FIAP GRADUAÇÃO

INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL

Profª . Miguel Bozer da Silva
profmiguel.silva@fiap.com.br

2020

Regressão Logística

- Alguns modelos de regressão também podem ser utilizados para classificação, um exemplo disso é a **Regressão Logística**

- A Regressão Logística é comumente utilizada para estimar a probabilidade de um dado valor de entrada pertencer a uma classe particular.
  - Ex.: Probabilidade de ser um SPAM
  - Caso a probabilidade for maior do que 50%, classificamos a entrada como um SPAM (classe 1)
  - Do contrário dizemos que não pertence a classe SPAM (classe 0)
- Definimos dessa forma os ***classificadores binários***

Regressão Logística

- E como isso funciona?
  - Assim como a Regressão Linear, calculamos um conjunto de parâmetros $\boldsymbol{\theta}$ que são multiplicados com a entrada e somamos um termo de bias (θ_0 que não multiplica nenhum dos valores da entrada de dados)
  - A equação da regressão logistica pode ser visualizada a seguir:

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

Regressão Logística

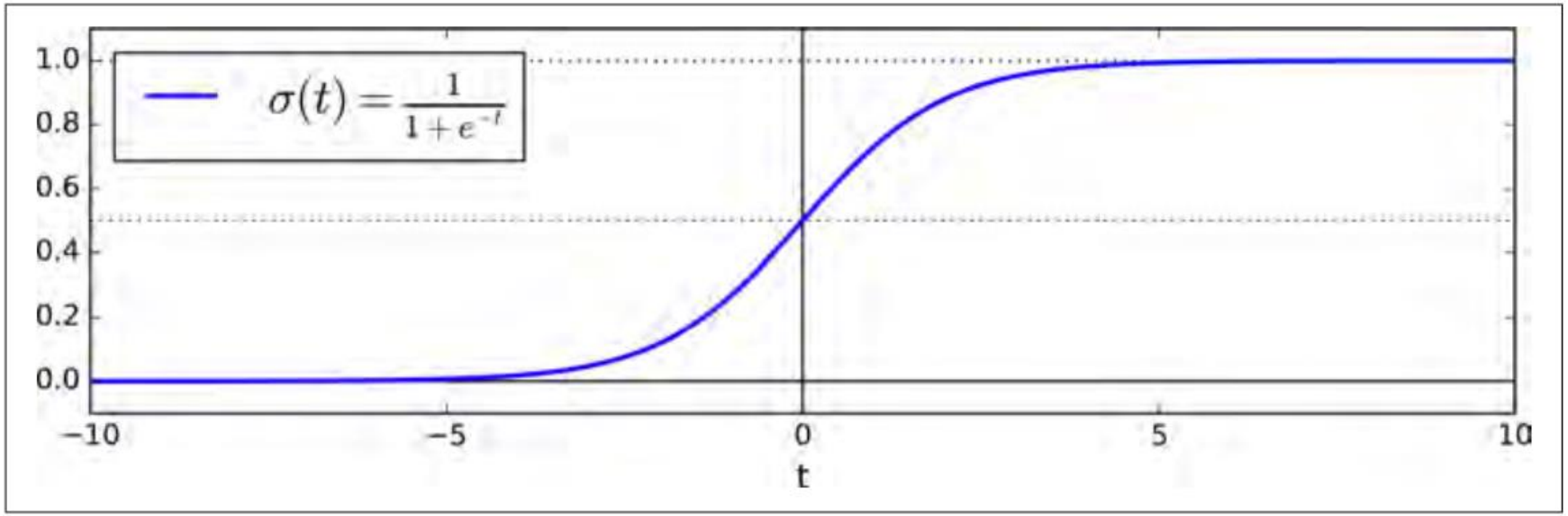
$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

- Onde:
  - $\hat{p}$ seria o valor entre 0 e 1 representando a nossa probabilidade de pertencer a uma dada classe;
  - σ a função sigmóide;
  - θ são os parâmetros da rede
  - $\boldsymbol{x}$ são os valores de entrada

Regressão Logística

- A função Sigmoide pode ser definida como:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

