CENTRO UNIVERSITÁRIO FEI
Leonardo Contador Neves – 118315-1

Tópicos Especiais em Aprendizagem:

K-Means

São Bernardo do Campo, SP

2018

SUMÁRIO

1 INTRODUÇÃO

Este relatório busca a implementação do algoritimo K-Means, um algoritmo de aprendizado não supervisionado para gerar grupos onde, as informações agrupadas tem similaridade.

2 REVISÃO BIBLIOGRÁFICA

Chamamos de k-means o método que tem por objetivo particionar um conjunto de observações em grupos, onde cada grupo k tem seus pontos médios bem definidos (seus centróides). Isso resulta em uma divisão do espaço de dados em um Diagrama de Voronoi.

O problema é computacionalmente difícil (NP-difícil), no entanto, existem algoritmos heurísticos eficientes que são comumente empregados e convergem rapidamente para um local optimum. Estes são geralmente semelhantes ao algoritmo de maximização da expectativa para misturas de distribuições gaussianas através de uma abordagem de refinamento iterativo utilizado por ambos os algoritmos. Além disso, ambos usam os centros de clusters para modelar dados, no entanto, a clusterização k-means tende a encontrar clusters de extensão espacial comparáveis enquanto o mecanismo de maximização da expectativa permite ter diferentes formas.

3 METODOLOGIA

Para aplicar o algoritmo do K-means, vamos começar por escolher o número de grupos ou *clusters* que vamos passar de parâmetro para nosso algoritmo. Escolhido o número de K, podemos agora selecionar, aleatoriamente, centróides iniciais para nossos agrupamentos. Após a seleção dos centróides iniciais aleatoriamente, vamos para os dois principais passos de nosso algoritmo, mas antes, vamos atribuir para cada ponto do nosso conjunto de informações, a qual grupo ele pertence fazendo uma distância euclidiana de cada ponto para os centróides iniciais.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \leq j \leq k\},$$

Equação 1: Atribuição do grupo pelo método da menor soma dos quadrados.

Com cada observação já nomeada vamos agora para os dois principais passos do K-means, o primeiro é o de recalcular os centróides de cada agrupamento fazendo a média dos pontos do grupo como mostra na figura a seguir:

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Equação 2: Cálculo do ponto médio das observações

De posse dos pontos médios das observações de cada grupos, vamos agora recalcular os grupos, sendo que os pontos médios descobertos vão ser os novos centróides dos agrupamentos. Agora temos que recalcular para cada observação, seguindo o critério da mínima distância da soma dos quadrados, o grupo que agora ela pertence e seguimos nesse laço de calculo e recalculo até que os centróides parem de se mover. A figura a seguir ilustra em um fluxograma os passos de execução do k-means, começando pela indicação do número k de gruopos, até o laço que recalcula os centróides a cada iteração.

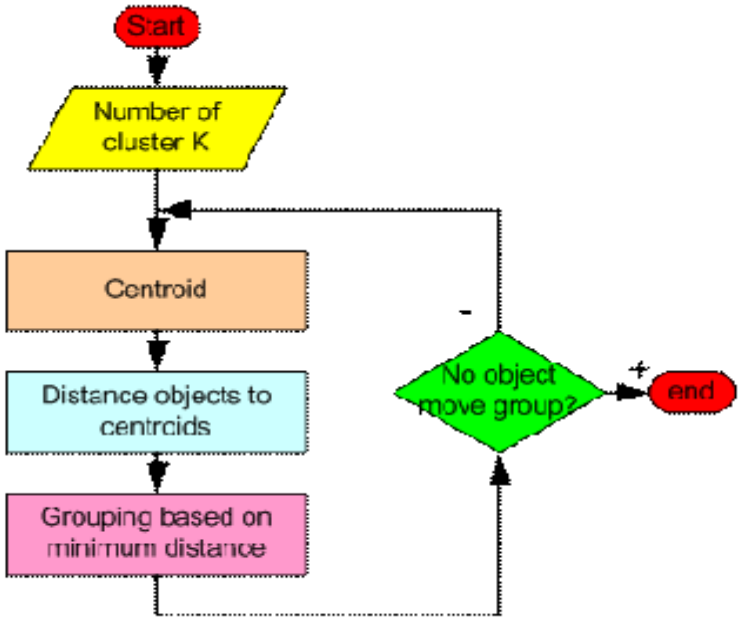


Imagem 1: Fluxograma que representa os passos do algoritmo k-means.

4 DESENVOLVIMENTO

A implementação do algoritmo k-means foi feita neste trabalho através de uma classe na linguagem de programação *Python* que tem dois métodos. Um dos métodos é o que calcula os centróides para nosso conjunto de informação, o método *fit* (ajuste) e o outro é um método auxiliar para cálculo da distância euclidiana entre dois pontos.

