Implementação de grafo sobre o conjunto de dados da flor *Iris*

Elian Babireski & Vinícios Bidin

5 de outubro de 2022

1 Introdução

Trata-se de um projeto que tem como proposta a implementação de um programa que receba como entrada um conjunto de dados sobre a flor *Iris*. O conjunto de dados apresenta informações sobre as medidas de comprimento e largura da sépala e da pétala de cento e cinquenta amostas de três espécies distintas da planta.

Em suma, cada amostra da planta deverá ser tratada como um ponto no espaço quadridimensional. Em seguida, deve-se calcular a distância euclidiana – conforme (1) – entre todos os pares de pontos.

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{1}$$

Calculadas as distâncias essas devem, em seguida, ser normalizadas por meio da equação abaixo (2). Por fim, deverá ser montado um grafo onde cada nodo corresponde a uma amostra, e uma aresta entre dois pontos existirá somente se a distância entre eles for menor ou igual a 0.3.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

2 Compilação

2.1 Dependências

São dependências do projeto, a biblioteca *Graphviz* sendo utilizada a linguagem *Dot* para a visualização do grafo gerado, utiliza-se ainda o *Makefile* para compilação de todo o projeto, para que não seja necessário ficar compilando os arquivos de forma individual.

2.2 Compilação

Uma vez no diretório do projeto, navegue para adentro, compile, utilizando *make*.

2.3 Execução

Para executar, basta executar o arquivo de saída chamado *main*, utilizando o comando *./main*. Para utilização, utilize as funcionalidades desejadas até que seja inserido um *0* para encerramento do programa. Os arquivos gerados ao se utilizar opções *4*, *5*, *6* do menu, são colocados no diretório *data/*. Para visualizar o grafo gerado, basta abrir o arquivo no formato *(Scalable Vector Graphics)*, chamado *graph.svg*.

3 Documentação

Abaixo encontra-se listada a documentação de todas as funções usadas na aplicação:

- `graph* create()`: cria e retorna um descritor da estrutura do grafo;
- `void addNode(graph *descriptor, sample sample)`: recebe um descritor de grafo e os dados de uma amostra e, em seguida, adiciona a amostra como um novo nodo do grafo;
- `void addEdge(graph *descriptor, int head, int tail)`: recebe um descritor de grafo e o índice de dois nodos e cria uma aresta entre eles;
- `void destroy(graph *descriptor)`: desaloca todo o grafo da memória;
- `float distance(sample p, sample q)`: calcula a distância euclidiana entre duas amostras;
- `float normalize(float x, float maximum, float minimum)`: recebe como entrada um valor e o normaliza com base no máximo e mínimo também fornecidos;
- `void readTxt(graph *descriptor)`: carrega o grafo de um arquivo .txt;
- `void readCsv(graph *descriptor)`: carrega os dados das amostras de um arquivo .cvs e gera os nodos;
- `void link(graph *descriptor)`: gera uma tabela com as distâncias normalizadas entre as amostras e, em seguida, gera as arestas entre os nodos de acordo com um limite definido (0.3);
- `void print(graph* graph)`: imprime o grafo no terminal;
- `void saveTxt(graph* graph)`: salva o grafo em um arquivo de texto;

- `void graphviz(graph *descriptor)`: gera um arquivo .dot que representa o grafo;
- `void plot()`: gera um arquivo .svg do grafo.

4 Resultados

Utilizado como limite o valor 0.3 percebeu-se que as *Setosas* quase formaram um grafo desconexo das demais plantas. As *Versicolors* e as *Virginicas*, entretanto, apresentaram muitas relações entre si. Após testes realizados pela equipe, concluiu-se que os resultados mais visualmente agradáveis ocorreram quando o limite estava ao redor de 0.15, porque assim o grupo das *Setosas* fica completamente desconexo do restante, e as outras duas espécies parecem um pouco mais separadas (embora ainda muito interconectadas).

Disso, pode-se concluir que as *Setosas* são mais facilmente diferenciáveis pelas dimensões da pétala e sépala do que as outras duas espécies de *Iris*.