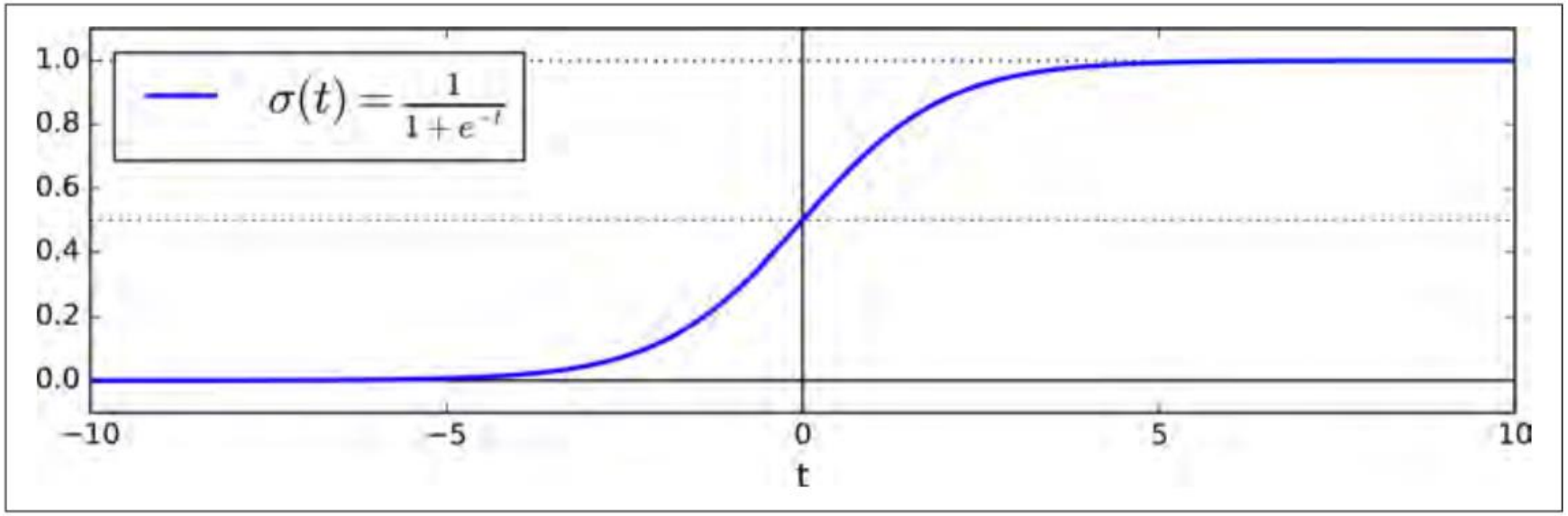


Regressão Logística

- Para podermos fazer a previsão de uma entrada $\boldsymbol{x}$ qualquer utilizamos a regra abaixo:

$$\hat{y} = \begin{cases} 0 \text{ if } \hat{p} < 0.5, \\ 1 \text{ if } \hat{p} \geq 0.5. \end{cases}$$

Regressão Logística



$$\hat{y} = \begin{cases} 0 \text{ if } \hat{p} < 0.5, \\ 1 \text{ if } \hat{p} \geq 0.5. \end{cases}$$

