

TP555 - Inteligência Artificial e  
Machine Learning:  
***k-Vizinhos mais Próximos***



**Inatel**

Felipe Augusto Pereira de Figueiredo  
felipe.figueiredo@inatel.br

## Motivação



- Predizer se o mercado de ações irá subir ou cair.
- Análise de crédito para diferenciar entre clientes de baixo e alto risco.
- Sistemas de recomendação.

Além dos exemplos de classificação que nós vimos na última aula, (Classificação de emails entre SPAM e pessoal (HAM), Detecção de símbolos (classificação de símbolos), Reconhecimento de dígitos escritos à mão, Classificação de texto), podemos também:

- Predizer se o mercado de ações irá subir (associado à um touro) ou cair (associado à um urso).
- Análise de crédito (pontuação de crédito): Diferenciar entre clientes de baixo e alto risco
- Sistemas de recomendação de produtos (filmes, bebidas, etc.)

## k-vizinhos mais próximos (k-NN)

- O algoritmo k-NN (do inglês, k-Nearest Neighbours) é uma das estratégias mais simples para se atacar problemas tanto de **classificação** quanto de **regressão**.
- Ele é um algoritmo do tipo **não-paramétrico**, pois diferentemente do que vimos até o momento, não há um **modelo** a ser treinado, tampouco se faz qualquer suposição a respeito dos dados.
- **Funcionamento:** o algoritmo k-NN necessita que todos os exemplos de treinamento,  $\mathbf{x}(i) = [x_1(i) \dots x_K(i)] \in \mathbb{R}^{K \times 1}$ , e seus respectivos rótulos  $y(i)$ ,  $i = 0, \dots, N - 1$  (i.e., as respostas desejadas) sejam armazenados em memória. Em seguida, dado um exemplo de entrada  $\mathbf{x}'$ , a saída para este exemplo dependerá dos rótulos associados aos  $k$  exemplos de treinamento mais próximos do exemplo de entrada  $\mathbf{x}'$  no espaço de atributos.
- Por exemplo, nós podemos tomar a média aritmética dos rótulos/saídas dos vizinhos mais próximos:

$$\hat{y}(\mathbf{x}') = \frac{1}{k} \sum_{\mathbf{x}(i) \in N_k(\mathbf{x}')} y(i),$$

onde  $N_k(\mathbf{x}')$  é a vizinhança de  $\mathbf{x}'$ , formada pelos exemplos de treinamento  $\mathbf{x}(i)$  que correspondem aos  $k$  vizinhos mais próximos de  $\mathbf{x}'$ .

- **OBS.:** Não confundam o número de atributos  $K$  com o número de vizinhos mais próximos,  $k$ .

k-NN é poderoso porque não assume nada sobre os dados, exceto que a medida da distância pode ser calculada consistentemente entre duas instâncias. Assim, ele é chamado de **não-paramétrico** ou **não-linear**.

O algoritmo k-NN usa 'similaridade de atributos' para prever os valores de quaisquer novos exemplos de dados. Isso significa que o novo exemplo recebe um valor com base na sua proximidade com os pontos no conjunto de treinamento.

## k-Vizinhos mais Próximos (k-NN)

- Portanto, o uso do k-NN envolve a definição de:
  - Uma **métrica de distância** que deve ser calculada no **espaço de atributos** a fim de identificar os vizinhos mais próximos.
  - Um valor para o parâmetro  $k$ , ou seja, a escolha do número de vizinhos que devem ser levados em consideração para a geração da saída correspondente ao exemplo de entrada,  $x'$ .
- Como  $k$  é um **hiperparâmetro** do algoritmo k-NN, pode-se utilizar, por exemplo, a abordagem da **validação cruzada  $q$ -fold** para encontrar o melhor valor de  $k$  (para não haver confusão com o parâmetro  $k$  do k-NN, utilizei  $q$  aos invés de  $k$  para especificar o número de pastas/dobras do  **$k$ -fold**).
- Devido a estas características, o k-NN é visto como um **algoritmo de aprendizado competitivo**, uma vez que os elementos do modelo (que são os próprios exemplos de treinamento) competem entre si pelo direito de influenciar a saída do algoritmo quando a **medida de similaridade (distância)** é calculada para cada novo dado de entrada.

