Trabalho Prático
Algoritmos aproximativos para o problema dos K-Centros

Kayque M. Siqueira[1], **Mateus A. Gomes**[2]

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

{mateusaugustoo100@gmail.com}
{kayque.original@hotmail.com}

Resumo. *Este artigo apresenta os resultados obtidos no desenvolvimento desse projeto ao analisar o desempenho de dois algoritmos aproximados - o primeiro baseado em ajustes do raio ótimo e o seguinte em abordagem maximizadora - no quesito de tempo de execução e qualidade da solução, entre outros aspectos. Serão utilizados conjuntos de dados reais e de dados sintéticos para a avaliação, considerando diferentes métricas, como o raio da solução, coeficiente de silhueta e índice de Rand ajustado (ARI). Os algoritmos foram implementados na linguagem Python 3. Por fim, os resultados obtidos serão analisados e discutidos, destacando as vantagens e limitações das abordagens implementadas.*

1. Introdução

O problema de k-center consiste em selecionar uma quantidade k de pontos (centros) em um conjunto de dados de n pontos, de modo que a maior distância entre um ponto e seu centro mais próximo seja minimizada. Esse problema é conhecido por ser NP-difícil, ou seja, não existe um algoritmo determinístico conhecido que possa resolvê-lo em tempo polinomial. Uma abordagem comum para problemas dessa natureza é o uso de algoritmos aproximativos, que oferecem soluções dentro de um fator aceitável de qualidade em relação à solução ótima.

Neste trabalho, serão abordados dois algoritmos 2-aproximados para o problema do k-centros. Para validar a eficácia dos algoritmos, realizamos experimentos em cerca de 30 conjuntos de dados, sendo alguns obtidos do UCI Machine Learning Repository (UCI MACHINE LEARNING REPOSITORY, 2024) e outros gerados de forma sintética, utilizando o Scikit Learn (SCIKIT-LEARN, 2024) e a distribuição normal multivariada. Além disso, variamos o parâmetro p da função de distância de Minkowski, utilizando os valores p=1 e p=2 que equivalem, respectivamente, a distância Manhattan e Euclidiana, com o objetivo de analisar o impacto dessas variações no desempenho dos algoritmos. Cada conjunto de dados foi submetido a numerosas execuções, divididas entre os dois algoritmos 2-aproximados e o algoritmo clássico K-Means, permitindo uma comparação detalhada em termos de qualidade da solução e tempo computacional.

2. Bases de dados

2.1. UCI Machine Learning Repository

Escolhemos ao todo 10 bases de dados presentes no site da UCI Machine Learning Repository (refenciado ao final deste documento). Os critérios de escolha desses repositórios foram os seguintes:

- Ter mais de 700 instâncias de dados;
- Ter no mínimo duas features numéricas para serem relacionadas
- Ter no mínimo um target para ser usado como label de classificação dos dados

2.2. Sklearn

Geramos ao todo 8 conjuntos de dados utilizando a biblioteca do Sklearn. Utilizamos três funções diferentes dessa biblioteca para gerar os dados necessários para análise, são elas: *make_blobs()*, *make_circles()* e *make_moons()*. A distribuição dos conjuntos de dados foi feita da seguinte forma:

- **Número de instâncias de pontos geradas para todas:** 500.
- **Blobs:**
  - 2 conjuntos de dados obtidos através da função *make_blobs()* com diferentes *seeds*.
  - 2 conjuntos de dados obtidos através da função *make_blobs()* com diferentes *seeds* e desvios padrão.
- **Moons:**
  - 2 conjuntos de dados obtidos através da função *make_moons()* com diferentes ruídos e *seeds*.
- **Circles:**
  - 2 conjuntos de dados obtidos através da função *make_circles()* com diferentes fatores, ruídos e *seeds*.

2.3. Dados Sintéticos

Como último conjunto de dados, geramos um total de 30 bases de dados genéricas utilizando uma função da biblioteca *numpy* que realiza uma distribuição normal multivariada. Todos os dados foram gerados com 5 centros e 200 pontos ao redor de cada centro, totalizando conjuntos de 800 pontos. Esses dados foram divididos em três grupos, com base no valor do desvio padrão:

1. **Desvio padrão: 0.5** — Grupos com sobreposição consideravelmente baixa.
2. **Desvio padrão: 1.5** — Grupos com sobreposição moderada.
3. **Desvio padrão: 3** — Grupos altamente sobrepostos.