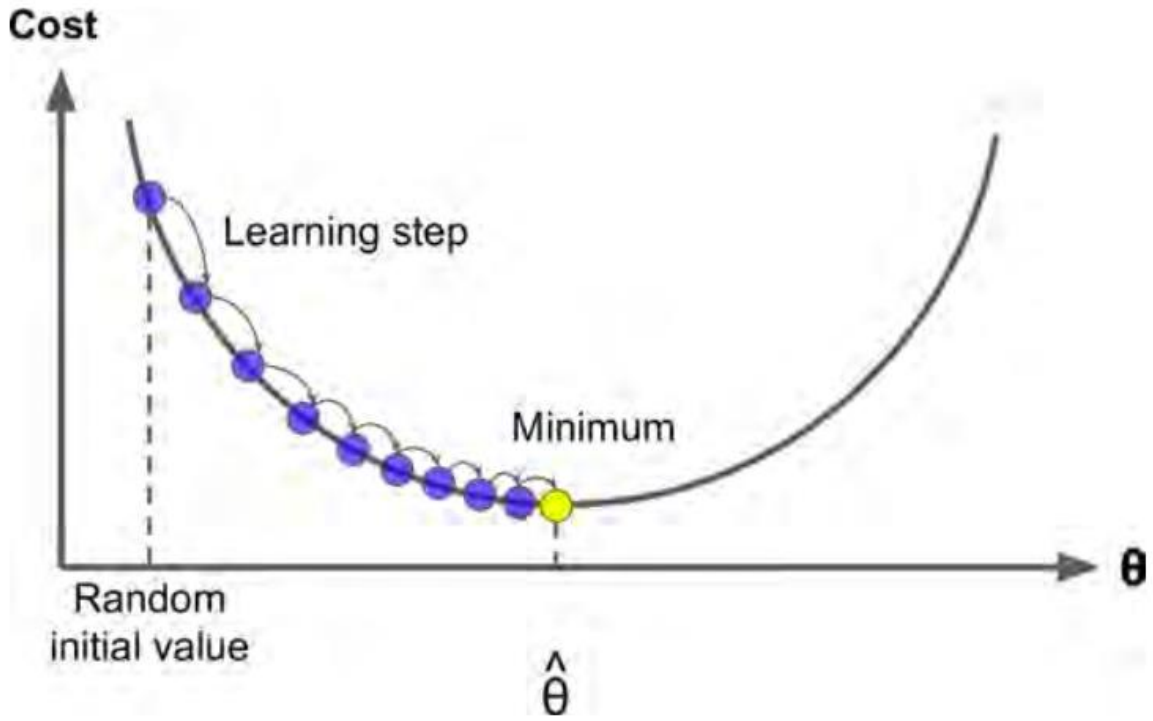
Note que se $\sigma(t) < 0{,}5$ quando $t < 0$ e $\sigma(t) \geq 0{,}5$ quando $t \geq 0$

Logo a Regressão logistica faz a previsão para a classe 1 quando $\theta^T \boldsymbol{x}$ é positivo e para a classe 0 quando é negativo

Regressão Logística

- Temos uma boa ideia do nosso modelo, mas como iremos treiná-lo?

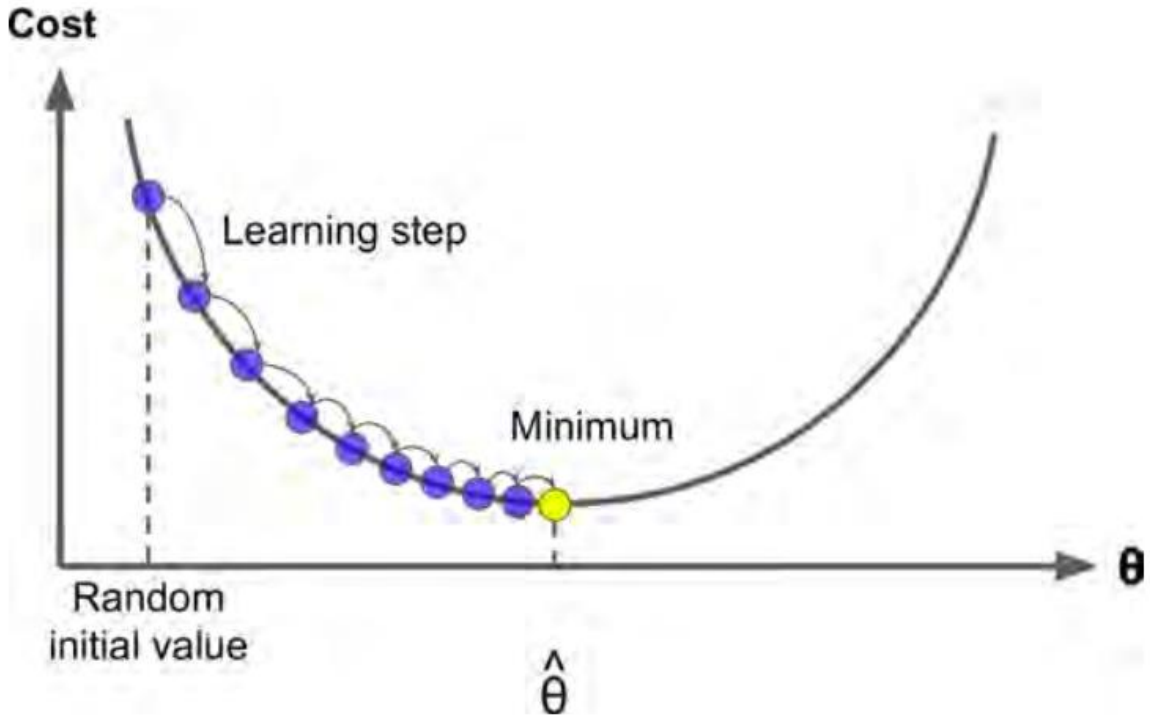
  – Podemos utilizar a idea do algoritmo do Gradiente Descendente!



