos, os k centroides são atualizados tornando-se os baricentros[3] dos elementos dos grupos e o processo é reiniciado. Quando não houver mais mudanças na posição dos centroides ou a mudança for mínima, o algoritmo é encerrado.

Outro método bastante difundido na literatura é o Algoritmo K-medoids ou PAM (*Partitioning Around Medoids*), proposto por Kaufman e Rousseeuw (1987). Segundo Kaufman e Rousseeuw (2009), esse algoritmo é baseado na busca de k objetos representativos entre os objetos do conjunto de dados. Ele é um algoritmo semelhante ao K-means, pois também busca minimizar a distância entre os objetos designados aos grupos com seus respectivos centroides, cuja diferença ocorre pela forma de escolha dos candidatos a centros. No K-means os objetos determinados como centroides são os respectivos baricentros dos grupos e não pertencem necessariamente aos dados do problema, enquanto que no K-medoids os objetos designados como centros (*medoids*) obrigatoriamente pertencem ao conjunto de dados, escolhidos de forma combinatória, tornando o método mais robusto a influência de *outliers* quando comparado ao K-means.

Ainda segundo Kaufman e Rousseeuw (2009) o Algoritmo K-medoids produz resultados satisfatórios em muitas situações, no entanto, não é prático para agrupar grandes conjuntos, devido à sua natureza combinatória na busca pelo melhor conjunto de k *medoids*. Por causa dessa dificuldade, Kaufman e Rousseeuw (1987) propuseram um novo método denominado CLARA (*Clustering LARge Applications*).

O Algoritmo CLARA, segundo Kaufman e Rousseeuw (2009), trabalha com múltiplas amostras representativas (no mínimo 5), extraídas do conjunto de dados de forma aleatória, ao invés de todo o conjunto. O tamanho das amostras depende do número de grupos, em que para uma clusterização com k grupos o tamanho das amostras é dado por $40+2k$. Sobre cada uma das amostras são determinados k *medoids* usando o algoritmo PAM com posterior atribuição de todos os objetos do conjunto ao *medoids* mais próximo. A distância média obtida para atribuição é usada como uma medida da qualidade da clusterização, sendo a que apresentar a menor distância média é escolhida como resultado do algoritmo.

Uma outra vertente de algoritmos de clusterização são os métodos hierárquicos, os quais decompõem o conjunto de dados em vários níveis de particionamento, geralmente representados por um dendrograma. O dendrograma é uma árvore que divide os dados recursivamente em subconjuntos até que um critério de finalização seja atendido. Cada nó na árvore representa um grupo, podendo ser utilizado a partir das folhas para a raiz (abordagem aglomerativa) ou, ainda, a partir da raiz para as folhas (abordagem divisiva)

[3] Baricentro é o centro de um conjunto de pontos, ou seja, dados $x_1, \ldots, x_m \in \mathbb{R}^n$ pontos, o baricentro é o ponto $\overline{x} = \frac{x_1+\ldots+x_m}{m}$.

pela fusão ou divisão de grupos em cada etapa, Ester et al. (1996). Embora algoritmos hierárquicos possam ser muito eficazes na descoberta de padrões, estes possuem a característica de requerer critérios de finalização do processo de fusão ou divisão. O algoritmo DIANA (*DIvisive ANAlysis clustering*), proposto por Kaufman e Rousseeuw (2009), é um exemplo de algoritmo hierárquico com abordagem divisiva, enquanto os algoritmos SLINK(*Single-LINKage*, Sibson (1973)) e CLINK (*Complete LINKage*, Defays (1977)) são exemplos de algoritmos hierárquico com abordagem aglomerativa.

De acordo com Han, Kamber e Pei (2011), métodos baseados em grade dividem o espaço, no qual o conjunto de dados esteja inserido, em um número finito de células, independentemente da distribuição dos objetos de entrada, quantizando o espaço em um número finito de células que formam uma estrutura de grade na qual todas as operações de agrupamento são realizadas. A principal vantagem dessa abordagem é o rápido tempo de processamento, o qual é independente do número de objetos e dependente do número de células em cada dimensão do espaço quantizado. Dois exemplos típicos de agrupamento por grade são: STING e CLIQUE.

O Algoritmo STING (*STatistical INformation Grid*), proposto por Wang et al. (1997), explora informações estatísticas armazenadas em células retangulares do espaço quantizado em grade, que são subdivididas de maneira hierárquica e recursiva, formando multiníveis de soluções dentro dessa estrutura. O algoritmo CLIQUE (*CLustering In QUEst*), proposto por Agrawal et al. (2005), busca encontrar grupos baseados em densidade dentro de subespaços obtidos do particionamento das dimensões em intervalos não sobrepostos, incorporando todos os objetos de dados em células.

Segundo Gan, Ma e Wu (2007), métodos de clusterização baseados em modelo buscam otimizar o ajuste entre os dados e modelos pré-definidos. Nessa metodologia, os dados são vistos como provenientes de uma mistura de distribuições de probabilidade, cada uma representando um grupo diferente. Assim, podemos esperar que um método de agrupamento específico funcione bem quando os dados estiverem em conformidade com o modelo. O modelo de mistura Gaussiana (GMM-*Gaussian Mixture Model*) é um bom exemplo dessa metodologia.

Uma mistura Gaussiana é uma função composta por k funções densidade de probabilidade Gaussianas, cada uma identificando um grupo. Para Gan, Ma e Wu (2007), o GMM fornece uma abordagem clássica e poderosa para a análise de agrupamentos, úteis para entender e sugerir critérios para clusterização.

Por fim, temos os métodos baseados em densidade, centrados na ideia de que centros de grupos são caracterizados por terem uma densidade mais alta que seus vizinhos incorporados ao mesmo cluster. Segundo Gan, Ma e Wu (2007) e Han, Kamber e Pei (2011) algoritmos de clusterização baseados em densidade são, em alguns casos, capazes

de encontrar grupos de formatos arbitrários, definidos como regiões densas separadas por regiões de baixa densidade. Os algoritmos DBSCAN e OPTICS são exemplos clássicos de algoritmos baseados em densidade.

Um dos mais importantes e conhecidos algoritmos baseado em densidade é o DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) proposto por Ester et al. (1996). A ideia chave desse algoritmo é que para cada ponto de um grupo, sua vizinhança, limitada por um raio, deve conter pelo menos um número mínimo de pontos, de tal forma que elementos do mesmo grupo sejam intimamente compactos e pontos fora da vizinhança sejam o mais distantes possível. Para Kassambara (2017), o DBSCAN pode descobrir grupos de diferentes formas e tamanhos, contendo ruídos (*outliers*), sendo seus resultados significativamente mais eficazes na descoberta de grupos não globulares do que algoritmos baseados em partição.

Outro algoritmo baseado em densidade é o OPTICS (*Ordering Points To Identify the Clustering Structure*) proposto por Ankerst et al. (1999). Esse algoritmo se assemelha ao DBSCAN, porém é menos sensível à variabilidade dos parâmetros de entrada. No OPTICS inicialmente são identificados elementos com maior densidade e, em seguida, os elementos do conjunto são ordenados linearmente de tal forma que, espacialmente, pontos mais próximos tornam-se vizinhos na ordenação.

Todos os algoritmos mencionados anteriormente são algoritmos clássicos de clusterização, amplamente difundidos no meio científico. Além desses, uma variedade enorme de trabalhos foram propostos, apresentando novas ideias e metodologias. A seguir apresentamos alguns desses trabalhos.

Menardi e Azzalini (2014) propuseram um algoritmo multivariado, intitulado PdfCluster, como uma extensão natural do procedimento de agrupamento baseado em densidade univariada desenvolvido por Azzalini e Torelli (2007). O Algoritmo PdfCluster, determina grupos que estão associados aos componentes conectados com densidade (estimada por estimador de densidade kernel) acima de um limite. A detecção das regiões conectadas é realizada por procedimentos descritos em Azzalini e Torelli (2007), para problemas com dimensão baixa e, em Menardi e Azzalini (2014), para dimensões altas. Em ambos os casos, após a identificação de vários núcleos de grupos, com alta densidade, os dados de densidade mais baixa são alocados seguindo uma abordagem semelhante à classificação supervisionada. Uma característica interessante do PdfCluster é a não necessidade da informação do número de grupos a priori, sendo esse determinado no próprio processo de clusterização.

Kulczycki e Charytanowicz (2010) propuseram um algoritmo de clusterização baseado na suposição de que grupos são relacionados com máximos locais da função do estimador de densidade kernel ($\widehat{f}$). Nesse algoritmo é realizada a transposição sucessiva

dos elementos do conjunto de dados na direção do vetor gradiente de $\widehat{f}$ com um passo fixo apropriado. O algoritmo interrompe sua execução quando a soma dos deslocamentos dos pontos da iteração k para $k+1$ não sofre alteração significativa. Segundo Kulczycki e Charytanowicz (2010), o algoritmo não exige a suposição preliminar do número de grupos, sendo esse parâmetro determinado pela estrutura interna de dados, no entanto, o algoritmo é muito sensível ao KDE e apresenta um alto custo de execução para problemas com número elevado de observações.

Rodriguez e Laio (2014) propuseram o Algoritmo *DPC* (*Density Peak Clustering*), baseado na suposição de que os centros dos grupos são cercados por vizinhos com menor densidade local e que estão a uma distância relativamente grande de qualquer ponto com maior densidade local. No entanto, esse algoritmo pode produzir um grupo ruim, porque a métrica de densidade depende fortemente da dimensão do conjunto de dados e a estratégia de atribuir os pontos restantes a um grupo incorreto pode causar propagação de erros. Para contornar esses problemas, Xie et al. (2016) propuseram o Algoritmo FKNN-DPC (*Fuzzy weighted K-Nearest Neighbors Density Peak Clustering*) que usa duas novas estratégias de atribuição baseadas em vizinhos mais próximos de K e vizinhos mais próximos de K ponderados difusos.

Liu, Xia e Yu (2000) desenvolveram um método de clusterização de dados multidimensionais (campo de estudo do aprendizado não supervisionado) empregando exclusivamente o método de árvore de decisão (ferramenta amplamente utilizada no aprendizado supervisionado) para particionar o espaço de dados em grupos e regiões esparsas com diferentes níveis de detalhes. O método transforma um problema não supervisionado em supervisionado acrescentando N observações fictícias uniformemente distribuídas, possibilitando o uso de árvores de decisão para detectar aglomerados "naturais" em grandes espaços de alta dimensão de forma eficiente.

Vale ressaltar que, todos os algoritmos mencionados aqui estão focados na clusterização de dados multidimensionais armazenados em matrizes do tipo $m \times n$. Entretanto, muitas aplicações modernas, como reconhecimento de imagem e vídeo, mineração de texto, pesquisa na internet, telecomunicações em larga escala e registros de redes sociais, geram grandes quantidades de dados com múltiplos aspectos que podem ser expressos por meio de matrizes multilineares, isto é, tensores. A clusterização de problemas expressos por tensores está além do foco deste trabalho. Para mais detalhes sugerimos os trabalhos de Huang et al. (2008), Sun e Li (2019), Wu, Lin e Zha (2019).

2.1.1 Métricas de validação

Todos os métodos de clusterização buscam determinar grupos de elementos que apresentem características semelhantes. No entanto, surge uma indagação: quando uma clusterização realizada por um algoritmo pode ser considerada satisfatória ou, no mínimo,

aceitável? A fim de responder essa indagação, é necessário avaliar os resultados dos algoritmos de clusterização, o qual é uma tarefa difícil e muitas vezes subjetiva. Por esse motivo, existem técnicas e índices destinados exclusivamente para a validação de um agrupamento. Segundo Nguyen e Rayward-Smith (2008), determinar a qualidade da clusterização envolve avaliar a qualidade dos grupos produzidos por meio de critérios externos, internos ou relativos.

De acordo com Kassambara (2017), critérios de validação externos são usados quando os resultados dos algoritmos podem ser comparados com alguma estrutura pré-especificada, com informações externas, como rótulos de classe e número de grupos, fazendo uma correspondência biunívoca entre cada observação real do conjunto com o resultado predito pelo algoritmo.

Para Race (2014) e Tharwat (2020), o conhecimento prévio dos rótulos de classes propicia o uso da matriz de confusão como critério de representação dos resultados. A matriz de confusão, também chamada matriz de correspondência, é uma tabela de dupla entrada que exibe a correspondência entre os rótulos de grupos determinados pelos algoritmos e os rótulos reais permitindo, de maneira simples, visualizar uma correspondência assertiva dos algoritmos em relação aos dados reais, fornecendo o número de acertos e erros de forma intuitiva.

Tabela 1 – Matriz de confusão

		Classes Preditas				Total
		P_1	P_2	$\cdots$	P_k	
Classes Reais	C_1	a_{11}	a_{12}	$\cdots$	a_{1k}	$\sum_{j=1}^{k} a_{1j}$
	C_2	a_{21}	a_{22}	$\cdots$	a_{2k}	$\sum_{j=1}^{k} a_{2j}$
	:	:	:	$\cdots$	:	:
	C_k	a_{k1}	a_{k2}	$\cdots$	a_{kk}	$\sum_{j=1}^{k} a_{kj}$
Total		$\sum_{i=1}^{k} a_{i1}$	$\sum_{i=1}^{k} a_{i2}$	$\cdots$	$\sum_{i=1}^{k} a_{ik}$	m

Fonte: O autor (2021)

Na Tabela 1, apresentamos uma estrutura geral de uma matriz de confusão multiclasses, em que k corresponde ao número de grupos e classes; a_{ii}, com $i = 1, \ldots, k$, corresponde a quantidade de elementos da classe C_i atribuídos corretamente ao grupo P_i; a_{ij}, com $i, j = 1, \ldots, k$ e $i \neq j$, é a quantidade de elementos da classe C_i atribuídos incorretamente ao grupo P_j. Cabe observar que a ssociação do grupo P_j com a classe C_i é realizada de forma a maximizar o traço da matriz de confusão.

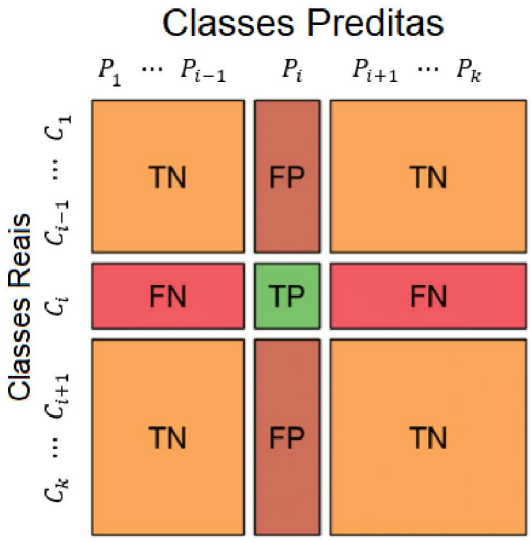
Em resumo, os elementos da diagonal principal da matriz de confusão são atri-

buições corretas, enquanto, os elementos fora da diagonal são atribuições incorretas. Assim o somatório $\sum_{j=1}^{k} a_{ij}$, com $i = 1, \ldots, k$, representa a quantidade de elementos que pertencem a classe real C_i, ou seja, a soma da *i-ésima* linha da matriz de confusão correspondente a quantidade de elementos da classe C_i; $\sum_{i=1}^{k} a_{ij}$, com $j = 1, \ldots, k$, é a quantidade de elementos atribuídos pelo algoritmo ao grupo denominado P_j, ou seja, a soma da *j-ésima* coluna da matriz de confusão correspondente a quantidade de elementos atribuídos ao grupo P_j; por fim, m é a quantidade total de elementos do conjunto de dados que pode ser obtida por

$$m = \sum_{i=1}^{k} \sum_{j=1}^{k} a_{ij}.$$

A disposição dos resultados do agrupamento na matriz de confusão fornece quatro tipos de classificação imediatas entre a associação da classe real com a predita. A Figura 1 apresenta um esquema de análise codificado por cores e siglas, sendo elas:

Figura 1 – Representação dos tipos de classificação da matriz de confusão



Fonte: Adaptado de Krüger (2016).

- TP (*True Positive* - Verdadeiro Positivo): a célula verde corresponde à classificação correta da classe C_i ao grupo P_i, ou seja, a classe foi alocada onde realmente deveria;
- TN (*True Negative* - Verdadeiro Negativo): as células em laranja também correspondem às classificações corretas em relação a Classe C_i, isto é, são todos os elementos das classes $C_{j \neq i}$ alocados nos grupos $P_{j \neq i}$, para $j = 1, \ldots, k$;
- FP (*False Positive* - Falso Positivo): as células na cor marrom estabelecem valores aos quais o algoritmo prevê a classe Positiva quando deveria prever a classe Negativa,

isto é, o grupo P_i possui elementos de classes distintas a C_i. Este erro é denominado "Erro Tipo I";

- FN (*False Negative* - Falso Negativo): nas células em vermelho, o agrupamento prevê valores designados à classe Negativa quando deveriam ser da classe Positiva, isto é, o algoritmo designa elementos da classe C_i a outros grupos diferentes ao P_i. Este erro é denominado "Erro Tipo II";

Com base na matriz de confusão, Tabela 1, e os tipos de classificação que ela fornece, Figura 1, é possível determinar métricas de avaliação do agrupamento: Acurácia, Precisão, Sensibilidade e F1-score.

- **Acurácia**

Segundo Han, Kamber e Pei (2011) e Tharwat (2020), a acurácia de um classificador é a porcentagem de elementos que são classificados corretamente pelo algoritmo, ou seja,

$$\text{acurácia} = \frac{\sum_{i=1}^{k} a_{ii}}{\sum_{i=1}^{k}\sum_{j=1}^{k} a_{ij}} = \frac{\sum_{i=1}^{k} a_{ii}}{m}.$$

Ela indica o desempenho geral do modelo, revelando a proporção do número total de previsões corretas, sendo também chamada de taxa de reconhecimento ou eficiência do modelo. Obviamente, essa é uma medida que produz valores no intervalo $[0, 1]$, que pode ser representada na forma de porcentagem. De forma simples, corroborando com Tharwat (2020), podemos relacionar a acurácia com a taxa de erro ou taxa de classificação incorreta, dada por:

$$\text{Erro} = 1 - \text{acurácia}.$$

A acurácia é uma boa indicação geral de como o modelo performou, porém, de acordo com Tharwat (2020), essa métrica é sensível a dados desequilibrados[4], sendo necessário observar outras métricas de validação associadas aos erros tipo I e II.

- **Precisão**

A precisão é uma métrica que indica, das classificações associada a cada grupo P_j, quantas foram acertadas. Essa métrica, associada ao Erro Tipo I, verifica dentro de

[4] Por dados desequilibrados, Tharwat (2020) entende como sendo os conjuntos onde uma classe supera a soma das outras classes.

cada grupo P_j a razão do número de elementos atribuídos de forma correta, isto é,

$$\text{precisão}_j = \frac{a_{jj}}{\sum_{i=1}^{k} a_{ij}} \qquad (j = 1, \ldots, k).$$

Quanto maior seu valor menor será a variabilidade entre os resultados obtidos no grupo P_j, de modo que é indicada para situações em que a designação de elementos de classes diferentes a um mesmo grupo é mais prejudicial.

- **Sensibilidade**

Outra métrica obtida da matriz de confusão é a sensibilidade, também denominada recall ou revocação, cujo objetivo é medir a frequência em que o algoritmo designa elementos de uma dada classe do problema ao grupo de forma correta, Erro Tipo II, sendo dada pela expressão:

$$\text{sensibilidade}_i = \frac{a_{ii}}{\sum_{j=1}^{k} a_{ij}} \qquad (i = 1, \ldots, k).$$

No sentido oposto a precisão, a sensibilidade é indicada para medir a qualidade do agrupamento nas situações em que a designação de elementos de uma mesma classe a diferentes grupo é mais prejudicial.

- **F1-score**

Por fim, a métrica F1-score, também denominada medida F, trata-se de uma média harmônica entre a precisão e a sensibilidade.

$$\text{F1-score}_i = 2 \times \frac{\text{precisão}_i \times \text{sensibilidade}_i}{\text{precisão}_i + \text{sensibilidade}_i}, \qquad i = 1, \ldots, k.$$

Tal métrica combina precisão e sensibilidade de modo a trazer uma única medida de qualidade, sendo que um baixo valor indica que a precisão ou a sensibilidade apresentam resultados insatisfatórios.

Cabe salientar que um agrupamento avaliado por medida externa é considerado adequado quanto mais próximos de 1 forem os valores das quatro métricas apresentadas.

Os critérios de validação externos são dependentes do conhecimento prévio das classes das observações. No entanto, nem sempre se tem tal informação, sendo necessário utilizar outros meios que sejam independentes de tais informações, como por exemplo, os critérios de validação internos, os quais utilizam apenas o conjunto de dados e o agrupamento imposto a ele para avaliar sua qualidade.

Segundo Kassambara (2017), as métricas internas refletem a compactação de elementos e a separação entre diferentes grupos. Assim, considerando que o objetivo dos algoritmos de clusterização é agrupar elementos semelhantes no mesmo grupo e elementos diferentes em grupos distintos, as métricas de validação interna geralmente são baseadas em dois critérios: a coesão e a separação.

Na coesão, cada elemento do mesmo grupo deve estar o mais próximo possível dos outros elementos do grupo. Por outro lado, na separação, elementos de diferentes grupos devem estar o mais separados possível. Várias abordagens para medir essa proximidade e separação de elementos de grupos são apresentadas na literatura: índice Ball-Hall (Ball e Hall (1965)), índice Banfield-Raftery (Banfield e Raftery (1993)), índice Calinski-Harabasz (Caliński e Harabasz (1974)), índice Dunn (Dunn (1974)), coeficiente de silhueta (Rousseeuw (1987)) e o Índice DB (Davies e Bouldin (1979)), sendo as duas últimas utilizadas nesse trabalho, e por esse motivo, são apresentadas a seguir.

- **Coeficiente de silhueta**

Para determinar o coeficiente de silhueta, precisamos apenas da partição obtida pela aplicação de algum algoritmo de clusterização e a informação de todas as proximidades entre os elementos do conjunto, por exemplo, a matriz de distâncias Euclidianas. O coeficiente de silhueta é calculado para cada objeto i dentre os m elementos do conjunto de dados, da seguinte forma

$$s(i) = \frac{b(i) - a(i)}{max\{b(i), a(i)\}}, \quad i = 1, \ldots, m$$

em que $a(i)$ é a diferença (dissimilaridade) média entre o objeto i e todos os outros pontos do mesmo grupo a qual i pertence e $b(i)$ é a dissimilaridade média mínima do objeto i aos objetos pertencentes a grupos diferentes. O coeficiente de silhueta é a média de todos os valores de saída $s(i)$ e está no intervalo $[-1, 1]$, tal que $s(i)$ próximo de 1 significa que o objeto i está bem agrupado, enquanto $s(i)$ próximo de -1 significa que não está bem agrupado.

Para Rousseeuw (1987), o coeficiente de silhueta é indicado para avaliar grupos aproximadamente esféricos e pode ser usado como métrica de validação de algoritmos ou para melhorar os resultados da análise de grupo, por exemplo, movendo um objeto com $s(i)$ negativo para seu vizinho de forma que o novo valor $\overline{s}(i)$ seja positivo ou no mínimo $\overline{s}(i) > s(i)$.

- **Índice DB**

O índice DB, proposto por Davies e Bouldin (1979), considera a razão entre a soma da dispersão interna dos agrupamentos (coesão, dissimilaridade intra-grupos) e a

distância entre os diferentes grupos (separação, dissimilaridade intergrupos). A dispersão interna de um agrupamento i é calculada como

$$S_i = \left\{ \frac{1}{T_i} \sum_{j=1}^{T_i} \|X_j - A_i\|^q \right\}^{\frac{1}{q}}, \qquad i = 1, \cdots, k$$

em que T_i é o número de elementos pertencentes ao grupo i; A_i é o centroide do grupo i; k é o número de grupos; q é um parâmetro que expressa a tendência ou dispersão dos pontos do grupo i em relação ao centroide desse grupo, de modo que $q = 1$, S_i se torna a distância média dos elementos do grupo ao seu centroide e, se $q = 2$, S_i se torna o desvio padrão das distâncias dos elementos ao centroide do grupo.

Por outro lado, a separação entre grupos é medida por

$$M_{ij} = \left\{ \sum_{l=1}^{k} \|a_{li} - a_{lj}\|^p \right\}^{\frac{1}{p}}$$

sendo a_{li} o $l-$ésimo componente do centroide $a_i \in \mathbb{R}^n$. Se $p = 1$, M_{ij} é a distância de *Manhattan*, caso $p = 2$, M_{ij} é a distância Euclideana entre os centroides i e j.

Com base nas expressões anteriores, determina-se uma matriz de ordem k, das razões que associam cada grupo i com o grupo j, com $i, j = 1, \cdots, k$ e $i \neq j$, dada por

$$R_{ij} = \frac{S_i + S_j}{M_{ij}}.$$

O índice Davies-Bouldin é então definido como

$$\overline{R} = \frac{1}{k} \sum_{i=1}^{k} R_i$$

em que

$$R_i = \max\{R_{ij}, \ \text{com} i \neq j, \ \ j = 1, \ldots, k\}.$$

Segundo Davies e Bouldin (1979), $\overline{R}$ é a média de todo o sistema das medidas de similaridade de cada grupo com o grupo mais semelhante, de forma que a melhor escolha do agrupamento é aquela que minimiza essa similaridade média.

Ambos os critérios, silhueta e índice DB, possuem a característica de não dependerem do número de grupos e do método de clusterização, o que os tornam adequados para avaliação de algoritmos de clusterização, por outro lado, são métricas baseadas em distância, normalmente a Euclidiana, o que tende a favorecer agrupamentos com grupos esféricos.

2.2 ESTIMADOR DE DENSIDADE KERNEL

Seja $x \in \mathbb{R}^n$ uma variável aleatória e f uma função de densidade de probabilidade (f.d.p.) associada a x. O conhecimento de f fornece uma descrição natural do comportamento da variável x e permite que características associadas a ela sejam estudadas e replicadas. No entanto, nem sempre conhecemos a f.d.p. das variáveis aleatórias envolvidas nos problemas provenientes de fenômenos reais. Diante disso, estimadores não paramétricos são utilizados para contornar tal dificuldade.

Para Scott (2015), o foco da estimação não-paramétrica é diferente da paramétrica. Nos estimadores paramétricos a ênfase está em obter o melhor estimador $\widehat{\theta}$ para um dado parâmetro θ, enquanto no caso não-paramétrico o objetivo está diretamente ligado à obtenção de uma boa estimativa $\widehat{f}$ da função de densidade f.

A estimativa de densidade kernel é uma técnica não-paramétrica para suavização de dados com base em amostras finitas, que fornece uma maneira simples de encontrar estruturas em conjuntos de dados sem a imposição de um modelo paramétrico. Segundo Wand e Jones (1994), essa é uma das técnicas de suavização de dados mais importantes e amplamente usadas. Por suavização de dados, Gramacki (2018) versa que esta consiste em encontrar soluções aproximadas de uma função que captura padrões importantes dos dados, considerando cada um dos elementos, desprezando ruídos, removendo variações aleatórias e mostrando componentes de tendência do conjunto, sendo a suavização um conceito fundamental na análise de dados.

De acordo com Silverman (1986), Wand e Jones (1994) e Gramacki (2018), seja $x \in \mathbb{R}^n$ o vetor das variáveis do espaço n-dimensional; $X \in \mathbb{R}^{m \times n}$ a matriz dos dados, sendo m o número de observações, n o número de características/atributos e x_{ij} o valor do j-ésimo atributo da observação i, com $i = 1, \ldots, m$ e $j = 1, \ldots, n$; H a matriz largura de banda ou matriz de suavização, de ordem n, não-aleatória, simétrica e definida positiva; $|H|$ o determinante de H e $K : \mathbb{R}^n \to \mathbb{R}$ a função kernel, satisfazendo a condição

$$\int K(x)dx = 1.$$

Dessa forma, o estimador de densidade kernel multivariado é dado por

$$(x, H) \in \mathbb{R}^n \times \mathbb{R}^{n \times n} \mapsto \widehat{f}(x, H) = m^{-1} \sum_{i=1}^{m} |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}(x - X_i)). \tag{2.1}$$

Cabe aqui observar alguns pontos importantes para o entendimento das discussões que virão a seguir. Em alguns momentos do texto, iremos nos referir às linhas e colunas da matriz de dados X. Dessa forma, usaremos as seguintes notações: $X_i \in \mathbb{R}^n$, $i = 1, \ldots, m$

para representar as linhas e $Y_j \in \mathbb{R}^m$, $j = 1, \ldots, n$ para representar as colunas, ou seja,

$$X_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{pmatrix} \qquad Y_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{mj} \end{pmatrix}.$$

De modo que,

$$X = \begin{pmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_m^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix} = \begin{pmatrix} Y_1 & Y_2 & \cdots & Y_n \end{pmatrix}.$$

Outro ponto importante é que a maioria das integrais que aparecem nesse trabalho são multidimensionais e impróprias, mas por simplicidade adotamos um abuso de notação representado-as com um único símbolo de integral e omitindo os parâmetros, isto é, as integrais do tipo

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n$$

são expressas por $\int f(x)dx$. Por fim, na matriz H adotamos a notação $H > 0$ para expressar que a matriz H é simétrica e definida positiva, conforme Definição 2.1 dada por Ribeiro e Karas (2013).

Definição 2.1. *Seja $A \in \mathbb{R}^{n \times n}$ uma matriz simétrica. A é dita definida positiva quando $x^T Ax > 0$, para todo $x \in \mathbb{R}^n \setminus \{0\}$. Tal propriedade é denotada por $A > 0$. Se $x^T Ax \geq 0$, para todo $x \in \mathbb{R}^n$, A é dita semidefinida positiva, fato este denotado por $A \geq 0$.*

Sobre a função kernel, segundo Silverman (1986), K é normalmente escolhida dentre as f.d.p. que apresentam a característica de serem unimodais e radialmente simétricas, sendo a densidade normal padrão multivariada

$$x \in \mathbb{R}^n \mapsto K(x) = (2\pi)^{-\frac{n}{2}} \exp\left(-\frac{1}{2} x^T x\right) \tag{2.2}$$

comumente usada. Além da f.d.p Gaussiana, Silverman (1986), Wand e Jones (1994), Scott (2015) e Gramacki (2018) relatam que as funções Epanechnikov, Biweight, Triangular, Uniforme e Triweight também são utilizadas como kernel no KDE.

Considerando as relações (2.1) e (2.2), o estimador de densidade kernel Gaussiano tem a forma

$$(x, H) \in \mathbb{R}^n \times \mathbb{R}^{n \times n} \mapsto \widehat{f}(x, H) = m^{-1}(2\pi)^{-\frac{n}{2}} |H|^{-\frac{1}{2}} \sum_{i=1}^{m} \exp\left(-\frac{1}{2} (x - X_i)^T H^{-1} (x - X_i)\right) \tag{2.3}$$

Para Gramacki (2018), o KDE multivariado pode ser visto como uma soma ponderada de "colisões" de densidade que são centradas em cada ponto X_i do conjunto de dados. A relação (2.3) pode ser entendida como uma função nas variáveis x e H, no entanto, uma vez fixada a matriz H temos uma função na variável $x \in \mathbb{R}^n$. A seguir abordamos uma forma de se determinar a matriz H.

2.2.1 Matriz de suavização ou matriz largura de banda

Na função (2.1) e/ou (2.3), escolher a matriz largura de banda ideal é um problema extremamente delicado, uma vez que pequenas variações na matriz H afetam significativamente a forma e a orientação do KDE.

Wand e Jones (1994) e Gramacki (2018) relatam que a matriz largura de banda é definida em três níveis de complexidade. Seja F o conjunto de matrizes do $\mathbb{R}^{n\times n}$, simétricas e definidas positivas. O caso mais simples de matrizes do conjunto F é quando o quadrado de um escalar $h > 0$ multiplica a matriz identidade de ordem n, isto é, $H \in S \subset F$, em que $S = \{h^2 I_n\}$. Considerando H construída dessa maneira, a expressão (2.1) pode ser reescrita como

$$(x,h) \in \mathbb{R}^n \times \mathbb{R}_+^* \mapsto \widehat{f}(x,h) = \frac{1}{mh^n} \sum_{i=1}^{m} K\left(\frac{x - X_i}{h}\right).$$

O segundo nível de complexidade da matriz largura de banda ocorre quando a matriz H é estabelecida por uma matriz diagonal definida positiva, ou seja, $H \in D \subset F$, sendo $D = \text{diag}(h_1^2, h_2^2, \ldots, h_n^2)$ com $h_i > 0, i = 1, \ldots, n$. Desta forma, reescrevendo (2.1), temos:

$$(x,H) \in \mathbb{R}^n \times \mathbb{R}^{n\times n} \mapsto \widehat{f}(x,H) = \frac{1}{m}\left(\prod_{j=1}^{n} h_j\right)^{-1} \sum_{i=1}^{m} K\left(\frac{x_1 - x_{i1}}{h_1}, \frac{x_2 - x_{i2}}{h_2}, \cdots, \frac{x_n - x_{in}}{h_n}\right).$$

Por fim, no terceiro nível, H é uma matriz completa, simétrica e definida positiva. Do ponto de vista teórico, o uso dessa matriz deve fornecer os melhores estimadores de densidade, mas o custo dessa precisão é também o mais alto e, portanto, sua análise formal é a mais difícil. Por outro lado, o uso de um único parâmetro de suavização h é uma ideia muito atraente para fins práticos, pois apenas um parâmetro deve ser estimado, porém, segundo Silverman (1986), o uso de um único parâmetro de suavização implica em um dimensionamento da função KDE igual em todas as direções, desprezando a variabilidade das componentes dos dados. Sobre esses argumentos pode ser mais apropriado usar um conjunto de parâmetros de suavização, considerando assim a variabilidade das coordenadas. Esse conjunto de parâmetros é expresso pelo uso da matriz diagonal, que corresponde ao segundo nível de complexidade.

Silverman (1986) argumenta ainda que, assim como em muitos outros procedimentos estatísticos multivariados, o KDE necessita de um pré-dimensionamento dos

dados, para evitar diferenças extremas da amplitude das variáveis. Se isso for feito, geralmente não há necessidade de considerar formas mais complicadas para a matriz H, onde um único parâmetro de suavização ou um vetor de parâmetros é o suficiente para expressar significativamente os dados por meio do KDE.

Independentemente do nível de complexidade da matriz H, o objetivo de se determinar a melhor matriz H para um conjunto de dados está na importância de reduzir a discrepância do estimador de densidade $\widehat{f}$ da densidade real f. Autores como Wand e Jones (1994), Silverman (1986) e Gramacki (2018) relatam que, ao considerar o erro do estimador para dados discretos, uma medida natural é o erro quadrático médio (do inglês, *Mean Square Error-MSE*), dado por

$$MSE_x(\widehat{f}(x,H)) = E\left[\left(\widehat{f}(x,H) - f(x)\right)^2\right],$$

em que E é a esperança matemática ou valor esperado de um vetor aleatória. Para maiores esclarecimentos, consultar os trabalhos de Härdle e Simar (2015) e Morettin e Bussab (2017).

A partir desse momento desenvolvemos a expressão do MSE com o intuito de determinar uma relação fechada para os elementos da matriz H. Desenvolvimentos semelhantes são apresentados nos trabalhos de Silverman (1986) e Wand e Jones (1994), contudo de forma mais direta, omitindo etapas do desenvolvimento. A fim de simplificar a escrita, adotamos algumas notações: $\widehat{f}(x,H) = \widehat{f}$ e $f(x) = f$. Com isso temos

$$\begin{aligned}
MSE_x(\widehat{f}) &= E\left[\left(\widehat{f} - f\right)^2\right] \\
&= E\left[\left(\widehat{f} - E(\widehat{f}) + E(\widehat{f}) - f\right)^2\right] \\
&= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2 + 2\left(\widehat{f} - E(\widehat{f})\right)\left(E(\widehat{f}) - f\right) + \left(E(\widehat{f}) - f\right)^2\right] \\
&= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] + 2\left(E\left[\left(\widehat{f} - E(\widehat{f})\right)\left(E(\widehat{f}) - f\right)\right]\right) + E\left[\left(E(\widehat{f}) - f\right)^2\right].
\end{aligned}$$

Nessa expressão, $\left(E(\widehat{f}) - f\right)$ e $E(\widehat{f})$ são constantes e, como a esperança de uma constante é a própria constante, segue que

$$\begin{aligned}
MSE_x(\widehat{f}) &= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] + 2\left(\left(E(\widehat{f}) - f\right)E\left[\left(\widehat{f} - E(\widehat{f})\right)\right]\right) + \left(E(\widehat{f}) - f\right)^2 \\
&= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] + 2\left(\left(E(\widehat{f}) - f\right)\left(E(\widehat{f}) - E\left[E(\widehat{f})\right]\right)\right) + \left(E(\widehat{f}) - f\right)^2 \\
&= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] + 2\left(\left(E(\widehat{f}) - f\right)\left(E(\widehat{f}) - E(\widehat{f})\right)\right) + \left(E(\widehat{f}) - f\right)^2 \\
&= E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] + \left(E(\widehat{f}) - f\right)^2.
\end{aligned}$$

Como

$$\text{var}(\widehat{f}) = E\left[\left(\widehat{f} - E(\widehat{f})\right)^2\right] \quad e \quad \text{viés} = E(\widehat{f}) - f$$

o MSE pode ser reescrito como

$$MSE_x(\widehat{f}) \quad = \quad \text{var}(\widehat{f}) + (\text{viés})^2$$

O MSE é um erro pontual, desse modo, uma maneira amplamente usada para medir o erro sistemático de forma global, ou seja, para todo o domínio, é por meio do erro quadrático médio integrado (do inglês, *Mean Integrated Squared Error-MISE*), dado por

$$MISE[\widehat{f}(x,H)] = \int \text{var}\widehat{f}(x,H)dx + \int (\text{viés})^2\, dx. \tag{2.4}$$

Silverman (1986) e Wand e Jones (1994) enfatizam que a decisão de trabalhar com o MISE para medir o desempenho global do KDE se deve, em grande parte, à sua simplicidade matemática. Segundo Wand e Jones (1994) para se obter uma aproximação assintótica simples ao MISE de um KDE multivariado é preciso supor certas hipóteses sobre as densidades f e K, além da matriz H, que são:

(h1) Cada entrada da matriz $\nabla^2 f(x)$ é contínua e $\int |\nabla^2 f(x)|^2 dx < \infty$, ou seja, é quadrado integrável;

(h2) $H = (H^m)_{m\in\mathbb{N}}$ é uma sequência de matrizes largura de banda tal que $m^{-1}|H|^{-\frac{1}{2}}$ e todas as entradas de H se aproximam de zero quando $m \to \infty$. Além disso, assume-se que a razão entre o maior e menor autovalor de H é limitada para todos m.

