Trabalho Prático 2

1st João Carlos Ferraz De Sousa
Departamento de Ciências da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
2023087990

2nd Luiz Felipe Gondim Araujo
Departamento de Ciências da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
2023028188

***Abstract*—Este trabalho apresenta uma análise comparativa entre variantes de um algoritmo 2-aproximado para o problema dos k-centros e o método *K*-Means. A avaliação empírica, conduzida com múltiplas métricas de distância e conjuntos de dados reais e sintéticos, investiga o desempenho em termos de qualidade de agrupamento e tempo de execução. Os resultados agregados de repetidas execuções evidenciam as diferenças e ganhos entre as abordagens, destacando sua sensibilidade à geometria dos dados.**

***Index Terms*—Clustering, K-centers, algoritmos, K-Means, análise.**

I. Introdução

O problema de agrupamento (clustering) desempenha um papel central em diversas aplicações de aprendizado de máquina, mineração de dados e reconhecimento de padrões. Em cenários onde o volume de dados é elevado ou onde restrições computacionais são relevantes, soluções exatas tornam-se inviáveis, motivando o uso de algoritmos aproximativos capazes de produzir resultados de boa qualidade com redução significativa de custo computacional. Entre os problemas clássicos de agrupamento, destaca-se o problema dos k-centros, cuja formulação busca selecionar k pontos-centro de forma a minimizar o maior raio necessário para cobrir todas as instâncias do conjunto de dados. Devido à sua natureza NP-difícil, abordagens aproximativas são amplamente utilizadas, com destaque para algoritmos com fator de aproximação 2, estudados tanto em teoria quanto em prática.

Neste trabalho, investigamos empiricamente o desempenho de duas variantes de algoritmos 2-aproximados para o problema dos k-centros: (i) a abordagem baseada em refinamento de intervalos para estimativa do raio ótimo e (ii) a estratégia gulosa de seleção de centros que maximiza a distância entre centros previamente escolhidos. Além disso, comparamos essas abordagens com o algoritmo de agrupamento K-Means, amplamente utilizado como método padrão na literatura e presente na biblioteca Scikit-Learn. A comparação considera tanto a qualidade da solução quanto a demanda computacional, de forma a evidenciar as características de convergência, robustez e sensibilidade dos métodos frente a diferentes tipos de dados.

As análises são conduzidas sob diferentes métricas de distância, uma vez que a escolha da métrica influencia diretamente a geometria percebida dos agrupamentos. Implementamos, a partir de suas definições matemáticas, as distâncias Minkowski (com diferentes valores de $p \geq 1$) e Mahalanobis, permitindo avaliar como a forma dos clusters (circulares, elípticos, sobrepostos, ruidosos, anisotrópicos) afeta o desempenho dos algoritmos. A experimentação utiliza 10 conjuntos de dados reais numéricos do UCI Machine Learning Repository e 50 conjuntos sintéticos gerados por técnicas clássicas: distribuições artificiais disponíveis na documentação do Scikit-Learn e amostras provenientes de distribuições normais multivariadas com diferentes graus de separação e alongamento.

Cada experimento é repetido 15 vezes por algoritmo, registrando-se medidas clássicas de avaliação de agrupamento, como o coeficiente de silhueta e o índice de Rand ajustado, além do raio resultante da solução e do tempo de execução. Resultados são agregados em termos de médias e desvios-padrão, permitindo comparações robustas entre métodos, métricas e características dos dados. Por fim, discutimos padrões empíricos observados, como o impacto da métrica Euclidiana em clusters não esféricos e a relação entre o refinamento do intervalo do raio e a qualidade final da solução.

II. Preparação dos conjuntos de dados

A preparação do conjunto de dados foi realizada em múltiplas etapas, envolvendo a leitura dos arquivos, tratamento de inconsistências, seleção das colunas relevantes e conversão dos dados para um formato adequado ao processamento posterior. Nesta seção, descrevemos o procedimento geral adotado, destacando pontos onde decisões específicas foram tomadas para o tratamento de dados.

A. Criação e escolha dos conjuntos de dados usados

Dados Reais: Para complementar a avaliação com dados do mundo real, foram selecionados dez conjuntos de dados representativos do repositório UCI Machine Learning Repository, abrangendo diversos domínios de aplicação e desafios característicos de problemas reais de mineração

de dados. A seleção criteriosa buscou incluir datasets com variadas características dimensionais, tamanhos amostrais, tipos de atributos e complexidades inerentes, proporcionando assim uma base robusta para a avaliação dos algoritmos de clusterização em condições autênticas.

O conjunto Electrical Grid Stability Simulated Data representa simulações de estabilidade em redes elétricas, contendo atributos numéricos relacionados a parâmetros físicos do sistema. O MAGIC Gamma Telescope consiste em dados de simulação de telescópios de raios gama para classificação de partículas cósmicas, apresentando uma natureza física de alta energia. O Wine Quality aborda a avaliação sensorial de vinhos com base em características físico-químicas, representando um problema típico de controle de qualidade na indústria alimentícia.

