TP2 - Algoritmos para Bioinformática I

Lucas Resende Pellegrinelli Machado - 2018126673

Setembro 2020

1 Introdução

Com o avanço da disponibilidade de dados no mundo, foi também necessária o desenvolvimento de diversas técnicas para nos auxiliar a entender como esses dados se comportam de forma mais eficiente.

No trabalho anterior foi discutido como usar o SVD para representar os dados em dimensões reduzidas para melhor entende-los. Nesse trabalho elevamos nossa capacidade de entender os dados disponibilizados usandos outros tipos de algoritmos em cima do SVD, particularmente métodos de agrupar esses pontos em dimensões reduzidas para entende-los melhores.

O tema do trabalho também é interessante visto que ele faz jús ao nome da disciplina e trata de como podemos trabalhar com genomas de primatas com as técnicas aprendidas em aula.

2 Entradas e saídas

De forma análoga ao primeiro trabalho, nossos métodos funcionam a partir de matrizes e vetores, porém agora essas entidades matemáticas não são dadas inicialmente e tem que ser criadas a partir de métodos de encoding de dados em outros formatos.

Os dados utilizados foram os disponibilizados pelo próprio Toolkit de bioinformática do matlab e consiste em genomas de 12 espécies de primatas e pode ser aberto com o seguinte código.

```
1 load primates
```

Inicialmente, transformaremos esses genomas importados em uma matriz por meio de um método chamado de vector space model que será explicado na seção de Metodologia. Após conseguir a matriz que representa o código genético de cada uma das espécies em questão, podemos então processar esses dados e gerar informações interesstantes.

O primeiro resultado a ser gerado por esse trabalho é um simples gráfico representando cada uma das espécies em um espaço 3D usando os valores singulares de um SVD para escolher quais os melhores pontos para explicar aqueles dados.

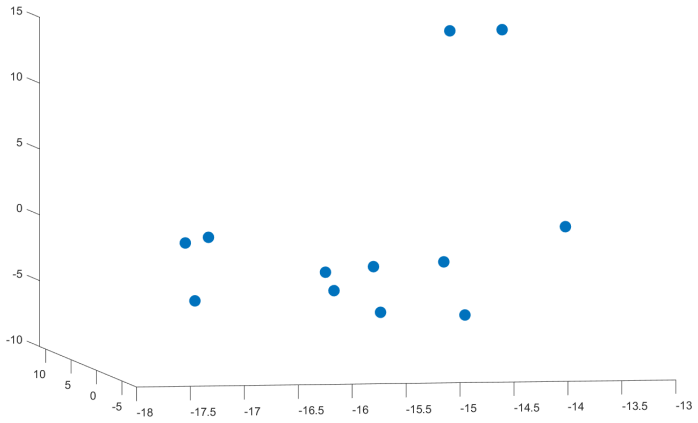

Figure 1: Plot mostrando cada uma das espécies no espaço 3D criado pelos 3 componentes cujos valores singulares eram os maiores após o SVD.

Após isso, foi gerado uma árvore hierárquica que mostra como essas espécies estão interligadas e quais os parentescos entre elas baseados nos genomas disponibilizadas pela biblioteca usada.

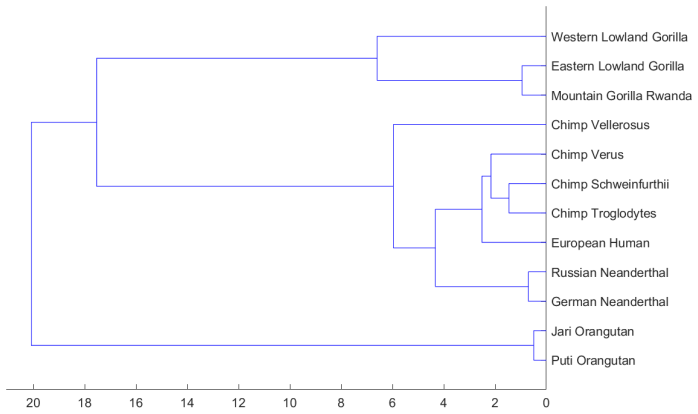

Figure 2: Gráfico mostrando a árvore hierárquica com os genomas dos primatas, mostrando a relação entre eles.

Agora, foi usado o algoritmo de clusterização K-Means para encontrar os grupos formados pelos pontos no espaço 3D criado. Visto que existem 4 grandes grupos nos dados utilizados (”Orangutan”, ”Gorilla”, ”Human/Neanderthal” e ”Chimp”) o K-Means foi executando procurando criar 4 grupos. Os resultados são mostrados pelo gráfico a seguir.

