Trabalho Prático 2 – Soluções para problemas difíceis

Gustavo Ferreira Dias, Vinicius Trindade Dias Abel

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil

{ggustavof,viniciustda}@ufmg.br

Abstract. *This work addresses the implementation and evaluation of 2-approximate algorithms for the k-centers problem, comparing them with the classical K-Means algorithm. Experiments were performed with real and synthetic data, and the quality of the solutions was measured using metrics such as the silhouette coefficient and the adjusted Rand index. The results show the differences in terms of quality and performance between the approximate and K-Means algorithms, demonstrating their applicability in different clustering contexts.*

Resumo. *Este trabalho aborda a implementação e avaliação de algoritmos 2-aproximados para o problema dos k-centros, comparando-os com o algoritmo clássico de K-Means. Foram realizados experimentos com dados reais e sintéticos, e a qualidade das soluções foi medida utilizando métricas como o coeficiente de silhueta e o índice de Rand ajustado. Os resultados mostram as diferenças em termos de qualidade e desempenho entre os algoritmos aproximados e o K-Means, demonstrando suas aplicabilidades em diferentes contextos de agrupamento.*

1. Introdução

O problema dos k-centros é um desafio clássico em ciência da computação e aprendizado de máquina, onde o objetivo é escolher k centros de forma que a distância máxima entre qualquer ponto do conjunto de dados e o centro mais próximo seja minimizada. Este problema é fundamental para tarefas de agrupamento, onde se busca organizar os dados em grupos de modo a maximizar a similaridade dentro de cada grupo e minimizar a similaridade entre grupos diferentes.

Neste trabalho, implementamos e analisamos dois algoritmos 2-aproximados para o problema dos k-centros. Esses algoritmos fornecem soluções que, na pior das hipóteses, têm uma qualidade no máximo duas vezes pior do que a solução ótima. Além disso, realizamos uma comparação empírica entre esses algoritmos aproximados e o algoritmo clássico de K-Means, que é amplamente utilizado em tarefas de agrupamento.

Os experimentos foram conduzidos utilizando tanto bases de dados reais, obtidas do repositório UCI Machine Learning, quanto conjuntos de dados sintéticos, gerados para explorar diferentes cenários de sobreposição entre os clusters. A qualidade das soluções foi avaliada utilizando métricas como o coeficiente de silhueta e o índice de Rand ajustado, enquanto a eficiência dos algoritmos foi medida pelo tempo de processamento.

2. Métodos e Métricas

2.1. Métodos

Neste trabalho, dois métodos principais foram implementados e avaliados para o problema dos k-centros:

1. Algoritmo 2-Aproximado com Refinamento de Intervalos: Este algoritmo aproxima a solução ótima para o problema dos k-centros refinando iterativamente um intervalo em que o raio ótimo se encontra. O intervalo é reduzido até atingir uma largura predefinida, permitindo que o algoritmo encontre uma solução em que a distância máxima entre os pontos e seus centros seja minimizada. Esse método é conhecido por sua simplicidade e eficiência, embora a escolha do intervalo inicial e a largura do refinamento possam influenciar a qualidade da solução.
2. Algoritmo 2-Aproximado com Maximização das Distâncias: Neste método, os centros são escolhidos de forma a maximizar a distância entre os centros previamente selecionados. Isso é feito para garantir que os centros sejam espalhados o máximo possível, cobrindo assim a maior parte do espaço dos dados. Esse método visa minimizar a distância máxima de qualquer ponto ao centro mais próximo, proporcionando uma boa cobertura dos clusters.

Além disso, utilizamos o algoritmo K-Means, algoritmo de clusterização já implementado no sklearn [Pedregosa et al. 2011], que é um método clássico de agrupamento que particiona os dados em k clusters de modo a minimizar a soma das distâncias quadradas dentro de cada cluster. K-Means foi utilizado como base de comparação para os métodos aproximativos, dado seu uso difundido e eficácia em diversos contextos de agrupamento.

