Algoritmos de Clusterização: K-Means

Prof. Mateus Mendelson
mendelson.mateus@gmail.com

mmendelson.com

1. Introdução

- É um algoritmo não supervisionado que separa dados não rotulados em K grupos.

- Os grupos (clusters) são formados de acordo com a similaridade das features entre os elementos.

- É um algoritmo iterativo e rápido.

- Serve para:

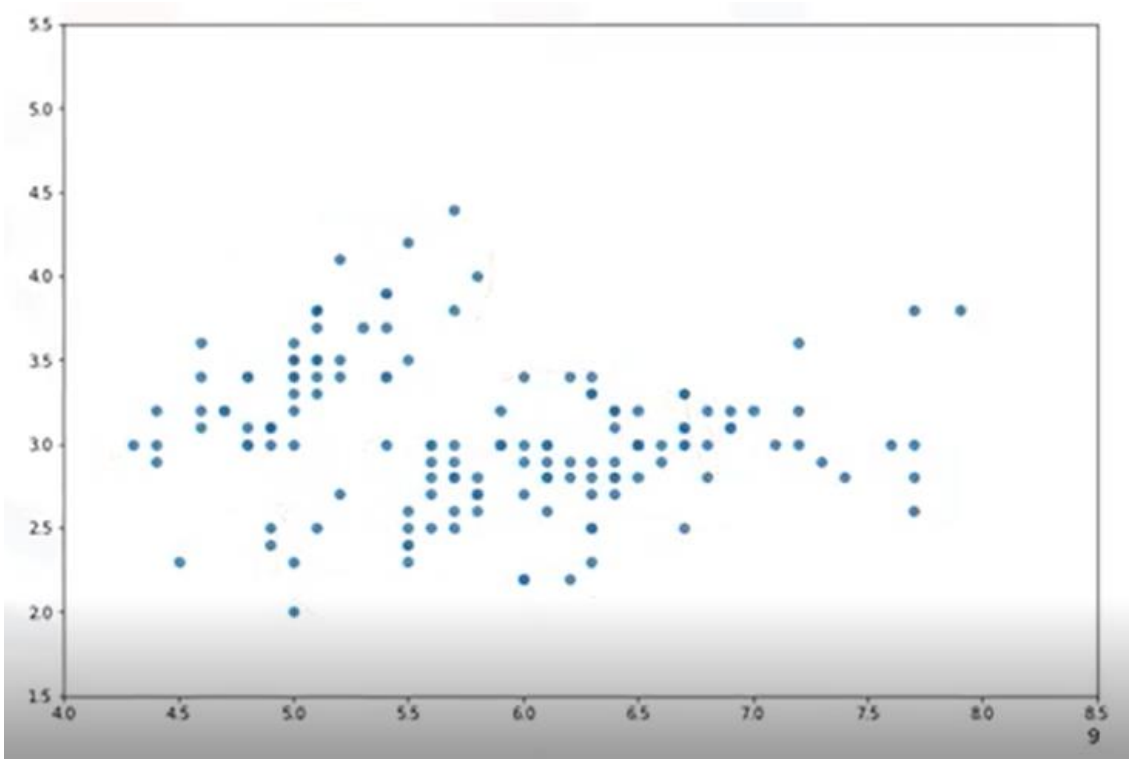
  - ✓ Descobrir usuários com comportamentos semelhantes
  - ✓ Detecção de anomalias
  - ✓ Categorizar vendedores de acordo com seus hábitos e resultados de vendas
  - ✓ etc

2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular

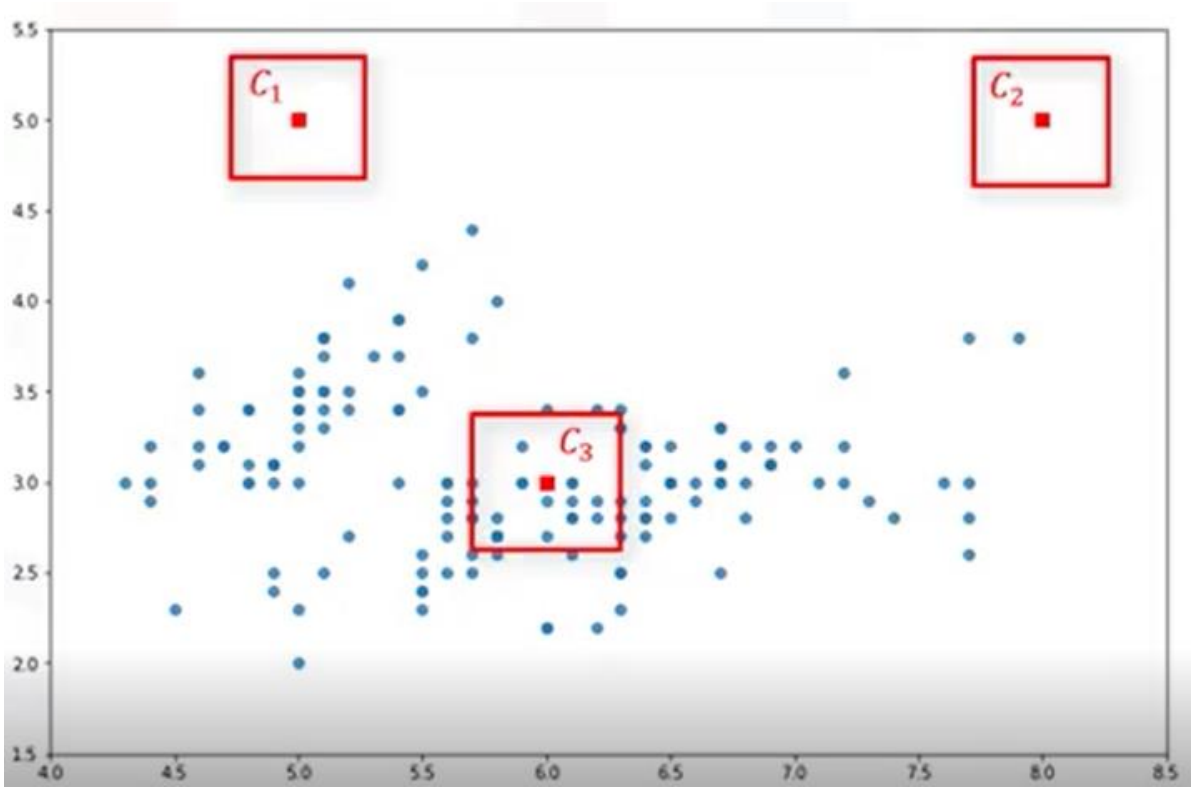
2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides



