Árvores Geradoras Mínimas aplicadas a Agrupamento de Dados

Igor Lúcio Rocha Alves - Matrícula: 3902
Nálbert Wattam Silva Mariano - Matrícula: 4856
Nelson Cândido Martins Junior - Matrícula: 3906

20 de Novembro de 2018

1 Introdução e Objetivos

Algoritmos para induzir Árvore Geradora Mínima (do inglês, *Minimum Spanning Tree)* a partir de um grafo *G(V, A)* são amplamente utilizados em vários cenários de aplicação. Um deles é conhecido como Agrupamento de Dados.

Agrupamento de Dados (ou *Clustering*) é uma técnica de análise de padrões em dados que descreve um conjunto de objetos em uma coleção de grupos (*clusters*) distintos. O objetivo dessa técnica é determinar que os itens que são mais parecidos entre si sejam colocados em um mesmo grupo. A figura a seguir mostra uma classificação de dados após agrupamento.

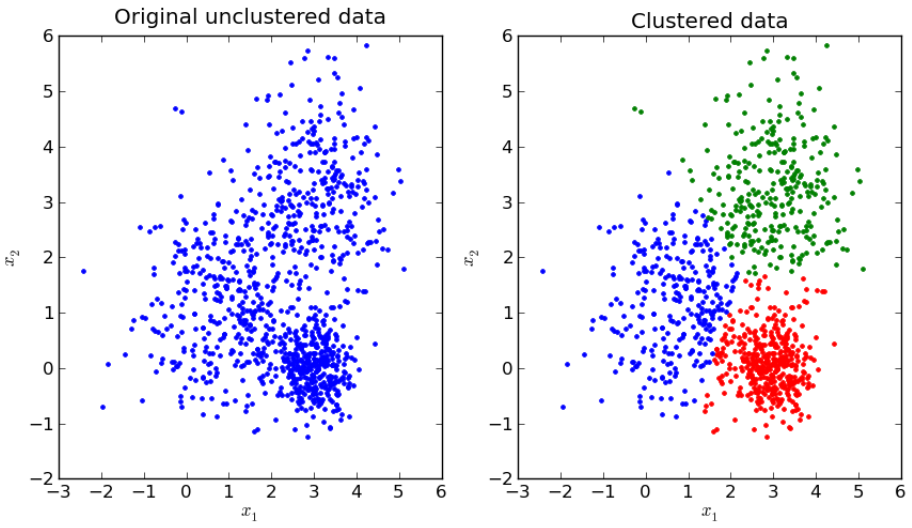

Figura 1: Exemplo de *data clustering* [1]

2 Árvores geradoras de custo mínimo

As árvores geradoras mínimas (MST) vêm, durante muito tempo, fazendo parte do interesse de matemáticos por conta da sua grande variedade de aplicações. Uma das aplicações mais comuns utilizando MST são o cabeamento utilizado por empresas de comunicações, que procuram realizar essas conexões da maneira mais simples e barata possível, onde o custo de realizar o cabeamento corresponde aos pesos das arestas. [2]

Uma MST consiste em encontrar a árvore geradora de um grafo com o menor custo possível e esse grafo deve ser não direcionado e ponderado, ou seja, com custo em suas arestas. Considerando que H seja uma árvore mínima de G é necessário provar duas propriedades separadas: H é uma

árvore mínima de G e H é a árvore com o menor custo de G. A árvore mínima H também precisa ser ser totalmente conectada e acíclica. [3]

