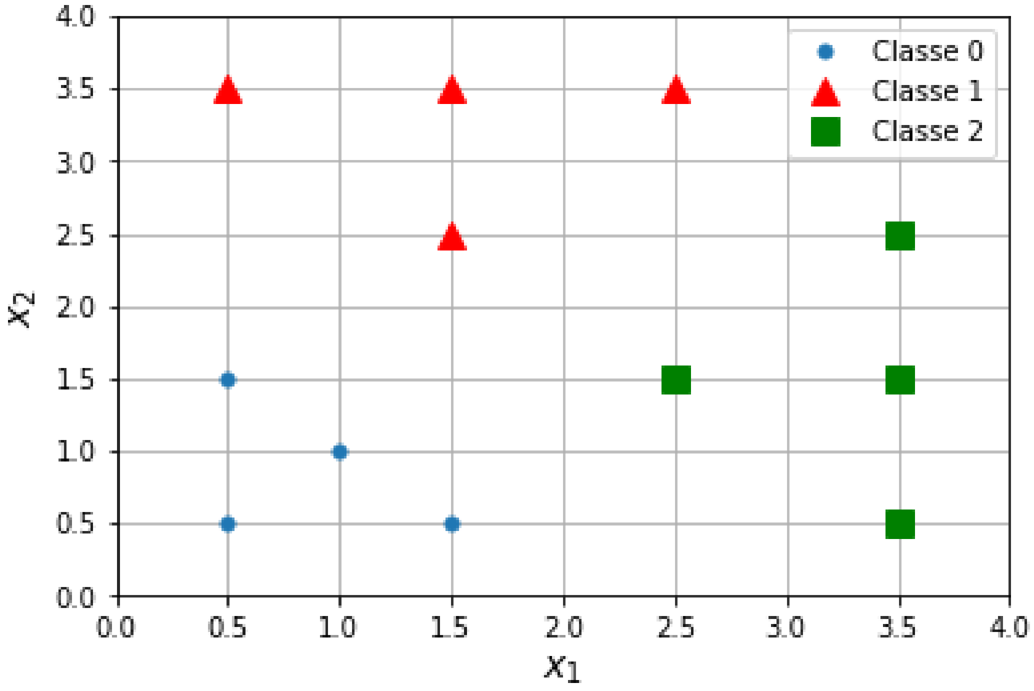
Laboratório #1

Exercício #2

Quantas funções discriminantes?

