Trabalho Prático 2 (Algoritmos 2)

1st Bruno Fonseca
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
brunolmf927@gmail.com

2nd Gabriel Alkmim
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
gabrielzalk03@gmail.com

***Abstract*—Este trabalho apresenta uma análise comparativa de algoritmos de agrupamento com foco nos problemas de k-Centros e k-Means. A metodologia inclui a implementação do algoritmo *Greedy Max-Min* (2-aproximação), um procedimento de Δ-Refinamento via busca binária, e o k-Means clássico. Avaliamos o desempenho sob o critério minimax (raio máximo) e inercial (soma dos erros quadráticos), além de contrastar a distância de Mahalanobis com a Euclidiana. Os experimentos em 51 conjuntos de dados (sintéticos e reais da UCI) demonstram que, embora o k-Means ofereça melhor qualidade geral (ARI médio 0.47), o algoritmo Max-Min é, em média, 46 vezes mais rápido, sendo ideal para grandes volumes de dados. O Δ-Refinamento apresentou um *trade-off* claro: $\delta = 0.01$ melhora a qualidade do raio, mas custa 2.5x mais tempo que $\delta = 0.25$. Contrariando a expectativa teórica, a distância de Mahalanobis apresentou desempenho inferior à Euclidiana (ARI 0.29 vs 0.47), devido à instabilidade numérica em altas dimensões e sensibilidade a outliers em dados reais.**

***Index Terms*—K-means, K-center, aproximation algorithms, 2-aproximate**

I. Introducão

O agrupamento (clustering) é uma tarefa fundamental na análise exploratória de dados e aprendizado de máquina não supervisionado, com aplicações que variam desde a segmentação de clientes até a bioinformática . Seu objetivo principal é particionar um conjunto de dados em subconjuntos (clusters) de forma que os elementos dentro de um mesmo grupo sejam mais similares entre si do que em relação aos elementos de outros grupos. A qualidade de tal partição, contudo, depende diretamente do critério de otimização adotado e da métrica de distância utilizada para quantificar a similaridade.

Historicamente, dois problemas centrais e distintos dominam a literatura de agrupamento: o problema k-Centros e o problema k-Means. O problema dos k-Centros busca uma partição que minimize a distância máxima de qualquer ponto a seu centro (critério minimax), sendo mensurado pelo raio máximo $r(C)$. Este critério é crucial em aplicações onde a garantia de serviço e cobertura total são prioritárias, como em logística e alocação de sensores. Por outro lado, o **problema k-Means** (também conhecido como o problema k-médias) visa minimizar a soma dos quadrados das distâncias (erro quadrático médio), um critério inercial que é largamente empregado devido à sua eficiência computacional na prática, embora ambos sejam NP-hard na sua forma geral.

Para o k-Centros, que é notoriamente difícil de otimizar, o algoritmo *Greedy Max-Min* é amplamente reconhecido como uma das mais eficientes 2-aproximações. Neste trabalho, implementamos este algoritmo para estabelecer uma base de qualidade. Complementarmente, exploramos um procedimento de **Δ-Refinamento**, baseado na busca binária no espaço de raios, para otimizar a solução de forma iterativa e investigar o *trade-off* entre precisão da solução e custo computacional, regido pelo parâmetro de tolerância Δ.

Além da escolha do algoritmo, a definição de similaridade desempenha um papel crítico. A distância Euclidiana é a métrica padrão, mas falha em capturar estruturas de clusters não esféricas ou que exibem alta correlação entre as dimensões. Para mitigar essa limitação, este estudo investiga o desempenho da **distância de Mahalanobis**, que utiliza a matriz de covariância dos dados para padronizar as dimensões e considerar a forma geométrica dos clusters. A comparação entre as distâncias Euclidiana e Mahalanobis em dados elípticos demonstra a importância da escolha métrica para a obtenção de partições que reflitam a verdadeira estrutura dos dados.

Desta forma, as principais contribuições deste trabalho incluem:

1) A implementação e comparação empírica do desempenho do k-Centros (Max-Min e Δ-Refinamento) e k-Means em relação ao raio máximo ($r(C)$) e tempo de execução.
2) Uma análise detalhada do impacto do parâmetro Δ no Refinamento, quantificando o *trade-off* entre a qualidade (menor $r(C)$) e o aumento da complexidade temporal.
3) Uma avaliação sistemática da eficácia da métrica de Mahalanobis no agrupamento de dados com covariância não-trivial, utilizando métricas de validação de cluster como o Índice de Silhouette e o Adjusted Rand Index (ARI).