Essa variação nos desvios padrões foi feita de maneira intencional para garantir uma diferença significativa na sobreposição entre os grupos, desde ”inexistente”até ”altamente sobrepostos”.

3. Testes

Para cada uma das três bases de dados, executamos 30 vezes cada algoritmo utilizando dois parâmetros: $p = 1$ (distância Manhattan) e $p = 2$ (distância Euclidiana). Além disso, executamos o algoritmo K-means uma vez para cada base. Para o algoritmo de ajustes, foram realizados mais 5 conjuntos de 30 testes para cada instância, onde avaliamos a largura do intervalo em que o raio ótimo se encontra (1-25%).

Esse mesmo procedimento foi aplicado a cada uma das 30 instâncias dos dados sintéticos (10 amostras com 3 desvios padrão diferentes), totalizando **6788** testes realizados.

4. Resultados

4.1. Greedy Algorithm 2-approximation

Aqui estão listados os resultado obtidos após a execução de todos os testes com o algoritmo de abordagem gulosa. Foi tirada a média dos 30 testes e cada linha nas tabelas seguintes representam dados com esses médias calculadas.

Para os dataframes obtidos na base de dados **UCI Machine Learning Repository**, esses foram os resultados:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means
UCI_1.txt	1.0	7.0	14.1425	0.0341	0.8031	0.2827	20.0271	0.0096
UCI_1.txt	2.0	7.0	13.6852	0.0319	0.8936	0.2837	20.0271	0.0101
UCI_2.txt	1.0	2.0	258.3108	0.0057	0.6780	0.6594	234.7540	0.0087
UCI_2.txt	2.0	2.0	253.1171	0.0067	0.6714	0.6537	234.7540	0.0086
UCI_3.txt	1.0	2.0	4428.3072	0.0080	0.4499	0.1056	4453.6799	0.0090
UCI_3.txt	2.0	2.0	4573.5100	0.0079	0.4296	0.0689	4453.6799	0.0081
UCI_4.txt	1.0	2.0	825.5482	0.0020	0.6781	0.0175	1193.3114	0.0099
UCI_4.txt	2.0	2.0	833.4786	0.0019	0.6790	0.0132	1193.3114	0.0098
UCI_5.txt	1.0	7.0	0.5486	0.1138	0.5491	0.0019	1.0226	0.0127
UCI_5.txt	2.0	7.0	0.4215	0.1043	0.5530	0.0006	1.0226	0.0130
UCI_6.txt	1.0	2.0	68.6073	0.0045	0.5599	-0.0044	73.4063	0.0091
UCI_6.txt	2.0	2.0	53.3667	0.0045	0.6194	-0.0050	73.4063	0.0091
UCI_7.txt	1.0	3.0	70.8828	0.0177	0.6609	0.0043	59.6650	0.0103
UCI_7.txt	2.0	3.0	55.4412	0.0173	0.6814	0.0027	59.6650	0.0105
UCI_8.txt	1.0	2.0	10.2731	0.0029	0.7097	0.0827	7.4438	0.0084
UCI_8.txt	2.0	2.0	7.5707	0.0029	0.6618	0.1157	7.4438	0.0085
UCI_9.txt	1.0	7.0	2.8350	0.0296	0.7072	0.0072	2.3862	0.0104
UCI_9.txt	2.0	7.0	2.1263	0.0310	0.7055	0.0060	2.3862	0.0108
UCI_10.txt	1.0	5.0	25.7931	0.0075	0.7854	0.0813	31.5460	0.0107
UCI_10.txt	2.0	5.0	21.5948	0.0074	0.4267	0.0867	31.5460	0.0123