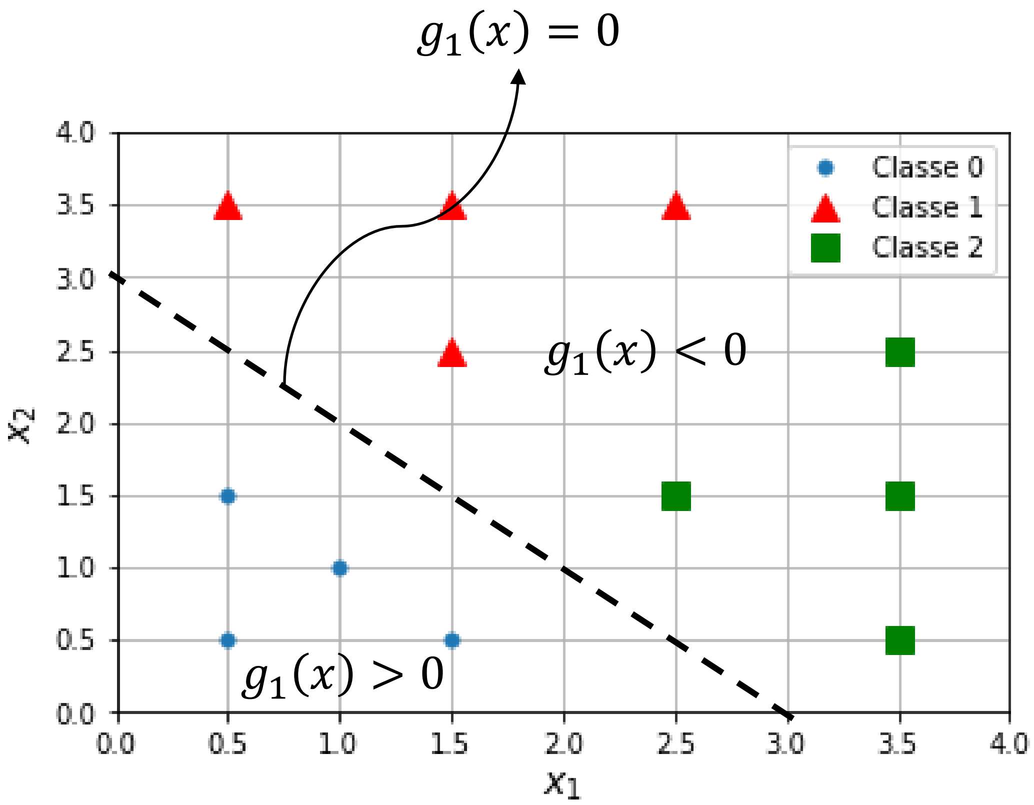


- Este é um problema com ***múltiplas classes***, onde $Q = 3$.
- Como temos três classes, não faz sentido falarmos em classes positiva e negativa, apenas em seus índices: 0, 1 e 2.
- Quantas funções discriminantes são necessárias para separar as classes?
  - No mínimo duas funções, $g_1(\boldsymbol{x})$ e $g_2(\boldsymbol{x})$.
- Qual o formato mais simples?
  - Retas da forma:

$$g_i(\boldsymbol{x}) = a_0 + a_1 x_1 + a_2 x_2, i = 1, 2.$$

- Agora vamos encontrar os pesos de cada uma das funções.

Encontrando os pesos $g_1(\boldsymbol{x})$



- Encontramos os pesos de $g_1(\boldsymbol{x})$ primeiro, pois ela separa a classe 0 perfeitamente das outras duas (1 e 2).
- A ***função discriminante*** que representa esta reta é definida como

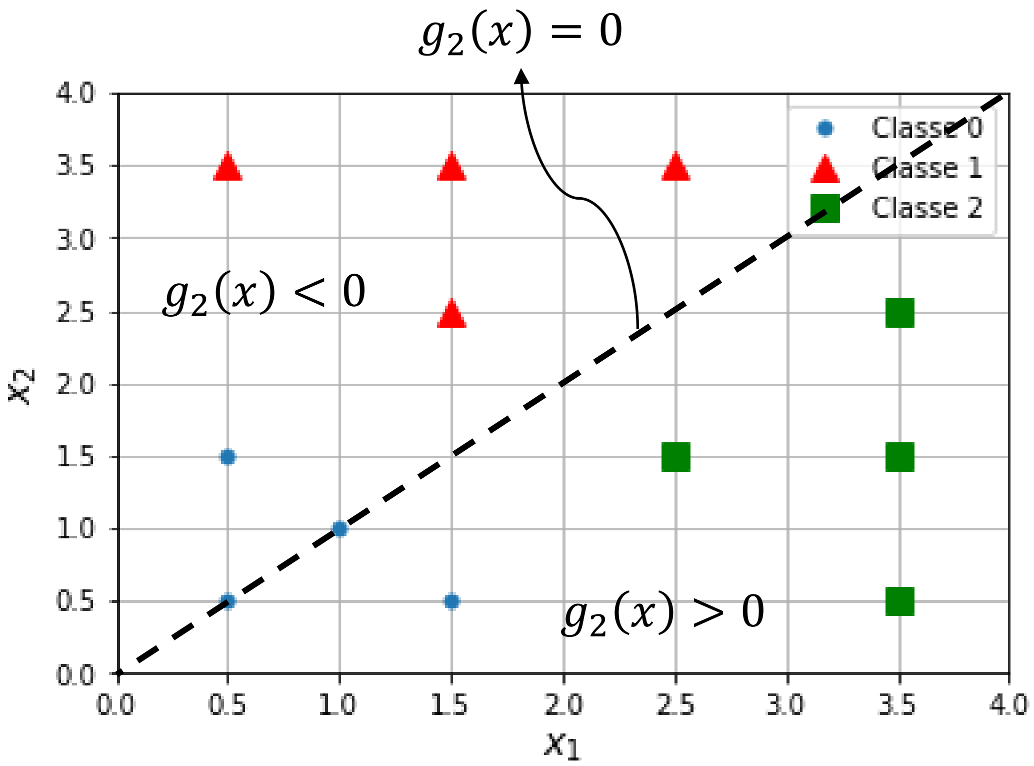
$$g_1(\boldsymbol{x}) = a_0 + a_1x_1 + a_2x_2$$

