Relatório Projeto 2 – Clustering com Minimum Spanning Tree (MST)

Eric Macedo Cabral
NUSP: 10653262
3 de junho de 2018

I. Descrição do problema

Realizar o agrupamento dos pontos do dataset proposto em [3] utilizando os métodos gulosos de Minimum Spanning Tree (MST) com os algoritmos de Kruskal [4] e Prim [5].

II. Especificações Técnicas da Solução

- Linguagem de programação: Python v3.6.5
- Dependências (pip):
  - *NumPy*[1] – Biblioteca de computação científica
  - *Matplotlib*[2] – Biblioteca de renderização de gráficos
  - *Tabulate*[3] – Biblioteca de tabulação para saída de dados no console
  - *scikit-learn*[4] – Biblioteca de aprendizado de máquina

Além dos dois métodos gulosos sugeridos, com o objetivo de fazer um comparativo entre as diferentes abordagens de clustering, foi implementado o algoritmo KMeans, que é um algoritmo especifico de clustering. A complexidade deste algoritmo não será analisada.

As métricas apresentadas na Tabela I foram obtidas numa máquina com as seguintes especificações:

- Intel Core i5-4402E
  - Intel HD Graphics 4400
- Memória RAM DDR3L 8GB (1600 MHz)
- Linux 4.14.47-1-MANJARO

III. Implementação

Na implementação dos algoritmos foram utilizadas as estruturas de dados nativas da linguagem Python, como também a biblioteca de computação científica NumPy. A utilização do NumPy justifica-se por esta biblioteca conseguir manipular grandes volumes de dados de forma mais otimizada que as implementações nativas do Python.

A forma utilizada nos três métodos para calcular a distancia entre dois vértices foi a norma Euclidiana:

$$E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Em ambas implementações (Prim e Kruskal), existe um passo extra que é passo de normalizar a classificação dos grupos encontrados para valores entre 1 e k. Ambos com complexidade assintótica de $O(n\ log\ n)$.

[1]http://www.numpy.org/
[2]https://matplotlib.org/
[3]https://pypi.org/project/tabulate/
[4]http://scikit-learn.org/

A. Kruskal

A implementação do algoritmo de Kruskal possui **complexidade assintótica de** $O(n^2)$, uma vez que precisam ser calculadas as distâncias entre todos vértices.

O espaço necessário para armazenar as distâncias calculada foi reduzido pela metade ao evitar a repetição de arestas que já foram calculadas, ao calcular a aresta do vértice A até o vértice B, não calcula-se a aresta do vértice B até o vértice A. O espaço e os cálculos são reduzidos de n^2 para $\frac{n^2}{2}$. Porém, isso não diminui a complexidade do algoritmo, visto que assintoticamente, $\frac{n^2}{2}$ é o mesmo que n^2. Após calcular a o vetor de distâncias, esse vetor é ordenado com custo $O(m\ log\ m)$, sendo m uma soma dos termos de uma progressão aritmética:

$$m = (n-1) * \frac{(a_1 + a_{n-1})}{2}$$

O algoritmo começa com n grupos. Cada operação de *union*, decrementa-se o número de grupo até que existam apenas k grupos.

Na operação de *find* do algoritmo, foi implementada a compressão de caminho de caminho com o fim de diminuir o número de iterações realizadas até encontrar o grupo. Isso faz a operação ter uma complexidade assintótico de $\Omega(1)$ e $O(log\ n)$.

B. Prim

Para implementar a *heap* binária do algoritmo de Prim, foi utilizada a biblioteca nativa do Python *Heapq*[5], com inserção de custo $O(log\ n)$. Após, é utilizada a biblioteca *Random* para embaralhar os vértices empilhados, com custo $O(n)$.

Por precisar calcular todas as distâncias entre os vértices do dataset, o algoritmo possui **complexidade assintótica de** $O(n^2)$. Poderia ser utilizada uma matriz de memória para evitar a repetição de cálculos de distância, mas isso não diminuiria a complexidade assintótica, como visto no algoritmo de Kruskal na Seção III-A.

IV. Resultados Obtidos

Com as implementações apresentadas na Seção III, foram obtidas as métricas da Tabela I. As métricas utilizadas foram o Coeficiente de Silhueta [1] e o Rand Index [2]. Ambas são métricas amplamente adotadas em ciência de dados. Esses

[5]https://docs.python.org/2/library/heapq.html

Tabela I
MÉTRICAS DE CLUSTERING

	Coeficiente de Silhueta	Rand Index	Tempo(médio)
Kruskal	0.282498	0.804207	3.91593s
Prim	0.282498	0.804207	4.28601s
KMeans	0.487793	0.74588	0.71739s

dados foram obtidos com o auxilio da biblioteca de aprendizagem de máquina *scikit-learn*.

A métrica de tempo foi calculada pelo tempo médio de 10 execuções dos algoritmos.