Tabela 1. Resultados Base UCI - Greedy Algorithm

Para a **base de dados sintética criada com a distribuição normal multivariada**, esses foram os resultados obtidos para 4 das 10 amostras:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means	Desvio padrão
sample1_sqtd0.5.txt	1.0	5.0	8.7489	0.0068	0.8129	0.9978	4.2149	0.0104	0.5
sample1_sqtd0.5.txt	2.0	5.0	6.7666	0.0068	0.8164	0.9979	4.2149	0.0100	0.5
sample1_sqtd1.5.txt	1.0	5.0	8.1026	0.0064	0.5591	0.7445	4.6635	0.0053	1.5
sample1_sqtd1.5.txt	2.0	5.0	6.2998	0.0069	0.5977	0.7518	4.6635	0.0054	1.5
sample1_sqtd3.0.txt	1.0	5.0	8.3013	0.0067	0.6585	0.7943	3.9313	0.0055	3.0
sample1_sqtd3.0.txt	2.0	5.0	6.2052	0.0068	0.6734	0.8846	3.9313	0.0049	3.0
sample2_sqtd0.5.txt	1.0	5.0	7.8842	0.0065	0.6888	0.7493	3.9947	0.0094	0.5
sample2_sqtd0.5.txt	2.0	5.0	5.9464	0.0070	0.6829	0.8070	3.9947	0.0110	0.5
sample2_sqtd1.5.txt	1.0	5.0	8.7636	0.0067	0.7897	0.9818	4.0585	0.0051	1.5
sample2_sqtd1.5.txt	2.0	5.0	6.7670	0.0065	0.7837	0.9450	4.0585	0.0051	1.5
sample2_sqtd3.0.txt	1.0	5.0	9.2854	0.0067	0.7139	0.8165	3.6223	0.0053	3.0
sample2_sqtd3.0.txt	2.0	5.0	6.6051	0.0063	0.7179	0.8107	3.6223	0.0052	3.0
sample3_sqtd0.5.txt	1.0	5.0	7.5717	0.0066	0.6050	0.7334	4.0219	0.0095	0.5
sample3_sqtd0.5.txt	2.0	5.0	6.2588	0.0063	0.5977	0.7420	4.0219	0.0088	0.5
sample3_sqtd1.5.txt	1.0	5.0	7.9490	0.0066	0.6627	0.8579	3.7493	0.0050	1.5
sample3_sqtd1.5.txt	2.0	5.0	5.9007	0.0066	0.6895	0.9045	3.7493	0.0052	1.5
sample3_sqtd3.0.txt	1.0	5.0	9.8184	0.0064	0.8705	0.9981	3.9736	0.0050	3.0
sample3_sqtd3.0.txt	2.0	5.0	7.1489	0.0067	0.8776	0.9990	3.9736	0.0050	3.0
sample4_sqtd0.5.txt	1.0	5.0	7.5539	0.0067	0.6124	0.7458	3.6134	0.0099	0.5
sample4_sqtd0.5.txt	2.0	5.0	5.9397	0.0068	0.5896	0.7453	3.6134	0.0095	0.5
sample4_sqtd1.5.txt	1.0	5.0	8.3917	0.0067	0.6895	0.8321	3.8883	0.0053	1.5
sample4_sqtd1.5.txt	2.0	5.0	6.2343	0.0080	0.6981	0.8961	3.8883	0.0068	1.5
sample4_sqtd3.0.txt	1.0	5.0	8.3795	0.0071	0.6541	0.7223	3.8970	0.0062	3.0
sample4_sqtd3.0.txt	2.0	5.0	6.2411	0.0073	0.6717	0.8108	3.8970	0.0067	3.0

Tabela 2. Resultados Base Sintética - Greedy Algorithm

Por fim, para a **base de dados do SKlearn**, esses foram os resultados obtidos pelo Greedy Algorithm:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means
blobs1.txt	1.0	3.0	7.1301	0.0020	0.8769	0.6435	4.2143	0.0084
blobs1.txt	2.0	3.0	5.4205	0.0019	0.8115	0.6712	4.2143	0.0091
blobs2.txt	1.0	3.0	10.6267	0.0019	0.7310	0.5005	8.0412	0.0100
blobs2.txt	2.0	3.0	9.3848	0.0020	0.6972	0.5520	8.0412	0.0091
moons1.txt	1.0	2.0	2.0753	0.0011	0.7193	0.7567	1.0099	0.0088
moons1.txt	2.0	2.0	1.5676	0.0010	0.6413	0.6011	1.0099	0.0083
moons2.txt	1.0	2.0	2.0763	0.0011	0.6174	0.7478	1.1949	0.0088
moons2.txt	2.0	2.0	1.6821	0.0009	0.5213	0.8648	1.1949	0.0090
noisy_circles1.txt	1.0	2.0	1.8754	0.0012	0.5087	0.8172	1.1543	0.0093
noisy_circles1.txt	2.0	2.0	1.4404	0.0010	0.5307	0.7046	1.1543	0.0084
noisy_circles2.txt	1.0	2.0	2.0924	0.0011	0.5994	0.7290	1.1954	0.0086
noisy_circles2.txt	2.0	2.0	1.5873	0.0010	0.5223	0.7897	1.1954	0.0083
varied1.txt	1.0	3.0	11.7451	0.0020	0.4886	0.6867	8.0412	0.0092
varied1.txt	2.0	3.0	8.6561	0.0020	0.4934	0.6510	8.0412	0.0098
varied2.txt	1.0	3.0	7.8049	0.0021	0.6170	0.7130	4.2182	0.0086
varied2.txt	2.0	3.0	5.5574	0.0018	0.6540	0.7883	4.2182	0.0087