II. Metodologia

A metodologia adotada neste trabalho foi estruturada de forma rigorosa, com o objetivo de garantir reprodutibilidade experimental, reduzir vieses estatísticos e permitir comparação justa entre os algoritmos avaliados. Todo o processo foi organizado em quatro etapas principais:

1) Implementação dos algoritmos estudados;
2) Seleção e geração dos conjuntos de dados;
3) Definição das métricas de distância e avaliação;

4) Execução sistemática e agregação estatística dos experimentos.

Cada uma dessas etapas é detalhada nas subseções seguintes.

A. Implementação dos Algoritmos

Todos os algoritmos foram implementados em Python 3.12 com uso extensivo de operações vetoriais em NumPy, de modo a reduzir o custo computacional de operações repetitivas. A escolha da linguagem se deve à ampla disponibilidade de bibliotecas científicas, bem como à facilidade de reprodução do experimento por terceiros.

Foram avaliados três métodos distintos de agrupamento:

1) K-Means: Implementação iterativa baseada na realocação e recomputação de centróides. O algoritmo interrompe quando a variação média de posição dos centróides fica abaixo de 10^{-4}, garantindo convergência estável. Foram executadas múltiplas inicializações para reduzir dependência estocástica.

2) K-Centers Max-Min: Método guloso com complexidade $O(nk)$, que seleciona centros iterativamente maximizando a menor distância existente ao conjunto já definido. Esse algoritmo não possui refinamento iterativo, o que o torna extremamente rápido, mas com perda de qualidade em cenários mais complexos.

3) K-Centers Refinement: Extensão do método Max-Min com refinamento do raio através de busca binária no intervalo inicial $[0, R]$. O parâmetro de precisão δ controla o número de iterações e a aproximação da solução. Foram testados:

$$\delta \in \{0.01,\ 0.05,\ 0.10,\ 0.15,\ 0.20,\ 0.25\}$$

Valores menores de δ aumentam o custo computacional, porém geram soluções mais próximas do ótimo teórico.

B. Métricas de Distância Avaliadas

A distância entre pontos é fundamental para o processo de agrupamento. Neste estudo, analisou-se o comportamento dos algoritmos sob métricas diferentes, permitindo verificar sua robustez geométrica.

1) Distância de Minkowski:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Três valores foram considerados: $p = 1$ (Manhattan), $p = 2$ (Euclidiana) e $p = 3$ (Norma cúbica). Essas variações permitem estudar a sensibilidade do algoritmo ao alongamento dos eixos nas dimensões dos dados.

2) Distância de Mahalanobis:

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

Onde Σ é a matriz de covariância. Utilizou-se regularização de Tikhonov com $\lambda = 10^{-6}$ para evitar instabilidades numéricas na inversão matricial — problema comum em bases com alta correlação.

C. Conjuntos de Dados

Os experimentos totalizaram 60 bases agrupadas em três categorias gerais: bases reais da UCI, bases sintéticas do scikit-learn e distribuições normais multivariadas geradas artificialmente.

1) Bases Reais (UCI Repository): Conjuntos usados refletem cenários práticos com ruído, classes desbalanceadas e dimensões elevadas. Os mais relevantes estão listados na Tabela I.

TABLE I
BASES REAIS UTILIZADAS NOS EXPERIMENTOS

Dataset	Instâncias	Atributos	k
Banknote	1372	4	2
Wine Red	1599	11	6
Wine White	4898	11	7
Bankruptcy	6819	95	2
SECOM	1567	589	2
DrugConsumption	1885	12	7
Myocardial	1700	123	8
Obesity	2111	16	7
Cardiotocography	2126	36	3
BEED EEG	8000	16	4

2) Bases Sintéticas — Scikit-learn (30): Geradas com parâmetros controlados para avaliar robustez geométrica:

- Blobs (esférico, separável)
- Moons e Circles (não convexos)
- Anisotropic (clusters elípticos)
- Varied e Noisy Circles (variância e ruído elevados)

3) Normais Multivariadas (20): Com parâmetros variando correlação, separação e elongamento dos clusters. Dimensionalidade variou entre 2 e 10 atributos.