- Temos 3 incógnitas e 3 equações:
  - $(x_1 = 0, x_2 = 3) \rightarrow 0 = a_0 + 3a_2 \therefore a_0 = -3a_2$
  - $(x_1 = 1, x_2 = 2) \rightarrow 0 = a_0 + a_1 + 2a_2 \therefore a_1 = -(a_0 + 2a_2)$
  - $(x_1 = 2, x_2 = 1) \rightarrow 0 = a_0 + 2a_1 + a_2 \therefore a_1 = -(a_0 + a_2)/2$
- Resolvendo o sistema, encontramos $a_0 = 3$, $a_1 = -1$, $a_2 = -1$, então

$$g_1(\boldsymbol{x}) = 3 - x_1 - x_2$$

- Substituindo alguns valores em $g_1(\boldsymbol{x})$, encontramos as regiões das duas classes que ela separa: classe 0 e a união das classes 1 e 2.

Encontrando os pesos $g_2(\boldsymbol{x})$

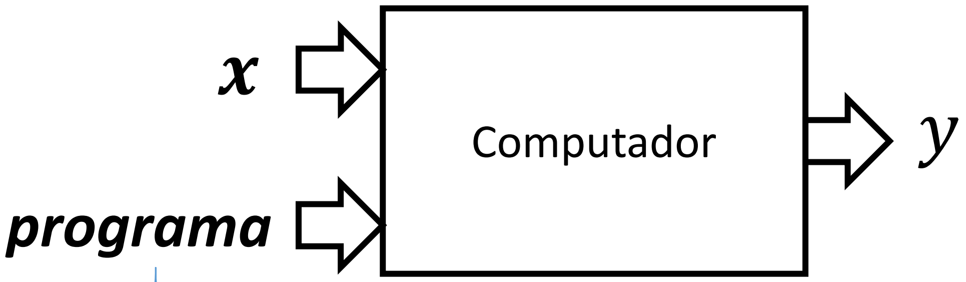


- Na sequência, encontramos os pesos de $g_2(\boldsymbol{x})$, que irá discriminar entre as classes 1 e 2, concluindo a classificação.
- A ***função discriminante*** que representa esta reta é definida como
$$g_2(\boldsymbol{x}) = a_0 + a_1 x_1 + a_2 x_2$$
- Novamente temos 3 incógnitas e 3 equações:
  - $(x_1 = 0, x_2 = 0) \rightarrow 0 = a_0$
  - $(x_1 = 1, x_2 = 1) \rightarrow 0 = a_0 + a_1 + a_2 \therefore a_1 = -a_2$
  - $(x_1 = 2, x_2 = 2) \rightarrow 0 = a_0 + 2a_1 + 2a_2 \therefore a_1 = -a_2$
- Resolvendo o sistema, encontramos $a_0 = 0$, $a_1 = 1$, $a_2 = -1$, então
$$g_2(\boldsymbol{x}) = x_1 - x_2$$

Trecho da função *predict*

```
# Usamos g_1(x) primeiro, pois ela separa exatamente a classe 0 das demais.
if(g_1(x) >= 0):
        y_pred[i] = 0
# Caso quando g_1(x) < 0
else:
        if(g_2(x) < 0):
                y_pred[i] = 1
        # caso quando g_2(x) >= 0
        else:
                y_pred[i] = 2
```