(h3) $K : \mathbb{R}^n \to \mathbb{R}$ é um kernel multivariado, limitado e compacto, que satisfaz

$$\int K(z)dz = 1, \tag{2.5}$$

$$\int zK(z)dz = 0, \tag{2.6}$$

$$\int zz^T K(z)dz = \mu_2(K)I, \tag{2.7}$$

onde $\mu_2(K) = \int z_i^2 K(z)dz$ é independente de i,

$$K(z) \geq 0 \text{ para todo } z \in \mathbb{R}^n \tag{2.8}$$

e

$$K(z) = K(-z). \tag{2.9}$$

A hipótese (h1) estabelece que cada componente da matriz Hessiana da f.d.p. elevada ao quadrado precisa ser integrável, ou seja, possui um valor limitado. Já a hipótese (h2) equivale a dizer que H se aproxima de zero na medida que m cresce, mas a uma taxa mais lenta que $m^{-1}|H|^{-\frac{1}{2}}$. Por último, se a função kernel K atende à hipótese (h3), temos que K é uma f.d.p. positiva, par, com vetor de médias $\mu = 0$ e matriz de covariância Σ, sendo essa uma matriz diagonal com seus elementos definidos por $\int z_i^2 K(z)dz$. Cabe observar que as expressões (2.6) e (2.7) são os primeiro e segundo momentos, respectivamente, da f.d.p. K. Por fim, na expressão (2.6) o integrando é um vetor do $\mathbb{R}^n$, sendo cada uma das suas componentes uma função $f_i : \mathbb{R}^n \to \mathbb{R}$, enquanto na expressão (2.7) o integrando é uma matriz do $\mathbb{R}^{n\times n}$ cujos elementos são funções $f_{ij} : \mathbb{R}^n \to \mathbb{R}$.

De acordo com Wand e Jones (1994), considerando a hipótese (h2) e a expansão em série de potências do viés quadrado e da variância, a soma dos termos principais do MISE é o chamado erro quadrático integrado médio assintótico (do inglês, *Asymptotic Mean Integrate Square Error-AMISE*), ou seja,

$$MISE = AMISE + erro.$$

De acordo com a hipótese (h2), temos então que, $AMISE \approx MISE$ a medida que $m \to \infty$.

Voltando ao problema de se determinar o $MISE$, para uma melhor compreensão da relação (2.4), fizemos seu desenvolvimento em partes, usando os pressupostos das hipóteses listadas e unindo os resultados, quando conveniente, para obtermos a expressão do AMISE.

Iniciamos o desenvolvimento pelo viés, para tal, definimos o conceito de convolução de duas funções, que segundo Fisher e Taş (2005) e Chacón e Duong (2018) é dada por:

Definição 2.2. *Sejam $f, g : \mathbb{R}^n \to \mathbb{R}$ funções integráveis, então a convolução $f * g$ é definida por*

$$(f * g)(x) = \int (f(y)g(x-y))dy = \int f(x-y)g(y)dy$$

para todo x para o qual as integrais existam.

Para Chacón e Duong (2018), a noção de convolução de funções pode ser estendida à convolução de duas distribuições de probabilidade, onde o KDE pode ser considerado a convolução da medida de probabilidade induzida pelo kernel K_H escalado com a distribuição empírica dos dados, sendo o valor esperado do KDE dado por

$$E[\widehat{f}(x, H)] = (K_H * f)(x).$$

Com base na definição (2.2) e na expressão anterior, segue que

$$E[\widehat{f}(x, H)] = \int K_H(x-y)f(y)dy \tag{2.10}$$

em que

$$K_H(x-y) = |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}(x-y)).$$

Segundo a hipótese (h3-2.9), temos

$$\begin{aligned} K_H(z) &= K_H(-z) \\ K_H(x-y) &= K_H(y-x) \\ &= |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}(y-x)). \end{aligned}$$

Fazendo uma mudança de variável, ou seja, $z = H^{-\frac{1}{2}}(y-x)$, temos $y = x + H^{\frac{1}{2}}z$, com $dy = |H|^{\frac{1}{2}}dz$. Substituindo essas informações em (2.10), segue que

$$\begin{aligned} E[\widehat{f}(x,H)] &= |H|^{-\frac{1}{2}} \int K(z) f(x + H^{\frac{1}{2}}z)(|H|^{\frac{1}{2}})dz \\ &= \int K(z) f(x + H^{\frac{1}{2}}z)dz. \end{aligned}$$

Aplicando Taylor de 2^a ordem em $f(x + H^{\frac{1}{2}}z)$, em torno de x, obtemos

$$f(x+H^{\frac{1}{2}}z) = f(x) + \left(H^{\frac{1}{2}}z\right)^T \nabla f(x) + \frac{1}{2}\left(H^{\frac{1}{2}}z\right)^T \nabla^2 f(x) \left(H^{\frac{1}{2}}z\right) + o\left(\left(H^{\frac{1}{2}}z\right)^T \left(H^{\frac{1}{2}}z\right)\right)$$

que substituído na valor esperado, resulta em

$$\begin{aligned} E[\widehat{f}(x,H)] &= \int K(z) \left[f(x) + \left(H^{\frac{1}{2}}z\right)^T \nabla f(x) + \frac{1}{2}\left(H^{\frac{1}{2}}z\right)^T \nabla^2 f(x) \left(H^{\frac{1}{2}}z\right) + \right. \\ &+ \left. o\left\{\left(H^{\frac{1}{2}}z\right)^T \left(H^{\frac{1}{2}}z\right)\right\}\right] dz \\ E[\widehat{f}(x,H)] &\approx \int K(z) f(x) dz + \int K(z) \left(H^{\frac{1}{2}}z\right)^T \nabla f(x) dz + \\ &+ \int \frac{1}{2}\left(H^{\frac{1}{2}}z\right)^T \nabla^2 f(x) \left(H^{\frac{1}{2}}z\right) dz. \end{aligned} \tag{2.11}$$

Como $K(z)$ é uma f.d.p., pela hipótese (h3-2.5), a primeira integral da expressão (2.11) é dada por

$$\begin{aligned} \int K(z) f(x) dz &= f(x) \int K(z) dz \\ &= f(x) \cdot 1 \\ &= f(x). \end{aligned} \tag{2.12}$$

Pela hipótese (h3-2.6), a segunda integral de (2.11) resulta em

$$\begin{aligned}
\int K(z)\left(H^{\frac{1}{2}}z\right)^T \nabla f(x)dz &= \int K(z)z^T H^{\frac{1}{2}}\nabla f(x)dz \\
&= \int (zK(z))^T H^{\frac{1}{2}}\nabla f(x)dz \\
&= \left[\int (zK(z))^T dz\right] H^{\frac{1}{2}}\nabla f(x) \\
&= \left[\int zK(z)dz\right]^T H^{\frac{1}{2}}\nabla f(x) \\
&= 0 \cdot H^{\frac{1}{2}}\nabla f(x), \\
&= 0. \qquad (2.13)
\end{aligned}$$

Substituindo (2.12) e (2.13) em (2.11), obtemos

$$\begin{aligned}
E[\widehat{f}(x,H)] &\approx f(x) + \frac{1}{2}\int \left(H^{\frac{1}{2}}z\right)^T \nabla^2 f(x)\left(H^{\frac{1}{2}}z\right)dz \\
&\approx f(x) + \frac{1}{2}\int z^T H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zK(z)dz. \qquad (2.14)
\end{aligned}$$

O integrando da relação (2.14) pode ser substituído por uma expressão equivalente que utiliza o conceito de traço[5] de uma matriz quadrada (tr(A)).

Proposição 2.1. *Considere os vetores $u, v \in \mathbb{R}^n$ e a matriz $A \in \mathbb{R}^{n\times n}$, tal que $A = uv^T$. Então*

$$tr(A) = u^T v = v^T u.$$

Usando a proposição (2.1) com $u = H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}z$ e $v = zK(z)$, segue que

$$\begin{aligned}
z^T H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zK(z) &= u^T v \\
&= \mathrm{tr}(uv^T) \\
&= \mathrm{tr}(H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zz^T K(z)).
\end{aligned}$$

Integrando essa última igualdade, temos

$$\begin{aligned}
\int z^T H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zK(z)dz &= \int \mathrm{tr}\left(H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zz^T K(z)\right)dz \\
&= \mathrm{tr}\left(\int H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}zz^T K(z)dz\right) \\
&= \mathrm{tr}\left(H^{\frac{1}{2}}\nabla^2 f(x)H^{\frac{1}{2}}\int zz^T K(z)dz\right).
\end{aligned}$$

[5] O traço de uma matriz quadrada de ordem n é a soma de todos os elementos da diagonal principal.

Substituindo esse último resultado em (2.14), obtemos

$$\begin{aligned} E[\widehat{f}(x,H)] &\approx f(x) + \frac{1}{2}\int \left(H^{\frac{1}{2}}z\right)^T \nabla^2 f(x) \left(H^{\frac{1}{2}}z\right) dz \\ &\approx f(x) + \frac{1}{2}\int z^T H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}} z K(z) dz \\ &\approx f(x) + \frac{1}{2}\text{tr}\left(H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}} \int zz^T K(z) dz\right). \end{aligned}$$

Aplicando a hipótese (h3-2.7), segue que

$$E[\widehat{f}(x,H)] \approx f(x) + \frac{1}{2}\text{tr}\left(H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}} \mu_2(K) I\right).$$

Antes de continuar, precisamos enunciar duas propriedades de traço de matriz, que serão utilizadas no desenvolvimento do AMISE, sendo elas:

$$\text{tr}(AB) = \text{tr}(BA)$$

$$\text{tr}(\alpha A) = \alpha \text{tr}(A).$$

Voltando ao AMISE, com base nas propriedade de traço, é possível reescrever o valor esperado da KDE como

$$\begin{aligned} E[\widehat{f}(x,H)] &\approx f(x) + \frac{1}{2}\text{tr}\left(H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}} \mu_2(K) I\right) \\ &\approx f(x) + \frac{1}{2}\mu_2(K)\ \text{tr}\left(H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}} I\right) \\ &\approx f(x) + \frac{1}{2}\mu_2(K)\ \text{tr}\left(H^{\frac{1}{2}} \nabla^2 f(x) H^{\frac{1}{2}}\right) \\ &\approx f(x) + \frac{1}{2}\mu_2(K)\ \text{tr}\left(H^{\frac{1}{2}} H^{\frac{1}{2}} \nabla^2 f(x)\right) \\ &\approx f(x) + \frac{1}{2}\mu_2(K)\ \text{tr}\left(H \nabla^2 f(x)\right). \end{aligned}$$

Com isso,

$$\begin{aligned} E[\widehat{f}(x,H)] - f(x) &\approx \frac{1}{2}\mu_2(K)\text{tr}\left(H\nabla^2 f(x)\right) \\ \left(E[\widehat{f}(x,H)] - f(x)\right)^2 &\approx \left(\frac{1}{2}\mu_2(K)\text{tr}\left(H\nabla^2 f(x)\right)\right)^2 \\ \int \left(E[\widehat{f}(x,H)] - f(x)\right)^2 dx &\approx \int \left(\frac{1}{4}\mu_2(K)^2 tr^2\left\{H\nabla^2 f(x)\right\}\right) dx \\ &\approx \frac{1}{4}\mu_2(K)^2 \int tr^2\left(H\nabla^2 f(x)\right) dx. \end{aligned} \tag{2.15}$$

Essa expressão fornece a primeira parte de (2.4), sendo necessário determinar a segunda parte, dada por $\int \text{var}[\widehat{f}(x,H)]dx$. De acordo com Wand e Jones (1994) e Chacón,

Duong e Wand (2011), a variância pode ser definida como

$$\begin{aligned}\text{var}\left[\widehat{f}(x,H)\right] &= m^{-1}\left[(K_H^2 * f)(x) - (K_H * f)(x)^2\right] \\ &= m^{-1}\left[\int K_H^2(x-y)f(y)dy - \left(\int K_H(x-y)f(y)dy\right)^2\right].\end{aligned}$$

Da mesma forma que no cálculo do viés, seja $K_H(x-y) = K_H(y-x) = |H|^{-\frac{1}{2}}K(H^{-\frac{1}{2}}(y-x))$. Com a mudança de variável $z = H^{-\frac{1}{2}}(y-x)$ temos que $y = x + H^{\frac{1}{2}}z$ e $dy = |H|^{\frac{1}{2}}dz$, com isso

$$\begin{aligned}\text{var}\left[\widehat{f}(x,H)\right] &= m^{-1}\left[\int |H|^{-1}K^2(z)f(x+H^{\frac{1}{2}}z)|H|^{\frac{1}{2}}dz\right. \\ &- \left.\left(\int |H|^{-\frac{1}{2}}K(z)f(x+H^{\frac{1}{2}}z)|H|^{\frac{1}{2}}dz\right)^2\right] \\ &= m^{-1}\left[|H|^{-\frac{1}{2}}\int K^2(z)f(x+H^{\frac{1}{2}}z)dz - \left(\int K(z)f(x+H^{\frac{1}{2}}z)dz\right)^2\right].\end{aligned}$$

Fazendo uma aproximação por meio de Taylor de 1^a ordem em torno de x, ou seja,

$$\begin{aligned}f\left(x+H^{\frac{1}{2}}z\right) &= f(x) + \left(H^{\frac{1}{2}}z\right)^T\nabla f(x) + o\left(\|H^{\frac{1}{2}}\|\right) \\ f\left(x+H^{\frac{1}{2}}z\right) &\approx f(x) + \left(H^{\frac{1}{2}}z\right)^T\nabla f(x),\end{aligned}$$

obtemos

$$\begin{aligned}\text{var}\left[\widehat{f}(x,H)\right] &\approx m^{-1}\left\{|H|^{-\frac{1}{2}}\int K^2(z)\left[f(x) + \left(H^{\frac{1}{2}}z\right)^T\nabla f(x)\right]dz\right. \\ &- \left.\left[\int K(z)\left[f(x) + \left(H^{\frac{1}{2}}z\right)^T\nabla f(x)\right]dz\right]^2\right\} \\ \text{var}\left[\widehat{f}(x,H)\right] &\approx m^{-1}\left\{|H|^{-\frac{1}{2}}\int K^2(z)f(x)dz + |H|^{-\frac{1}{2}}\int K^2(z)\left(H^{\frac{1}{2}}z\right)^T\nabla f(x)dz\right. \\ &- \left.\left[\int K(z)f(x)dz + \int K(z)\left(H^{\frac{1}{2}}z\right)^T\nabla f(x)dz\right]^2\right\}. \qquad (2.16)\end{aligned}$$

Na expressão (2.16) há quatro integrais a serem calculadas, as quais analisamos individualmente a seguir. Na primeira integral, conforme Wand e Jones (1994), denotamos $\int K^2(z)dz = R(K)$, logo

$$\begin{aligned}\int K^2(z)f(x)dz &= f(x)\int K^2(z)dz \\ &= f(x)R(K).\end{aligned}$$

Já na terceira integral,

$$\begin{aligned}\int K(z)f(x)dz &= f(x)\int K(z)dz \\ &= f(x)\cdot 1 \\ &= f(x).\end{aligned}$$

A quarta é exatamente a mesma desenvolvida em (2.13), ou seja,

$$\int K(z)\left(H^{\frac{1}{2}}z\right)^T \nabla f(x)dz = 0.$$

Por último, a segunda integral requer um pouco mais de atenção, onde

$$\begin{aligned}\int K^2(z)\left(H^{\frac{1}{2}}z\right)^T \nabla f(x)dz &= \int K^2(z)z^T H^{\frac{1}{2}}\nabla f(x)dz \\ &= \int \left(zK^2(z)\right)^T H^{\frac{1}{2}}\nabla f(x)dz \\ &= \left(\int zK^2(z)dz\right)^T H^{\frac{1}{2}}\nabla f(x).\end{aligned}$$

Este último integrando é o produto de uma função simétrica por uma função ímpar, logo seu resultado é zero, isto é,

$$\begin{aligned}\int K^2(z)\left(H^{\frac{1}{2}}z\right)^T \nabla f(x)dz &= (0)^T H^{\frac{1}{2}}\nabla f(x) \\ \int K^2(z)\left(H^{\frac{1}{2}}z\right)^T \nabla f(x)dz &= 0.\end{aligned}$$

Substituindo os resultados das quatro integrais obtidas anteriormente na expressão (2.16), obtemos

$$\begin{aligned}\text{var}\left[\widehat{f}(x,H)\right] &\approx m^{-1}\left\{|H|^{-\frac{1}{2}}f(x)R(K) - |H|^{-\frac{1}{2}}\cdot 0 - f^2(x) - 0\right\} \\ &\approx m^{-1}|H|^{-\frac{1}{2}}R(K)f(x) - m^{-1}f^2(x) \\ \int \text{var}\left[\widehat{f}(x,H)\right]dx &\approx \int m^{-1}|H|^{-\frac{1}{2}}R(K)f(x)dx - \int m^{-1}f^2(x)dx \\ &\approx m^{-1}|H|^{-\frac{1}{2}}R(K) - m^{-1}\int f^2(x)dx.\end{aligned}$$

Pelas suposições de integrabilidade nas hipóteses (h1) a (h3), é possível combinar essa última relação com a expressão (2.15) para obter o AMISE do KDE multivariado,

$$AMISE[\widehat{f}(x,H)] \approx \frac{1}{4}\mu_2^2(K)\int tr^2\left(H\nabla^2 f(x)\right)dx + m^{-1}|H|^{-\frac{1}{2}}R(K) - m^{-1}\int f^2(x)dx. \tag{2.17}$$

A integral do primeiro membro da expressão (2.17) pode ser substituída por uma expressão equivalente, para tal, precisamos definir alguns operadores e propriedades de matrizes. Para maiores detalhes consultar Magnus e Neudecker (2019).

Definição 2.3. *Seja A uma matriz quadrada de ordem n. O vetor de A, denotado por $vec(A)$, é o vetor de ordem $n^2 \times 1$ obtido pelo empilhamento das colunas de A na ordem da esquerda para a direita. Isto é, seja*

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}.$$

Então

$$vec(A) = \begin{bmatrix} a_{11} \\ \vdots \\ a_{n1} \\ a_{12} \\ \vdots \\ a_{n2} \\ \vdots \\ a_{1n} \\ \vdots \\ a_{nn} \end{bmatrix}.$$

Definição 2.4. *Seja A uma matriz quadrada de ordem n, a metade do vetor de A denotada por $vech(A)$ é o vetor $\frac{1}{2}n(n+1) \times 1$ obtido da $vec(A)$, eliminando cada uma das entradas acima da diagonal principal de A.*

Definição 2.5. *Seja A uma matriz quadrada, simétrica e de ordem n. Existe uma matriz D_n, de ordem $n^2 \times \frac{1}{2}n(n+1)$, denominada "matriz de duplicação", constituída de zeros e uns, escolhida de forma apropriada tal que*

$$vec(A) = D_n vech(A).$$

Propriedade 2.1. *Seja A uma matriz quadrada de ordem n, então valem as seguintes propriedades:*

I)

$$D_n^T vec(A) = vech(A + A^T - diag(A))$$

onde a matriz $diag(A)$ é uma matriz formada pela diagonal da matriz A.

II)

$$tr(A^T B) = (vecA)^T (vecB).$$

Voltando na relação (2.17), para facilitar a apresentação dos resultados, adotamos a notação $\nabla^2 f$ para $\nabla^2 f(x)$. Considerando a primeira integral, segue que

$$\int tr^2 \left(H \nabla^2 f\right) dx \;\; = \;\; \int \mathrm{tr}\left(H \nabla^2 f\right) \mathrm{tr}\left(H \nabla^2 f\right) dx.$$

Como H é uma matriz simétrica, definida positiva, logo $H = H^T$. Então pela Propriedade (2.1), temos que

$$\mathrm{tr}\left(H\nabla^2 f\right) = (\mathrm{vec}H)^T \left(\mathrm{vec}\nabla^2 f\right) = \left(\mathrm{vec}\nabla^2 f\right)^T (\mathrm{vec}H)\,.$$

Logo,

$$\int tr^2\left(H\nabla^2 f\right) dx \;=\; \int (\mathrm{vec}H)^T \left(\mathrm{vec}\nabla^2 f\right)\left(\mathrm{vec}\nabla^2 f\right)^T (\mathrm{vec}H)\, dx.$$

Pela propriedade (2.1) e pela simetria de H, $\mathrm{vec}H = D_n\ \mathrm{vech}H$, então

$$\begin{aligned}\int tr^2\left(H\nabla^2 f\right) dx &= \int (D_n\ \mathrm{vech}H)^T \left(\mathrm{vec}\nabla^2 f\right)\left(\mathrm{vec}\nabla^2 f\right)^T (\mathrm{vec}H)\, dx \\ &= \int \left(\mathrm{vech}^T H\right) D_n^T \left(\mathrm{vec}\nabla^2 f\right)\left(\mathrm{vec}\nabla^2 f\right)^T (\mathrm{vec}H)\, dx\end{aligned}$$

mas,

$$D_n^T \mathrm{vec}\nabla^2 f = \mathrm{vech}\left(\nabla^2 f + \left(\nabla^2 f\right)^T - \mathrm{diag}\nabla^2 f(x)\right)$$

e, como a Hessiana da função f é uma matriz simétrica, segue que $\nabla^2 f = \left(\nabla^2 f\right)^T$, logo

$$D_n^T \mathrm{vec}\nabla^2 f = \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right).$$

Portanto,

$$\int \mathrm{tr}^2\left(H\nabla^2 f\right) dx \;=\; \int (\mathrm{vech}H)^T\, \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(\mathrm{vec}\nabla^2 f\right)^T (\mathrm{vec}H)\, dx.$$

Por outro lado, vec $H = D_n$ vech H, então

$$\begin{aligned}&\int \mathrm{tr}^2\left(H\nabla^2 f\right) dx = \\ =\ &\int (\mathrm{vech}H)^T\, \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(\mathrm{vec}\nabla^2 f\right)^T (D_n\ \mathrm{vech}\ H)\, dx \\ =\ &\int (\mathrm{vech}H)^T\, \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(D_n^T\ \mathrm{vec}\nabla^2 f\right)^T (\mathrm{vech}\ H)\, dx \\ =\ &\int (\mathrm{vech}H)^T\, \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(\mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\right)^T (\mathrm{vech}\ H)\, dx \\ =\ &(\mathrm{vech}H)^T \left[\int \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(\mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\right)^T dx\right](\mathrm{vech}\ H)\end{aligned}$$

denotando

$$\Psi_F = \int \mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\left(\mathrm{vech}\left(2\nabla^2 f - \mathrm{diag}\nabla^2 f\right)\right)^T dx,$$

temos

$$\int tr^2\left(H\nabla^2 f\right) dx \;=\; (\mathrm{vech}H)^T\, \Psi_F\, (\mathrm{vech}\ H)\,.$$

Substituindo este resultado em (2.17), obtemos finalmente a expressão do AMISE, dada por

$$\begin{aligned} AMISE[\widehat{f}(x,H)] &= \frac{1}{4}\mu_2^2(K)\,(\text{vech}H)^T\,\Psi_F\,(\text{vech }H) + m^{-1}|H|^{-\frac{1}{2}}R(K) \\ &\quad -m^{-1}\int f^2(x)dx. \end{aligned} \tag{2.18}$$

Como mencionado, a escolha adequada da matriz H reduz a discrepância entre o estimador $\widehat{f}$ e a densidade real f. Logo, almejamos que o AMISE seja o menor possível, o que nos leva ao seguinte problema:

$$\begin{aligned} &\min \quad AMISE\left[\widehat{f}(x,H)\right] \\ &\text{s.a.} \quad H > 0. \end{aligned}$$

Para Wand e Jones (1994), Chacón, Duong e Wand (2011) e Gramacki (2018), não é possível determinar uma fórmula explícita e fechada para H que minimize a expressão (2.18). No entanto, supondo que f seja uma f.d.p. normal multivariada do espaço $\mathbb{R}^n$, com vetor de médias μ e matriz covariância Σ, ou seja, $f \sim N(\mu,\Sigma)$ e, considerando $\widehat{f}$ como o KDE multivariado Gaussiano, o AMISE pode ser rescrito como

$$AMISE\widehat{f}(x,H) = m^{-1}(4\pi)^{-\frac{n}{2}}|H|^{-\frac{1}{2}} + \frac{1}{16}(4\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\left(2\text{tr}\left(H\Sigma^{-1}H\Sigma^{-1}\right) + tr^2\left(H\Sigma\right)\right) + c.$$

Minimizando em função de H, obtemos

$$H^*_{AMISE} = \left(\frac{4}{n+2}\right)^{\frac{2}{n+4}}\Sigma m^{-\frac{2}{n+4}}.$$

Tomando como hipótese que as n variáveis que expressam o conjunto de dados não sejam correlacionadas, temos

$$H^*_{AMISE} = \left(\frac{4}{n+2}\right)^{\frac{2}{n+4}}\begin{bmatrix} \sigma^2_{Y_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2_{Y_n} \end{bmatrix} m^{-\frac{2}{n+4}}.$$

Logo,

$$H^*_{AMISE} = diag\left(\left(\frac{4}{n+2}\right)^{\frac{2}{n+4}}\sigma^2_{Y_j}m^{-\frac{2}{n+4}}\right).$$

Portanto, cada elemento da diagonal de H^*_{AMISE} pode ser expresso por

$$\begin{aligned} h_j^2 &= \left(\frac{4}{n+2}\right)^{\frac{2}{n+4}}\sigma^2_{Y_j}m^{-\frac{2}{n+4}} \\ h_j^2 &= \left[\left(\frac{4}{n+2}\right)^{\frac{1}{n+4}}\sigma_{Y_j}m^{-\frac{1}{n+4}}\right]^2 \\ h_j &= \left(\frac{4}{n+2}\right)^{\frac{1}{n+4}}\sigma_{Y_j}m^{-\frac{1}{n+4}} \end{aligned}$$

sendo n a dimensão do espaço, m o número de observações e σ_{Y_j}, com $j = 1, \ldots, n$, o desvio padrão dos elementos da j-ésima coluna da matriz X.

Azzalini e Torelli (2007) e Menardi e Azzalini (2014) sugerem que a matriz de suavidade deva ser multiplicada pela constante $\frac{3}{4}$, denotado por fator de encolhimento, a fim de aliviar o excesso de suavização determinado pelo cálculo das larguras de banda sob a premissa de normalidade multivariada. Neste trabalho, para minimizar os efeitos inferidos pelas hipóteses na determinação da matriz H introduzimos uma variável $\alpha \in \mathbb{R}_+^*$, de forma semelhante a proposta de Matioli et al. (2017) para o caso univariado, ou seja,

$$\begin{aligned} H &= \alpha \cdot H_{AMISE}^* \\ H &= \alpha \cdot \operatorname{diag}\left(\left(\frac{4}{n+2}\right)^{\frac{2}{n+4}} \sigma_{Y_j}^2 m^{-\frac{2}{n+4}}\right). \end{aligned} \tag{2.19}$$

A relação definida em (2.19) é empregada nos algoritmos MulticlusterKDE, AdditiveclusterKDE e TreeKDE com o intuito de determinar a matriz largura de banda. Dessa forma, o KDE Gaussiano, em ambos os algoritmos, é definido como (2.3) e com matriz largura de banda definida conforme (2.19).

2.3 ÁRVORE DE DECISÃO

Os métodos baseados em árvores são algoritmos de aprendizagem supervisionada que habilitam modelos preditivos com alta precisão, estabilidade e facilidade de interpretação. Segundo Liu, Xia e Yu (2000), esses métodos, do ponto de vista geométrico, particionam o espaço recursivamente em um conjunto de regiões hiper-retangulares por uma série de testes (ou cortes) e, em seguida, ajustam um modelo simples de acordo com a variável resposta. Se a variável resposta é categórica, temos uma árvore de decisão ou classificação; se é contínua, temos uma árvore de regressão.

De acordo com Maimon e Rokach (2014), uma árvore de decisão consiste em nós que formam uma árvore enraizada em um nó inicial, denominado "nó raiz", o qual não possui arestas de entrada e contempla todas as observações do problema. Todos os outros nós têm exatamente uma aresta de entrada e aqueles que possuem arestas de saída são referidos como "nós internos" ou "nós de teste". Por outro lado, nós que não possuem arestas de saída são chamados de "nós folhas" ou "nós terminais" ou ainda "nós de decisão". Em uma árvore de decisão, cada nó interno divide o espaço em dois ou mais subespaços de acordo com o método utilizado, sendo a separação binária uma das mais comuns e a utilizada neste trabalho.

Sejam $X \in \mathbb{R}^{m \times n}$ o espaço de um determinado evento e $y \in \mathbb{R}^m$ o vetor de classificação das observações desse evento. A construção da árvore de decisão começa com seu nó raiz, composto por todas as m observações. A partir desse nó, o método determina,

mediante uma regra, a melhor variável, indicada pelo índice γ, e o melhor valor c_γ para fazer uma divisão do nó em outros nós, sendo o nó analisado denominado "nó pai" e os nós provenientes da divisão denominados "nós filhos". Considerando uma partição binária, os nós filhos são delimitados pelas regiões do espaço, denominadas R_1 e R_2, e descritas por Hastie, Tibshirani e Friedman (2009) como

$$R_1(\gamma, c_\gamma) = \{X(i,:) \mid X(i,\gamma) \leq c_\gamma, \text{com } i = 1, \ldots, m\}$$

e

$$R_2(\gamma, c_\gamma) = \{X(i,:) \mid X(i,\gamma) > c_\gamma, \text{com } i = 1, \ldots, m\}$$

em que $X(i,:)$ a i-ésima observação do conjunto de dados (i-ésima linha da matriz X) e $X(i,\gamma)$ é o elemento da i-ésima linha e da coluna γ da matriz X.

Segundo Maimon e Rokach (2014), existem vários critérios (regras) que podem ser usados para selecionar a componente γ e o valor de corte c_γ, todos tendo a variável y como pilar central. Esses critérios podem ser caracterizados por dependência, distância ou impurezas. Dentre as mais comuns temos a entropia, o índice Gini, a razão de verossimilhança e a razão de ganho.

A árvore de decisão é um método recursivo em que cada um dos nós filhos, obtidos da partição do conjunto de recurso, na iteração subsequente é considerado como nó pai e, então, o processo de busca e divisão é repetido até que um critério de parada seja atendido. Vários critérios podem ser utilizados como, por exemplo, número mínimo de observações por nó, quantidade máxima de nós ou quantidade máxima de partições (iterações).

2.4 MÉTODOS DE OTIMIZAÇÃO

Nesta seção, abordamos conceitos básicos de otimização de funções reais que servirão de suporte para os algoritmos de clusterização propostos nesse trabalho. Para um maior aprofundamento sobre otimização, consultar os trabalhos de Nocedal e Wright (1999), Izmailov e Solodov (2012), Izmailov e Solodov (2014) e Ribeiro e Karas (2013).

Para Ribeiro e Karas (2013), um problema de otimização pode ser expresso como

$$\begin{aligned} &\text{minimizar} \quad f(x) \\ &\text{sujeito a} \quad x \in \Omega \end{aligned} \tag{2.20}$$

sendo $f : \mathbb{R}^n \to \mathbb{R}$ e $\Omega \subset \mathbb{R}^n$.

No caso em que $\Omega = \mathbb{R}^n$, temos um problema de otimização irrestrito. Caso $\Omega = \{x \in \mathbb{R}^n \mid c_\epsilon(x) = 0, \quad c_I(x) \leq 0\}$, sendo $c_\epsilon : \mathbb{R}^n \to \mathbb{R}^m$ e $c_I : \mathbb{R}^n \to \mathbb{R}^p$, temos um problema de otimização restrito, com m restrições de igualdade e p restrições de desigualdade.

Segundo Nocedal e Wright (1999), problemas no formato (2.20) podem ser classificados como linear, não-linear, convexo ou não-convexo, de acordo com a natureza da função objetivo e restrições. Além disso, podem ser considerados problemas de grande ou pequeno porte dependendo do número de variáveis e, ainda, podem ser problemas diferenciáveis ou não, mediante a suavidade das funções a qual o problema é expresso.

Técnicas de otimização buscam determinar pontos de mínimos locais da função objetivo que atendam ao conjunto de restrições. Por ponto de mínimo, ou minimizador da função f, Ribeiro e Karas (2013) definem:

Definição 2.6. *Considere uma função $f : \mathbb{R}^n \to \mathbb{R}$ e $x^* \in \Omega \subset \mathbb{R}^n$. O ponto x^* é um minimizador local de f em Ω quando existe um $\delta > 0$, tal que $f(x^*) \leq f(x)$, para todo $x \in B(x^*, \delta) \cap \Omega$. Caso $f(x^*) \leq f(x)$, para todo $x \in \Omega$, x^* é dito minimizador global de f em Ω.*

Nos limitamos neste trabalho a apresentar conceitos referentes a problemas de otimização irrestritos, ou seja, $\Omega = \mathbb{R}^n$. Desse modo, todo ponto $x^* \in \mathbb{R}^n$ é um ponto viável para o problema (2.20), no entanto, ser viável não garante que o ponto seja uma solução. Para que x^* seja solução do problema, há condições necessárias e suficientes que o caracterizam como ponto de mínimo da função. A seguir apresentamos essas condições e para maiores detalhes consultar Izmailov e Solodov (2014) e Ribeiro e Karas (2013).

Teorema 2.1 (Condição necessária de 1^a ordem). *Seja $f : \mathbb{R}^n \to \mathbb{R}$ diferenciável no ponto $x^* \in \mathbb{R}^n$. Se o ponto x^* é um minimizador local de f, então*

$$\nabla f(x^*) = 0.$$

O Teorema 2.1 estabelece que se x^* é um minimizador da função f, então $\nabla f(x^*) = 0$, porém, sua recíproca não é verdadeira, isto é, o fato de $\nabla f(x^*) = 0$ não garante que x^* seja um minimizador de f. De qualquer modo, tal resultado fornece mecanismos para a resolução de problemas de otimização e motiva a definição de ponto crítico.

Definição 2.7. *Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função diferenciável. Se $\nabla f(x^*) = 0$, o ponto $x^* \in \mathbb{R}^n$ é dito ponto crítico ou ponto estacionário de f.*

Em geral, algoritmos de otimização são desenvolvidos para encontrar pontos críticos e o Teorema 2.1 fornece um bom critério de parada, isto é, $\|\nabla f(x^*)\| \leq \epsilon$, sendo $\epsilon > 0$ uma precisão pré-estabelecida.

Teorema 2.2 (Condição necessária de 2^a ordem). *Seja $f : \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável no ponto $x^* \in \mathbb{R}^n$. Se x^* é um minimizador local de f, então a matriz Hessiana de f no ponto x^* é semidefinida positiva, isto é,*

$$d^T \nabla^2 f(x^*) d \geq 0$$

para todo $d \in \mathbb{R}^n$.

Assim como no Teorema 2.1, a recíproca do Teorema 2.2 também não é verdadeira, isto é, dado um ponto crítico x^*, o fato de $\nabla^2 f(x^*)$ ser uma matriz semidefinida não garante que x^* seja um minimizador de f.

Teorema 2.3 (Condição suficiente de 2^a ordem). *Seja* $f : \mathbb{R}^n \to \mathbb{R}$ *duas vezes diferenciável no ponto* $x^* \in \mathbb{R}^n$. *Se* x^* *é um ponto estacionário da função* f *e* $\nabla^2 f(x^*)$ *é definida positiva, então* x^* *é um minimizador local estrito de* f.

