TRABALHO PRÁTICO 2

Gabriel Martins Medeiros Fialho - 2020006540

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

Resumo. *O objetivo do trabalho prático é implementar um algoritmo para buscar uma solução 2-aproximada para o problema dos K-CENTROS, no qual temos um conjunto de pontos como entrada e buscamos retornar partições desse conjunto utilizando centros com o objetivo de minimizar o raio máximo dos centros.*

1. Introdução

Para a implementação do algoritmo, foi necessário implementar também o algoritmo da distância de minkowski, que é uma generalização para as distâncias euclidiana e de manhattan. Utilizando essa generalização foi possível construir uma matriz nxn de distâncias entre todos os pontos, permitindo um funcionamento mais rápido do algoritmo guloso para resolver o problema dos k-centros, que foi implementado logo a seguir. Foram realizados testes em 10 datasets, utilizando duas colunas de atributos numéricos de cada um e os resultados foram impressos na tela: desenho dos pontos e centros no plano cartesiano, raio, silhueta, indice de Rand ajustado e tempo de execução para 30 testes diferentes.

2. Detalhamento

A função de distância de minkowski foi implementada utilizando funções vetoriais da biblioteca NumPy, como foi especificado. A função é definida por:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}.$$

Utilizando numpy temos: pow(np.sum(np.power(np.absolute(x-y),p)), 1/float(p))

Para cada dataset, foram feitas matrizes de distâncias tanto para p=1 quanto para p=2

Essas matrizes foram encontradas iterando sobre um array obtido do dataframe em questão. O loop calcula, para cada ponto, sua distância até cada ponto do array, incluindo o próprio ponto, cuja distância é zero. Dessa forma, é possível utilizar a matriz de distâncias para acelerar o processo de obtenção dos novos centros durante o algoritmo guloso.

A função que executa o posicionamento dos k-centros foi implementada utilizando o algoritmo guloso 2-aproximativo visto em sala, que permite um fator de aproximação 2 mesmo sem executar a busca binária.

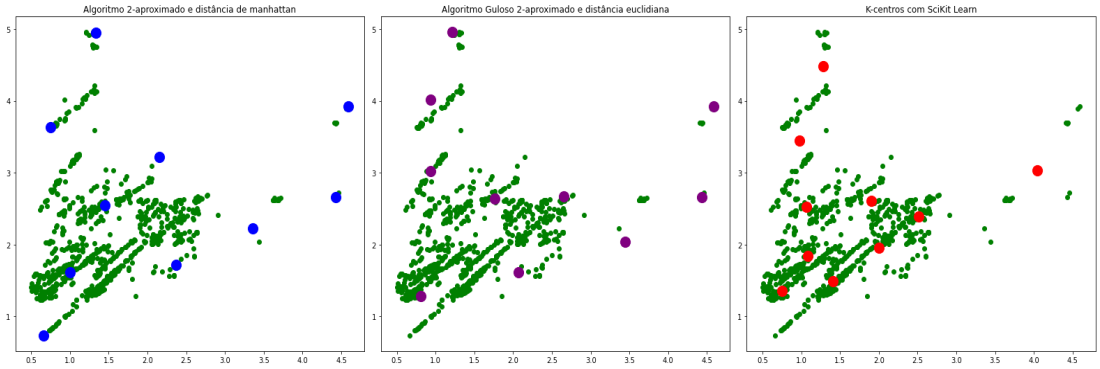
O algoritmo implementado tem o seguinte funcionamento:

- Recebemos o conjunto de pontos, o número k de centros desejados e uma matriz de distância calculada utilizando a distância de minkowski., podendo ser tanto a distância de manhattan quanto a distância euclidiana
- Caso o k seja maior ou igual ao número de pontos, a solução é o próprio conjunto de pontos
- Caso contrário, escolhemos um ponto arbitrário e procuramos o ponto mais distante desse ponto arbitrário e adicionamos ao conjunto de centros
- Com isso, procuramos um ponto que tenha a maior distância mínima ao conjunto atual de centros, iterando sobre todos os pontos fora do centro e calculando a distância desses pontos até cada ponto do centro. Pegando para cada ponto a distância mínima entre o ponto e os centros e depois escolhendo o ponto que obteve a maior distância mínima ao centro, teremos uma boa opção para adicionar ao conjunto de centros.
- Quando atingimos nosso número máximo de centros, o conjunto de centros é retornado, assim como seus índices, o raio obtido durante a execução e também um array label, que relaciona cada ponto ao centro mais próximo, o que será importante para medir a silhueta e índice de Rand ajustado no futuro