Tabela 3. Resultados Base SKlearn - Greedy Algorithm

4.2. Adjust Algorithm 2-approximation

Aqui estão listados alguns dos resultado obtidos após a execução de todos os testes com o algoritmo que aproxima o raio ótimo. **Foi tirada a média dos 30 testes** e cada linha nas tabelas seguintes representam dados com esses médias calculadas. Para esse algoritmo também foi necessário calcular os intervalos de refinamento com o raio ótimo, logo o número de resultados obtidos é expressivo. Sendo assim, visando a legibilidade e o entendimento, os dados presentes nas seguintes tabelas são de apenas algumas das bases de dados de cada tipo. Foram escolhidos **5 intervalos de refinamento**, são eles: 0.01, 0.03, 0.05, 0.08 e 0.16.

Para os dataframes obtidos na base de dados **UCI Machine Learning Repository**, esses foram alguns dos resultados:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means	Refinamento(%)
UCIDF1.txt	1.0	7.0	48.2008	0.4278	0.4389	0.2963	20.0271	0.0096	0.01
UCIDF1.txt	1.0	7.0	51.0062	0.3334	0.4358	0.3038	20.0271	0.0047	0.03
UCIDF1.txt	1.0	7.0	45.0469	0.2534	0.4840	0.3037	20.0271	0.0046	0.05
UCIDF1.txt	1.0	7.0	16.8492	0.1962	0.5075	0.2952	20.0271	0.0047	0.08
UCIDF1.txt	1.0	7.0	24.2570	0.1223	0.5008	0.2462	20.0271	0.0047	0.16
UCIDF1.txt	2.0	7.0	49.9018	0.3979	0.4224	0.3008	20.0271	0.0048	0.01
UCIDF1.txt	2.0	7.0	44.3001	0.3149	0.4633	0.3100	20.0271	0.0045	0.03
UCIDF1.txt	2.0	7.0	39.3779	0.2434	0.4992	0.3041	20.0271	0.0046	0.05
UCIDF1.txt	2.0	7.0	14.7820	0.1888	0.5172	0.3022	20.0271	0.0046	0.08
UCIDF1.txt	2.0	7.0	26.3352	0.1133	0.4995	0.2384	20.0271	0.0045	0.16
UCIDF2.txt	1.0	2.0	361.5644	0.2553	0.5604	0.5353	234.7540	0.0094	0.01
UCIDF2.txt	1.0	2.0	353.6194	0.2234	0.6016	0.5970	234.7540	0.0040	0.03
UCIDF2.txt	1.0	2.0	354.5075	0.1908	0.6179	0.6126	234.7540	0.0040	0.05
UCIDF2.txt	1.0	2.0	268.1141	0.1488	0.7134	0.7946	234.7540	0.0035	0.08
UCIDF2.txt	1.0	2.0	251.1753	0.1205	0.7563	0.8674	234.7540	0.0035	0.16
UCIDF2.txt	2.0	2.0	350.1300	0.2554	0.5703	0.5751	234.7540	0.0035	0.01
UCIDF2.txt	2.0	2.0	320.7998	0.2188	0.6367	0.6701	234.7540	0.0037	0.03
UCIDF2.txt	2.0	2.0	324.1737	0.1846	0.6398	0.6718	234.7540	0.0037	0.05
UCIDF2.txt	2.0	2.0	305.2909	0.1521	0.6880	0.7339	234.7540	0.0037	0.08
UCIDF2.txt	2.0	2.0	244.8023	0.1247	0.7608	0.8764	234.7540	0.0037	0.16
UCIDF6.txt	1.0	2.0	85.4032	0.1917	0.3260	0.0043	73.4063	0.0083	0.01
UCIDF6.txt	1.0	2.0	90.8695	0.1737	0.3072	0.0030	73.4063	0.0040	0.03
UCIDF6.txt	1.0	2.0	89.7093	0.1500	0.3114	0.0041	73.4063	0.0040	0.05
UCIDF6.txt	1.0	2.0	85.3868	0.1244	0.3243	0.0029	73.4063	0.0041	0.08
UCIDF6.txt	1.0	2.0	86.2271	0.1030	0.3366	0.0120	73.4063	0.0043	0.16
UCIDF6.txt	2.0	2.0	77.0360	0.1937	0.3303	0.0030	73.4063	0.0040	0.01
UCIDF6.txt	2.0	2.0	79.1716	0.1761	0.3217	0.0024	73.4063	0.0039	0.03
UCIDF6.txt	2.0	2.0	80.4160	0.1493	0.3192	0.0012	73.4063	0.0041	0.05
UCIDF6.txt	2.0	2.0	74.5742	0.1289	0.3721	0.0040	73.4063	0.0041	0.08
UCIDF6.txt	2.0	2.0	67.5482	0.1053	0.3777	0.0040	73.4063	0.0042	0.16