Regressão Logística

- Temos uma boa ideia do nosso modelo, mas como iremos treiná-lo?

  – Podemos utilizar a idea do algoritmo do Gradiente Descendente!



Com os passos calculados em:

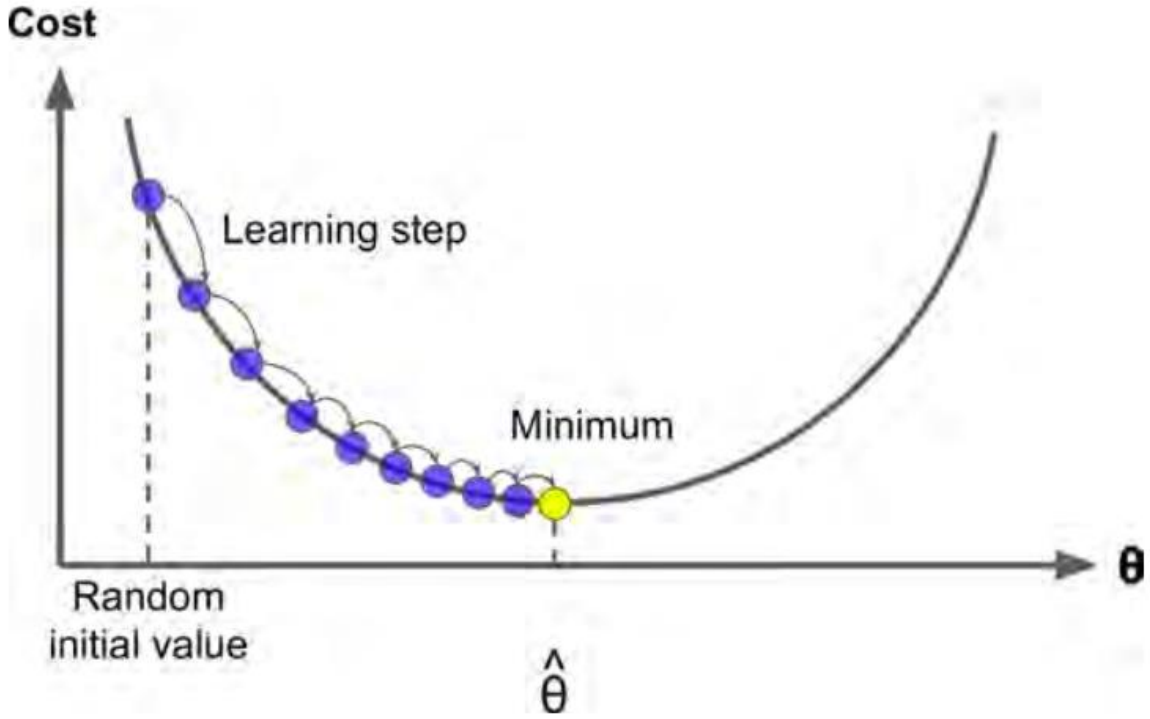
$$\boldsymbol{\theta}^{(next\ step)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$J(\boldsymbol{\theta})$ é a função custo

Regressão Logística

- Temos uma boa ideia do nosso modelo, mas como iremos treiná-lo?

