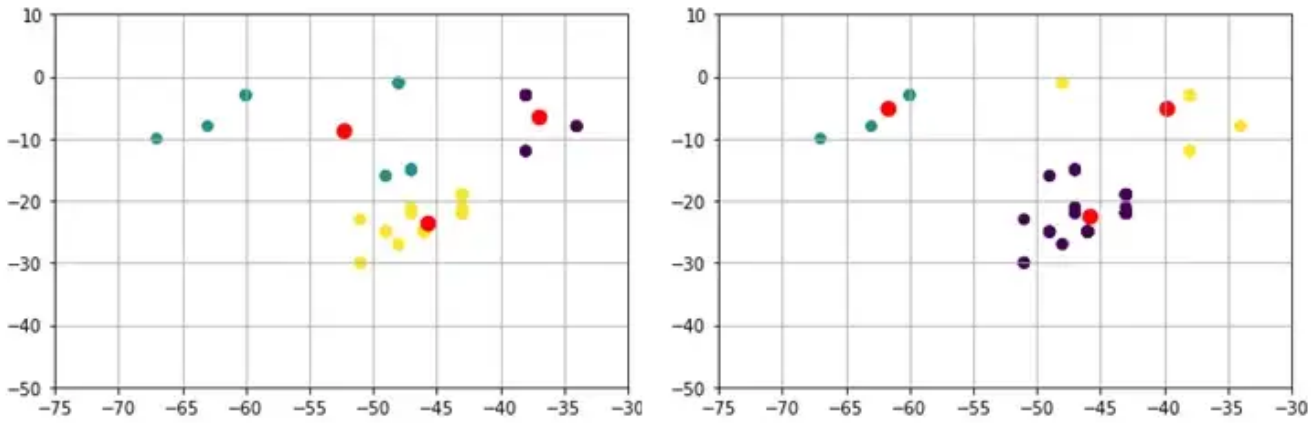
Published in Programadores Ajudando Programadores

Bruno Anastacio Follow

May 19, 2020 · 8 min read

K-means: o que é, como funciona, aplicações e exemplo em Python

Neste post, vamos usar o Machine Learning em favor da logística



O que é K-Means?

K-Means é um algoritmo de clusterização (ou agrupamento) disponível na biblioteca Scikit-Learn.

É um algoritmo de aprendizado não supervisionado (ou seja, que não precisa de inputs de confirmação externos) que avalia e clusteriza os dados de acordo com suas características, como por exemplo:

- **clientes/características semelhantes**
- **séries/gênero da série ou faixa etaria**
- **usuarios de uma rede social/usuario influenciador**
- **paciente/sintoma ou característica semelhante**

Por exemplo, se eu tenho uma rede de lojas com abrangência nacional, qual seria os melhores lugares para construir os centros logísticos de abastecimento?

Podemos começar a responder isso com K-means.

Como funciona?

1. Primeiro, preciso definir um 'K', ou seja, um número de clusters (ou agrupamentos).
2. Depois, preciso definir, aleatoriamente, um centroide para cada cluster.
3. O próximo passo é calcular, para cada ponto, o centroide de menor distância. Cada ponto pertencerá ao centroide mais próximo (lembrar do exemplo do CD logístico e das lojas: cada loja (ponto) deve ser atendida pelo CD (centróide) mais próximo)
4. Agora, devo reposicionar o centróide. A nova posição do centroide deve ser a média da posição de todos os pontos do cluster.
5. Os dois ultimos passos são repetidos, iterativamente, até obtermos a posição ideal dos centróides.

K-Means na prática, em Python:

O primeiro passo é importar as bibliotecas necessárias:

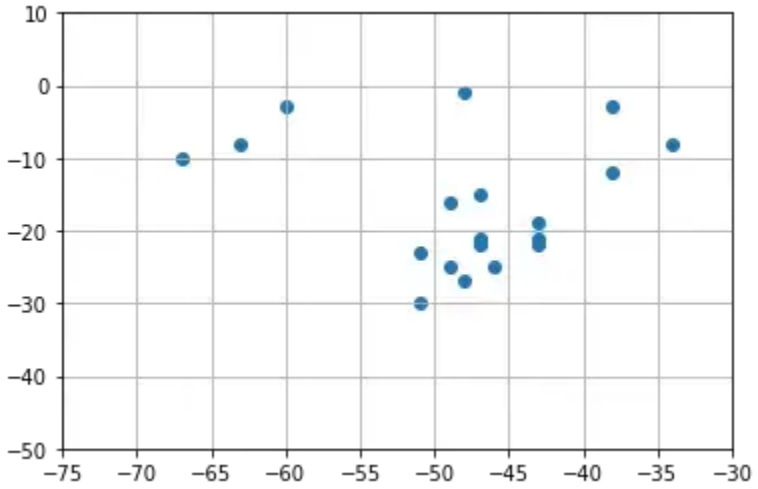
```
import numpy as np #para manipular os vetores
```


