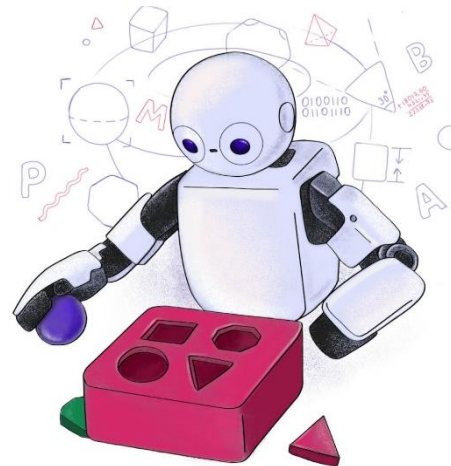


# TP557 - Tópicos avançados em IoT e Machine Learning:

## *Medindo a precisão de um modelo de ML*



***Inatel***

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

# O que vamos ver?

- Neste tópico vamos ver como *medir o desempenho* de um *modelo de aprendizado de máquina* ao longo do seu processo de aprendizagem.
- Para isso, como já discutido brevemente antes, usaremos uma função chamada de *função de erro* ou de *perda*.
- Idealmente, o *processo de treinamento* tem como *objetivo minimizar o erro* e, conseqüentemente, *aumentar a precisão do modelo*.
- Além disso, veremos em breve *diferentes estratégias para minimizar o erro*.
- Porém, primeiro, vamos aprender como calcular o erro.

# Mapeando $x$ em $y$

$$x = \{-1, 0, 1, 2, 3, 4\}$$

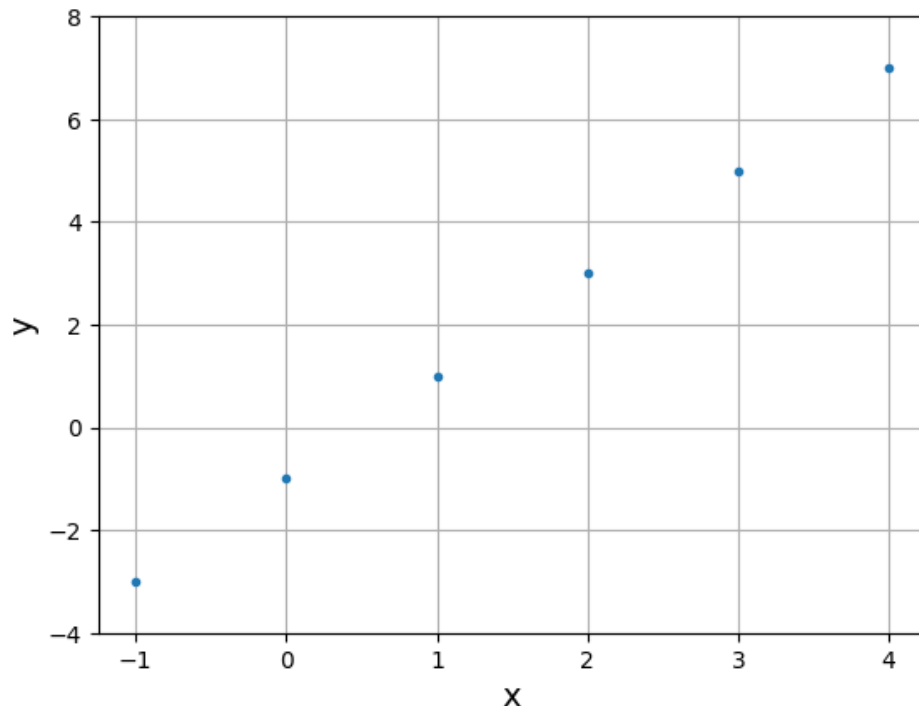
$$y = \{-3, -1, 1, 3, 5, 7\}$$

- Considerem esses dois conjuntos de números.
- Qual a relação entre os dois conjuntos?
  - Ou seja, qual é a **função que mapeia** os valores de  $x$  em  $y$ ?
- Nós sabemos que  $y$  é uma função de  $x$ , nós só não sabemos que função é essa.
- Que tal plotarmos esses pontos?

# Mapeando $x$ em $y$ : Função hipótese

$$x = \{-1, 0, 1, 2, 3, 4\}$$

$$y = \{-3, -1, 1, 3, 5, 7\}$$



- Ao plotarmos os pontos, percebemos que existe uma **relação linear** entre eles.
- Podemos criar a **hipótese** de que uma **reta** explica esse mapeamento.
- Portanto, usamos a função de uma reta para definir o mapeamento:
$$y = a_0 + a_1x.$$
- Agora precisamos encontrar os valores dos parâmetros (ou pesos)  $a_0$  e  $a_1$ .

# Suposição e Predição

$$\hat{y}[n] = h(x[n]) = -1 + 3x[n]$$

$$\mathbf{x} = \{-1, 0, 1, 2, 3, 4\}$$

Predições feitas pela função hipótese atual:

$$\hat{\mathbf{y}} = \{-4, -1, 2, 5, 8, 11\}$$



- Vamos começar **atribuindo** alguns **valores aleatórios para os pesos** ( $a_0 = -1$  e  $a_1 = 3$ ), ou seja, vamos fazer uma **suposição** (dar um **palpite**) sobre os valores.
- Como medimos se essa função hipótese é boa ou ruim?
  - Usamos os valores de  $x[n]$ ,  $\forall n$  para obter o mapeamento (i.e., predição) feito pela função e comparamos com os valores de saída esperados,  $\mathbf{y}$ .