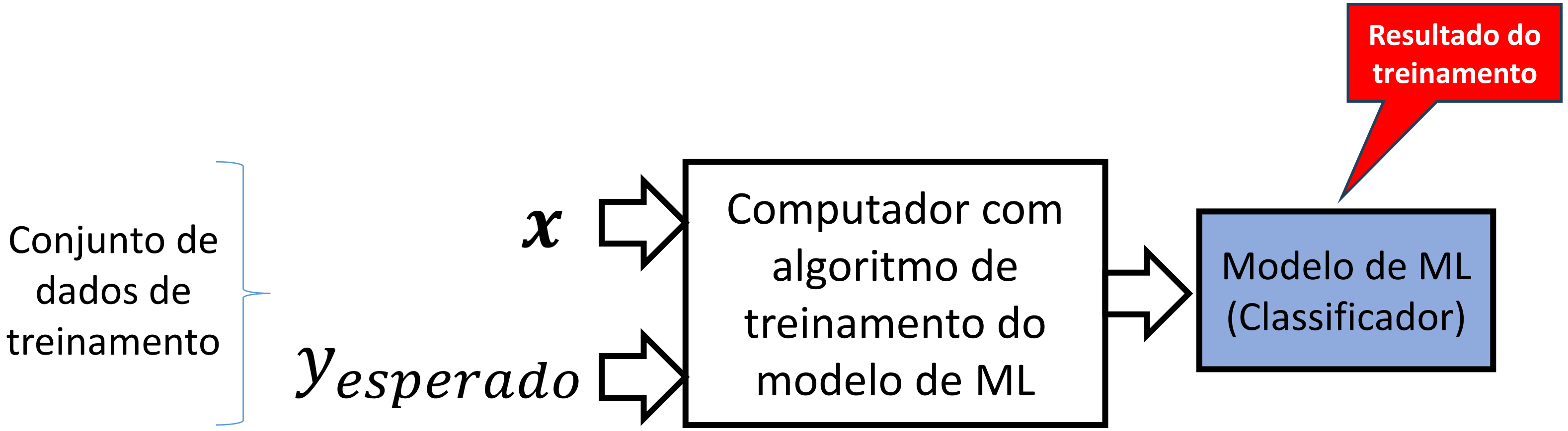
Paradigma da programação tradicional




```
def predict(x):
    if(g1(x) >= 0):
        y_pred[i] = 0
    # Caso quando g1(x) < 0
    else:
        if(g2(x) < 0):
            y_pred[i] = 1
        # caso quando # g2(x) >= 0
        else:
            y_pred[i] = 2
```

- Resolvemos o exercício usando a forma tradicional.
- Fornecemos para o computador as ***entradas*** (i.e., $\boldsymbol{x}$) e a ***sequência de regras*** de mapeamento criadas por nós (i.e., o ***programa***).