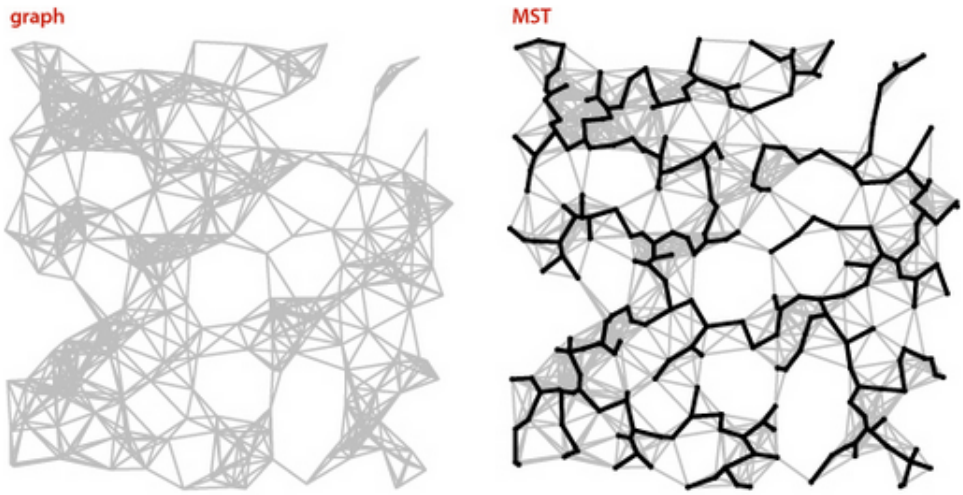

Figura 2: Comparativo: Grafo X Árvore Geradora Mínima [4]

2.1 Algoritmo de Prim

O algoritmo de Prim encontra a árvore geradora mínima de um grafo identificando a aresta com o menor peso para o vértice inicial e prosseguindo a partir desse vértice. [5]

A cada passo iterativo o algoritmo busca a aresta com o menor peso e a adiciona a MST que está sendo criada junto com o novo vértice que se conecta ao vértice que já faz parte da MST. O procedimento é completado quando todos os vértices presentes no grafo original também estão presentes na árvore geradora mínima criada. Como no algoritmo de Kruskal o de Prim também deve gerar uma árvore acíclica e totalmente conectada.

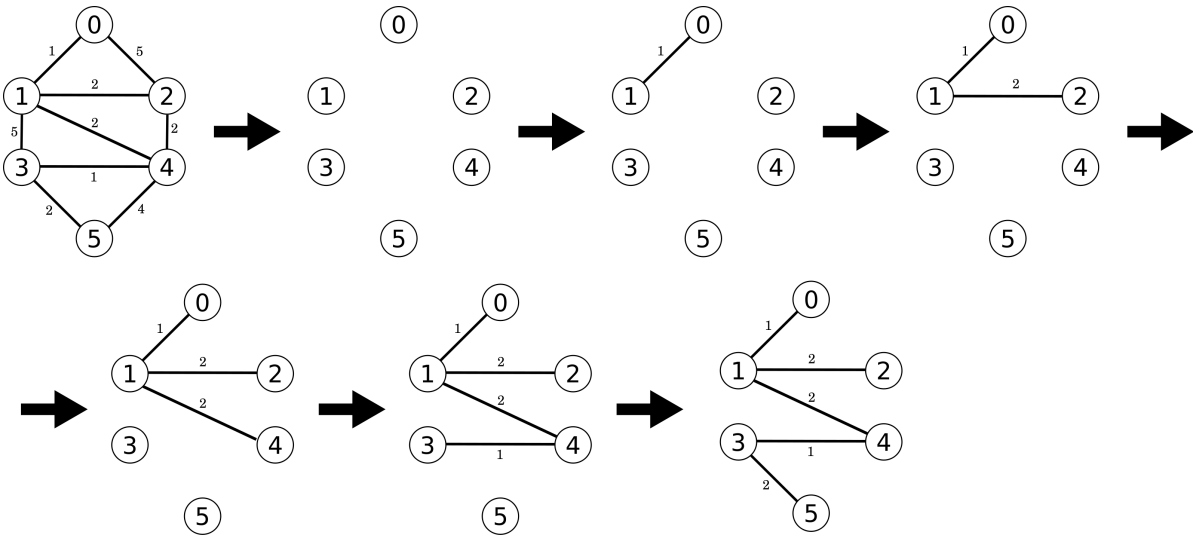

Figura 3: Execução do algoritmo de Prim: passo a passo [6]

2.2 Algoritmo de Kruskal

O algoritmo de Kruskal é um algoritmo guloso que procura a árvore geradora mínima de um grafo conectado e com peso em suas arestas. Consiste em alguns passos para alcançar a MST: [5]