## k-Vizinhos mais Próximos (k-NN)

- Além disso, o k-NN explora a ideia conhecida como **lazy learning**, uma vez que o algoritmo não “*constrói*” um **modelo** até o instante em que uma predição é solicitada.
- O k-NN segue o paradigma de **aprendizado-baseado em instâncias**, onde ao invés de obter/treinar um modelo à partir do conjunto de treinamento que generalize, ele compara novos exemplos de entrada com os exemplos do conjunto de treinamento armazenados na memória.
- O k-NN tem como desvantagem o fato de que todos os dados de treinamento precisam ser armazenados e consultados para se identificar os vizinhos mais próximos.
- Portanto, a **predição** será demorada dependendo do tamanho do conjunto de treinamento, pois deve-se calcular a **distância** entre o exemplo de entrada e todos os exemplos do conjunto de treinamento.
- Além disto, como vimos, o conjunto de treinamento deve ser armazenado em memória, e caso esse conjunto seja muito grande, pode não haver memória o suficiente para armazená-lo.

O **lazy learning** é uma abordagem de aprendizado no qual a generalização dos dados de treinamento é, em teoria, atrasada até que uma consulta seja feita ao sistema, em oposição ao **eager learning**, em que o sistema tenta generalizar os dados de treinamento antes de receber consultas.

**Eager learning** tem uma etapa de treinamento do modelo. **Lazy learning** não tem uma fase de treinamento.

## Métricas de distância

- **Definição:** Uma métrica de distância fornece a distância entre os elementos de um conjunto. Se a distância é igual a zero, os elementos são equivalentes; caso contrário, os elementos são diferentes um do outro.
- Existem várias **métricas de distância**, mas vamos discutir apenas as mais utilizadas.
- **Distância de Minkowski:** é uma métrica no **espaço vetorial normado** (ou seja, um **espaço vetorial** no qual uma **norma**,  $p(\cdot)$ , é definida) que satisfaz algumas propriedades. A **norma vetorial**,  $p(\cdot)$ , é uma função que mapeia  $\mathbb{R}^{K \times 1} \rightarrow \mathbb{R}$  e que exibe algumas propriedades que veremos à seguir.
- Sejam 2 vetores,  $\mathbf{v}$  e  $\mathbf{u} \in \mathbb{R}^{K \times 1}$ , a norma  $p(\cdot)$  dos vetores é uma função com valores não-negativos com as seguintes propriedades
  - $p(\mathbf{v} + \mathbf{u}) \leq p(\mathbf{v}) + p(\mathbf{u})$  (ou seja, a norma satisfaz a desigualdade do triângulo).
  - $p(a\mathbf{v}) = |a|p(\mathbf{v})$ , para todo  $a \in \mathbb{R}$  (ou seja, a norma é absolutamente escalável).
  - Se  $p(\mathbf{v}) = 0$ , então  $\mathbf{v} = \mathbf{0}$ , ou seja, o **vetor nulo** (ou seja, a norma é positiva definida).

- **Desigualdade de Triângulos:** se a distância é uma norma, a distância calculada entre dois pontos será sempre uma linha reta.
- O **vetor nulo** tem comprimento nulo, ou seja, norma nula.
- **Fator de escala**,  $a$ : a direção do vetor não muda quando você o multiplica por um número positivo, embora seu comprimento seja alterado.

Observe que as propriedades acima não determinam uma forma única de uma função de norma, de fato, existem muitas normas válidas diferentes.

## Distância de Minkowski

- A **distância de Minkowski** de ordem  $p$  é calculada usando-se a equação abaixo

$$p(x; y) = \left( \sum_{i=1}^K |x_i - y_i|^p \right)^{1/p}.$$

- A **distância de Minkowski** é uma métrica de **distância generalizada**, ou seja, podemos manipular a equação acima para calcular a distância entre dois pontos de dados de formas diferentes.
- **Casos particulares:**
  - Para  $p = 1$ , temos a **distância de Manhattan**:  $p(x; y) = \sum_{i=1}^K |x_i - y_i|$ .
  - Para  $p = 2$ , temos a **distância Euclidiana**:  $p(x; y) = \sqrt{\sum_{i=1}^K |x_i - y_i|^2}$ .