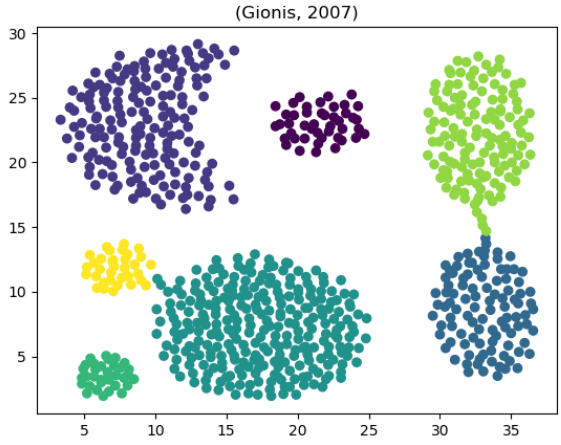

Figura 1. Fonte: Elaborado pelo autor com a biblioteca Matplotlib

A Figura 1 apresenta o clustering ideal proposto por [3]. O clustering obtido pelos métodos gulosos de Kruskal e Prim apresentam a mesma estrutura, como pode ser observado nas Figuras 2 e 3. Na Figura 4, o clustering do método KMeans.

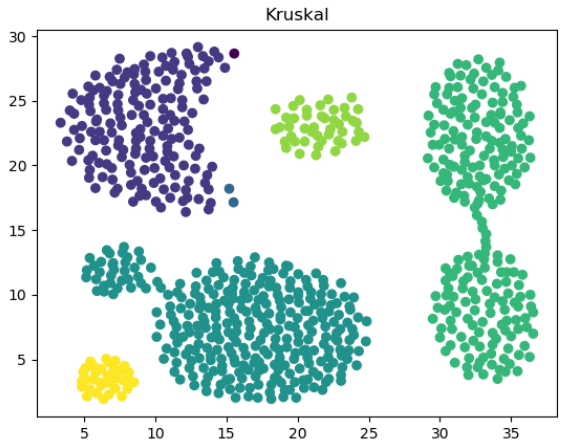

Figura 2. Fonte: Elaborado pelo autor com a biblioteca Matplotlib

Os métodos gulosos são métodos determinísticos, logo terão sempre o mesmo resultado.

Observa-se que os métodos gulosos possuem a dificuldade de separar clusters diferentes com pontos muito próximos,

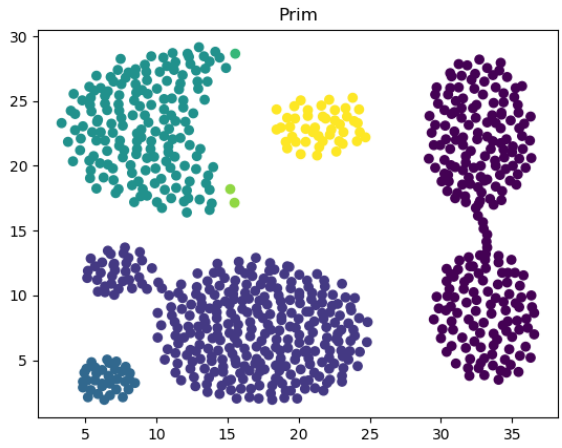

Figura 3. Fonte: Elaborado pelo autor coma a biblioteca Matplotlib

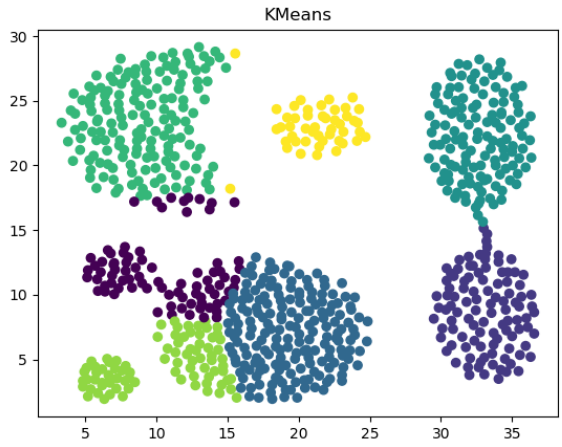

Figura 4. Fonte: Elaborado pelo autor com a biblioteca Matplotlib

esse é um comportamento esperado dado a implementação de remover as maiores arestas. Consequentemente, os algoritmos gulosos acabam gerando clusters muito pequenos com 2 ou menos vértices, ou clusters muito grande que idealmente deveriam ser 2 ou mais cluster. Já o método KMeans, tende a agrupar pontos de acordo com a densidade de pontos, o que resolve o problema presente nos métodos gulosos.

REFERÊNCIAS

[1] Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65

[2] Rand, W. M. Objective criteria for the evaluation of clustering methods (1971). Journal of the American Statistical Association. American Statistical Association. 66 (336): 846–850.

[3] Gionis, A., H. Mannila, and P. Tsaparas, Clustering aggregation. ACM Transactions on Knowledge Discovery from Data (TKDD), 2007. 1(1): p. 1-30.

[4] Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem (1956). Proceedings of the American Mathematical Society. 7: 48–50.

[5] Prim, R. C. Shortest connection networks And some generalizations(1957). Bell System Technical Journal, 36 (6): 1389–1401.