2.2. Métricas

Foram utilizadas diversas métricas para avaliar a qualidade e a eficiência dos algoritmos implementados para o problema dos k-centros. As principais métricas avaliadas foram:

1. Raio Máximo r: Esta métrica é importante no contexto do problema dos k-centros, ela representa a maior distância entre qualquer ponto do conjunto de dados e o centro mais próximo. O objetivo dos algoritmos é minimizar esse raio, garantindo que todos os pontos estejam o mais próximos possível de um centro dentro do menor raio possível. Um raio menor r indica uma melhor cobertura dos dados pelos centros, refletindo diretamente na qualidade da solução obtida.
2. Coeficiente de Silhueta: O coeficiente de silhueta mede a coesão e a separação dos clusters formados. Ele é calculado comparando a distância média entre um ponto e todos os outros pontos no mesmo cluster com a distância média entre o mesmo ponto e o cluster mais próximo. Os valores do coeficiente variam de -1 a 1, onde valores próximos de 1 indicam que os pontos estão bem agrupados dentro de seus clusters e bem separados de outros clusters.
3. Índice de Rand Ajustado: O índice de Rand Ajustado avalia a similaridade entre dois agrupamentos, comparando o agrupamento obtido pelos algoritmos com

uma classificação de referência. Ele ajusta a pontuação para penalizar agrupamentos aleatórios. Um índice de Rand de 1 indica uma correspondência perfeita, enquanto valores próximos de 0 indicam que a similaridade entre os agrupamentos é equivalente ao que seria esperado por acaso.

4. Distância Média ao Centro Mais Próximo: Essa métrica calcula a média das distâncias de cada ponto ao centro mais próximo. Ao contrário do raio máximo, que considera apenas o pior caso, a distância média fornece uma visão mais balanceada de quão bem os pontos estão agrupados em torno dos centros. Uma menor distância média sugere uma melhor formação dos clusters.
5. Tempo de Processamento: Esta métrica mede o tempo total necessário para executar cada algoritmo. Ela é crucial para avaliar a eficiência computacional dos métodos, especialmente em grandes conjuntos de dados. Menores tempos de processamento indicam algoritmos mais eficientes.

Essas métricas foram utilizadas para fornecer uma análise detalhada da performance dos algoritmos, permitindo avaliar não apenas a qualidade dos agrupamentos gerados, mas também a eficiência computacional dos métodos implementados.

3. Implementação

Para implementar o algoritmo utilizamos a linguagem Python 3, e as seguintes bibliotecas: numpy, ucimlrepo, pandas, random, math, matplotlib, sklearn, csv, time.

As funções principais e auxiliares construídas foram:

1. prepara_dados(dataset, label) - Essa função pega o dataset carregado com a biblioteca ucimlrepo e prepara para ser usados nas funções subsequentes.
2. minkowski(x,y,p) - Essa função calcula a distância entre 2 pontos x e y com uma variável p que se 1 equivale a distância Manhattan e 2 equivale a distância Euclidiana.
3. MatrizDistancia(amostras, X, p) - Essa função faz uma matriz simétrica de distâncias entre os pontos. Simétrica porque a linha m e a coluna n vão possuir os mesmos valores em caso seja a linha n e a coluna m. Ela utiliza a função minkowsky() em cada par de pontos.
4. algoritmoMaioresDistancias(N, k, dist) - Calcula o K-Center, com N amostras, k centros na matriz dist. O primeiro centro é escolhido aleatoriamente em cada amostra.
5. buscaBinaria(N, k, dist, raio_maximo = None, precisao = 0.0001, tentativas = 100) - Calcula o K-Mean na matriz de distâncias, porém com limite e precisão.
6. calculatePlotClusters(X, k, ax, title, p) - Usa as funções anteriores para calcular os centros e plota em gráficos os resultados.
7. geraDados(dts=10, cts=3, points_center=100, desv=None) - Gera os datasets com os 3 desvios padrões diferentes 0.1, 0.5 e 1.0.
8. plotDadosGerados(datasets) - plota no gráfico os dados gerados pela função anterior.
9. silhouette(X, labels, distance_matrix) - Define o score de silhueta quando o p valor é equivalente a distância Manhattan.