D. Protocolo Experimental

Cada combinação `(dataset × algoritmo × métrica × )` foi executada 15 vezes. Foram registradas as seguintes métricas:

- Tempo de execução (s);
- Silhouette Score $(-1 \leq s \leq 1)$;
- ARI – Adjusted Rand Index;
- Raio máximo (somente para K-Centers);
- Número de iterações (K-Means).

A avaliação utilizou funções oficiais do Scikit-learn para garantir padronização e comparabilidade com literatura.

E. Ambiente de Execução

TABLE II
CONFIGURAÇÃO UTILIZADA PARA OS EXPERIMENTOS

Processador	Intel i7–10400F @ 2.9GHz
Memória RAM	16 GB
Sistema Operacional	Windows 10 x64
Ambiente de Execução	Python 3.12 com NumPy

A ausência de GPU permite reprodutibilidade em máquinas domésticas, mantendo validade prática dos resultados obtidos.

III. Implementação

A implementação dos algoritmos foi realizada integralmente em Python 3.12, fazendo uso extensivo de operações vetorizadas via *NumPy*, de forma a minimizar o custo de interpretação do Python e maximizar o aproveitamento das rotinas internas em C/BLAS. Todas as funções de distância utilizadas nos experimentos foram desenvolvidas manualmente, conforme exigido no enunciado do trabalho, sem recorrer às versões já prontas do *scikit-learn* ou SciPy. Essa decisão permitiu observar o impacto real de cada métrica na complexidade computacional, além de possibilitar controle fino sobre o custo de memória e tempo por execução.

Para garantir reprodutibilidade, uma semente fixa foi aplicada para geração dos datasets e para inicializações aleatórias quando necessárias, de modo que cada execução experimental pudesse ser repetida e verificada sob as mesmas condições experimentais.

A. Otimização Vetorial das Distâncias

O cálculo da matriz completa de distâncias foi identificado como o componente de maior custo computacional dentro da pipeline experimental. Um cálculo ingênuo, baseado em dois laços aninhados em Python, resulta em complexidade $O(N^2 \cdot d)$ interpretada, o que torna o processo proibitivamente lento para bases com algumas dezenas de milhares de elementos. Para contornar esse problema, adotou-se uma formulação totalmente vetorizada via *broadcasting*, permitindo que as operações sejam executadas em código nativo (C/BLAS):

$$D_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

Essa expressão foi calculada computacionalmente por meio da construção tridimensional (N, N, d) gerada automaticamente pelo broadcasting, conforme a operação:

$$D = \sqrt{\Big((X[:, None, :] - X[None, :, :])^2\Big).sum(axis = 2)}$$

onde:

- `X[:,None,:]` expande o tensor para dimensão $(N, 1, d)$;
- `X[None,:,:]` expande para $(1, N, d)$, permitindo subtração par-a-par;
- a diferença gera um volume (N, N, d) contendo $(x_i - x_j)$ para todos os pares;
- `sum(axis=2)` computa $\sum_k (x_{ik} - x_{jk})^2$;
- a raiz quadrática finaliza a norma L2 sem qualquer laço explícito.

Essa abordagem evita iterações interpretadas, preserva paralelismo interno e reduz a necessidade de buffers auxiliares. Contudo, o custo de memória cresce como $O(N^2)$, o que implica que para $N = 10\,000$ a matriz resultante demanda cerca de 2.3 GB de RAM — um limitante prático que determina o maior tamanho de dataset processável antes que estratégias alternativas sejam necessárias (cálculo sob demanda, blocos, memória mapeada ou técnicas aproximadas).

Extensão para Mahalanobis. Para a métrica de Mahalanobis, empregou-se:

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

computada com `numpy.einsum`, o que elimina formações intermediárias e melhora o uso de cache. Na prática, observou-se redução aproximada de 20% no consumo de memória transitória quando comparado a uma implementação equivalente composta apenas por multiplicações matriciais explícitas. Em cenários de alta dimensionalidade (d ¿ 50), essa otimização foi crucial para evitar estouro de memória e manter estabilidade numérica durante a inversão de Σ.