O dataset Letter Recognition contém características estatísticas de imagens de letras impressas, desafiando os algoritmos a reconhecer padrões em dados de reconhecimento óptico de caracteres. O Iranian Churn modela o comportamento de clientes de telecomunicações no Irã, refletindo problemas típicos de negócios e marketing. O Hepatitis C Virus (HCV) for Egyptian patients oferece dados clínicos e laboratoriais de pacientes, representando o domínio médico com sua complexidade inerente e importância crítica.

O conjunto HTRU2 caracteriza pulsares celestes detectados por radiotelescópios, apresentando desafios de classificação em astronomia. O Room Occupancy Estimation contém dados de sensores ambientais para detecção de ocupação em salas, representando aplicações de Internet das Coisas. O Website Phishing aborda a identificação de sites fraudulentos, crucial para segurança digital. Finalmente, o Blood Transfusion Service Center modela o comportamento de doadores de sangue, representando problemas do terceiro setor.

Para o tratamento desses dados, inicialmente, foi realizada uma análise exploratória completa para identificar valores ausentes, outliers e inconsistências. Para lidar com dados faltantes, adotou-se estratégias apropriadas ao contexto de cada dataset, incluindo imputação por média ou mediana para variáveis numéricas e moda para categóricas, ou exclusão de instâncias quando a quantidade de valores missing era insignificante.

As variáveis categóricas foram submetidas a processos de codificação adequados, utilizando one-hot encoding para atributos nominais e label encoding para ordinais quando aplicável. Em todos os casos, as variáveis de destino foram separadas do conjunto de features, mantendo apenas os atributos preditivos para a análise de clusterização.

Um desafio significativo encontrado foi a heterogeneidade das escalas dos atributos entre os diferentes datasets. Para garantir comparabilidade entre os algoritmos, todos os conjuntos foram submetidos a padronização utilizando a técnica Z-score, que transforma os dados para apresentarem média zero e desvio padrão unitário. Esta etapa foi crucial para evitar que variáveis com escalas maiores dominassem inadvertidamente o processo de clusterização.

A diversidade destes datasets reais - variando em dimensionalidade (de 4 a 10 features), tamanho amostral (de 1385 a 20000 instâncias) e domínios de aplicação - proporciona um teste abrangente para os algoritmos de clusterização, avaliando sua performance em condições que refletem desafios genuínos encontrados em aplicações práticas reais.

Dados Sintéticos 1: Para conduzir uma avaliação abrangente dos algoritmos de agrupamento, foi desenvolvida uma metodologia sistemática de geração de dados sintéticos que abrange uma ampla variedade de cenários e desafios típicos encontrados em problemas reais de clusterização. A abordagem adotada permitiu a criação de seis conjuntos de dados distintos, cada um projetado para testar capacidades específicas dos algoritmos sob condições controladas e reproduzíveis.

O processo de geração utilizou funções especializadas da biblioteca scikit-learn, configuradas com parâmetros específicos para produzir estruturas de dados com características bem definidas. Foram geradas por volta de 1000 amostras para cada conjunto, um tamanho balanceado que permite observar a escalabilidade dos algoritmos sem comprometer o tempo de processamento. A semente aleatória foi fixada em 30 para garantir a reprodutibilidade dos experimentos.

O primeiro conjunto de dados, denominado ”círculos ruidosos”, foi concebido para simular estruturas concêntricas não lineares. Através da função make_circles, foram gerados dois círculos concêntricos com fator de escala de 0.5 entre os raios, acrescidos de ruído Gaussiano com desvio padrão de 0.05. Esta configuração apresenta um desafio significativo para algoritmos baseados exclusivamente em distâncias euclidianas, uma vez que os clusters não são linearmente separáveis.

O segundo conjunto, ”luas ruidosas”, reproduz duas formações semicirculares interligadas através da função make_moons, com mesmo nível de ruído aplicado aos círculos. Esta estrutura testa a capacidade dos algoritmos de identificar agrupamentos com morfologias complexas e não convexas, simulando cenários onde os clusters possuem formas intricadas que não se adequam a modelos esféricos simples.

Para estabelecer uma baseline de performance, foi gerado um conjunto de ”blobs” isotrópicos através da função make_blobs, que produz clusters Gaussianos esféricos com

variâncias unitárias. Este conjunto representa o caso ideal para algoritmos baseados em centróides, como K-means, servindo como referência para o desempenho máximo esperado em condições ótimas.

Como controle negativo, um conjunto ”sem estrutura” foi criado através de uma distribuição uniforme no espaço bidimensional. Esta configuração permite avaliar a propensão dos algoritmos a detectar falsos positivos, ou seja, identificar clusters onde efetivamente não existem agrupamentos naturais.