  – A partir de uma função custo que é calculada a cada passo podemos encontrar um valor um valor ótimo para os parâmetros



Com os passos calculados em:

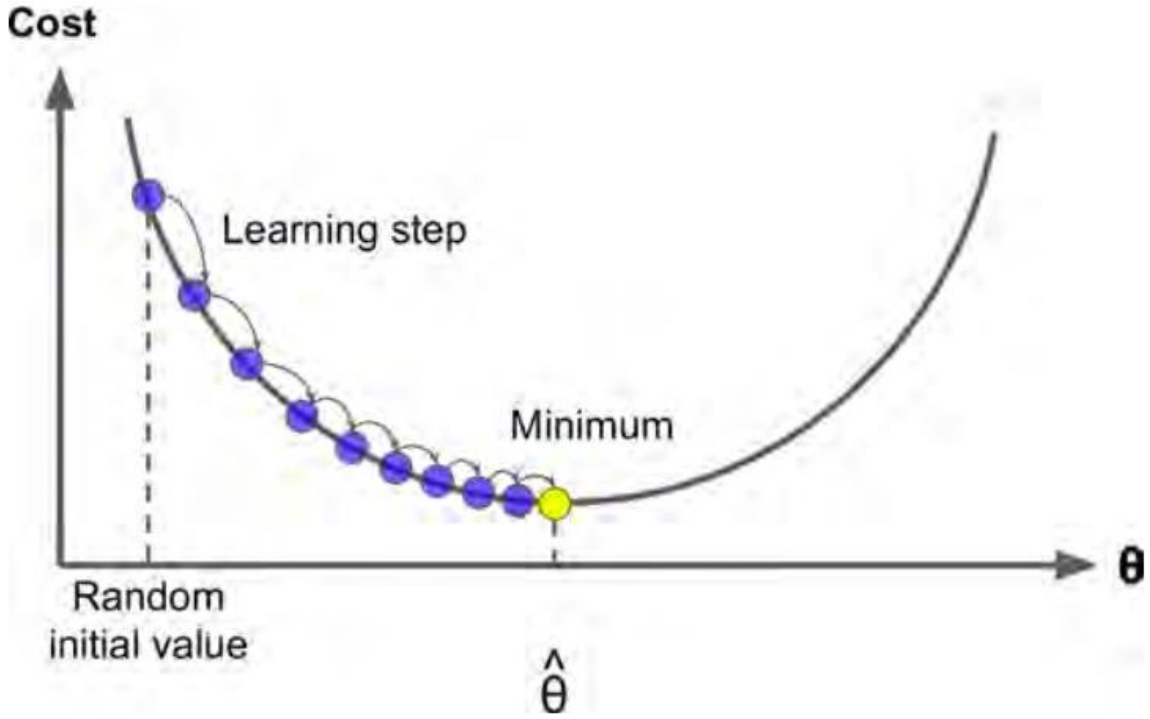
$$\boldsymbol{\theta}^{(next\ step)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$J(\boldsymbol{\theta})$ é a função custo

Regressão Logística

- Temos uma boa ideia do nosso modelo, mas como iremos treiná-lo?

  – **Isso ocorre na fase de treinamento do modelo, onde encontramos os parâmetros para o conjunto de treinamento.**



Com os passos calculados em:

$$\boldsymbol{\theta}^{(next\ step)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$J(\boldsymbol{\theta})$ é a função custo