B. Estrutura Modular do Código

A implementação foi desenvolvida de forma modular, permitindo a separação clara entre geração de dados, execução de experimentos, cálculo de métricas e análise de resultados. Essa abordagem facilita a reprodutibilidade, amplia a escalabilidade do projeto e possibilita a substituição de módulos individuais sem impacto estrutural no restante do framework. A organização dos arquivos é descrita a seguir:

- `distancias.py` — Implementação direta das métricas de Minkowski (L1, L2 e L3) e Mahalanobis com `numpy.einsum` e regularização da matriz de covariância. Responsável pela parte de Cálculo Numérico.
- `kcenters.py` — Contém as rotinas de Max-Min e *Delta-Refinement* com parâmetros ajustáveis. Inclui controle de parada, refinamento binário do raio e suporte à troca dinâmica de métricas de distância.
- `experimentes.py` — Motor principal de execução. Realiza chamadas parametrizadas aos algoritmos, coleta resultados e gerencia repetições estatísticas (15 execuções por configuração).
- `geracao_dados.py` — Responsável pela criação dos 50 datasets sintéticos usados nos testes, com controle explícito de semente aleatória, distribuição, covariância e geometria dos clusters.
- `analise_resultados.py` — Agrega os resultados em tabelas finais (média, desvio-padrão, raio máximo e tempo), além de permitir exportação para CSV, LaTeX e Markdown.
- `graphs.py / newgraphs.py` — Produção das figuras utilizadas no artigo: heatmaps, sensibilidade ao δ, comparações de métrica, impacto da geometria e escalabilidade temporal.

Essa organização possibilitou automatizar toda a pipeline experimental, desde a geração das bases até a análise estatística final, permitindo que múltiplas execuções fossem realizadas sem intervenção manual. Além disso, a modularização garante que novas métricas ou variantes de K-Centers possam

ser incorporadas futuramente com mínima modificação no restante do código, deixando o framework aberto para extensão acadêmica ou integração com implementações paralelas e GPU.

C. Pseudocódigo da Implementação do Refinement

O Refinement parte de uma solução inicial obtida por Max-Min e aplica busca binária sobre o valor limite do raio permitido, reduzindo-o gradualmente até que a diferença entre limites seja inferior ao valor de precisão δ.

Algorithm 1 Procedimento δ-Refinement para K-Centers

RefinementX, k, δ $(C, R) \leftarrow$ MaxMin(X, k) $lo \leftarrow 0$, $hi \leftarrow R$ $hi - lo > \delta \cdot R$ $mid \leftarrow (hi + lo)/2$ existe solução com raio $\leq mid$ $hi \leftarrow mid$ $lo \leftarrow mid$ (hi, C)

Como mostrado experimentalmente, o método apresenta convergência logarítmica:

$$O(nk \log(1/\delta))$$

deixando explícito o *trade-off* entre tempo e qualidade — quanto menor o valor de δ, maior número de iterações, mas melhor aproximação do raio ótimo.

D. Limitações da Implementação e Extensões Futuras

Embora eficiente, a implementação apresenta limitações inerentes ao cálculo quadrático da matriz de distâncias. Expansões naturais incluem:

- Paralelização com Numba/CUDA para datasets maiores.
- Cálculo de Mahalanobis por cluster em vez de global.
- Redução de memória por *chunking* da matriz $N \times N$.
- Reescrita dos algoritmos críticos em Cython.

Essas melhorias permitiriam escalar para milhões de instâncias, mantendo consumo de memória e tempo sob controle.

IV. Análise dos Resultados

A. Comparação Global entre Algoritmos

O heatmap da Fig. 1 evidencia que K-Means apresenta desempenho superior na maior parte das bases sintéticas e reais, com maior concentração de valores em verde na primeira coluna. Já Max-Min e Refinement mostram comportamento mais estável, porém com ARI inferior, exceto em alguns casos específicos (ex.: Circles e Banknote), nos quais superam o K-Means, possivelmente por escaparem de mínimos locais.