Para examinar a sensibilidade dos algoritmos a anisotropias, um conjunto de dados elípticos foi gerado através da aplicação de uma transformação linear matricial [[0.6, -0.6], [-0.4, 0.8]] sobre blobs isotrópicos. Esta transformação produz clusters alongados com orientações preferenciais, simulando situações onde a distribuição dos dados apresenta direcionalidade específica.

Finalmente, um conjunto de ”variâncias variadas” foi criado com a função make_blobs configurada com desvios padrão diferenciados [1.0, 2.5, 0.5] para cada cluster. Esta abordagem permite testar a robustez dos algoritmos frente a agrupamentos com diferentes densidades e dispersões, um desafio comum em dados reais onde os clusters podem exibir heterogeneidade interna.

Todos os conjuntos de dados foram submetidos a um processo de padronização utilizando StandardScaler, que transforma os dados para apresentarem média zero e variância unitária. Esta etapa é crucial para eliminar viés de escala e permitir comparações equitativas entre algoritmos com diferentes pressupostos sobre a distribuição dos dados.

A diversidade intencional destes conjuntos sintéticos proporciona um ambiente de teste abrangente, capaz de avaliar múltiplas dimensões do desempenho dos algoritmos, incluindo sua capacidade de lidar com ruído, adaptar-se a formas não convencionais, responder a diferentes densidades e orientações, e manter especificidade adequada na ausência de estrutura real. Esta metodologia constitui uma base sólida para a avaliação comparativa de algoritmos de clusterização em condições controladas que mimetizam desafios do mundo real.

Dados Sintéticos 2: Para a geração desses conjuntos de dados sintéticos foi empregada uma abordagem baseada em amostragem de pontos em duas dimensões usando a distribuição normal multivariada. Cada conjunto é composto por k agrupamentos (clusters), onde cada cluster i é modelado por

$$X_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i),$$

sendo $\boldsymbol{\mu}_i \in \mathbb{R}^2$ o centro do cluster e Σ_i a matriz de covariância que controla sua forma, orientação e dispersão.

As médias $\boldsymbol{\mu}_i$ foram definidas de modo a permitir diferentes níveis de separação entre clusters, variando desde cenários sem sobreposição até casos altamente sobrepostos. O grau de separação foi controlado pela distância de Mahalanobis entre médias:

$$D_M(\mu_i, \mu_j) = \sqrt{(\mu_i - \mu_j)^\top \Sigma^{-1} (\mu_i - \mu_j)},$$

ajustada de forma a atender aos níveis desejados de sobreposição.

A forma dos agrupamentos foi controlada pela matriz de covariância, construída como

$$\Sigma_i = R(\theta_i) \begin{bmatrix} \sigma_{i,1}^2 & 0 \\ 0 & \sigma_{i,2}^2 \end{bmatrix} R(\theta_i)^\top,$$

onde $R(\theta_i)$ é uma matriz de rotação. Dessa forma, foi possível gerar clusters circulares (quando $\sigma_{i,1} = \sigma_{i,2}$) ou alongados/ elípticos (quando $\sigma_{i,1} \neq \sigma_{i,2}$), inclusive com diferentes orientações no plano.

Para cada conjunto, foram gerados pelo menos $N_{\min}$ exemplos, distribuídos entre os k clusters. Os parâmetros $(\boldsymbol{\mu}_i, \Sigma_i)$ foram ajustados para criar contextos variados de separabilidade, densidade e formato, garantindo diversidade entre os conjuntos utilizados nos experimentos.

B. Limpeza e Tratamento Inicial

Após a leitura, foi realizado um processo de limpeza para garantir consistência e removê-los erros estruturais. As principais ações aplicadas foram:

- remoção de colunas irrelevantes ou com muita disparidade de valor númerico em relação as demais;
- globalizar o uso de virgula para separar colunas;
- padronização de valores categóricos, convertendo rótulos para uma forma numérica;
- tratamento de valores faltantes.

Essas medidas foram muito usadas principalmente para tratar os dados reais coletados, pois o modelo original deles era muito diverso.

C. Conversão da Coluna de Rótulo

Quando presente, a última coluna foi tratada separadamente para garantir consistência no formato de rótulo. Em datasets onde o rótulo era representado por texto (como “stable”/“unstable” ou letras), este foi convertido para valores numéricos. Por exemplo, no dataset electrical, os valores ’stable’ foram trasnformados em 1 e ’unstable’ em 0. No dataset letter-recognition, as letras maiúsculas foram convertidas para suas posições no alfabeto, como A=1, B=2.

D. Geração dos Arquivos Finais

Após todas as etapas de preparação, os arquivos foram exportados no formato `.csv`, substituindo vírgulas e corrigindo automaticamente qualquer inconsistência de espaçamento. Todos os arquivos resultantes foram armazenados em um diretório unificado para facilitar futuras

análises.