# O quão boa é a função hipótese?

$$\hat{y}[n] = h(x[n]) = -1 + 3x[n]$$

$$\mathbf{x} = \{-1, 0, 1, 2, 3, 4\}$$

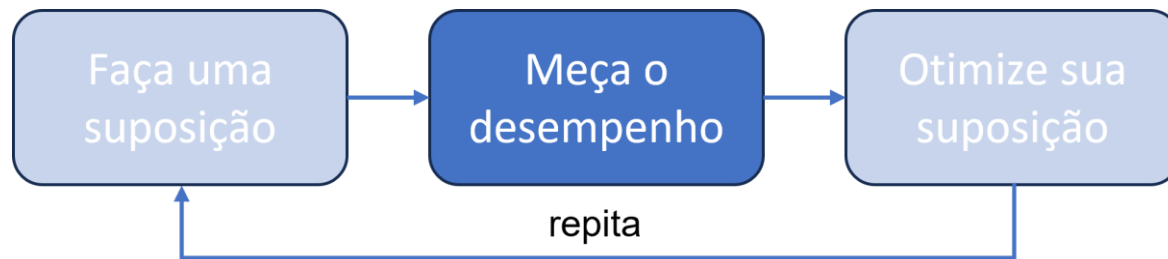
Predições feita pela função hipótese atual:

$$\hat{\mathbf{y}} = \{-4, -1, 2, 5, 8, 11\}$$

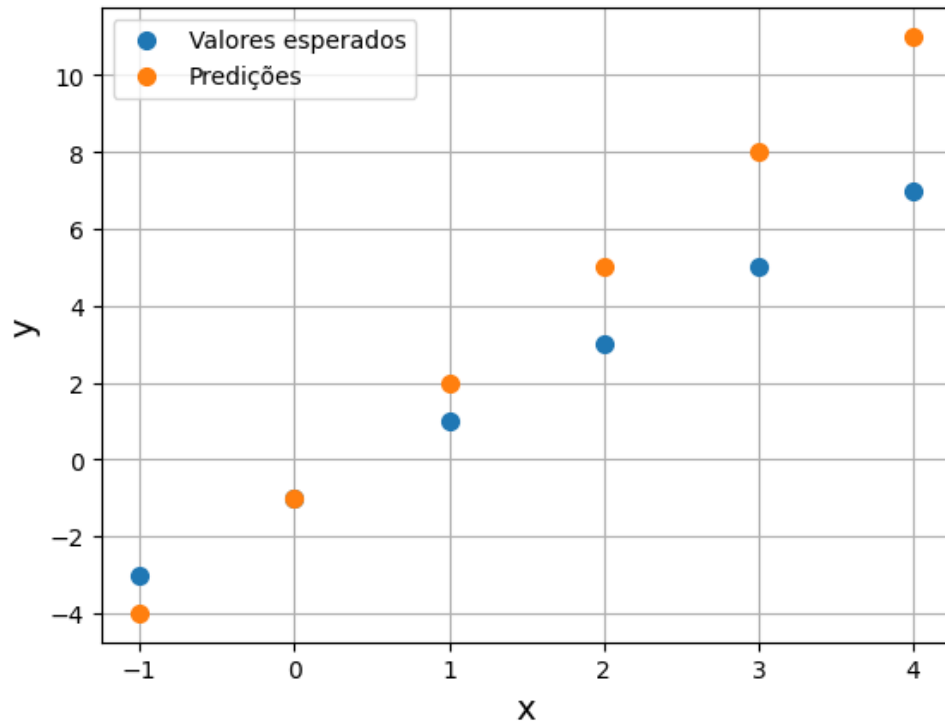
Valores de saída esperados:

$$\mathbf{y} = \{-3, -1, 1, 3, 5, 7\}$$

- Vemos que os **valores** preditos e esperados **não são os mesmos**.
- Os três primeiros valores até são próximos, mas os últimos três já estão mais distantes.
- Existe uma maneira de **formalizarmos** um **cálculo do quão bom ou ruim** essa função hipótese e suas predições são?