Tabela 4. Resultados Base Sintética - Adjust Algorithm

Para a **base de dados sintética criada com a distribuição normal multivariada**, o algoritmo de ajustes forneceu esses resultados para 3 das 10 amostras:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means	Refinamento(%)
sample1.txt	1.0	5.0	15.9402	0.1960	0.4227	0.6275	4.6635	0.0156	0.01
sample1.txt	1.0	5.0	14.5070	0.1577	0.4086	0.6091	4.6635	0.0068	0.03
sample1.txt	1.0	5.0	13.4704	0.1384	0.4762	0.6977	4.6635	0.0075	0.05
sample1.txt	1.0	5.0	7.9832	0.0947	0.6385	0.8588	4.6635	0.0068	0.08
sample1.txt	1.0	5.0	7.1683	0.0729	0.6375	0.8071	4.6635	0.0069	0.16
sample1.txt	2.0	5.0	13.6157	0.1827	0.4067	0.5939	4.6635	0.0066	0.01
sample1.txt	2.0	5.0	14.0044	0.1582	0.3987	0.5961	4.6635	0.0074	0.03
sample1.txt	2.0	5.0	9.9305	0.1286	0.4948	0.6954	4.6635	0.0074	0.05
sample1.txt	2.0	5.0	6.0585	0.0926	0.6252	0.8398	4.6635	0.0070	0.08
sample1.txt	2.0	5.0	5.7639	0.0652	0.7315	0.7698	4.6635	0.0062	0.16
sample4.txt	1.0	5.0	20.7859	0.1601	0.4434	0.6455	3.8883	0.0104	0.01
sample4.txt	1.0	5.0	17.5076	0.1384	0.4924	0.6544	3.8883	0.0052	0.03
sample4.txt	1.0	5.0	12.9771	0.1190	0.6131	0.7756	3.8883	0.0058	0.05
sample4.txt	1.0	5.0	8.3032	0.0874	0.7547	0.9503	3.8883	0.0050	0.08
sample4.txt	1.0	5.0	7.3957	0.0641	0.7720	0.8947	3.8883	0.0056	0.16
sample4.txt	2.0	5.0	15.7297	0.1590	0.4930	0.6351	3.8883	0.0052	0.01
sample4.txt	2.0	5.0	15.7751	0.1340	0.4681	0.6613	3.8883	0.0053	0.03
sample4.txt	2.0	5.0	12.2837	0.1147	0.5701	0.7647	3.8883	0.0053	0.05
sample4.txt	2.0	5.0	6.8012	0.0906	0.7329	0.9278	3.8883	0.0055	0.08
sample4.txt	2.0	5.0	6.4246	0.0610	0.7665	0.7967	3.8883	0.0053	0.16
sample7.txt	1.0	5.0	15.9516	0.1339	0.2588	0.4387	4.6446	0.0097	0.01
sample7.txt	1.0	5.0	14.7961	0.1113	0.2713	0.4491	4.6446	0.0057	0.03
sample7.txt	1.0	5.0	7.3376	0.0897	0.3793	0.5405	4.6446	0.0057	0.05
sample7.txt	1.0	5.0	6.5775	0.0713	0.4127	0.5673	4.6446	0.0055	0.08
sample7.txt	1.0	5.0	7.5314	0.0439	0.4799	0.3501	4.6446	0.0055	0.16
sample7.txt	2.0	5.0	10.2791	0.1374	0.3113	0.4715	4.6446	0.0056	0.01
sample7.txt	2.0	5.0	9.0929	0.1152	0.3059	0.4762	4.6446	0.0057	0.03
sample7.txt	2.0	5.0	8.5769	0.0908	0.3385	0.5158	4.6446	0.0056	0.05
sample7.txt	2.0	5.0	5.0230	0.0682	0.4057	0.5612	4.6446	0.0054	0.08
sample7.txt	2.0	5.0	6.2945	0.0417	0.5759	0.3435	4.6446	0.0056	0.16