2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides



Fonte: Coursera Machine Learning with Python course

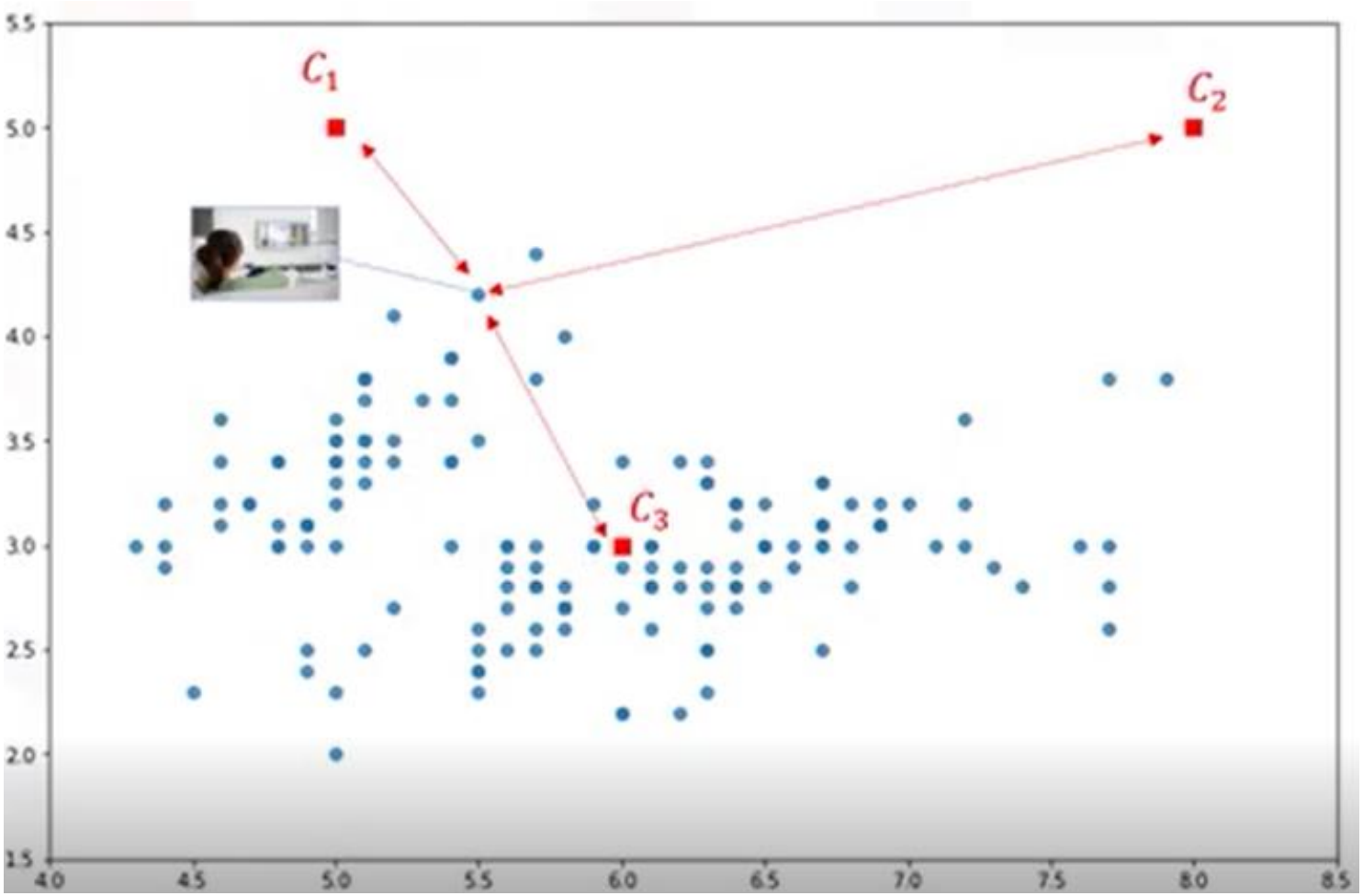
2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides. Há duas opções para escolher as coordenadas iniciais dos centroides:
   - ✓ Sortear quaisquer coordenadas
   - ✓ Sortear pontos do próprio conjunto de dados

2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides
3. Calcular a distância de cada ponto para cada centroide

