MACHINE LEARNING

Clusterização Com k-means

cognitiza

Your Data & AI school

Prof. Altino Dantas

Clusterização

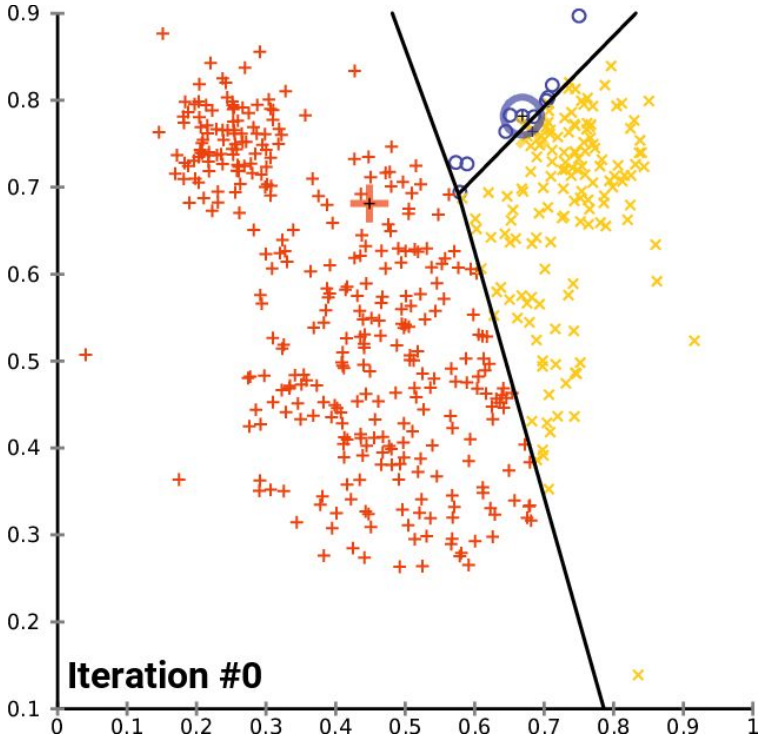
- Aprendizado não-supervisionado;
- Chamamos de não supervisionado porque os dados não possuem uma classificação, ou seja, não há uma variável dependente, ou alvo.
- Vários algoritmos: k-means, Mean Shift, Spectral clustering etc;
- Diversas aplicações: segmentação de clientes, detecção de outliers, rotular novos dados e mesclar pontos próximos em um mapa;

Caracterização | k-means

- Não confundir com knn, que é um algoritmo de aprendizado supervisionado para tarefas de classificação;
- Dado um conjunto de amostras com diferentes características, ele tenta agrupar tais amostras em ***k*** grupos;

K-means

1. Determinar o valor de *k;*
    a. Arbitrário ou com alguma estratégia mais rebuscada;
2. Aleatoriamente atribuir cada ponto a um grupo;
3. Determinar as coordenadas do centróide de cada grupo;
4. Determinar as distâncias de cada ponto até os centróides e reatribuir cada ponto ao centróide do grupo mais próximo com base na distância mínima;
5. Calcular novamente os centróides;
6. Repetir 4 e 5 até que novas alterações não possam ser feitas.



Fonte: wikipedia

k-means

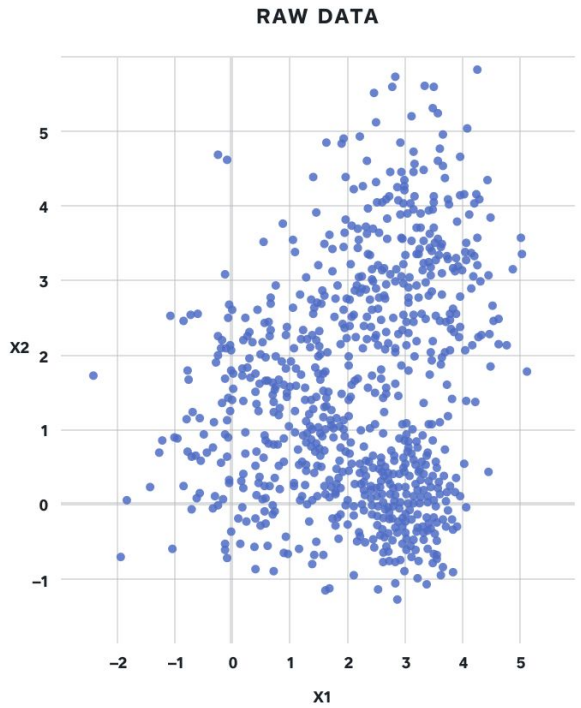
- Em python, o biblioteca *sklearn* possui uma implementação:
  - **sklearn.cluster**.KMeans