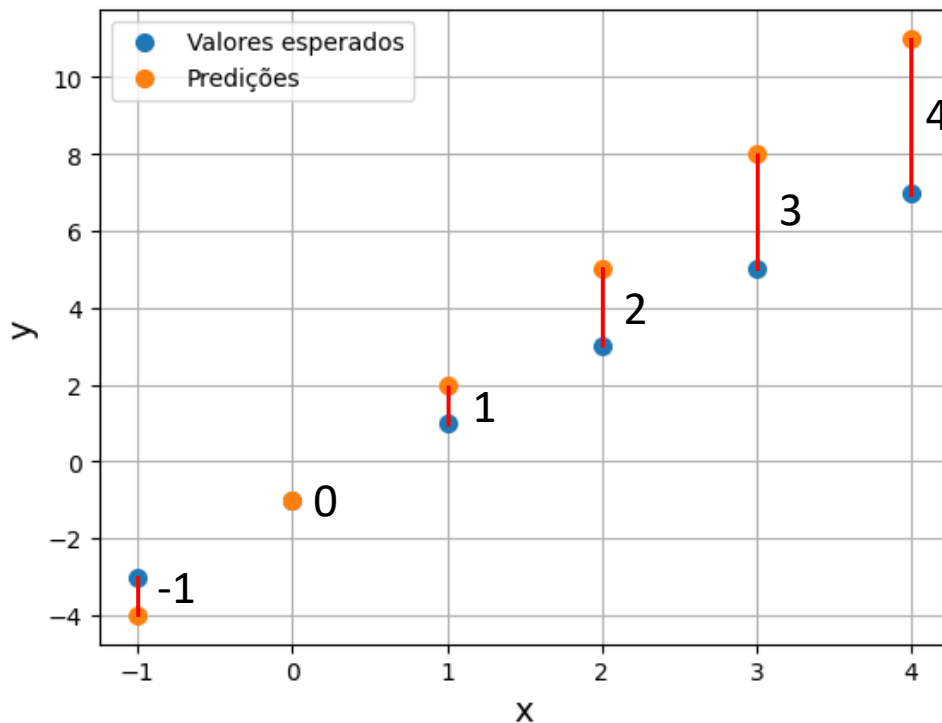


# Vamos medir as distâncias!



- Talvez nós possamos definir uma métrica de desempenho plotando os valores esperados e preditos.
- Na figura ao lado, temos os pontos preditos e esperados para cada valor de  $x$ .

# Vamos medir as distâncias!

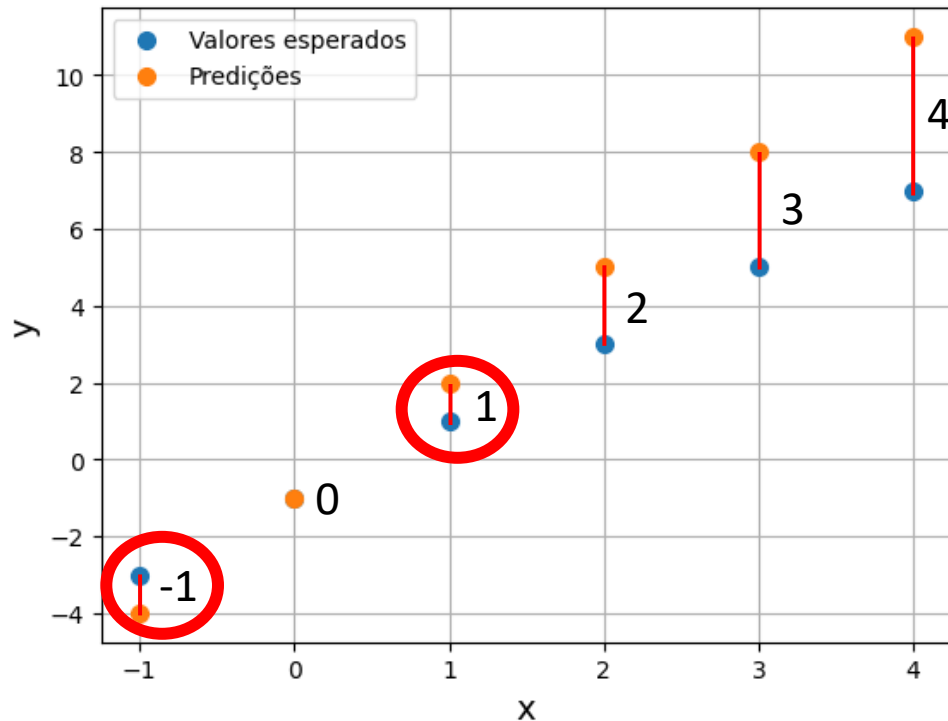


$$\text{diff}[n] = \hat{y}[n] - y[n]$$

- Podemos traçar uma reta entre cada ponto e, pelo comprimento dessas retas (ou *diferença entre os pontos*), *descobrir se a função hipótese é boa ou não*.
- O comprimento de cada reta é mostrado ao lado delas.
- Talvez nós possamos *calcular a média dos comprimentos* para obter o **erro médio** causado pela função hipótese atual.



