Implementação de algoritmos aproximativos para o problema dos K-Centros

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte, MG – Brasil

joaomarcoscampos@dcc.ufmg.br, rafamg@ufmg.br

Abstract. *The k-center problem involves finding K central points in a set to minimize the maximum distance between any point and its nearest center. As an NP-hard problem, finding optimal solutions is computationally complex, with no known polynomial-time algorithms for all cases. Approximate algorithms, which provide good solutions within reasonable time and estimate proximity to the optimal solution, are a viable alternative. This work aims to explore and assess the efficiency and quality of solutions provided by several of these approximate algorithms.*

Resumo. *O problema dos k-centros, amplamente estudado na Ciência da Computação, busca identificar K pontos centrais em um conjunto de pontos para minimizar a maior distância entre um ponto e seu centro mais próximo. Sendo NP-difícil, encontrar soluções ótimas é computacionalmente complexo, sem algoritmos conhecidos que resolvam todos os casos em tempo polinomial. Como alternativa, algoritmos aproximativos oferecem boas soluções em tempo razoável e permitem estimar a proximidade à solução ótima. Este trabalho explora e avalia a eficiência e qualidade das soluções fornecidas por esses algoritmos aproximativos.*

1. Introdução

O problema dos k-centros/agrupamento é bastante conhecido no âmbito da Ciência da Computação (mais especificamente, em Aprendizado de Máquina), muito por sua ampla aplicação acadêmica ou no mercado. A ideia dele é relativamente simples: dado um conjunto de pontos e um valor K, deseja-se encontrar K pontos centrais tais que todos os pontos estejam o mais perto possível dos centros encontrados. Mais formalmente, o objetivo é minimizar a maior distância entre um ponto e o centro mais próximo.

Dentro disso, há diversas variações. É possível, para um mesmo conjunto de pontos, tentar encontrar diferentes números de centros, ou até mudar a forma como a distância entre cada ponto é calculada. De qualquer forma, o problema sempre tem uma característica marcante: é NP-difícil, ou seja, é pelo menos tão difícil quanto qualquer problema na classe NP (problemas para os quais é possível verificar uma solução em tempo polinomial), o que significa que, se for possível encontrar uma solução eficiente para ele, então pode-se resolver todos os problemas de NP de maneira eficiente.

Encontrar soluções ótimas para o problema de agrupamento pode ser uma tarefa computacionalmente custosa, especialmente porque o problema é NP-difícil, o que significa que não há algoritmos conhecidos que encontrem soluções ótimas em tempo polinomial para todos os casos. Nesse contexto, o uso de algoritmos aproximativos, que

fornecem soluções boas (embora não ótimas) em tempo razoável e para os quais é possível estimar a proximidade da solução obtida em relação à ótima, surge como uma alternativa promissora. O objetivo deste trabalho é explorar a implementação desses algoritmos aproximativos e avaliar sua eficiência e a qualidade das soluções que produzem.

2. Implementação dos algoritmos

Como solicitado na especificação do trabalho, foi feita a implementação de dois algoritmos aproximativos. Antes disso, para evitar computações repetidas e desnecessárias, calculamos a distância entre cada par de pontos do conjunto S e as armazenamos numa matriz de distâncias. Ademais, implementamos a função de distância de Minkowski, que, basicamente, calcula a distância entre dois pontos com base num parâmetro.

O primeiro algoritmo busca encontrar os k centros utilizando uma estratégia de busca binária para minimizar o raio máximo de um agrupamento. Primeiro, ele calcula o maior raio possível (distância máxima entre pontos). Em seguida, divide o intervalo de possíveis raios e, em cada iteração, verifica se é possível agrupar os pontos com k centros, dado um raio intermediário. Se for possível, o algoritmo ajusta o intervalo para buscar raios menores; caso contrário, busca em raios maiores. O processo termina quando o intervalo é suficientemente pequeno, retornando os centros e o menor raio encontrado que permite o agrupamento. Note que o fator que define o quão pequeno o intervalo será pode variar; falaremos mais sobre isso posteriormente no artigo.

O segundo algoritmo, por sua vez, começa escolhendo um ponto aleatório do conjunto de dados como o primeiro centro. Em seguida, adiciona novos centros um por um. A cada iteração, calcula a distância mínima de todos os pontos aos centros já selecionados e escolhe como próximo centro o ponto que está mais distante de qualquer centro já selecionado. O objetivo é espalhar os centros pelo espaço de forma a maximizar a distância mínima entre centros consecutivos. No final, o algoritmo retorna os centros selecionados e a maior distância mínima encontrada.