III. Implementação

A aplicação foi estruturada como um pipeline modular composto por quatro etapas principais: leitura e preparação dos dados, cálculo das distâncias, execução dos algoritmos de agrupamento e rotinas de sumarização e agregação dos resultados. Essa organização permite isolar responsabilidades, facilitar reuso e manter a consistência entre todos os experimentos realizados, especialmente diante do grande número de execuções exigido pelo trabalho.

A. Leitura e preparação dos dados

A etapa inicial consiste na rotina de leitura dos arquivos CSV, projetada para lidar com datasets sintéticos e reais que podem apresentar variações estruturais significativas. O leitor tenta inicialmente carregar o arquivo assumindo a presença de cabeçalho; caso encontre colunas não numéricas nas posições que deveriam representar atributos, a função relê o arquivo sem cabeçalho. Essa estratégia evita falhas em bases heterogêneas e elimina dependência explícita de formatação prévia.

Os rótulos, presentes na última coluna, são convertidos para inteiros por meio de fatorização automática sempre que necessário. Essa decisão garante que algoritmos de avaliação supervisionada, como o índice de Rand ajustado, recebam um vetor de classes bem definido e consistente. Além disso, todos os atributos são convertidos para ponto flutuante imediatamente após a leitura, impondo uniformidade numérica e evitando erros posteriores durante o cálculo de distâncias ou operações vetorizadas. Como resultado, qualquer dataset contendo apenas valores numéricos e um label final pode ser processado sem intervenções adicionais.

B. Cálculo das distâncias

O módulo de distâncias foi implementado de forma vetorizada para maximizar eficiência computacional. A métrica de Minkowski utiliza operações de *broadcasting* para construir implicitamente o tensor tridimensional contendo todas as diferenças par-a-par entre pontos; em seguida, aplica a potência p, soma ao longo das dimensões espaciais e calcula a raiz correspondente. Essa abordagem evita laços explícitos em Python e garante desempenho próximo ao de bibliotecas otimizadas.

A distância de Mahalanobis segue a formulação clássica baseada na matriz de covariância dos dados. Como alguns datasets podem gerar matrizes singulares ou mal condicionadas, aplica-se uma regularização mínima antes da inversão. A construção da matriz completa de distâncias é feita de forma explícita, ponto a ponto, o que mantém fidelidade ao cálculo quadrático $(x-y)^\top S^{-1}(x-y)$. Ambas as matrizes são pré-calculadas antes da execução dos algoritmos, evitando recomputações dispendiosas e garantindo condições idênticas de comparação entre diferentes algoritmos e repetições.

C. Algoritmos de clustering

A camada de algoritmos implementa três métodos distintos: *farthest-first*, *interval refinement* e K-Means.

O algoritmo *farthest-first* seleciona centros de maneira gulosa, sempre escolhendo o ponto mais distante do conjunto atual de centros. Essa escolha utiliza diretamente a matriz de distâncias pré-computada, dispensando cálculos adicionais e reduzindo substancialmente o custo de cada iteração. A atribuição dos pontos aos centros e o cálculo do raio final são realizados por uma rotina unificada e reutilizável.

Já o algoritmo *interval refinement* segue o procedimento aproximado clássico para o problema de k-center. Ele realiza uma busca binária sobre o valor do raio ótimo e, para cada valor intermediário, aplica um teste de viabilidade baseado em coberturas sucessivas com bolas de raio $2r$. Essa rotina foi implementada sem depender de estruturas externas, permitindo controle total sobre a política de aleatoriedade, o critério de parada e a estratégia de remoção de pontos cobertos.

O K-Means é incluído como algoritmo de comparação. Utiliza-se a implementação do `scikit-learn` para otimizar os centros, mas toda a avaliação subsequente, incluindo o cálculo do raio, é feita com a mesma rotina utilizada pelos demais métodos, preservando homogeneidade entre as métricas registradas nos experimentos.

D. Execução controlada dos experimentos

Toda a lógica de execução é centralizada na função `run_experiments`. Ela determina automaticamente o número de clusters k a partir do vetor de rótulos do dataset, pré-calcula todas as matrizes de distância relevantes (para diferentes valores de p e para Mahalanobis, quando solicitado) e então executa cada algoritmo quinze vezes sob diferentes sementes aleatórias.

Essa abordagem garante que todas as comparações sejam feitas sob condições controladas, uma vez que cada repetição recebe uma semente aleatória independente e todos os algoritmos operam sobre a mesma matriz de distâncias pré-computada, evitando vieses decorrentes de recomputações distintas. Ademais, cada execução registra informações completas sobre tempo, raio e métricas de qualidade, assegurando que os resultados produzidos possam ser posteriormente sumarizados e comparados de forma consistente.