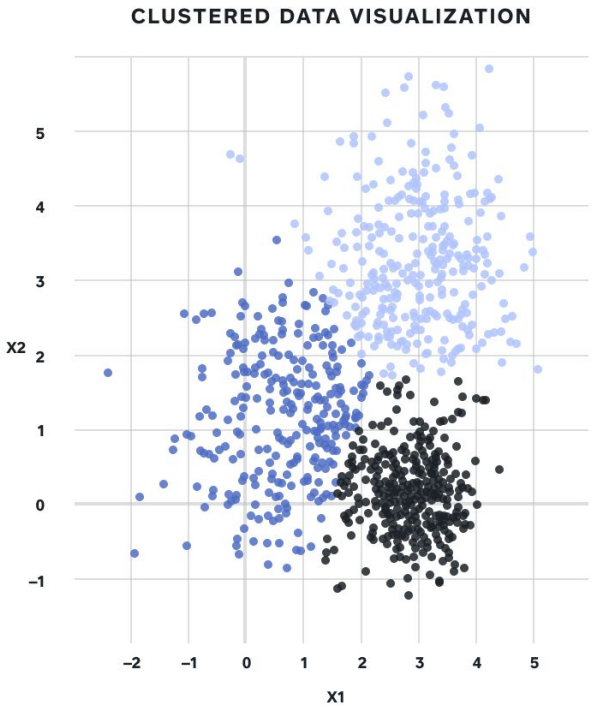
- Os principais parâmetros são:
  - n_clusters,
  - init,
  - n_init,
  - max_iter,
  - precompute_distances

Como definir o *k*

Definindo *k*

k=2





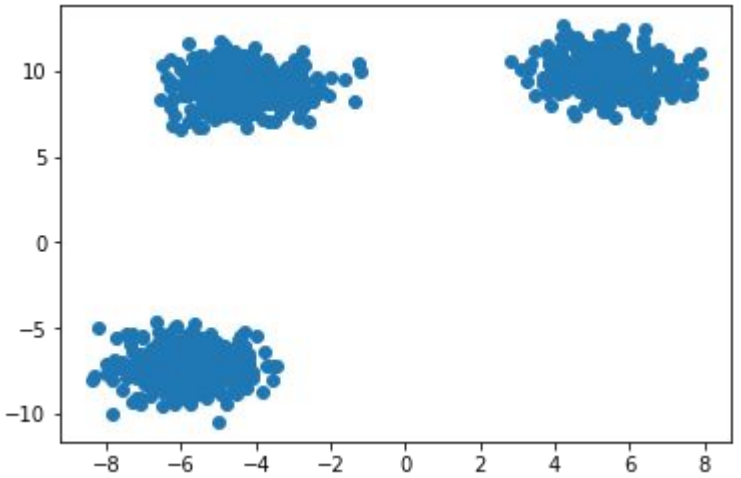
k=3

Definindo *k*

- O tamanho de *k* pode ser definido arbitrariamente;
- Dependendo do contexto, sabe-se quantos grupos se espera;
- Quando não, é preciso utilizar alguma estratégia que infira, a partir dos próprios dados, qual seria a quantidade de grupos mais provável;
- No geral, testa-se uma variação no *k* e escolhe-se o mais adequado segundo alguma métrica;

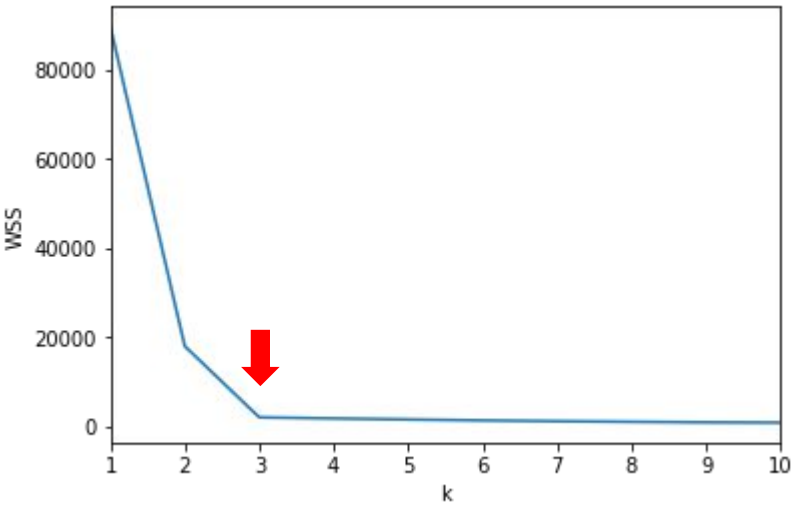
Definindo *k*

- Dois métrodos:
  - Médoto do cotovelo;
  - Método da silhueta;