Tabela 5. Resultados Base UCI - Adjust Algorithm

Por fim, para a **base de dados do SKlearn**, esses foram os resultados obtidos pelo Greedy Algorithm:

Nome do Arquivo	P	K	Raio	Tempo de execução	Silhueta	Índice de Rand	Raio do K-Means	Tempo do K-Means	Refinamento(%)
blobs1.txt	1.0	3.0	9.1081	0.0774	0.4372	0.565	4.2143	0.0109	0.01
blobs1.txt	1.0	3.0	8.8211	0.0649	0.4267	0.6282	4.2143	0.0053	0.03
blobs1.txt	1.0	3.0	9.9414	0.0531	0.3717	0.5147	4.2143	0.005	0.05
blobs1.txt	1.0	3.0	7.5708	0.0394	0.4994	0.7225	4.2143	0.005	0.08
blobs1.txt	1.0	3.0	6.8204	0.033	0.5925	0.8318	4.2143	0.0062	0.16
blobs1.txt	2.0	3.0	7.4212	0.079	0.3985	0.5717	4.2143	0.0048	0.01
blobs1.txt	2.0	3.0	7.1773	0.0654	0.408	0.5705	4.2143	0.0047	0.03
blobs1.txt	2.0	3.0	6.8681	0.0527	0.4663	0.6418	4.2143	0.0052	0.05
blobs1.txt	2.0	3.0	6.5546	0.0418	0.4919	0.6854	4.2143	0.0052	0.08
blobs1.txt	2.0	3.0	4.6458	0.0303	0.6175	0.8846	4.2143	0.005	0.16
noisy_$circles1.txt$	1.0	2.0	2.106	0.0597	0.2513	0.0202	1.1543	0.0097	0.01
noisy_$circles1.txt$	1.0	2.0	2.0656	0.0556	0.2553	0.0265	1.1543	0.007	0.03
noisy_$circles1.txt$	1.0	2.0	2.0494	0.0465	0.2547	0.0101	1.1543	0.0059	0.05
noisy_$circles1.txt$	1.0	2.0	2.0016	0.0414	0.2535	0.0146	1.1543	0.0062	0.08
noisy_$circles1.txt$	1.0	2.0	2.0511	0.0311	0.2551	0.0275	1.1543	0.0057	0.16
noisy_$circles1.txt$	2.0	2.0	1.5642	0.0613	0.2546	0.018	1.1543	0.0049	0.01
noisy_$circles1.txt$	2.0	2.0	1.5535	0.0523	0.256	0.021	1.1543	0.0048	0.03
noisy_$circles1.txt$	2.0	2.0	1.6752	0.0439	0.2605	0.0199	1.1543	0.0052	0.05
noisy_$circles1.txt$	2.0	2.0	1.5335	0.0369	0.2586	0.0223	1.1543	0.0062	0.08
noisy_$circles1.txt$	2.0	2.0	1.5312	0.0283	0.2553	0.0174	1.1543	0.005	0.16
moons2.txt	1.0	2.0	2.5458	0.0579	0.3912	0.2686	1.1949	0.0096	0.01
moons2.txt	1.0	2.0	2.4226	0.0491	0.3692	0.2642	1.1949	0.0048	0.03
moons2.txt	1.0	2.0	2.4059	0.0404	0.3659	0.2201	1.1949	0.0046	0.05
moons2.txt	1.0	2.0	2.5156	0.0338	0.386	0.221	1.1949	0.005	0.08
moons2.txt	1.0	2.0	2.3969	0.0257	0.3841	0.2718	1.1949	0.0047	0.16
moons2.txt	2.0	2.0	1.9082	0.057	0.3848	0.2669	1.1949	0.0049	0.01
moons2.txt	2.0	2.0	1.9512	0.0471	0.3659	0.2352	1.1949	0.0049	0.03
moons2.txt	2.0	2.0	1.8549	0.0393	0.3716	0.2028	1.1949	0.0047	0.05
moons2.txt	2.0	2.0	1.8027	0.0308	0.3964	0.2811	1.1949	0.0046	0.08
moons2.txt	2.0	2.0	1.7952	0.028	0.4036	0.254	1.1949	0.0063	0.16