FIAP GRADUAÇÃO

KNN

kNN – k-Nearest Neighbor

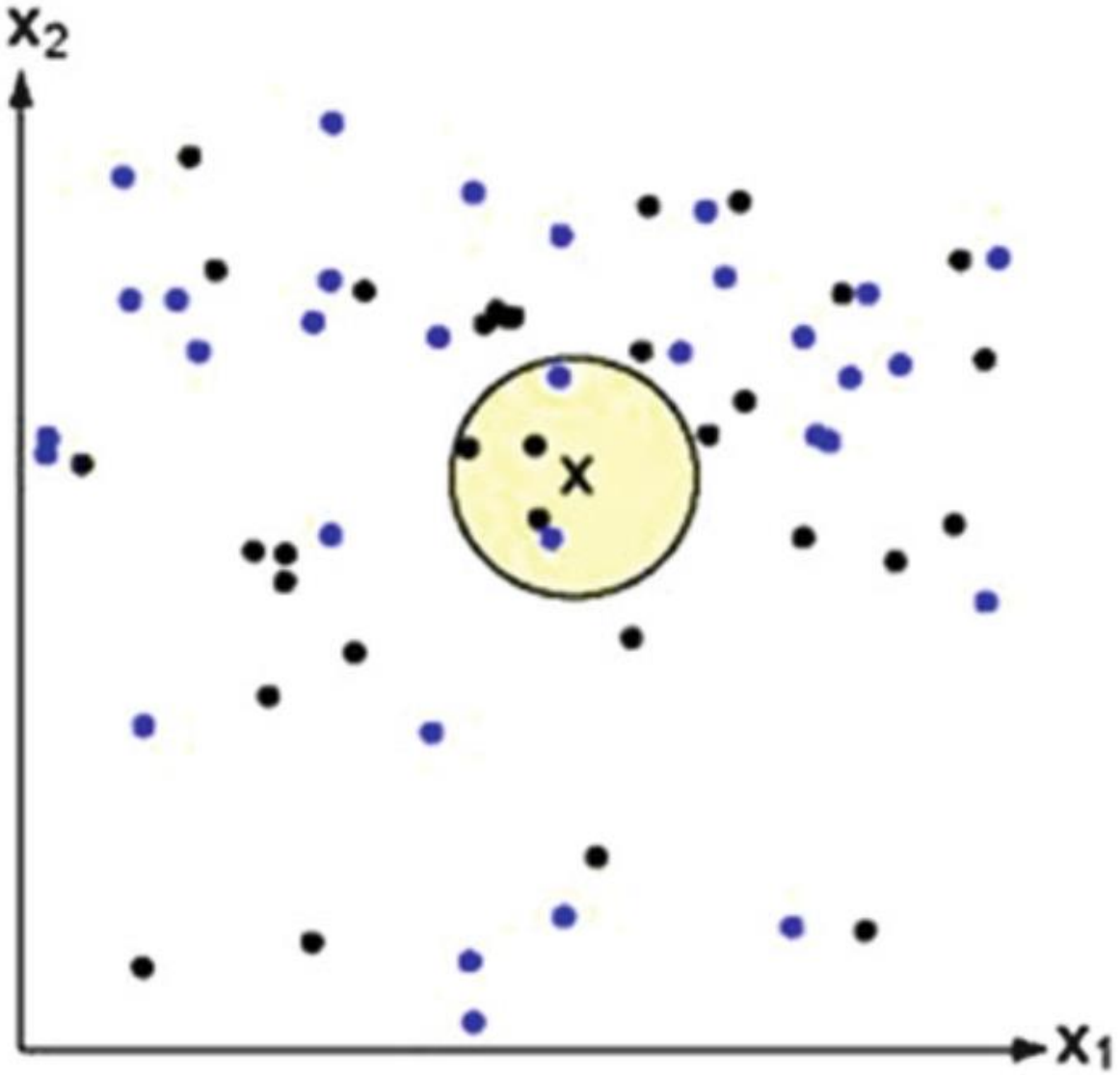
- O k-nearest neighbor ou k-vizinhos mais próximos é um algoritmo de classificação que faz a classificação a partir de uma entre o exemplo que desejo classificar com um conjunto de dados já classificados.

- A ideia basicamente é:
  - Se um animal anda igual a um pato, parece um pato e faz o mesmo som do pato (“quack”), então ele é um pato!
  - Isto é, a classe de um animal desconhecido foi determinada a partir da próximidade das características com um outro animal já classificado

kNN – k-Nearest Neighbor

- O *k*-NN inicia-se a partir do ponto onde se encontra o exemplo $(\mathbf{x})$ que desejamos classificar.
- Na sequência encontramos os *k* vizinhos mais próximos a esse ponto $\mathbf{x}$.
- Entre os *k* viznhos mais próximos verificamos qual a classe mais frequente entre eles (c_i)
- Classificamos o exemplo $\mathbf{x}$ como pertencente a classe c_i