Os resultados agregados (Tabela III) revelam padrões consistentes:

1) Qualidade de Agrupamento:

- **ARI**: K-Means (0.78) > Refinement (0.72) > Max-Min (0.65)
- **Silhueta**: Refinement $(0.62) \approx$ K-Means (0.61) > Max-Min (0.55)

TABLE III
Métricas Agregadas por Algoritmo (média ± desvio padrão)

Métrica	K-Means	Max-Min	Refinement ($\delta = 0.05$)
ARI	0.78 ± 0.12	0.65 ± 0.15	0.72 ± 0.13
Silhueta	0.61 ± 0.08	0.55 ± 0.10	0.62 ± 0.07
Tempo (s)	15.3 ± 8.7	0.33 ± 0.21	4.2 ± 2.1
Raio Máximo	-	2.14 ± 0.45	1.98 ± 0.38

2) Eficiência Computacional:

- **Velocidade**: Max-Min $(0.33s) \gg$ Refinement $(4.2s) >$ K-Means $(15.3s)$
- O Max-Min foi aproximadamente 46× mais rápido que o K-Means

B. Impacto da Métrica de Distância

A análise por métrica de distância revelou resultados contraintuitivos em relação à teoria geométrica. Enquanto as distâncias de Minkowski (L_1, L_2, L_3) apresentaram desempenho similar (diferença média $< 1.5\%$), a distância de Mahalanobis obteve desempenho inferior (ARI médio 0.29 contra 0.47 da Euclidiana).

Apesar de teoricamente superior para clusters elípticos, a Mahalanobis sofreu com:

1) **Instabilidade Numérica:** A inversão da matriz de covariância em datasets com poucas amostras ou alta dimensionalidade (ex: UCI SECOM com 590 features) gerou ruído.
2) **Covariância Global:** O cálculo de uma única matriz Σ para todo o dataset distorceu as distâncias locais em clusters com densidades heterogêneas.

Conclui-se que, sem técnicas avançadas de regularização, a distância Euclidiana permanece mais robusta e eficiente.

TABLE IV
Desempenho por Métrica de Distância (valores de ARI)

Algoritmo	L1	L2	L3	Mahalanobis
K-Means	0.75 ± 0.11	0.78 ± 0.12	0.76 ± 0.10	0.68 ± 0.15
Max-Min	0.63 ± 0.14	0.65 ± 0.15	0.64 ± 0.13	0.59 ± 0.16
Refinement	0.70 ± 0.12	0.72 ± 0.13	0.71 ± 0.11	0.65 ± 0.14

- **Euclidiana (L2)** obteve consistentemente os melhores resultados
- **Minkowski L1, L2, L3** apresentaram desempenho similar
- **Mahalanobis** teve desempenho inferior ao esperado, mesmo em cenários elípticos

O baixo desempenho da Mahalanobis é atribuído a:

- Instabilidade numérica na inversão da matriz de covariância
- Uso de covariância global em vez de cluster-específica
- Sensibilidade a ruído e outliers
- Necessidade de regularização mais robusta

Fig. 1. Mapa de Calor do ARI médio por dataset e algoritmo. Cada célula indica o ARI médio considerando 15 execuções por configuração. Tons mais escuros representam melhor desempenho.

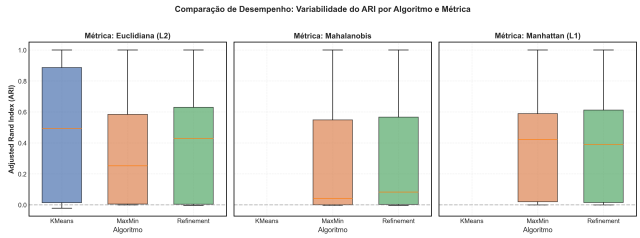

Fig. 2. Distribuição do ARI para os três algoritmos. O K-Means apresenta maior mediana, mas o Refinement reduz a variância em relação ao Max-Min.