Tabela 6. Resultados Base SKlearn - Adjust Algorithm

O restante dos resultados obtidos estão separados de forma organizada no **repositório desta pesquisa**. Para uma análise mais detalhada, pessoal e completa dos dados, investigue os resultados armazenados.

5. Análises e Conclusões

Com a figura 1, alocada após as referências, é possível observar o desempenho dos dois algoritmos 2-aproximados em relaçao ao K-Means da biblioteca sklearn.

É possível observar que o algoritmo 2-aproximado baseado em maximização consegue encontrar soluções melhores que o K-Means em algumas bases de dados. Isso se deve ao fato de algumas particularidades na distribuição dos pontos. Por exemplo, se houvessem dois focos de pontos, um com 500 pontos em seu entorno e outro com 499, o algoritmo 2-aproximado em questão selecionaria um centro no foco com mais pontos e o centro seguinte iria para o segundo foco. Assim, o raio da soluçao seria bem proximo do ideial e faria esse algoritmo ter uma disputa acirrada com o K-Means para essa base. Portanto, apesar de ser precipitado dizer que o K-Means perde para tal aproximação, tudo depende da distribuição dos pontos.

Com o desenvolvimento desse projeto, é possível destacar pontos importantes em relação à qualidade e ao desempenho dos algoritmos, tais como:

O algoritmo 2-aproximado que aproxima o raio ótimo tem um tempo de execução maior. Esse mesmo algoritmo encontra soluções piores nas bases de dados construidas a partir da distribuição normal multivariada.

O K-Means é mais rápido e possui maior qualidade de solução em um cenário geral. Para instâncias especificas, em que os pontos estão distribuidos como em vários focos, o algoritmo 2-aproximado que busca maximizar soluções parciais apresenta um resultado final satisfatório e eficiente. Por exemplo, na figura 2, temos 5 focos principais onde há maior concentração de pontos. Assim, o algoritmo baseado em maximização encontrou uma ótima solução.

Referências

SCIKIT-LEARN. **Plot Cluster Comparison**. Acesso em: 15 ago. 2024. 2024. Disponível em: ¡`https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py`¿.

UCI MACHINE LEARNING REPOSITORY. **UCI Machine Learning Repository**. Acesso em: 15 ago. 2024. 2024. Disponível em: ¡`https://archive.ics.uci.edu/`¿.