A função de execução individual, `run_experiment_once`, encapsula o cálculo de rótulos,

raio, tempo de execução e métricas supervisionadas (silhueta e Rand ajustado). O procedimento ignora erros silenciosamente em casos degenerados, garantindo que os experimentos nunca sejam interrompidos por exceções pontuais.

E. Sumarização e agregação dos resultados

Com o volume total de execuções (50 datasets, múltiplas métricas de distância, três algoritmos e dezenas de repetições), tornou-se essencial implementar um mecanismo adequado de consolidação das métricas. A função `summarize_results_to_csv` recebe os resultados brutos de um único dataset e produz uma tabela resumida contendo médias e desvios-padrão do raio, índice de silhueta, Rand ajustado e tempo. Cada configuração experimental gera exatamente uma linha, permitindo análises diretas entre algoritmos e métricas.

Para a visão global do desempenho dos algoritmos, a função `aggregate_all_summaries` percorre todos os arquivos de resumo da pasta de resultados e agrupa as linhas por algoritmo, métrica e parâmetros correspondentes. Em seguida, calcula estatísticas agregadas entre todos os datasets, produzindo uma tabela única que sintetiza o comportamento médio dos métodos ao longo de todo o conjunto de experimentos. Essa tabela é especialmente útil para análise comparativa e para a elaboração das conclusões do trabalho. Lembrando que para cada tipo de conjunto de dados (reais, sintéticos 1 e 2) há uma agregação dessa.

Em conjunto, essas decisões de implementação fornecem um pipeline completo, modular e eficiente, capaz de processar múltiplos datasets, executar centenas de experimentos reprodutíveis e produzir resultados comparáveis de forma automatizada e organizada.

IV. Instruções de instalação e execução

Para executar a aplicação efetue os seguintes passos para instalação em seu computador (Linux):

- Instale a virtualenv no seu sistema caso nao tenha: apt install python3.10-venv;
- Crie uma virtualenv na pasta raiz: python3 -m venv env;
- Ative a env: source env/bin/activate;
- Em seguida, instale as dependências: pip install -r requirements.txt;
- Certifique-se que os conjuntos de dados estejam na pasta data, separados em real, synthetic_1 e synthetic_2;
- Inicie a aplicação: python3 src/evaluation/run_csv.py;
- Espere a execução e depois será possivel visualizar o resultado da análise na pasta results, onde para cada tipo de conjunto de dados, terá o sumário de cada arquivo .csv e depois uma analise geral.

V. Análise de complexidade

Foi feita a análise de complexidade temporal e espacial das principais funções implementadas no programa, sendo elas para cálculo de distâncias, rotinas auxiliares, algoritmos aproximados de k-center, rotinas de execução, leitura e sumarização dos resultados da análise dos algoritmos.

a) minkowski_distance_matrix: A função calcula todas as distâncias pareadas entre n pontos em d dimensões utilizando a formulação vetorizada $|x_i - x_j|^p$. A operação de broadcasting gera um tensor de dimensão $n \times n \times d$ e a redução ao longo da dimensão espacial exige $\Theta(n^2 d)$ operações elementares (potenciação, soma e radiciação). Logo, a complexidade temporal assintótica é $\Theta(n^2 d)$ e a complexidade espacial é $\Theta(n^2)$ para armazenar a matriz de distâncias resultante.

b) mahalanobis_distance_matrix: A função executa três etapas dominantes, a primeira realiza o cálculo da matriz de covariância com custo $\Theta(nd^2)$; depois faz a inversão da matriz de covariância regularizada com custo $\Theta(d^3)$; e, por fim, a avaliação das n^2 distâncias pareadas, cada qual envolvendo produtos quadráticos de dimensão d. Na implementação apresentada, a última etapa conduz a um custo assintótico $\Theta(n^2 d^2)$ no pior caso. Portanto, a complexidade temporal dominante é $\Theta(n^2 d^2)$ e a complexidade espacial principal permanece $\Theta(n^2)$ devido à matriz de distâncias.

c) assign_labels_from_centers: A rotina atribui n pontos ao centro mais próximo entre k centros, calculando distâncias ponto–centro via broadcasting, o que constrói um tensor de dimensão $n \times k \times d$. A computação das distâncias custa $\Theta(nkd)$, a operação de $\arg\min$ custa $\Theta(nk)$ e o cálculo do raio (máxima distância dentro de cada grupo) custa $\Theta(n)$. Assim, a complexidade temporal total é $\Theta(nkd)$ e a memória temporária requerida é $\Theta(nk)$.

d) kcenter_farthest_first: O algoritmo guloso, assumindo disponibilidade prévia da matriz de distâncias D, executa k iterações nas quais cada passo escolhe o vértice mais afastado por uma operação $\arg\max$ sobre um vetor de tamanho n e atualiza um vetor de distâncias mínimas em custo $\Theta(n)$, isto fornece $\Theta(kn)$ para a fase de seleção de centros. A chamada final a `assign_labels_from_centers` adiciona custo $\Theta(nkd)$. Portanto, com D pré-calculada o custo dominante resulta em $\Theta(nkd)$, se a matriz D for computada internamente soma-se $\Theta(n^2 d)$.