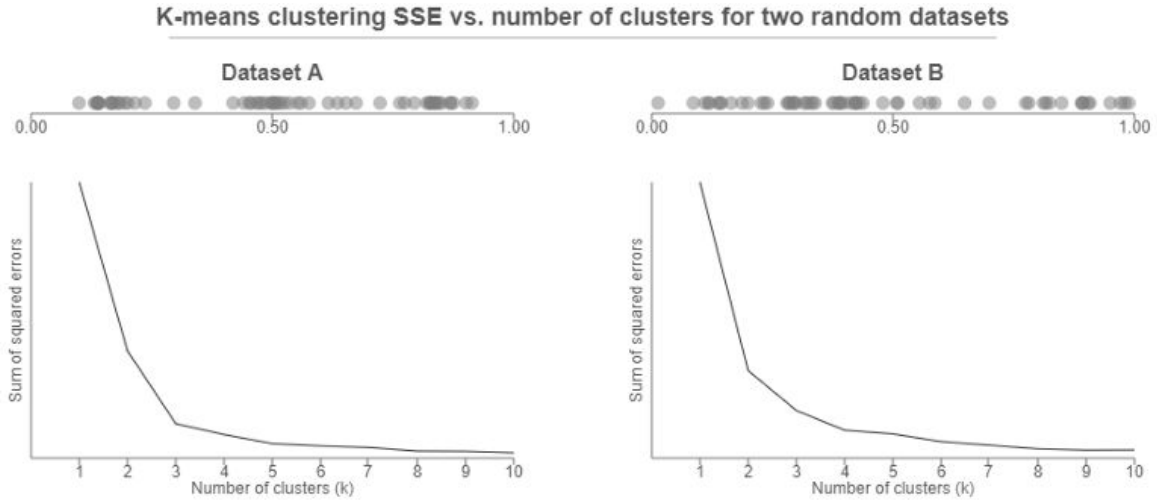
Definindo *k* / cotovelo

- Método mais conhecido e utilizado;
- Cálcula a soma dos erros quadráticos dentro dos grupos (WSS) para diferentes valores de k;
- Escolhe-se o *k* até o qual a queda no WSS é menor;
- Plota-se o gráfico de WSS vs *k*.



Definindo *k* / cotovelo

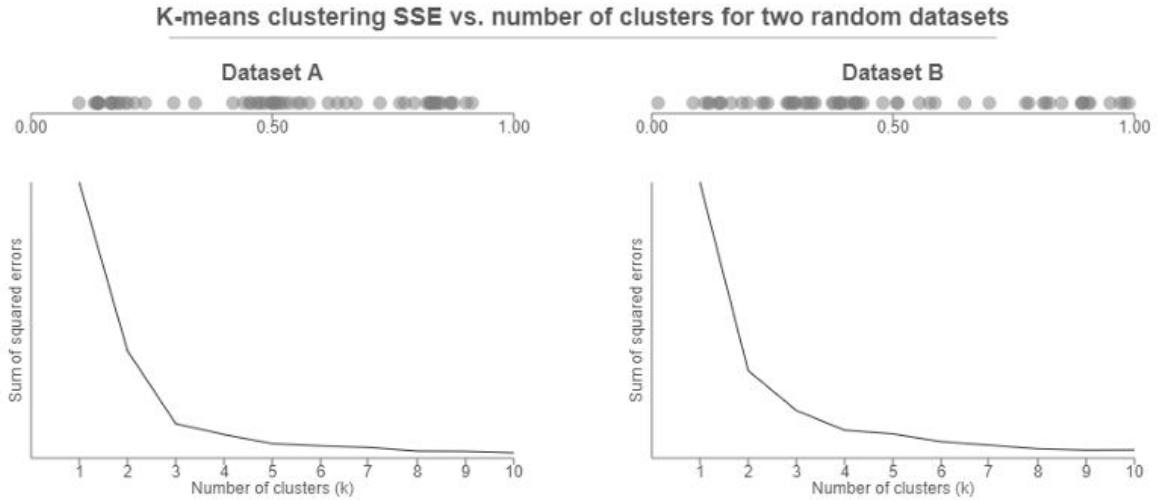
Em alguns casos, não há como se definir facilmente o “cotovelo”;