# Temos um problema!

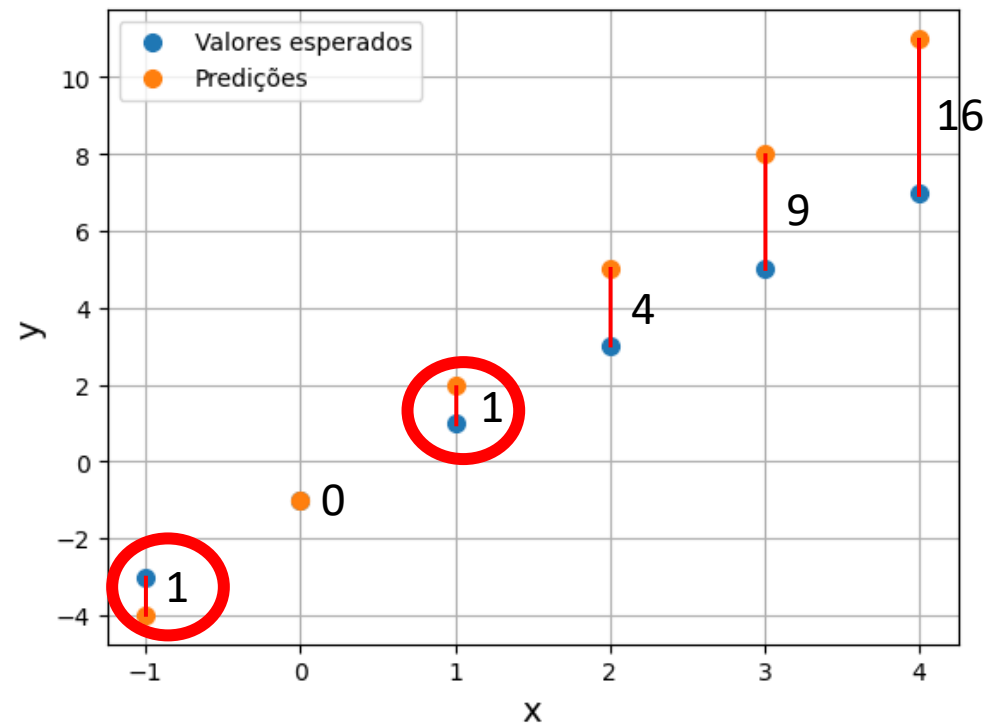


$$\text{erro médio} = \frac{1}{6} \sum_0^5 \text{diff}[n]$$

$$\text{erro médio} = \frac{-1 + 0 + 1 + 2 + 3 + 4}{6} = 1.5$$

- Ao somarmos as diferenças, **valores negativos podem reduzir ou até mesmo anular valores positivos**.
- Mesmo as predições para os dois pontos,  $x = -1$  e  $1$ , estando erradas, seus **erros se cancelariam**, afetando a medida de desempenho.
- Isso poderia sugerir que 3 de 6 predições estão corretas, o que sabemos não ser verdade.
- O que podemos fazer para resolver esse problema?

# Quadrado dos comprimentos



$$\text{erro quadrático médio (EQM)} = \frac{1}{6} \sum_{n=0}^5 \text{diff}[n]^2$$

$$\text{EQM} = \frac{1 + 0 + 1 + 4 + 9 + 16}{6} = 5.17$$

- Podemos ***eleva*** ***ao quadrado todos*** ***os comprimentos***, fazendo com que todos eles sejam positivos e não se cancelem mais.
- Isso não afeta nossa métrica, pois aplicamos a todos os comprimentos.
- Usando essa métrica de ***erro médio***, vemos que ***nossa função hipótese não é boa***, pois o ***valor ainda está longe de zero***, que é o menor valor possível e nosso objetivo final.
- O que devemos fazer?

# Otimizando a suposição

$$\hat{y}[n] = h(x[n]) = -2 + 2x[n]$$

$$\mathbf{x} = \{-1, 0, 1, 2, 3, 4\}$$

Predições feitas pela função hipótese atual:

$$\hat{\mathbf{y}} = \{-4, -2, 0, 2, 4, 6\}$$

Valores de saída esperados:

$$\mathbf{y} = \{-3, -1, 1, 3, 5, 7\}$$

Diferença entre predições e valores esperados:

$$\mathbf{diff}^2 = \{1, 1, 1, 1, 1, 1\}$$

- Vamos supor outros valores para  $a_0$  e  $a_1$  e fazer predições com esta nova função hipótese.
- O novo palpite para os valores dos pesos será  $a_0 = -2$  e  $a_1 = 2$ .

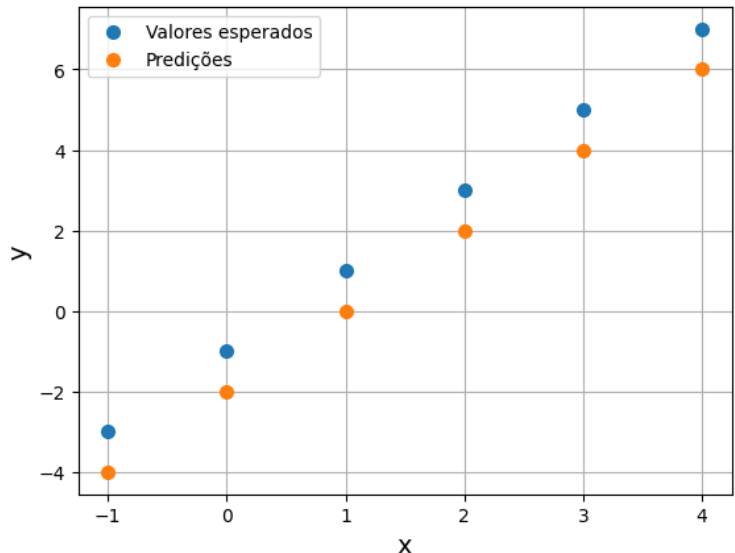


# O quão boa é a nova função hipótese?

$$\hat{y}[n] = h(x[n]) = -2 + 2x[n]$$

$$\mathbf{x} = \{-1, 0, 1, 2, 3, 4\}$$

$$\text{EQM} = \frac{1 + 1 + 1 + 1 + 1 + 1}{6} = 1.0$$



- Já temos uma **função melhor**, pois o erro caiu de 5.17 para 1.0.
- Estamos indo na direção correta!
- Comparando as predições com os valores esperados, notamos que as **retas são paralelas**, sendo a única diferença um **deslocamento**.
- O descolamento pode ser alterado através de  $a_0$  (coeficiente linear).