e) kcenter_interval_refinement: A rotina realiza uma busca binária sobre o espaço de raios com número de iterações da ordem de $O\big(\log(1/\text{width\_frac})\big)$. Em cada iteração, a subrotina `feasible(r)` realiza uma cobertura gulosa que, no pior caso, pode selecionar até k centros e verificar, para cada centro, os pontos a remover, resultando em $\Theta(kn)$ por iteração. Assim, o custo da busca binária é $\Theta(kn \log(1/\text{width\_frac}))$. Somando a etapa final de

atribuição de rótulos $\Theta(nkd)$ obtemos a complexidade total $\Theta\big(nk(d+\log(1/\text{width\_frac}))\big)$ assumindo matriz de distâncias pré-calculada, e adicionando $\Theta(n^2 d)$ caso tal matriz precise ser obtida internamente.

f) run_experiment_once: Essa função encapsula a execução de um único algoritmo (farthest, interval ou KMeans) e a computação de métricas de qualidade. O custo da chamada é dominado pelo algoritmo subjacente escolhido, adicionalmente, a avaliação da silhueta com a implementação padrão impõe um custo $\Theta(n^2 d)$ em tempo pela necessidade implícita de distâncias entre pares, enquanto o cálculo do Adjusted Rand Index é $\Theta(n)$. Logo, a complexidade assintótica desta função é $\Theta(n^2 d)$, dominada pela avaliação da silhueta quando realizada.

g) run_experiments: Aqui o pré-cálculo das distâncias Minkowski custa $\Theta(|P|\, n^2 d)$, onde $|P|$ é o número de valores de p e a pré-computação de Mahalanobis pode custar até $\Theta(n^2 d^2)$. Para cada repetição, o custo dominante é $\Theta(n^2 d)$ (devido à silhueta ou ao próprio algoritmo), de modo que, denotando por A o número total de algoritmos considerados e por W o número de valores de `width_frac`, o custo agregado pode ser escrito como $\Theta\big(|P|\, n^2 d + R(A + AW)\, n^2 d\big)$; a memória é dominada pelo armazenamento das matrizes de distância com custo $\Theta(|P| n^2)$.

h) summarize_results_to_csv: Para cada experimento recebido como parâmetro, percorre-se uma lista de R execuções, extraindo e filtrando vetores de métricas de tamanho no máximo R, sobre eles, calculam-se média e desvio padrão, ambos com custo $\Theta(R)$. Assim, dado E experimentos distintos, o custo total é $\Theta(ER)$. A escrita do arquivo CSV é linear no número de linhas geradas, isto é, $\Theta(E)$. A memória adicional requerida é $\Theta(E)$, correspondente ao armazenamento temporário das linhas agregadas.

i) aggregate_all_summaries: A função lê todos os arquivos de resumo provenientes de diferentes datasets, agrupando-os por chave $(\text{algoritmo}, \text{métrica}, p, \text{width\_frac})$. Seja D o número de arquivos de resumo e E o número de experimentos por arquivo, o custo de leitura total é $\Theta(DE)$. Cada agrupamento agrega as métricas por meio de médias sobre listas de tamanho igual ao número de datasets associados à chave, resultando em custo adicional $\Theta(G \cdot D)$, onde G é o número de grupos distintos. A escrita final do CSV é $\Theta(G)$. A memória adicional é $\Theta(G + DE)$, dominada pelo dicionário de agregação contendo todas as linhas lidas.

Em conclusão, observa-se que o custo global do sistema é dominado pela computação de distâncias pareadas e pela medida de silhueta, ambas de natureza quadrática (ou pior) em n, enquanto os algoritmos aproximados e as rotinas de sumarização têm custos lineares ou quase lineares nos parâmetros experimentais. Dessa forma, o gargalo principal encontra-se nas operações geométricas de alto custo, ao passo que as etapas de agrupamento e análise estatística têm impacto marginal no tempo total de execução.

VI. Experimentos e resultados

O estudo comparativo realizado entre os três algoritmos de agrupamento, K-means, K-center (Farthest-First) e K-center (Interval Refinement), revelou padrões interessantes de desempenho que variam conforme as características dos dados e as métricas de distância utilizadas.

Abaixo está representado nas tabelas o agregado das métricas de cada tipo de conjuntos de dados. Essa forma de apresentação separada dos dados foi escolhida com intuito de evidenciar o desempenho de cada algoritmo a depender do tipo do conjunto de dados. Aliás, foi decidido mostrar apenas os três testes mais relevante com o algoritmo Interval Refinement para evitar poluir o artigo.