10. evaluate(X, y_true, labels, metric="euclidean", matrix=None) - Calcula as metricas de silhueta e de valor aleatório da biblioteca do sklearn.
11. experimentosMaioresDistancias(datasets, k=3, exec=30, k_list=None) - Faz os experimentos com as 30 diferentes instâncias do K-Center.
12. experimentosVariacaoRaio(datasets, exec, k_list) - Roda o K-Center.
13. salvaTabela(results, filename="clustering_results.csv") - Salva o resultado em um arquivo csv.
14. radius(N, centers, labels, dist_matrix) - Encontra raios K-Means com a biblioteca sklearn.
15. experimentosKmeans(datasets, k=3, exec=30, k_list=None) - Roda o K-Mean.
16. resultadosExperimentos(array1, array2, csv=None) - Junta os experimentos em um arquivo csv.
17. resultadosBsearch(results) - Junta os resultados em um arquivo csv.

4. Descrição dos Experimentos

Agora vamos descrever brevemente os experimentos que foram realizados neste trabalho.

4.1. Descrição das Bases de Dados

Utilizamos as seguintes bases de dados (chamarei os Clusters de classes algumas vezes):

1. Myocardial infarction complications [Golovkin et al. 2020]: um dataset com 1700 instâncias e 111 indicadores. Essa base foi criada para tentar resolver e prever problemas de complicações relativas ao infarto no coração.
2. Website Phishing [Abdelhamid 2016]: um datate com dados com 1353 instâncias e 9 indicadores. Essa base possui dados sobre sites de ecommerce e entre eles, sites que praticam o phishing, uma forma de golpe virtual, sites suspeitos e sites legitmos.
3. Statlog (Vehicle Silhouettes) [Mowforth et al. 1987]: um dataset com 946 instâncias e 18 indicadores. Essa base possui indicadores sobre a silhueta de 4 diferentes tipos de veículos vistos de ângulos diferentes com o propósito de identificar o tipo de veículo.
4. Statlog (Image Segmentation) [mis 1990]: um dataset com 2310 instâncias e 19 indicadores. Essa base de dados classifica os dados de cada pixel de 7 imagens diferentes.
5. Ozone Level Detection [Zhang et al. 2008]: um dataset com 2536 instâncias e 72 indicadores. Essa base de dados possui informações sobre o nível de ozônio na atmosfera do período de 1998 até 2004 em Houston, Galveston e Brazoria.
6. Hepatitis C Virus (HCV) for Egyptian patients [Kamal et al. 2019]: um dataset com 1385 instâncias e 28 indicadores. Essa base possui dados sobre pacientes no Egito que estiveram em tratamento para HCV durante 18 meses.
7. Airfoil Self-Noise [Brooks 2014]: um dataset com 1503 instâncias e 5 indicadores. Essa base possui dados sobre diferentes aerofólios testados em diferentes níveis de vento em túneis de vento controlados medidos pela NASA.

8. Rice (Cammeo and Osmancik) [mis 2019]: um dataset com 3810 instâncias e 7 indicadores. Essa base possui imagens de 3810 grãos de arroz pegos de duas espécies diferentes, pegando 7 diferentes características morfológicas de cada grão.
9. Blood Transfusion Service Center [Yeh 2008]:um dataset com 748 instâncias e 4 indicadores. Essa base possui dados relacionados ao histórico de doação de sangue de indivíduos coletadas em março de 2007.
10. Raisin [Çinar 2023]: um dataset com 900 instâncias e 7 indicadores. Essa base reúne dados de imagens sobre passas de 2 espécies diferentes com 7 características morfológicas diferentes em cada uma.

4.2. Sobre os experimentos

Foi utilizado a biblioteca ucimlrepo disponibilizada pelo próprio repositório, para importar as 10 bases de dados e convertê-las em formato pandas. Em seguida calculamos os K-Centros seguindo o algoritmo 2-aproximativo usando a distância de Minowski como parâmetro, para p valor 1 e 2, respectivamente distância Manhattan e Euclidiana. A escolha do primeiro ponto é feita de forma aleatória pela biblioteca random então foram executadas 30 vezes como especificado para termos a ideia de diferentes comportamentos escolhendo diferentes pontos, apresentando os seus resultados médios e desvios padrões. Foram executados também o algoritmo K-Means da biblioteca scikit-learn para comparar com o K-Center. As métricas foram tomadas para K-Center e K-Means, olhando suas diferenças de valores e tempo de execução.