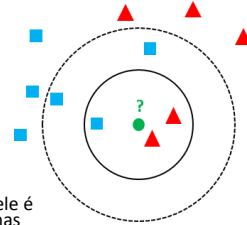
## k-NN para classificação

- Com relação ao problema da classificação, a saída da equação

$$\hat{y}(x') = \frac{1}{k} \sum_{x(i) \in N_k(x')} y(i),$$

gerada pelo k-NN equivale a tomar o voto majoritário dos  $k$  vizinhos mais próximos de  $x'$ .

- Ou seja, um novo exemplo de entrada,  $x'$ , é classificado como sendo pertencente à classe que contiver o maior número de vizinhos de  $x'$ .
- Exemplo de classificação com k-NN:** na figura ao lado, o exemplo de teste (ponto verde) pode ser classificado como pertencente à classe quadrados azuis ou à classe triângulos vermelhos. Se  $k = 3$  (círculo com linha sólida), ele é atribuído à classe de triângulos vermelhos pois existem 2 triângulos e apenas 1 quadrado dentro do círculo interno. Se  $k = 5$  (círculo tracejado), ele é atribuído à classe de quadrados azuis (3 quadrados vs. 2 triângulos dentro do círculo externo).
- Observação:** uma técnica bastante útil é atribuir pesos diferentes à contribuição de cada vizinho à decisão final de tal forma que vizinhos mais próximos contribuem mais do que vizinhos mais distantes. Portanto, uma alternativa usual é definir os pesos como sendo inversamente proporcionais às distâncias dos vizinhos ao exemplo de entrada  $x'$ .





## Exemplo: Classificação k-NN com SciKit-Learn

```
# Import all necessary libraries.
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
```

```
# Number of examples.
N = 100
centers = [[-1, 0], [0, 0]]
```

```
# Create a 2-class dataset for classification.
x, y = make_blobs(n_samples=N, centers=centers, random_state=42)
```

```
# Create an instance of NeighborsClassifier and fit the data.
clf = KNeighborsClassifier(k, weights='distance')
```

```
# Train the classifier.
clf.fit(x, y)
```

```
# Predict.
clf.predict(x)
```

Figura mostra a fronteira de decisão criada pelo k-NN para diferentes valores de  $k$ . Como podemos ver, à medida que  $k$  aumenta, a fronteira tende a ficar mais suave e menos regiões isoladas são criadas para cada classe.

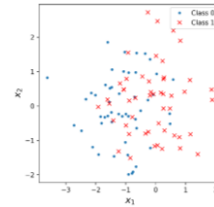
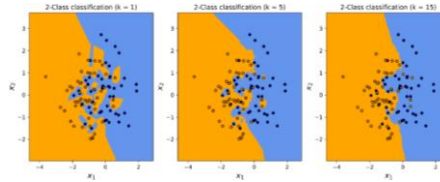


Figura com a distribuição dos exemplos de treinamento.



Exemplo: knn\_classification\_2\_classes.ipynb

**Exemplo:** knn\_classification\_2\_classes.ipynb

## k-NN para regressão

- Seja  $N_k(\mathbf{x}')$  o conjunto formado pelos  $k$  exemplos de treinamento  $\mathbf{x} \in \mathbb{R}^{K \times 1}$  mais próximos ao exemplo de entrada  $\mathbf{x}'$ . As saídas associadas a estes exemplos de treinamento são denotadas por  $y_j(\mathbf{x} \in N_k(\mathbf{x}'))$ ,  $j = 1, \dots, k$ .
- Desta forma, quando utilizado para **regressão**, a saída do algoritmo k-NN para um novo exemplo de entrada  $\mathbf{x}'$  pode ser escrita de forma geral como

$$\hat{y}(\mathbf{x}') = \frac{\sum_{j=1}^k w_j y_j(\mathbf{x} \in N_k(\mathbf{x}'))}{\sum w_j},$$

onde  $w_j$ ,  $j = 1, \dots, k$  representa o peso associado ao  $j$ -ésimo vizinho de  $\mathbf{x}'$ .