3. Seleção dos dados para avaliação

Para verificar o funcionamento e avaliar os algoritmos implementados, foi feita uma seleção de dados do *UCI Machine Learning Repository*. Filtramos os conjuntos por aqueles que só continham valores numéricos e, a fim de ter pontos o suficiente para realizar uma avaliação satisfatória, buscamos apenas aqueles com pelo menos 700 pontos. Com isso, chegamos a 11 conjuntos (seriam 10, mas um deu erro, então pegamos mais um): Raisin, Wine Quality, Spambase, Statlog Vehicle Silhouettes, Banknote Authentication, Pen Based Recognition Of Handwritten Digits, Optical Recognition Of Handwritten Digits, Magic Gamma Telescope, Waveform Database Generator Version 1, Statlog Image Segmentation, Cardiotocography.

Também geramos 20 conjuntos de dados sintéticos para a realização das avaliações. 10 desses conjuntos foram gerados seguindo um exemplo de comparação de algoritmos de agrupamentos da documentação da biblioteca Scikit-Learn. Para os outros 10, geramos conjuntos bidimensionais usando a distribuição normal multivariada, conforme especificado pelo professor.

4. Experimentos e Resultados

Cada algoritmo foi testado em todos os conjuntos de dados, totalizando 30 testes por algoritmo em 30 datasets diferentes. Com base nesses testes, foram geradas métricas e realizadas análises.

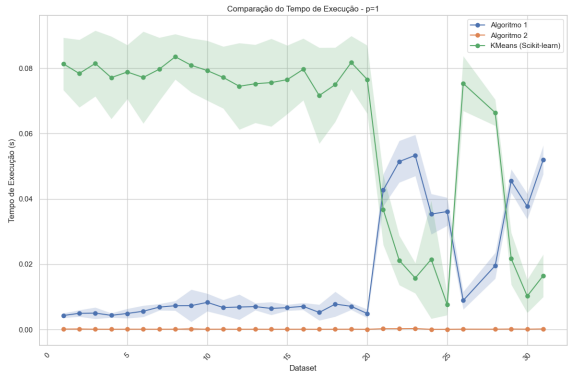

Figure 1. Tempo - p=1

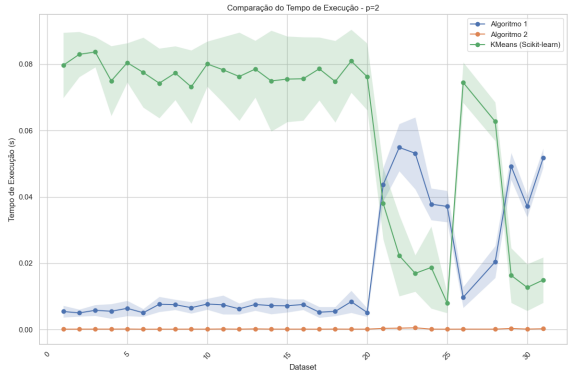

Figure 2. Tempo - p=2

Uma análise inicial focou no tempo de execução em função do parâmetro da distância de Minkowski. Observou-se que o parâmetro p não exerce uma grande influência no tempo de execução dos algoritmos, o que é esperado, pois a operação é calculada rapidamente para ambos os valores de p. Embora o impacto no tempo de execução seja pequeno, o parâmetro pode afetar significativamente outros resultados, e uma análise mais detalhada será realizada para cada caso.

Para p = 1:

Foi avaliada inicialmente a métrica ”Silhouette Score”. O Silhouette Score é uma métrica que avalia a qualidade de um clustering, medindo o quão bem um objeto está agrupado em relação aos outros clusters. Observou-se que os algoritmos mostram consistência em alguns datasets, mas variam em outros.

Embora os resultados sejam similares entre os algoritmos, alguns apresentam uma margem de erro maior, como o algoritmo 2, que frequentemente captura o raio máximo. Apesar de algumas variações, todos os algoritmos apresentaram resultados satisfatórios para esta métrica.