Paradigma do ML

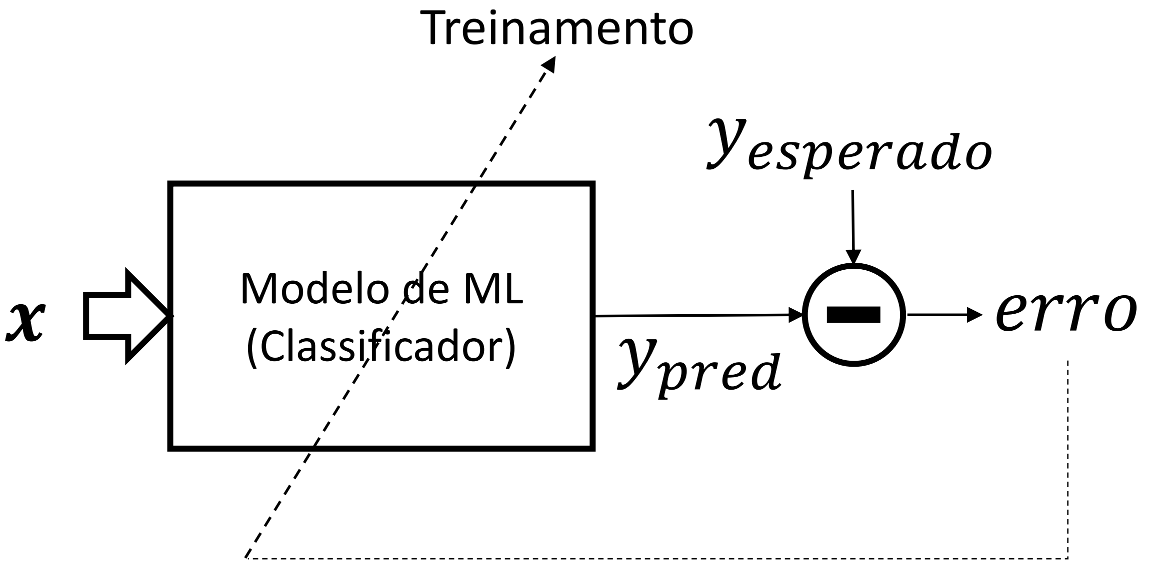


$\approx$

def *predict*($\boldsymbol{x}$):
    if($g_1(\boldsymbol{x})$ >= 0):
        y_pred[i] = 0
    # Caso quando $g_1(\boldsymbol{x})$ < 0
    else:
        if($g_2(\boldsymbol{x})$ < 0):
            y_pred[i] = 1
        # caso quando # $g_2(\boldsymbol{x})$ >= 0
        else:
            y_pred[i] = 2

- Fornecemos as ***entradas*** e as ***respostas esperadas***, i.e., atributos e classes, ao computador e deixamos que ele ***aprenda, através de treinamento***, um ***modelo*** que ***mapeie as entradas nas respostas esperadas da melhor forma possível***.

Objetivo do curso

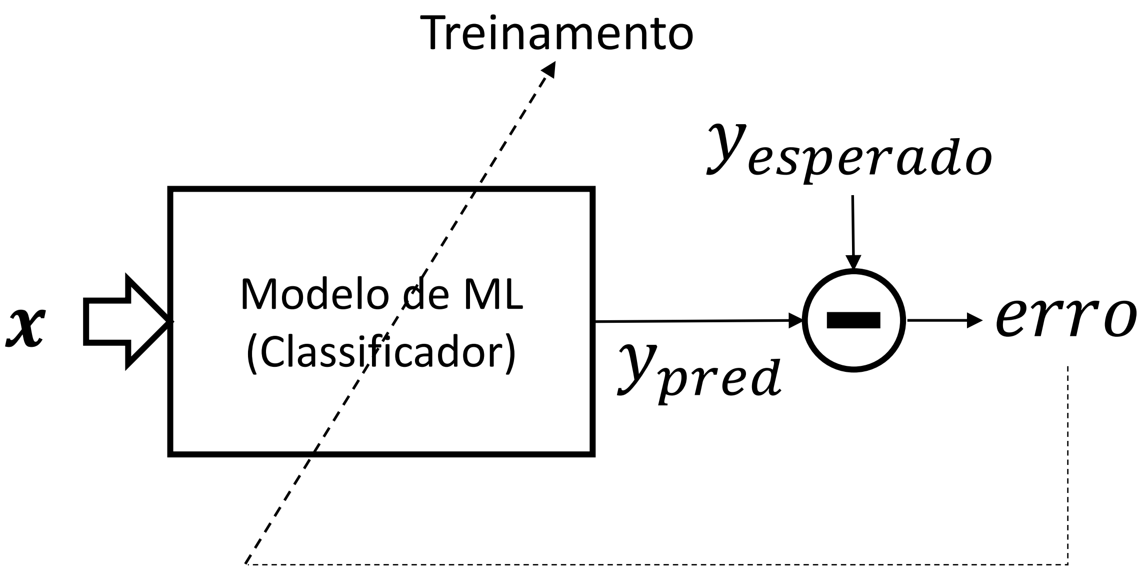


$\approx$

```
def predict(x):
    if(g_1(x) >= 0):
        y_pred[i] = 0
    # Caso quando g_1(x) < 0
    else:
        if(g_2(x) < 0):
            y_pred[i] = 1
        # caso quando # g_2(x) >= 0
        else:
            y_pred[i] = 2
```

Treinar modelos de ML que aprendam, ***sem serem explicitamente programados***, a classificar as entradas, $\boldsymbol{x}$, em suas respectivas classes.

Objetivo do curso



$\approx$

def *predict*($\boldsymbol{x}$):
    if($g_1(\boldsymbol{x})$ >= 0):
        y_pred[i] = 0
    # Caso quando $g_1(\boldsymbol{x}) < 0$
    else:
        if($g_2(\boldsymbol{x})$ < 0):
            y_pred[i] = 1
        # caso quando # $g_2(\boldsymbol{x})$ >= 0
        else:
            y_pred[i] = 2

Ou seja, o objetivo é ***encontrar*** ***funções discriminantes*** (i.e., equação e seus respectivos pesos) que ***minimizem*** o ***erro de classificação***.