De acordo com Nocedal e Wright (1999) e Ribeiro e Karas (2013), minimizadores globais são difíceis de reconhecer e ainda mais difíceis de determinar. Por essa razão, métodos de otimização irrestritos buscam por mínimos locais, mas muitas vezes se contentam com pontos estacionários. Esses métodos exigem, a priori, um ponto de partida, um ponto inicial, geralmente denotado por $x^0 \in \mathbb{R}^n$. A partir desse ponto os métodos geram uma sequência de iterações $\{x^k\}_{k=0}^{\infty}$, caracterizando-os como processos iterativos. Esse processo é encerrado quando não é possível fazer mais progresso ou tenha atingido uma solução satisfatória.

Segundo Ribeiro e Karas (2013), uma forma de construir um método de otimização implica escolher, a partir de um determinado ponto, uma direção para dar o próximo passo, sendo uma direção a qual a função f decresce uma escolha razoável. Essa direção é definida como direção de descida.

Definição 2.8. *Considere uma função* $f : \mathbb{R}^n \to \mathbb{R}$, *um ponto* $\overline{x} \in \mathbb{R}^n$ *e uma direção* $d \in \mathbb{R}^n \setminus \{0\}$. d *é uma direção de descida para* f, *a partir de* $\overline{x}$, *quando existe* $\delta > 0$ *tal que* $f(\overline{x} + td) < f(\overline{x})$, *para todo* $t \in (0, \delta)$.

Uma condição para que uma dada direção d seja de descida é dada pelo Teorema 2.4 a seguir, o qual pode ser encontrado, juntamente com sua demonstração nos trabalhos de Izmailov e Solodov (2012) e Ribeiro e Karas (2013).

Teorema 2.4. *Se* $\nabla f(\overline{x})^T d < 0$, *então* d *é uma direção de descida para* f *a partir de* $\overline{x}$.

De acordo com Ribeiro e Karas (2013) uma classe de direções de descida pode ser definida por uma transformação do gradiente de f via matrizes definidas positivas. Seja $H : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ uma função contínua que associa a cada $x \in \mathbb{R}^n$ uma matriz definida positiva $H(x) \in \mathbb{R}^{n \times n}$. Assim, se $\nabla f(x) \neq 0$, uma direção de descida pode ser definida como

$$d = -H(x)\nabla f(x).$$

É fácil verificar que d, definida conforme a expressão anterior, é uma direção de descida, isto é, $H(x)$ é por definição uma matriz definida positiva, logo,

$$\nabla f(x)^T H(x) \nabla f(x) > 0 \ \ \forall x \in \mathbb{R}^n,$$

com isso temos

$$\begin{aligned} \nabla f(x)^T H(x) \nabla f(x) &> 0 \\ -\nabla f(x)^T H(x) \nabla f(x) &< 0 \\ \nabla f(x)^T \left(-H(x) \nabla f(x)\right) &< 0 \\ \nabla f(x)^T d &< 0, \end{aligned}$$

o que mostra que d é uma direção de descida.

Para Nocedal e Wright (1999), a estratégia usada para definir a matriz H distingue um método de otimização de outro, sendo que a maioria das estratégias utilizam informações de primeira ou segunda ordem da função objetivo. De acordo com Izmailov e Solodov (2012), se $H(x^k) = I \in \mathbb{R}^{n \times n}$ para todo $k \in \mathbb{N}$, temos o Método do Gradiente, o qual é globalmente convergente, no entanto, com convergência linear na melhor das hipóteses. Caso $H(x^k) = \left(\nabla^2 f(x^k)\right)^{-1}$, temos o Método de Newton, o qual é localmente convergente, porém com convergência quadrática, desde que sejam atendidas algumas hipóteses. Para maiores detalhes consultar Izmailov e Solodov (2012) e Ribeiro e Karas (2013).

Entre os Métodos do Gradiente e Newton, há uma classe de métodos, conhecida como Métodos quase-Newton, que buscam obter uma matriz H que seja uma boa aproximação da Hessiana da função f, avaliada no ponto $x^k \in \mathbb{R}^n$, cuja construção emprega apenas informações de primeira ordem. Nesse contexto, os Métodos quase-Newton constituem uma classe de métodos com melhor desempenho com relação a convergência, quando comparado ao Método do Gradiente, e com relação ao custo computacional, quando comparado ao Método de Newton.

Segundo Izmailov e Solodov (2012), historicamente o Método DFP, proposto por Davidon-Fletcher-Powell, é o primeiro Método quase-Newton. Este determina a cada iteração uma matriz $H \in \mathbb{R}^{n \times n}$, simétrica e definida positiva, por meio da expressão

$$H_{k+1}^{DFP} = H_k + \frac{p^k (p^k)^T}{(p^k)^T q^k} - \frac{H_k q^k (q^k)^T H_k}{(q^k)^T H_k q^k},$$

sendo p^k e $q^k \in \mathbb{R}^n$, tal que $(p^k)^T q^k > 0$, $p^k = x^{k+1} - x^k$ e $q^k = \nabla f(x^{k+1}) - \nabla f(x^k)$. Com isso, temos

$$H_{k+1}^{DFP}(x^k) \approx \left(\nabla^2 f(x^k)\right)^{-1}.$$

Para Nocedal e Wright (1999) e Ribeiro e Karas (2013), o Método BFGS proposto por Broyden-Fletcher-Goldfarb-Shanno é o método quase-Newton mais popular. O

Método BFGS tem uma relação semelhante ao Método DFP, porém esse busca encontrar uma aproximação para a matriz Hessiana, ao invés da sua inversa, ou seja, o método busca encontrar

$$B_{k+1} \approx \nabla^2 f(x^{k+1}),$$

em que

$$H_{k+1} = (B_{k+1})^{-1}.$$

Segundo Ribeiro e Karas (2013), a relação que determina a matriz $H \in \mathbb{R}^{n \times n}$, que atualiza o passo do Método BFGS, é dada por

$$H_{k+1}^{BFGS} = H_k + \left(1 + \frac{(q^k)^T H_k q^k}{(p^k)^T q^k}\right) \frac{p^k (p^k)^T}{(p^k)^T q^k} - \frac{p^k (q^k)^T H_k + H_k q^k (p^K)^T}{(p^k)^T q^k}.$$

A partir do Método BFGS Nocedal (1980) propôs o Método quase-Newton com memória limitada, denominado Método L-BFGS, o qual pode ser visto como uma variação do Método BFGS com a imposição de uma restrição na armazenagem de informações de iterações anteriores.

Segundo Nocedal e Wright (1999), os métodos quase-Newton de memória limitada são uma classe de algoritmos indicados para resolver problemas de grande porte cujas matrizes Hessianas não podem ser calculadas a um custo razoável ou são densas demais para serem manipuladas facilmente. Essa classe de métodos recebe o nome de memória limitada devido ao armazenamento de aproximações de baixo custo da matriz Hessiana, ou de sua inversa, ao invés de sua forma completa. De acordo com Nocedal (1980), métodos quase-Newton aproximam a Hessiana por matrizes simétricas que necessitam $\frac{n(n+1)}{2}$ posições de armazenamento para cada aproximação, onde n é a dimensão do espaço. Considerando problemas de grande porte, tal armazenamento torna-se uma prática custosa e até inviável. Então a proposta dos métodos de memória limitada ao invés de armazenar aproximações de ordem $n \times n$, salvam apenas alguns vetores de comprimento n que representam implicitamente as aproximações da matriz Hessiana, sendo atualizados a cada iteração, descartando as informações mais antigas e substituindo-as pelas informações mais recentes.

De acordo com Nocedal e Wright (1999), apesar de suprimir informações acerca da matriz Hessiana, os métodos quase-Newton com memória limitada geram uma taxa aceitável de convergência, embora essa seja linear. Para maiores detalhes, consultar os trabalhos de Nocedal (1980) e Nocedal e Wright (1999).

Partindo do Método L-BFGS, Byrd et al. (1995) propuseram uma extensão desse método para problemas de otimização não lineares de alta dimensão com restrições de caixa do tipo $l \leq x \leq u$, denominado Método L-BFGS-B, sigla em inglês para *Limited memory BFGS with Bound constraints*. Esse método necessita apenas de informações

da primeira derivada da função objetivo e da atualização quase-Newton com memória limitada para aproximar a matriz Hessiana.

Nesse trabalho, propomos novos algoritmos de clusterização que utilizam em suas metodologias a minimização da função KDE e, para tal, utilizamos o método quase-Newton L-BFGS-B.

3 CONTRIBUIÇÕES DO TRABALHO

Neste capítulo apresentamos as contribuições desse trabalho ao campo da clusterização de dados multidimensionais. Tais contribuições referem-se a três algoritmos de clusterização intitulados MulticlusterKDE, AdditiveclusterKDE e TreeKDE e, também, uma métrica de validação interna, denominada índice CD (do inglês, *Clustering Density index*).

Os algoritmos propostos possuem a característica de serem centrados na múltipla otimização da função do estimador de densidade kernel Gaussiano e não exigirem, a priori, o número de grupos, sendo esse para os algoritmos MulticlusterKDE e AdditiveclusterKDE um parâmetro opcional. Além disso, são simples, bem definidos e param em um número finito de passos, de modo a sempre convergirem, independentemente do conjunto de dados a ser agrupado.

A métrica de validação consiste na máxima razão entre a dissimilaridade intragrupos (coesão) e a intergrupos (separação) do agrupamento, refletindo a compactação de elementos e a separação entre diferentes grupos sobre o pior caso.

Para um melhor entendimento das contribuições desse trabalho, inicialmente apresentamos os conceitos relacionados ao índice CD e, na sequência, os algoritmos MulticlusteKDE, AdditiveclusterKDE e TreeKDE. Cabe observar que todos os experimentos numéricos foram realizados com o uso do *Software R*, sendo que o principal motivo pela sua escolha foi o fato de ser um *software* livre, com código aberto e uma linguagem acessível, totalmente flexível, o que permite desenvolver facilmente funções e pacotes, além de possuir uma boa capacidade gráfica.

3.1 ÍNDICE DE DENSIDADE DA CLUSTERIZAÇÃO PARA VALIDAÇÃO INTERNA DE AGRUPAMENTOS

Na Seção 2.1.1, apresentamos duas métricas internas para validação de clusterização disponíveis na literatura, o coeficiente de silhueta e o índice DB. Nessa seção, descrevemos uma das contribuições dessa pesquisa para o campo de estudo sobre clusterização, intitulada índice de Densidade da Clusterização ou, resumidamente índice CD.

No aspecto geral, o índice CD é definido como a máxima razão entre a dispersão interna de cada grupo e a separação do respectivo centroide do grupo ao centroide mais próximo. Supondo que um agrupamento tenha k grupos, o índice CD calcula k razões, e dentre essas, a maior é definida como sendo a métrica de avaliação do agrupamento, ou seja, valor do índice CD.

De forma mais detalhada, supondo que um conjunto de dados, $X \in \mathbb{R}^{m \times n}$, tenha um agrupamento com k grupos provenientes de algum algoritmo de clusterização. A coesão do j-ésimo grupo é calculada pela média das distâncias de todas as observações pertencentes a esse grupo a seu respectivo centroide, ou seja,

$$S_j = \left\{ \frac{1}{T_j} \sum_{i=1}^{M_j} \|\overline{X}_i - \mu_j\| \right\}, \quad j = 1, \ldots, k$$

em que T_j é a quantidade de elementos pertencentes ao grupo j, $\mu_j \in \mathbb{R}^n$ é o centroide do grupo j e $\overline{X} \subset X$ são todos os elementos que pertencem ao grupo j.

Além disso, a separação entre os grupos é medida da seguinte forma

$$D_j = \{min\{\|\mu_j - \mu_i\|; i = 1, \ldots, k; i \neq j\}\}, \quad j = 1, \ldots, k$$

em que μ_j é o centroide do grupo analisado e μ_i é qualquer outro centroide diferente de μ_j. Na separação entre grupos, cada grupos é analisado de forma isolada, ou seja, fixado o grupo j, calculamos a distância do centroide μ_j a todos os outros $k-1$ centroides. A mínima distância encontrada é considerada como medida de separação do grupo j.

Com base nos valores S_j e D_j definimos a razão sobre cada um dos grupos da seguinte maneira

$$CD_j = \frac{S_j}{D_j}, \quad j = 1, \ldots, k.$$

O índice CD corresponde ao valor máximo obtido, ou seja,

$$\text{índice CD} = max\{CD_j; j = 1, \ldots, k\}.$$

Esse resultado exprime quantitativamente a qualidade do agrupamento de um conjunto de dados X em k grupos, onde quanto mais próximo de zero for o valor do índice, melhor é a qualidade do agrupamento.

A justificativa para que o índice CD ideal seja um valor próximo de zero é simples e intuitiva. Como o índice CD consiste em uma razão, logo devemos avaliar o comportamento do seu numerador e do seu denominador. O numerador mede a coesão do grupo, dessa forma, quanto mais compacto o grupo j, menor o valor de S_j. Por outro lado, o denominador avalia a separação entre grupos e, é evidente que, para um agrupamento ser considerando bom, almejamos que os centroides dos grupos estejam o mais distante possível um do outro, logo D_j deverá ser o maior possível. Nestes termos um agrupamento é considerado adequado quando S_j for o menor possível e D_j o maior possível, o que conduz $\frac{S_j}{D_j}$ a um número pequeno. Note que, quando o grupo consistir de apenas um elemento, este será o próprio centroide e S_j será nulo. Por outro lado, em virtude da estrutura dos problemas de clusterização, D_j poderá assumir um valor alto, no entanto nunca assumirá realmente um valor infinito.

Em contrapartida, se o agrupamento for formado por grupos dispersos e os centroides forem próximos, a ordem de grandeza do numerador e do denominador da razão se invertem, provocando que $\frac{S_j}{D_j}$ seja um número grande caracterizando um agrupamento insatisfatório. No entanto, para que um agrupamento seja considerado inadequado, não há a necessidade de que o índice CD seja um número muito grande, basta que seja superior a 1.

Supondo que o valor da métrica seja superior a 1 e considerando a natureza da razão $\frac{S_j}{D_j}$, então $S_j > D_j$, isto é, a distância média dos elementos do grupo ao seu centroide é superior a distância deste ao centroide vizinho mais próximo, ou seja, o centroide mais próximo apresenta uma distância menor que alguns elementos do próprio grupo, o que caracteriza o agrupamento como insatisfatório. Dessa forma, o valor do índice CD superior a 1 caracteriza um agrupamento inconsistente, com centroides deslocados ou com sobreposição de centroides num mesmo conjunto de observações tornando o agrupamento indesejável.

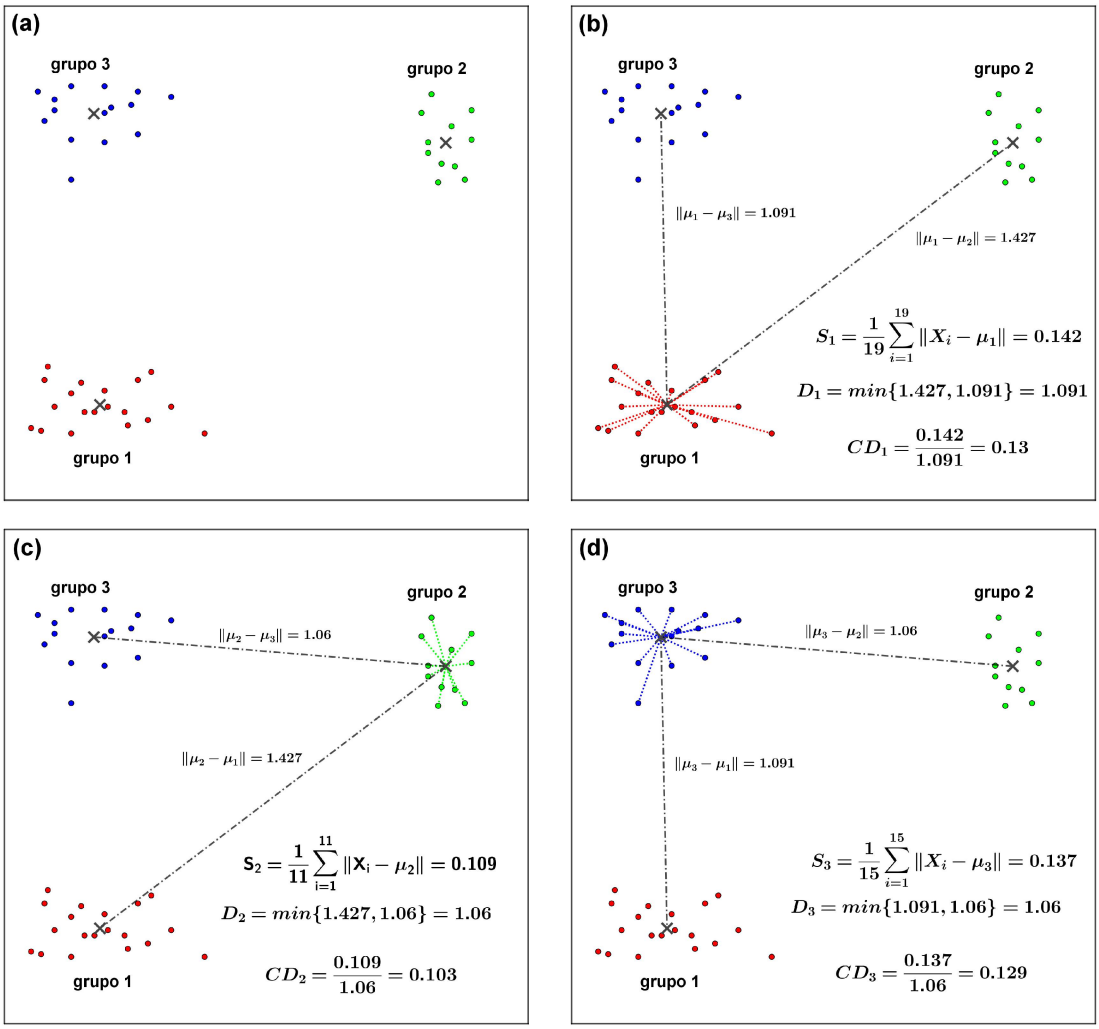
De acordo com a estrutura do índice CD, k razões são calculadas e a máxima é definida como métrica final de validação do agrupamento. A escolha por associar o valor do índice CD com a máxima razão se deve ao fato que, se o grupo que propiciou a pior razão é um grupo considerado bom, compacto e distante de outro centroide, então todos os demais $k-1$ grupos terão características semelhantes ou melhores, justificando a escolha pela máxima razão. Porém, a recíproca dessa afirmação não é verdadeira, visto que um valor do índice CD alto não implica que todos os grupos sejam considerados ruins e sim que existe ao menos um mensurado como insatisfatório.

Para ilustrar o funcionamento do índice CD, escolhemos um exemplo simples com 45 observações de 2 atributos e com uma clusterização constituída por 3 grupos. A clusterização juntamente com os centroides são representados no quadro (a) da Figura 2.

Primeiramente calculamos uma razão para cada um dos grupos e selecionamos como métrica de validação o seu valor máximo, ou seja, $CD = max\{CD_j\}$, com $CD_j = \frac{S_j}{D_j}$, $j = 1, \ldots, 3$.

O quadro (b) da Figura 2 ilustra a determinação de CD_1. Para tanto, inicialmente determinamos as distâncias Euclidianas, representadas por segmentos pontilhados na cor vermelha, de cada elemento do grupo 1 ao seu respectivo centroide. Em seguida, calculamos o valor médio dessas distâncias, $S_1 = 0,142$, que representa a coesão do grupo 1. Já a medida de separação do grupo 1 em relação aos demais grupos é dada pelo valor mínimo entre a distância Euclidiana do centroide 1 aos centroides 2 e 3, representadas no quadro (b) da Figura 2 pelos segmentos na cor preta. Conforme é possível observar, a distância entre o centroide 1 aos centroides 2 e 3 é de 1,427 e 1,091, respectivamente. Logo $D_1 = min\{1,427; 1,091\} = 1,091$ é a medida de separação do grupo 1 aos demais

Figura 2 – Ilustração do índice CD



Fonte: O autor (2021).

grupos do problema. Dessa forma, temos que $CD_1 = \frac{0{,}142}{1{,}091} = 0,13$.

De forma análoga, calculamos os valores $CD_2 = 0,103$ e $CD_3 = 0,129$, conforme apresentado nos quadros (c) e (d) da Figura 2. Com base nas três razões, temos $CD = max\{0,13; 0,103; 0,129\} = 0,13$, o qual quantifica a qualidade da clusterização do conjunto de dados do exemplo em 3 grupos.

Na seção 4.1 conduzimos experimentos numéricos que demonstram o desempenho do índice CD, comparado a outras métricas de validação internas amplamente utilizadas na literatura.

3.2 ALGORITMO MULTICLUSTERKDE

O Algoritmo MulticlusterKDE é um algoritmo composto de duas etapas. A primeira objetiva determinar o número de grupos e seus respectivos centroides, enquanto

a segunda etapa consiste na designação das observações aos grupos, com a possibilidade de redução da quantidade de grupos em um número pré determinado ou uma busca pela melhor configuração. A seguir, apresentamos uma descrição geral do Algoritmo MulticlusterKDE, um exemplo para ilustrar o seu funcionamento e, para finalizar a seção, apresentamos uma discussão a respeito da sua convergência.

3.2.1 Algoritmo MulticlusterKDE: descrição geral

Como mencionado, o Algoritmo MulticlusterKDE é dividido em duas etapas: a primeira é composta por um laço e uma tomada de decisão, sendo a mesma responsável por determinar múltiplos centroides; a segunda etapa é composta por uma tomada de decisão e por laços, sendo essa responsável pela designação dos elementos aos centroides e pela escolha da melhor configuração para clusterização.

Para o Algoritmo MulticlusterKDE, a ser apresentado a seguir, são necessários: uma matriz $X \in \mathbb{R}^{m \times n}$ com os dados do problema, sendo m o número de observações e n o número de atributos; um parâmetro $\alpha \in \mathbb{R}_+^*$, responsável pela suavidade da função KDE, calculado conforme (2.19) para determinar a matriz H, e $nc \in \mathbb{Z}_+^*$ um número inteiro fornecido de forma opcional pelo usuário que tem a função de limitar o número de grupos, isto é, a quantidade máxima de subdivisões, de modo que, sua ausência não inviabiliza a execução do algoritmo.

Na primeira etapa, o laço (linhas 3 a 14) tem como principal tarefa determinar a quantidade de centroides ou, equivalentemente, controlar o número de iterações que o algoritmo realizará. Note que o laço começa com a determinação do minimizador, x^*, da função KDE $(-\widehat{f})$, tendo x^0 como ponto inicial, o qual foi selecionado de forma aleatória antes do início do laço.

Após a determinação do minimizador, o algoritmo tem sua primeira tomada de decisão, sendo essa estabelecida como o único critério de parada do algoritmo. Nessa tomada de decisão o algoritmo verifica se o ponto x^* já pertence ou não ao conjunto dos centroides. Em caso positivo, o laço é encerrado, caso contrário, o valor x^* é armazenado no conjunto dos centroides, o respectivo valor de $\widehat{f}(x^*)$ é armazenado no vetor das imagens e o vetor de distâncias entre o ponto x^* e todas as observações do conjunto X é calculado, com posterior incorporação desse vetor, de dimensão m, na matriz de distância entre centroides e o conjunto X.

Ainda dentro do laço da primeira etapa, o algoritmo executa o processo de seleção do novo ponto inicial, o qual será utilizado na próxima repetição. Este ponto é escolhido em X possuindo a máxima distância aos elementos do conjunto dos centroides. Para tal, o algoritmo analisa a mínima distância em cada uma das linhas da matriz de distância, determinando um vetor de mínimas distâncias. Sobre esse vetor, uma nova busca é reali-

Algoritmo 1: MulticlusterKDE

Entrada: $X \in \mathbb{R}^{m\times n}$, $\alpha \in \mathbb{R}_+^*$, $nc \in \mathbb{Z}_+$;
Escolha $k = 0$, $x^k \in X$ e **faça** $S = \emptyset$, $F = \emptyset$, $C = ones(1:m)$, $rep = 0$, $D = \emptyset$, $s = \infty$ (para índice CD ou DB) ou $s = -\infty$ (para silhueta);
Determine: H e $\widehat{f}$ de acordo com (2.19) e (2.3);
Enquanto $rep < 1$ **faça**
    A partir de x^k determine: $x^* \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \left\{-\widehat{f}(x)\right\}$;
    Se $x^* \in S$ **então**
        $rep = rep + 1$;
    Senão
        $S = S \cup x^*$, $F = F \cup \widehat{f}(x^*)$;
        $D = D \cup \{\text{dist}(x^*, X(i,:)), i = 1, \ldots, m\}$;
        $m_k = \max\left\{\min_{j=1,\cdots,k}\{D(:,j)\}\right\}$;
        $x^{k+1} = x \in X \mid \text{dist}(x, S) = m_k$;
        $k = k + 1$;
    Fim
Fim
Ordene F em ordem decrescente;
Ordene S e D de acordo com F;
Se $nc \neq \emptyset$ **então**
    faça $nc = min\{nc, k\}$;
    Para $i = 1 : m$ **faça**
        Para $j = 1 : nc$ **faça**
            Se $dist(X(i,:), S(j,:)) = min\{D(i, 1:nc)\}$ **então**
                $C[i] = j$
            Fim
        Fim
    Fim
    $s =$ métrica de validação de X de acordo com C
Senão
    Para $l = 2 : k$ **faça**
        $B = ones(1:m)$;
        Para $i = 1 : m$ **faça**
            Para $j = 1 : l$ **faça**
                Se $dist(X(i,:), S(j,:)) = min\{D(i, 1:l)\}$ **então**
                    $B[i] = j$
                Fim
            Fim
        Fim
        $r =$ métrica de validação de X de acordo com B;
        Se r *é melhor que* s **então**
            $s = r$; $C = B$; $S = S(1:l,:)$;
        Fim
    Fim
Fim
Retorna C,s,S

zada com intuito de obter o elemento correspondente à máxima distância para ser o novo ponto inicial para otimização da função KDE, finalizando assim a iteração.

O Algoritmo MulticlusterKDE repete o processo descrito, até que haja uma repetição de um centroide, obtendo ao final dessa etapa k minimizadores distintos da função KDE $(-\widehat{f})$, candidatos a centroides da clusterização, os quais serão utilizados na próxima etapa.

Com o término da primeira etapa, o algoritmo ordena os centroides obtidos em ordem decrescente de acordo com os valores funcionais do vetor de imagens (linha 15). Da mesma forma, ordena as colunas da matriz de distâncias (linha 16).

A segunda etapa do Algoritmo MulticlusterKDE é constituída por uma tomada de decisão principal e por outra secundária, além de dois laços excludentes. No início dessa etapa ocorre a primeira tomada de decisão (linha 17), em que o algoritmo verifica se o parâmetro nc foi ou não informado. Em caso afirmativo ($nc \neq \emptyset$), o algoritmo seleciona os nc centroides que possuem os maiores valores de imagem e as nc primeiras colunas da matriz de distâncias D, em seguida, realiza a designação de cada cada elemento do conjunto X ao centroide mais próximo (linhas 19 a 25), retornando com a clusterização do conjunto X em exatos nc grupos.

Caso o usuário não tenha informado o número de grupos ($nc = \emptyset$), o Algoritmo MulticlusterKDE busca, dentro do conjunto dos centroides, a melhor clusterização segundo uma métrica de validação. Para tal, uma busca com os centroides obtidos na primeira etapa é realizada (linhas 28 a 42), iniciando com 2 centroides e, em seguida, um novo é adicionado à análise, até que os k centroides sejam inseridos. A cada nova clusterização o algoritmo verifica se a configuração com i ($i = 2, \ldots, k$) grupos é a melhor. Em caso afirmativo, o algoritmo armazena as informações da clusterização com i grupos, caso contrário, descarta tal configuração. Ao final, a clusterização que apresentar a melhor avaliação, segundo a métrica, é apresentada como solução pelo Algoritmo MulticlusterKDE.

Vale salientar que os processos correspondentes as linhas 18 a 26 e 28 a 42 são excludentes, ou seja, apenas um deles é realizado, dependendo da informação ou não do parâmetro opcional nc. Independente do caminho tomado pelo Algoritmo MulticlusterKDE este retorna com a clusterização C, o valor da métrica de validação s e o conjunto dos centroides S.

3.2.2 Exemplo do Algoritmo MulticlusterKDE

Para ilustrar o funcionamento do Algoritmo MulticlusterKDE, foi escolhido um exemplo simples com 45 observações e 2 atributos, gerados randomicamente. Os dados são apresentados na Tabela 2 e representados pelo gráfico de dispersão do lado esquerdo

da Figura 3. Além dos dados, para executar o Algoritmo MulticlusterKDE utilizamos o parâmetro $\alpha = 0,75$, que aplicado na relação (2.19) resultou na seguinte matriz de suavidade

$$H = \begin{bmatrix} 0,0464 & 0 \\ 0 & 0,0595 \end{bmatrix}.$$

Tabela 2 – Banco de dados utilizado no exemplo

Observações	Atributos		Observações	Atributos		Observações	Atributos	
	x_1	x_2		x_1	x_2		x_1	x_2
1	4,97	1,98	16	5,97	2,95	31	4,90	1,90
2	4,94	1,98	17	6,01	2,91	32	4,85	2,00
3	5,07	1,93	18	6,04	3,05	33	4,82	2,10
4	4,78	1,92	19	5,97	2,99	34	5,10	2,10
5	5,06	1,98	20	6,10	3,00	35	4,83	2,15
6	5,01	2,00	21	4,85	3,11	36	5,20	2,00
7	4,81	1,91	22	4,90	3,00	37	5,30	1,90
8	4,93	2,09	23	5,02	3,12	38	5,95	3,10
9	4,90	2,05	24	5,00	2,99	39	6,05	2,90
10	5,13	1,95	25	4,82	3,07	40	6,08	2,00
11	5,00	2,06	26	4,80	3,18	41	5,00	3,15
12	5,14	2,13	27	5,10	3,02	42	5,00	3,20
13	5,98	3,17	28	5,08	3,13	43	5,10	3,18
14	6,10	3,11	29	4,90	3,20	44	5,20	3,16
15	6,00	2,84	30	4,85	3,15	45	4,90	2,85

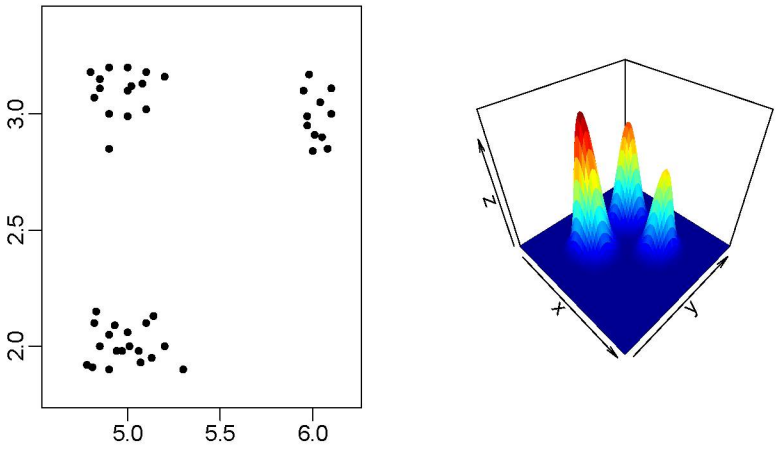
Fonte: O autor (2021).

O gráfico do lado direito da Figura 3 ilustra o comportamento da função KDE com kernel Gaussiano para o conjunto de dados. Tanto a matriz H quanto a função $\widehat{f}$ são construídas apenas uma única vez pelo Algoritmo MulticlusterKDE, independentemente do número de iterações que o mesmo irá realizar.

Uma vez construída a matriz H e definida a função KDE, escolhemos um ponto inicial arbitrário $x_1 = (4,97; 1,98)$. Este ponto é utilizado para determinar o primeiro maximizador da função KDE. Na Figura 4, o ponto inicial é representado pelo símbolo "+" na cor vermelha. A partir deste ponto, iniciamos o processo iterativo até que o critério de parada seja atendido.

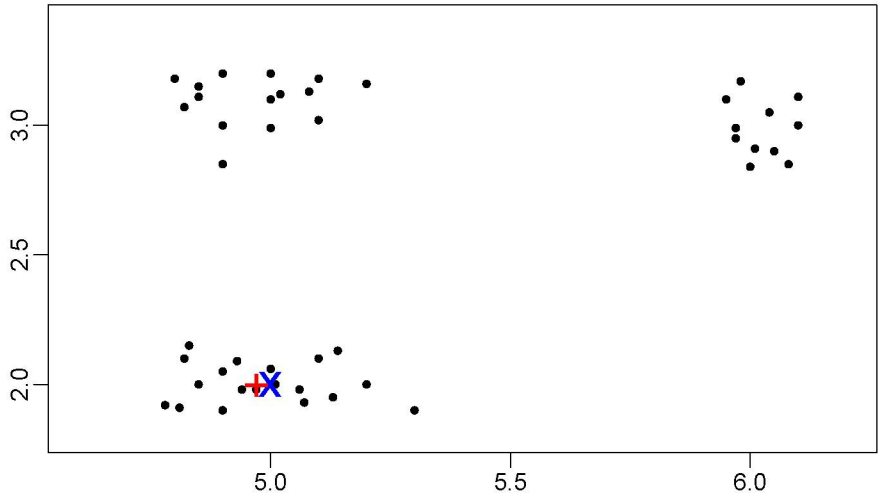
Para a otimização da função KDE, utilizamos a rotina *optim*, do *Software R* (R Core Team (2019)), com especificação do Método L-BFGS-B. Vale salientar que esta rotina realiza a minimização de funções, logo executamos $min(-\widehat{f})$, que é equivalente a $max(\widehat{f})$. O Método L-BFGS-B exige, a priori, uma restrição de caixa, de modo que optamos por utilizar em todas as iterações os limitantes das variáveis, formando assim uma região compacta.

Figura 3 – Representação dos dados e da função do estimador de densidade kernel



Fonte: O autor (2021).

Figura 4 – Determinação do primeiro centroide da clusterização



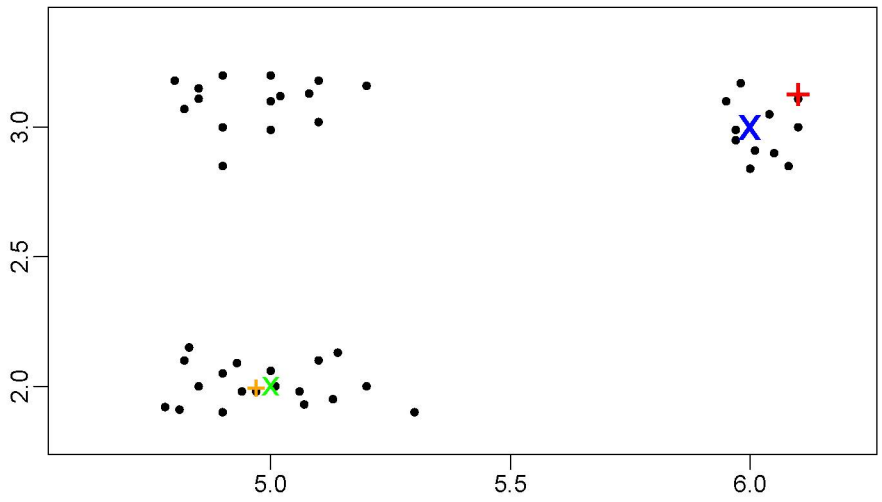
Fonte: O autor (2021).

Dessa forma, partindo do ponto inicial x_1, o algoritmo determinou $x_1^* = (5,00; 2,00)$ como o minimizador de $-\widehat{f}$, o qual é imediatamente definido como um centroide da clusterização. Na Figura 4, este ponto é representado pelo símbolo "$\times$" na cor azul. Vale observar que x_1^* não é necessariamente um elemento do conjunto X. O ponto x_1^* é então designado para o conjunto dos centroides, vazio até o momento, como primeiro centroide e o valor $f(x_1^*) = 19,26$ é atribuído ao vetor das imagens dos centroides. Nesse momento, o algoritmo determina o vetor de distâncias dos elementos de X ao centroide x_1^*.

Na sequência, o Algoritmo MulticlusterKDE determina o ponto do conjunto X que apresenta a máxima distância Euclidiana aos centroides já definidos. O algoritmo

então retorna o ponto $x_2 = (6,10; 3,11)$ (representado pelo símbolo "+" na cor vermelha - Figura 5), cuja distância à x_1^* é de 1,56. Dessa forma, o algoritmo finaliza a primeira iteração da primeira etapa.

Figura 5 – Determinação do segundo centroide da clusterização



Fonte: O autor (2021).