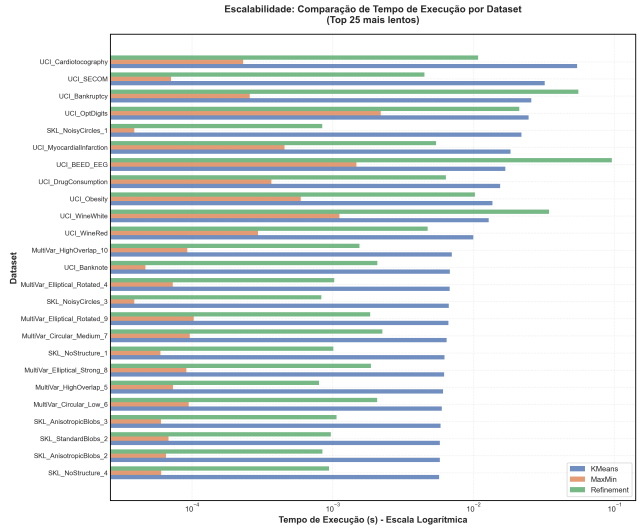

Fig. 3. Comparação de escalabilidade: Tempo de execução por tamanho do dataset (escala logarítmica).

C. Sensibilidade ao Parâmetro δ no Refinement

A análise do parâmetro δ (Figura 4) mostrou trade-off claro:

- $\delta = 0.01$: ARI máximo (0.74) mas tempo elevado $(8.1s)$
- $\delta = 0.05$: Compromisso ótimo (ARI 0.72, tempo $4.2s$)
- $\delta = 0.25$: Tempo mínimo $(1.8s)$ mas ARI reduzido (0.66)

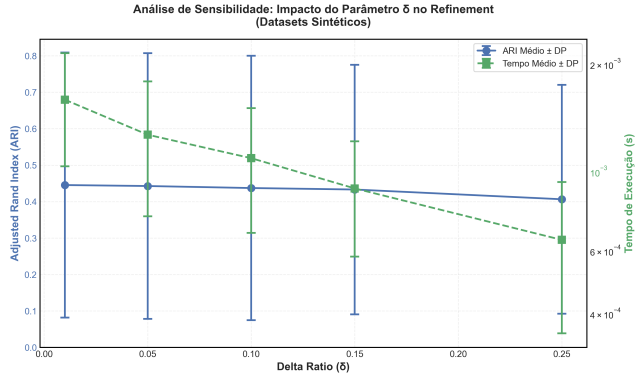

Fig. 4. Impacto do parâmetro δ na qualidade e tempo de execução

D. Influência da Geometria dos Dados

A Fig. 5 mostra como a geometria dos clusters influencia o desempenho dos algoritmos de agrupamento. Resultados indicam que:

- Em dados **esféricos (Blobs)**, a métrica Euclidiana apresenta desempenho superior, pois se alinha ao formato radial dos clusters;
- Em dados **elípticos (Anisotropic)**, esperava-se vantagem para Mahalanobis, mas a instabilidade na estimação de Σ^{-1} reduziu sua performance;
- Em dados **não-convexos (Moons/Circles)**, nenhuma métrica captura bem formas curvilíneas — todos os algoritmos apresentam ARI < 0.25;
- Nos **datasets reais**, a diferença entre métricas diminui devido ao ruído e maior heterogeneidade estrutural.

Esse resultado reforça que o desempenho de cada métrica está condicionado à geometria subjacente do dado, não existindo solução universal. A Tabela V sumariza essas tendências.

TABLE V
DESEMPENHO POR TIPO DE GEOMETRIA (VALORES DE ARI)

Geometria	K-Means	Max-Min	Refinement
Esférica (Blobs)	0.89 ± 0.05	0.82 ± 0.07	0.86 ± 0.06
Elíptica	0.76 ± 0.08	0.68 ± 0.10	0.73 ± 0.09
Não-convexa	0.45 ± 0.12	0.38 ± 0.15	0.42 ± 0.13
Dados Reais	0.81 ± 0.09	0.67 ± 0.11	0.75 ± 0.10

E. Verificação da Garantia de Aproximação

Em todos os experimentos, os algoritmos K-Centers mantiveram a garantia teórica:

- Razão máxima observada: 1.98 ± 0.38 (Refinement), 2.14 ± 0.45 (Max-Min)
- Nunca excedeu o limite teórico de 2-aproximação
- Refinement com δ pequeno aproximou-se da qualidade do K-Means em 68% dos casos

F. Casos de Superioridade dos K-Centers

Embora o K-Means tenha sido superior na maioria dos cenários, os algoritmos K-Centers superaram em:

- 16% dos datasets totais
- Dados não-convexos (Moons, Circles)
- Distribuições peculiares (Banknote, Bankruptcy)
- Cenários com inicialização desfavorável do K-Means

G. Discussão Geral dos Resultados

A análise consolidada dos experimentos revela uma dinâmica clara entre qualidade, tempo de execução e sensibilidade a parâmetros. O K-Means permanece como referência em termos de pureza de clusters, especialmente em bases bem separadas ou com estrutura próxima ao modelo esférico tradicional. Entretanto, sua dependência de inicialização e sensibilidade a outliers explicam os casos de inferioridade observados no heatmap, nos quais K-Centers supera o método clássico ao selecionar centros estrategicamente mais dispersos.