# Nova suposição

$$\hat{y}[n] = h(x[n]) = -1 + 2x[n]$$

$$\mathbf{x} = \{-1, 0, 1, 2, 3, 4\}$$

Predições feitas pela função hipótese atual:

$$\hat{\mathbf{y}} = \{-3, -1, 1, 3, 5, 7\}$$

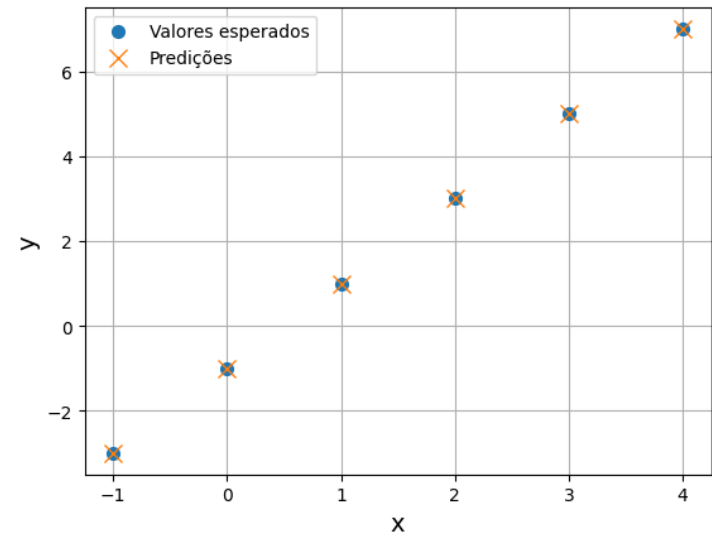
Valores de saída esperados:

$$\mathbf{y} = \{-3, -1, 1, 3, 5, 7\}$$

Diferença entre predições e valores esperados:

$$\mathbf{diff}^2 = \{0, 0, 0, 0, 0, 0\}$$

$$EQM = \frac{0 + 0 + 0 + 0 + 0 + 0}{6} = 0.0$$



- Vamos **aumentar o valor de  $a_0$**  para  $-1$ , fazer novas predições e calcular o erro.
- Bingo! Nossa nova suposição mapeia perfeitamente  $x$  em  $y$ , resultando em um EQM igual a 0.



# O processo



- O **processo iterativo de otimização** (ou atualização) dos pesos será sempre esse:
  - **Palpite** (i.e., pesos),
  - **Medida da qualidade** do palpite,
  - **Otimização do palpite** (i.e., atualização dos peso) baseada na informação do erro.
- As **iterações se repetem** até que algum **critério de parada** seja atingido: **número máximo de iterações**, **erro cair abaixo de um limiar**, etc.
- Esse é o processo por trás do algoritmo de otimização que veremos adiante.

# Outras medidas de erro

$$\text{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2}$$

$$\text{MAE} = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{y}[n] - y[n]|$$

$$\text{CE} = -\frac{1}{N} \sum_{n=0}^{N-1} \sum_{i=0}^{Q-1} y_i[n] \log_e(\hat{y}_i[n])$$

Usadas em  
problemas de  
aproximação  
de curvas

Usada em problemas de classificação

- Além do EQM (*Mean Squared Error* – MSE), existem diversas outras métricas de erro que podemos usar, como por exemplo:
  - Raíz do Erro Quadrático Médio (*Root Mean Squared Error* – RMSE)
  - Erro Absoluto Médio (*Mean Absolute Error* – MAE);
  - Entropia Cruzada (*Cross-Entropy*);
  - Etc.

# Mean Squared Error – MSE

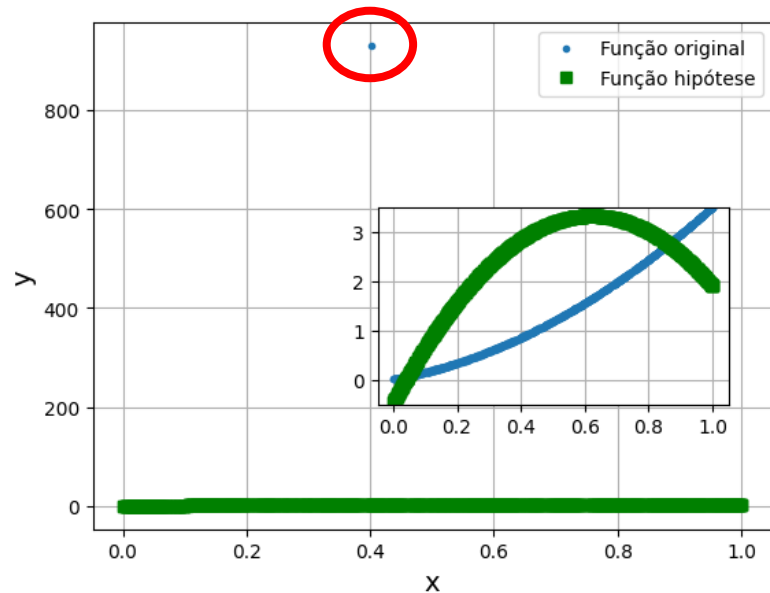
$$\text{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2$$

- É a função de erro *mais usada em problemas de regressão*.
- Calcula a média do *quadrado da diferença* entre os valores preditos e esperados.
- Assim, ele *penaliza mais (i.e., amplifica) erros maiores* (devido ao quadrado), o que pode ser útil se desejarmos que o modelo seja *mais sensível a esses erros*.