Como é possível ver, o algoritmo de K-Means conseguiu agrupar os pontos de forma correta, diferenciando os grupos esperados.

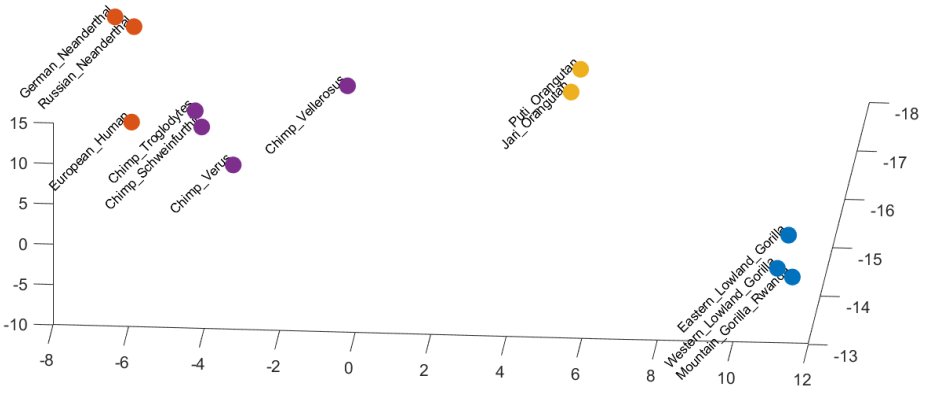

Figure 3: Plot mostrando cada uma das espécies no espaço 3D e coloridas de forma a diferenciar os grupos gerados pelo K-Means

3 Metodologia

Para gerar o vector space model a partir do genoma das espécies foi necessário contar a quantidade de cada padrão de 6 caracteres dentro de cada genoma. Isso foi feito por meio de uma janela deslizante que passa por cada uma dos conjuntos de 6 caracteres no código genético e conta no total quantos daquele padrão apareceu. Isso gera para cada uma das espécies um vetor de $4^6 = 4096$ onde cada posição é a quantidade de um certo padrão. Como isso é feito para as 12 espécies, ao final teremos uma matriz de dimensões 4096×12.

Após criada, podemos fazer a decomposição da matriz por meio do SVD para encontrar as dimensões mais importantes para a representação da matriz original e assim poder reduzir a dimensionalidade dela com maior eficiência.

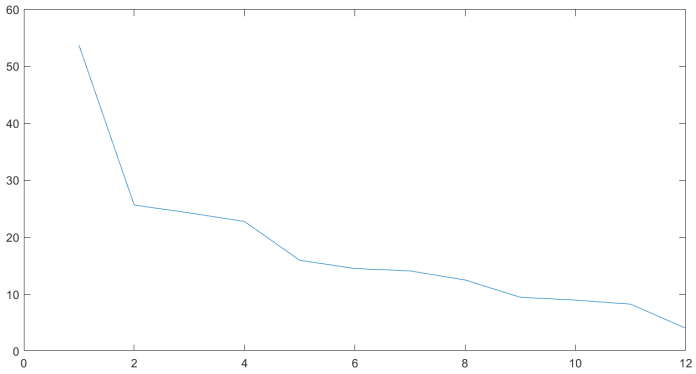

Figure 4: Gráfico mostrando a magnitude dos valores singulares obtidos pelo SVD do vector space model

Para o plot dos pontos no espaço 3D basta considerar os valores das dimensões com maiores valores singulares como a tripla de cooredenadas (x, y, z) para cada uma das espécies.

Para agrupar os dados foram usados 2 algoritmos:

1. O algoritmo K-Means começa criando grupos de forma aleatória entre os pontos no espaço 3D e iterativamente checa se alguma troca melhoraria o resultado final, e caso esse seja o caso, ele faz essa troca até que o algoritmo se estabilize e assim temos os quatro grupos finais definidos.

2. As árvores hierárquicas são cridas por inicialmente encontrando a similaridade de cada um dos pontos dados para ela, agrupando-os em uma árvore hierárquica binária. Depois disso o algoritmo corta alguns dos galhos da árvore que não são necessários.

Com esses resultados podemos ter uma ideia de quais pontos estão correlacionados de alguma forma, o que é uma informação interessante no processo de análise de dados.