2. O algoritmo K-Means



Fonte: Coursera Machine Learning with Python course

2. O algoritmo K-Means

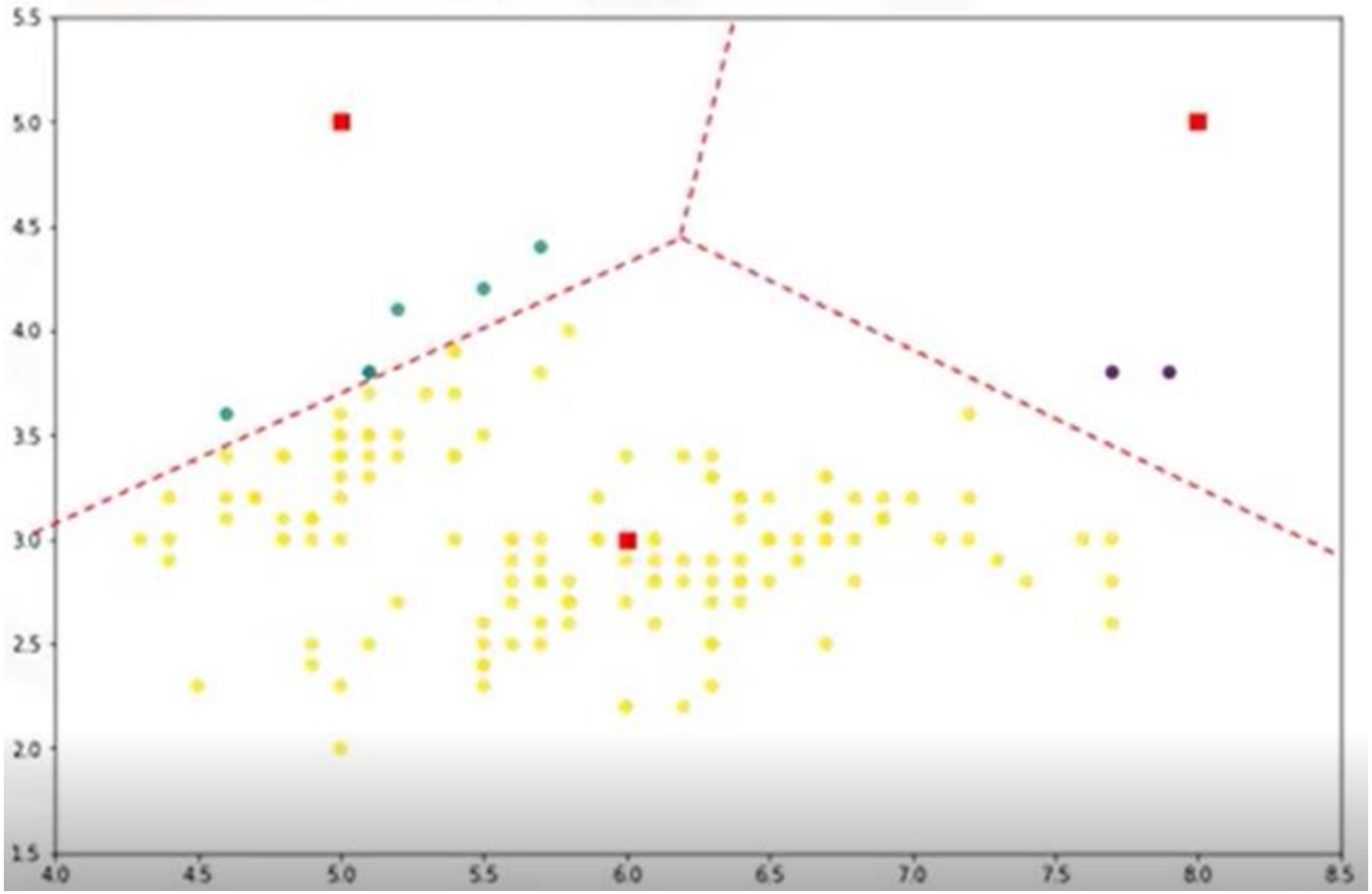
1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides
3. Calcular a distância de cada ponto para cada centroide
   - ✓ Como calcular distância entre dois pontos (x_1, y_1), (x_2, y_2)? Distância euclidiana!

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

2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides
3. Calcular a distância de cada ponto para cada centroide
4. Associar cada ponto ao centroide mais próximo

2. O algoritmo K-Means

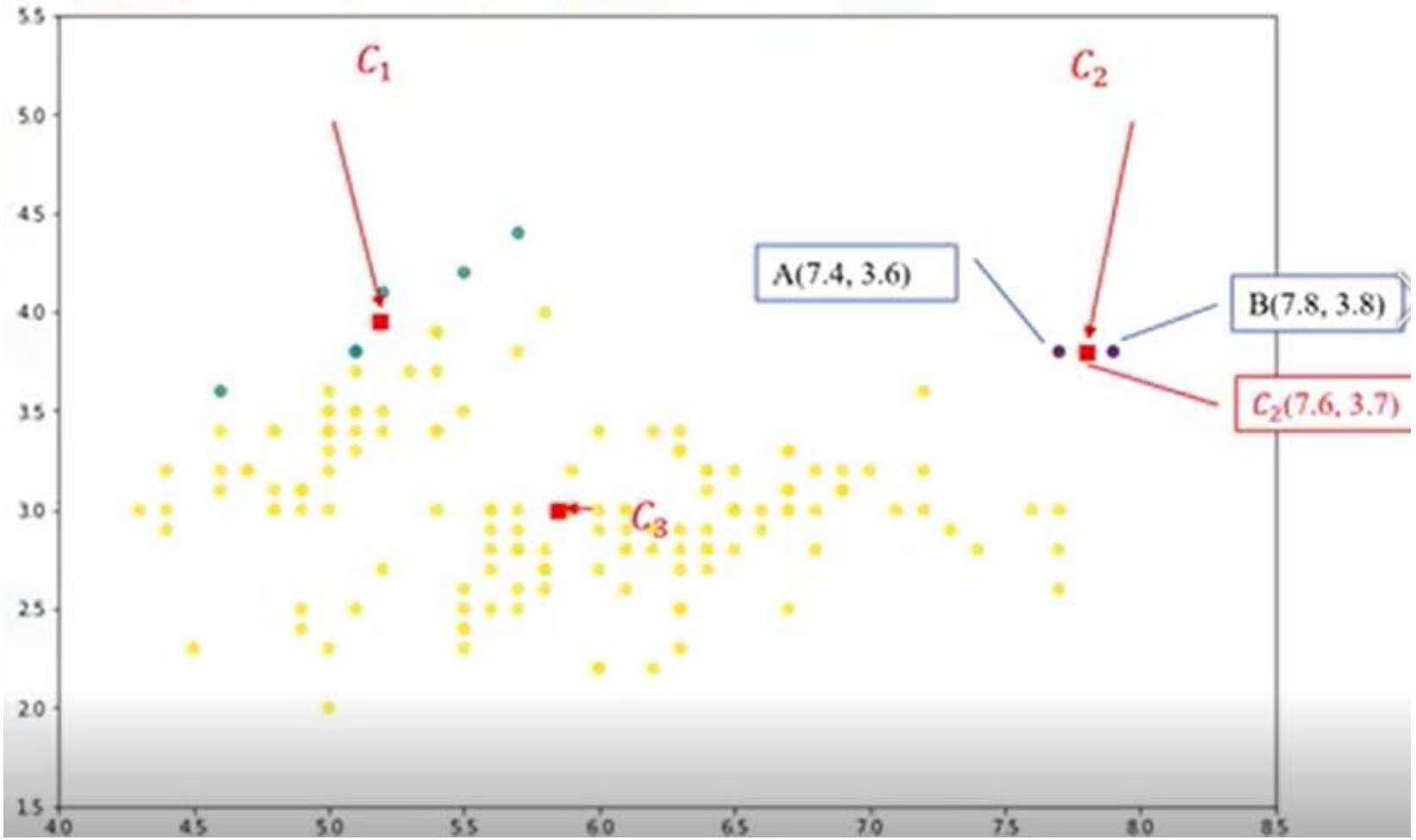


Fonte: Coursera Machine Learning with Python course

2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides
3. Calcular a distância de cada ponto para cada centroide
4. Associar cada ponto ao centroide mais próximo
5. Atualizar as coordenadas de cada centroide
   - ✓ As novas coordenadas de cada centroide são calculadas como sendo a média das coordenadas de todos os pontos associados aquele centroide