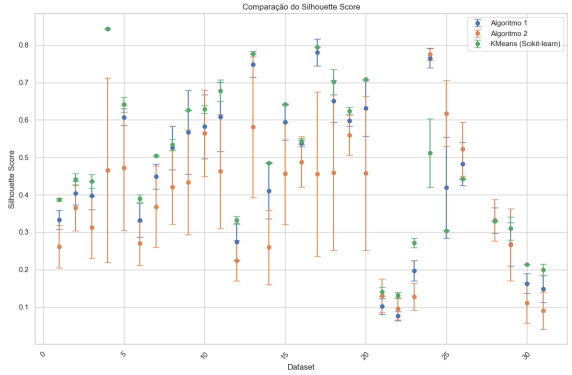

Figure 3. silhouette score p=1

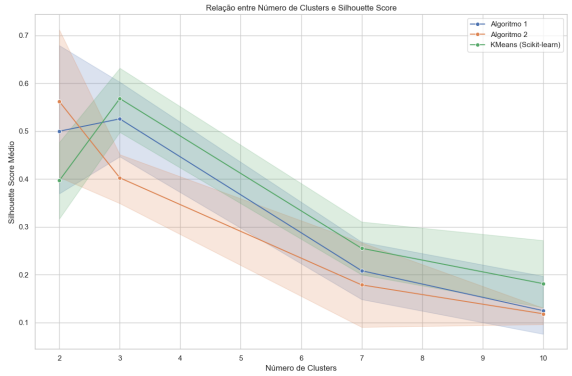

Figure 4. Relaçao clusters - silhouette

Para entender melhor o comportamento dos algoritmos, analisou-se como as respostas variavam com o número de clusters e as métricas resultantes. Investigou-se a relação entre o número de clusters, o tempo de execução, a métrica Adjusted Rand Index (ARI) e o Silhouette Score.

O Adjusted Rand Index (ARI) é uma métrica que avalia a similaridade entre duas partições de dados. É uma forma de medir a concordância entre dois clusters ou entre um clustering e uma verdade conhecida (rótulo verdadeiro).O Rand Index (RI) mede a concordância entre duas partições como a proporção de pares de pontos que são corretamente agrupados ou separadamente agrupados em ambas as partições. O ARI ajusta o RI para que o valor esperado para partições aleatórias seja zero.

Observou-se que, conforme o número de clusters aumentava, o ARI apresentava variações, com o algoritmo do sklearn se destacando como o mais adaptado. O ARI exibiu máximos e mínimos, indicando que o desempenho dos algoritmos varia com o número de clusters, enquanto a análise para poucos clusters foi inconclusiva. O tempo de execução também apresentou comportamentos distintos: o Algoritmo 2 manteve-se rápido, enquanto o outro algoritmo aproximativo teve um aumento no tempo de execução com o número de clusters. O K-Means mostrou maior estabilidade.

Para p = 2:

As avaliações foram repetidas para p = 2 e os resultados foram os mostrados a partir da figura 7:

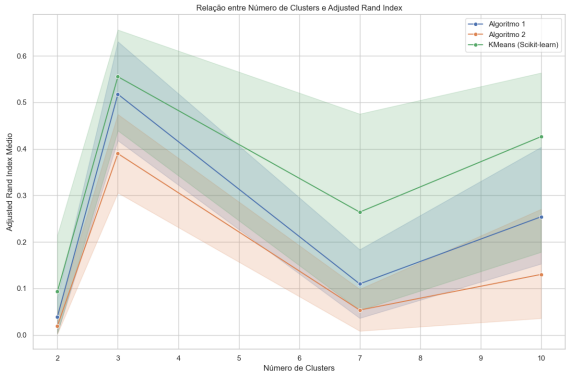

Figure 5. Relaçao clusters - ARI

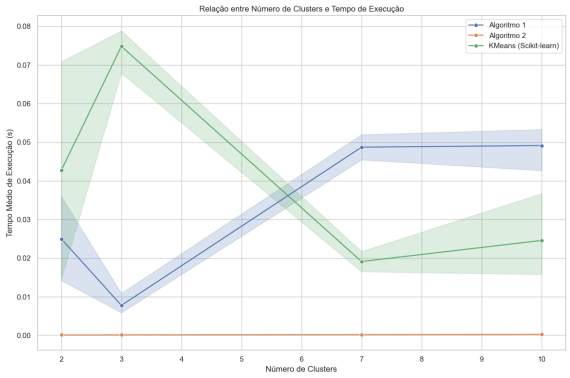

Figure 6. Relação clusters - tempo de execução

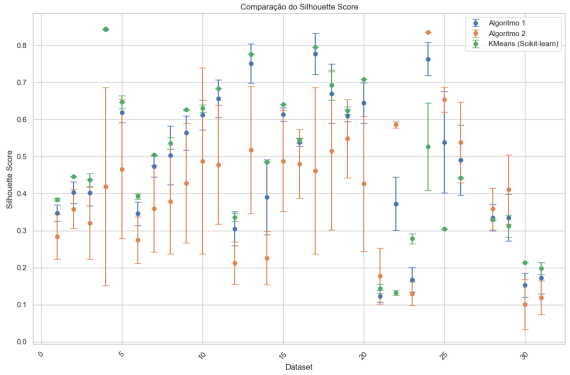

Figure 7. Sillouette score p=2

A métrica de Silhouette Score mostrou algumas diferenças, mas os resultados gerais permaneceram próximos aos observados para p = 1. O K-Means se destacou na métrica, embora, em alguns casos, os algoritmos aproximativos tenham mostrado um desempenho superior. Foi também analisada a influência da métrica de distância na relação entre o número de clusters e as demais métricas.

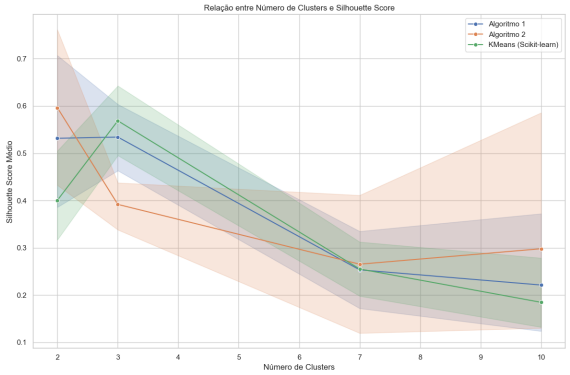

Figure 8. Relação clusters - silhouette

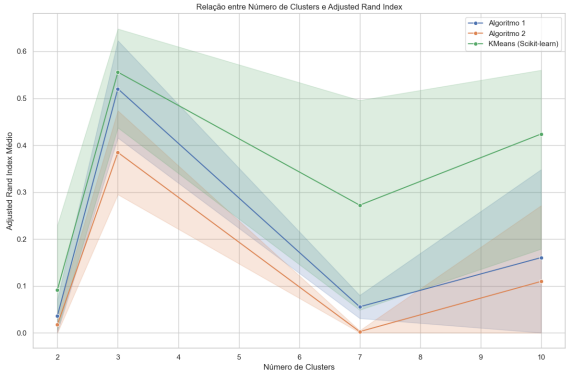

Figure 9. Relação clusters - ARI

Observou-se que a métrica de distância tem um impacto significativo. Notou-se uma melhora nos algoritmos aproximativos na métrica de Silhouette Score com o aumento

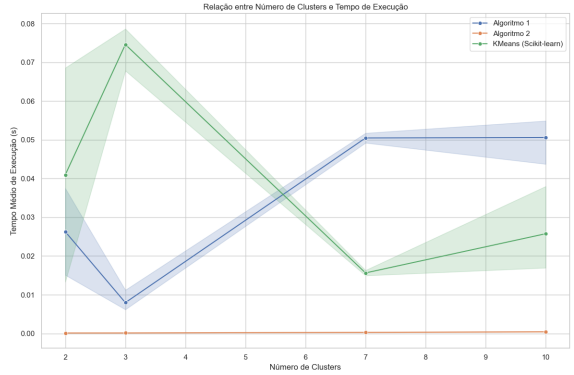

Figure 10. Relação clusters - tempo de execução

do número de clusters, especialmente no segundo algoritmo, que se adaptou melhor. O primeiro algoritmo apresentou uma deterioração, embora em menor grau.

O tempo de execução foi consistente para ambas as métricas de distância, mas o ARI apresentou pequenas variações nos resultados dos algoritmos aproximativos.

4.1. Avaliação da largura do intervalo para o 1° algoritmo aproximativo

A fim de avaliar a sensibilidade do algoritmo baseado em refinamento de intervalos, fizemos alterações na largura do intervalo em que o raio ótimo se encontra e observamos como a solução foi afetada, variando-o entre 1% e 25% com saltos de 5% entre cada.