kNN – k-Nearest Neighbor

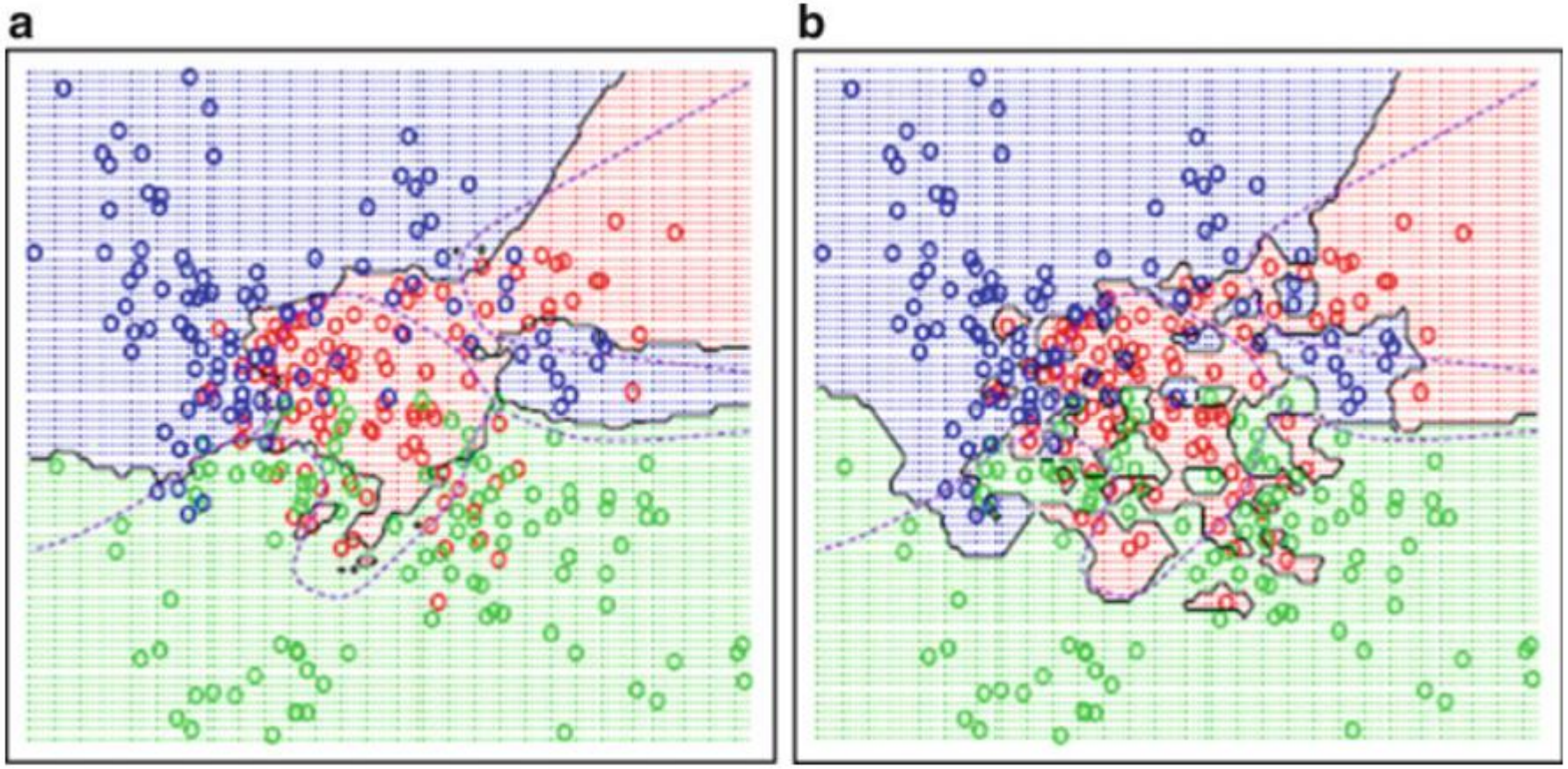


- Exemplo de duas *features* x_1 e x_2 distintas
  - Classe 1 – Preto
  - Classe 2 – Azul
  - Ponto novo para ser classificado representado por um x

- *K* = 5 vizinhos mais próximos de x:
  - 3 da classe 1
  - 2 da classe 2
  - **Logo x pertence a classe 1**

kNN – k-Nearest Neighbor

- Podemos ter mais de duas classes e variar o valor de *k*



k=15

k=1

kNN – k-Nearest Neighbor

- Para duas classes, *k* deve ser ímpar para evitar empates.

- Para mais de duas classes *k* ser ímpar não é o suficiente para evitar empates. Por exemplo 7-NN:
  - C_1 com três exemplos próximos
  - C_2 com três exemplos próximos
  - C_3 com um exemplos próximos

- Nesses casos temos que definir um critério para escolha da classe.