2. O algoritmo K-Means



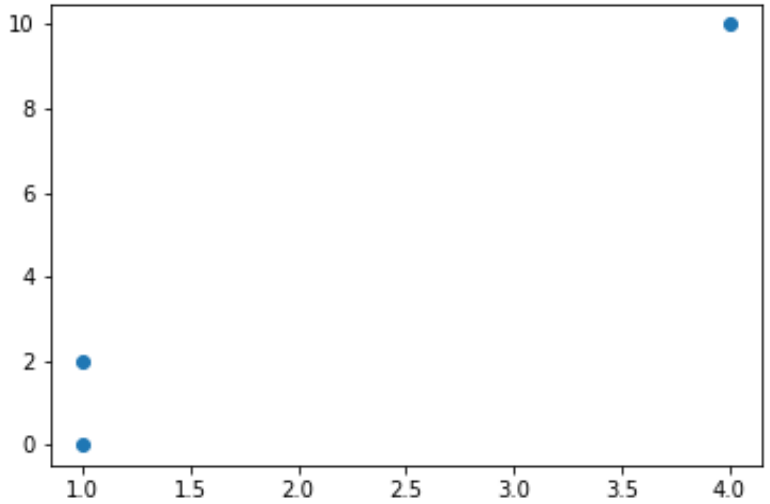
Fonte: Coursera Machine Learning with Python course

2. O algoritmo K-Means

1. Definir a quantidade K de clusters que queremos calcular
2. Sortear as coordenadas iniciais para cada um dos K centroides
3. Calcular a distância de cada ponto para cada centroide
4. Associar cada ponto ao centroide mais próximo
5. Atualizar as coordenadas de cada centroide
6. Executar os passos de 3. a 5. até que os centroides não alterem mais suas posições ou que elas variem muito pouco
7. Retornar as coordenadas dos centroides como resultado do algoritmo

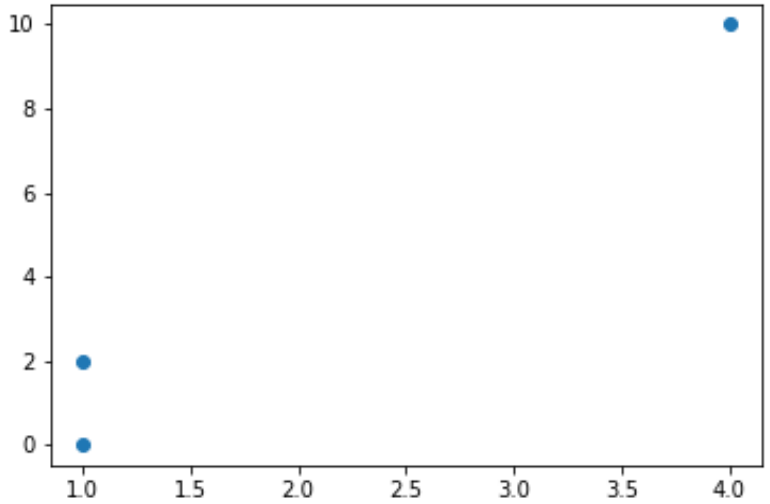
2. O algoritmo K-Means

- Considere os seguintes pontos:



2. O algoritmo K-Means

- Iremos formar dois clusters (K = 2)

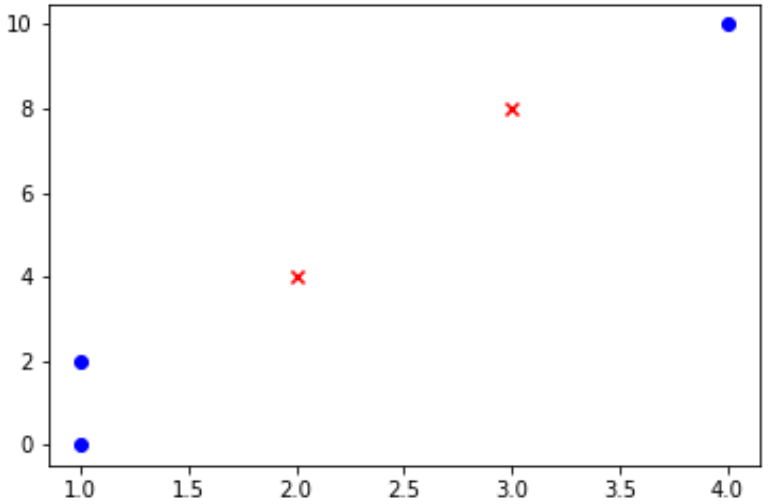


2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0)
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)



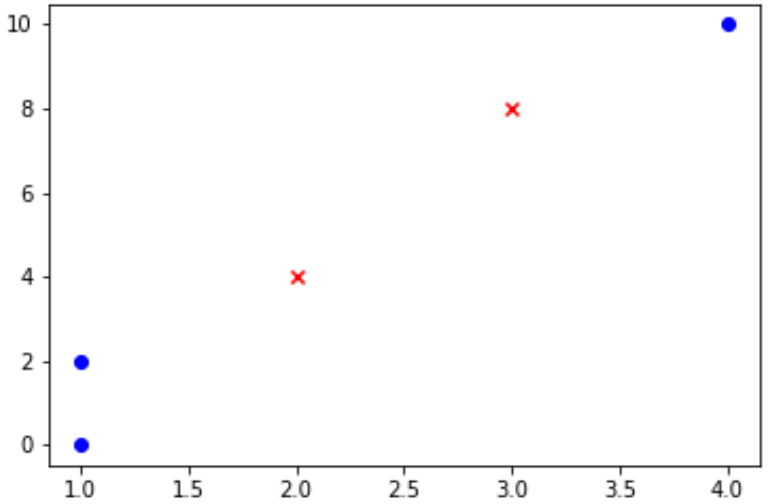
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0)
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P1C1} = \sqrt{(1-2)^2 + (0-4)^2} = 4{,}12$