Vamos ao exemplo: a rede de lojas Bruno tem 19 lojas em algumas das principais cidades do país. A empresa pensa em construir 3 centros logísticos para abastecer as lojas. Mas, qual seria a posição ótima para cada um desses três hubs, considerando apenas a posição (coordenadas geográficas) das lojas?

Abaixo, plotamos no gráfico a representação em coordenadas de cada uma das 19 cidades onde a rede possui filiais.

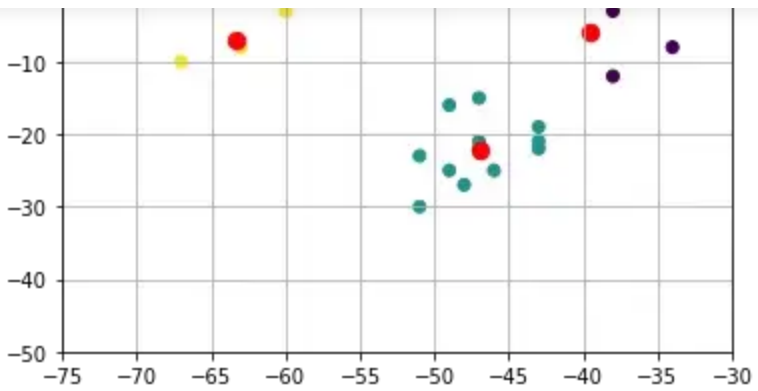
```
dataset = np.array(

#matriz com as coordenadas geográficas de cada loja

[[-25, -46], #são paulo
[-22, -43], #rio de janeiro
[-25, -49], #curitiba
[-30, -51], #porto alegre
[-19, -43], #belo horizonte
[-15, -47], #brasilia
[-12, -38], #salvador
[-8, -34], #recife
[-16, -49], #goiania
[-3, -60], #manaus
[-22, -47], #campinas
[-3, -38], #fortaleza
[-21, -47], #ribeirão preto
[-23, -51], #maringa
[-27, -48], #florianópolis
[-21, -43], #juiz de fora
[-1, -48], #belém
[-10, -67], #rio branco
[-8, -63] #porto velho])

plt.scatter(dataset[:,1], dataset[:,0]) #posicionamento dos eixos x
e y

plt.xlim(-75, -30) #range do eixo x

plt.ylim(-50, 10) #range do eixo y

plt.grid() #função que desenha a grade no nosso gráfico
```


norte representa Belém.



Vamos utilizar o algoritmo KMeans, do pacote **Scikit-Learn** para agrupar (clusterisar) as nossas filiais em 3 grupos. Cada grupo será servido por um centro logístico, que será representado por um centróide (os pontos em vermelho no gráfico).

```
kmeans = KMeans(n_clusters = 3, #numero de clusters

init = 'k-means++', n_init = 10, #algoritmo que define a posição dos
clusters de maneira mais assertiva

max_iter = 300) #numero máximo de iterações

pred_y = kmeans.fit_predict(dataset)

plt.scatter(dataset[:,1], dataset[:,0], c = pred_y) #posicionamento
dos eixos x e y

plt.xlim(-75, -30) #range do eixo x

plt.ylim(-50, 10) #range do eixo y

plt.grid() #função que desenha a grade no nosso gráfico

plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,0
], s = 70, c = 'red') #posição de cada centroide no gráfico

plt.show()
```



Nossa clusterização apontou três posições para os nossos centros logísticos. Vamos ver onde, aproximadamende, eles ficariam?

- [-7, -63.33333333] — Humaitá/AM
- [-6, -39.5] — Acopiara/CE
- [-22.16666667, -47] — Mogi Guaçu/SP

Melhorando um pouco o exemplo

Neste segundo cenário, vamos considerar que em algumas das maiores cidades nós temos mais que uma loja.

