TP2 - Relatório

Nome: Rodrigo Ferreira Araújo — Matrícula: 2020006990

Algoritmos 2 - Julho 2022

1 Introdução

Esse trabalho prático, implementado em Python, visa aplicar de o algoritmo 2 - aproximativo para resolver o problema de clustering: isto é, com k centros, agrupar n pontos de modo que cada grupo fique mais perto possível de seu respectivo centro. Além disso, por meio de métricas de qualidade como tempo de execução, silhueta, índice de Rand ajustado e raio máximo da solução, iremos comparar o método nativo da biblioteca scikit learn para a resolução do clustering (K-means) com a abordagem aproximativa e analisar os resultados.

2 Métodos e Métricas

- dist(X, Y, p): Método que define a métrica de distância entre pontos X e Y (de tamanho n_features da base de dados) de acordo com a distância de Minkowski:

$$(\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

- get_dist_matrix(data, p): Calcula e armazena a matriz de distância de todos os pontos para todos os pontos, sob a lógica da distância de Minkowski.

- approx_kmeans(dist_matrix, k): Sob a posse da matriz de distâncias, podemos aplicar a abordagem aproximativa para o problema de clustering: Começamos com um conjunto C de centros vazio e uma escolha arbitrária para um centro. A partir daí, enquanto tivermos menos do que k centros,

iremos escolher um novo centro no dataset que maximize a distância para com todos os centros presentes em C até então. Este método retorna os índices dos centros e dos pontos restantes, bem como um vetor de novas classificações (labels) para cada ponto de acordo com o centro mais próximo.

- get_max_radius(Centers, indexes, new_labels, dist_matrix): Retorna o raio máximo do problema, ou seja, o máximo dentre os raios máximos dos pontos com relação aos seus centros.

Em seguida, no código fonte, são feitas etapas preparativas para os experimentos: Está incluso o carregamento e tratamento individual dos datasets, o que inclui a separação e tratamento das colunas de classes, tratamento de valores não numéricos e o cálculo da matriz de distâncias (para efeitos de comparação de tempos de execução, o grande gargalo gerado para o cálculo desta matriz foi desconsiderado na etapa de experimento) e a decisão do valor de k para o dataset.

3 Experimentos

Roteiro: Com os dados necessários para o experimento, podemos testar e comparar ambos os métodos (aproximado e nativo do scikit) para clusterizar diferentes datasets. Para cada dataset, iremos fazer 30 iterações para cada método, armazenar os resultados (tempo de execução, raio da solução, silhueta e índice de Rand) e compará-los ao final.

Obs: Para efeitos de comparação, como cada uma das 30 iterações da abordagem produz resultados métricos diferentes (mas próximos), iremos considerar as médias das 30 instâncias de métricas geradas.

3.1 dataset 1:

Link: https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset

Classes: "Kecimen" = 0, "Besni" = 1.

Tempo de Processamento da Matriz de distâncias: 4min e 39 seg.

Métricas	Scikit KMeans	Kmeans aproximado (valores médios)
Tempo de Execução (seg)	0.1338	0.017
Raio Máximo do Problema	130279.273	191830.156
Silhueta	0.654	0.637
Índice de Rand Ajustado	0.166	0.075

3.2 dataset 2:

Link: https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work

Classes: Educação: "High school" = 1, "Graduate" = 2, "Postgraduate" = 3, "Master And Doctor" = 4.

Tempo de Processamento da Matriz de distâncias: 2min 23seg.

Métricas	Scikit KMeans	Kmeans aproximado (valores médios)
Tempo de Execução (seg)	0.0621	0.016
Raio Máximo do Problema	121.10	147.09
Silhueta	0.507	0.363
Índice de Rand Ajustado	-0.04	-0.004

3.3 dataset 3:

Link: https://archive.ics.uci.edu/ml/datasets/
Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+

Classes: Nível de obesidade: "Peso Insuficente" = 0, "Peso Normal" = 1, "Sobre Peso I" = 2, "Sobre Peso II" = 3, "Obesidade Nível I" = 4, "Obesidade Nível II" = 5, "Obesidade Nível III" = 6.

Tempo de Processamento da Matriz de distâncias: 25min 23seg.

Métricas	Scikit KMeans	Kmeans aproximado (valores médios)
Tempo de Execução (seg)	0.230	0.033
Raio Máximo do Problema	45.47	54.18
Silhueta	0.482	0.281
Índice de Rand Ajustado	0.287	0.149

3.4 dataset 4:

Link: https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records

Classes: Tem Diabetes?: "Sim" = 1, "Não" = 0

Tempo de Processamento da Matriz de distâncias: 1min 23seg.

Métricas	Scikit KMeans	Kmeans aproximado (valores médios)
Tempo de Execução (seg)	0.125	0.022
Raio Máximo do Problema	26.37	30.69
Silhueta	0.329	0.451
Índice de Rand Ajustado	0.243	0.198

4 Conclusões:

4.1 Tempo de execução:

Para os datasets analisados, houve uma perceptível, mas não significativa, diferença no tempo de execução do processamento dos K centros. Descartando-se a criação da matriz de distâncias, a lógica simplificada e gulosa, característica da abordagem aproximada adotada, obteve tempos menores.

4.2 Raio da solução:

Pode-se observar, agora, que a abordagem do scikit learn obteve resultados melhores quanto ao raio máximo dos clusters em comparação com a média dos resultados do algoritmo aproximado. Apesar das 30 iterações ("número mágico"), a solução do algoritmo aproximado converge fracamente para a solução do scikit learn.

4.3 Silhueta:

A silhueta mede a qualidade da clusterização realizada analisando o quão próximos estão os pontos de um cluster para o seu centro. Nesse sentido, a abordagem do scikit se sai melhor novamente, com valores mais próximos de 1 (indicam uma boa segregação de clusters) comparado com valores menores da abordagem aproximada.

4.4 Índice de Rand Ajustado:

Conclusão semelhante à anterior, contudo o índice de rand compara a proximidade entre dois clusters diferentes (o setado pelas classes do dataset e o obtido após a previsão do Kmeans). Nesse sentido, os valores de Rand obtidos por meio da abordagem do scikit mostraram-se levemente mais próximos de 1, isto é, estão prevendo "melhor" os clusteres em comparação às classes fornecidas pelo dataset