4.1 CLASSE PARA CÁLCULO DO K-MEANS

4.1.1 Método de *fit* (Cálculo dos centróides do k-means)

Esse é o método principal da classe onde é desenvolvido o fluxograma do algoritmo ilustrado na imagem 1. Primeiramente vamos começar com algumas variáves que iremos utilizar no decorrer do algoritmo como, o número de *clusters* "*k_numbers*", *a posição inicial dos centróides "k_centroids"* e o tamanho do meu conjunto de dados.

A imagem a baixo mostra o uso da biblioteca *Random* para retornar pontos do meu conjunto de observações para podermos apontar como centróides iniciais à partir do meu número k de centróides.

```
# Checking the k_number param
if k_number == 0:
    print("Please, choose the k number!")
    return []
k_centroids = []
# Getting k initial centroids
if centroids == []:
    k_centroids = random.sample(data, k=k_number)

data_c = data
dim = len(data[0])

k_centroids_old = np.zeros([k_number,len(data[0])])
iter = 0
```

Agora, com as variáveis iniciadas, vamos para o laço principal do algoritmo que analisa a movimentação dos centróides dos grupos. Com os centróides iniciados, temos que nomear para cada ponto do meu conjunto de dados a que grupo ele pertence. A imagem a seguir mostra, pelo método da distácia euclidiana, essa atribuição, onde a variável "*cluster_names*" vai ser preenchida com o número do grupo que cada ponto pertence.

```
  # Getting means of each centroid
  k_centroids_means = [0] * len(k_centroids)
  k_centroids_count_lines = [0] * len(k_centroids)
  cluster_names = []
  k_centroids_old = k_centroids

  for x in data_c:
      for centroid,i in zip(k_centroids,range(len(k_centroids))):
          # k_centroids_means[i] = k_centroids_means[i] + self.euclidianDistance(centroid, x
          k_centroids_means[i] = self.euclidianDistance(centroid, x)

      cluster_names.append((k_centroids_means.index(min(k_centroids_means))))
      k_centroids_count_lines[(k_centroids_means.index(min(k_centroids_means)))] += 1
```

O próximo passo agora é recalcular o ponto médio de cada grupo, a imagem a seguir mostra o calculo que cada ponto de cada grupo é usado para calcular através de uma média simples o ponto médio dos vetores e assim a variável "*k_centroids_aux*" é populada com esses valores. Ao final, é mostrado na tela o estado da iteração atual.

```
k_centroids_aux = [[0] * dim for x in range(len(k_centroids))]

for i,j in zip(data_c,cluster_names):
    for x in range(dim):
        k_centroids_aux[j][x] = k_centroids_aux[j][x] + i[x]

for x in range(len(k_centroids_aux)):
    for y in range(dim):
        k_centroids_aux[x][y] = round(k_centroids_aux[x][y]/k_centroids_count_lines[x],2)

print('Iteration: ' + str(iter))
```

Por fim, a variável que guarda os centróides é atualizada, os pontos são nomeados para cada grupo existente e são geradas duas variáveis pela classe, a "*targets*" que é uma lista que

contém os grupos de cada ponto do conjunto de informações e a variável "*centroids*", uma lista que contém os pontos centrais de cada grupo. A figura a seguir ilustra essas atribuições no código.

```
        k_centroids = k_centroids_aux
        iter+=1

        targets = []
        for x in data_c:
            aux = []
            for y in k_centroids:
                aux.append(self.euclidianDistance(x,y))
            targets.append((aux.index(min(aux))))

    self.targets = targets
    self.centroids = k_centroids
```

Para o critério de parada do algoritmo, foi usado o método de movimentação, onde os passos anteriormente descritos são executados até que o ponto central de todos os grupos pare de se mover, ou seja, o algoritmo se estabilize.

5 RESULTADOS

O algoritmo implementado foi testado em quatro bases de dados. Para a primeira, temos a seguinte tabela:

1	1,9	7,3
2	3,4	7,5
3	2.5	6,8
4	1,5	6,5
5	3,5	6,4
6	2,2	5,8
7	3,4	5,2
8	3,6	4
9	5	3,2
10	4,5	2,4
11	6	2,6
12	1.9	3
13	1	2,7
14	1.9	2,4
15	0,8	2
16	1,6	1,8
17	1	1

Tabela 1: Conjunto de observações.

Onde foi rodado o algoritmo descrito para encontrar três agrupamentos. Na imagem a seguir, é mostrado o conjunto de dados (cor azul) e os três pontos dos centróides iniciais selecionados aleatoriamente (cor vermelha).