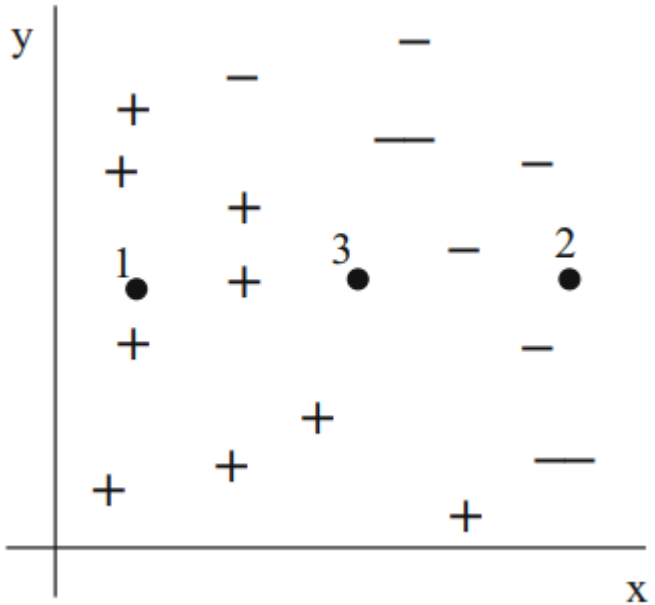
kNN – k-Nearest Neighbor

- Valores maiores de *k* ajudam a evitar empates

- Não há um método de treinamento para o k-NN
  – Sua vantagem se dá no fato de ser intuitivo

kNN – k-Nearest Neighbor

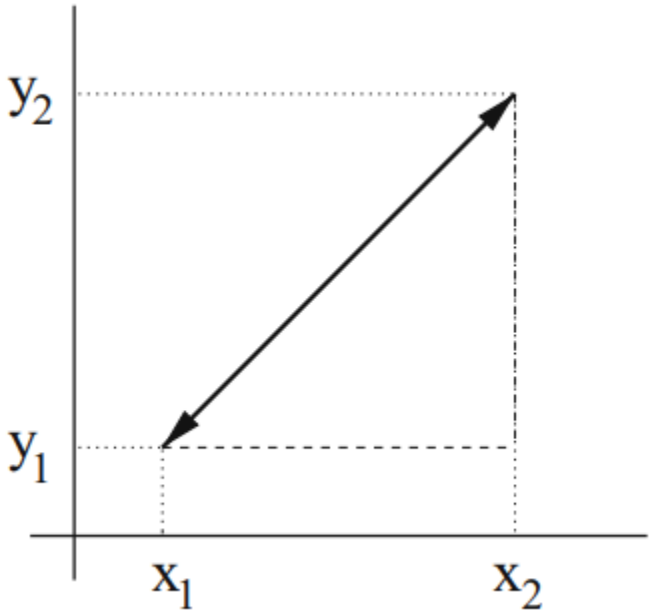
- Certas situações podem ser problemáticas para esse algoritmo:



Como garantir a correta classificação do exemplo 3, sendo que ele está na fronteira entre a classe + e -?

kNN – k-Nearest Neighbor

- Agora que entedemos a ideia do k-NN, como fazer para encontrar os exemplos mais próximos?



Podemos calcular a **distância Euclidiana** do ponto que desejamos classificar com os outros pontos já classificados

kNN – k-Nearest Neighbor

- **Distância Euclidiana:** Em um plano, a distância geométrica entre dois pontos, $\mathbf{x} = (x_1, x_2)$ e **y**$= (y_1, y_2)$ é dada por:

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

- Para um cenário com *n* diferentes *features* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ e **y**$= (y_1, y_2, \ldots y_n)$ temos:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\Sigma_{i=1}^{n}(x_i - y_i)^2}$$

Referências Bibliográficas

- DOUGHERTY, Geoff. **Pattern Recognition and Classification**: an introduction. New York: Springer International Publishing, 2013.

- GÉRON, Aurélien. Hands-On Machine Learning with Scikit-Learn and TensorFlow. Sebastopol: O’reilly Media, 2017

- KUBAT, Miroslav. **An Introduction to Machine Learning**. 2. ed. Ebook: Springer International Publishing, 2017.

FIAP GRADUAÇÃO