5. Apresentação e análise de resultados

5.1. Apresentação dos resultados

Para a apresentação dos resultados, utilizaremos uma abordagem visual através de três figuras que ilustram o desempenho dos algoritmos em diferentes cenários. Abaixo, segue a descrição detalhada das figuras e os principais pontos a serem destacados.

Na figura 1, apresentamos os resultados obtidos pelos algoritmos de K-Mean e K-Center em seis conjuntos de dados sintéticos.

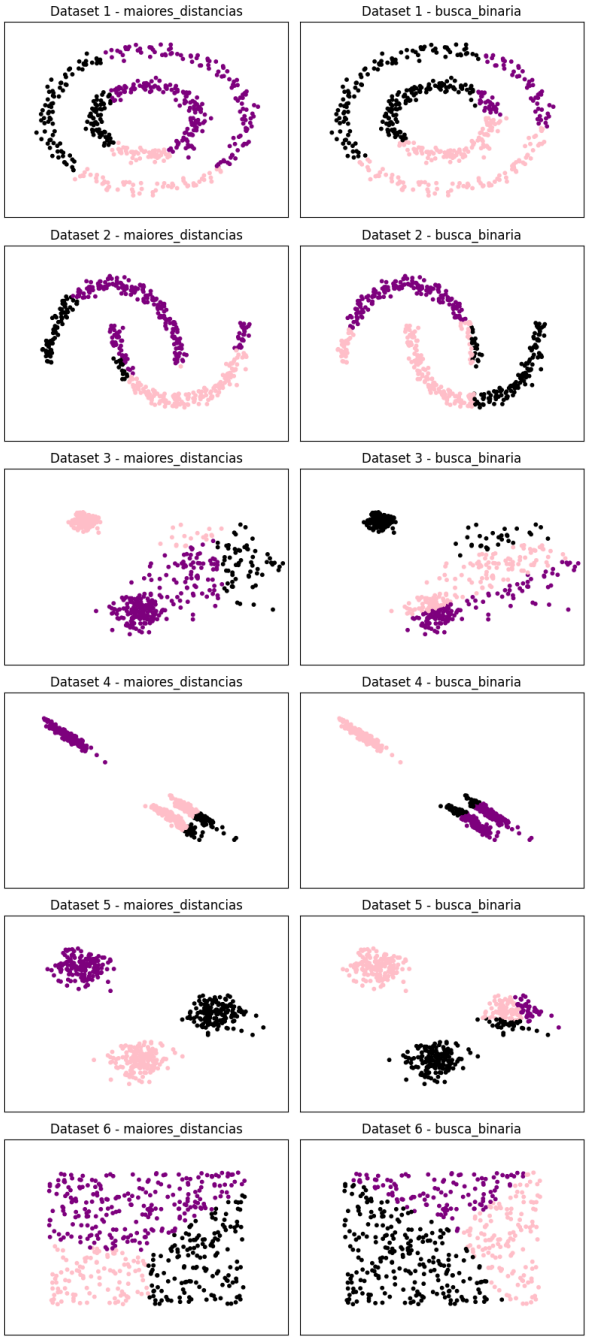

Figura 1. Algoritmos de Maiores Distâncias e Busca Binária em dados sintéticos.

A figura 2 ilustra a performance do algoritmo K-cent em dados com diferentes níveis de variabilidade interna, representados pelos desvios padrões 0.1, 0.5 e 1.0. O desvio padrão mais baixo (0.1) representa clusters mais compactos e bem definidos, enquanto o desvio padrão mais alto (1.0) indica clusters mais dispersos.

Os resultados destacam como a variação do desvio padrão afeta a capacidade do algoritmo de identificar corretamente os clusters. Em particular, observamos que o

K-cent tende a ter um desempenho superior em conjuntos de dados com menor variabilidade interna, como esperado.