Fonte: bl.ocks.org

Definindo *k* / silhueta

Em alguns casos, não há como se definir facilmente o “cotovelo”;



Fonte: bl.ocks.org

Definindo *k* / silhueta

- O valor da silhueta mede o quão semelhante um ponto é comparado com seu próprio grupo (coesão) em comparação com outros grupos (separação);
- O valor de silhueta varia entre - 1 e 1;
- Quanto maior, melhor o valor, indicando que os pontos estão alocados nos grupos corretos;

Definindo *k* / silhueta

- O valor é dados da seguinte forma:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

- a(i) é a medida de similaridade entre o ponto i e os pontos do mesmo grupo:

For each data point $i \in C_i$ (data point i in the cluster C_i), let

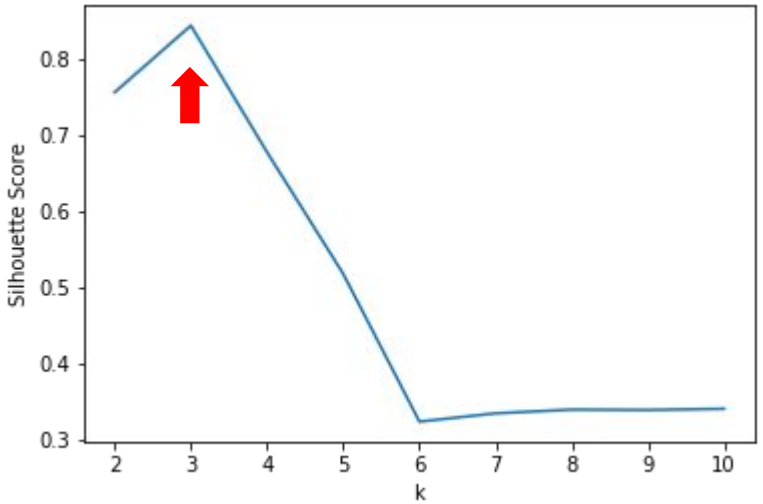
$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

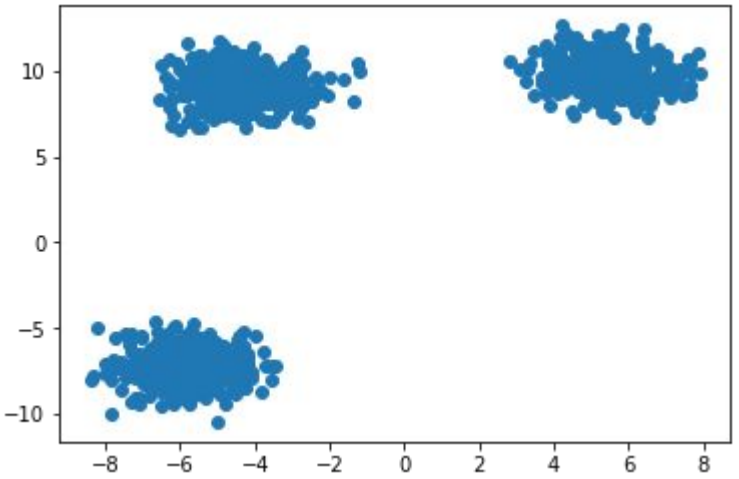
- b(i) é a medida de dissimilaridade entre o ponto i e os pontos em outros grupos:

For each data point $i \in C_i$, we now define

$$b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i, j)$$

Definindo *k* / silhueta





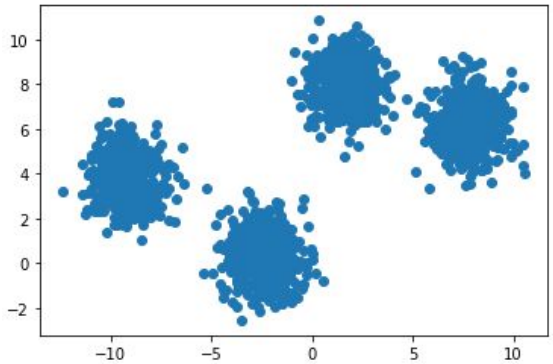
- Maiores valores de silhueta são preferíveis.