2. O algoritmo K-Means

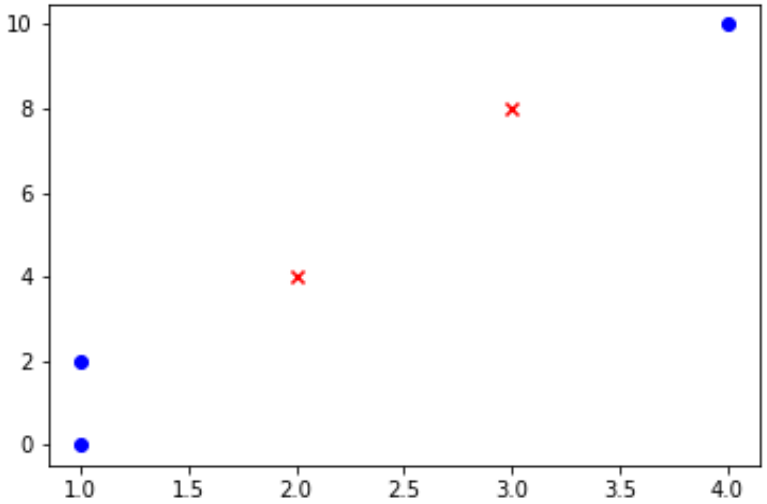
- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0)
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P1C1} = 4{,}12$

$d_{P1C2} = \sqrt{(1-3)^2 + (0-8)^2} = 8{,}25$



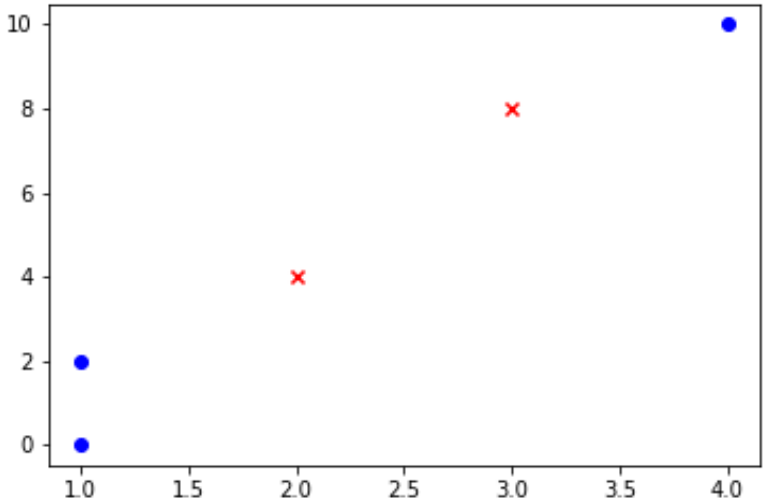
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P1C1} = 4{,}12$
$d_{P1C2} = 8{,}25$



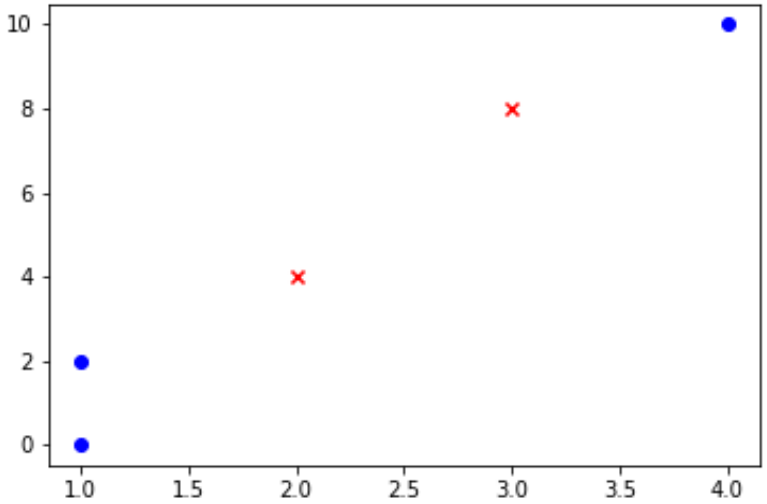
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P2C1} = \sqrt{(1 - 2)^2 + (2 - 4)^2} = 2{,}24$



2. O algoritmo K-Means

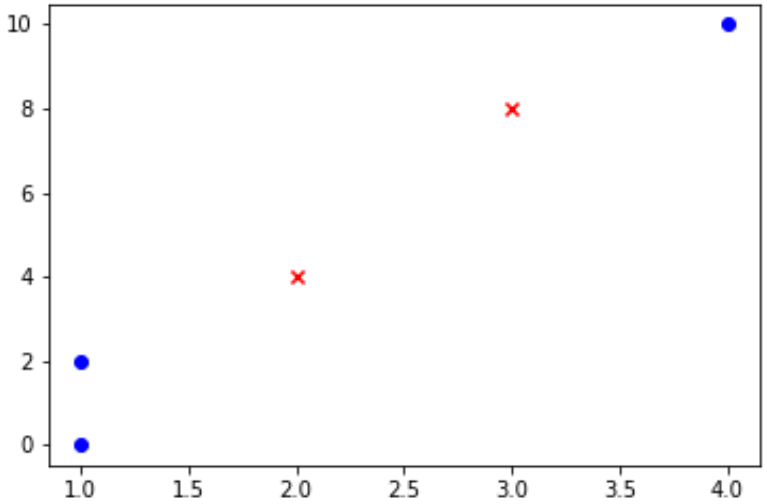
- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2)
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P2C1} = 2{,}24$

$d_{P2C2} = \sqrt{(1-3)^2 + (2-8)^2} = 6{,}32$



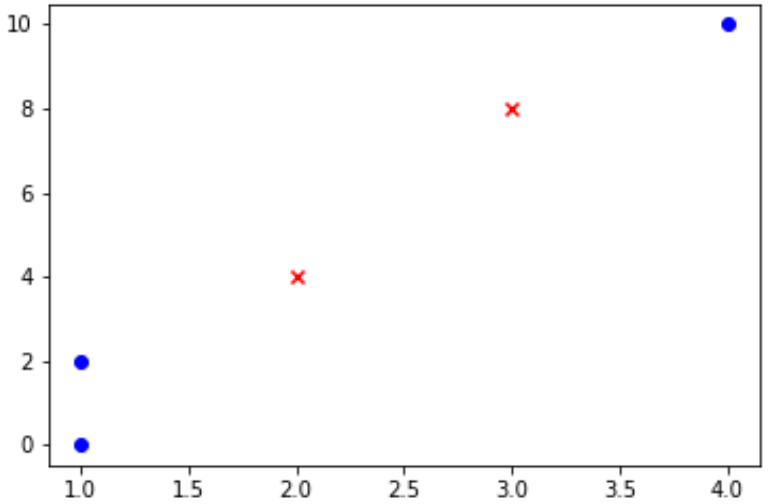
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P2C1} = 2{,}24$
$d_{P2C2} = 6{,}32$