Tabela 1: Resultados dos algoritmos em Dados Reais

Algoritmo	Métrica	width_frac	Raio Médio	Silhueta Média	ARI Médio	Tempo Médio (s)
kmeans	minkowski_p1	-	-	0.406	0.060	0.0791
kmeans	minkowski_p2	-	-	0.407	0.047	0.0819
kmeans	mahalanobis	-	-	0.406	0.050	0.0745
farthest	minkowski_p1	-	189.210	0.382	0.042	0.0006
farthest	minkowski_p2	-	120.065	0.425	0.060	0.0007
farthest	mahalanobis	-	150.468	0.337	0.064	0.0006
interval	minkowski_p1	0.05	172.484	0.389	0.049	0.0059
interval	minkowski_p2	0.05	105.370	0.436	0.066	0.0049
interval	mahalanobis	0.01	151.265	0.352	0.058	0.0057

Tabela 2: Resultados dos algoritmos em Dados Sintéticos 1

Algoritmo	Métrica	width_frac	Raio Médio	Silhueta Média	ARI Médio	Tempo Médio (s)
kmeans	minkowski_p1	-	-	0.572	0.655	0.0524
kmeans	minkowski_p2	-	-	0.572	0.655	0.0500
kmeans	mahalanobis	-	-	0.571	0.653	0.0539
farthest	minkowski_p1	-	5.118	0.507	0.564	0.0005
farthest	minkowski_p2	-	3.880	0.523	0.571	0.0006
farthest	mahalanobis	-	4.154	0.505	0.573	0.0006
interval	minkowski_p1	0.05	4.479	0.547	0.601	0.0062
interval	minkowski_p2	0.05	3.326	0.559	0.604	0.0063
interval	mahalanobis	0.01	3.541	0.524	0.620	0.0081

Tabela 3: Resultados dos algoritmos em Dados Sintéticos 2

Algoritmo	Métrica	width_frac	Raio Médio	Silhueta Média	ARI Médio	Tempo Médio (s)
kmeans	minkowski_p1	-	-	0.671	0.759	0.0517
kmeans	minkowski_p2	-	-	0.671	0.759	0.0431
kmeans	mahalanobis	-	-	0.668	0.756	0.0589
farthest	minkowski_p1	-	8.194	0.579	0.602	0.0005
farthest	minkowski_p2	-	6.265	0.598	0.627	0.0006
farthest	mahalanobis	-	7.673	0.534	0.559	0.0007
interval	minkowski_p1	0.05	6.929	0.644	0.684	0.0052
interval	minkowski_p2	0.05	5.234	0.647	0.684	0.0055
interval	mahalanobis	0.01	7.720	0.552	0.576	0.0066

O K-means destacou-se consistentemente como o algoritmo com melhor qualidade de agrupamento, alcançando os valores mais elevados de Silhueta Média e Índice de Rand Ajustado (ARI) na maioria dos cenários analisados. Esta superioridade foi particularmente evidente em conjuntos de dados sintéticos bem separados, onde a estrutura esférica dos clusters se alinha bem com as premissas do algoritmo. No entanto, é importante ressaltar que o K-means apresenta duas limitações significativas: não calcula um raio de cluster, o que restringe sua aplicabilidade em problemas onde essa métrica é crítica, e demonstrou ser consistentemente o algoritmo mais lento, especialmente em conjuntos de dados de maiores dimensões.

Esta combinação de alta qualidade de agrupamento com alto custo computacional posiciona o K-means como a escolha ideal para aplicações onde a precisão é prioritária e o tempo de processamento não é uma restrição severa.

Em contrapartida, o K-center (Farthest-First) emergiu como o algoritmo mais eficiente em termos computacionais, sendo significativamente mais rápido em todos os cenários testados. Esta vantagem em velocidade o torna particularmente atraente para aplicações em tempo real ou para conjuntos de dados muito grandes onde o tempo de processamento é uma preocupação. Além da velocidade, o Farthest-First produziu raios de cluster menores que o Interval Refinement na maioria dos casos, indicando uma compactação mais eficiente dos grupos. No entanto, esta eficiência vem com um custo em termos de qualidade de agrupamento, com valores de ARI e Silhueta geralmente inferiores aos do K-means, embora ainda superiores ao Interval Refinement em configurações com valores elevados de width_frac.

O K-center (Interval Refinement) apresenta uma abordagem intermediária que oferece um controle refinado sobre o raio dos clusters através do parâmetro width_frac. Nossos experimentos demonstraram que valores de width_frac entre 0.05 e 0.1 tipicamente produzem o melhor equilíbrio entre raio do cluster, ARI e Silhueta, representando um ponto ótimo para a maioria das aplicações práticas. Entretanto, quando o width_frac excede 0.15, observa-se um aumento significativo no raio dos clusters acompanhado por uma degradação acentuada na qualidade do agrupamento, sugerindo que valores elevados deste parâmetro devem ser evitados. O tempo de execução do Interval Refinement posiciona-se consistentemente entre o Farthest-First e o K-means, oferecendo um compromisso razoável entre velocidade e controle sobre as características dos clusters.