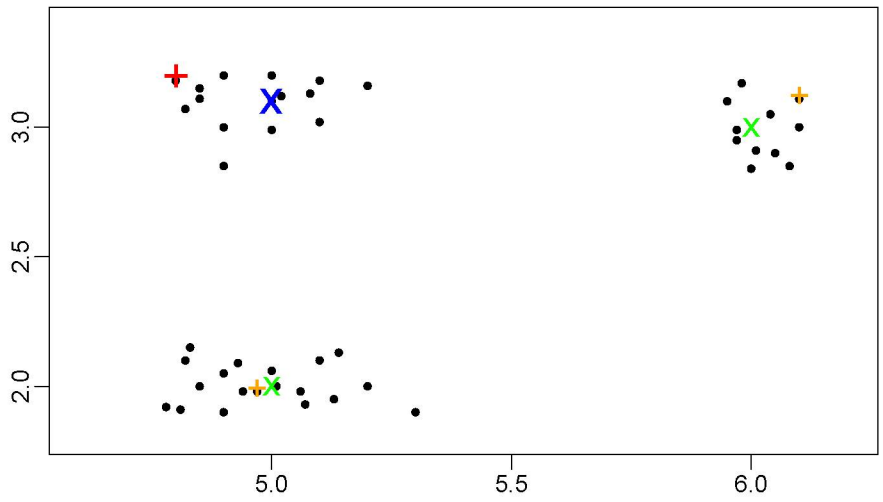
A segunda iteração inicia com $x_2 = (6,10; 3,11)$ e o algoritmo determina $x_2^* = (6,00; 3,00)$ como minimizador de $-\widehat{f}$ (Figura 5). Nesse momento, o Algoritmo MulticlusterKDE faz uma análise desse novo ponto verificando se x_2^* pertence ou não ao conjunto dos centroides. A repetição de um centroide é o único critério de parada para o primeiro laço do algoritmo, visto que, uma vez repetido o ponto de mínimo da função, a busca por um novo ponto inicial conduziria novamente a um ponto inicial já utilizado que, por sua vez, determinaria o mesmo ponto de mínimo, provocando um loop infinito.

Como $x_2^* = (6,00; 3,00)$ não pertence ao conjunto dos centroides, o mesmo é designado para esse conjunto, que passa agora a possuir 2 elementos, $\{(5,00; 2,00), (6,00; 3,00)\}$, e a imagem $f(x_2^*) = 12,53$ é acrescida ao vetor das imagens ótimas. O algoritmo determina então o vetor de distâncias entre o ponto x_2^* e o conjunto X, incorporando tal vetor à matriz de distâncias dos centroides ao conjunto, cuja ordem passa a ser 45×2.

O próximo passo do Algoritmo MulticlusterKDE é determinar qual das 45 observações do conjunto X apresenta a máxima distância Euclidiana à ambos os elementos do conjunto dos centroides. Para tal, o algoritmo faz uma varredura em todas as 45 observações do problema, determinando a menor distância entre cada observação e os dois centroides, resultando em um vetor contendo todas as mínimas distâncias de cada linha da matriz de distâncias. Em seguida, o algoritmo identifica o elemento do conjunto X associado a máxima distância apresentada no vetor de distâncias, que nesse caso é o

ponto $x_3 = (4,80; 3,18)$. Este elemento é então escolhido como novo ponto inicial para a iteração seguinte (ponto representado com o símbolo de "+" na cor vermelha - Figura 6), finalizando assim a segunda iteração.

Figura 6 – Determinação do terceiro centroide da clusterização



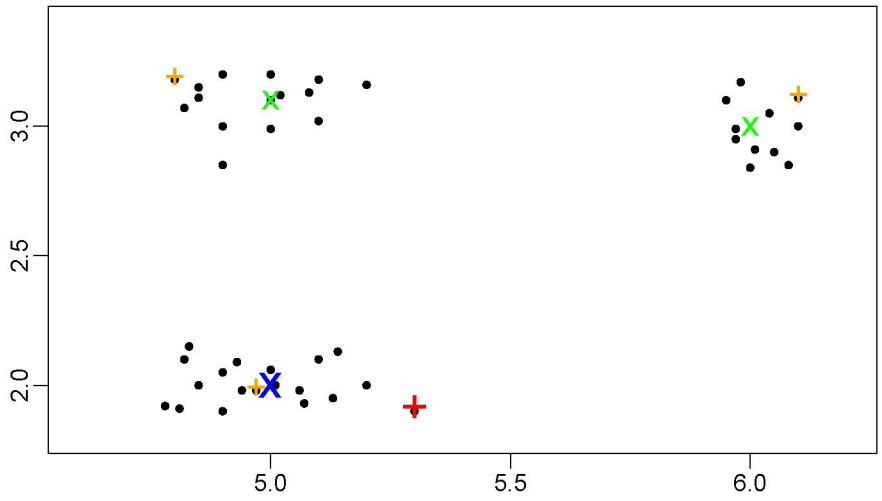
Fonte: O autor (2021).

Considerando $x_3 = (4,80; 3,18)$ como ponto inicial na minimização de $-\widehat{f}$, o algoritmo determina o minimizador $x_3^* = (5,00; 3,10)$ (ponto representado pelo símbolo "×" na cor azul - Figura 6). Novamente, o algoritmo avalia se o ponto $x_3^* = (5,00; 3,00)$ pertence ou não ao conjunto dos centroides. Como x_3^* não pertence, ele é então armazenado no conjunto, o qual passa ser constituído pelos pontos $\{(5,00; 2,00), (6,00; 3,00), (5,00; 3,10)\}$. A imagem $f(x_3^*) = 15,69$ é atribuída ao vetor de imagens e na matriz de distâncias é acrescida mais uma coluna composta pela distância de X a x_3^*. Na sequência, o algoritmo determina o elemento $x_4 = (5,30; 1,90)$, ponto representado em vermelho pelo símbolo "+" na Figura 7, pertencente ao conjunto X que apresenta a máxima distância aos centroides x_1^*, x_2^*, e x_3^*.

Utilizando $x_4 = (5,30; 1,90)$ como ponto inicial, o Algoritmo MulticlusterKDE determina $x_4^* = (5,00; 2,00)$ como minimizador de $-\widehat{f}$. Porém, esse já é um centroide e, pelo critério de parada, o algoritmo finaliza a sua primeira etapa.

No início de sua segunda etapa, o Algoritmo MulticlusterKDE primeiramente ordena o conjunto dos centroides de acordo com o vetor das imagens. Essa ordenação ocorre de forma decrescente, com o intuito de determinar quais centroides possuem o maior valor funcional (imagem), que consequentemente corresponderá, em virtude das característica da função KDE, a uma região de máxima densidade, com alta concentração de observações. Para o exemplo considerado, o conjunto dos centroides ordenado segundo

Figura 7 – Determinação do quarto centroide da clusterização



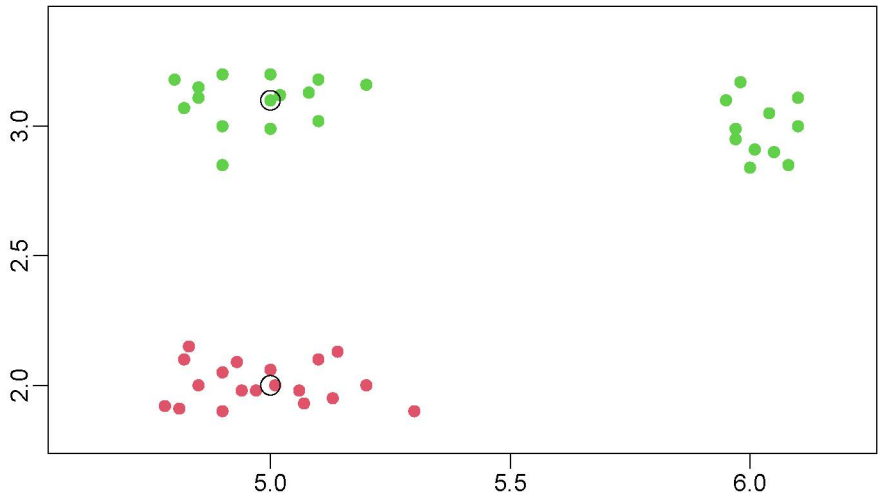
Fonte: O autor (2021).

o vetor de imagens é dado por $\{(5,00; 2,00), (5,00; 3,10), (6,00; 3,00)\}$, da mesma forma, o algoritmo ordena as colunas da matriz de distâncias.

A partir desse momento o caminho a ser executado pelo algoritmo depende da atribuição feita ao parâmetro *nc*. Nesse exemplo, o parâmetro *nc* não foi informado, então o algoritmo inicia uma busca pela melhor configuração dos centroides obtidos na etapa anterior. Para tanto, o algoritmo executa um laço, que inicia a clusterização do conjunto de dados com 2 centroides, e a partir daí é acrescentado um novo centroide até que o número máximo, determinado na primeira etapa (3 nesse exemplo), seja alcançado. Para o exemplo, há duas possibilidades de agrupamento para o conjunto X, isto é, 2 ou 3 grupos. Para avaliar qual é a melhor configuração, o Algoritmo MulticlusterKDE utiliza métricas de validação internas, sendo que para esse exemplo utilizamos a métrica discutida na seção anterior, isto é, o índice CD.

Inicialmente o algoritmo analisa o agrupamento utilizando os dois primeiros centroides do conjunto ordenado. Considerando as duas primeiras colunas da matriz de distâncias o algoritmo determina para cada uma das observações o centroide mais próximo. Assim para a primeira observação $(4,97; 1,98)$, por exemplo, o algoritmo determinou as distâncias $0,036$ e $1,12$ aos centroides $(5,00; 2,00)$ e $(5,00; 3,10)$, respectivamente. Logo, como a menor distância está relacionada ao primeiro centroide, então a primeira observação dos dados pertencerá ao grupo 1. Essa mesma análise é repetida para as outras 44 observações configurando a clusterização do problema em 2 grupos, conforme ilustrado na Figura 8. Na sequencia, o algoritmo avalia por meio do índice CD a qualidade desse agrupamento, obtendo o resultado $CD = 0,4671$.

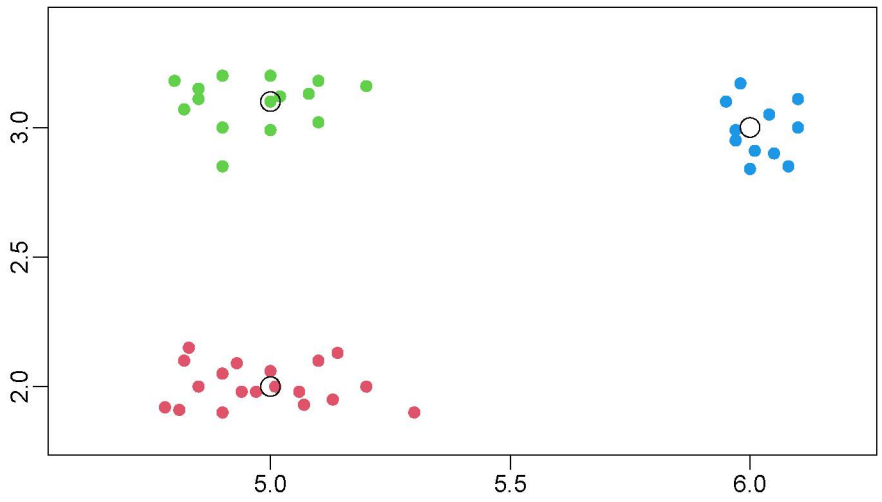
Figura 8 – Clusterização para 2 centroides



Fonte: O autor (2021).

No segundo momento, o algoritmo acrescenta o terceiro centroide e designa todas as observações ao respectivo centroide mais próximo, com base na mínima distância Euclidiana. O resultado dessa clusterização está expresso na Figura 9 e o valor obtido para o índice CD é de $CD = 0,1302$.

Figura 9 – Clusterização para 3 centroides



Fonte: O autor (2021).

Como o valor do índice CD para a clusterização com 3 grupos foi o menor, essa configuração é selecionada como resultado final pelo Algoritmo MulticlusterKDE.

3.2.3 Convergência do Algoritmo MulticlusterKDE

A seguir, enunciamos o Teorema de Weierstrass, o qual aborda conceitos que corroboram com a discussão sobre a convergência do algoritmo proposto.

Teorema 3.1 (Teorema de Weierstrass)**.** *Sejam $D \subset \mathbb{R}^n$ um conjunto compacto não-vazio e $f : D \to \mathbb{R}$ uma função contínua. Então, f tem minimizador e maximizador em D.*

Demonstração. Izmailov e Solodov (2014) [p.8, Teorema 1.2.1]. □

Esse resultado, juntamente com o teorema a seguir, contribuirão para a compreensão dos argumentos que comprovam que o Algoritmo MulticlusterKDE está bem definido e converge em um número finito de passos. Outro ponto que precisa ser levado em consideração na demonstração da convergência é o método de otimização utilizado. Como mencionado, optamos pelo Método L-BFGS-B, que é um método quase-Newton de memória limitada para problemas com restrições de caixa. Nestes termos, podemos enunciar a convergência do Algoritmo MulticlusterKDE da seguinte forma:

Teorema 3.2. *O Algoritmo MulticlusterKDE está bem definido e executa no máximo $m + 1$ passos.*

Demonstração. A cada iteração do Algoritmo MulticlusterKDE é preciso determinar x^* como minimizador de $-\widehat{f}$, a qual é uma função suave, contínua e possui derivadas de todas as ordens. Utilizamos o Método L-BFGS-B para otimização da função KDE cujos limitantes inferior e superior do conjunto de dados constituíram a restrição de caixa necessária para o emprego do método. Como os limitantes definem um conjunto limitado e compacto ($l \leq x \leq u$) e de acordo com o 3.1, todo conjunto compacto possui um minimizador, então todas as iterações do algoritmo apresentarão um ponto de mínimo, garantindo assim que o próximo passo do Algoritmo MulticlusterKDE existirá, ou seja, o passo do algoritmo é bem definido.

Para finalizar, precisamos mostrar que o algoritmo determina uma solução em um número finito de passos. Seja $k \in \mathbb{Z}_+^*$ o número de grupos determinado pelo algoritmo, e $m \in \mathbb{Z}_+^*$ o número de elementos do conjunto de dados $X \in \mathbb{R}^{m \times n}$, tal que $k \leq m$. No cenário mais favorável temos $k < m$, ou seja, o algoritmo determina em k iterações todos os pontos estacionários e distintos do problema, de forma que será preciso realizar apenas mais uma iteração, $k + 1$, para que ocorra a repetição de um centroide, atendendo ao critério de parada do algoritmo. Dessa maneira, o algoritmo convergirá em $k + 1 \leq m$ iterações. Supondo agora o pior cenário, ou seja, $k = m$, onde cada elemento do conjunto X esteja isolado, formando grupos com apenas um único elemento. Nesse cenário, o algoritmo determinaria os m centroides em m iterações e precisaria de mais uma para que ocorresse a repetição de um centroide, totalizando $m + 1$ iterações para se obter a solução.

Como m é um número finito, em qualquer cenário o algoritmo para em um número finito de iterações, mais especificamente, em no máximo $m + 1$ iterações. □

Na próxima subseção apresentamos a terceira contribuição desse trabalho que é o Algoritmo AdditiveclusterKDE.

3.3 ALGORITMO ADDITIVECLUSTERKDE

O Algoritmo AdditiveclusterKDE é um algoritmo baseado na clusterização de múltiplas amostras com seleção do melhor conjunto de centroides para o agrupamento da população (conjunto de dados) segundo uma métrica de validação interna. Para apresentar o Algoritmo AddittiveclusterKDE, primeiramente, realizamos uma descrição geral da sua execução, com a discussão da rotina principal e de suas sub-rotinas. Na sequência, apresentamos um exemplo bidimensional para facilitar a compreensão dos principais aspectos do algoritmo e, para finalizar a seção, discutimos a sua convergência.

3.3.1 Algoritmo AdditiveclusterKDE: descrição geral

O Algoritmo AdditiveclusterKDE, a ser apresentado a seguir, é constituído principalmente por um laço principal, duas tomadas de decisão e um laço secundário. Para sua execução são necessários: uma matriz $X \in \mathbb{R}^{m \times n}$ com os dados do problema, sendo m o número de observações e n o número de atributos; um parâmetro $\alpha \in \mathbb{R}_+^*$ responsável pela suavidade da função KDE, calculado conforme (2.19) para determinar a matriz H; um escalar $A \in \mathbb{Z}_+^*$ que define a quantidade de amostras utilizadas; $m_A \in \mathbb{Z}_+^*$, com $m_A \leq m$, o tamanho de cada amostra selecionada durante o processo iterativo e $nc \in \mathbb{Z}_+^*$ um número inteiro fornecido de forma opcional pelo usuário que tem a função de limitar o número de grupos e é recomendado para problemas de classificação.

No Algoritmo AdditiveclusterKDE, o objetivo do laço principal (linhas 1 a 22) é realizar a clusterização para cada uma das amostras selecionadas. A cada passo do algoritmo, este seleciona uma amostra com m_A elementos (linha 2) por meio da amostragem aleatória simples e, em seguida, ocorre a primeira tomada de decisão, dependente do parâmetro nc. Supondo que o usuário tenha conhecimento do número de grupos que constitui o problema, então o parâmetro nc é conhecido e pode ser informado no início da execução. Nesse caso, o Algoritmo AdditiveclusterKDE executa a sub-rotina nc-centroides, descrita na seção 3.3.1.2, para obter nc centroides (linha 4).

Por outro lado, se o problema não possui nenhuma informação sobre o número de grupos, o parâmetro nc não precisa e não deve ser informado, logo o Algoritmo AdditiveclusterKDE executará a sub-rotina $k-$centroides, descrita na seção 3.3.1.1, a qual determina de forma autônoma k centroides (linha 6), que corresponde, de acordo com a métrica de validação interna, a clusterização mais adequada para a amostra selecionada.

Algoritmo 2: AdditiveclusterKDE

Entrada: $X \in \mathbb{R}^{m \times n}$, $\alpha \in \mathbb{R}_+^*$, $A \in \mathbb{Z}_+^*$, $m_A \in \mathbb{Z}_+^*$, $nc \in \mathbb{Z}_+^*$

1 **Para** $i = 1 : A$ **faça**
2 Selecione $\overline{X} \in \mathbb{R}^{m_A \times n} \subset X \in \mathbb{R}^{m \times n}$ por amostragem aleatória simples;
3 **Se** $nc \neq \emptyset$ **então**
4 Determine $S \in \mathbb{R}^{nc \times n}$ centroides pela sub-rotina $nc-$centroides tendo $\overline{X} \in \mathbb{R}^{m_A \times n}$, nc e α como parâmetros de entrada.
5 **Senão**
6 Determine $S \in \mathbb{R}^{k \times n}$ centroides pela sub-rotina $k-$centroides tendo $\overline{X} \in \mathbb{R}^{m_A \times n}$ e α como parâmetros de entrada.
7 **Fim**
8 Defina: $B = \emptyset$, $s = \infty$ (para índice CD ou DB) ou $s = -\infty$ (para silhueta);
9 **Para** $i = 1 : m$ **faça**
10 $t = \infty$;
11 **Para** $j = 1 : k$ **faça**
12 **Se** $dist(S(j,:), X(i,:)) < t$ **então**
13 $\omega = j$, $t = dist(S(j,:), X(i,:))$
14 **Fim**
15 **Fim**
16 $B = B \cup \{\omega\}$
17 **Fim**
18 $r =$ métrica de validação de X de acordo com B;
19 **Se** r *é melhor que* s **então**
20 $s = r$, $C = B$, $P = S$
21 **Fim**
22 **Fim**
23 **Retorna** C,P,s

Independentemente da decisão tomada, ao final desta etapa, o Algoritmo AdditiveclusterKDE retorna com um conjunto de centroides e inicia o seu laço secundário, responsável pela designação de todas as observações a um dos centroides com base no critério de mínima distância Euclidiana (linha 9 a 17). Após a clusterização, o algoritmo emprega uma métrica de validação interna (índices CD, DB ou coeficiente de silhueta) para medir a qualidade da clusterização proveniente dos centroides obtidos pela amostra $\overline{X}$ (linha 18).

Na sequência, o Algoritmo AdditiveclusterKDE tem a segunda tomada de decisão, a qual é responsável por avaliar se o valor da métrica de validação, proveniente da iteração atual, tem o melhor valor obtido até o momento (linhas 19 a 21). Em caso afirmativo, o algoritmo armazena as informações referentes a clusterização, centroides e métrica de validação, caso contrário, a clusterização da iteração é descartada. O Algoritmo AdditiveclusterKDE é finalizado assim que executar a clusterização de todas as A amostras selecionadas, armazenando a melhor clusterização do conjunto X de acordo com

a métrica de validação utilizada.

3.3.1.1 Sub-rotina k-centroides

A sub-rotina, denominada k-centroides, Algoritmo 3 a seguir, tem o objetivo de determinar k centroides de forma autônoma, a cada repetição do laço principal do Algoritmo 2, quando o parâmetro nc não é fornecido. Embora, nessa pesquisa, estejamos abordando o k-centroides como uma sub-rotina, o mesmo equivale a um algoritmo de clusterização completo, podendo ser empregado na clusterização de dados multidimensionais como algoritmo principal.

O k-centroides é um algoritmo construtivo que aumenta o número de grupos (centroides) a cada iteração por meio do particionamento do conjunto de dados. Desse modo, o processo iterativo é iniciado com apenas um único grupo e, a cada nova iteração, caso os critérios de parada não sejam atendidos, um novo centroide é adicionado, consequentemente, uma nova clusterização.

A determinação desse novo grupo é realizada por meio da obtenção de um segundo centroide no grupo cujos elementos são mais heterogêneos. A verificação do grau de heterogeneidade dos elementos é feita pela dispersão interna do grupo, ou seja, pelo cálculo da distância média de seus elementos ao seu respectivo centroide.

Estruturalmente, o Algoritmo 3 é constituído por: um laço principal, responsável por controlar o número de iterações; por 3 tomadas de decisão, as quais estão interligadas entre si e são responsáveis por finalizar o laço principal; e por um laço secundário, responsável pela designação dos elementos de X ao centroide mais próximo de acordo com a mínima distância Euclidiana.

Para o Algoritmo k-centroides são necessários: uma matriz $X \in \mathbb{R}^{m_A \times n}$ com os dados da amostra selecionada, sendo m_A o número de observações e n o número de variáveis; um parâmetro $\alpha \in \mathbb{R}_+^*$ usado na expressão (2.19) para determinar a matriz diagonal H, responsável pela suavidade da função KDE; os vetores limitantes $l, u \in \mathbb{R}^n$ para a restrição de caixa utilizada no processo de otimização da função e, por fim, um ponto inicial escolhido arbitrariamente entre os elementos de X.

Em seu laço principal (linhas 3 a 34), a primeira operação realizada é a otimização da função KDE com kernel Gaussiano (linha 4). Os pontos estacionários determinados na otimização são definidos como centroides da clusterização. Optamos por utilizar para o processo de otimização da função KDE o Método L-BFGS-B, disponibilizado no *Software R* na rotina *optim*, visto que esse é um método quase-Newton com memória limitada e com restrição de caixa, o qual assegura uma boa velocidade de convergência e se adapta de forma natural à metodologia empregada pelo Algoritmo $k-$ centroides.

Como mencionado, no início de cada iteração um minimizador x^* é determinado,

Algoritmo 3: k-centroides

Entrada: $X \in \mathbb{R}^{m_A \times n}$,$\alpha \in \mathbb{R}_+^*$;

Escolha $k = 0$, $x^k \in X$ e **faça** $S = \emptyset$, $Parar = 0$, $rep = 0$, $t = 0$, $D = \emptyset$, $s = \infty$ (para índice CD ou DB) ou $s = -\infty$ (para silhueta), $l, u \in \mathbb{R}^n$ tal que $l_j = \min\{X(:,j)\}$ e $u_j = \max\{X(:,j)\}$ para $j = 1, \ldots, n$;

Determine: H e $\widehat{f}$ de acordo com (2.19) e (2.3);

Enquanto $Parar = 0$ **faça**

  A partir de x^k determine: $x^* \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ -\widehat{f}(x) : l \leq x \leq u \right\}$;

  Se $x^* \in S$ **então**

    $Parar = 1$;

  Senão

    $S = S \cup x^*$, $k = k + 1$, $B = ones(1 : m_A)$, $D = D \cup \{dist(x^*, X(i,:)), i = 1, \ldots, m_A\}$;

    Para $i = 1 : m_A$ **faça**

      Para $j = 1 : t$ **faça**

        Se $dist\,(X(i,:), S(j,:)) = min\{D(i,:)\}$ **então**

          $B[i] = j$

        Fim

      Fim

    Fim

    $r =$ métrica de validação de X de acordo com B.;

    Se r *é melhor que* s **então**

      $s = r$, $C = B$, $P = S$,$rep = 0$

    Senão

      $rep = rep + 1$

    Fim

    Se $r > 1$ *ou* $rep > 10$ **então**

      $Parar = 1$

    Senão

      Determine: $Med = \left\{ \frac{1}{m_j} \sum_{i=1}^{m_j} \|X_i - \mu_j\| \right\} \quad (j = 1, \ldots, p)$;

      $\omega = j$ tal que $Med(j) = max(Med)$;

      $\overline{X} \subset X$ onde $X(i,) \in \overline{X} \iff B(i) = \omega$;

      $l_j = \min\{\overline{X}(:,j)\}$ e $u_j = \max\{\overline{X}(:,j)\}$ para $j = 1, \ldots, n$;

      $m_t = max \left\{ \min_{i=1,\ldots,m_\omega} \{dist(S(\omega,:), \overline{X}(i,:))\} \right\}$;

      $x^{t+1} = x \in \overline{X} | dist(x, S(\omega,:)) = m_t$;

      $t = t + 1$

    Fim

  Fim

Fim

Retorna C,s,P

levando o algoritmo a sua primeira tomada de decisão, que consiste em verificar se esse minimizador é distinto daqueles determinados em iterações anteriores (linhas 3 a 34). Caso x^* já faça parte do conjunto dos centroides, o algoritmo é encerrado, caso contrário, x^* é designado ao conjunto dos centroides. A repetição de um ponto estacionário é o primeiro critério de parada do Algoritmo k-centroides, pois devido à forma de escolha do ponto inicial para o método de otimização a partir da segunda iteração (processo esse que será descrito mais adiante), uma vez repetido um minimizador, o algoritmo entra num loop infinito, obtendo sempre o mesmo ponto x^* como minimizador de $-\widehat{f}$, inviabilizando assim a continuidade do processo.

Considerando que x^* seja um minimizador diferente aos obtidos nas iterações anteriores, então o algoritmo designa x^* para o conjunto dos centroides (linha 8) e, partir disso, realiza uma nova clusterização pelo critério de mínima distância Euclidiana de cada um dos elementos do conjunto X aos centroides determinados (linha 9 a 15). Após essa etapa, o algoritmo avalia a qualidade dessa clusterização, por meio da aplicação de uma das métricas de validação interna (linha 16).

Com base no valor da métrica, o algoritmo realiza sua segunda tomada de decisão (linhas 17 a 21), verificando se a clusterização atual é a melhor. Em caso afirmativo, o algoritmo armazena as informações sobre os centroides, a clusterização e o valor da respectiva métrica (linha 18), além de zerar o contador de iterações sucessivas consideradas improdutivas. Em caso negativo, o algoritmo acrescenta em uma unidade o contador de iterações improdutivas (linha 20).

Em seguida, ainda com base no valor da métrica resultante na iteração atual, o algoritmo tem sua terceira e última tomada de decisão, responsável por avaliar a possibilidade de finalização do algoritmo (linhas 22 a 32). Nesse trabalho, optamos por considerar três possíveis critérios de parada, dois envolvendo a métrica de validação e um relacionado a repetição do centroide (minimizador de $-\widehat{f}$) já mencionado. Com relação a métrica de validação, o algoritmo é finalizado quando o valor da métrica é superior a um valor pré-fixado. Nesse trabalho adotamos para as métricas índice CD e DB o valor 1, enquanto que para o coeficiente de silhueta, adotamos o valor 1 para o resultado da expressão $1 - \mathit{silhueta}$, o qual é equivalente ao coeficiente de silhueta apresentar valores menores que zero. Clusterizações com essas avaliações, segundo as métricas de validação expostas, apresentam características insatisfatórias com grupos dispersos e/ou centroides próximos. O algoritmo é finalizado também quando não há melhora no valor da métrica de validação por um número fixo de iterações consecutivas. Após testes numéricos, escolhemos fixar em 10 o número máximo de iterações improdutivas.

Caso nenhum dos critérios seja atendido, o Algoritmo 3 continua sua execução em busca de um grupo que possui maior heterogeneidade entre seus elementos, verificando a máxima distância média dos elementos de cada grupo a seu respectivo centroide (linhas

25 e 26). Uma vez definido o grupo mais heterogêneo (linha 27), o algoritmo determina os limitantes inferior (l) e superior (u) (linha 28), delimitando assim uma nova região de busca na otimização da função KDE.

Para finalizar a iteração, o algoritmo determina um novo ponto inicial para iteração seguinte (linhas 29 e 30). Como o $k-$centroides sempre almeja por um novo minimizador dentro do grupo mais heterogêneo, a escolha do ponto inicial não pode ser arbitrária, mas sim escolhida de forma a evitar uma repetição do minimizador. Nesse sentido, o ponto inicial escolhido, a partir da segunda iteração, tem como característica ser um elemento pertencente ao conjunto X interno a região definida pelos limitantes l e u e que possua a máxima distância ao centroide já pertencente ao grupo selecionado. Essa estratégia tende a evitar uma repetição do minimizador da função KDE num cenário onde existe mais de um ponto de mínimo local. No entanto, tal escolha pode gerar um loop infinito em caso de repetição do minimizador, visto que, uma vez obtido um mesmo ponto de mínimo, o agrupamento se mantém o mesmo, assim como o grupo mais heterogêneo e, consequentemente, a escolha do ponto inicial, formando dessa forma um ciclo sem fim. Por outro lado, em todo o processo há apenas um único ponto inicial totalmente randômico, o que garante ao algoritmo que uma vez escolhido o mesmo ponto inicial na primeira iteração a clusterização final será sempre a mesma.

Ao fim de sua execução, o Algoritmo 3 retorna k centroides que produziram a melhor clusterização, segundo uma métrica de validação, dentre todas as iterações. Esses centroides serão utilizados pelo Algoritmo AdditiveclusterKDE para expandir a clusterização a todo o conjunto de dados.

3.3.1.2 Sub-rotina *nc*-centroides

Nessa seção apresentamos a sub-rotina *nc*-centroides, Algoritmo 4 a seguir, vinculada ao Algoritmo AdditiveclusterKDE, a qual objetiva determinar exatos nc centroides, com $nc \in \mathbb{Z}_+^*$. O seu funcionamento é semelhante a sub-rotina k-centroides, porém mais simples, visto que não necessita realizar avaliações da clusterização a cada iteração, além de ter seus critérios de paradas extremamente definidos pelo parâmetro nc. A utilização dessa rotina é recomendada para problemas em que o usuário já tenha o conhecimento do número de grupos e/ou problemas de classificação.

Para execução do Algoritmo *nc*-centroides são necessários: uma matriz $X \in \mathbb{R}^{m_A \times n}$ contendo os dados da amostra selecionada (ou dados do problema), sendo m_A o número de observações e n o número de variáveis; um parâmetro $\alpha \in \mathbb{R}_+^*$ usado na expressão (2.19) para determinar a matriz diagonal H, responsável pela suavidade da função KDE; um parâmetro nc responsável por estabelecer o número máximo de centroides que o algoritmo obterá durante sua execução; os vetores limitantes $l, u \in \mathbb{R}^n$ para a restrição de caixa, utilizada no processo de otimização da função e, por fim, o ponto

Algoritmo 4: nc-centroides

Entrada: $X \in \mathbb{R}^{m_A \times n}$, $\alpha \in \mathbb{R}_+^*$, $nc \in \mathbb{Z}_+^*$;

1 **Escolha** $k = 0$, $x^k \in X$ e **faça** $S = \emptyset$, $Parar = 0$, $D = \emptyset$, $l, u \in \mathbb{R}^n$ tal que $l_j = \min\{X(:,j)\}$ e $u_j = \max\{X(:,j)\}$ para $j = 1, \ldots, n$;

2 **Determine**: H e $\widehat{f}$ de acordo com (2.19) e (2.3);

3 **Enquanto** $Parar = 0$ **faça**

4 A partir de x^k determine: $x^* \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ -\widehat{f}(x) : l \leq x \leq u \right\}$;

5 **Se** $x^* \in S$ **então**

6 $Parar = 1$;

7 **Senão**

8 $S = S \cup x^*$, $B = ones(1 : m_A)$;

9 $D = D \cup \{dist(x^*, X(i,:)), i = 1, \ldots m_A\}$;

10 **Para** $i = 1 : m_A$ **faça**

11 **Para** $j = 1 : k$ **faça**

12 **Se** $dist\,(X(i,:), S(j,:)) = min\{D(i,:)\}$ **então**

13 $B[i] = j$

14 **Fim**

15 **Fim**

16 **Fim**

17 **Se** $k = nc$ **então**

18 $Parar = 1$, $C = B$, $P = S$

19 **Senão**

20 **Determine**: $Med = \left\{ \frac{1}{m_j} \sum_{i=1}^{m_j} \|X_i - \mu_j\| \right\} \quad (j = 1, \cdots, k)$;

21 $\omega = j$ tal que $Med(j) = max(Med)$;

22 $\overline{X} \subset X$ onde $X(i,:) \in \overline{X} \iff B(i) = \omega$;

23 $l_j = \min\{\overline{X}(:,j)\}$ e $u_j = \max\{\overline{X}(:,j)\}$ para $j = 1, \ldots, n$;

24 $m_k = max \left\{ \min_{i=1,\cdots,m_\omega} \left\{ dist(S(\omega,:), \overline{X}(i,:) \right\} \right\}$;

25 $x^{k+1} = x \in \overline{X} \mid dist(x, S(\omega,:)) = m_k$;

26 $k = k + 1$

27 **Fim**

28 **Fim**

29 **Fim**

30 **Retorna** C,P

inicial (x^k) utilizado no processo de otimização da função, sendo o primeiro escolhido de forma aleatória entre um dos elementos de X.

O Algoritmo 4 é composto por um laço principal (linhas 3 a 29), responsável por controlar o número de iterações sendo limitado superiormente pelo parâmetro *nc*. Dentro do laço há duas tomadas de decisão interligadas, que controlam os critérios de parada e, além disso, um laço secundário responsável pela designação das observações aos centroides pelo critério de mínima distância Euclidiana. Ao iniciar o laço principal, a

primeira operação realizada é a otimização da função KDE (linha 4). Da mesma forma que na sub-rotina $k-$ centroides, nessa também utilizamos o Método L-BFGS-B.

A cada nova iteração o $nc-$ centroides obtém um minimizador de $-\widehat{f}$ candidato a centroide. Sobre esse minimizador, o algoritmo realiza sua primeira tomada de decisão (linhas 5 a 28), verificando se esse já é um centroide. Caso se verifique uma repetição, o algoritmo encerra sua execução, caso contrário, acrescenta o minimizador ao conjunto dos centroides (linha 8), atualiza a matriz de distâncias dos centroides ao conjunto X (linha 9) e realiza a designação das observações de X aos k centroides já determinados (linhas 10 a 16). Na sequência, a rotina realiza sua última tomada de decisão (linhas 17 a 27), em que é verificado se com a inserção de mais esse ponto, o conjunto dos centroides conterá exatos nc elementos, ou seja, $k = nc$. Em caso afirmativo, o algoritmo encerra sua execução retornando como solução os nc centroides, caso contrário, retoma o processo de busca por novos centroides, determinando o grupo mais heterogêneo, os novos limitantes e o ponto inicial para a otimização da função KDE.

Para verificar qual grupo é o mais heterogêneo, o Algoritmo 4 determina a distância média de todos os elementos de cada grupo ao seu respectivo centroide (linha 20) e aquele que apresentar a maior distância média (linhas 21 e 22), será o grupo utilizado para definir os novos limitantes l e u (linha 23) para o processo de otimização da função KDE. Por fim, para encerrar a iteração, o algoritmo determina um novo ponto inicial que possui a característica de pertencer à região delimitada por $l \leq x \leq u$ e apresentar a máxima distância Euclidiana ao centroide do grupo mais heterogêneo (linhas 24 e 25).

O algoritmo repete o processo até atender ao critério de parada. Ao fim de sua execução retorna k centroides, $k \leq nc$, que serão utilizados pelo Algoritmo 2 para expandir a clusterização a todo o conjunto de dados.

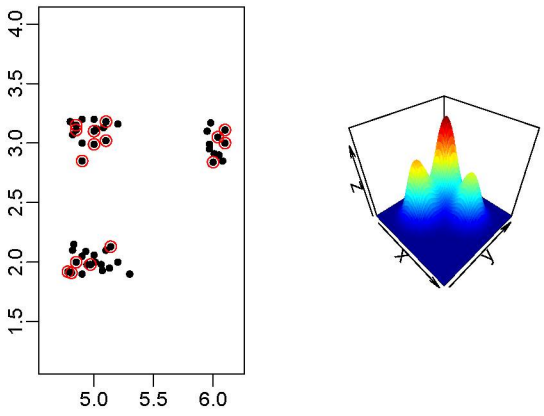
3.3.2 Exemplo do Algoritmo AdditiveclusterKDE

Após ter definido formalmente o Algoritmo AdditiveclusterKDE e suas duas sub-rotinas, apresentamos agora um exemplo simples, no espaço bidimensional, a fim de ilustrar o seu funcionamento. De forma semelhante à Seção3.2, utilizamos a mesma base de dados disposta na Tabela 2 e os seguintes parâmetros: $\alpha = 0,75$, $A = 2$ e $m_A = 18$, que correspondem ao parâmetro de suavidade da função KDE, número de amostras e tamanho de cada amostra, respectivamente. Optamos por não informar o parâmetro nc, de modo que o algoritmo execute apenas a sub-rotina k-centroides. Dentro da sub-rotina, optamos por utilizar a métrica de validação, índice CD, como critério de seleção da melhor clusterização.