1. Criar uma floresta F (um conjunto de árvores), onde cada vértice do grafo é uma árvore por si só;
2. Criar um conjunto A que contém todas as arestas presentes no grafo;
3. Enquanto o conjunto de arestas ainda não está vazio e a floresta ainda não se tornou uma única árvore:
   (a) Remove a aresta com o menor peso no conjunto A;
   (b) Se essa aresta conecta duas árvores presentes na floresta F ela é adicionada a floresta, combinando as duas árvores em uma só;
   (c) Caso essa aresta não consiga conectar duas árvores ela é descartada.

Após seguidos todos os passos o algoritmo entregará a árvore geradora mínima do grafo em questão.

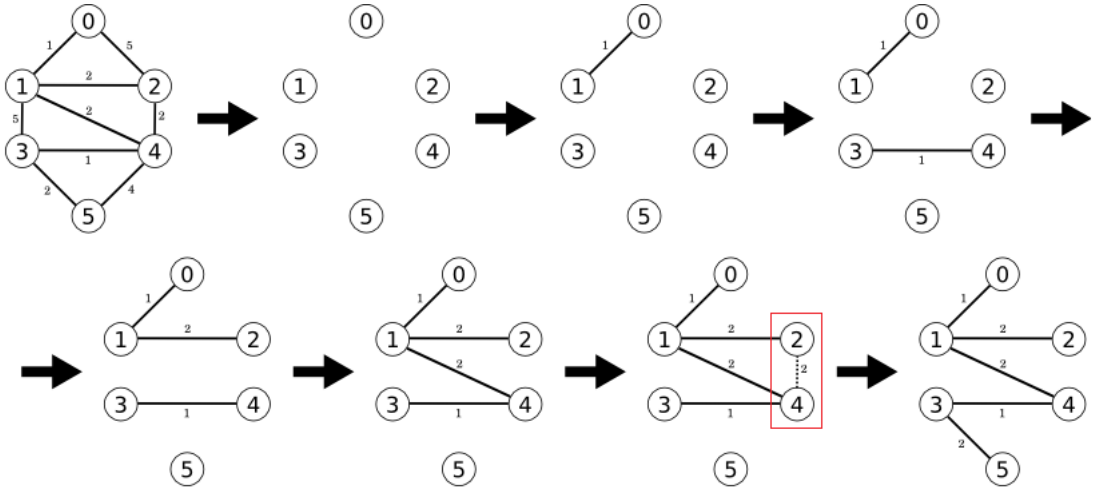

Figura 4: Execução do algoritmo de Kruskal: passo a passo [7]

2.3 Agrupamento dos Dados

Agrupamento de dados consiste em organizar os dados de acordo com as suas características. Cada grupo deve compartilhar características com os outros dados ali também presentes. A cada novo subgrupo o nível de semelhança entre os dados deve aumentar, se tornando cada vez mais homogêneos. [8]

O agrupamento de dados se trata de aprendizado não supervisionado enquanto o supervisionado se dá por meio da classificação de dados. No agrupamento de dados os dados são inferidos de acordo com a sua proximidade. Os grupos de dados (*clusters*) são aglomeração de pontos no espaço, onde esses pontos se encontram perto de outros pontos presentes no mesmo *cluster* e longe dos pontos fora deste. [9]

3 Materiais e Métodos

Os algoritmos estudados neste trabalho foram implementados sobre a linguagem de programação Java (versão 8) e avaliados em um *hardware* composto com processador Intel Core^{TM} i7-8550U, 8GB de RAM e 240GB de SSD (*Solid State Drive*).

Para avaliar os algoritmos, foi utilizado uma entrada contendo 150 objetos (vértices) nos quais as arestas nele contidas representam arestas de um grafo completo ponderado.

4 Resultados

5 Conclusões

A resolução do problema, de maneira geral, foi satisfatória. Fora utilizado uma representação do grafo do tipo de lista de adjacências objetivando um ganho no desempenho computacional da execução do mesmo. Foram empregados estruturas de dados auxiliares na implementação dos problemas tais como: *Union Find* e fila de prioridade.