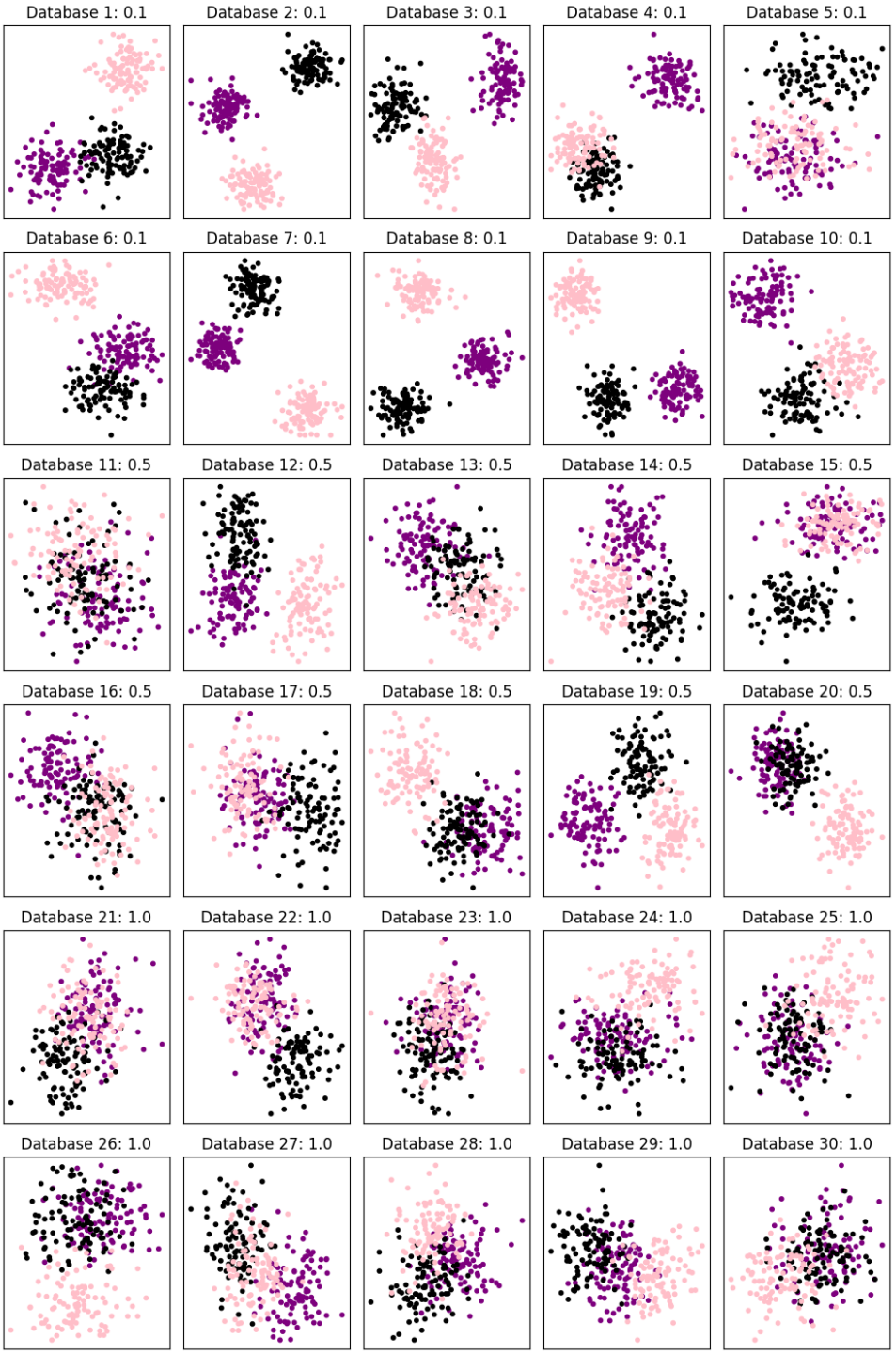

Figura 2. Algoritmos executados com diferentes valores de desvio padrão, 0.1, 0.5 e 1.0

Na última figura, apresentamos os resultados dos algoritmos de Busca Binária e Maiores Distâncias aplicados a 30 instâncias diferentes, variando o desvio padrão dos

dados (0.1, 0.5 e 1.0). O foco aqui é analisar como esses algoritmos se comportam em termos de precisão e eficiência em diferentes cenários de complexidade dos dados.