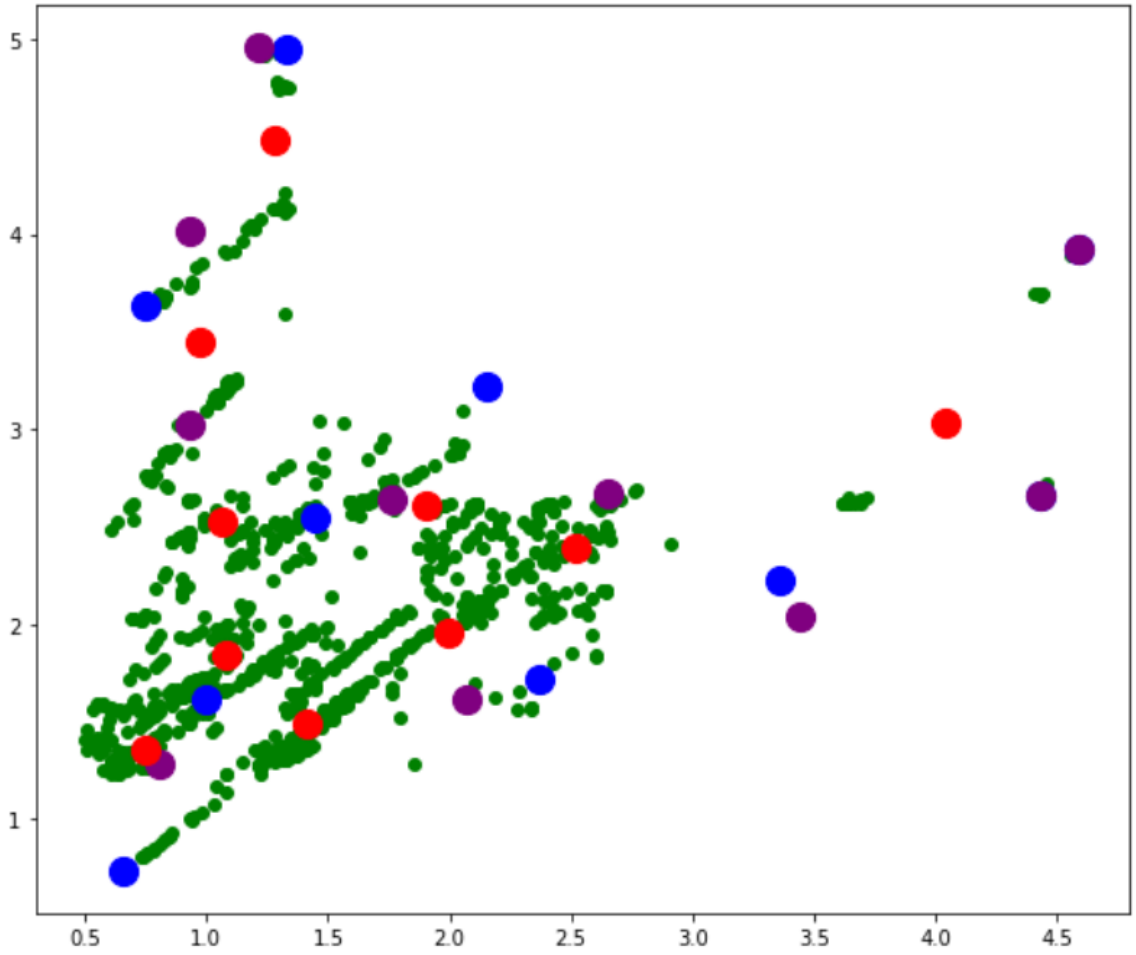
3. Visualização do algoritmo

Primeiramente, foi feita a plotagem dos pontos e dos centros em um plano cartesiano para fins de análise. A partir desse momento já é possível perceber o funcionamento do algoritmo, comparando-o com o algoritmo do SciKit Learn. Ambos atingem resultados plausíveis.
Primeiramente, para cada dataframe, é feito o cálculo das matrizes de distância que irão auxiliar o algoritmo dos k-centros