4 Conclusões

O trabalho teve sucesso em aplicar os conceitos tanto do primeiro trabalho como algumas novas inovações para que o resultado final fosse ainda mais surpreendente. O fato de ser necessário converter um dado em um formato diferente para uma matriz antes de começar a aplicar os algoritmo foi um desafio interessante visto que isso é algo mais prático que já começar com os dados em forma de matriz, caso raro na vida real.

O uso dos algoritmo de agrupamento em cima dos resultados do SVD também adicionaram resultados mais interessantes visto que além de uma visualização mais plausível, temos resultados concretos em cima dos dados, que é o que eu esperava nesse tópico de mineração de dados.

A curiosidade para outras aplicações desse tipo de algoritmo também foi desenvolvida visto que a generalidade desse método parece ser incrível, podendo ser usado em diversos contextos.

5 Referências bibliográficas

1. Série de vídeos "Singular Value Decomposition" por Steve Brunton.

   https://www.youtube.com/playlist?list=PLMrJAkhIeNNSVjnsviglFoY2nXildDCcv

2. Relatório de aula sobre SVD do departamento de ciência da computação da universidade de Carnegie Mellon.

   https://www.cs.cmu.edu/ venkatg/teaching/CStheory-infoage/book-chapter-4.pdf

3. Documentação do Matlab sobre as árvores hierárquicas
   https://www.mathworks.com/help/stats/hierarchical-clustering.html
4. Documentação do Matlab sobre o K-Means
   https://www.mathworks.com/help/stats/kmeans.html

6 Anexos

6.1 Processamento de dados

O script abaixo foi a base usada para gerar a matriz de vector space models usada para o resto dos problemas.

```
load primates

% Definindo constantes para a execução do programa
rel = containers.Map({'A', 'T', 'C', 'G'}, {0, 1, 2, 3});

% Definindo o tamanho da janela deslizante
window_size = 6;

% Criando a matriz onde os valores serão armazenados
groups = zeros(4^window_size, length(primates));
names = string.empty

% Populando o vector space model
for i = 1:length(primates)
    genome = primates(i).Sequence;
    names(i) = replace(primates(i).Header, "_", " ")
    for j = window_size:length(genome)
        item = genome(j-window_size+1: j);
        if ~contains(item, "N")
            ind = 0;

            for k = 1:window_size
                ind = ind + rel(item(k)) * 4^(window_size-k);
            end

            groups(ind+1, i) = groups(ind+1, i) + 1;
        end
    end
end

% Calculando a decomposição SVD da matriz
[T, S, D] = svd(groups);
```

```
33
34  % Calcula os pontos no sistema coordenada definido pelo SVD
35  S3 = S(1:3, 1:3);
36  D3 = D(:, 1:3);
37  Aux = S3 * transpose(D3);
38  x = Aux (1, :);
39  y = Aux (2, :);
40  z = Aux (3, :);
```

6.2 Plot em espaço reduzido

Para plotar os pontos no espaço tri-dimensional como pedido no primeiro item da atividade o código a seguir foi executado.

```
1  % Plotando todos os pontos
2  plot3(x, y, z, ".", 'MarkerSize', 35);
```

6.3 Árvore hierárquica

Agora para criar a árvore hierárquica, utilizaremos a função linkage para criar uma árvore com os dados e a função dendrogram para criar o diagrama com a árvore.

```
1  % Criando a árvore
2  tree = linkage(transpose(Aux (1:3, :)), 'average');
3
4  % Plotando o diagrama
5  dendrogram(tree, 'Orientation', 'left', 'Labels', names);
```

6.4 K-Means

Após isso, foi necessário executar o algoritmo de K-Means e depois plotar os pontos com cores baseadas no resultado do algoritmo de K-Means com o código a seguir.

```
1  % Define o número de grupos para o K-Means
2  km_groups = 4
3
4  % Executa o K-Means
5  [idx] = kmeans(transpose(Aux(1:3, :)) , km_groups);
6
7  % Plota cada grupo do kmeans de uma cor diferente
8  for i = 1:km_groups
9      p = plot3(x(idx==i), y(idx==i), z(idx==i), ".");
10     set(p, 'MarkerSize', 35);
11     hold on;
```

```
end

% Nomes das espécies do lado dos pontos
set(0, 'DefaultTextInterpreter', 'none')
for i = 1:length(x)
    ht = text(x(i), y(i), z(i), primates(i).Header);
    set(ht, 'Rotation', 45);
    set(ht, 'FontSize', 8);
    set(ht, 'HorizontalAlignment', 'right');
end
```