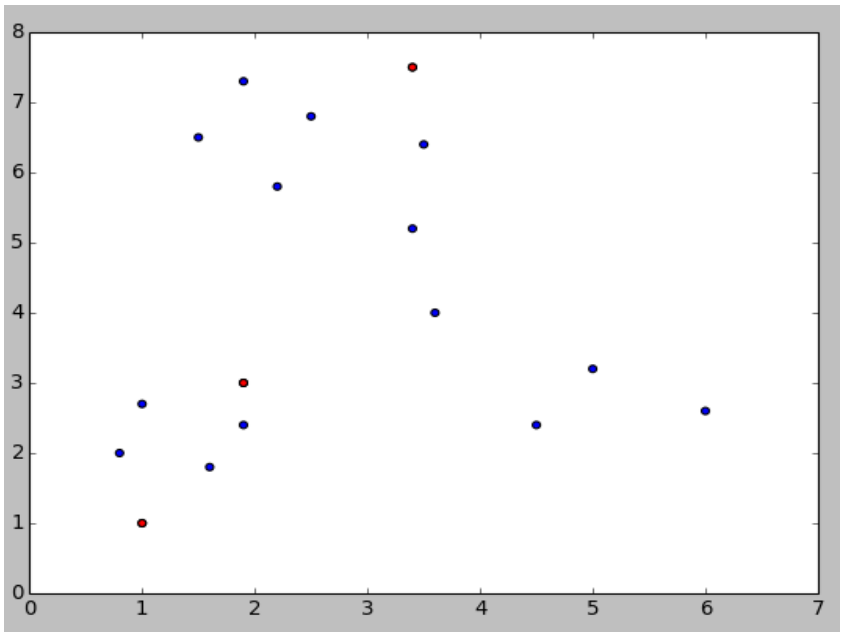


Gráfico 1: Representação do conjunto de dados

Com três iterações do algoritmo, pelo critério de movimentação dos centróides, chegamos na imagem a seguir, onde os pontos vermelhos representam o centro dos grupos encontrados para esse conjunto de informação.

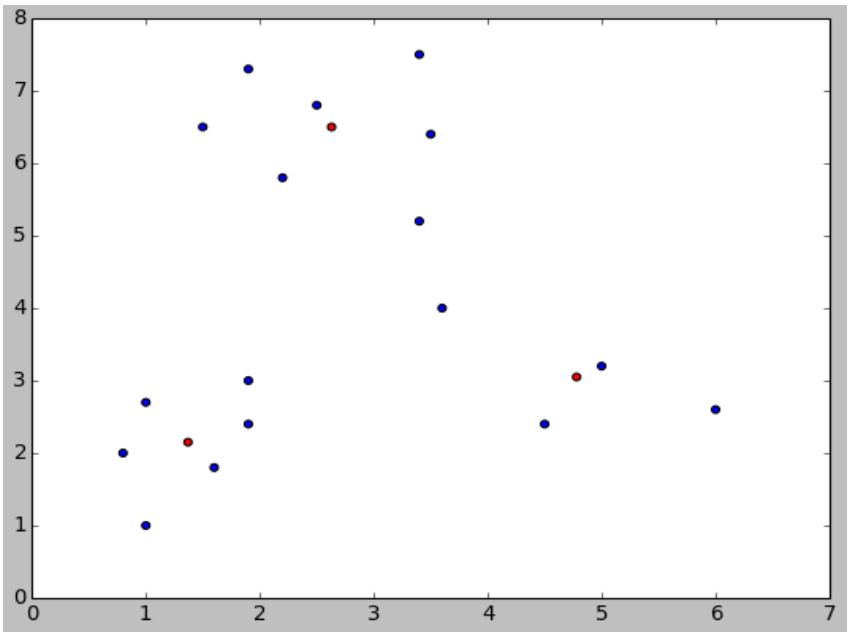


Gráfico 2: Última iteração do algoritmo para o primeiro conjunto de dados

A próxima base de dados utilizada para a validação do algoritmo foi a base Iris. A base basicamente tem 4 dimensões e com seis iterações, foi possível encontrar três grupos para o conjunto de informações, a imagem a seguir ilustra as iterações o o conjunto de *targets* gerado, ou seja, o vetor que mostra o grupo pertencente a cada observação da base de dados.

Imagem 2: Chamada do algoritmo para a base de dados Iris.

As bases seguintes para a análise junto ao algoritmo, foram utilizadas as bases “winequality-red” e a diretamente do banco de bases da *UCI Irvine Machine Learning Repository.* A imagem a seguir ilustra a execução do algoritmo para a primeira base, que basicamente possui 12 dimensões e nos diz algumas características quimicas dos vinhos vermelhos, podendo-se classificar por qualidade. O número de k escolhido foi 10, referente ao número máximo da qualidade que é atribuida ao conjunto de informações. A ideia aqui é que o algoritmo agrupo as características por qualidade.