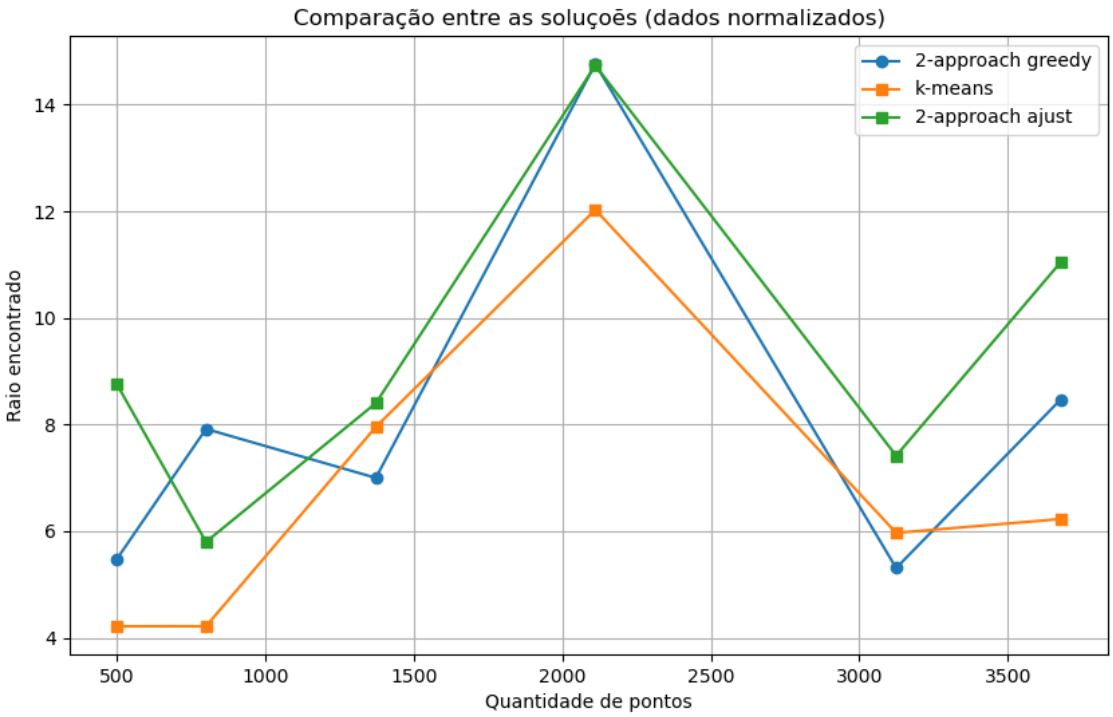

Figura 1. Comparação dos algoritmos

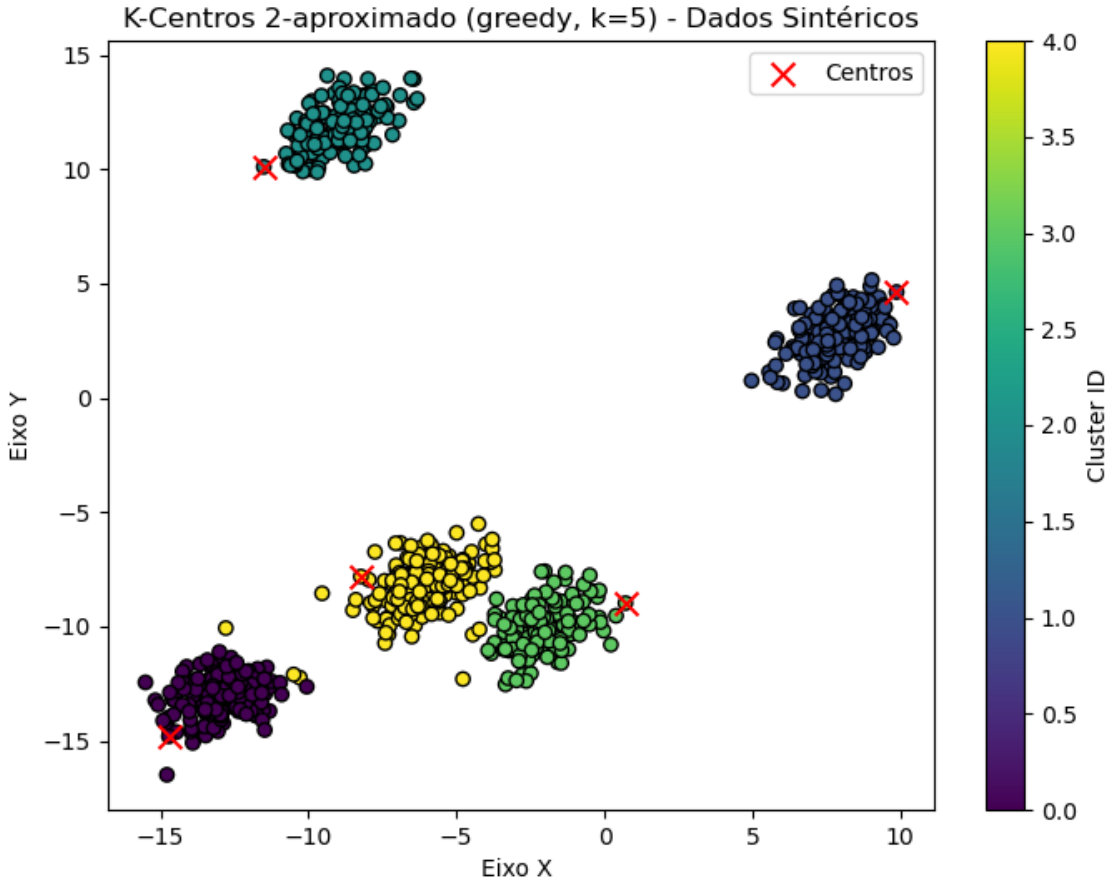

Figura 2. K-Centros 2-aproximado