5.1 k-agrupamento por Prim

Usamos a ideia de fila de prioridade implementando a estrutura (*Min Heap*) para obter o peso mínimo e guardar suas referidas chaves (que seria como se fosse uma identificação do elemento em questão). O objetivo principal é percorrer todos os vértices do grafo da forma mais otimizada possível e armazenar os vértices na fila que ainda não foram incluídos na MST. Todas as operações de sua estrutura é $\mathcal{O}(\log V)$. Os passos detalhados são:

1. Crie um *Min Heap* de tamanho V onde V é o número de vértices no gráfico dado. Cada nó do *heap min* contém o número do vértice e o valor da chave do vértice.
2. Inicialize *Min Heap* com o primeiro vértice como raíz (o valor da chave atribuído ao primeiro vértice é 0). O valor da chave atribuído a todos os outros vértices é ∞ (ou um número grande).
3. Enquanto o *Min Heap* não estiver vazio, faça:
   (a) Extraia o nó do valor mínimo do *Min Heap*. Deixe o vértice extraído ser u.
   (b) Para cada vértice adjacente v de u, verifique se v está em *Min Heap* (ainda não incluído no MST). Se v estiver em *Min Heap* e seu valor de chave for maior que o peso de u-v, atualize o valor da chave de v como peso de u-v.

Olhando o código superficialmente o mesmo aparenta ser $\mathcal{O}(V^2)$ por ter dois laços *while* aninhados. Entretanto, se analisarmos bem podemos observar que as instruções no laço interno são executadas em $\mathcal{O}(V + A)$ vezes (semelhante a busca em largura). O laço mais interno possui a operação *decreaseKey()* que leva tempo de $\mathcal{O}(\log V)$. Portanto, a complexidade geral é $\mathcal{O}(A \log V)$.

5.2 k-agrupamento por Kruskal

Foi utilizado a estrutura de dados *Union Find* com o objetivo de detectar ciclos na árvore geradora mínima. Os passos detalhados são:

1. Classifique todas as bordas em ordem não decrescente do seu peso.
2. Escolha a menor aresta. Verifique se ele forma um ciclo com a árvore de abrangência formada até o momento. Se o ciclo não for formado, inclua essa aresta. Mais, descarte-o.
3. Repita o passo 2 até que haja arestas (V-1) na árvore de abrangência.

A ordenação das arestas leva o tempo $\mathcal{O}(A \log A)$. Após isso, iteramos em todas as arestas e aplicamos o algoritmo *Union Find*. As operações de *Find* e *Union* podem levar mais tempo; chegando a $\mathcal{O}(\log V)$. Portanto, a complexidade geral do tempo é $\mathcal{O}(A \log V)$.

Referências

[1] M. NK, "K-means clustering in python." https://mubaris.com/posts/kmeans-clustering/, 2017.

[2] P. Jayawant and K. Glavin, "Minimum spanning trees," *Involve, a Journal of Mathematics*, vol. 2, no. 4, pp. 439–450, 2009.

[3] J. C. Gower and G. J. Ross, "Minimum spanning trees and single linkage cluster analysis," *Applied statistics*, pp. 54–64, 1969.

[4] P. Feofiloff, "Árvores geradoras de custo mínimo." https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/mst.html, 2017.

[5] L. Najman, J. Cousty, and B. Perret, "Playing with kruskal: algorithms for morphological trees in edge-weighted graphs," in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 135–146, Springer, 2013.

[6] U. Ingeniería Informática, "Análisis de algoritmos." http://arantxa.ii.uam.es/~aa/practicas/P1/enunciadop1.html, 2010–2011.

[7] D. Henrique, "Algoritmo de kruskal: um exemplo." https://sites.google.com/site/inatelmaratona/categorias/grafos/arvore-geradora-minima, 2016.

[8] E. R. Hruschka, "Agrupamento de dados - introdução." ICMC USP, 2013.

[9] M. K. Albertini, "Agrupamento de dados - uma visão geral." FACOM/UFU, 2015.