Só em São Paulo, agora temos 21 lojas. No total, nossa rede teria 86 lojas.

```
novo_dataset = np.array(
#matriz com as coordenadas geográficas de cada loja
[
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
```

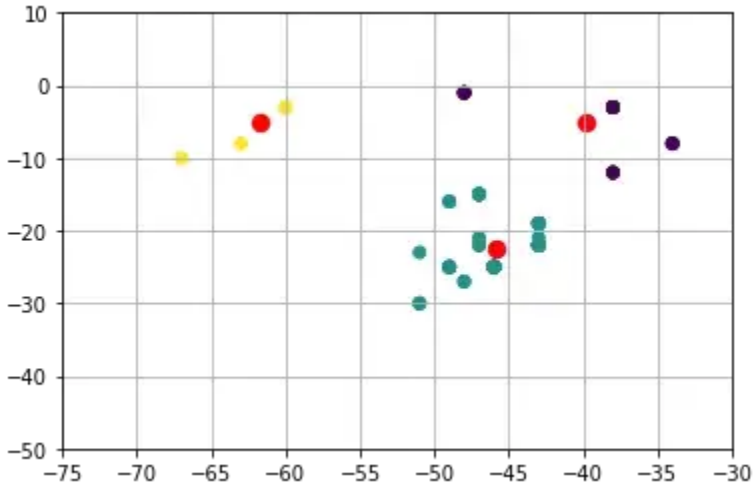
```
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-25, -46], #são paulo
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
[-22, -43], #rio de janeiro
```


```
[-25, -49], #curitiba
[-25, -49], #curitiba
[-25, -49], #curitiba
[-25, -49], #curitiba
[-30, -51], #porto alegre
[-30, -51], #porto alegre
[-30, -51], #porto alegre
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-19, -43], #belo horizonte
[-15, -47], #brasilia
[-15, -47], #brasilia
[-15, -47], #brasilia
[-15, -47], #brasilia
[-15, -47], #brasilia
[-15, -47], #brasilia
[-12, -38], #salvador
[-12, -38], #salvador
[-12, -38], #salvador
[-8, -34], #recife
[-8, -34], #recife
[-8, -34], #recife
```


```
[-3, -60], #manaus
[-3, -60], #manaus
[-3, -60], #manaus
[-3, -60], #manaus
[-22, -47], #campinas
[-22, -47], #campinas
[-3, -38], #fortaleza
[-3, -38], #fortaleza
[-3, -38], #fortaleza
[-3, -38], #fortaleza
[-3, -38], #fortaleza
[-3, -38], #fortaleza
[-21, -47], #ribeirão preto
[-23, -51], #maringa
[-27, -48], #florianópolis
[-27, -48], #florianópolis
[-21, -43], #juiz de fora
[-1, -48], #belém
[-1, -48], #belém
[-1, -48], #belém
[-1, -48], #belém
[-10, -67], #rio branco
[-8, -63] #porto velho
]
)
plt.scatter(novo_dataset[:,1], novo_dataset[:,0]) #posicionamento
dos eixos x e y
```


Agora, nossos centros logísticos ficariam nos seguintes locais:

1. [-5, -61.66666667] — Beruri/AM
2. [-5.125, -39.75] — Boa Viagem/CE
3. [-22.55384615, -45.90769231] — Consolação/MG



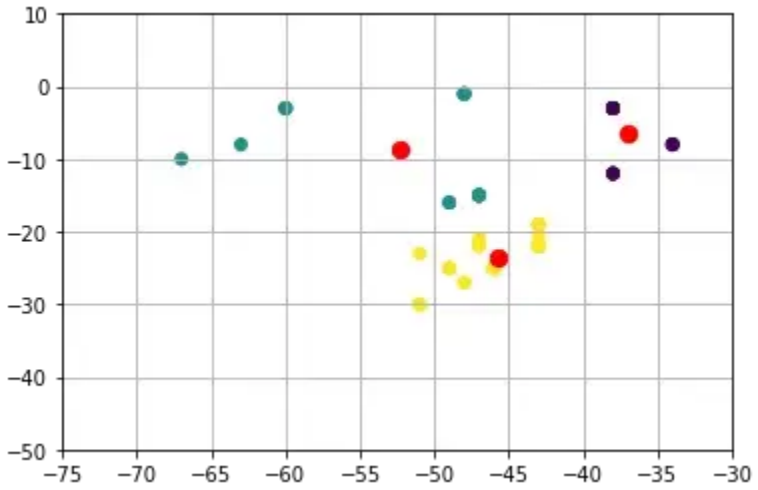
‘K-means++’ vs ‘random’ e a aleatoriedade

Como dissemos no inicio, quando usamos `k-means`, o algoritmo, desde o início, define os melhores pontos de iniciação dos centróides. Para porvar isso, vamos reiniciar nosso algoritmo usando `'random'` dessa vez.