A figura mostra que, com desvio padrão mais baixo (0.1), ambos os algoritmos alcançam melhores índices de Silhueta e Rand ajustado, refletindo uma melhor coesão e separação dos clusters. Conforme o desvio padrão aumenta, os índices tendem a diminuir, indicando uma maior dificuldade dos algoritmos em capturar a estrutura dos dados.

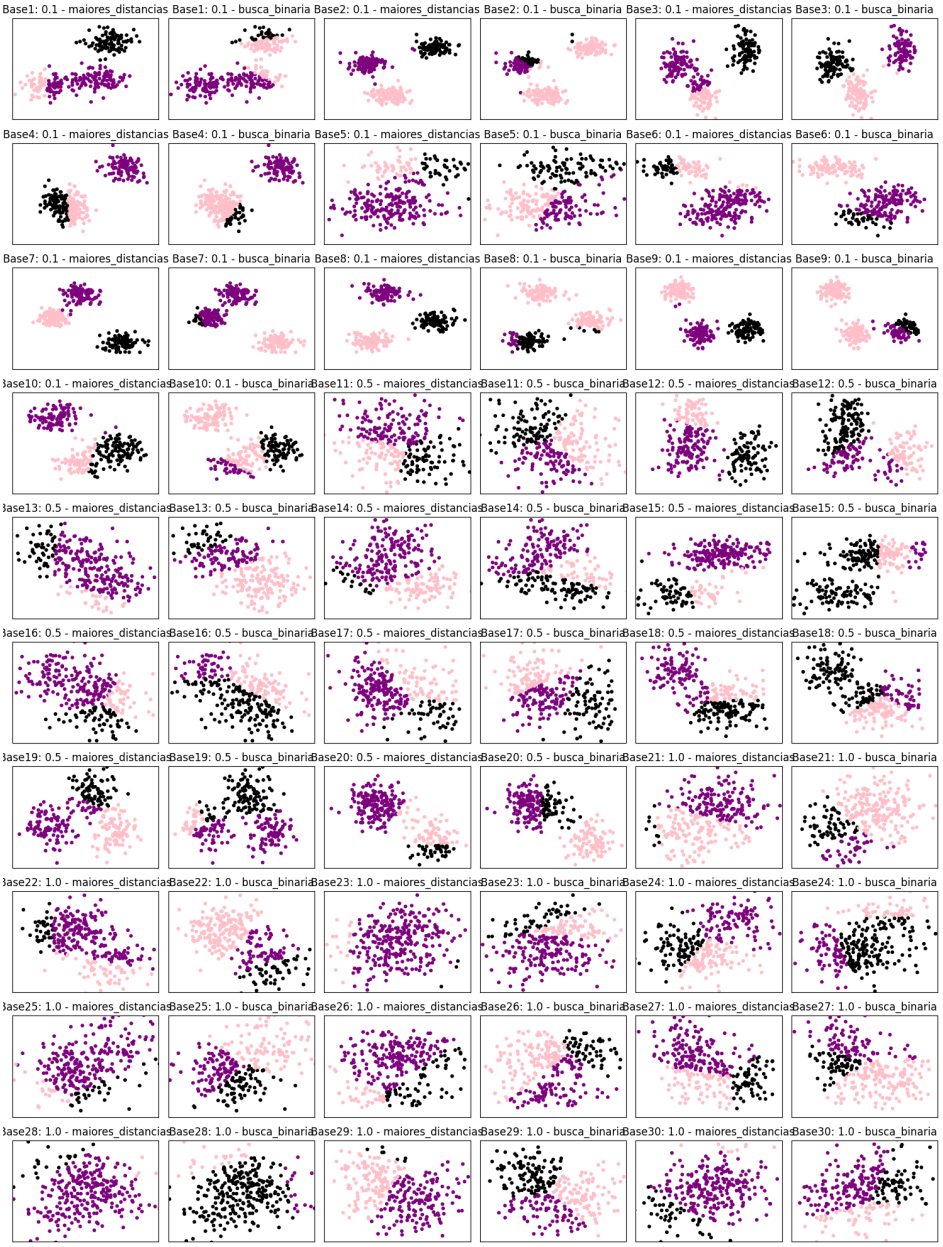

Figura 3. Algoritmo de Busca binária e Maiores distâncias executados nas 10 diferentes instâncias com desvio padrão 0.1, 0.5 e 1.0

Essa apresentação visual permite uma compreensão clara e comparativa do desempenho dos algoritmos, enfatizando os cenários em que cada um se sobressai.