Considerando a primeira iteração, o lado esquerdo da Figura 10 apresenta o

gráfico de dispersão do conjunto de dados, ou seja, 45 observações representados pelos pontos em preto, conforme Tabela 2. Os pontos destacados por círculos em vermelho, representam a primeira amostra composta por 18 elementos, 40% da população, selecionados do conjunto de dados pelo processo de amostragem aleatória simples. Já o lado direito da Figura 10 ilustra a função KDE proveniente dos dados da amostra.

Figura 10 – Gráfico de dispersão dos dados e elementos selecionados para compor a primeira amostra e a respectiva função KDE



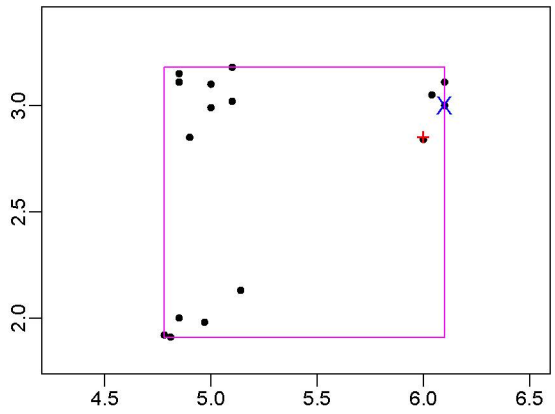
Fonte: O autor (2021).

Uma vez selecionada a amostra, o Algoritmo AdditiveclusterKDE entra na subrotina $k-$centroides (Algoritmo 3), responsável por determinar os centroides da amostra que, posteriormente, serão utilizados como centroides para a clusterização da população. O Algoritmo 3 tem como parâmetros de entrada o subconjunto contendo os dados da amostra, $\overline{X} \in \mathbb{R}^{18\times 2}$ e o escalar $\alpha = 0,75$ utilizado para determinação da matriz H e da função KDE, conforme expressões (2.19) e (2.3), respectivamente. Cabe salientar que, diferentemente do Algoritmo MulticlusterKDE, o Algoritmo AdditiveclusterKDE determina uma matriz H e uma função KDE para cada amostra, ou seja, A vezes.

No início, o Algoritmo $k-$centroides determina os limitantes inferior e superior da amostra, utilizados na restrição de caixa representada na Figura 11 pelo retângulo na cor magenta. Em seguida, seleciona de forma aleatória o ponto $x_1 = (6,00; 2,84)$, representado na Figura 11 pelo símbolo "+" na cor vermelha, como ponto ponto inicial o qual é empregado no processo de otimização da função KDE.

Tomando x_1 como ponto inicial no processo de otimização da função KDE pelo Método L-BFGS-B, obtemos $x_1^* = (6,10; 3,00)$ como minimizador de $-\widehat{f}$, representado na Figura 11 pelo símbolo "×" na cor azul. Cabe observar que o ponto inicial sempre será uma das observações da amostra que atenda a restrição de caixa, por outro lado, o minimizador não é necessariamente um elemento da amostra, mas ainda é um ponto que atende à restrição $l \leq x \leq u$. Sobre a restrição de caixa, na primeira iteração do

Figura 11 – Primeiro centroide



Fonte: O autor (2021).

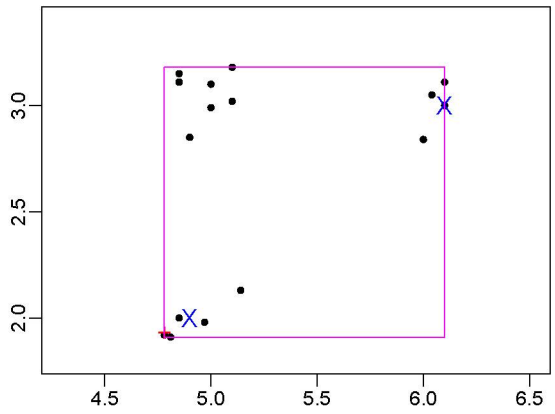
Algoritmo 3, ela abrange toda a amostra, visto que todos os elementos são interpretados como um único grupo.

Após a obtenção de um novo minimizador, o Algoritmo 3 avalia se tal ponto já é um centroide. Como x_1^* é o primeiro minimizador determinado, não há nenhuma restrição ao seu uso como centroide. Tendo x_1^* como primeiro centroide, o algoritmo realiza a designação das observações ao centroide mais próximo, que obviamente será x_1^*. Cabe observar que o agrupamento propiciado pelo primeiro centroide constituirá em um único grupo formado pelo próprio conjunto. Essa clusterização obviamente não terá atendido a nenhum critério de parada do algoritmo, além de apresentar os mesmos limitantes inferior e superior. No entanto, com a obtenção do primeiro centroide, o processo de escolha do ponto inicial tem alterações, isto é, a partir da existência de centroides, a escolha de pontos inicias para o processo de otimização não ocorre mais de forma aleatória e sim determinística. Logo, para finalizar a primeira iteração, o algoritmo busca por um novo ponto inicial que atenda as condições de pertencer a caixa e ter a máxima distância Euclidiana ao ponto x_1^*. No nosso exemplo esse ponto é $x_2 = (4,78; 1,92)$ representado, pelo simbolo "+" na cor vermelha, na Figura 12.

Na segunda iteração, partindo de x_2 como ponto inicial, o algoritmo determina o minimizador $x_2^* = (4,90; 2,00)$, representado na Figura 12 pelo símbolo "$\times$" na cor azul. Com a determinação deste segundo minimizador, o $k-$centroides verifica se esse é igual ou não ao centroide da primeira iteração. Como podemos observar $x_1^* \neq x_2^*$, logo, ele é designado ao conjunto dos centroides e o processo é repetido.

Uma vez determinado dois centroides, realizamos a designação de todos os elementos da amostra aos centroides pelo critério de mínima distância Euclidiana, resultando em dois grupos, conforme Figura 13. A partir desta clusterização, o índice CD obtido é de

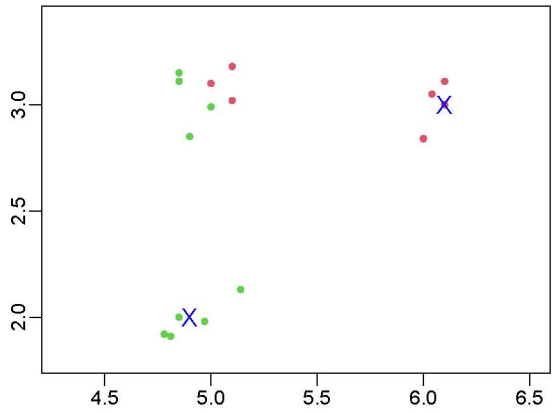
Figura 12 – Segundo centroide



Fonte: O autor (2021).

$CD = 0,559$. Nesse momento, o algoritmo avalia se a clusterização atual é a melhor. Considerando que na segunda iteração ocorre a primeira divisão da amostra em subgrupos, então o índice CD dessa iteração é melhor.

Figura 13 – Clusterização com 2 centroides

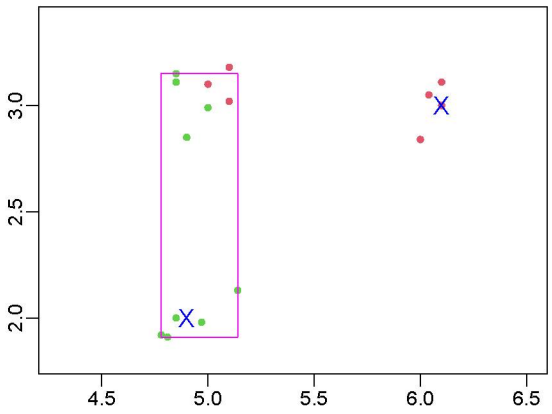


Fonte: O autor (2021).

Com a subdivisão da amostra em dois grupos, o algoritmo avalia qual desses é mais heterogêneo, por meio da máxima distância média dos seus elementos ao respectivo centroide. No exemplo, o grupo 2 (elementos na cor verde na Figura 13) é o mais heterogêneo e sobre esse, o algoritmo constrói as novas restrições de caixa, conforme mostrado na Figura 14.

Dentro da nova restrição de caixa, imposta pelos novos limitantes l e u, sempre haverá um centroide determinado em iterações anteriores e partindo desse o algoritmo seleciona o elemento interno à caixa que apresente a máxima distância ao centroide.

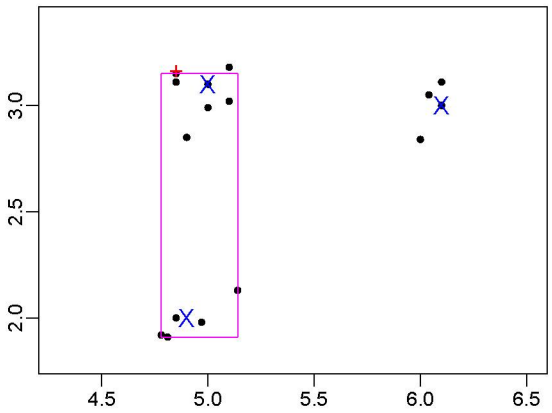
Figura 14 – Segunda restrição de caixa



Fonte: O autor (2021).

Considerando nosso exemplo, o ponto $x_3 = (4,85; 3,15)$, representado na Figura 15 pelo símbolo + na cor vermelha, é o elemento que apresenta a máxima distância a x_2^*. Com isso a segunda iteração é finalizada.

Figura 15 – Terceiro centroide

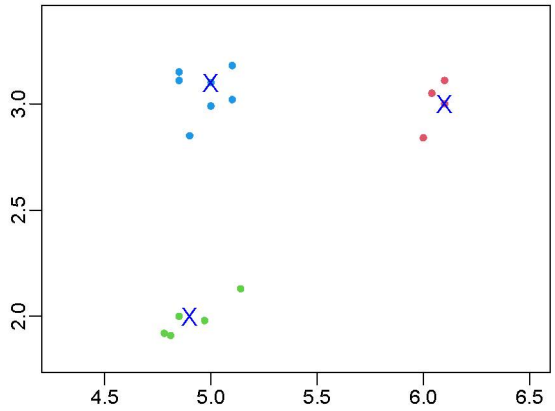


Fonte: O autor (2021).

Tomando x_3 como ponto inicial na terceira iteração, o processo de otimização retorna com $x_3^* = (5,00; 3,10)$ como minimizador da função KDE, conforme apresentado na Figura 15 pelo símbolo $\times$ na cor azul. Com a determinação de x_3^*, o algoritmo verifica que esse ainda não pertence ao conjunto dos centroides e realiza a designação de todas as observações da amostra aos 3 centroides, resultando em um agrupamento com 3 grupos, conforme ilustrado pela Figura 16. Avaliando essa nova clusterização por meio do índice CD, obtemos $CD = 0,1262$, e comparado com o resultado da iteração anterior $0,1262 < 0,559$ conclui-se que até o momento a clusterização com 3 grupos é a mais adequada para

o conjunto de dados.

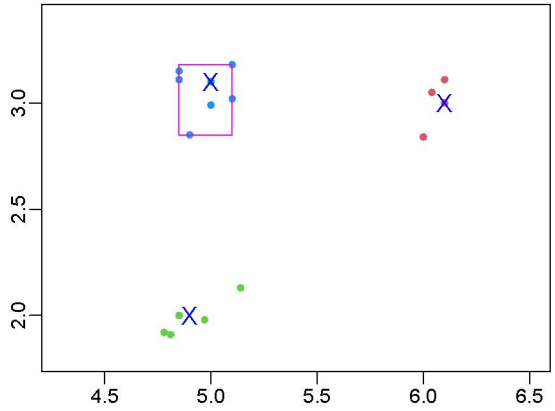
Figura 16 – Clusterização com 3 centroides



Fonte: O autor (2021).

Como houve uma redução no valor do índice CD, nenhum dos critérios de parada do algoritmo foram atendidos, logo a execução continua. Como próximo passo, o algoritmo determina qual dos 3 grupos é o mais heterogêneo por meio da máxima distância média dos elementos de cada grupo aos seus respectivos centroides. Em nosso exemplo, o grupo 3, representado pelos pontos em azul na Figura 16, é o grupo mais heterogêneo e sobre este, o algoritmo determina os limitantes inferior e superior que definem a nova restrição de caixa, ilustrada na Figura 17.

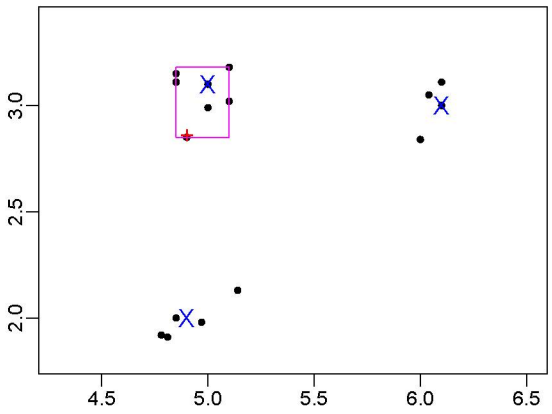
Figura 17 – Terceira restrição de caixa



Fonte: O autor (2021).

Com base nessa nova restrição, o algoritmo seleciona o ponto $x_4 = (4,90; 2,85)$ como ponto interior à caixa que apresenta a máxima distância ao centroide x_3^*. Tomando x_4 como ponto inicial, o Método L-BFGS-B retorna o minimizador $x_4^* = (6,00; 3,10)$, con-

Figura 18 – Quarto centroide



Fonte: O autor (2021).

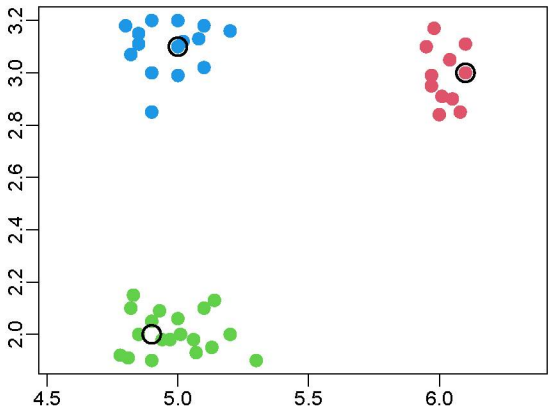
forme apresentado na Figura 18. No entanto, $x_4^* = x_3^*$, encerrando com isso o laço principal do Algoritmo k-centroides, de acordo com o critério de parada referente a repetição de centroides. Com o término do processo de busca, para nosso exemplo, a configuração com 3 grupos, conforme mostrado na Figura 16, é a melhor subdivisão da primeira amostra, com $CD = 0,1262$.

Os centroides obtidos pela sub-rotina $k-$centroides são utilizados pelo Algoritmo AdditiveclusterKDE para expandir a clusterização a todas as observações do conjunto de dados (população) pelo critério de mínima distância Euclidiana. No nosso exemplo essa expansão resulta na clusterização apresentada pela Figura 19, a qual avaliada pelo índice CD apresenta um valor de $CD = 0,1334$. Devemos relembrar que os procedimentos descritos anteriormente são provenientes da primeira amostra, que resultará na primeira clusterização viável do problema.

Com a clusterização proveniente dos centroides da primeira amostra o Algoritmo AdditiveclusterKDE finaliza sua primeira repetição do laço principal. Na segunda repetição, este seleciona uma nova amostra, com mesmo tamanho que a anterior, e reinicia o processo. Por se tratar de um processo iterativo análogo, omitiremos os detalhes da sua execução, apresentado apenas a amostra selecionada, a função KDE (Figura 20) e o resultado final (Figura 21), para fins de comparação.

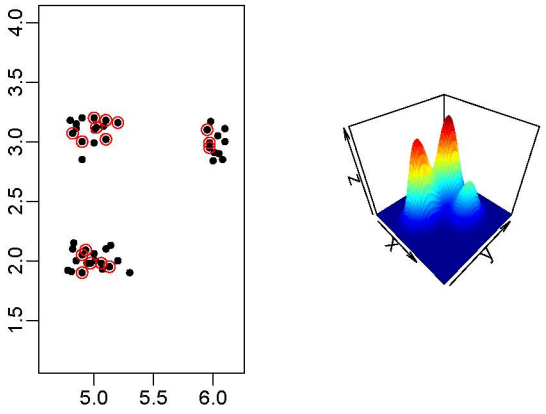
Como podemos observar, por ser um exemplo simples e com 3 grupos bem definidos, mesmo utilizando amostras diferentes, o algoritmo determinou o mesmo agrupamento com apenas pequenas variações nos centroides. Essas variações não propiciaram nenhuma mudança na clusterização dos dados, consequentemente, não produziram um valor da métrica de validação inferior à informada na primeira iteração, ocasionando o descarte da solução produzida pela segunda amostra. Como havíamos informado no início da execução

Figura 19 – Clusterização da população com centroides provenientes da amostra 1



Fonte: O autor (2021).

Figura 20 – Gráfico de dispersão dos dados e elementos selecionados para segunda amostra e a respectiva função KDE

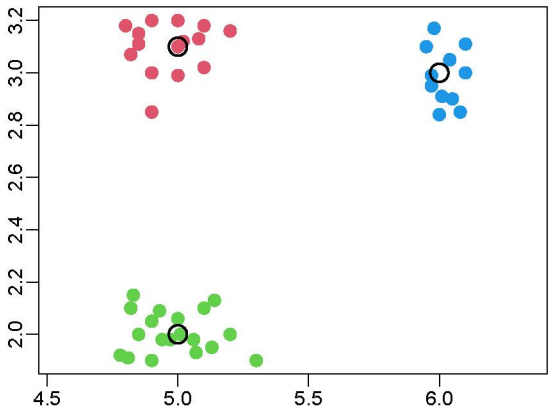


Fonte: O autor (2021).

do AdditiclusterKDE o parâmetro $A = 2$, logo apenas duas amostras foram utilizadas, sendo os resultados da primeira selecionada como solução ótima pelo Algoritmo AdditiveclusterKDE, conforme apresentado na Figura 19.

Uma última característica interessante a ser observada nos resultados provenientes das duas amostras é a ordem em que os centroides foram determinados. Observando as Figuras 19 e 21, percebemos que a clusterização é exatamente a mesma, no entanto, as cores estão invertidas. O *Software R*, tem como ordem padrão das cores de plotagem a sequência: vermelho, verde, azul. Logo a disposição das cores nos gráficos representam a ordem que os centroides foram determinados, enfatizando que apesar das duas amostras resultarem, no final, em um mesmo agrupamento, o caminho percorrido foi diferente. Essa diferença é proveniente da escolha do ponto inicial, o qual é o único escolhido de forma

Figura 21 – Clusterização da população com centroides provenientes da amostra 2



totalmente aleatória.

3.3.3 Convergência do Algoritmo AdditveclusterKDE

Nessa seção apresentamos argumentos que garantem que o algoritmo AdditveclusterKDE está bem definido e para em um número finito de passos. Tal resultado será apresentado pelo teorema a seguir.

Teorema 3.3. *O Algoritmo AdditiveclusterKDE está bem definido e executa no máximo* $A \cdot (m_A + 1)$ *passos.*

Demonstração. Para provar a convergência do Algoritmo AdditiveclusterKDE precisamos apenas provar a convergência das sub-rotinas $k-$centroides e $nc-$centroides, visto que a otimização da função KDE ocorre durante suas execuções.

A cada iteração dos algoritmos $k-$centroides ou $nc-$centroides é preciso determinar x^* como minimizador de $-\widehat{f}$, cuja função é contínua, suave, com derivadas de todas as ordens, sujeito a restrição de caixa $l \leq x \leq u$, definindo assim um conjunto compacto como domínio de busca. De acordo com o Teorema de Weierstrass (Teorema 3.1) a cada iteração existirá pelo menos um ponto de mínimo global contido no conjunto compacto $l \leq x \leq u$, sendo esse um ponto estacionário ou ponto pertencente à fronteira, de toda forma, ao final da execução do processo de otimização da função KDE, os Algoritmos 3 ou 4 retornam um ponto x^*, candidato a centroide, garantindo dessa forma que os algoritmos estão bem definidos.

Para completar a demonstração, falta mostrarmos que o número de passos executados pelas sub-rotinas é um número finito. Consideremos primeiramente, por simplicidade, o Algoritmo $nc-$centroides. Esse exige, a priori, o número máximo de grupos e, por consequência, o número máximo de passos executados. Supondo que $nc \leq m_A$, temos

como pior caso $nc = m_A$, onde o Algoritmo 4 tentará obter uma quantidade de centroides igual à quantidade de observações da amostra, ou seja, cada grupo terá apenas um elemento que será o próprio centroide. Nessas condições, o número de passos será de no máximo m_A. Supondo agora que $nc > m_A$, temos o valor do parâmetro nc superior ao número de elementos da amostra. Nesta situação, o Algoritmo 4 buscaria encontrar os nc centroides e no pior dos cenários, no passo $m_A + 1$ encontraria um centroide repetido, ocasionando o encerramento do algoritmo pela primeira condição de parada. Logo no pior cenário, o $nc-$centroides executa no máximo $m_A + 1$ iterações, o qual é um valor finito.

Analisando agora o número máximo de passos do Algoritmo $k-$centroides. Considerando como pior cenário, a subdivisão da amostra de m_A elementos em exatos m_A grupos, o algoritmo precisaria realizar m_A iterações para obter os m_A grupos e mais uma para repetir um centroide, atendendo dessa forma o critério de parada, ou seja, ele executaria no máximo $(m_A + 1)$ iterações, que é um número finito. No caso em que o número de grupos formados pelo algoritmo é menor que a quantidade de observações da amostra, isto é, $k < m_A$, o algoritmo $k-$centroides executaria um número finito de iterações, na ordem de $k + 10$, sendo o valor 10 fixado como número máximo de iterações improdutivas e $k + 10 \leq m_A + 1$. Em ambos os cenários, o algoritmo $k-$centroides executa um número finito de passos.

Dessa forma, temos que os algoritmos $k-$centroides e $nc-$centroides são bem definidos e executam no máximo $(m_A + 1)$ iterações. Como o Algoritmo AdditiveclusterKDE executa uma dessas duas sub-rotinas A vezes, então o número máximo de passos proveniente do Algoritmo AdditiveclusterKDE é de $A \cdot (m_A + 1)$, o qual é um número finito. □

Na próxima subseção apresentamos a quarta e última contribuição da tese a qual denominamos de Algoritmo TreeKDE.

3.4 ALGORITMO TREEKDE

Esta seção destina-se a descrever uma nova abordagem, denominada Algoritmo TreeKDE, que se baseia em uma estrutura de árvore de decisão associada à otimização da função do estimador de densidade kernel. Ela utiliza a otimização da função unidimensional do KDE, obtida a partir da projeção ortogonal do conjunto de dados nos eixos coordenados, a fim de criar partições do espaço original de forma semelhante a uma estrutura de árvore de decisão. Para facilitar a compreensão da abordagem proposta, iniciamos a seção apresentando uma descrição inicial da similaridade do Algoritmo TreeKDE com a abordagem de arvore de decisão, seguida de um exemplo, no espaço bidimensional. Para encerrar a seção, apresentamos o algoritmo formalmente e discutimos aspectos da sua convergência.

3.4.1 Descrição inicial do Algoritmo TreeKDE

A metodologia empregada pelo Algoritmo TreeKDE para agrupar dados multidimensionais é semelhante àquela empregada pela árvore de decisão para problemas de classificação, porém usaremos cluster como terminologia ao invés de nó. Nesse sentido, na primeira iteração do algoritmo ($k = 1$), todas as observações do problema constituem o cluster raiz ou, também chamado, cluster pai. A partição do cluster pai em dois clusters filhos é determinada por meio do minimizador da função KDE, que denominamos de variável de partição.

Definição 3.1. *Considere* $\widehat{f} : \mathbb{R} \to \mathbb{R}_+^*$ *a função KDE unidimensional e* $x^* \in \mathbb{R}$ *um ponto de mínimo da função* $\widehat{f}$*. Dizemos que* x^* *é uma variável de partição quando* $|\widehat{f}'(x^*)| < \epsilon$*, sendo* $\epsilon \in (0, 1)$ *uma precisão pré-estabelecida.*

De forma geral, a metodologia proposta pelo Algoritmo TreeKDE pode ser executada em 3 etapas:

Etapa 1) Selecionar o cluster pai;

Etapa 2) Definir a variável de partição;

Etapa 3) Obter o particionamento do cluster pai em 2 clusters filhos.

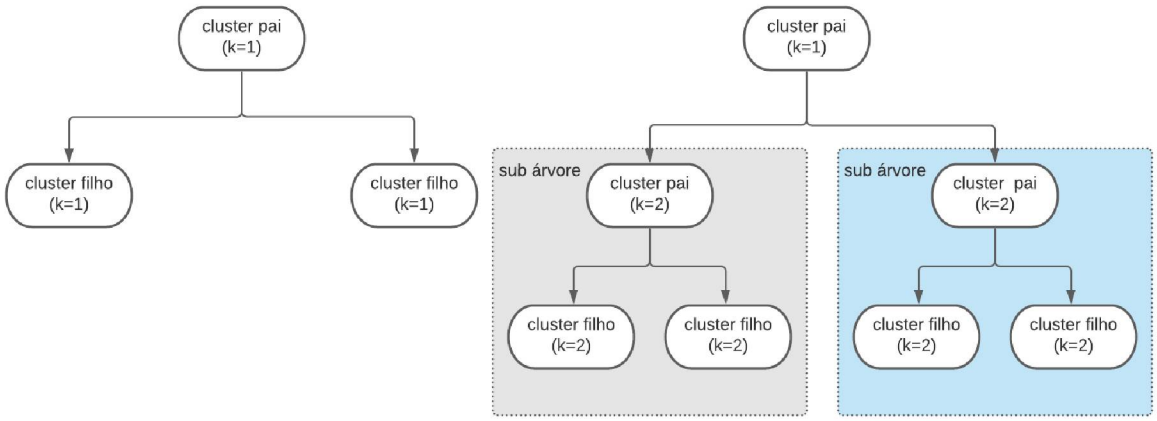
Conforme mostrado na Figura 22 (lado esquerdo), ao final da primeira iteração ($k = 1$), o algoritmo apresenta 2 clusters filhos, que serão definidos no início da segunda iteração ($k = 2$) como clusters pais. As três etapas descritas acima são repetidas recursivamente sobre cada uma das sub árvores, conforme ilustrado na Figura 22 (lado direito). O algoritmo repete o processo até atingir o critério de parada, que nesse trabalho optamos por ser a classificação de todos os grupos como clusters folha.

3.4.2 Exemplo do Algoritmo TreeKDE

Para ilustrar o funcionamento do Algoritmo TreeKDE, utilizamos os dados da Tabela 2 com a adição de uma nova observação, $(6,05; 2,45)$, conforme apresentado pelo gráfico de dispersão na Figura 23. Além dos dados, para executar o TreeKDE foram utilizados os parâmetros $\alpha = 0,75$ e $MinPts = 5$. Com o intuito de facilitar a associação do problema de clusterização com árvores de decisão, adotamos algumas terminologias muito comuns a árvore de decisão, conforme apresentado na Seção 2.3.

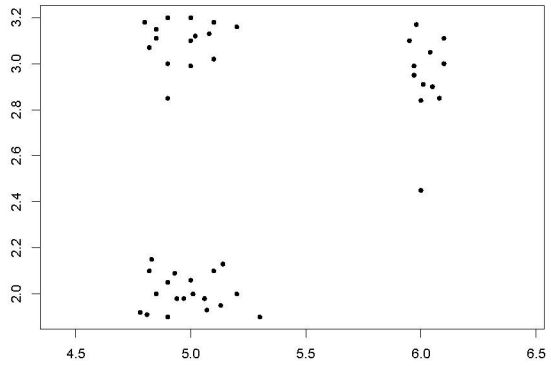
Observe que no início da primeira iteração ($k = 1$) o conjunto de dados pode ser caracterizado como um agrupamento de um único grupo, denominado cluster raiz ou cluster pai da iteração. Em seguida, todos os elementos do cluster raiz são selecionados e analisados pelo Algoritmo TreeKDE para particioná-lo em dois clusters filhos. Nesse momento, para a partição do cluster raiz, o algoritmo determina a projeção ortogonal das

Figura 22 – Associação entre o Algoritmo TreeKDE e o método da árvore de decisão



Fonte: O autor (2021).

Figura 23 – Gráfico de dispersão dos dados

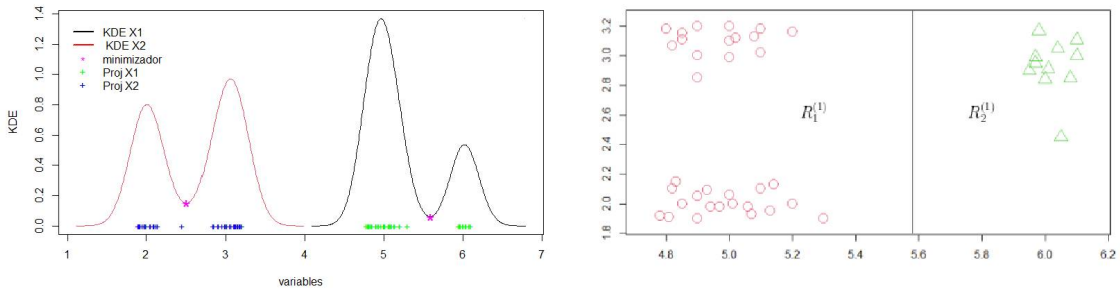


Fonte: O autor (2021).

componentes x_1 e x_2 de cada observação sobre os eixos coordenados, conforme ilustrado pelos pontos verdes e azuis, respectivamente, no quadro esquerdo da Figura 24. A partir das projeções sobre cada eixo, o Algoritmo TreeKDE determina as respectivas funções unidimensionais do KDE, que também são expressas no quadro esquerdo da Figura 24 pelas curvas nas cores preta e vermelha. Essas funções são determinadas de acordo com a expressão dada em (2.3) e o coeficiente de suavização dado por (2.19) com $\alpha = 0,75$ e $n = 1$.

Agora, a otimização de ambas as funções unidimensionais do KDE é realizada pelo algoritmo, por meio da rotina *optimise* do *Software R*, a fim de encontrar os minimizadores que são os candidatos a variável de partição do cluster pai. Obviamente, como podemos

Figura 24 – Primeira iteração do Algoritmo TreeKDE



Fonte: O autor (2021).

observar pela Figura 24, o algoritmo determinou dois minimizadores, $x_1^* = 5,59$ e $x_2^* = 2,50$, um para cada função KDE.

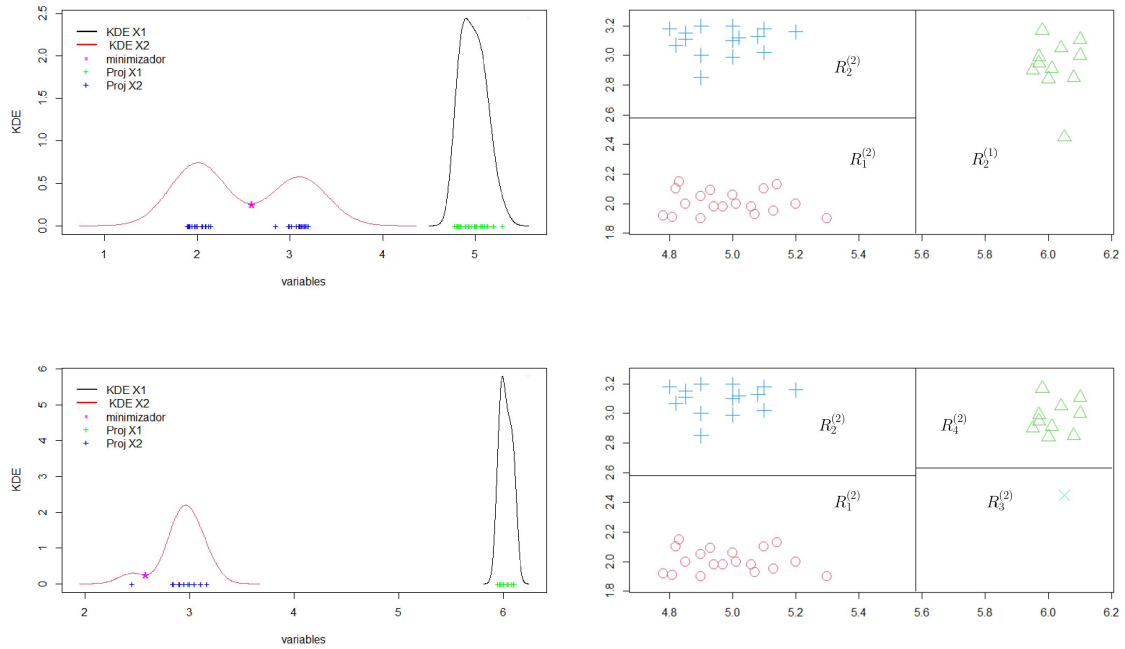
Mesmo que o algoritmo tenha encontrado dois candidatos a variável de partição, ele sempre definirá a região de partição com base no minimizador com o menor valor funcional. Nesse sentido, o algoritmo selecionou $x_1^* = 5,59$. Portando, o cluster raiz é particionado nas regiões $R_1^{(1)}(1;5,59) = \{X(i,:) \mid X(i;1) \leq 5,59 \text{ com } i = 1,\ldots,46\}$ e $R_2^{(1)}(1;5,59) = \{X(i,:) \mid X(i;1) > 5,59 \text{ com } i = 1,\ldots,46\}$, conforme mostrado no gráfico do lado direito da Figura 24, cujas regiões constituirão individualmente o cluster filho 1 e 2, respectivamente.

Ao particionar o cluster pai em dois clusters filhos, o Algoritmo TreeKDE termina sua primeira iteração. Assim, na segunda iteração, ambos os clusters filhos, anteriormente encontrados, tornam-se os clusters pais e o processo iterativo é repetido até que um critério de parada seja satisfeito. Neste trabalho, optamos por utilizar como critério de parada para o TreeKDE a classificação de todos os grupos como clusters folhas, o qual será detalhado mais adiante.

Na segunda iteração, as funções KDE obtidas pela projeção ortogonal de cada componente x_1 e x_2 são minimizadas. Para o grupo 1 ($R_1^{(1)}$), o minimizador selecionado como variável de partição é $x_2^* = 2,58$, conforme ilustrado no canto superior esquerdo da Figura 25. Assim, a partição em dois novos clusters filho é definida pelas regiões $R_1^{(2)}(2;2,58) = \{R_1^{(1)}(i,:) \mid R_1^{(1)}(i;2) \leq 2,58 \text{ com } i = 1,\ldots,35\}$ e $R_2^{(2)}(2;2,58) = \{R_1^{(1)}(i,:) \mid X(i;2) > 2,58 \text{ com } i = 1,\ldots,35\}$, conforme mostrado no canto superior direito da Figura 25.

Da mesma forma, para o grupo 2 ($R_2^{(1)}$), destacado no canto inferior esquerdo da Figura 25, o minimizador selecionado como variável de partição é $x_2^* = 2,63$, resultando na partição de dois clusters filhos, definidos pelas regiões $R_3^{(2)}(2;2,63) = \{R_2^{(1)}(i,:) \mid R_2^{(1)}(i;2) \leq 2,63 \text{ com } i = 1,\ldots,11\}$ e $R_4^{(2)}(2;2,63) = \{R_2^{(1)}(i,:) \mid X(i;2) > 2,63 \text{ com } i =$

Figura 25 – Segunda iteração do Algoritmo TreeKDE



Fonte: O autor (2021).

$1, \ldots, 11\}$, conforme mostrado no canto inferior direito da Figura 25.

Observando o cluster filho $R_3^{(2)}$, verificamos que esse possui um único elemento e, uma vez que, estabelecemos um número mínimo de 5 elementos para cada grupo ($MinPts = 5$) a partição do cluster pai, região $R_2^{(1)}$, não será considerada e, portanto, $R_2^{(1)}$ é classificado como um cluster folha.

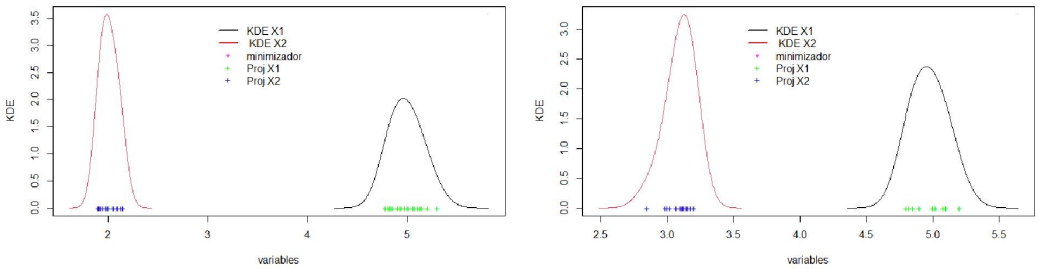
Em resumo, o Algoritmo TreeKDE determinou ao fim da segunda iteração ($k = 2$) um cluster folha ($R_2^{(1)}$) e dois clusters filhos ($R_1^{(2)}$ e $R_2^{(2)}$) provenientes do cluster pai ($R_1^{(1)}$).