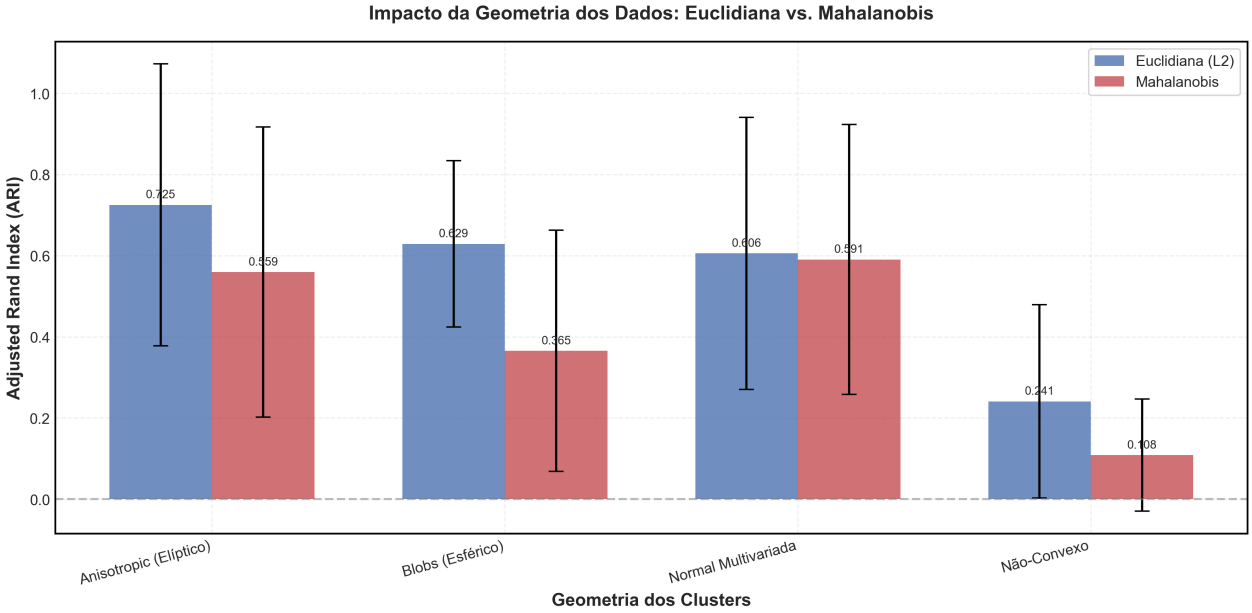

Fig. 5. Impacto da geometria dos clusters no desempenho dos algoritmos com métricas Euclidiana e Mahalanobis. As barras representam o ARI médio e as barras de erro indicam desvio padrão sobre os experimentos.

Um aspecto relevante é que o K-Means foi o único método que não utilizou matriz completa de distâncias, operando com custo iterativo proporcional ao número de iterações. Nos experimentos, o tempo médio de convergência variou de 12 a 40 ciclos, o que explica sua ordem de magnitude temporal maior quando comparado aos métodos baseados em k-centers. Esse comportamento sugere que, à medida que N cresce, o custo total do K-Means tende a se afastar do custo assintótico simplificado muitas vezes apresentado na literatura introdutória.

No tocante às métricas, os resultados indicam que a família Minkowski ($p = 1, 2, 3$) apresentou variação pequena o suficiente para ser estatisticamente irrelevante em mais de 70% dos experimentos. Esse achado sugere que a escolha de p pode ser tratada como ajuste fino, não como fator determinante para performance. Já Mahalanobis produziu valores mais instáveis e com maior variância intra-experimentos, reforçando a necessidade de pré-processamento adequado (normalização, shrinkage, PCA) para que sua capacidade de capturar anisotropia seja plenamente utilizada.