A análise da influência das métricas de distância revelou padrões distintos de desempenho. A distância Minkowski p=2 (Euclidiana) mostrou-se superior em clusters de formato esférico, alinhando-se com as expectativas teóricas sobre seu comportamento. Já a Minkowski p=1 (Manhattan) demonstrou maior robustez à presença de outliers, mantendo desempenho estável mesmo em condições de dados ruidosos. A métrica Mahalanobis, por sua vez, confirmou sua superioridade em clusters elípticos ou com estruturas de covariância não-esféricas, embora tenha se mostrado sensível à precisão na estimação da matriz de covariância, requerendo amostras suficientemente grandes para estimativas confiáveis.

O comportamento dos algoritmos variou significativamente conforme o tipo de dados analisado. Nos dados reais, o K-means manteve sua superioridade em termos de ARI e Silhueta, enquanto o Interval Refinement com width_frac entre 0.01 e 0.1 produziu os menores raios de cluster. Nos dados sintéticos 1 e 2, o K-means novamente dominou em qualidade de agrupamento, com o Farthest-First oferecendo uma alternativa rápida e razoavelmente eficaz, e o Interval Refinement com width_frac de 0.05 emergindo como a opção mais balanceada quando se considera conjuntamente a qualidade do agrupamento, o raio dos clusters e o tempo de processamento.

Esta análise abrangente demonstra que a seleção do algoritmo ideal depende criticamente dos requisitos específicos da aplicação, do tipo de dados disponíveis e das métricas de desempenho consideradas prioritárias, não existindo uma solução única que se mostre superior em todos os cenários possíveis.

VII. Conclusão

O desenvolvimento deste projeto proporcionou uma compreensão ampla e integrada das diversas etapas envolvidas na análise de algoritmos de agrupamento aproximado. Desde o tratamento inicial dos dados até a execução sistemática dos experimentos, foi necessário construir soluções robustas para leitura de arquivos, padronização de rótulos, normalização de diferentes formatos de entrada e preparação de matrizes de características. Esses cuidados garantiram que tanto datasets sintéticos quanto reais pudessem ser processados de maneira uniforme e confiável.

A implementação dos algoritmos de clustering, incluindo variantes aproximadas de k-center e o método de k-mean, exigiu atenção especial à eficiência computacional e ao uso apropriado de operações vetorizadas, além de decisões arquiteturais que permitissem reutilizar cálculos custosos, como as matrizes de distância. Da mesma forma, o pipeline de experimentação foi estruturado de modo a permitir múltiplas repetições controladas, coleta padronizada de métricas e integração transparente com as rotinas de sumarização.

O módulo de agregação dos resultados desempenhou papel essencial ao possibilitar a consolidação das estatísticas provenientes de cada dataset e a posterior fusão em tabelas globais. Essa etapa evidenciou a importância de um formato coerente de saída e da automação do processo de análise, reduzindo o esforço manual e garantindo consistência entre diferentes execuções.

De maneira geral, o projeto foi extremamente proveitoso para o nosso aprendizado. As dificuldades encontradas, como lidar com datasets heterogêneos, projetar funções genéricas e eficientes, implementar algoritmos não triviais a partir de suas formulações teóricas e estruturar um pipeline experimental completo, contribuíram significativamente para o amadurecimento técnico necessário em tarefas de análise de desempenho de algoritmos. A superação desses desafios consolidou nosso entendimento tanto dos aspectos computacionais quanto metodológicos envolvidos na avaliação de métodos de agrupamento.

AGRADECIMENTOS

Os autores gostariam de agradecer ao professor Renato Vimieiro e ao monitor da disciplina pela elaboração desse trabalho prático, de fato a experiência de elaborar um modelo de artigo ciêntifico é de muitas formas gratificante.

REFERENCES

[1] J. A. Hartigan and M. A. Wong, "A K-Means Clustering Algorithm," Journal of the Royal Statistical Society: Series C (Applied Statistics), vol. 28, no. 1, pp. 100–108, 1979.

[2] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," Theoretical Computer Science, vol. 38, pp. 293–306, 1985.

[3] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53–65, 1987.

[4] L. Hubert and P. Arabie, "Comparing partitions," Journal of Classification, vol. 2, no. 1, pp. 193–218, 1985.

[5] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651–666, 2010.

[6] P. C. Mahalanobis, "On the generalized distance in statistics," Proceedings of the National Institute of Sciences of India, vol. 2, no. 1, pp. 49–55, 1936.

[7] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," Expert Systems with Applications, vol. 40, no. 1, pp. 200–210, 2013.