Como as imagens mostram, as diferenças usando as métricas indicadas para avaliação (silhueta e índice de Rand ajustado) mostram semelhanças entre os valores apontados.

```
Dataset 1 Interval 0.01 completed
Silhouette score 1: 0.35042449690037114
Silhouette score sklearn: 0.382968483445139
Adjusted Rand score 1: 0.004215079330415679
Adjusted Rand score sklearn: -0.0013628397567622267
Execution time 1: 0.004726886749267578 seconds
Execution time sklearn: 0.08905529975891113 seconds

Dataset 1 Interval 0.05 completed
Silhouette score 1: 0.3208388955439474
Silhouette score sklearn: 0.3905012189537918 6
Adjusted Rand score 1: 0.011939031848497424
Adjusted Rand score sklearn: -0.0018781199031285523
Execution time 1: 0.003468751907348633 seconds
Execution time sklearn: 0.08516621589660645 seconds

Dataset 1 Interval 0.1 completed
Silhouette score 1: 0.35772544486553226
Silhouette score sklearn: 0.38022455664665505
Adjusted Rand score 1: 0.006235988358043799
Adjusted Rand score sklearn: -0.0018864746102919823
Execution time 1: 0.0030989646911621094 seconds
Execution time sklearn: 0.08879017829895 02 seconds
```

Figure 11. Intervalos entre 1% e 10%

```
Dataset 1 Interval 0.15 completed
Silhouette score 1: 0.33001236142244733
Silhouette score sklearn: 0.38864166273524686
Adjusted Rand score 1: 0.0001238802616073 1316
Adjusted Rand score sklearn: -0.0016078540841587012
Execution time 1: 0.003301858901977539 seconds
Execution time sklearn: 0.08400654792785645 seconds

Dataset 1 Interval 0.2 completed
Silhouette score 1: 0.3675237558560513
Silhouette score sklearn: 0.38007378217651194
Adjusted Rand score 1: 0.0013635268080610136
Adjusted Rand score sklearn: -0.0018781199031285523
Execution time 1: 0.0034279823303222656 seconds
Execution time sklearn: 0.08638763427734375 seconds

Dataset 1 Interval 0.25 completed
Silhouette score 1: 0.34266690686194895
Silhouette score sklearn: 0.3906064352463991
Adjusted Rand score 1: 0.020404086620567464
Adjusted Rand score sklearn: -0.0017556596493363366
Execution time 1: 0.003278255462646484 4 seconds
Execution time sklearn: 0.08879566192626953 seconds
```

Figure 12. Intervalos entre 15% e 25%

Para visualizar melhor as diferenças, plotamos os centros e comparamo-os com os resultados do algoritmo da biblioteca Scikit-Learn. Como é possível observar nas figuras 13, 14 e 15, os centros não ficam muito mais próximos dos gerados pela biblioteca supracitada; no entanto, notamos resultados mais coerentes com intervalos de 15% e 25%.

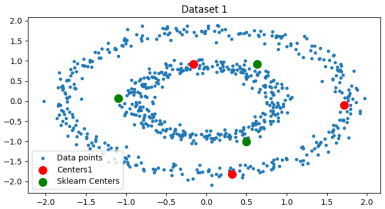

Figure 13. 5%

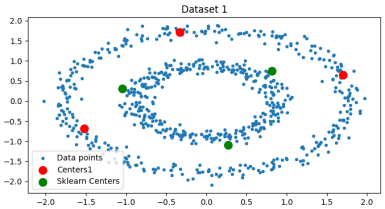

Figure 14. 15%

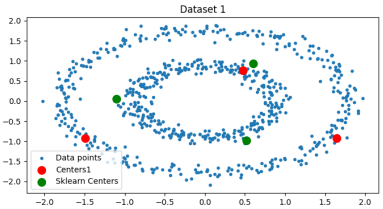

Figure 15. 25%

5. Conclusão

O trabalho foi bastante interessante para que pudéssemos explorar e observar mais sobre o funcionamento prático de algoritmos aproximativos e como eles se comparam com implementações amplamente usadas na área do desenvolvimento.

Além disso, foi possível entender melhor como funciona o algoritmo de K-Centros e de que forma os clusters são escolhidos. Com isso, o desenvolvimento do trabalho agregou bastante.

6. Referências

https://archive.ics.uci.edu/

Documentação - Scikit-Learn