Abaixo vou usar uma única iteração, para que você possa ver de onde nossos centróides podem partir. Lembrando que random os lança aleatoriamente, ou seja, para cada vez que você rodar essa célula, ele vai iniciar de um lugar diferente.

```
`kmeans = KMeans(n_clusters = 3, #numero de clusters
init = ‘random’, n_init = 10, #algoritmo que define a posição dos
clusters de maneira mais assertiva
```

```
plt.xlim(-75, -30) #range do eixo x

plt.ylim(-50, 10) #range do eixo y

plt.grid() #função que desenha a grade no nosso gráfico

plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,0
], s = 70, c = 'red') #posição de cada centroide no gráfico

plt.show()
```

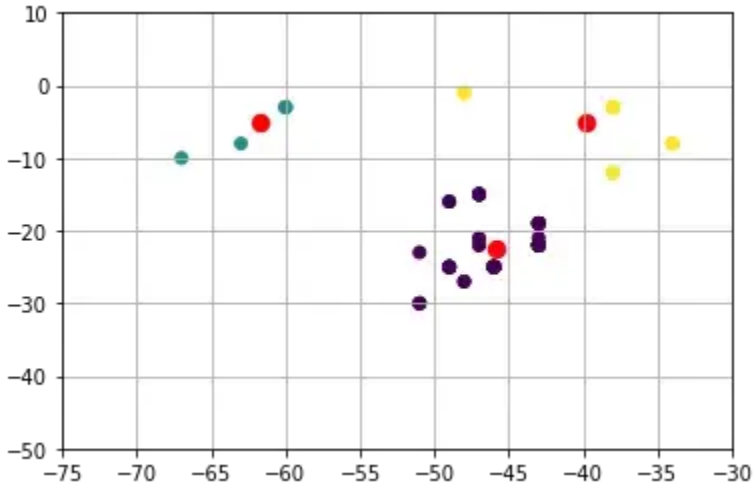


...mas o que importa é que depois de um número de iterações, ele vai achar o nosso ponto ótimo.

Para o exemplo abaixo, vou definir 100 iterações:

```
kmeans = KMeans(n_clusters = 3, #numero de clusters

init = 'random', n_init = 10, #algoritmo que define a posição dos
clusters de maneira mais assertiva

max_iter = 100) #numero máximo de iterações

pred_y = kmeans.fit_predict(novo_dataset)

plt.scatter(novo_dataset[:,1], novo_dataset[:,0]) #posicionamento
dos eixos x e y
```

```
plt.grid() #função que desenha a grade no nosso gráfico

plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,0
], s = 70, c = 'red') #posição de cada centroide no gráfico

plt.show()
```



Dá pra ficar ainda melhor

Aqui ainda ficou faltando tratar algumas situações, que pretendo abordar nos próximos posts, com por exemplo:

- Como escolher o número de clusters (método elbow)
- Como calcular a distância entre os dados: distância euclidiana
- Qual é a condição de parada ideal (nada de loop infinito)

Também é válido dizer que fizemos uma análise de dados utilizando apenas dois tipos de dados: localização e distância entre lojas. Isso é claramente insuficiente para uma análise “de verdade”. Certamente que Beruri e Humaitá não são polos logisticos e temos motivos claros para isso. Nesse texto a gente trata mais sobre esse assunto de qualidade e quantidade de dados.

Um hub logístico em Beruri é a prova de que o K-Means, sozinho, não funciona

Um texto sobre Ciência de Dados, e não “Ciência de Algoritmos”

medium.com

Se gostou ou encontrou algum erro, deixe nos comentários, afinal é só o meu primeiro post sobre ML por aqui. O notebook desse estudo está no GitHub, neste link.