Após isso, a função encontra_k_centros é executada, retornando as coordenadas dos centros, os índices desses centros no dataframe, no array e na matriz de distâncias, o raio máximo dos centros e um array label, que relaciona cada ponto ao seu centro. Após obtidos esses valores, é feita a plotagem das amostras do dataframe, juntamente com os k-centros retornados pelo algoritmo guloso, além da plotagem dos k-centros retornados pelo algoritmo da biblioteca SciKit Learn. Os plots são feitos lado a lado, mostrando cada retorno, respectivamente para o algoritmo 2-aproximativo utilizando distância de manhattan, euclidiana e o algoritmo do SciKit Learn. A partir da visualização dos dados, já é posssível perceber o bom funcionamento do algoritmo, que, assim como o algoritmo da biblioteca SciKit Learn, parece retornar resultados aparentemente plausíveis, com os centros bem distribuidos:



Também é impressa a visualização dos 3 algoritmos em um mesmo plot, para visualizar melhor as semelhanças entre eles:



4. Testes

A arbitrariedade da escolha dos centros iniciais afeta diretamente o desempenho e precisão do algoritmo. Por isso, para cada conjunto de dados, foram feitos 30 testes para analisar as métricas e resultados do algoritmo. Foram feitos 30 testes para cada dataframe, sendo que 15 desses testes utilizam p = 1, ou seja, a distância de manhattan e os outros 15 utilizam p = 2, a distância euclidiana. Para cada rodada do algoritmo, foi medido o seu raio, a sua silhueta, seu índice de rand ajustado e seu tempo de execução.

Para evitar poluição no relatório, apenas os resultados de média e desvio padrão serão mostrados. Como o desvio padrão é baixo em todos os casos, é possível perceber que o algoritmo foi consistente. Os resultados completos podem ser analisados no notebook.

Dataframe 1

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	0.519558	0.074949	0.313097	0.044861	0.430549	0.06965	0.105534	0.005587

Dataframe 2

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	0.024112	0.002756	0.557847	0.086824	0.384457	0.08839	0.128731	0.034721

Dataframe 3

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	3.632493	0.595878	0.215568	0.036292	0.192471	0.088808	0.076339	0.004249

Dataframe 4

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	0.605221	0.079654	0.313387	0.040788	0.447151	0.106536	0.077492	0.003726

Dataframe 5

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	5.728345	0.645462	0.440583	0.055234	0.500877	0.063434	0.078905	0.005448

Dataframe 6

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	0.173458	0.024718	0.243198	0.049036	0.348659	0.066567	0.07708	0.003851

Dataframe 7

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	10.623507	1.497862	0.312737	0.026854	0.42388	0.065056	0.078185	0.005282

Dataframe 8

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	194.614416	25.106923	0.312655	0.035321	0.481748	0.072216	0.078505	0.003524

Dataframe 9

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	0.102296	0.011204	0.356401	0.031083	0.450129	0.047927	0.078379	0.004475

Dataframe 10

	Média Raio	Desvio Padrão Raio	Média Silhueta	Desvio Padrão Silhueta	Média Indice de Rand	Desvio Padrão Rand	Média de Tempo	Desvio Padrão de Tempo
0	57.306126	7.902398	0.257213	0.013852	0.419376	0.04529	0.078623	0.004628

5. Conclusão

Sabemos que o problema dos k-centros é um problema NP-difícil. Nesse trabalho, foi possível encontrar uma implementação que obtém uma boa aproximação em tempo polinomial, que encontra uma solução nunca pior do que o dobro da solução ótima verdadeira. Esse é um bom exercício para entrar em contato com o mundo dos problemas NP-Difíceis, usando a abordagem de algoritmos aproximativos, que é uma das maneiras de lidar com esses problemas. Se não podemos encontrar uma solução ótima, barata e rápida para todo caso, podemos utilizar uma solução quase ótima, barata e rápida, o que pode ser uma excelente abordagem em alguns casos. O trabalho trata, em especial, de uma abordagem gulosa para a resolução desses problemas.

6. References

Basic statistics in pandas DataFrame | by Kasia Rachuta | Medium

Silhouette Coefficient. This is my first medium story, so… | by Ashutosh Bhardwaj | Towards Data Science

#108: Scikit-learn 105:Unsupervised Learning 9: Intuition for Clustering evaluation - YouTube

k-means Elbow Method and Silhouette Method - YouTube

sklearn.metrics.adjusted_rand_score — scikit-learn 1.1.1 documentation

Microsoft PowerPoint - AM_Aula06.pptx - AM_Aula06.pdf

7. Conjuntos de dados utilizados:

google_review_ratings.csv

online_shoppers_intention.csv

dow_jones_index.zip

UCI Machine Learning Repository: Gesture Phase Segmentation Data Set

e-shop data and description.zip

Index of /ml/machine-learning-databases/wine-quality

egll

new

RF-PC

Software