- Os pesos associados aos vizinhos podem ser **uniformes** ou **inversamente proporcionais à distância**.

## Exemplo: Regressão k-NN com SciKit-Learn

```
# Import all necessary libraries.
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor

# Generate sample data.
M = 40
np.random.seed(42)
X = np.sort([5*np.random.rand(M, 1) - 3], axis=0)
T = np.linspace(-3, 3, 100)[0, np.newaxis]
y = (0.5*X**2 + X + 2).ravel()
y_noisy = (0.5*X**2 + X + 2).ravel()
y += np.random.randn(M)

# Fit regression model
n_neighbors = 5

plt.figure(figsize=(15,5))
for i, weights in enumerate(['uniform', 'distance']):
    knn = KNeighborsRegressor(n_neighbors, weights=weights)
    y_ = knn.fit(X, y).predict(T)
    plt.subplot(1, 2, i + 1)
    plt.scatter(X, y, color='darkorange', label='noisy data')
    plt.plot(X, y_orig, color='red', label='original data')
    plt.plot(T, y_, color='navy', label='prediction')
    plt.axis('tight')
    plt.legend()
    plt.title("KNeighborsRegressor (k = %i, weights = '%s') % (n_neighbors, weights))

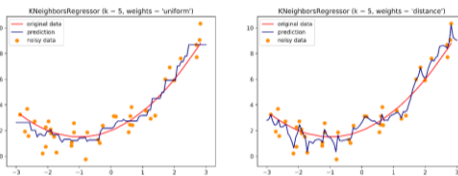
plt.show()
```

Importa classe  
KNeighborsRegressor.

Cria dados para regressão.

Peso de cada vizinho é  
uniforme ou o inverso da  
distância para o exemplo de  
entrada.  
Armazena conjunto de dados na  
memória e realiza predição.

A figura abaixo compara a regressão feita com o algoritmo k-NN quando os pesos associados aos vizinhos são **uniformes** (figura da esquerda) **inversamente proporcionais à distância** (figura da direita).



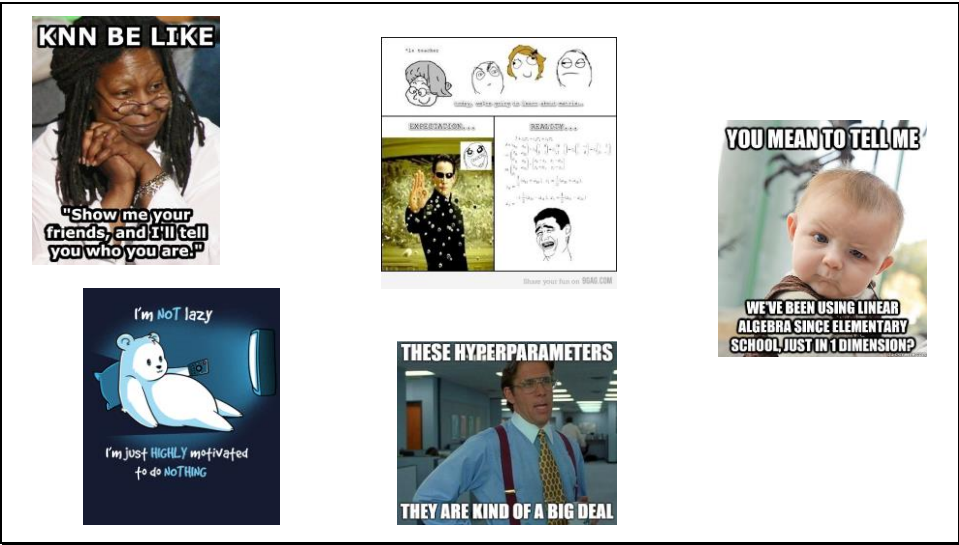
Exemplo: knn\_regression.ipynb

Exemplo: knn\_regression.ipynb

## Avisos

- Exemplos já estão disponíveis no site.
- Lista #6 já está disponível no site.
- Lista #5 pode ser entregue até dia 15/05.
- Vou enviar um vídeo sobre Árvores de Decisão ainda esta semana .

Obrigado!



Figuras