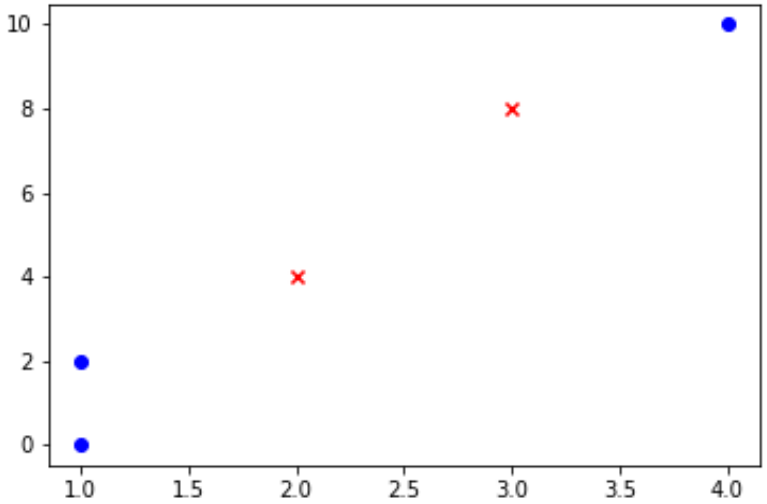
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P3C1} = \sqrt{(4 - 2)^2 + (10 - 4)^2} = 6{,}32$



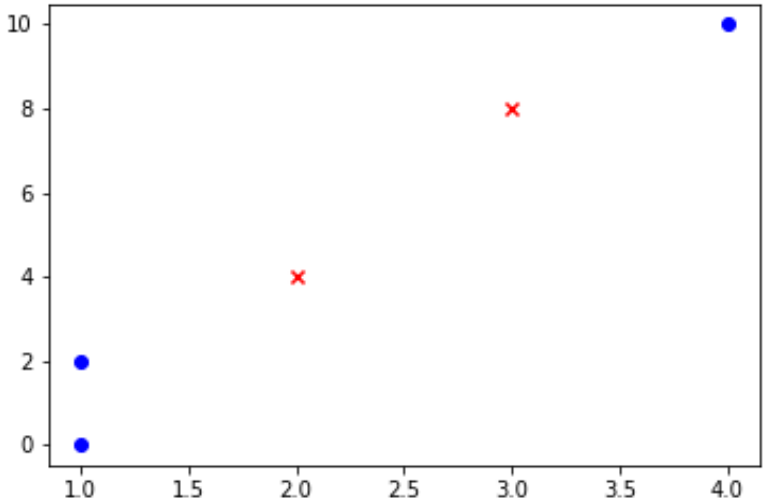
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10)

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P3C1} = 6{,}32$
$d_{P3C2} = \sqrt{(4-3)^2 + (10-8)^2} = 2{,}24$



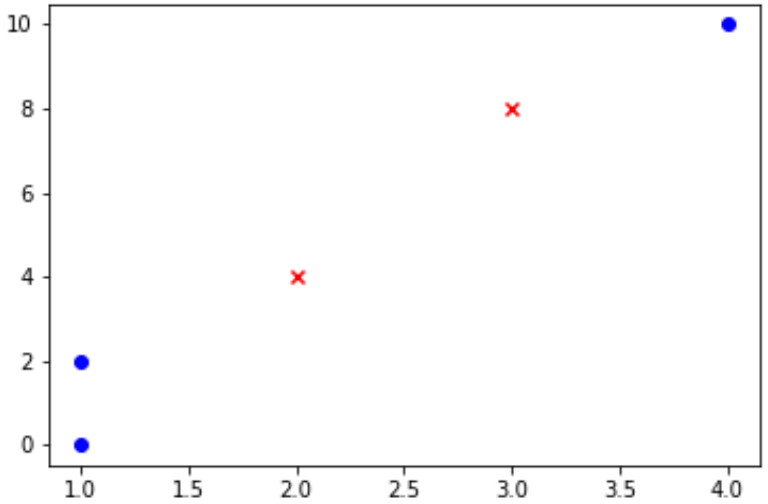
2. O algoritmo K-Means

- Sorteamos dois valores iniciais para os centroides (em vermelho)

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10) -> C2

Centroide_1: (2, 4)
Centroide_2: (3, 8)

$d_{P3C1} = 6{,}32$
$d_{P3C2} = 2{,}24$

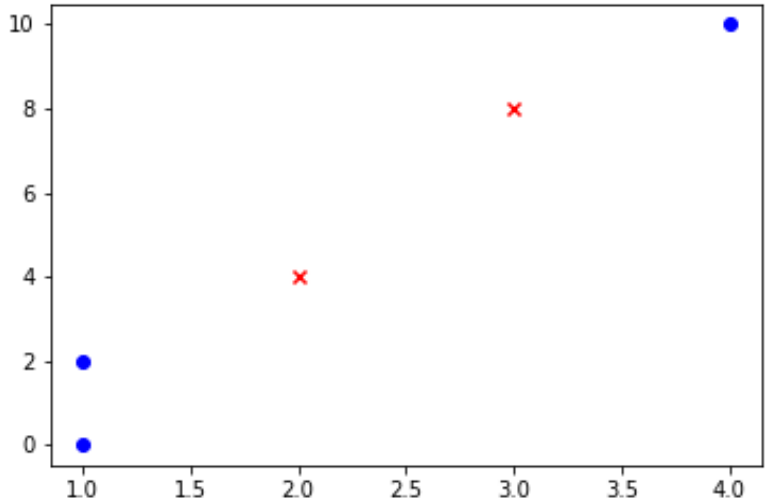


2. O algoritmo K-Means

- Agora, vamos atualizar os centroides, por meio da média das coordenadas dos pontos pertencentes a cada cluster