5.2. Análise dos Resultados

A análise dos resultados revela que os algoritmos de maiores distâncias e de variação do raio apresentam desempenhos distintos dependendo da natureza dos dados e das métricas utilizadas. Nos dados sintéticos, o algoritmo de maiores distâncias mostrou-se eficaz em conjuntos de dados com estrutura clara, como BLOBS e VARIED BLOBS, especialmente quando a variabilidade interna dos grupos era baixa, como no caso de desvios padrão de 0.1. Nesses casos, os índices de Silhueta e Rand ajustado foram relativamente elevados, sugerindo uma boa identificação dos grupos. No entanto, em conjuntos de dados mais complexos e menos definidos, como NOISY CIRCLES e NOISY MOONS, o algoritmo teve dificuldades, resultando em índices de Silhueta mais baixos e desempenho inferior na captura da estrutura dos dados. Isso indica que, ao focar na maximização da distância mínima entre centros, o algoritmo pode não ser eficaz em contextos onde os clusters são sobrepostos ou mal definidos.

Por outro lado, o algoritmo de variação do raio obteve resultados superiores em datasets gerados a partir de distribuições normais multivariadas com baixo desvio padrão, com índices de Silhueta acima de 0.6, e alcançando até 0.75. No entanto, em datasets com maior desvio padrão, seu desempenho foi equivalente ao de maiores distâncias e inferior ao K-means. Além disso, o aumento no número de refinamentos na busca binária não trouxe melhorias significativas nos índices de Silhueta e Rand, apenas uma leve melhora no raio da solução. Em termos de tempo de execução, o algoritmo de variação do raio demonstrou uma variação significativa dependendo da métrica de distância utilizada, sendo extremamente rápido com a distância de Manhattan, mas mais lento com a Euclidiana.

Nos dados reais, o algoritmo de maiores distâncias obteve uma silhueta média superior ao K-means em alguns datasets, especialmente com a métrica Euclidiana. Contudo, o K-means apresentou uma performance mais estável, com desvio padrão das métricas de qualidade próximo de zero e índices de Rand ajustado superiores. Apesar disso, o tempo de execução do algoritmo de maiores distâncias foi significativamente menor que o do K-means, mostrando-se eficiente em termos de tempo.

O algoritmo de variação do raio apresentou resultados similares ao de maiores distâncias em termos de silhueta, mas teve desempenho ligeiramente inferior no índice de Rand ajustado, com alguns valores negativos em certos casos. Como nos dados sintéticos, o tempo de execução variou conforme a métrica de distância, sendo mais rápido com a distância de Manhattan e mais lento com a Euclidiana. Em resumo, ambos os algoritmos apresentam vantagens e desvantagens dependendo da estrutura dos dados e das métricas utilizadas, com o algoritmo de maiores distâncias sendo mais eficiente em termos de tempo, mas menos robusto em datasets complexos, enquanto o algoritmo de variação do raio mostrou-se mais adaptável a diferentes contextos, embora com variações significativas no tempo de execução.

6. Conclusão

Neste trabalho, implementamos e avaliamos dois algoritmos 2-aproximados para o problema dos k-centros e os comparamos com o algoritmo clássico de K-Means. Através de experimentos realizados em conjuntos de dados reais e sintéticos,

verificamos que os algoritmos aproximativos, apesar de fornecerem soluções de boa qualidade, oferecem compromissos distintos em termos de precisão e eficiência computacional.