# Mean Squared Error – MSE

$$\text{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2$$



Função original:  $y = 1 + x + 2x^2$

- Porém, como a diferença é elevada ao quadrado, ele **pode amplificar a influência de outliers**.
  - Outliers podem **distorcer ou deslocar a função hipótese**, fazendo com que ela se distancie de um mapeamento que represente o comportamento dos dados “normais”.
- Por usar o quadrado da diferença, **a métrica não fica na mesma escala dos dados originais**.
  - Por exemplo, se estivermos predizendo o preço de casas em dólares, os erros estarão em dólares<sup>2</sup>.

# Root Mean Squared Error – RMSE

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2}$$

- Calcula a raiz quadrada do MSE, **retornando a métrica à mesma escala dos dados originais.**
- Isso é útil para interpretar o erro na mesma unidade do rótulo, o que pode facilitar a compreensão do impacto do erro.
- Assim como o MSE, **também amplifica erros maiores.**

# Mean Absolute Error – MAE

$$\text{MAE} = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{y}[n] - y[n]|$$

- Calcula a *média das diferenças absolutas* entre as previsões do modelo e os valores esperados.
  - Ou seja, *penaliza os erros de maneira uniforme*.
- Ao contrário do MSE e RMSE, não eleva os erros ao quadrado, o que o torna *menos sensível a outliers*.

# Mean Absolute Error – MAE

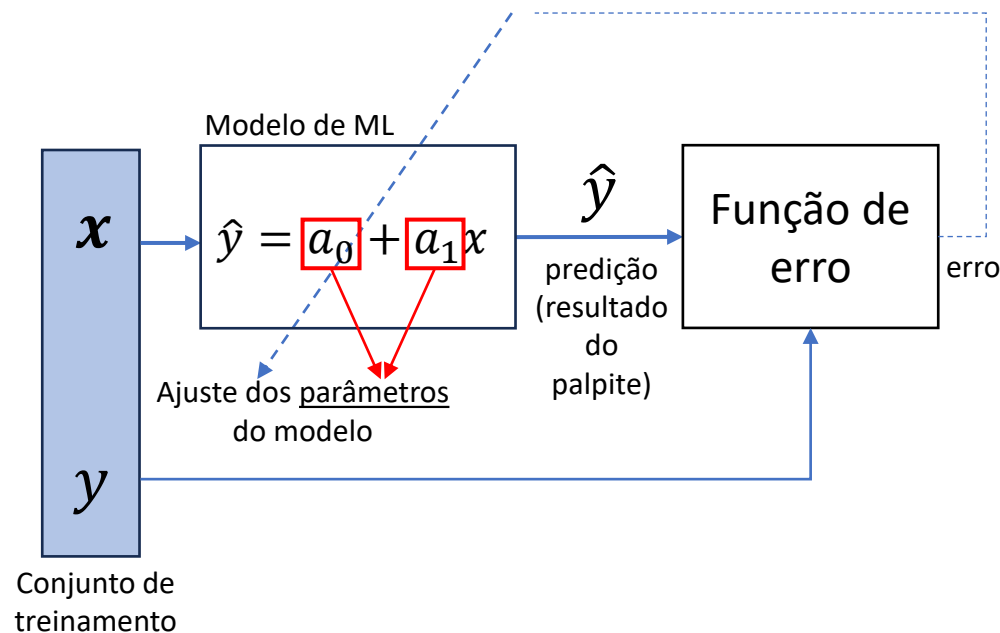
$$\text{MAE} = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{y}[n] - y[n]|$$

- *Representa os erros na mesma escala dos dados originais*, facilitando a interpretação.
- Boa opção quando desejamos um *erro mais uniforme em todas as predições* e desejamos *minimizar a influência de outliers*.