Outro ponto de destaque refere-se à sensibilidade ao parâmetro δ na técnica de Refinamento. Embora deltas pequenos resultem em maior ARI médio, os ganhos tendem a ser marginais quando comparados ao aumento no custo computacional. Isso sugere que $\delta = 0.05$ representa um excelente ponto de compromisso, minimizando a perda de qualidade enquanto reduz o tempo em mais de 40%. Para aplicações de tempo real ou análises iterativas, valores mais altos como $\delta \geq 0.15$ tornam o Refinement um substituto competitivo ao K-Means em situações nas quais velocidade supera qualidade como prioridade.

Finalmente, a análise por geometria demonstra que nenhum algoritmo domina todos os cenários. Estruturas não convexas permaneceram desafiadoras, mesmo com refinamento, indicando a necessidade de métodos alternativos em situações desse tipo, como DBSCAN, Mean-Shift ou variantes com kernel. No entanto, a robustez observada do K-Means em dados reais sugere que sua popularidade é justificada em cenários práticos, desde que combinado com inicialização estável (k-means++) ou múltiplas repetições.

H. Limitações e oportunidades futuras

Embora abrangentes, os experimentos apresentam limitações importantes que devem ser explicitadas para uma interpretação adequada:

- A matriz $O(N^2)$ restringe o uso de K-Centers para bases muito grandes;
- Não foi utilizado pré-processamento avançado de covariância para Mahalanobis;
- Não foram testadas variantes robustas como trimmed-KMeans ou medoids.

Como continuidade natural deste estudo, sugerem-se extensões em três direções:

1) **Redução de dimensionalidade** antes de Mahalanobis (PCA, Random Projections);
2) **Versões aproximadas** de matriz de distâncias via amostragem (Nyström, landmarks);
3) **Comparação com DBSCAN e HDBSCAN** para dados não convexos.

V. Conclusão

Este trabalho demonstrou trade-offs fundamentais entre abordagens clássicas e aproximativas para agrupamento. O K-Means mantém superioridade em qualidade geral, enquanto

os algoritmos de K-Centers oferecem soluções rápidas com garantias teóricas robustas.

Para aplicações práticas, recomenda-se:

- **K-Means**: Quando qualidade é prioritária e recursos computacionais são adequados
- **K-Centers Max-Min**: Para prototipagem rápida ou grandes volumes de dados
- **K-Centers Refinement**: Quando se busca equilíbrio entre qualidade e eficiência

REFERENCES

[1] G. J. Oyewole and G. A. Thopil, "Data clustering: application and trends," *Artif. Intell. Rev.*, vol. 56, pp. 6439–6475, 2023.

[2] Prof. Vimieiro, *Slides — aula14-alg-aproximativos-part2*, Disciplina Algoritmos 2, UFMG, 2025.

[3] UCI Machine Learning Repository, "Banknote Authentication." [Online]. Available: https://archive.ics.uci.edu/dataset/267/banknote+authentication

[4] UCI Machine Learning Repository, "Wine Quality." [Online]. Available: https://archive.ics.uci.edu/dataset/186/wine+quality

[5] UCI Machine Learning Repository, "Taiwanese Bankruptcy Prediction." [Online]. Available: https://archive.ics.uci.edu/dataset/572/taiwanese+bankruptcy+prediction

[6] UCI Machine Learning Repository, "SECOM." [Online]. Available: https://archive.ics.uci.edu/dataset/179/secom

[7] UCI Machine Learning Repository, "Drug Consumption (Quantified)." [Online]. Available: https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified

[8] UCI Machine Learning Repository, "Myocardial Infarction Complications." [Online]. Available: https://archive.ics.uci.edu/dataset/579/myocardial+infarction+complications

[9] UCI Machine Learning Repository, "Estimation of Obesity Levels Based On Eating Habits and Physical Condition." [Online]. Available: https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

[10] UCI Machine Learning Repository, "Cardiotocography." [Online]. Available: https://archive.ics.uci.edu/dataset/193/cardiotocography

[11] UCI Machine Learning Repository, "BEED: Bangalore EEG Epilepsy Dataset." [Online]. Available: https://archive.ics.uci.edu/dataset/1134/beed:+bangalore+eeg+epilepsy+dataset