

# TP555 - AI/ML

## Lista de Exercícios #9

### k-Means

1. Cite alguns exemplos de aplicações reais do algoritmo k-Means.
2. Neste exercício, você irá utilizar o algoritmo k-Means com  $k = 3$  para encontrar manualmente os centróides ótimos para o conjunto de dados de treinamento abaixo. Considere os centróides iniciais, C0, C1 e C2, dados ao lado. Utilize a **distância Euclidiana** para encontrar o cluster a que cada exemplo de entrada pertence. Apresente todos os cálculos necessários para se encontrar os centróides ótimos.

x1	x2
1	4
4	3
4	5
3	6
6	7
3	3
2	5
2	2
2	3

C0		C1		C2	
x1	x2	x1	x2	x1	x2
5	3	1	3	3	4

Em seguida, faça o seguinte

- A. Crie uma figura mostrando os dados de treinamento.
- B. Utilizando os centróides iniciais dados acima, instancie um objeto da classe KMeans da biblioteca SciKit-Learn.

```
km = KMeans(n_clusters=3, init=init_clusters)
```

- C. Treine o modelo e imprima os centróides ótimos. Os valores encontrado pelo KMeans devem ser os mesmos que você encontrou manualmente. Os valores ótimos podem ser impressos como mostrado abaixo.

```
for i in range(0,3):  
    print('Centroid %d: (%1.2f, %1.2f)' % (i,km.cluster_centers_[i][0],km.cluster_centers_[i][1]))
```

- D. Quantas iterações foram necessárias para se treinar o modelo? (Dica: a documentação da classe KMeans pode ser acessada via:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>)

- E. Crie uma figura com os dados de treinamento indicando através de cores ou marcadores diferentes à que clusters cada um deles pertence além de mostrar os centróides encontrados pelo k-Means.
3. Crie um conjunto de dados de treinamento utilizando a função **make\_blobs** como mostrado abaixo.

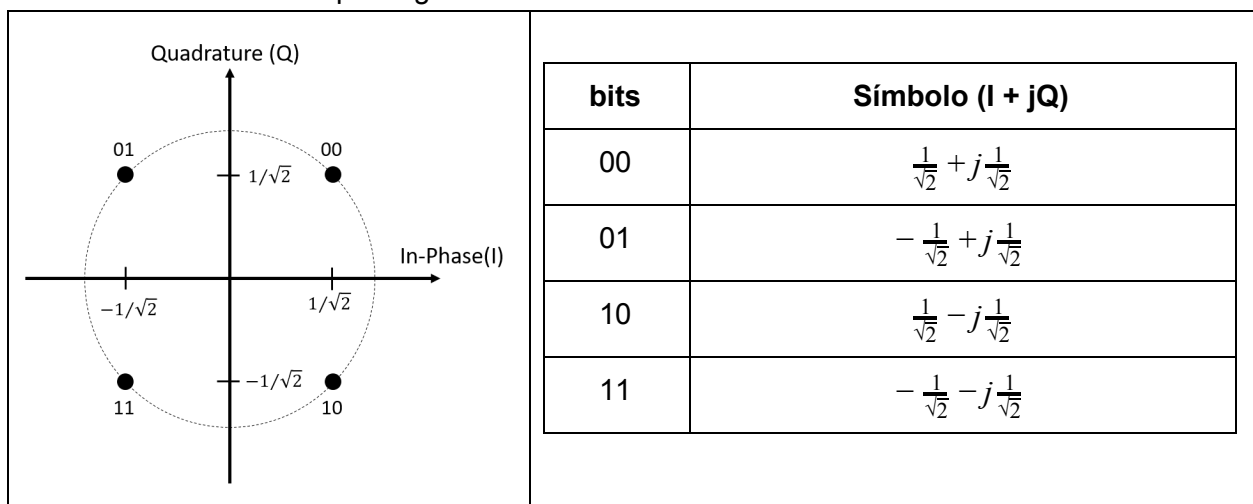
```
X, y = make_blobs(n_samples=150, n_features=2, centers=5, cluster_std=1.0, shuffle=True, random_state=42)
```

Em seguida, faça o seguinte

- A. Crie uma figura mostrando os dados de treinamento.
- B. Após inspecionar a figura, decida quantos clusters devem ser utilizados com o algoritmo do k-Means.
- C. Instancie um objeto da classe KMeans da biblioteca SciKit-Learn.

```
km = KMeans(n_clusters=????, init=init_clusters)
```

- D. Treine o modelo e imprima os centróides ótimos.
- E. Quantas iterações foram necessárias para se treinar o modelo? (Dica: a documentação da classe KMeans pode ser acessada via: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>)
- F. Crie uma figura com os dados de treinamento indicando através de cores ou marcadores diferentes à que clusters cada um deles pertence além de mostrar os centróides encontrados pelo k-Means.
4. Neste exercício, você irá utilizar o algoritmo do k-Means para clusterizar os dados da modulação digital QPSK, ou seja, realizar a detecção de símbolos QPSK. Os símbolos QPSK são dados pela figura e tabela abaixo.