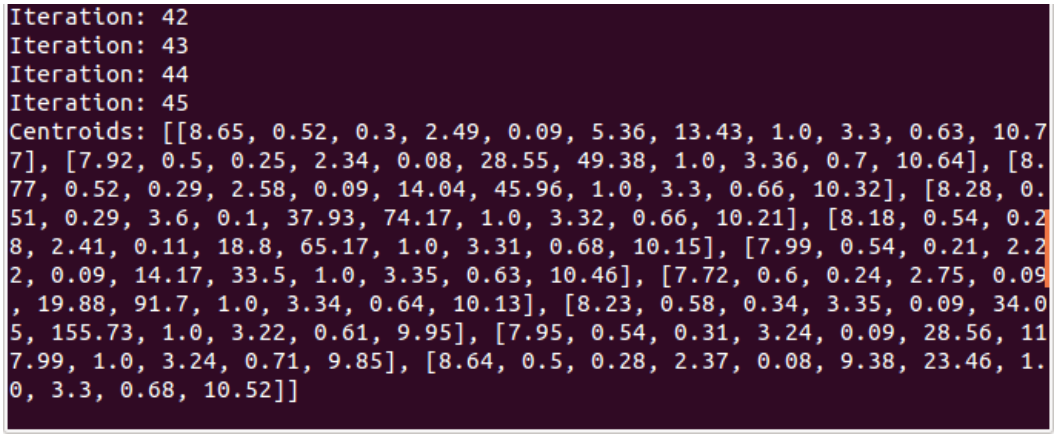
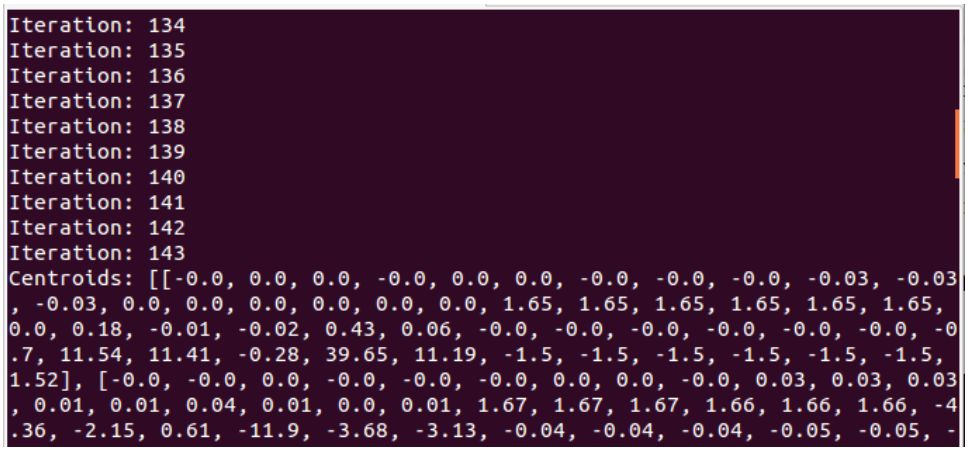


Imagem 3: Execução do algoritmo para a base de dados "winequality-red".

A próxima base usada representa informações elétricas extraidas da placa *driver* de um motor elétrico. As medições das 49 variáveis são feitas intercalando com algumas peças defeituosas do driver, resultando em medições diferentes (velocidade de rotação, corrente elétrica, etc.) para cada manipulação. A página da base falou que havia 11 classes no conjunto de informações e assim, este foi o número k escolhido para começar o algoritmo.

Com um totral de 143 iterações, o algoritmo chegou na resposta dos centróides, como mostra a figura a baixo.



5 CONCLUSÃO

Com a análise dos resultados das bases testadas junto ao algoritmo k-means, podemos notar que, por suas propriedades de média e distância quadrática, com o aumento do número k de clusters, o tempo e o número que iterações que o algoritmo leva para chegar numa estabilidade aumenta. Foi notado também que, o método de inicialização dos centróides é de estrema importância para o algoritmo, uma vez que dependendo de onde os centróides iniciais começam, o algoritmo pode demorar para chegar numa estabilidade. A classe implmentada permite que o usuário coloque manualmente os centróides iniciais e o algoritmo começa com eles como ponto de partida para o processo todo.

Como próximos passos, podemos notar que só a média foi usada para recalcular e assim um parâmetro de cálculo de moda pode ser adequado como parâmetro do método, bem como o método de cálculo das distâncias.

REFERÊNCIAS

K-MEANS. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=K-means&oldid=52414774>. Acesso em: 20 jun. 2018.

DISTÂNCIA EUCLIDIANA. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2016. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Dist%C3%A2ncia_euclidiana&oldid=45689943>. Acesso em: 23 mai. 2016.

NumPy Reference. Disponível em: <https://docs.scipy.org/doc/numpy-1.15.1/reference/index.html>. Acesso em: 10 set. 2018.