O processo é repetido na próxima iteração ($k = 3$) considerando os dois clusters filhos obtidos anteriormente como os novos clusters pais. No entanto, nessa iteração nenhum dos cluster pai foi particionado devido ao fato de que o Algoritmo TreeKDE não encontrou nenhuma variável de partição, conforme mostrado na Figura 26. Sendo assim, o algoritmo é encerrado.

Nesse sentido, o Algoritmo TreeKDE agrupou o conjunto de dados em 3 grupos. A solução do agrupamento e o processo iterativo com base na árvore de decisão são ilustrados na Figura 27.

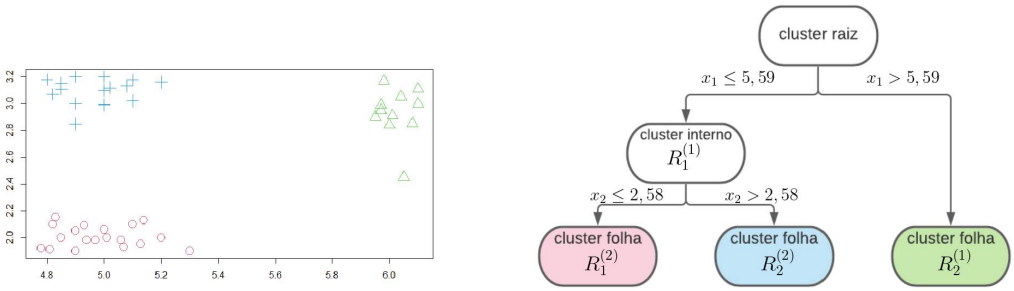
Na seção seguinte, apresentamos uma descrição formal do Algoritmo TreeKDE.

Figura 26 – Classificação dos clusters em folhas



Fonte: O autor (2021).

Figura 27 – Resultado final da clusterização proveniente do Algoritmo TreeKDE



Fonte: O autor (2021).

3.4.3 Algoritmo TreeKDE: descrição geral

Para o Algoritmo TreeKDE, a ser apresentado a seguir, são necessários: uma matriz $X \in \mathbb{R}^{m \times n}$ com os dados do problema, sendo m o número de observações e n o número de atributos; um parâmetro $\alpha \in \mathbb{R}_+^*$, responsável pela suavidade da função KDE calculado conforme (2.19) para determinar h e, por fim, um número inteiro $MinPts \in \mathbb{Z}_+^*$ que estabelece um número mínimo de observações para cada grupo, sendo $2 \leq MinPts \leq \frac{m}{2}$.

A cada nova iteração (linhas 2 a 35), o Algoritmo TreeKDE seleciona os clusters pais ainda não classificados como clusters folha (linhas 3 a 33) e, então, o processo de partição é iniciado.

Para cada cluster pai o algoritmo determina: as projeções ortogonais das componentes nos eixos coordenados, uma função KDE unidimensional ($\widehat{f}$) para cada projeção e o respectivo minimizador (x^*) para cada uma das funções KDE (linhas 7 a 16). Note que x^* é determinado, em cada caso, a partir da minimização de $\widehat{f}$ sujeita a restrição $l \leq x \leq u$, onde l e u são os limites inferior e superior das projeções, respectivamente.

O menor minimizador que satisfaça a condição $|\widehat{f}'(x_i^*)| < \epsilon$ $(i = 1, \ldots, n)$ é

Algoritmo 5: TreeKDE

Entrada: $X \in \mathbb{R}^{m\times n}$,$\alpha \in \mathbb{R}_+^*$, $MinPts \in \mathbb{Z}_+^*$

Faça: $t = 0$, $k = 1$, $k_f = 0$, $C_f = \emptyset$, $X = X \cup ones(1 : m)$, $\epsilon \in (0, 1)$;
Enquanto $k_f < k$ **faça**
  Para $j = 1 : k$ **faça**
    Se $j \notin C_f$ **então**
      Selecione $\overline{X} \in \mathbb{R}^{m_j\times n} \subset X \in \mathbb{R}^{m\times n} \mid X(:, n + 1) = j$;
      $fmin = \infty$; $var = 0$; $g = k$;
      Para $i = 1 : n$ **faça**
        $v = \overline{X}(:, i)$, $l = min(v)$, $u = max(v)$, $\sigma = std(v)$;
        Determine: h e $\widehat{f}$ de acordo com (2.19) e (2.3);
        $x^* \in \underset{x\in\mathbb{R}}{\operatorname{argmin}} \left\{\widehat{f}(x) : l \leq x \leq u\right\}$;
        Se $|\widehat{f}'(x_i^*)| < \epsilon$ *e* $\widehat{f}(x_i^*) < fmin$ **então**
          $var = i$;
          $x_p^* = x_i^*$;
          $fmin = \widehat{f}(x_i^*)$;
        Fim
      Fim
      Se $var = 0$ **então**
        $C_f = C_f \cup \{j\}$;
        $k_f = k_f + 1$
      Senão
        $X_{aux} = X$;
        $X(:, n + 1) = \{g + 1 \mid \overline{X}(:, var) > x_p^*\}$;
        $m_{g+1} = \#(X(:, n + 1) = g + 1)$;
        Se $min\{m_{g+1}, m_j - m_{g+1}\} \geq MinPts$ **então**
          $g = g + 1$
        Senão
          $X = X_{aux}$;
          $C_f = C_f \cup \{j\}$;
          $k_f = k_f + 1$
        Fim
      Fim
    Fim
  Fim
  $k = g$; $t = t + 1$
Fim
$cluster = X(:, n + 1)$;
Retorna $cluster$

selecionado pelo algoritmo como variável de partição (linhas 11 a 15). Se o algoritmo não determinar nenhuma variável de partição, o cluster pai é classificado como um cluster folha (linhas 17 a 19), caso contrário, o cluster pai é particionado em dois clusters filhos (linhas 20 a 31).

Uma vez que a partição foi realizada, o algoritmo avalia se os clusters filhos obtidos têm um número mínimo de elementos ($MinPts$). Em caso afirmativo, a partição é realizada (linhas 24 e 25), caso contrário, a partição é descartada e o cluster pai classificado como um cluster folha (linhas 26 a 30). O processo é repetido até que todos os grupos sejam classificados como clusters folhas e, por fim, o algoritmo apresenta a solução com a clusterização do conjunto de dados (linha 36).

3.4.4 Convergência do Algoritmo TreeKDE

No próximo teorema, mostramos que o Algoritmo TreeKDE é bem definido e converge em um número finito de iterações.

Teorema 3.4. *O Algoritmo TreeKDE é bem definido e executa no máximo* $t = \left\lceil log_2 \left(\frac{m}{MinPts}\right)\right\rceil$ *passos.*

Demonstração. A cada iteração o Algoritmo TreeKDE precisa determinar um minimizador x^* da função unidimensional, obtido a partir das projeções ortogonais das componentes nos eixos coordenados.

Visto que a função unidimensional $\widehat{f}$ é contínua, suave com derivados de todas as ordens e definida em um intervalo fechado, pelo Teorema de Weierstrass, existe um minimizador x^* em cada iteração, o que implica que o Algoritmo TreeKDE está bem definido.

Agora, mostraremos que o algoritmo determina a solução em um número finito de passos. Devido à partição ser binária de cada cluster pai em dois clusters filhos, o número de grupos, k, determinado em cada passo do Algoritmo TreeKDE é de no máximo 2^t. Dessa forma, seja $m \in \mathbb{Z}_+^*$ o número de observações e $MinPts \in \mathbb{Z}_+^*$, o número mínimo de observações em cada grupo, considerando $2 \leq MinPts \leq \frac{m}{2}$, então o número de grupos é limitado em $1 \leq nc \leq \frac{m}{MinPts} \leq \frac{m}{2}$. Segue que

$$k \leq \frac{m}{MinPts} \Rightarrow 2^t \leq \frac{m}{MinPts}.$$

Daí segue que

$$t \leq log_2 \left(\frac{m}{MinPts}\right).$$

Como $t \in \mathbb{Z}_+^*$, temos

$$t \leq \left\lceil \log_2 \left(\frac{m}{MinPts}\right)\right\rceil.$$

Concluímos então que o Algoritmo TreeKDE é bem definido e para em um número finito de passos de no máximo $\left\lceil \log_2 \left(\frac{m}{MinPts} \right) \right\rceil$. □

4 EXPERIMENTOS NUMÉRICOS

Neste capítulo, apresentamos resultados que evidenciam e qualificam os algoritmos MulticlusterKDE, AdditiveclusterKDE e TreeKDE como métodos de clusterização de dados multidimensionais e, também, o índice CD como métrica de validação interna.

Além dos três algoritmos propostos nesse trabalho, submetemos os conjuntos de dados a 8 algoritmos amplamente difundidos na literatura para resolução de problemas de clusterização. O principal motivo pela escolha desses algoritmos foi o fato de estarem disponíveis na biblioteca do *Software R* (R Core Team (2019)). Na Tabela 3, apresentamos a relação dos algoritmos, o pacote necessário (ou seção de referência) e os parâmetros de entrada em cada caso.

Tabela 3 – Algoritmos utilizados para comparação

Algoritmos	pacote do *Software R*	parâmetros
MulticlusterKDE	Seção 3.2	α
AdditiveclusterKDE	Seção 3.3	α, A, m_A
TreeKDE	Seção 3.4	α, $MinPts$
K-means	*stats*(R Core Team (2019))	k
K-medoids	*cluster*(Maechler et al. (2019))	k
CLARA	*cluster*(Maechler et al. (2019))	k
GMM	*mclust*(Scrucca et al. (2016))	k
DIANA	*cluster*(Maechler et al. (2019))	k
CLINK	*stats*(R Core Team (2019))	k
PdfCluster	*pdfcluster*(Azzalini, Menardi et al. (2014))	*hmult*, *bwtype*
DBSCAN	*dbscan*(Hahsler, Piekenbrock e Doran (2019))	*eps*, $MinPts$

Fonte: O autor (2021).

Vale ressaltar que os parâmetros utilizados nos experimentos numéricos apresentados neste capítulo não foram submetidos a nenhum tipo de otimização e sim escolhidos de forma a melhor se adaptar aos dados dos problemas.

Para a execução dos testes numéricos utilizamos um notebook com processador Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz com sistema operacional Windows 10 home 64 bits. A versão do *Software R* utilizada foi a 3.6.1 (2019-07-05) executado via Rstudio Version 1.1.463.

Para uma melhor exposição dos resultados, esse capítulo foi subdividido em 4 seções. Inicialmente, analisamos a qualidade do índice CD como métrica de validação interna para clusterização. Na sequência, mostramos os resultados da clusterização de 8 problemas bidimensionais, com enfoque na visualização gráfica das soluções e nas características dos algoritmos. Em seguida, abordamos 6 problemas multidimensionais ($n > 2$)

e, por fim, exploramos alguns problemas de classificação resolvidos por algoritmos de clusterização.

4.1 ANÁLISE DE DESEMPENHO DO ÍNDICE CD

Para realizar a verificação do desempenho do índice CD como métrica de validação interna, optamos por utilizar como base para os testes o Algoritmo CLARA, uma vez que esse necessita apenas da informação do número de grupos como parâmetro de entrada e possui um tempo de execução consideravelmente baixo. Cabe observar que outros algoritmos, tais como: K-means, K-medoids e GMM, também poderiam ser utilizados para essa verificação.

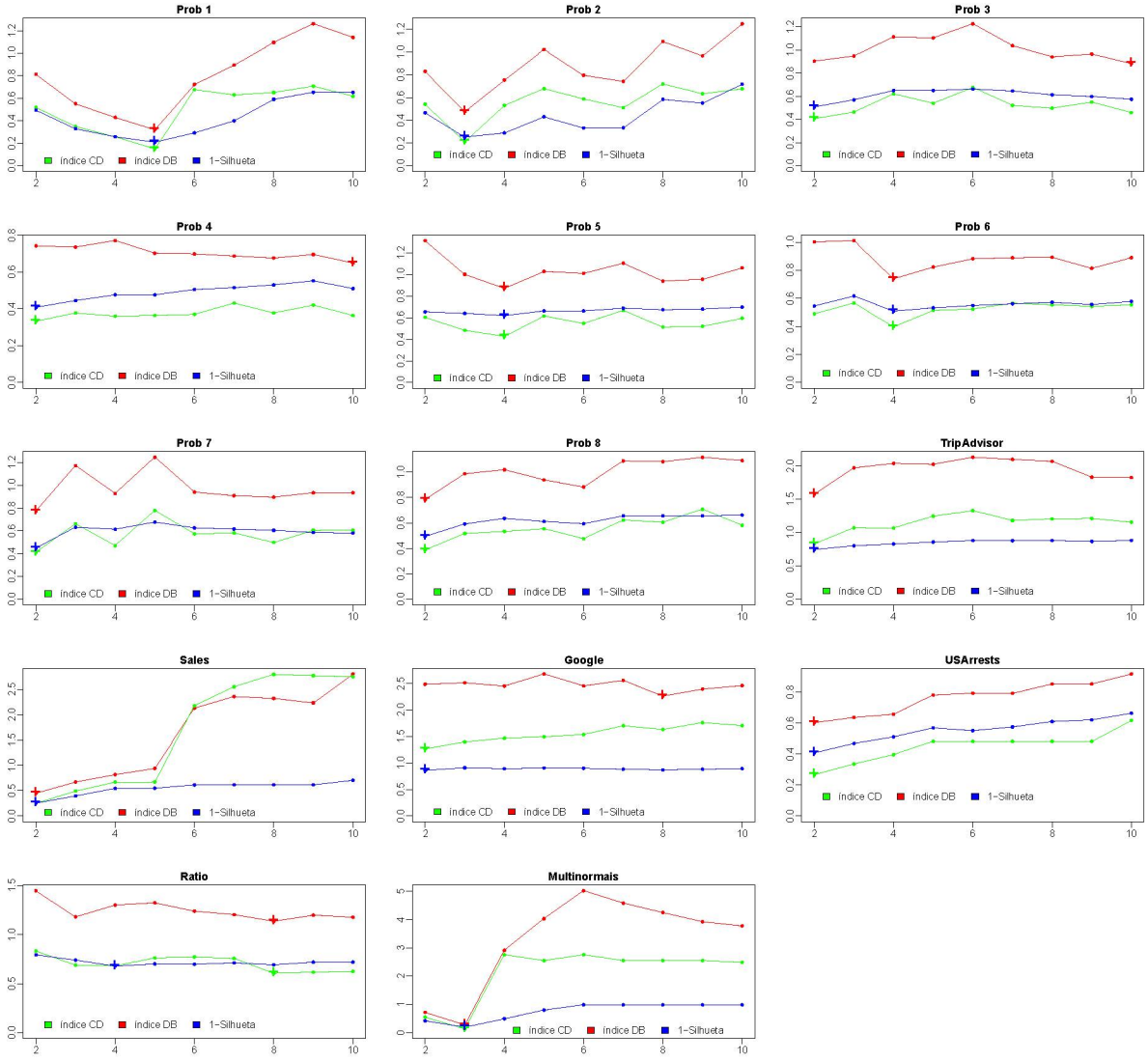
Para analisar o desempenho do índice CD, executamos 9 configurações de agrupamentos, ou seja, consideramos $k = 2, \ldots, 10$, sendo k o número de grupos, em 14 problemas diferentes, totalizando 126 execuções. A partir dos agrupamentos quantificamos os agrupamentos por meio de 3 métricas de validação interna: o índice DB, proposto por Davies e Bouldin (1979); o coeficiente de silhueta, proposto por Rousseeuw (1987) e o índice CD, contribuição desse trabalho e apresentado na Seção 3.1.

A Figura 28 ilustra o desempenho das 3 métricas de validação para os problemas testados. Em cada um dos gráficos apresentados o eixo horizontal expressa o número de grupos, enquanto o eixo vertical expressa os valores das métricas de validação. Como discutido, quanto menor for o valor do índice CD (Seção 3.1) e do índice DB (Seção 2.1.1), melhor é a qualidade da clusterização. Por outro lado, o coeficiente de silhueta avalia de forma diferente e, assim, quanto mais próximo de 1, melhor é o agrupamento (Seção 2.1.1). Devido ao comportamento oposto do coeficiente de silhueta, nos gráficos apresentados na Figura 28, plotamos o resultado de $1 - \textit{silhueta}$, com o intuito de uniformizar a apresentação dos resultados. Ainda sobre a Figura 28, os pontos representados pelo simbolo "+", destacam em qual configuração (número de grupos) as métricas obtiveram o menor valor, sendo tal clusterização indicada como ótima sob a ótica de cada uma das métricas.

Os 14 conjuntos de dados utilizados não possuem nenhuma informação externa sobre o número de grupos considerados como ideal, se enquadrando no tipo de situação em que as métricas de validação interna são utilizadas. Optamos, nessa seção, em omitir maiores informações sobre cada um dos problemas utilizados, visto que os mesmos serão reutilizados nas seções seguintes juntamente com o detalhamento de cada caso.

Para uma análise detalhada, as clusterizações consideradas ótimas são reapresentadas na Tabela 4 a seguir, juntamente com o tempo total gasto pelas métricas para realizar as avaliações nas 9 diferentes configurações de agrupamentos ($k = 2, \ldots, 10$), executados pelo Algoritmo CLARA para cada problema. Cabe salientar que o tempo apresentado não contempla o tempo de execução do Algoritmo CLARA no processo de

Figura 28 – Avaliação das 3 métricas para os diferentes problemas com variação do número de grupos na clusterização



Fonte: O autor (2021).

clusterização, apenas de avaliação pelas métricas.

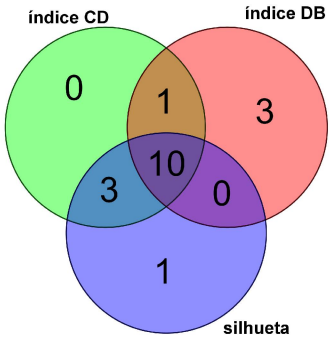
Analisando primeiramente as clusterizações indicadas como ótimas pelas 3 métricas, encontramos uma correspondência mútua em 10 dos 14 problemas testados. Outro fato interessante é que não houve divergência entre as 3 métricas, sempre pelo menos duas delas apresentaram o mesmo resultado para o número de grupos. De forma mais específica, o índice CD não divergiu simultaneamente das duas outras métricas em nenhum dos problemas analisados. Na Figura 29 ilustramos tais correspondências por meio do diagrama de Venn. Note que há uma correspondência mútua entre as 3 métricas em 71,4% dos problemas. Comparando as métricas duas a duas, os resultados do índice CD coincidiram em 85,7% dos casos com o coeficiente de silhueta e em 78,6% com o índice DB.

Tabela 4 – Resultado da clusterização avaliada pelas métricas

Problemas	Número de grupos			Tempo(s)		
	índice CD	índice DB	silhueta	CD	DB	silhueta
Prob 1	5	5	5	0,082	0,018	0,764
Prob 2	3	3	3	0,071	0,032	0,267
Prob 3	2	10	2	0,056	0,031	0,677
Prob 4	2	10	2	0,101	0,042	0,778
Prob 5	4	4	4	0,097	0,02	1,793
Prob 6	4	4	4	0,121	0,451	17,997
Prob 7	2	2	2	0,066	0,034	1,658
Prob 8	2	2	2	0,288	0,276	184,755
TripAdvisor	2	2	2	0,073	0,014	1,676
Sales	2	2	2	0,136	0,108	2,306
Google	2	8	2	0,295	0,259	63,282
USArrests	2	2	2	0,053	0,022	0,031
Ratio	8	8	4	0,062	0,001	0,031
Multinormais	3	3	3	0,277	0,162	0,116

Fonte: O autor (2021).

Figura 29 – Diagrama de Venn para similaridade do número de grupos considerados como ótimos



Fonte: O autor (2021).

A avaliação do desempenho do índice CD por meio, exclusivamente, do percentual de correspondência é simples e de fácil visualização, contudo pode gerar questionamentos da real confiabilidade da qualidade da métrica em avaliar a clusterização multidimensional. Nesse sentido, para garantir uma análise consistente, usamos dois testes de concordância, o primeiro, denominado Kappa (κ), proposto por Cohen (1960) e executado por meio da rotina "Kappa2" disponível no *Software R* pelo pacote "irr"(Gamer, Lemon e Puspendra (2019)) e o segundo, proposto por Fleiss (1971), denominado Fleiss-κ, calculado pela rotina "kappam.fleiss", também disponível no pacote "irr" do *Software R*.

O coeficiente Kappa avalia a concordância entre dois avaliadores, neste caso, entre os índices DB e CD, entre o índice CD e o coeficiente de silhueta e, por último, entre

o índice DB e o coeficiente de silhueta. Enquanto o coeficiente de Fleiss-κ, avalia a concordância entre n avaliadores, nesse caso os 3 avaliadores, índices DB, CD e o coeficiente de silhueta.

Considerando o número ótimo de grupos para cada uma das 3 métricas sobre os 14 problemas analisados, conforme Tabela 4, aplicamos o teste de concordância de Kappa e Fleiss-κ e os resultados são apresentados na Tabela 5.

Tabela 5 – Resultados dos testes de concordância

	CD e DB	CD e silhueta	DB e silhueta	Todas as métricas
κ	0,71	0,884	0,614	0,724
Concordância	substancial	quase perfeita	substancial	substancial
p-valor	2,3e-08	4,01e-08	1,03e-06	0

Fonte: O autor (2021).

De acordo com Landis e Koch (1977), resultados para κ e Fleiss-κ entre 0,61 e 0,80 indicam que existe uma concordância substancial entre os avaliadores, enquanto valores superiores a 0,81 indicam uma concordância quase perfeita. Além disso, o p-valor inferior a 0,001 revela que a concordância entre as métricas não é puramente aleatória. Analisando a Tabela 5, percebemos que os valores para κ, foram todos superiores a 0,61, com destaque para a correspondência de 0,884 entre o índice CD e o coeficiente de silhueta, indicando uma correspondência quase perfeita. O coeficiente de Fleiss-κ também apresentou uma concordância substancial entre as 3 métricas analisadas com um valor de 0,724. Por fim, todas os testes de Kappa e Fleiss-κ, apresentaram um p-valor $< 0,001$, com isso rejeitamos a hipótese de que a concordância entre as métricas é puramente aleatória, caracterizando-as como concordantes.

Analisando agora o tempo de execução utilizado pelas 3 métricas para avaliar os 9 diferentes agrupamentos para cada um dos 14 problemas expostos na Tabela 4, ou seja, para as 126 avaliações, temos que o índice CD precisou de um total de 1,778 segundos, o índice DB levou 1,470 segundos, enquanto o coeficiente de silhueta precisou de 276,131 segundos para avaliar todas clusterizações. Observamos uma proximidade de valores do tempo gasto pelos índices CD e DB, com uma leve vantagem para o índice DB. Por outro lado, o coeficiente de silhueta apresentou tempo superior às outras duas, indicando a inviabilidade da sua utilização para problemas de dimensões mais elevadas.

Com base nos resultados expostos é possível garantir que o índice CD, proposto nesse trabalho, é eficiente para medir a qualidade de agrupamentos quando comparado às métricas já consagradas na literatura tais como o índice DB (Davies e Bouldin (1979)) e o coeficiente de Silhueta (Rousseeuw (1987)). O índice CD mostrou uma correspondência substancial com o índice DB com tempo de execução similar, enquanto que com o coeficiente de silhueta sua correspondência foi quase perfeita com um tempo de execução muito

menor, evidenciando seu uso como métrica de validação interna para clusterização.

4.2 COMPARAÇÃO ENTRE ALGORITMOS PARA DIFERENTES ESTRUTURAS DE DADOS BIDIMENSIONAIS

Elaboramos essa seção com o intuito de apresentar algumas características dos algoritmos de clusterização utilizados nesse trabalho, aplicados a conjuntos de dados bidimensionais com diferentes estruturas de agrupamentos. Embora os problemas bidimensionais forneçam algumas particularidades sobre os algoritmos, essas podem não se aplicar em problemas de dimensões mais altas.

Com o objetivo de facilitar a comparação entre os algoritmos, organizamos os resultados na forma de uma matriz de clusterização, apresentada pela Figura 30. Cada uma das linhas da figura corresponde aos algoritmos de clusterização utilizados nos experimentos numéricos desse trabalho (Tabela 3), enquanto as colunas correspondem a 8 problemas bidimensionais, nomeados de "Prob 1" a "Prob 8". Os 6 primeiros problemas foram gerados aleatoriamente, enquanto os 2 últimos são provenientes de pacotes do *Software R*. Cabe ressaltar que as clusterizações apresentadas na Figura 30 não foram submetidas a nenhum tipo de validação, ficando a clusterização focada na visualização gráfica que melhor se adaptou aos dados.

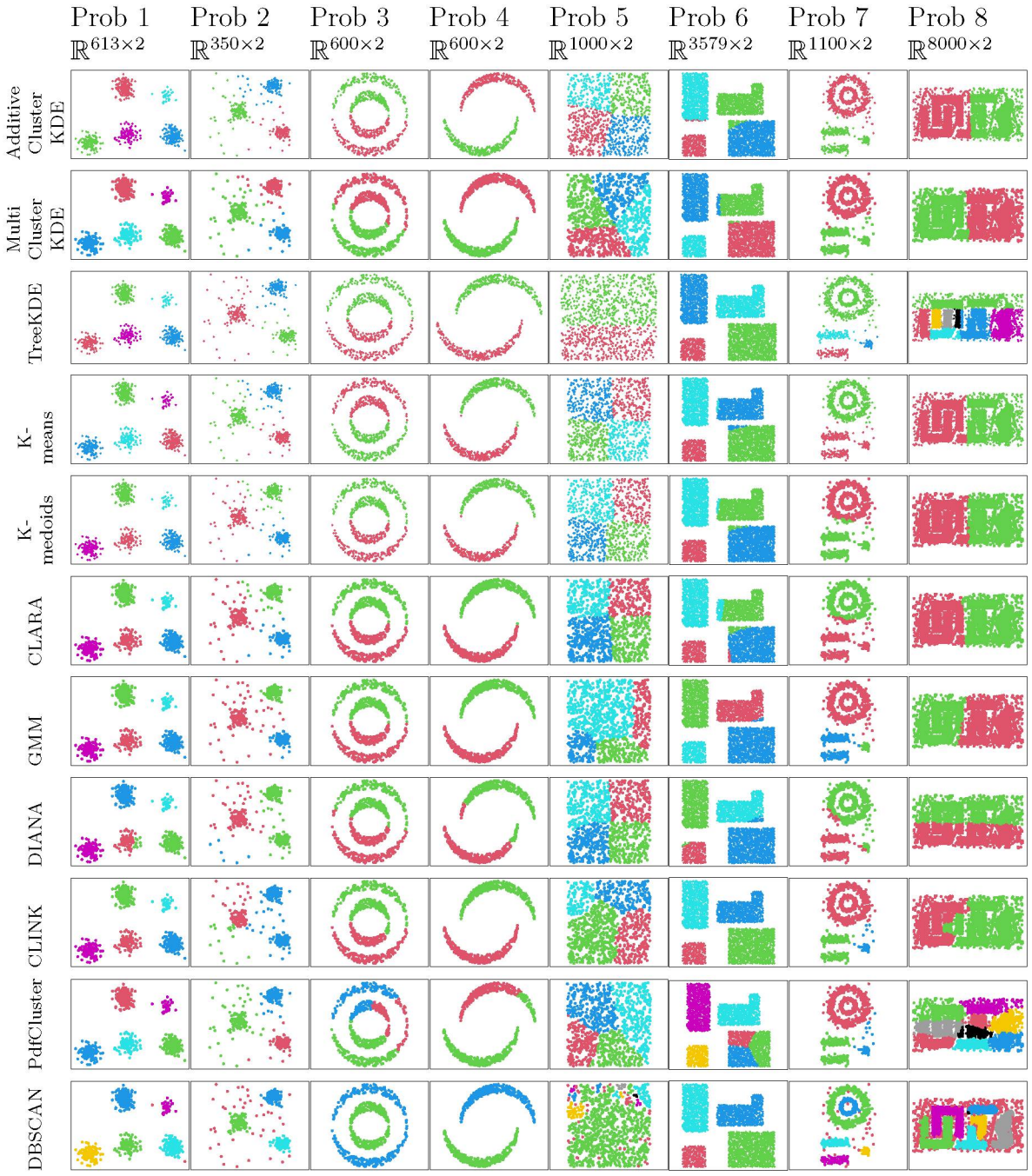
Por meio da Figura 30, observamos que os algoritmos AdditiveclusterKDE, MulticlusterKDE, K-means, K-medoids e CLARA possuem a característica de formar grupos esféricos, centrados em centroides ou medoids e com designação das observações pelo critério de mínima distância. Dentre esses o K-means, K-medoids e o CLARA necessitam, a priori, do número de grupos, enquanto os algoritmos AdditiveclusterKDE e MulticlusterKDE não possuem essa exigência.

Os algoritmos GMM e PdfCluster por sua vez, têm grupos mais direcionados ao formato de elipsoides definidos por distribuições normais de probabilidade provenientes de subconjuntos de pontos. O GMM exige a informação do número de grupos enquanto o PdfCluster não exige. No entanto, o Algoritmo PdfCluster, tem uma tendência em determinar um número elevado de grupos, como por exemplo no "Prob 8".

De forma geral, os algoritmos AdditiveclusterKDE, MulticlusterKDE, K-means, K-medoids, CLARA, GMM e PdfCluster são mais indicados na clusterização de problemas com grupos globulares, como por exemplo, os problemas: "Prob 1" e "Prob 2".

O Algoritmo TreeKDE por sua vez realiza o particionamento do espaço de recurso do problema em regiões retangulares (caso o problema seja de dimensão maior que 2 em regiões hiper-retangulares). Ele não exige a informação do número de grupos e também não utiliza matriz de distâncias para o processo de designação das observações aos grupos. Tal algoritmo é indicado, principalmente, para problemas cujos grupos podem ser inscritos

Figura 30 – Comparação do comportamento de diferentes algoritmos para estruturas de dados variadas



Fonte: O autor (2021).

em regiões retangulares, linearmente separáveis por retas paralelas aos eixos coordenados. Os problemas "Prob 1", "Prob 4" e "Prob 6" são bons exemplos de sua aplicabilidade, especialmente o "Prob 6".

Os algoritmos DIANA e CLINK não apresentam uma estrutura visual definida de seus agrupamentos, visto suas dependências com a matriz de dissimilaridade. Tais algoritmos podem formar grupos com baixa concentração de observações mesmo que estas

sejam claramente pertencentes a grupos com maior densidade, como pode ser observado no "Prob 1" e "Prob 2".

Por último, destacamos o Algoritmo DBSCAN, o qual constrói grupos arbitrários baseados na proximidade e quantidade de vizinhos, isto é, a partir de um ponto qualquer enquanto existir um determinado número de observações em torno desse ponto, segundo um raio pré-estabelecido, o algoritmo inclui pontos vizinhos ao grupo, conforme é possível observar nos problemas "Prob 3", "Prob 7" e "Prob 8" da Figura 30. Outro ponto sobre o DBSCAN é o fato deste não agrupar todas as observações do problema, podendo classificar algumas observações como ruídos (*outliers*). Esses ruídos estão representados pelos pontos em vermelho nos gráficos na linha do Algoritmo DBSCAN. Assim, dependendo do problema e dos parâmetros de entrada, a quantidade de ruídos pode representar uma parte considerável do número total de observações do problema, por exemplo, "Prob 8".

Podemos ver que a maioria dos algoritmos utilizados nos experimentos, possuem a característica de formar grupos globulares e necessitam, a priori, da informação do número de grupos. Dentre os algoritmos com grupos globulares, os algoritmos AdditiveclusterKDE, MulticlusterKDE e PdfCluster tem a vantagem de conseguirem identificar de forma autônoma o número de grupos dos problemas.

A formação de grupos globulares, em geral, é proveniente do uso de distância Euclidiana para designação de observações aos centroides (ou medoids). Os algoritmos TreeKDE e DBSCAN são exceções a essa estrutura. O Algoritmo TreeKDE consegue detectar grupos arbitrários, desde que estes estejam inscritos em partições retangulares do espaço de recurso. O DBSCAN, apesar de também utilizar distância em sua metodologia, possui a característica de detectar automaticamente o número de grupos e com formatos arbitrários.

Dentre os algoritmos analisados, os algoritmos TreeKDE, AdditiveclusterKDE, MulticlusterKDE e PdfCluster utilizam o KDE como ferramenta no processo de clusterização. O Algoritmo TreeKDE trabalha com o KDE unidimensional, enquanto os outros três com a KDE multidimensional. A vantagem do uso do KDE unidimensional é o baixo tempo de execução da clusterização em relação ao tempo gasto pela abordagem multidimensional, cuja relação será evidenciada mais detalhadamente na Seção 4.4.

Nesse contexto, o três algoritmos propostos neste trabalho, além de utilizarem o KDE para construção da função de densidade, aplicam também a múltipla otimização dessa função no processo de clusterização, porém com objetivos distintos. Nos algoritmos MulticlusterKDE e AdditiveclusterKDE a otimização tem o foco na obtenção de pontos com alta densidade de observações em sua vizinhança, que serão caracterizados como centroides dos grupos, em outras palavras, o objetivo é obter pontos de máximo da função KDE multidimensional. Por outro lado, o Algoritmo TreeKDE busca pontos de mínimos

das funções KDEs unidimensionais (variáveis de partição), que identificam baixas concentrações de observações para realizar a separação linear do conjunto de dados. Apesar dos diferentes objetivos, matematicamente não há diferença entre obter pontos de máximo ou de mínimo da função, ficando a grande diferença entre os algoritmos marcada pelo fato das KDE serem multidimensional e unidimensional.

Conforme o exposto, podemos observar que cada algoritmo tem sua particularidade, sendo indicado para um determinado conjunto de problemas e situações.

4.3 TESTES NUMÉRICOS COM PROBLEMAS DE CLUSTERIZAÇÃO

Nesta seção, mostramos os resultados da clusterização de 6 problemas multidimensionais ($n > 2$), em que o número de grupos é desconhecido, sendo submetidos a 11 algoritmos diferentes (Tabela 3) e avaliados pelas métricas de validação interna: índice CD, índice DB e coeficiente de silhueta.

Para execução dos experimentos, algumas diretrizes foram adotadas para escolha dos parâmetros de entrada. No Algoritmo AdditiveclusterKDE fixamos o número de amostras em 5 ($A = 5$) e o tamanho de cada uma das amostras utilizadas ($m_A \in \mathbb{Z}_+^*$) foi estimado por meio de frações da quantidade total de observações de cada conjunto (m), da seguinte forma,

$$m_A \cong \begin{cases} 0,4 \times m, & \text{se} \quad m < 100 \\ 0,2 \times m, & \text{se} \quad 100 \leq m < 500 \\ 0,1 \times m, & \text{se} \quad 500 \leq m < 1000 \\ 0,05 \times m, & \text{se} \quad 1000 \leq m < 5000 \\ 0,025 \times m, & \text{se} \quad m \geq 5000. \end{cases}$$

Por último, o parâmetro α foi ajustado a cada problema com o objetivo de obter a melhor clusterização.

Os algoritmos MulticlusterKDE, TreeKDE e PdfCluster tiveram seus parâmetros ajustados em testes com o intuito de se obter clusterizações que propiciassem os melhores resultados segundo as métricas de validação. Nos algoritmos K-means, K-medoids, CLARA, GMM, DIANA e CLINK foram executadas clusterizações com variações no número de grupos de 2 até 20 ($k = 2, \ldots, 20$) com seleção da clusterização com melhores resultados de acordo com as métricas de validação.

O Algoritmo DBSCAN possui dois parâmetros de entrada obrigatórios, o $minPts$ e o eps. O $minPts$ estabelece o número mínimo de pontos que o Algoritmo DBSCAN precisa considerar para continuar com o processo de alocação de elementos em um mesmo grupo. Para esse parâmetro, optamos por usar o valor fixo $minPts = 5$, valor esse considerado como padrão no *Software R*. Para definir o outro parâmetro, executamos uma

varredura de 100 diferentes valores para *eps*, igualmente espaçados entre a mínima e a máxima distância Euclidiana entre os elementos do conjunto analisado.

Ainda sobre o Algoritmo DBSCAN, este tem a característica de considerar pontos de baixa densidade como ruídos, os quais não são incluídos na clusterização. Como a exclusão de elementos do conjunto afetam diretamente a validação pelas métricas internas, optamos nesse trabalho por apresentar a clusterização do Algoritmo DBSCAN com a menor quantidade de ruídos[1] concomitante com os melhores resultados das métricas de validação.

Por fim, os resultados dos experimentos expostos nesta seção são apresentados em tabelas as quais possuem as informações do número de grupo (k) e os valores das 3 métricas de validação interna já mencionadas, o tempo de execução dos algoritmos no processo de clusterização e os parâmetros de entrada utilizados em cada um dos algoritmos, para os respectivos problemas. Cabe observar que o tempo de execução apresentado é apenas referente ao processo de clusterização que proporcionou o melhor resultado, ou seja, o tempo apresentado se refere a uma única execução dos algoritmos, não sendo computado o tempo gasto para executar todas as outras clusterizações e também o cálculo das métricas de validação.