O resultado do seu 'clusterizador' (neste caso, um detector) pode ser comparado com a curva da taxa de erro de símbolo (SER) teórica, a qual é dada por

$$SER = \text{erfc}\left(\sqrt{\frac{E_s}{2N_0}}\right) - \frac{1}{4}\text{erfc}\left(\sqrt{\frac{E_s}{2N_0}}\right)^2$$

Utilizando a classe KMeans do módulo cluster da biblioteca sklearn, faça o seguinte

- A. Construa um detector para realizar a detecção dos símbolos QPSK.
    - a. Gere  $N = 1000000$  símbolos QPSK aleatórios.
    - b. Passe os símbolos através de um canal AWGN.
    - c. Detecte a probabilidade de erro de símbolo para cada um dos valores do vetor  $E_s/N_0 = [-2, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]$ .
  - B. Apresente um gráfico comparando a SER simulada e a SER teórica versus os valores de  $E_s/N_0$  definidos acima.
  - C. Podemos dizer que a curva simulada se aproxima da curva teórica da SER?
- (Dica: Como a ordem dos **centróides** encontrados pelo k-Means é aleatória, o valor do símbolo que o **centróide** representa pode ser encontrado através de estimativa por máxima verossimilhança (ML), ou seja, testa-se o **centróide** de um símbolo detectado contra todos os símbolos possíveis, sendo o símbolo escolhido aquele que apresentar o menor erro.)
- (Dica: A função **erfc** pode ser importada da seguinte forma: *from scipy.special import erfc*).
- (Dica: A função **train\_test\_split** pode dividir qualquer número de vetores de entrada em vetores de treinamento e teste. Veja o exemplo abaixo onde três vetores de entrada, a, e c, são divididos em vetores de treinamento e teste.

```
# Split array into random train and test subsets.
a_train, a_test, b_train, b_test, c_train, c_test = train_test_split(a, b, c, random_state=42)
```

Para mais informações, leia a documentação da função **train\_test\_split**: [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

(Dica: Uma rápida revisão sobre taxa de erro de símbolo pode ser encontrada no link: <http://www.dsplg.com/2007/11/06/symbol-error-rate-for-4-qam/>).

5. Neste exercício, você irá aprender e utilizar 2 métodos para se escolher o parâmetro k, ou seja, o número de clusters. Crie um conjunto de dados utilizando o trecho de código abaixo.

```
N = 1000
# Generating the sample data from make_blobs
# This particular setting has one distinct cluster and 3 clusters placed close
# together.
X, y = make_blobs(n_samples=N, n_features=2, centers=4, cluster_std=1,
center_box=(-10.0, 10.0), shuffle=True, random_state=1)
```

Leia as referências abaixo para aprender sobre os métodos do cotovelo e da silhueta. Em seguida, faça o seguinte

- A. Plote os dados do conjunto de testes.
- B. Visualmente, quantos clusters você acha que seriam necessários para agrupar os dados?
- C. Utilizando o método do cotovelo, encontre o valor mais apropriado para k.
- D. Com o(s) resultado(s) do método do cotovelo, crie uma figura com os dados de treinamento indicando através de cores ou marcadores diferentes à que clusters

cada um deles pertence além de mostrar os centróides encontrados pelo k-Means.

- E. Utilizando o método da silhueta, encontre o(s) valor(es) mais apropriado(s) para k.
- F. Com o(s) resultado(s) do método da silhueta, crie uma figura com os dados de treinamento indicando através de cores ou marcadores diferentes à que clusters cada um deles pertence além de mostrar os centróides encontrados pelo k-Means.

### **Referências**

[1] 'Elbow Method', <https://jtemporal.com/kmeans-and-elbow-method/>

[2] 'Elbow and Silhouette Methods',  
<https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb>

[3] 'Elbow and Silhouette Methods',  
<https://medium.com/@masarudheena/4-best-ways-to-find-optimal-number-of-clusters-for-clustering-with-python-code-706199fa957c>