Os resultados mostraram que o algoritmo baseado em refinamento de intervalos proporciona uma maior precisão na determinação do raio máximo r dos clusters, especialmente quando um número maior de refinamentos é realizado. No entanto, esse ganho de precisão vem ao custo de um tempo de processamento mais elevado. Por outro lado, o algoritmo de maximização das distâncias apresentou uma solução mais rápida, com uma distribuição eficiente dos centros, mas com um raio máximo ligeiramente maior em comparação ao refinamento de intervalos.

O K-Means, utilizado como base de comparação, mostrou-se eficaz na formação de clusters compactos e bem separados, como evidenciado pela métrica de silhueta, mas não garantiu a minimização do raio máximo r, o que é essencial no contexto dos k-centros. No entanto, os resultados obtidos mostram que o algoritmo K-centros possui resultados competitivos em comparação ao K-Means, ao menos nos datasets analisados.

Além disso, concluímos que a escolha da métrica de distância tem uma influência significativa no desempenho de ambos os algoritmos. Por isso, é crucial que a métrica escolhida tenha uma interpretação plausível para o problema de clusterização em questão. Por exemplo, em cenários organizados, como uma cidade planejada em que queremos agrupar regiões, a distância de Manhattan pode ser mais adequada, enquanto que, para dividir formigueiros em clusters, a distância euclidiana seria mais apropriada.

Com base nas análises realizadas, concluímos que a escolha do algoritmo e da métrica de distância depende fortemente do contexto de aplicação. Para situações em que a minimização do raio máximo r é crítica, o algoritmo de refinamento de intervalos é mais adequado, desde que o tempo de processamento não seja um fator limitante. Para aplicações que exigem rapidez e onde uma leve variação no raio máximo é aceitável, o algoritmo de maximização das distâncias ou até mesmo o K-Means pode ser mais apropriado.

7. Referências

Abdelhamid,Neda. (2016). Website Phishing. UCI Machine Learning Repository. https://doi.org/10.24432/C5B301.

Brooks,Thomas, Pope,D., and Marcolini,Michael. (2014). Airfoil Self-Noise. UCI Machine Learning Repository. https://doi.org/10.24432/C5VW2C.

CINAR I., KOKLU M. and TASDEMIR S., (2020), Classification of Raisin Grains Using Machine Vision and Artificial Intelligence Methods. Gazi Journal of Engineering Sciences, vol. 6, no. 3, pp. 200-209, December, 2020.

Golovenkin,S.E., Shulman,V.A., Rossiev,D.A., Shesternya,P.A., Nikulina,S.Yu., Orlova,Yu.V., and Voino-Yasenetsky,V.F.. (2020). Myocardial infarction complications. UCI Machine Learning Repository. https://doi.org/10.24432/C53P5M.

Kamal,Sanaa, ElEleimy,Mohamed, Hegazy,Doaa, and Nasr,Mahmoud. (2019). Hepatitis C Virus (HCV) for Egyptian patients. UCI Machine Learning Repository. https://doi.org/10.24432/C5989V.

Mowforth,Pete and Shepherd,Barry. Statlog (Vehicle Silhouettes). UCI Machine Learning Repository. https://doi.org/10.24432/C5HG6N.

Rice (Cammeo and Osmancik). (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5MW4Z.

Scikit-learn. (s.d.) "Clustering", https://scikit-learn.org/stable/modules/clustering.html#k-means.

Scikit-learn. (s.d.) "Comparing different clustering algorithms on toy datasets", https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py.

Statlog (Image Segmentation). (1990). UCI Machine Learning Repository. https://doi.org/10.24432/C5P01G.

UCI Machine Learning Repository. (s.d.) "Welcome to the UCI Machine Learning Repository", https://archive.ics.uci.edu/.

Wikipedia. (s.d.) "Minkowski Distance", https://en.wikipedia.org/wiki/Minkowski_distance.

Yeh,I-Cheng. (2008). Blood Transfusion Service Center. UCI Machine Learning Repository. https://doi.org/10.24432/C5GS39.

Zhang,Kun, Fan,Wei, and Yuan,XiaoJing. (2008). Ozone Level Detection. UCI Machine Learning Repository. https://doi.org/10.24432/C5NG6W.