- ***TripAdvisor*** **($X \in \mathbb{R}^{980\times 10}$)**

O primeiro problema analisado é o proveniente do banco de dados *TripAdvisor*. O conjunto consiste em avaliações de usuários ao *TripAdvisor.com*, sobre 10 categorias de atrações da Ásia oriental: galerias de arte, boates, lanchonetes (*juice bars*), restaurantes, museus, resorts, parques, praias, cinemas e instituições religiosas. Cada usuário avalia todas as atrações visitadas como Excelente (4), Muito boa (3), Média (2), Ruim (1) ou Terrível (0). A classificação média do usuário sobre cada tipo de atração é atribuída ao "*Travel Reviews Data Set*", disponibilizado por Asuncion e Newman (2007), sendo este constituído por 980 registros numéricos. Os resultados da clusterização desse conjunto estão dispostos na Tabela 6.

De acordo com a Tabela 6, observamos que a maioria dos algoritmos estabeleceram que o melhor agrupamento é o constituído por 2 grupos. O Algoritmo PdfCluster foi o que apresentou solução com a maior quantidade de grupos, 17 grupos. Para os demais algoritmos, apesar do número de grupos ser o mesmo, o resultados para as métricas de validação apresentaram valores distintos, sendo que os algoritmos AdditiveclusterKDE e MulticlusterKDE obtiveram os melhores resultados nas 3 métricas, seguidos pelos algoritmos DBSCAN, TreeKDE e DIANA.

[1] Em cada um dos problemas, a informação do número de elementos considerados como ruídos (*outliers*) é expresso como uma nota em cada tabela.

Tabela 6 – Resultado dos algoritmos aplicados ao conjunto *TripAdvisor*

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	2	0,46	0,91	0,44	10,3345	$\alpha = 3$, $A = 5$, $m_A = 98$
MulticlusterKDE	2	0,46	0,91	0,44	8,5465	$\alpha = 0,01$
TreeKDE	2	0,70	1,45	0,32	0,0378	$\alpha = 2$; $MinPts = 4$
K-means	2	0,71	1,43	0,30	0,0060	$k = 2$
K-medoids	2	0,75	1,48	0,28	0,8241	$k = 2$
CLARA	2	0,83	1,57	0,26	0,0030	$k = 2$
GMM	5	1,63	2,47	0,10	1,8718	$k = 5$
DIANA	2	0,70	1,45	0,32	2,9980	$k = 2$
CLINK	2	0,71	1,51	0,28	0,0469	$k = 2$
PdfCluster	17	1,41	2,02	0,04	359,7566	-
DBSCAN[1]	2	0,60	0,94	0,29	0,0312	$eps = 0,89$; $MinPts = 5$

[1]ruído=41

Fonte: O autor (2021).

Em relação ao tempo de execução, os algoritmos CLARA e K-means foram os que apresentaram os menores tempos computacionais, seguidos pelos algoritmos TreeKDE, DBSCAN e CLINK. Os algoritmos PdfCluster, AdditiveclusterKDE e MulticlusterKDE foram os que utilizaram maior tempo computacional para obter a solução, chegando a ultrapassar 359 segundos (Algoritmo PdfCluster). Cabe observar que estes 3 algoritmos utilizam o estimador de densidade kernel multivariado em suas metodologias o que exige um elevado custo de processamento devido a complexidade de operações.

- ***Sales*** $(X \in \mathbb{R}^{811\times52})$

O segundo problema analisado é proveniente do banco de dados *Sales Transactions Dataset Weekly*, denominado "*sales*", o qual é constituído pela venda de 811 produtos durante 52 semanas, utilizado no trabalho de Tan e Lau (2014) e disponibilizado por Asuncion e Newman (2007). Os resultados da clusterização desse conjunto estão dispostos na Tabela 7.

Novamente a maioria dos algoritmos indicou que a melhor clusterização é com 2 grupos, com exceção dos algoritmos PdfCluster e GMM. A melhor clusterização, de acordo com as métricas de validação, foi proveniente do Algoritmo DBSCAN, no entanto, com 3 elementos excluídos da avaliação. Dentre os algoritmos que agruparam 100% dos dados, os melhores resultados foram alcançados pelos algoritmos AdditiveclusterKDE, MulticlusterKDE e CLINK. De modo geral, todos os algoritmos obtiveram resultados similares com exceção do GMM e PdfCluster, sendo que este último não conseguiu determinar mais que um único grupo não sendo possível utilizar métricas de validação.

Quanto ao tempo de execução, de forma semelhante ao problema anterior, os algoritmos K-means e CLARA apresentaram os melhores desempenhos, enquanto PdfCluster,

Tabela 7 – Resultado dos algoritmos aplicados ao conjunto *Sales*

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	2	0,23	0,43	0,75	6,7140	$\alpha = 1$; $A = 5$; $m_A = 81$
MulticlusterKDE	2	0,23	0,43	0,75	35,8364	$\alpha = 0,1$
TreeKDE	2	0,23	0,50	0,69	0,2694	$\alpha = 2$; $MinPts = 50$
K-means	2	0,29	0,49	0,74	0,0001	$k = 2$
K-medoids	2	0,32	0,51	0,73	0,1825	$k = 2$
CLARA	2	0,25	0,45	0,75	0,0030	$k = 2$
GMM	3	0,69	0,74	0,42	3,2469	$k = 3$
DIANA	2	0,24	0,44	0,75	1,0963	$k = 2$
CLINK	2	0,23	0,43	0,75	0,1092	$k = 2$
PdfCluster	1	-	-	-	332,8038	-
DBSCAN[1]	2	0,22	0,42	0,75	0,1002	$eps = 61,49$; $MinPts = 5$

[1]ruído=3

Fonte: O autor (2021).

MulticlusterKDE e AdditiveclusterKDE tiverem os maiores tempos computacionais.

- **Google** ($X \in \mathbb{R}^{5456 \times 24}$)

O conjunto de dados *Travel review ratings*, denominado "Google", é constituído por 5456 avaliações do Google sobre atrações de 24 categorias em toda a Europa e está disponível em Asuncion e Newman (2007). A classificação do Google varia de 1 a 5 e a classificação média do usuário por categoria é calculada, sendo as categorias: igrejas, *resorts*, praias, parques, teatros, museus, shoppings, zoológicos, restaurantes, *pubs*, locais, pizzarias, outros alojamentos, bares de suco, galeria de arte, clubes de dança, piscinas, academias, padarias, beleza e spas, cafés, pontos de vista, monumentos e jardins. Os resultados dessa clusterização estão dispostos na Tabela 8.

Tabela 8 – Resultado dos algoritmos aplicados ao conjunto Google

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	2	0,97	1,85	0,16	2,4573	$\alpha = 3$; $A = 5$; $m_A = 136$
MulticlusterKDE	2	1,00	1,87	0,15	107,2036	$\alpha = 1$
TreeKDE	2	1,02	1,96	0,15	0,6012	$\alpha = 5,5$; $MinPts = 200$
K-means	15	1,05	1,88	0,17	0,0244	$k = 15$
K-medoids	6	1,15	2,13	0,13	22,4933	$k = 6$
CLARA	2	1,26	2,49	0,14	0,0101	$k = 2$
GMM	2	1,49	2,85	0,12	5,4718	$k = 2$
DIANA	3	0,95	1,88	0,14	467,7150	$k = 3$
CLINK	2	1,20	2,43	0,14	5,4739	$k = 2$
PdfCluster	Error: cannot allocate vector of size 26.6 Gb					
DBSCAN[1]	3	0,80	1,09	0,101	1,2354	$esp = 4,36$; $MinPts = 5$

[1]ruído=76

De acordo com os resultados, é possível observar que para esse conjunto, não houve uma concordância entre os algoritmos sobre o número ideal de grupos, sendo, no

entanto, a configuração com 2 grupos a mais frequente, 6 vezes. O Algoritmo DBSCAN foi o que apresentou o menor índice CD e DB, no entanto, foi o que apresentou o pior coeficiente de silhueta. Outro ponto sobre o DBSCAN é que este deixou de agrupar 76 observações, classificadas como ruídos.

Os algoritmos propostos neste trabalho apresentaram resultados que nos levam a classificar os desempenhos como intermediários, ficando bem avaliados nas três métricas. Cabe observar que a clusterização do conjunto Google mostrou-se de grande complexidade, devido à variabilidade das soluções e os valores das métricas. Considerando o índice CD, observamos que 7 algoritmos apresentaram valores superiores a 1 como melhor desempenho e, conforme discutido na Seção 3.1, uma clusterização com essa avaliação é considerada ineficiente. Da mesma forma que o índice CD, o índice DB e o coeficiente de silhueta também evidenciaram as mesmas deficiências no agrupamento do conjunto. O algoritmo PdfCluster, por sua vez, apresentou um erro de capacidade de memória excedida, conforme pode ser observado na Tabela 8.

Quanto ao tempo de execução, o Algoritmo DIANA foi o que mais demorou para realizar a clusterização desejada, ultrapassando 467 segundos. Por outro lado, os algoritmos TreeKDE, K-means e CLARA executaram o agrupamento em menos de 1 segundo, evidenciando a grande diferença entre os tempos de execução dos algoritmos utilizados nesses experimentos. Considerando apenas os algoritmos MulticlusterKDE e AdditiveclusterKDE, fica evidente as implicações do uso de amostragem pelo Algoritmo AdditiveclusterKDE no tempo de execução quando comparado ao Algoritmo MulticlusterKDE.

- ***USArrests*** $(\mathbb{R}^{50\times4})$

O conjunto de dados "*USArrests*", foi apresentado inicialmente por McNeil (1977) e, também, discutido em Kassambara (2017) e pode ser obtido no pacote "datasets"(R Core Team (2019)). Esse conjunto contém a quantidade de detenções a cada 100.000 habitantes pelos crimes de agressão, assassinato e estupro levando em consideração os 50 estados dos EUA no ano 1973. Além das informações dos crimes também é fornecida a porcentagem da população que vive em áreas urbanas. Desta forma, a amostra contém 50 observações, cada uma com 4 atributos (*murder*, *assault*, *urbanpop* e *rape*), sendo este o menor conjunto em número de observações abordado nesse trabalho. Os resultados da clusterização são apresentados na Tabela 9.

De acordo com os resultados expostos, Tabela 9, percebemos que esse problema apresentou uma maior similaridade com relação ao número de grupos considerados ideal, assim como os valores das métricas de validação. Com exceção do Algoritmo DBSCAN todos os outros algoritmos acusaram a clusterização com 2 grupos como a mais adequada.

Tabela 9 – Resultado dos algoritmos aplicados ao conjunto *USArrests*

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	2	0,27	0,59	0,59	0,3684	$\alpha = 2$, $A = 5$, $m_A = 20$
MulticlusterKDE	2	0,27	0,59	0,59	0,6475	$\alpha = 0,5$
TreeKDE	2	0,30	0,58	0,58	0,0050	$\alpha = 0,9$; $MinPts = 5$
K-means	2	0,27	0,60	0,59	0,0010	$k = 2$
K-medoids	2	0,27	0,60	0,59	0,0020	$k = 2$
CLARA	2	0,27	0,60	0,59	0,0199	$k = 2$
GMM	2	0,29	0,67	0,55	0,1574	$k = 2$
DIANA	2	0,29	0,67	0,54	0,0130	$k = 2$
CLINK	2	0,30	0,57	0,57	0,0001	$k = 2$
PdfCluster	2	0,34	0,68	0,51	0,0911	$hmult = 0,25$
DBSCAN[1]	4	0,44	0,80	0,47	0,0311	$eps = 25,6$; $MinPts = 5$

[1]ruído=3

Fonte: O autor (2021).

Com base nas métricas, percebemos uma proximidade das soluções, com vários algoritmos indicando os melhores resultados. Cabe observar que a pequena diferença entre os resultados é consequência da designação de elementos de forma diferente, isto é, apesar dos algoritmos apresentarem o mesmo número de grupos, isso não significa que a disposição dos elementos seja a mesma, visto que cada algoritmo possui sua própria metodologia. Como o problema *USArrests* é de pequeno porte, a divergência de um único elemento já pode provocar uma variação significativa no valor das métricas de validação.

- ***Ratio*** $(X \in \mathbb{R}^{75\times5})$

O conjunto "*Ratio*" faz parte do pacote "*clusterSim*"(Walesiak e Dudek (2019)) e é constituído por 75 observações com 5 atributos, gerados artificialmente. De forma semelhante ao conjunto *USArrests* esse também é um conjunto com baixo número de observações e os resultados de sua clusterização estão apresentados na Tabela 10.

O conjunto *Ratio*, de forma semelhante ao conjunto Google, não apresentou uma concordância no número de grupos para os diferentes algoritmos aqui utilizados. Considerando o índice CD, o Algoritmo GMM foi o que apresentou o melhor resultado, com uma clusterização com 7 grupos. Observando os resultados para o índice DB, o melhor resultado foi obtido pelo algoritmo K-means com 20 grupos e, por fim, o algoritmo K-medoids com o melhor coeficiente de silhueta, também com 7 grupos.

Quanto a tempo de execução, observamos que o Algoritmo AdditiveclusterKDE, apesar de trabalhar com amostras, apresentou dificuldades com um conjunto de pequeno porte, visto que o seu tempo de execução foi superior aos demais algoritmos, indicando que seu uso é mais apropriado para problemas com elevado número de observações.

Tabela 10 – Resultado dos algoritmos aplicados ao conjunto *Ratio*

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	5	0,68	1,29	0,28	2,7908	$\alpha = 2$, $A = 5$, $m_A = 30$
MulticlusterKDE	10	0,68	1,19	0,24	2,0997	$\alpha = 1$
TreeKDE	5	0,76	1,32	0,30	0,0156	$\alpha = 0,5$; $MinPts = 10$
K-means	20	0,62	0,91	0,24	0,0001	$k = 20$
K-medoids	7	0,61	1,16	0,32	0,0030	$k = 7$
CLARA	8	0,61	1,14	0,31	0,0050	$k = 8$
GMM	7	0,60	1,12	0,31	0,0378	$k = 7$
DIANA	20	0,84	1,06	0,11	0,0020	$k = 20$
CLINK	5	0,72	1,22	0,31	0,0001	$k = 5$
PdfCluster	4	0,75	1,36	0,19	0,2916	-
DBSCAN[1]	3	0,96	1,62	0,20	0,0001	$eps = 5,09$; $MinPts = 5$

[1]ruído=9

Fonte: O autor (2021).

- **Multinormais ($X \in \mathbb{R}^{123\times181}$)**

O conjunto Multinormais foi gerado randomicamente por múltiplas amostras normalmente distribuídas de acordo com os seguintes passos. Inicialmente geramos dois valores inteiros positivos n e g, em que n representa a dimensão das amostras e g a quantidade de amostras com distribuição normal. Para cada uma das g amostras normais geramos m_i pontos, totalizando $m = \sum_{i=1}^{g} m_i$ elementos do espaço $\mathbb{R}^n$. Para construir as amostras normais, geramos aleatoriamente o vetor de médias μ com matriz de variância Σ, simétrica e definida positiva. De forma resumida, ao fim temos o conjunto

$$X = \bigcup_{i=1}^{g} N \sim (m_i, \mu_i, \Sigma_i). \tag{4.1}$$

Executando os passos descritos, obtivemos um conjunto de 123 observações do espaço $\mathbb{R}^{181}$, ou seja, temos um problema com o número de atributos superior ao número de observações. Os resultados da clusterização desse conjunto estão expostos na Tabela 11.

Com base na Tabela 11, é notório que o conjunto de dados gerados aleatoriamente constitui 3 grupos bem definidos, sendo que apenas o Algoritmo PdfCluster não conseguiu determinar esse resultado. Quanto ao tempo de execução, todos os algoritmos que utilizam o KDE multidimensional apresentaram os maiores tempo de execução, principalmente para esse problema, em que o número de variáveis é maior em relação aos demais problemas. No entanto, devemos destacar que o uso do KDE permite aos algoritmos AdditiveclusterKDE, MulticlusterKDE e TreeKDE a detecção do número de grupos de forma autônoma, sem a necessidade de realizar uma varredura como realizado nos algoritmos K-means, CLARA, K-medoids, GMM, CLINK e DIANA.

Tabela 11 – Resultado dos algoritmos aplicados ao conjunto Multinormais.

Algoritmo	k	CD	DB	Silhueta	tempo	Parâmetros
AdditiveclusterKDE	3	0,14	0,28	0,80	9,2543	$\alpha = 1$, $A = 5$, $m_A = 25$
MulticlusterKDE	3	0,14	0,28	0,80	15,5350	$\alpha = 0,25$
TreeKDE	3	0,14	0,28	0,80	0,3691	$\alpha = 1,5$; $MinPts = 5$
K-means	3	0,14	0,28	0,80	0,0040	$k = 3$
K-medoids	3	0,14	0,28	0,80	0,0150	$k = 3$
CLARA	3	0,14	0,28	0,80	0,0120	$k = 3$
GMM	3	0,14	0,28	0,80	0,1003	$k = 3$
DIANA	3	0,14	0,28	0,80	0,0318	$k = 3$
CLINK	3	0,14	0,28	0,80	0,0032	$k = 3$
PdfCluster	1	-	-	-	3,7221	-
DBSCAN[1]	3	0,14	0,28	0,80	0,1444	$eps = 18,8$; $MinPts = 5$

[1]ruído=0

Fonte: O autor (2021).

4.3.1 Comentários adicionais sobre os problemas de clusterização

Finalizando os experimentos numéricos para problemas de clusterização, observamos que os algoritmos propostos neste trabalho mostraram-se competitivos em relação aos outros algoritmos da literatura submetidos à comparação. Com base nos valores das métricas de validação, provenientes de cada uma das clusterizações produzidas, os algoritmos AdditiveclusterKDE e MulticlusterKDE apresentaram bons resultados. Por outro lado, o Algoritmo TreeKDE não apresentou os melhores desempenhos, no entanto, é preciso ressaltar que as métricas, aqui utilizadas, se baseiam em distância Euclidiana diferentemente da clusterização proveniente do Algoritmo TreeKDE.

Quanto ao tempo de execução, sem dúvidas os algoritmos de particionamento tais como K-means e CLARA apresentaram os melhores desempenho em uma análise geral, no entanto, esses são algoritmos que necessitam da informação do número de clusters para serem executados, necessitando de várias execuções para conseguir a solução ótima. Outro algoritmo que se mostrou extremamente rápido foi o DBSCAN, porém esse também requer uma quantidade enorme de tentativas para se obter a clusterização ideal, sendo que neste trabalho, por exemplo, para cada um dos problemas, foram realizadas 100 diferentes clusterizações com variações dos parâmetros.

Com relação aos três algoritmos propostos nesse trabalho, fica evidente que o Algoritmo TreeKDE apresentou um tempo de execução muitas vezes menor que os outros dois. Tal vantagem ocorre pelo fato de que o Algoritmo TreeKDE utiliza a otimização da função KDE unidimensional para a determinação das variáveis de partição, enquanto os outros dois realizam a otimização da função KDE multidimensional na determinação dos centroides.

Comparando os algoritmos MulticlusterKDE e AdditiveclusterKDE fica evidente

a diferença de tempo de suas execuções quando temos problemas com número elevado de observações, sendo essa diferença consequência de dois princípios. O primeiro se deve ao fato do Algoritmo AdditiveclusterKDE trabalhar com amostras e o segundo pelo uso da restrição de caixa que restringe o espaço de busca tornando o processo de otimização da função KDE mais eficiente.

4.4 TESTES NUMÉRICOS COM PROBLEMAS DE CLASSIFICAÇÃO

Nessa seção, apresentamos os resultados da classificação de 8 problemas (*Iris, Wine, Olive oil, Seeds, Facebook, WBDC, Divorce e Heart*), a qual se tem o conhecimento prévio do número de clusters e das classes que cada uma das observações pertencem. O objetivo do uso desse tipo de problema é a possibilidade de reconstrução da estrutura já definida pelo conjunto de dados, evidenciando a capacidade dos algoritmos em classificar dados com número de grupos e classes conhecidas.

Para os problemas de classificação, foram analisados os desempenhos de 8 algoritmos: AdditiveclusterKDE, MulticlusterKDE, K-means, K-medoids, CLARA, GMM, DIANA e CLINK. Em relação a Tabela 3, ficaram de fora desses experimentos os algoritmos TreeKDE, PdfCluster e DBSCAN, pois nesses não é possível delimitar um número exato de clusters.

Nessa seção, apresentamos a avaliação da qualidade dos algoritmos por meio do emprego das métricas de validação externas, discutidas na Seção 2.1.1. Para ilustrar os resultados dos testes para cada um dos problemas, foram construídas duas tabelas. A primeira apresenta as matrizes de confusão provenientes de cada um dos algoritmos executados, cujos nomes foram abreviados a fim de facilitar a descrição, isto é, o "Algoritmo AdditiveclusterKDE" é denominado "Additive"e o "Algoritmo MulticlusterKDE" por "Multi". Já a segunda, exibe a quantidade de observações classificadas incorretamente, ou seja, o erro absoluto (E_{abs}), o tempo de execução em segundos e os valores das métricas: acurácia, precisão, sensibilidade e F1-score.

- ***Iris*** $(X \in \mathbb{R}^{150\times4})$

O conjunto de dados "*Iris*" é um conjunto multivariado introduzido no trabalho de Fisher e Marshall (1936), contendo 150 observações de 3 variedades de plantas da espécie *Iris*, igualmente divididas entre as variedades *Setosa*, *Versicolour* e *Virgínica*. Cada uma das observações é constituída por 4 atributos ou características, sendo eles o comprimento e a largura da pétala e o comprimento e a largura da sépala, todas em centímetros.

Para execução da classificação desse conjunto, no Algoritmo AdditiveclusterKDE os parâmetros de entrada $\alpha = 0,25$, $m_A = 30$, $A = 10$ e $nc = 3$ foram utilizados. Já no

Algoritmo MulticlusterKDE empregamos os parâmetros $\alpha = 0,1$ e $nc = 3$. Para todos os outros algoritmos foi utilizado $k = 3$ como parâmetro de entrada. Com base nessas informações, os algoritmos foram executados e os resultados da classificação do problema das *Iris* são apresentados na Tabela 12 e Tabela 13.

Tabela 12 – Matriz de confusão para o conjunto *Iris*

Variedades	Additive			Multi			K-means			K-medoids		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Setosa	50	0	0	50	0	0	50	0	0	50	0	0
Versicular	0	48	2	0	49	1	0	48	2	0	47	3
Virginica	0	4	46	0	4	46	0	4	46	0	4	46
	CLARA			GMM			DIANA			CLINK		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Setosa	50	0	0	50	0	0	50	0	0	50	0	0
Versicular	0	48	2	0	47	3	0	31	19	0	29	21
Virginica	0	4	46	0	2	48	0	11	39	0	2	48

Fonte: O autor (2021).

Tabela 13 – Comparativo dos resultados para o conjunto *Iris*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	6	0,96	0,96	0,96	0,96	3,276
MulticlusterKDE	5	0,97	0,97	0,97	0,97	2,115
K-means	6	0,96	0,96	0,96	0,96	0,052
K-medoids	7	0,95	0,95	0,95	0,95	0,055
CLARA	6	0,96	0,96	0,96	0,96	0,109
GMM	5	0,97	0,97	0,97	0,97	0,199
DIANA	30	0,80	0,80	0,80	0,80	0,063
CLINK	23	0,85	0,88	0,85	0,84	0,056

Fonte: O autor (2021).

De acordo com os resultados, fica evidente uma similaridade entre os resultados de 6 dos 8 algoritmos, com uma leve vantagem para os algoritmos MulticlusterKDE e GMM. Com base na acurácia e no F1-score, percebemos uma classificação satisfatória do conjunto *Iris*, sendo que apenas os algoritmos DIANA e CLINK apresentaram resultados não satisfatórios para classificação do problema. Outro fator interessante é a classificação precisa da espécie Setosa em todos os algoritmos, ficando os erros de classificação exclusivamente entre as espécies *Versicular* e *Virginica*.

- ***Wine*** $(X \in \mathbb{R}^{178\times13})$

O conjunto "*Wine*" foi proposto inicialmente por Forina et al. (1988) como resultado de uma análise química de vinhos cultivados na Itália. O conjunto é constituído de 178

vinhos cultivados numa mesma região da Itália, mas produzidos a partir de três diferentes cultivares (*Barolo*, *Grignolino*, *Barbera*). A amostra é composta por 59 vinhos de *Barolo*, 71 de *Grignolino* e 48 de *Barbera*. Cada um dos vinhos possui 28 características químicas, no entanto segundo Menardi e Azzalini (2014), apenas 13 são comumente escolhidas entre as 28 originais, sendo elas: *Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines e Proline.*

Para resolver esse problema de classificação, no Algoritmo AdditiveclusterKDE consideramos os parâmetros $\alpha = 0,25$, $A = 10$, $m_A = 36$ e $nc = 3$, enquanto no Algoritmo MulticlusterKDE os valores $\alpha = 3$ e $nc = 3$ foram utilizados. Para os demais algoritmos o parâmetro $k = 3$ foi escolhido.

Tabela 14 – Matriz de confusão para o conjunto *Wine*

Cultivares	Additive			Multi			K-means			K-medoids		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Barolo	59	0	0	59	0	0	59	0	0	59	0	0
Grignolino	2	67	2	4	66	1	6	63	2	4	66	1
Barbera	0	1	47	0	0	48	0	0	48	0	1	47
	CLARA			GMM			DIANA			CLINK		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Barolo	59	0	0	58	1	0	58	1	0	58	1	0
Grignolino	4	66	1	0	68	3	2	68	1	6	65	0
Barbera	0	1	47	0	0	48	47	1	0	0	20	28

Fonte: O autor (2021).

Tabela 15 – Comparativo dos resultados para o conjunto *Wine*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	5	0,97	0,97	0,97	0,97	2,189
MulticlusterKDE	5	0,97	0,97	0,98	0,97	4,082
K-means	8	0,96	0,96	0,96	0,96	0,002
K-medoids	6	0,97	0,97	0,97	0,97	0,970
CLARA	6	0,97	0,97	0,97	0,97	0,004
GMM	4	0,98	0,98	0,98	0,98	0,185
DIANA	52	0,71	0,5	0,65	0,55	0,003
CLINK	27	0,85	0,89	0,83	0,84	0,006

Fonte: O autor (2021).

De acordo com os resultados dispostos na Tabela 14 e Tabela 15, temos uma classificação com eficiência superior a 95% em 6 dos 8 algoritmos analisados, sendo o GMM o que apresentou a melhor classificação, no entanto, em termos absolutos, isso corresponde a apenas uma observação classificada corretamente a mais que os algoritmos Additive-

clusterKDE e MulticlusterKDE. Novamente os algoritmos DIANA e CLINK foram os que apresentaram os piores resultados.

Vale observar que embora os resultados sejam semelhantes ou até mesmo iguais, a classificação dos vinhos apresentada pelos algoritmos são diferentes, como pode ser observado pelas matrizes de confusão da Tabela 14. Por exemplo, os algoritmos AdditiveclusterKDE e MulticlusterKDE apresentaram 5 classificações erradas, no entanto, com matrizes de confusão distintas.

- ***Seeds*** $(X \in \mathbb{R}^{210\times7})$

O conjunto de dados, denominado "*Seeds*", coletado e analisado por Charytanowicz et al. (2010) e disponibilizado por Asuncion e Newman (2007) apresenta informações sobre sementes de três variedades de trigo: *Kama*, *Rosa* e *Canadian*. Tal conjunto é composto por 210 observações, 70 sementes para cada uma das 3 variedades. Cada observação é composta por 7 parâmetros geométricos das sementes de trigo: área (A), perímetro (P), compacidade $(C = \frac{4\pi A}{P^2})$, comprimento do núcleo, largura do núcleo, coeficiente de assimetria e comprimento do sulco do núcleo.

Para realizar a classificação dessas sementes o Algoritmo AdditiveclusterKDE foi executado com os parâmetros $\alpha = 0,1$, $A = 10$, $m_A = 42$ e $nc = 3$, enquanto no Algoritmo MulticlusterKDE utilizamos $\alpha = 0,5$ e $nc = 3$. De forma semelhante aos outros problemas, nos demais algoritmos o parâmetro $k = 3$ foi utilizado, resultando na classificação exposta na Tabela 16 e Tabela 17.

Tabela 16 – Matriz de confusão para o conjunto *Seeds*

Variedades	Additive			Multi			K-means			K-medoids		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Kama	60	2	8	60	2	8	60	1	9	57	1	12
Rosa	5	65	0	10	60	0	10	60	0	10	60	0
Canadian	3	0	67	3	0	67	2	0	68	0	0	70
	CLARA			GMM			DIANA			CLINK		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Kama	60	2	8	58	2	10	57	11	2	63	3	4
Rosa	9	61	0	1	69	0	0	70	0	19	51	0
Canadian	4	0	66	0	0	70	8	0	62	15	0	55

Fonte: O autor (2021).

Com base nos resultados, novamente o Algoritmo GMM apresentou o melhor resultado, superando o Algoritmo AdditiveclusterKDE em 5 classificações corretas. Para esse conjunto, todos os algoritmos apresentaram uma acurácia, assim como um F1-score, superior a 0,8, evidenciado a homogeneidade dos algoritmos em classificar as diferentes variedades de sementes de trigo.

Tabela 17 – Comparativo dos resultados para o conjunto *Seeds*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	18	0,91	0,92	0,91	0,91	2,466
MulticlusterKDE	23	0,89	0,89	0,89	0,89	3,824
K-means	22	0,90	0,90	0,90	0,90	0,008
K-medoids	23	0,89	0,90	0,89	0,89	0,017
CLARA	23	0,89	0,89	0,89	0,89	0,890
GMM	13	0,94	0,94	0,94	0,94	0,080
DIANA	21	0,90	0,90	0,90	0,90	0,015
CLINK	41	0,80	0,84	0,80	0,81	0,004

Fonte: O autor (2021).

- ***Olive oil*** $(X \in \mathbb{R}^{572\times 8})$

O conjunto de dados "*Olive oil*" foi originalmente apresentado por Forina et al. (1983) e usado por vários pesquisadores para validação de algoritmos de clusterização, dentre estes Menardi e Azzalini (2014). De acordo com Forina et al. (1983) e Menardi e Azzalini (2014) os dados são referentes a azeites produzidos em 9 áreas da Itália, concentradas em 3 macro-áreas geográficas: Sul, Ilha da Sardinia e Centro-Norte. Na macro-área do Sul, temos as regiões da *Apulia north, Apulia south, Calabria* e *Sicily*. Na macro-área Sardina temos as regiões da costa e do interior da *Sardina* e, por último, na macro região centro-norte temos as regiões da *Liguria.east, Liguria.west* e *Umbria*. Ainda segundo Menardi e Azzalini (2014), o conjunto de dados em sua forma bruta é de natureza composicional, totalizando 10.000 elementos. Em seu trabalho, os autores adotaram uma transformação aditiva log-ratio (ARL), reduzindo o conjunto de análise para 572 observações. Cada uma das observações é constituída de 8 medições químicas do azeite (*Palmitic, Palmitoleic, Stearic, Oleic, Linoleic, Linolenic, Arachidic, eicosenoic*), além das informações da sua região e macro-região de origem.

De forma semelhante ao trabalho de Menardi e Azzalini (2014), optamos por utilizar a informação de classificação do azeite pelas 3 macro-regiões, estruturando o conjunto em 3 grupos, onde 323 observações são da macro-região Sul, 98 da Ilha da *Sardina* e 151 da macro-região Centro-Norte. Vale observar que, segundo Tharwat (2020), este conjunto apresenta classes desequilibradas, visto que macro-região Sul apresenta mais observações que as outras duas regiões.

Como parâmetros de entrada, no Algoritmo AdditiveclusterKDE utilizamos $\alpha = 5$, $m_A = 57$, $A = 10$ e $nc = 3$. No Algoritmo MulticlusterKDE, os parâmetros $\alpha = 1,5$ e $nc = 3$ foram empregados, enquanto nos demais algoritmos foi estabelecido $k = 3$. Os resultados da classificação estão dispostos na Tabela 18 e Tabela 19.

De acordo com a Tabela 18 e Tabela 19, os melhores resultados referentes a clas-

Tabela 18 – Matriz de confusão para o conjunto *Olive oil*

macro-area	Additive			Multi			K-means			K-medoids		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Sul	322	1	0	318	0	5	322	1	0	322	1	0
Sardina	0	86	12	0	97	1	0	98	0	0	98	0
Centre-North	0	8	143	0	23	128	0	98	53	0	83	68
	CLARA			GMM			DIANA			CLINK		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
Sul	322	0	1	281	0	42	272	0	51	323	0	0
Sardina	0	0	98	0	97	1	97	0	1	0	92	6
Centre-North	0	48	103	0	0	151	0	65	86	0	87	64

Fonte: O autor (2021).

Tabela 19 – Comparativo dos resultados para o conjunto *Olive oil*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	21	0,96	0,94	0,94	0,94	5,060
MulticlusterKDE	29	0,95	0,92	0,94	0,93	5,683
K-means	99	0,83	0,83	0,78	0,73	0,005
K-medoids	84	0,85	0,85	0,82	0,77	0,101
CLARA	147	0,74	0,50	0,56	0,53	0,006
GMM	43	0,92	0,93	0,95	0,93	0,546
DIANA	214	0,63	0,45	0,47	0,46	0,258
CLINK	93	0,84	0,81	0,79	0,75	0,027

Fonte: O autor (2021).

sificação do azeite, foram determinados pelos algoritmos AdditiveclusterKDE e o MulticlusterKDE nessa ordem, respectivamente. O Algoritmo GMM foi o que apresentou a terceira melhor classificação, no entanto, com mais que o dobro de classificações equivocadas quando comparado ao Algoritmo AdditiveclusterKDE. Como ambas as medidas de acurácia e F1-Score são altas, é possível enfatizar a qualidade da classificação propiciada por esses algoritmos. Por outro lado, os algoritmos CLARA e DIANA foram os que apresentaram os piores resultados com destaque para os baixos valores do F1-Score, provocados pela classificação equivocada de todos os azeites da macro-região da Sardina.

- ***Facebook*** $(X \in \mathbb{R}^{7050\times 9})$

O conjunto de dados "*Facebook Live Sellers in Thailand*", denominado aqui apenas por "*Facebook*", consiste em informações de páginas do *facebook* de 10 vendedores varejistas tailandeses da área de moda e de cosméticos, Dehouche e Wongkitrungrueng (2018). O conjunto é composto por 7050 observações de postagens de vídeo, fotos, *status* e *links*. Cada uma das postagens é avaliada por 9 medidas numéricas (*reactions*, *comments*, *shares*, *likes*, *loves*, *wows*, *hahas*, *sads*, e *angrys*).

A variável que caracteriza as classes do problema é o atributo tipo da publicação, o qual é dividido em 4 categorias: vídeo, fotos, *status* e *links* com, respectivamente, 4288, 2334, 365 e 63 observações para cada uma das categorias. Da mesma forma que o problema "*Olive oil*", o problema "*Facebook*" também apresenta classes desbalanceadas e, é dentre os conjuntos discutidos nessa seção, o que apresenta o maior número de observações, 7050.

Para realizar sua classificação consideramos no Algoritmo AdditiveclusterKDE os parâmetros $\alpha = 0,1$, $A = 10$, $m_A = 176$ e $nc = 4$, no Algoritmo MulticlusterKDE $\alpha = 0,5$ e $nc = 4$, enquanto nos demais algoritmos o parâmetro $k = 4$ foi utilizado. Os resultados da classificação estão dispostos na Tabela 20 e Tabela 21.

Tabela 20 – Matriz de confusão para o conjunto *Facebook*

Tipo de status	Additive				Multi				K-means			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
photo	4027	17	170	74	4128	1	152	7	4064	0	223	1
video	1684	555	68	27	1873	401	56	4	1854	316	107	57
status	277	3	71	14	333	0	32	0	287	0	78	0
link	49	0	13	1	57	0	6	0	49	0	14	0
	K-medoids				CLARA				GMM			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
photo	4025	17	245	1	3252	788	247	1	2868	224	1107	89
video	1429	617	149	139	1055	868	160	251	788	625	429	492
status	280	0	85	0	201	79	85	0	156	42	148	19
link	49	0	14	0	44	5	14	0	38	2	23	0
	DIANA				CLINK							
	C1	C2	C3	C4	C1	C2	C3	C4				
photo	4283	0	4	1	3369	9	908	2				
video	2306	22	0	6	908	745	412	269				
status	365	0	0	0	208	0	157	0				
link	63	0	0	0	47	0	16	0				

Fonte: O autor (2021).

Tabela 21 – Comparativo dos resultados para o conjunto *Facebook*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	2396	0,66	0,47	0,35	0,34	4,791
MulticlusterKDE	2489	0,65	0,44	0,31	0,29	28,920
K-means	2592	0,63	0,46	0,32	0,30	0,043
K-medoids	2323	0,67	0,46	0,36	0,36	20,189
CLARA	2845	0,60	0,35	0,34	0,34	0,107
GMM	3409	0,52	0,38	0,34	0,31	10,235
DIANA	2745	0,61	0,40	0,25	0,20	946,099
CLINK	2779	0,61	0,46	0,38	0,35	3,636

Fonte: O autor (2021).