# Qual devo usar?

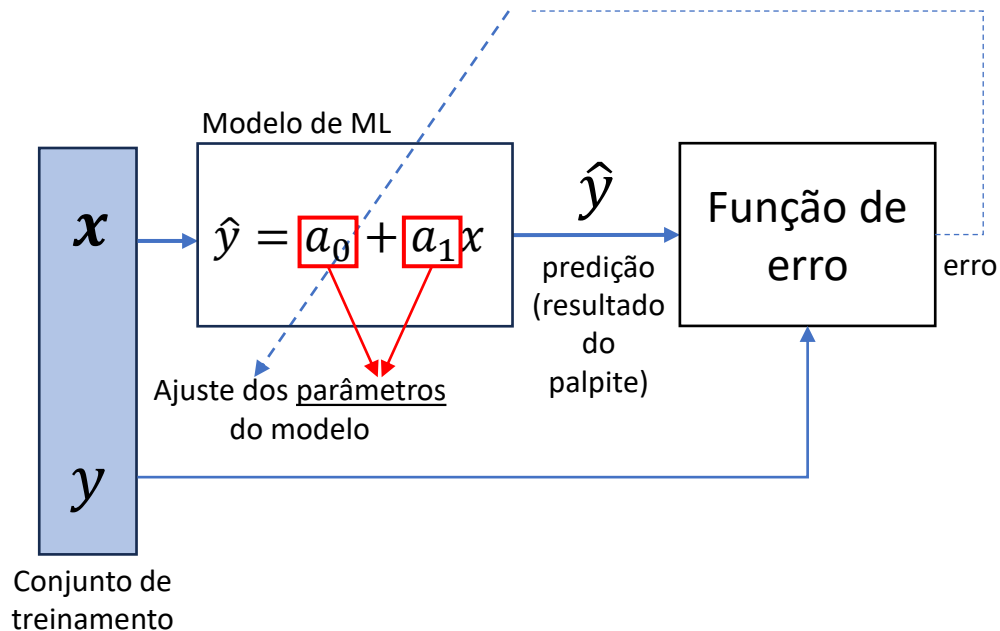
O recomendável é ***experimental diferentes funções de erro*** durante a ***fase de desenvolvimento*** para determinar qual delas se ***alinha*** melhor com os ***objetivos e as características*** específicas do seu ***problema de regressão***.

# Regressão ou aproximação de funções




- O problema que estamos vendo neste tópico é conhecido como ***regressão*** ou ***aproximação de funções***.
- Nele, a função com formato de reta, que definimos como ***função hipótese***, é o ***modelo de ML*** e o ***objetivo da regressão é encontrar os parâmetros (ou pesos) que minimizam a função de erro***.

# Regressão ou aproximação de funções



- **OBS.:** No exemplo que vimos nesse tópico, conseguimos visualizar os dados e, facilmente, identificar a equação (ou formato) da função hipótese, mas em outros casos, além dos pesos, temos que descobrir o formato da função.
- Veremos adiante formas de definir o melhor modelo para uma aproximação.

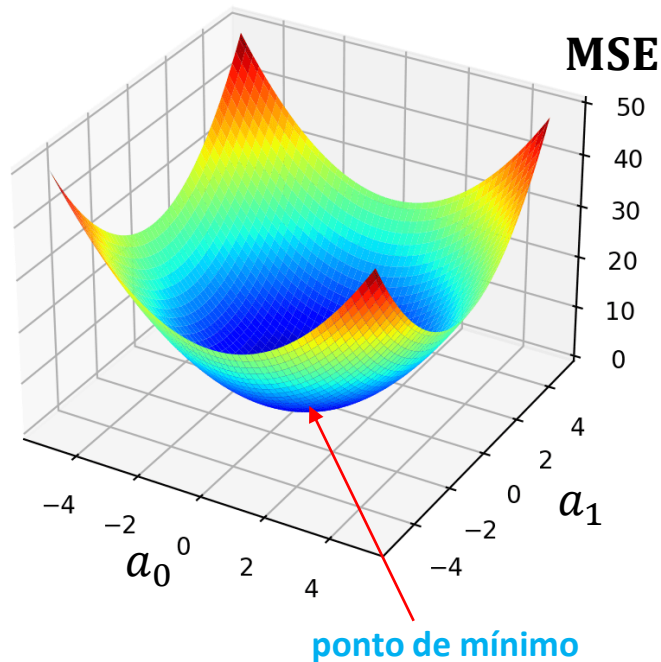
# Ponto de mínimo ou solução ótima

$$\begin{aligned}\text{MSE} &= \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2 \\ &= \frac{1}{N} \sum_{n=0}^{N-1} (\boxed{a_0} + \boxed{a_1}x[n] - y[n])^2\end{aligned}$$


- Dado que o conjunto de treinamento é conhecido e fixo, notamos que a ***função de erro é função dos pesos*** do modelo.
- Assim, o ***objetivo da regressão é encontrar o conjunto de pesos que minimiza a função de erro.***
- Esse é um ***problema de minimização.***
  - ***Qual é o conjunto de pesos que resulta no menor erro possível?***



