Implementação e Análise de Algoritmos Aproximativos para o Problema dos k-Centros

Bruno Oliveira Souza Santos[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
30.123-970 - Belo Horizonte, MG - Brasil

brunooss@ufmg.br

Abstract. *This work addresses the implementation and empirical analysis of approximation algorithms for the k-center problem, focusing on the comparison between the 2-approximation algorithm and the K-Means algorithm. The results are evaluated based on metrics such as the maximum radius of the clusters and execution time, using both real and synthetic datasets.*

Resumo. *Este trabalho aborda a implementação e análise empírica de algoritmos aproximativos para o problema dos k-centros, com foco na comparação entre os algoritmos 2-aproximado e o algoritmo de força bruta. Os resultados são avaliados com base em métricas como o raio máximo dos centros encontrados e o tempo de execução, utilizando conjuntos distintos de dados reais e sintéticos.*

1. Introdução

O problema dos k-centros é um caso clássico de otimização combinatória que tem grande relevância em aplicações de agrupamento de dados. É caracterizado por um conjunto de n pontos, dentre os quais devem ser encontrados k centros, de modo que os pontos estejam tão próximos quanto possível destes. Isto é, cada ponto se situa a uma distância máxima *r* de um dos centros encontrados.

Este problema pode ser resolvido por meio de algoritmos de força bruta, uma vez que é um problema computacionalmente difícil (NP-difícil). Sendo assim, à medida que o conjunto de pontos aumenta, a execução do algoritmo tem seu custo aumentado exponencialmente, sendo uma opção inviável para extensos conjuntos.

Sendo assim, uma alternativa interessante para solucionar o problema em tempo plausível é a abordagem de algoritmos aproximativos que, apesar de não apresentarem necessariamente soluções ótimas, garantem proximidade com o resultado ótimo.

Nesse sentido, serão observadas duas implementações de algoritmos 2-aproximados para o problema dos k-centros e compará-los com o algoritmo de força bruta, em relação ao tempo de execução e à otimalidade do raio máximo fornecido por cada um deles. Para fins de comparação, foram utilizados conjuntos de dados reais e sintéticos.

2. Descrição do Problema

O problema dos k-centros (*clustering*) pode ser formalizado como a minimização do raio máximo de um conjunto de pontos. Nesse sentido, o problema é formulado tal como a seguir:

Seja um conjunto $S = \{s_1, s_2, s_3, ..., s_n\}$, uma métrica $dist : S \times S- > R^+$ e um inteiro k definidos, deve-se encontrar um conjunto de pontos $C = \{c_1, c_2, c_3, ..., c_k\}$ responsáveis por dividir o conjunto inicial em k grupos, de modo que a distância máxima $r(C)$ entre um ponto arbitrário e seu centro seja tal que:

- $r(C) = max(dist(s_1, C), dist(s_2, C), ..., dist(s_i, C))$
- $dist(s_i, C) = min(dist(s_i, c_1), dist(s_i, c_2), ..., dist(s_i, c_k))$

Ou seja, o raio máximo $r(C)$ dos grupos deve ser minimizado.

3. Metodologia

Serão implementados dois algoritmos aproximativos para o problema dos k-centros: o primeiro se baseia na seleção de centros que maximizem a distância mínima entre os ponto do conjunto e os centros escolhidos. Já o segundo é caracterizado por refinar a estimativa do raio ótimo da solução em um intervalo de valores possíveis.

Em ambos os algoritmos, o raio ótimo é obtido a partir de uma função de cálculo de distância entre os pontos do conjunto. Para o cálculo das distâncias, utilizou-se a distância de *Minkowsky* com parâmetro $p = 1, 2$, sendo o primeiro a distância de *Manhattan* e o segundo a distância Euclidiana.

A fim de analisar a otimalidade dos algoritmos aproximativos, estes serão submetidos a uma série de testes, com conjuntos de dados e quantidade esperada de centros diferentes. Para cada experimento realizado, os resultados dos algoritmos serão comparados tanto entre si quanto com o algoritmo de força bruta, com solução ótima.

3.1. Quantidade de Testes

Embora os algoritmos utilizados sejam 2-aproximados, a escolha dos centros iniciais pode influenciar significativamente a qualidade da solução. Considerando essa variabilidade, cada conjunto de dados será testado em 30 execuções distintas de cada algoritmo.