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10) -> C2

Centroide_1:
novo_x = (1 + 1)/2 = 1
novo_y = (0 + 2)/2 = 1

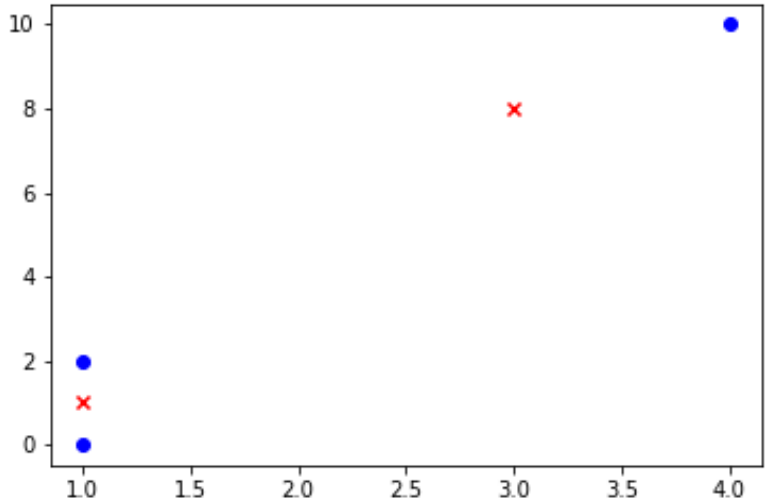


2. O algoritmo K-Means

- Agora, vamos atualizar os centroides, por meio da média das coordenadas dos pontos pertencentes a cada cluster

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10) -> C2

Centroide_1: (1, 1)

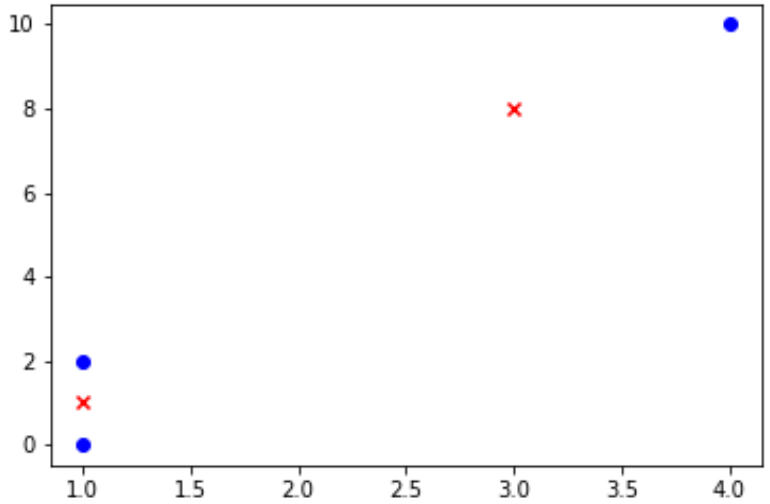


2. O algoritmo K-Means

- Agora, vamos atualizar os centroides, por meio da média das coordenadas dos pontos pertencentes a cada cluster

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10) -> C2

Centroide_1: (1, 1)
Centroide_2:
novo_x: (4)/1 = 4
novo_y: (10)/1 = 10

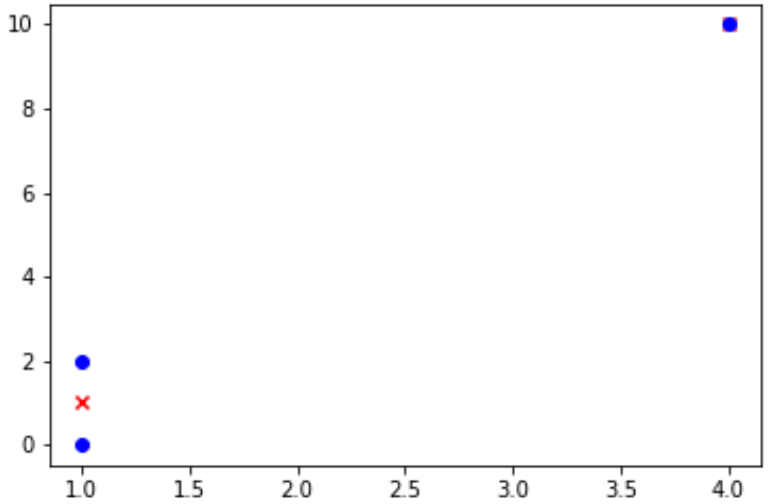


2. O algoritmo K-Means

- Agora, vamos atualizar os centroides, por meio da média das coordenadas dos pontos pertencentes a cada cluster

Ponto_1: (1, 0) -> C1
Ponto_2: (1, 2) -> C1
Ponto_3: (4, 10) -> C2

Centroide_1: (1, 1)
Centroide_2: (4, 10)

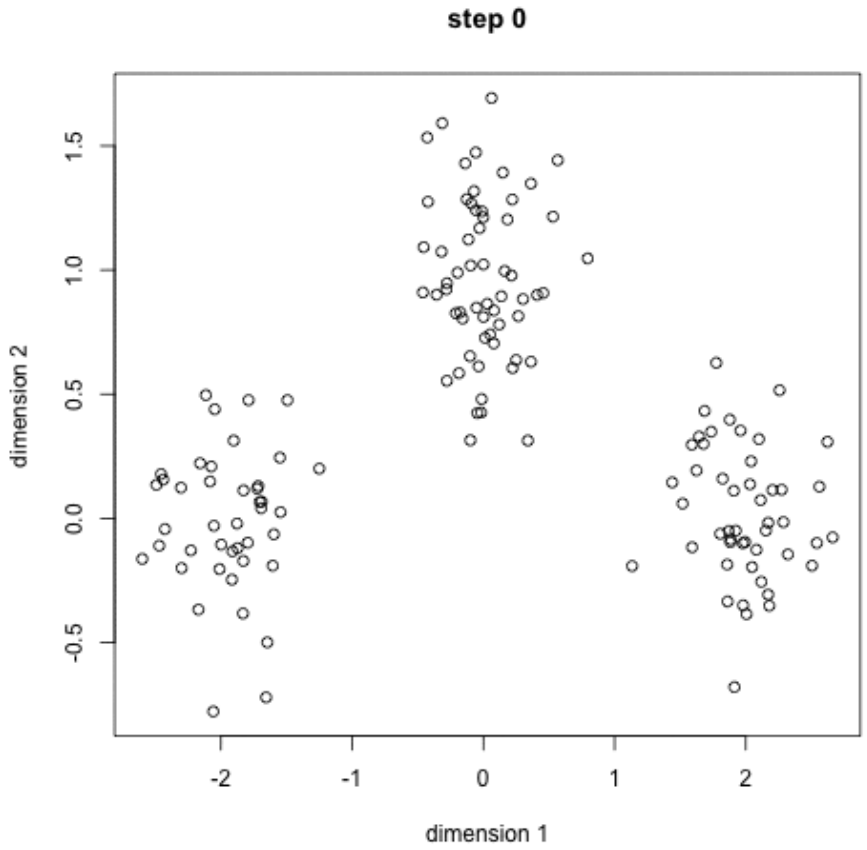


2. O algoritmo K-Means

- O processo se repete até que os centroides convirjam!