Exemplo de código

```
1 import pandas as pd
2 from sklearn import metrics
3 from sklearn.cluster import KMeans
4 import matplotlib.pyplot as plt
5 from sklearn.datasets import make_blobs
```

```
1 a, b = make_blobs(n_samples = 1800, centers = 4, n_features=2, shuffle=True, random_state=20)
```

```
1 plt.scatter(a[:, 0], a[:, 1]);
```

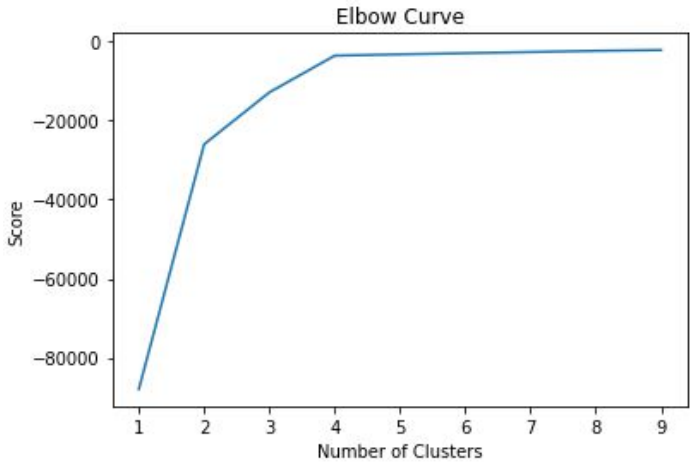


Gerando dados artificialmente

Exemplo de código

```
1 n_clusters = range(1, 10)
2
3 kmeans = [KMeans(n_clusters=i) for i in n_clusters]
4 score = [kmeans[i].fit(a).score(a) for i in range(len(kmeans))]
5
6 plt.plot(n_clusters,score)
7 plt.xlabel('Number of Clusters')
8 plt.ylabel('Score')
9 plt.title('Elbow Curve')
10 plt.show()
```

Análise do cotovelo



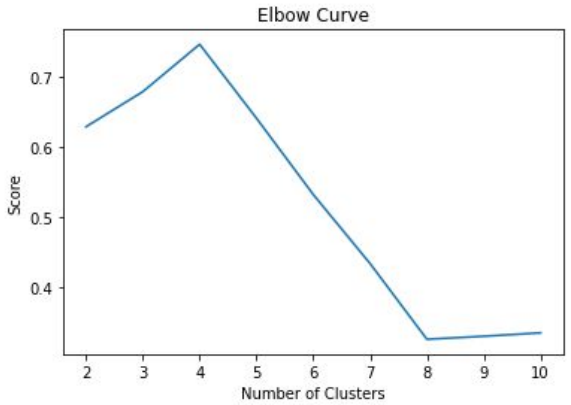
Exemplo de código

```
sil = []
kmax = 10

for k in range(2, kmax+1):
  kmeans = KMeans(n_clusters = k).fit(a)
  labels = kmeans.labels_
  sil.append(metrics.silhouette_score(a, labels, metric = 'euclidean'))
```

```
plt.plot(range(2,11),sil)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

Análise da silhueta



Exemplo de código

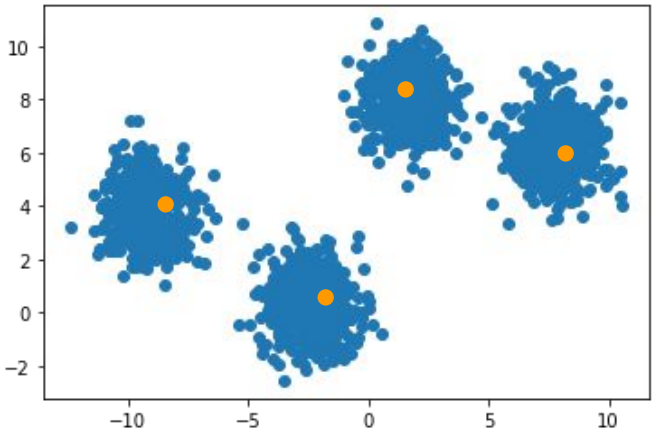
Realizando inferências

```
n_clusters = range(1, 10)

kmeans_ = [KMeans(n_clusters=i) for i in n_clusters]
score = [kmeans_[i].fit(a).score(a) for i in range(len(kmeans_))]
```

```
1 kmeans_[3].predict([[2,8],[-8,4], [8,6],[-2,0]])
```

```
array([2, 3, 1, 0], dtype=int32)
```



Exercício

- Reproduza os códigos de exemplo;
- Carregue os dados do arquivo “dados_clusters.csv” e descubra qual a quantidade ideal de *clusteres* no conjunto de dados;
- Apresente o gráfico de dispersão com os valores.

Obrigado.