3.2. Métricas para Análise

Em cada execução, serão armazenados o raio da solução e as seguintes métricas de avaliação de agrupamentos: a métrica de silhueta e o índice de Rand ajustado. Essas métricas são calculadas por meio de implementações disponíveis na biblioteca *Scikit Learn*[2].

Além das métricas de qualidade dos agrupamentos, também será registrado o tempo de processamento de cada execução, a fim de observar a eficiência dos algoritmos aproximativos em relação ao algoritmo de força bruta.

Ademais, para o algoritmo baseado em refinamento de intervalos, a análise se concentrará em como a largura do intervalo contendo o raio ótimo afeta a qualidade da solução. O número de refinamentos será variado entre 1% e 25%, com cinco valores distintos nessa faixa que seguem uma progressão linear.

Por fim, objetiva-se analisar os resultados obtidos em todos os experimentos, por meio da consolidação em uma tabela com as médias e os desvios-padrão por experimento para cada algoritmo.

4. Conjuntos de Dados

A fim de avaliar empiricamente os algoritmos aproximativos para o problema de k-centros, obteve-se 30 conjuntos de dados distintos, sendo 10 deles conjuntos de dados reais e 20 conjuntos de dados sintéticos.

4.1. Conjuntos de Dados Reais

Os dados reais foram obtidos da UCI Machine Learning Repository [4], selecionando-se 10 conjuntos de dados com, no mínimo, 700 instâncias, e somente atributos numéricos, os quais são detalhados a seguir:

- **"Pen-Based Recognition of Handwritten Digits"**[5]: contém dados de dígitos manuscritos capturados por um tablet, os quais devem ser segregados em grupos de seus dígitos correspondentes. Serão utilizados 16 atributos de características extraídas dos traços, com $k = 10$, correspondendo aos dígitos de 0 a 9.
- **"Letter Recognition"**[6]: consiste em dados para reconhecimento de letras maiúsculas do alfabeto americano, serão empregados 16 atributos numéricos extraídos de imagens. O valor de k será definido como 26, de modo que cada centro representa uma letra do alfabeto.
- **"Dry Bean"**[7]: apresenta dados morfológicos de diferentes tipos de feijões secos. Serão analisadas 16 características relacionadas à forma, tamanho e textura, com $k = 7$, alinhado às sete classes de espécies de feijão presentes no conjunto.
- **"Rice (Cammeo and Osmancik)"**[8]: semelhante ao anterior, contém características de dois tipos de arroz, para as quais serão utilizados 7 atributos morfológicos dos grãos. Definindo $k = 2$, os centros representam os dois tipos de arroz em questão.

- **"Phishing Websites"**[9]: fornece características para detecção de sites de phishing. Serão analisados 30 atributos binários relacionados às características dos sites, com $k = 2$, distinguindo entre sites de phishing e não-phishing.
- **"Optical Recognition of Handwritten Digits"**[10]: tal como o primeiro conjunto, consiste em imagens de dígitos manuscritos, porém serão utilizados 64 atributos correspondentes aos pixels de imagens 8x8. Analogamente, $k = 10$, representando os dígitos de 0 a 9.
- **"Statlog (Shuttle)"**[11]: contém dados de sensores de ônibus espacial. Serão analisados 9 atributos numéricos, com $k = 7$, correspondendo às sete classes presentes no conjunto original.
- **"HTRU2"**[12]: contém características de pulsos de rádio de estrelas de nêutrons, serão utilizados 8 atributos numéricos. O valor de k será definido como 2, distinguindo entre pulsos gerados por pulsares e outros sinais.
- **"Image Segmentation"**[13]: fornece características extraídas de imagens para segmentação. Serão analisados 19 atributos de características de imagem, com $k = 7$, alinhado às sete classes de texturas presentes no conjunto.
- **"Banknote Authentication"**[14]: contém variáveis obtidas da imagem de cédulas, utilizando transformada wavelet para determinar a autenticidade. Serão utilizados 4 atributos numéricos, com $k = 2$, distinguindo entre cédulas autênticas e falsas.