2. O algoritmo K-Means

- Ou seja, é assim que o K-Means funciona:



Fonte: https://towardsdatascience.com/an-easy-introduction-to-unsupervised-learning-with-4-basic-techniques-da7fbf0c3adf

2. O algoritmo K-Means

• Não há garantias de que os centroides finais sejam os melhores.

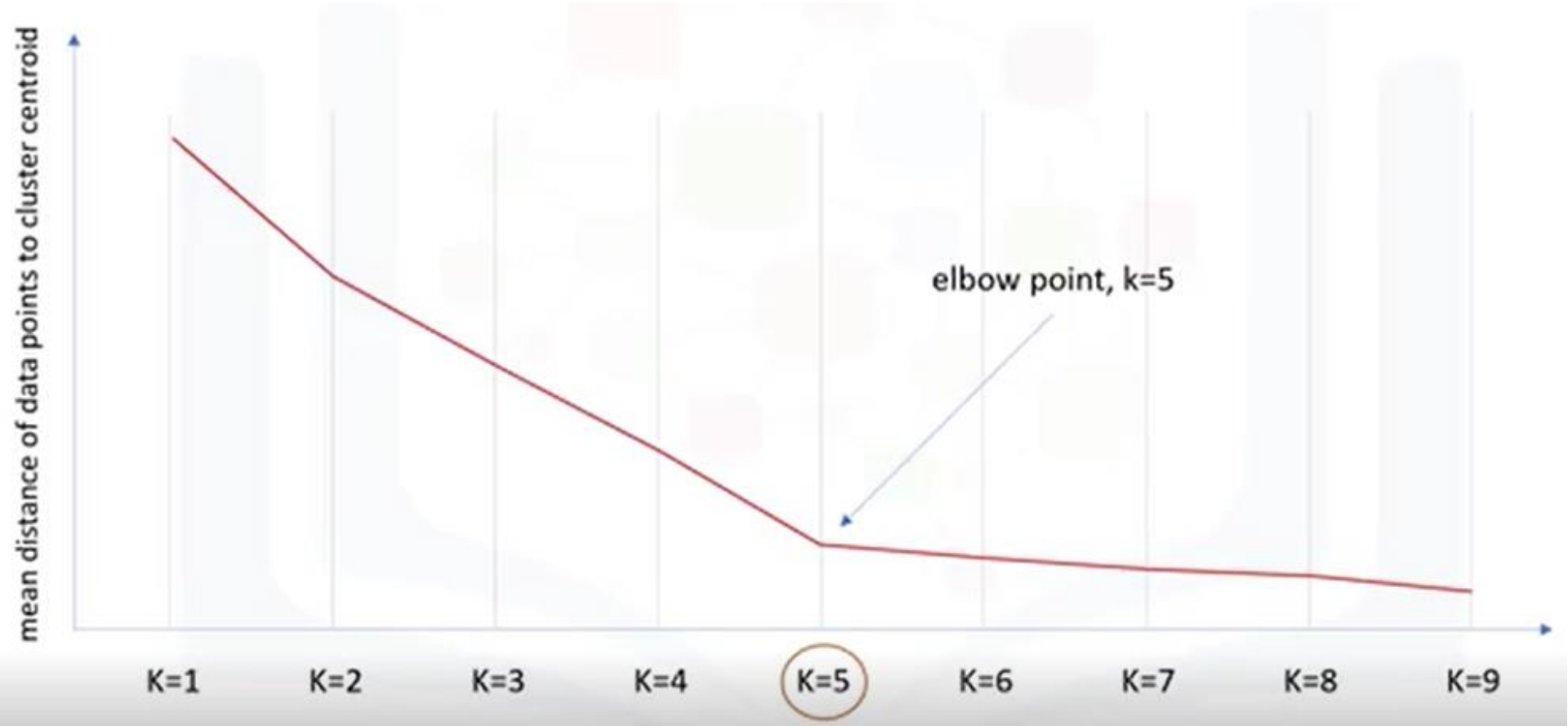
• Por isso, devemos executar o algoritmo múltiplas vezes alterando os pontos iniciais dos centroides.

• A métrica que devemos utilizar para decidir qual o melhor conjunto de centroides é o cálculo da distância média entre os pontos de cada cluster e seu centroide.

- ✓ Quanto menor for essa distância média, melhor é o resultado obtido.

2. O algoritmo K-Means

- Como escolher o melhor K?



Fonte: Coursera Machine Learning with Python course

3. Mini-projeto

• Para este projeto, iremos utilizar o dataset de COVID-19 dos casos nos Estados Unidos no período de 21 de janeiro até 09 de abril de 2020 processado.

• Tarefa: implementar a função "fit_k_means(pontos, parada, max_iter)", com K fixo e igual a 3. Retorne os centroids finais.

- ✓ pontos: conjunto de pontos 2D (casos x mortes) que serão clusterizados
- ✓ parada: valor da variação dos clusters que indicará o fim do treinamento, i.e., se distancia(centroide_antigo, centroide_novo) <= parada então termine o treinamento
- ✓ max_iter: quantidade máxima de vezes que o algoritmo deve ser repetido caso "parada" não seja alcançada
- ✓ Desafio: adicionar parâmetro com a quantidade de centroides K variável

3. Mini-projeto

• O seu relatório será o notebook exportado para um arquivo HTML e deve conter:

- ✓ Um scatter plot mostrando os centroides (com marcador x) e seus respectivos pontos (cada cluster deve estar em uma cor distinta)
- ✓ Para cada cluster, também devem ser exibidas as distâncias médias entre os pontos e seu respectivo centroide final
- ✓ Discorra sobre cada cluster: o que eles indicam?
- ✓ Desafio: implementar uma visualização iterativa do processo de treinamento igual ao gif do início da aula
- ✓ Desafio: plotar o gráfico que permite visualizar o elbow point, variando o valor de K e indicar qual o melhor valor
- ✓ Desafio: compare os resultados obtidos pelo seu algoritmo com os da função do K-Means do sklearn