De acordo com os resultados apresentados é perceptível a ineficiência dos algoritmos em classificar o conjunto *Facebook* devido à diferença entre o valor da acurácia e das outras medidas, principalmente o F1-score. Essa discrepância caracteriza uma acurácia duvidosa, que foi obtida apenas pela alta concentração de elementos de uma mesma classe em um mesmo grupo. Este fato é facilmente notado ao se observar a quantidade de elementos dispostos na posição $(1, 1)$ da matriz de confusão com relação aos elementos da posição $(4, 4)$, em que praticamente nenhum algoritmo conseguiu classificar as observações da classe *link* corretamente.

Apesar da ineficiência na classificação desse conjunto os algoritmos que apresentaram os melhores resultados foram os algoritmos K-medoids e AdditiveclusterKDE, enquanto os algoritmos DIANA e GMM apresentaram as piores classificações.

- **WBDC** ($X \in \mathbb{R}^{569\times30}$)

O conjunto *Breast Cancer Wisconsin (Diagnostic)*, denominado "WBDC" é constituído por 569 observações de pacientes de *Wisconsin*, submetidos a um exame, "*fine needle aspirate (FNA)*", para diagnosticar câncer de mama. Para cada uma das observações, o conjunto de dados apresenta a identificação do paciente (a qual será excluída no processo de classificação), o resultado do exame com diagnóstico maligno ou benigno (variável de classificação) e outros 30 atributos provenientes do exame FNA. Das 569 observações do conjunto 212 foram diagnosticadas com câncer de mama maligno, enquanto 357 foram diagnosticados como benigno.

Esse conjunto representa um problema de classificação binária, para tal utilizamos nos algoritmos os seguintes parâmetros: no Algoritmo AdditiveclusterKDE, $\alpha = 0,5$, $A = 10$, $m_A = 57$ e $nc = 2$; no Algoritmo MulticlusterKDE, $\alpha = 0,1$ e $nc = 2$; nos algoritmos K-means, K-medoids, CLARA, GMM, DIANA e CLINK utilizamos $k = 2$. Com base nesses parâmetros os algoritmos retornaram os seguintes resultados, apresentados na Tabelas 22 e 23.

Tabela 22 – Matriz de confusão para o conjunto WBDC

Variedades	Additive		Multi		K-means		K-medoids		CLARA	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
Malignant	177	35	201	11	158	54	164	48	153	59
Benign	7	350	53	304	5	352	5	352	4	353
	GMM		DIANA		CLINK					
	C1	C2	C1	C2	C1	C2				
Malignant	157	55	153	59	151	61				
Benign	86	271	9	348	6	351				

Fonte: O autor (2021).

Tabela 23 – Comparativo dos resultados para o conjunto WBDC

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	42	0,93	0,94	0,91	0,92	3,573
MulticlusterKDE	64	0,89	0,88	0,90	0,88	10,142
K-means	59	0,90	0,92	0,87	0,88	0,006
K-medoids	53	0,91	0,93	0,88	0,90	0,053
CLARA	63	0,89	0,92	0,86	0,87	0,027
GMM	141	0,75	0,74	0,75	0,74	0,790
DIANA	68	0,88	0,90	0,85	0,86	0,520
CLINK	67	0,88	0,91	0,85	0,87	0,047

Fonte: O autor (2021).

Os resultados para classificação do conjunto WBDC foram similares, sendo que o Algoritmo AdditiveclusterKDE apresentou a melhor classificação, seguido pelos algoritmos K-medoids, K-means, MulticlusterKDE e CLARA. O Algoritmo GMM foi o único com uma acurácia inferior a 0,88.

- ***Divorce*** $(X \in \mathbb{R}^{170\times54})$

O conjunto *Divorce Predictors*, denominado "*Divorce*", oriundo do trabalho de Yöntem et al. (2019), é constituído por 170 observações de casais que responderam um questionário composto por 54 perguntas com informações pessoais sobre a vida de casados. Dos 170 casais, 84 eram divorciados e os outros 86 casados. Dessa forma, temos um problema de decisão binária constituído por 54 atributos mensurados por valores inteiros de 0 a 4. Para classificação, utilizamos nos algoritmos os seguintes parâmetros: no Algoritmo AdditiveclusterKDE, $\alpha = 0,5$, $A = 10$, $m_A = 34$ e $nc = 2$; no Algoritmo MulticlusterKDE, $\alpha = 2$ e $nc = 2$; nos demais algoritmos $k = 2$. Os resultados da classificação estão dispostos na Tabela 24 e Tabela 25.

Tabela 24 – Matriz de confusão para o conjunto *Divorce*

Estado civil	Additive		Multi		K-means		K-medoids		CLARA	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
Casados	86	0	86	0	86	0	86	0	86	0
Divorciados	4	80	4	80	4	80	4	80	4	80
	GMM		DIANA		CLINK					
	C1	C2	C1	C2	C1	C2				
Casados	86	0	86	0	86	0				
Divorciados	21	63	4	80	4	80				

Fonte: O autor (2021).

Os resultados da classificação mostram que apenas o Algoritmo GMM não apresentou o mesmo resultado para todas as métricas usadas, isto é, 7 dos 8 algoritmos apre-

Tabela 25 – Comparativo dos resultados para o conjunto *Divorce*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	4	0,98	0,98	0,98	0,98	2,941
MulticlusterKDE	4	0,98	0,98	0,98	0,98	4,873
K-means	4	0,98	0,98	0,98	0,98	0,004
K-medoids	4	0,98	0,98	0,98	0,98	0,070
CLARA	4	0,98	0,98	0,98	0,98	0,055
GMM	21	0,88	0,9	0,88	0,87	0,254
DIANA	4	0,98	0,98	0,98	0,98	1,017
CLINK	4	0,98	0,98	0,98	0,98	0,016

Fonte: O autor (2021).

sentaram 0,98 para acurácia, precisão, sensibilidade e F1-Score. Esse fato mostra que o conjunto *Divorce* é facilmente classificado por quase todos os algoritmos utilizados nessa seção.

- ***Heart failure clinical records*** **($X \in \mathbb{R}^{299\times12}$)**

O conjunto *Heart failure clinical records*, denominado "*Heart*", segundo Chicco e Jurman (2020), é composto por 299 registros médicos de pacientes com insuficiência cardíaca, coletados no *Faisalabad Institute of Cardiology* e no *Allied Hospital* em *Faisalabad* (Punjab, Paquistão), durante o período de abril a dezembro de 2015. O conjunto contém 12 recursos, que relatam informações clínicas, corporais e de estilo de vida dos pacientes, além da informação sobre óbito, ou seja, 203 dos pacientes são sobreviventes (valor da variável resposta igual a 0), enquanto 96 foram a óbito (valor da variável resposta igual a 1).

Para classificar os pacientes utilizamos os seguintes parâmetros: no Algoritmo AdditiveclusterKDE, $\alpha = 0,1$, $A = 10$, $m_A = 60$ e $nc = 2$; no Algoritmo MulticlusterKDE, $\alpha = 3$ e $nc = 2$; nos demais $k = 2$. Os resultados da classificação estão dispostos na Tabela 26 e Tabela 27.

Tabela 26 – Matriz de confusão para o conjunto *Heart*

Evento de óbito	Additive		Multi		K-means		K-medoids		CLARA	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
Vida	197	6	200	3	112	91	112	91	107	96
Morte	50	46	69	27	54	42	57	39	44	52
	GMM		DIANA		CLINK					
	C1	C2	C1	C2	C1	C2				
Vida	137	66	198	5	181	22				
Morte	66	30	82	14	87	9				

Fonte: O autor (2021).

Tabela 27 – Comparativo dos resultados para o conjunto *Heart*

Algoritmo	E_{abs}	Acurácia	Precisão	Sensibilidade	F1-Score	tempo(s)
AdditiveclusterKDE	56	0,81	0,84	0,72	0,75	1,318
MulticlusterKDE	72	0,76	0,82	0,63	0,64	3,146
K-means	145	0,52	0,50	0,49	0,49	0,001
K-medoids	148	0,51	0,48	0,48	0,47	0,013
CLARA	140	0,53	0,53	0,53	0,53	0,016
GMM	132	0,56	0,49	0,49	0,49	0,080
DIANA	87	0,71	0,72	0,56	0,53	0,047
CLINK	109	0,64	0,48	0,49	0,46	0,001

Fonte: O autor (2021).

Com base nos resultados, percebemos que o Algoritmo AdditiveclusterKDE apresentou o melhor resultado (acurácia de 0,81), seguido pelo Algoritmo MulticlusterKDE (acurácia de 0,76) e DIANA (acurácia de 0,71). Apenas os três algoritmos citados apresentaram uma acurácia superior a 0,7 e, dentre esses, apenas o AdditiclusterKDE apresentou um F1-Score também superior a 0,7, evidenciando a dificuldade dos algoritmos em classificar de forma satisfatória os pacientes com insuficiência cardíaca.

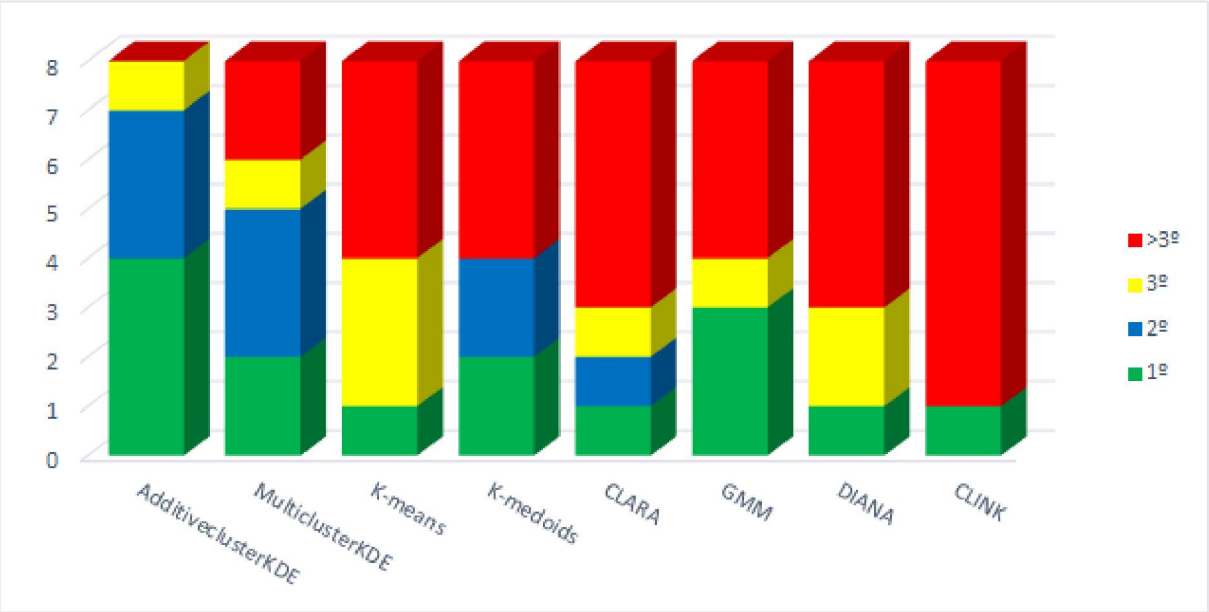
4.4.1 Comentários adicionais sobre os problemas de classificação

Nessa seção abordamos 8 problemas de classificação, os quais foram submetidos a 8 diferentes algoritmos de clusterização, sendo 2 deles propostos nesse trabalho e outros 6 já consagrados na literatura. Ao final dos experimentos, é possível observar que o Algoritmo AdditiveclusterKDE foi o que apresentou uma maior ocorrência de melhores resultados, conforme gráfico de colunas empilhadas, ilustrado pela Figura 31.

Ainda com base nos resultados indicados pelo gráfico, as colunas em verde representam a quantidade de vezes que o algoritmo apresentou a melhor acurácia dentre os resultados dos 8 algoritmos. De forma análoga, as colunas em azul representam a quantidade de vezes que cada algoritmo teve o segundo melhor resultado e as colunas em amarelo o terceiro melhor resultado. Para posições superiores ao terceiro lugar, temos as colunas na cor vermelha. Vale observar que, em caso de empate os algoritmos receberam a mesma classificação.

Nesse sentido, fica evidente que os dois algoritmos propostos nesse trabalho obtiveram um bom desempenho, principalmente o Algoritmo AdditiveclusterKDE que apresentou todas as soluções entre as três primeiras posições, sendo em 4 delas como a melhor. O Algoritmo MulticlusterKDE apresentou por duas vezes o melhor resultado e, em 6 dos 8 problemas, com classificação até a terceira melhor solução. Entre os demais algoritmos o GMM foi o que apresentou o melhor desempenho, obtendo 3 vezes a melhor classificação, no entanto, em outros 4 problemas apresentou uma classificação acima da terceira posição.

Figura 31 – Ranqueamento dos resultados para os problemas classificação



Fonte: O autor (2021).

Quanto ao tempo de execução, assim como ressaltado na seção anterior, os dois algoritmos propostos, não apresentaram os melhores tempos, figurando em todos os problemas com valores intermediários ou piores dependendo da dimensão dos dados. Os algoritmos K-means e CLARA executaram todas as classificações em tempos inferiores a 1 segundo, enquanto os algoritmos K-medoids, GMM, DIANA e CLINK se mostraram eficientes em problemas de baixa dimensão, no entanto encontraram algumas dificuldades em problemas maiores.

Analisando os tempos de execução apenas dos algoritmos AdditiveclusterKDE e MulticlusterKDE, observamos que esses apresentaram um desempenho não proporcional ao número de observações, isto é, o tempo de execução cresce a uma taxa inferior a razão do número de observações. Este fato se mostrou evidente no problema *Facebook*, em que os dois algoritmos propostos apresentaram tempos relativamente baixos quando comparados ao Algoritmo DIANA, por exemplo.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentamos três novos algoritmos para clusterização de dados multidimensionais, MulticlusterKDE, AdditiveclusterKDE e TreeKDE, além de uma nova métrica de validação interna para clusterização, índice CD.

Como visto, os algoritmos MulticlusterKDE e AdditiveclusterKDE possuem uma metodologia similar, sendo baseados na múltipla otimização da função KDE, com kernel Gaussiano multivariado utilizado na determinação do número de grupos, de seus respectivos centroides e com posterior alocação das observações pelo critério de mínima distância Euclidiana.

O Algoritmo MulticlusterKDE tem como principal característica, a construção da função KDE uma única vez durante a execução, contemplando todo o conjunto de dados e, a partir dessa função, é realizada a determinação dos centroides com alta concentração de observações definindo, dessa forma, grupos com formatos globulares. Mostramos que esse é um algoritmo bem definido que executa no máximo $m + 1$ passos.

O Algoritmo AdditiveclusterKDE, por sua vez, também trabalha com a obtenção de centroides, porém por meio de múltiplas amostras, com posterior clusterização da população tendo como base o conjunto de centroides provenientes das amostras e designação pelo critério de mínima distância Euclidiana. As principais características do Algoritmo AdditiveclusterKDE, além do uso de amostragem, é o fato deste realizar a otimização da função KDE sujeita a restrição de caixa definida com base no grupo mais heterogêneo, limitando assim, o espaço de busca por otimizadores da função. Mostramos também que o Algoritmo AdditiveclusterKDE é bem definido e executa no máximo $A \cdot (m_A + 1)$ passos.

Quando comparamos esses dois algoritmos, observamos que o Algoritmo MulticlusterKDE é um algoritmo mais simples, no entanto, o uso de amostras e da restrição de caixa sobre o grupo mais heterogêneo, propicia ao Algoritmo AdditiveclusterKDE melhores resultados e um tempo de execução menor quando submetido a problemas com elevado número de observações. Em ambos os algoritmos a classificação de uma nova observação ao problema é realizada pelo processo de designação ao centroide mais próximo, segundo o critério de mínima distância Euclidiana.

Já o Algoritmo TreeKDE apresenta uma metodologia diferente comparada aos dois algoritmos precedentes, apesar de ainda estar centrado no uso da função KDE. O Algoritmo TreeKDE está alicerçado em uma associação da metodologia de árvore de decisão com a otimização de funções KDE unidimensionais, construídas a partir da projeção ortogonal do conjunto de observações sobre os eixos coordenados. A redução da KDE multidimensional para unidimensional confere ao Algoritmo TreeKDE uma redução sig-

nificativa da complexidade da otimização envolvida, tornando sua execução muito mais rápida.

As principais características do Algoritmo TreeKDE são a formação de grupos arbitrários, linearmente separáveis e limitados por regiões hiper-retangulares do espaço e a independência da matriz de distância para alocação de observações aos grupos. Outro ponto interessante sobre o Algoritmo TreeKDE é a possibilidade de classificação de uma nova observação ao problema, sendo que esta é feita de forma simples e intuitiva, necessitando apenas da comparação dos valores de variáveis, assim como ocorre com o método de árvores de decisão. Além disso, uma outra característica relevante é que o Algoritmo TreeKDE é bem definido e sua rotina é executada em no máximo $\lceil log_2 \left(\frac{m}{MinPts}\right) \rceil$ passos.

Os três algoritmos propostos foram implementados no *Software R* e aplicados em problemas de clusterização e classificação de diversas dimensões. Os resultados numéricos indicaram que os algoritmos são satisfatórios para clusterização de dados multidimensionais com uma leve vantagem para o Algoritmo AdditiveclusterKDE quando comparados os valores das métricas de validação. Com relação ao tempo computacional o Algoritmo TreeKDE apresentou desempenho superior quando comparado aos algoritmos MulticlusterKDE e AdditiveclusterKDE. Além da eficiência, um dos aspectos principais dos algoritmos propostos é o de não requerer o número de grupos como parâmetro de entrada, sendo esse determinado pelo próprio algoritmo durante sua execução. Outro aspecto positivo é que os algoritmos são bem definidos e convergem independentemente do conjunto de dados a serem agrupados.

Como aspecto negativo, destacamos a dependência dos algoritmos à função KDE, que por sua vez é estritamente dependente da matriz largura de banda, H. Com o intuito de minimizar erros provenientes da estimação da função KDE, introduzimos, nos três algoritmos, um escalar (α) de controle de suavidade da função KDE.

Quanto ao tempo de execução, é possível observar pelos resultados numéricos, que os algoritmos AdditiveclusterKDE e MulticlusterKDE não possuem os menores tempos computacionais dentre os algoritmos comparados, no entanto, também não apresentam os maiores tempos, principalmente se o conjunto analisado tiver um número elevado de observações. Por outro lado, o Algoritmo TreeKDE se configurou como um dos algoritmos com menor tempo computacional durante os testes, ficando atrás apenas de algoritmos como CLARA e K-means. Se compararmos os algoritmos que utilizam a função KDE como metodologia de clusterização, sua execução se dá muitas vezes mais rápida que os demais, evidenciando o uso das KDEs unidimensionais em detrimento ao uso da KDE multidimensional.

Ainda sobre o tempo computacional é preciso considerar a relação existente entre o tempo de execução dos algoritmos propostos e a estrutura de cada conjunto de da-

dos a ser agrupado, uma vez que, quanto mais grupos forem definidos maior o número de iterações, no caso dos algoritmos MulticlusterKDE e AdditiveclusterKDE, e maior o número de partições, no caso do Algoritmo TreeKDE, aumentando consequentemente o tempo computacional.

Propomos ainda neste trabalho uma nova métrica de validação interna de clusterização, o índice de Densidade da Clusterização, o índice CD. Este possui a característica de avaliar cada grupo por meio da máxima razão entre a dispersão interna dos grupos com a separação entre grupos. De acordo com os experimentos, o índice CD, apresentou uma concordância significativa com as outras duas métricas de validação já consagradas na literatura, o índice DB e o coeficiente de silhueta. Com relação ao tempo de execução, o índice CD mostrou-se similar ao índice DB e superior ao coeficiente de silhueta, principalmente para problemas com elevado número de observações.

Mediante ao exposto, julgamos que as contribuições desse trabalho são significativas para o campo de estudo da clusterização podendo contribuir para o desenvolvimento de novas pesquisas.

Para finalizar, gostaríamos de observar que as informações sobre o Algoritmo MulticlusterKDE referem-se ao conteúdo do artigo Scaldelai et al. (2020) intitulado "*MulticlusterKDE: a new algorithm for clustering based on multivariate kernel density estimation*", publicado pela revista *Journal of Applied Statistics* no ano de 2020.

5.1 SUGESTÕES PARA FUTUROS TRABALHOS

Como sugestões/perspectivas para futuros trabalhos, procuraremos investigar uma abordagem multiobjetivo para a clusterização empregada nos algoritmos MulticlusterKDE e AdditiveclusterKDE, focada na seleção autônoma do melhor parâmetro α que maximiza a qualidade da clusterização.

No Algoritmo TreeKDE, concentraremos nossos esforços em buscar alternativas ao uso de partições ortogonais apenas sobre os eixos coordenados. Uma proposta vislumbrada a um curto espaço de tempo consiste em realizar, além das projeções ortogonais sobre os eixos, projeções ortogonais sobre as bissetrizes dos eixos. Essa ideia aumentaria, a cada iteração, um número de projeções na ordem de $n^2 - n$ que, acrescido às n projeções executadas pelo atual algoritmo, resultaria em n^2 projeções ortogonais a cada cluster pai. Supondo uma clusterização com k grupos, o número de otimizações, de funções KDE unidimensionais, seria na ordem de $(2k-1) \cdot n^2$, elevando o tempo de execução. No entanto, o uso de projeções não paralelas aos eixos, tende a melhorar consideravelmente a solução de uma gama de problemas, além de particionar o conjunto de dados em regiões hiper-poligonais, diferentes das hiper-retangulares utilizadas pelo Algoritmo TreeKDE.

Outra linha interessante para pesquisas futuras e ainda pouco explorada é a ex-

pansão do uso do estimador de densidade kernel para conjuntos de dados expressos por meio de matrizes de múltiplas vias, isto é, Tensores. Os tensores possuem uma gama de aplicações modernas, como reconhecimento de imagem e vídeo, mineração de texto, pesquisa na internet, telecomunicações e registro de redes sociais entre outros. A clusterização de tensores ainda é um campo pouco explorado e, ainda menos explorado, no contexto do estimador de densidade kernel.

Por fim, como pesquisa futura sugerimos o desenvolvimento de uma nova métrica de validação interna que contemple grupos não globulares de formatos arbitrários, ou seja, não dependente do cálculo de distância, por se tratar de um campo carente na literatura merecendo ser investigada.

REFERÊNCIAS

AGRAWAL, R.; GEHRKE, J.; GUNOPULOS, D.; RAGHAVAN, P. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, Springer Science and Business Media LLC, v. 11, n. 1, p. 5–33, jul. 2005. DOI: 10.1007/s10618-005-1396-1.

AHUJA, M.; BAL, J. Exploring cluster analysis. *International Journal of Computer and Information Technology (IJCIT)*, v. 3, p. 594–597, 2014. ISSN 2279 – 0764.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. OPTICS. *ACM SIGMOD Record*, Association for Computing Machinery (ACM), v. 28, n. 2, p. 49–60, jun. 1999. DOI: 10.1145/304181.304187.

ASUNCION, A.; NEWMAN, D. *UCI machine learning repository*. 2007. Disponível em: ⟨"http://archive.ics.uci.edu/ml"⟩.

AZZALINI, A.; MENARDI, G. et al. Clustering via nonparametric density estimation: The R package pdfcluster. *Journal of Statistical Software*, Foundation for Open Access Statistics, v. 57, n. 11, p. 1–26, 2014. DOI: 10.18637/jss.v057.i11.

AZZALINI, A.; TORELLI, N. Clustering via nonparametric density estimation. *Statistics and Computing*, Springer, v. 17, n. 1, p. 71–80, 2007. DOI: 10.1007/s11222-006-9010-y.

BALL, G. H.; HALL, D. J. Isodata, a novel method of data analysis and pattern classification. 1965.

BANFIELD, J. D.; RAFTERY, A. E. Model-based gaussian and non-gaussian clustering. *Biometrics*, JSTOR, p. 803–821, 1993. DOI: 10.2307/2532201.

BYRD, R. H.; LU, P.; NOCEDAL, J.; ZHU, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, v. 16, n. 5, p. 1190–1208, 1995. DOI: 10.1137/0916069.

CALIŃSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, Taylor & Francis, v. 3, n. 1, p. 1–27, 1974. DOI: 10.1080/03610927408827101.

CHACÓN, J. E.; DUONG, T. *Multivariate kernel smoothing and its applications*. [S.l.]: CRC Press, 2018. ISBN 781498763011.

CHACÓN, J. E.; DUONG, T.; WAND, M. Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*, JSTOR, v. 21, n. 2, p. 807–840, 2011. DOI: 10.5705/ss.2011.036a.

CHARYTANOWICZ, M.; NIEWCZAS, J.; KULCZYCKI, P.; KOWALSKI, P. A.; LUKASIK, S.; ŻAK, S. Complete gradient clustering algorithm for features analysis of x-ray images. In: *Information technologies in biomedicine*. [S.l.]: Springer Berlin Heidelberg, 2010. p. 15–24. DOI: 10.1007/978-3-642-13105-9_2.

CHICCO, D.; JURMAN, G. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC medical informatics and decision making*, Springer, v. 20, n. 1, p. 16, 2020. DOI: 10.1186/s12911-020-1023-5.

COHEN, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, Sage Publications Sage CA: Thousand Oaks, CA, v. 20, n. 1, p. 37–46, 1960. DOI: 10.1177/001316446002000104.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, n. 2, p. 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

DEFAYS, D. An efficient algorithm for a complete link method. *The Computer Journal*, Oxford University Press, v. 20, n. 4, p. 364–366, 1977. DOI: 10.1093/comjnl/20.4.364.

DEHOUCHE, N.; WONGKITRUNGRUENG, A. Facebook live as a direct selling channel, 2018, proceedings of ANZMAC 2018. In: *The 20th Conference of the Australian and New Zealand Marketing Academy*. Adelaide (Australia): [s.n.], 2018.

DUNN, J. C. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, Taylor & Francis, v. 4, n. 1, p. 95–104, 1974. DOI: 10.1080/01969727408546059.

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.

FISHER, B.; TAŞ, K. The convolution of functions and distributions. *Journal of mathematical analysis and applications*, Elsevier, v. 306, n. 1, p. 364–374, 2005. DOI:10.1016/j.jmaa.2005.01.004.

FISHER, R. A.; MARSHALL, M. Iris data set. *UC Irvine Machine Learning Repository*, v. 440, 1936.

FLEISS, J. L. Measuring nominal scale agreement among many raters. *Psychological bulletin*, American Psychological Association, v. 76, n. 5, p. 378, 1971. DOI: 10.1037/h0031619.

FORINA, M.; ARMANINO, C.; LANTERI, S.; TISCORNIA, E. Classification of olive oils from their fatty acid composition. In: LONDON: APPLIED SCIENCE PUBLISHERS, 1983. *Food research and data analysis: proceedings from the IUFoST Symposium, September 20-23, 1982, Oslo, Norway/edited by H. Martens and H. Russwurm, Jr.* [S.l.], 1983.

FORINA, M.; LEARDI, R.; ARMANINO, C.; LANTERI, S.; CONTI, P.; PRINCI, P. Parvus: An extendable package of programs for data exploration, classification and correlation. *Journal of Chemometrics*, v. 4, n. 2, p. 191–193, 1988.

GAMER, M.; LEMON, J.; PUSPENDRA, I. F. *irr: Various Coefficients of Interrater Reliability and Agreement*. Singh ¡puspendra.pusp22@gmail.com, 2019. R package version 0.84.1. Disponível em: ⟨https://CRAN.R-project.org/package=irr⟩.

GAN, G.; MA, C.; WU, J. *Data clustering: theory, algorithms, and applications*. [S.l.]: SIAM, 2007. ISBN 978-0-898716-23-8.

GRAMACKI, A. *Nonparametric kernel density estimation and its computational aspects*. [S.l.]: Springer, 2018. DOI: 10.1007/978-3-319-71688-6. ISBN 978-3-319-71687-9.

HAHSLER, M.; PIEKENBROCK, M.; DORAN, D. dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, v. 91, n. 1, p. 1–30, 2019. DOI: 10.18637/jss.v091.i01.

HAN, J.; KAMBER, M.; PEI, J. *Data mining concepts and techniques third edition*. [S.l.: s.n.], 2011. 83–124 p. ISBN 978-0-12-381479-1.

HäRDLE, W. K.; SIMAR, L. *Applied Multivariate Statistical Analysis*. [S.l.]: Springer Berlin Heidelberg, 2015. ISBN 978-3-662-45170-0.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer Science & Business Media, 2009. ISBN 978-0387848570.

HUANG, H.; DING, C.; LUO, D.; LI, T. Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*. [S.l.: s.n.], 2008. p. 327–335. DOI: 10.1145/1401890.1401933.

IZMAILOV, A.; SOLODOV, M. *Otimização, volume 2: métodos computacionais*. 2. ed. Rio de Janeiro: IMPA, 2012. v. 2. ISBN 978-85-244-0268-5.

IZMAILOV, A.; SOLODOV, M. *Otimização, volume 1: condições de otimalidade, elementos de análise convexa e de dualidade*. 3. ed. Rio de Janeiro: Impa, 2014. v. 1. ISBN 978-85-244-0389-7.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. *An Introduction to Statistical Learning*. [S.l.]: Springer New York, 2013. DOI:10.1007/978-1-4614-7138-7.

KASSAMBARA, A. *Practical guide to cluster analysis in R: unsupervised machine learning*. [S.l.]: STHDA, 2017. v. 1. ISBN 978-1542462709.

KAUFMAN, L.; ROUSSEEUW, P. J. Clustering by means of medoids. statistical data analysis based on the l1 norm. *Y. Dodge, Ed*, p. 405–416, 1987.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. [S.l.]: John Wiley & Sons, 2009. v. 344.

KRÜGER, F. *Activity, Context, and Plan Recognition with Computational Causal Behaviour Models*. Tese (Doutorado) — University, 2016.

KULCZYCKI, P.; CHARYTANOWICZ, M. A complete gradient clustering algorithm formed with kernel estimators. *International Journal of Applied Mathematics and Computer Science*, Sciendo, v. 20, n. 1, p. 123–134, 2010. DOI: 10.2478/v10006-010-0009-3. Disponível em: ⟨https://doi.org/10.2478/v10006-010-0009-3⟩.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. *biometrics*, JSTOR, p. 159–174, 1977. DOI: 10.2307/2529310.

LIU, B.; XIA, Y.; YU, P. S. Clustering through decision tree construction. In: *Proceedings of the ninth international conference on Information and knowledge management.* [S.l.: s.n.], 2000. p. 20–29. DOI:10.1145/354756.354775.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability.* [S.l.], 1967. v. 1, n. 14, p. 281–297.

MAECHLER, M.; ROUSSEEUW, P.; STRUYF, A.; HUBERT, M.; HORNIK, K. *cluster: Cluster Analysis Basics and Extensions.* [S.l.], 2019. R package version 2.1.0 — For new features, see the 'Changelog' file (in the package source).

MAGNUS, J. R.; NEUDECKER, H. *Matrix differential calculus with applications in statistics and econometrics.* Third edition. [S.l.]: John Wiley & Sons, 2019. ISBN 0-471-98632-1.

MAIMON, O. Z.; ROKACH, L. *Data mining with decision trees: theory and applications.* [S.l.]: World scientific, 2014. v. 81. ISBN-10 981-277-171-9.

MATIOLI, L. C.; SANTOS, S.; KLEINA, M.; LEITE, E. A. A new algorithm for clustering based on kernel density estimation. *Journal of Applied Statistics*, Informa UK Limited, v. 45, n. 2, p. 347–366, jan. 2017. DOI: 10.1080/02664763.2016.1277191.

MCNEIL, D. R. Interactive data analysis. *New York: Wiley*, 1977.

MENARDI, G.; AZZALINI, A. An advancement in clustering via nonparametric density estimation. *Statistics and Computing*, Springer, v. 24, n. 5, p. 753–767, 2014. DOI: 10.1007/s11222-013-9400-x.

MORETTIN, P. A.; BUSSAB, W. O. *Estatística básica.* 9. ed. São Paulo: Saraiva Educação SA, 2017. ISBN 978-85-472-2022-8.

MOUNT, J.; ZUMEL, N. *Practical data science with R.* [S.l.]: Simon and Schuster, 2019. ISBN 978-1-617-29587-4.

NGUYEN, Q. H.; RAYWARD-SMITH, V. J. Internal quality measures for clustering in metric spaces. *International Journal of Business Intelligence and Data Mining*, Inderscience Publishers, v. 3, n. 1, p. 4–29, 2008. DOI: 10.1504/IJBIDM.2008.017973.

NOCEDAL, J. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, v. 35, n. 151, p. 773–782, 1980. DOI: 10.1090/S0025-5718-1980-0572855-7.

NOCEDAL, J.; WRIGHT, S. *Numerical optimization springer-verlag.* [S.l.: s.n.], 1999. ISBN 0-387-98793-2.

R Core Team. *R: A Language and Environment for Statistical Computing.* Vienna, Austria, 2019. Disponível em: ⟨https://www.R-project.org/⟩.

RACE, S. L. *Iterative consensus clustering.* Tese (Doutorado), 2014. 153 f. (Doutorado em Filosofia) – Faculdade de Graduação da Universidade Estadual da Carolina do Norte, 2014. Disponível em: https://repository.lib.ncsu.edu/handle/1840.16/9251. Acesso em : 17 de jan 2021.

RIBEIRO, A. A.; KARAS, E. W. *Otimização Contínua: Aspectos teóricos e computacionais*. São Paulo: Cengage Learning, 2013. ISBN 978-85-221-1501-3.

RODRIGUEZ, A.; LAIO, A. Clustering by fast search and find of density peaks. *Science*, American Association for the Advancement of Science, v. 344, n. 6191, p. 1492–1496, 2014. DOI: 10.1126/science.1242072.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987. DOI: 10.1016/0377-0427(87)90125-7.

SAID, A.; TORRA, V. *Data Science in Practice*. [S.l.]: Springer, 2019. ISBN 978-3-319-97555-9.

SCALDELAI, D.; MATIOLI, L. C.; SANTOS, S. R.; KLEINA, M. Multiclusterkde: a new algorithm for clustering based on multivariate kernel density estimation. *Journal of Applied Statistics*, Informa UK Limited, p. 1–24, 2020. DOI: 10.1080/02664763.2020.1799958.

SCOTT, D. W. *Multivariate density estimation: theory, practice, and visualization*. 2. ed. [S.l.]: John Wiley & Sons, 2015. ISBN 978-0-471-69755-8.

SCRUCCA, L.; FOP, M.; MURPHY, T. B.; RAFTERY, A. E. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, v. 8, n. 1, p. 289–317, 2016. DOI: 10.32614/RJ-2016-021.

SHAH, G. H.; BHENSDADIA, C.; GANATRA, A. P. An empirical evaluation of density-based clustering techniques. *International Journal of Soft Computing and Engineering (IJSCE)*, Citeseer, v. 2, p. 216–223, 2012. ISSN 2231-2307.

SIBSON, R. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, Oxford University Press, v. 16, n. 1, p. 30–34, 1973. DOI: 10.1093/comjnl/16.1.30.

SILVERMAN, B. W. *Density estimation for statistics and data analysis*. [S.l.]: CRC press, 1986. v. 26. ISBN 0-412-24620-1.

SUN, W. W.; LI, L. Dynamic tensor clustering. *Journal of the American Statistical Association*, Taylor & Francis, v. 114, n. 528, p. 1894–1907, mar. 2019. DOI: 10.1080/01621459.2018.1527701.

TAN, S. C.; LAU, J. P. S. Time series clustering: A superior alternative for market basket analysis. In: SPRINGER, SINGAPORE. *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*. [S.l.], 2014. p. 241–248. DOI: 10.1007/978-981-4585-18-7_28.

THARWAT, A. Classification assessment methods. *Applied Computing and Informatics*, Emerald Publishing Limited, 2020. DOI: 10.1016/j.aci.2018.08.003.

WALESIAK, M.; DUDEK, A. *clusterSim: Searching for Optimal Clustering Procedure for a Data Set*. [S.l.], 2019. R package version 0.48-3. Disponível em: ⟨https://CRAN.R-project.org/package=clusterSim⟩.

WAND, M. P.; JONES, M. C. *Kernel smoothing*. [S.l.]: Chapman and Hall/CRC, 1994. ISBN 978-0-412-55270-0.

WANG, W.; YANG, J.; MUNTZ, R. et al. Sting: A statistical information grid approach to spatial data mining. In: *VLDB*. [S.l.: s.n.], 1997. v. 97, p. 186–195.

WU, J.; LIN, Z.; ZHA, H. Essential tensor learning for multi-view spectral clustering. *IEEE Transactions on Image Processing*, IEEE, v. 28, n. 12, p. 5910–5922, 2019. DOI: 10.1109/TIP.2019.2916740.

XIE, J.; GAO, H.; XIE, W.; LIU, X.; GRANT, P. W. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. *Information Sciences*, Elsevier, v. 354, p. 19–40, 2016. DOI: 10.1016/j.ins.2016.03.011.

YÖNTEM, M. K.; ADEM, K.; İLHAN, T.; KILIÇARSLAN, S. Divorce prediction using correlation based feature selection and artificial neural networks. *Nevşehir Hacı Bektaş Veli Üniversitesi SBE Dergisi*, v. 9, n. 1, p. 259–273, 2019.