4.2. Conjuntos de Dados Sintéticos

Para os dados sintéticos, foram gerados conjuntos de dados utilizando duas abordagens distintas: A primeira abordagem consistiu em exemplos fornecidos pela documentação da biblioteca *Scikit-Learn* [2], que fornece exemplos de dados para segregação em grupos[3].

A segunda abordagem, por sua vez, envolveu a geração de dados em duas dimensões usando a distribuição normal multivariada, a partir de ferramentas da biblioteca *NumPy*[1]. Nesse caso, os pontos foram amostrados em torno de um determinado número de centros, com diferentes médias para cada centro. O desvio padrão foi controlado para que a sobreposição entre os grupos variasse entre inexistente até altamente sobrepostos, gerando uma quantidade razoável de pontos em cada centro.

5. Implementação

Foram implementados três algoritmos para a resolução do problema dos k-centros utilizando a linguagem de programação *Python 3*, fazendo uso de funções pré-existentes de bibliotecas como *NumPy* e *Scikit-Learn* para operações matriciais e para o cálculo das métricas de avaliação supracitadas. Além disso, ocorreu a implementação de funções auxiliares para o desenvolvimento dos algoritmos em questão.

5.1. Funções Auxiliares

Foram implementadas funções auxiliares como `minkowski_dist` e `pairwise_distances`, as quais são descritas a seguir:

- `minkowski_dist`: função responsável por calcular a distância de `Minkowsky` entre dois pontos no espaço multidimensional, sendo esta parametrizada $p >= 1$, que altera a forma pela qual a distância é calculada. Caso $p = 1$, a função se transforma na distância `Manhattan` e, caso $p = 2$, a função equivale à distância Euclidiana.

- `pairwise_distances`: função que calcula, para n pontos, uma matriz $n \times n$ de distâncias entre os pontos, fazendo uso da função anterior. A implementação dessa função tem importância devido ao reuso da matriz de distâncias pelos algoritmos, que só precisa ser calculada uma vez.

5.2. Algoritmo de Escolha de Centros para Maximizar a Distância

Este algoritmo visa selecionar *k* centros de forma a maximizar a distância mínima entre qualquer ponto do conjunto e os centros selecionados. A implementação começa escolhendo o primeiro ponto como centro inicial e, em seguida, itera para selecionar os centros subsequentes. A cada iteração, o próximo centro é escolhido como o ponto mais distante dos centros já selecionados. Esse processo continua até que *k* centros sejam escolhidos.

5.3. Algoritmo com Refinamento do Intervalo do Raio Ótimo

Este algoritmo busca melhorar a seleção dos *k* centros ao refinar a estimativa do raio ótimo da solução em um intervalo de valores possíveis. A ideia central é usar uma abordagem de busca binária para iterativamente reduzir o intervalo que contém o raio ótimo: inicialmente, a o intervalo [0, r_{max}] contém o raio ótimo e, para cada iteração, o intervalo é reduzido pela metade. A quantidade de iterações é parametrizada, a fim de analisar a eficácia do algoritmo para diferentes refinamentos.

5.4. Algoritmo de Força Bruta

O algoritmo de força bruta retorna a solução ótima e foi implementado com o intuito de analisar a qualidade da solução dos algoritmos aproximativos. Este teve sua implementação baseada no algoritmo *K-Means* disponível na biblioteca *Scikit-Learn*.

6. Resultados e Discussão

6.1. Comparação de Resultados

Os resultados obtidos pelos algoritmos foram organizados em tabelas e gráficos, que permitem uma análise detalhada da performance entre o algoritmo 2-aproximado e o K-Means. A tabela a seguir apresenta uma comparação entre o tempo de execução, o raio da solução, e as métricas de silhueta e índice de Rand ajustado para os três algoritmos testados: o algoritmo de força bruta, o algoritmo 2-aproximado e o K-Means.

Algorithm Type	Radius Mean	Radius Std	Silhouette Mean	Silhouette Std
approx1	227.94	830.53	0.3197	0.1470
approx2_5	388.24	1217.22	0.2547	0.2612
approx2_10	305.60	981.35	0.2402	0.1934
approx2_15	267.17	905.12	0.2303	0.1629
approx2_20	247.18	874.84	0.2300	0.1648
approx2_25	243.75	875.30	0.2356	0.1587
np	294.96	978.60	0.3937	0.1039

Tabela 1. Médias e desvios-padrão de Radius e Silhouette por tipo de algoritmo

Os resultados mostram que, à medida que o refinamento do segundo algoritmo aproximativo aumenta, a média do raio encontrado fica mais próxima do valor obtido pelo

Algorithm Type	Rand Index Mean	Rand Index Std	Duration Mean	Duration Std
approx1	0.3069	0.1963	0.0234	0.0216
approx2_5	0.3284	0.2921	0.0026	0.0013
approx2_10	0.3629	0.2631	0.0033	0.0012
approx2_15	0.3489	0.2492	0.0050	0.0027
approx2_20	0.3436	0.2475	0.0063	0.0051
approx2_25	0.3615	0.2331	0.0066	0.0026
np	0.3706	0.2480	0.0047	0.0044

Tabela 2. Médias e desvios-padrão de Rand Index e Duration por tipo de algoritmo

algoritmo de força bruta. Entretanto, seu tempo de execução também aumenta, tornando-se até mesmo mais custoso que o algoritmo de forã bruta. Em geral, o algoritmo de força bruta apresenta tempos de execução significativamente maiores, especialmente quando comparado ao primeiro algoritmo 2-aproximado (*approx1*), que também apresenta valores mais distantes do ótimo.

7. Conclusão

Os resultados obtidos indicam uma clara compensação entre a precisão da solução e o tempo de execução nos algoritmos aproximativos para o problema dos k-centros. À medida que o refinamento do segundo algoritmo aproximativo é aumentado, observa-se que o raio médio da solução encontrada se aproxima cada vez mais do valor ótimo obtido pelo algoritmo de força bruta. No entanto, essa melhoria na qualidade da solução vem acompanhada de um custo computacional elevado, fazendo com que o tempo de execução do segundo algoritmo aproximativo ultrapasse o do algoritmo de força bruta em alguns casos.

Por outro lado, o primeiro algoritmo 2-aproximado se destaca por sua eficiência, apresentando tempos de execução significativamente menores em comparação tanto ao segundo algoritmo quanto ao algoritmo de força bruta. No entanto, essa eficiência vem ao custo de uma precisão reduzida, com os valores do raio da solução encontrados distantes do ótimo.

Em suma, a escolha do algoritmo mais adequado depende dos requisitos específicos do problema em questão. Se a prioridade for a precisão da solução, o segundo algoritmo aproximativo, com refinamentos, se mostra uma opção válida, mesmo com o aumento do tempo de execução. Contudo, se a eficiência computacional for crucial, o primeiro algoritmo aproximativo oferece uma solução aceitável com um custo de tempo significativamente menor.

8. Referências

Referências

[1] NumPy (n.d.) *Reference*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans

[2] Scikit-learn (n.d.) *KMeans clustering*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans

[3] Scikit-learn (n.d.) *Cluster comparison example*. https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py

[4] UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/; acesso em 15 de agosto de 2024. Wesley, Massachusetts, 2nd ed.

[5] E. Alpaydin and C. Kaynak (1998) *Pen-Based Recognition of Handwritten Digits*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

[6] P. W. Frey and D. J. Slate (1991) *Letter Recognition*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

[7] M. Koklu and I. A. Ozkan (2020) *Dry Bean Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset

[8] I. Cinar and M. Koklu (2019) *Rice (Cammeo and Osmancik) Image Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Rice+(Cammeo+and+Osmancik)

[9] R. M. Mohammad, F. Thabtah, and L. McCluskey (2015) *Phishing Websites Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

[10] E. Alpaydin and C. Kaynak (1998) *Optical Recognition of Handwritten Digits*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[11] Jason Catlett (1993) *Statlog (Shuttle) Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)

[12] R. Lyon, R. Brooke-Smith, M. Reece, A. Knowles, and S. Stappers (2010) *HTRU2 Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/372/htru2

[13] Vision Group, University of Massachusetts (1990) *Image Segmentation Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Image+Segmentation

[14] I. M. Güler and E. D. Übeyli (2005) *Banknote Authentication Dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/267/banknote+authentication