# Ponto de mínimo ou solução ótima

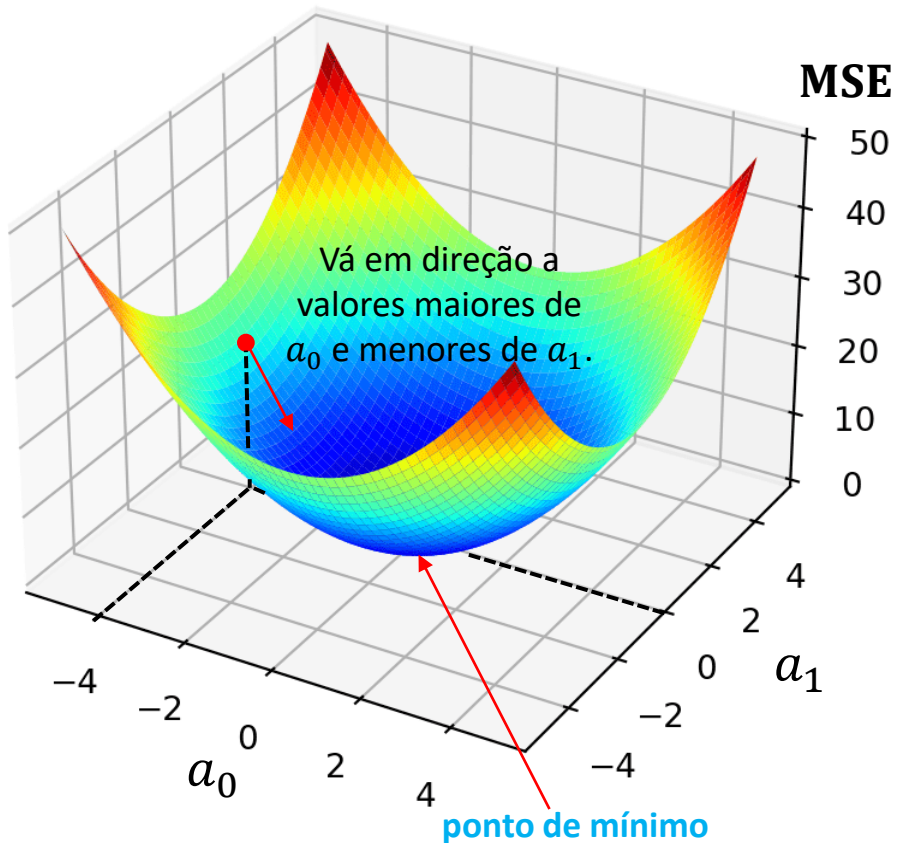


$$\begin{aligned} \text{MSE} &= \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}[n] - y[n])^2 \\ &= \frac{1}{N} \sum_{n=0}^{N-1} (\boxed{a_0} + \boxed{a_1} x[n] - y[n])^2 \end{aligned}$$

A red arrow points from the question mark in the second equation to the 'ponto de mínimo' label in the figure above.

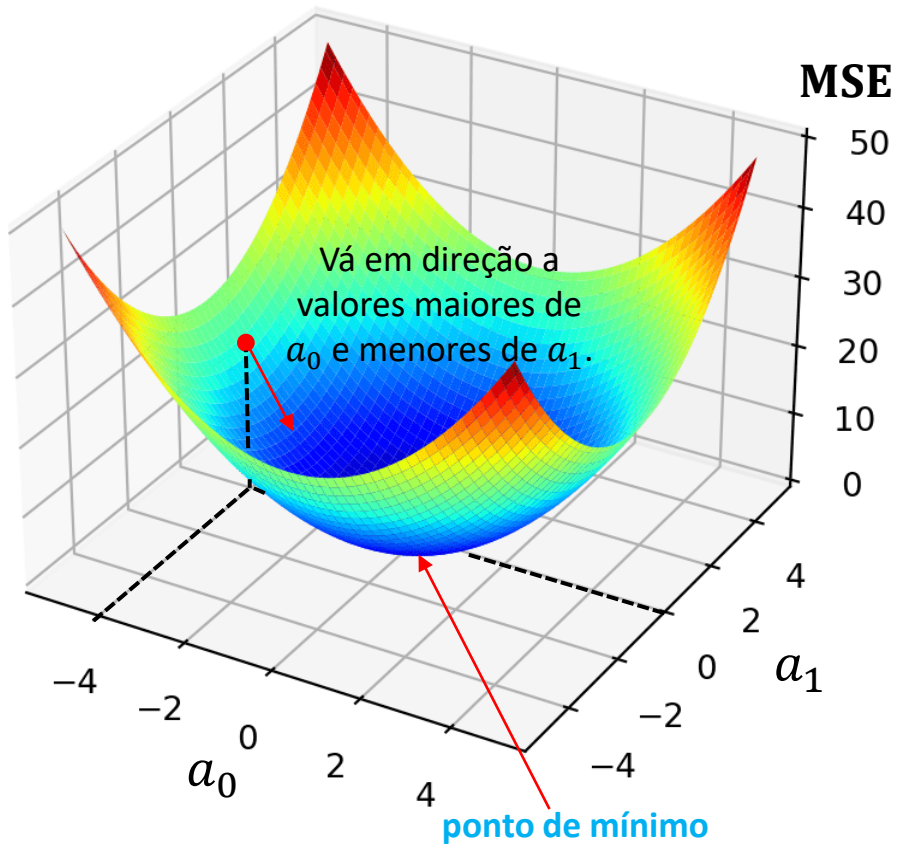
- Se *plotarmos o erro em função da variação dos pesos*, nós plotamos o que é conhecido como *superfície de erro*.
- Seu *ponto mais baixo* é chamado de *ponto de mínimo* e nos dá o *conjunto de pesos ótimo*, ou seja, os pesos que minimizam a função de erro.
- O algoritmo de otimização, que veremos em breve, *caminha* através da superfície sempre *indo em direção ao seu ponto mais baixo*.

# Otimização (treinamento) do modelo



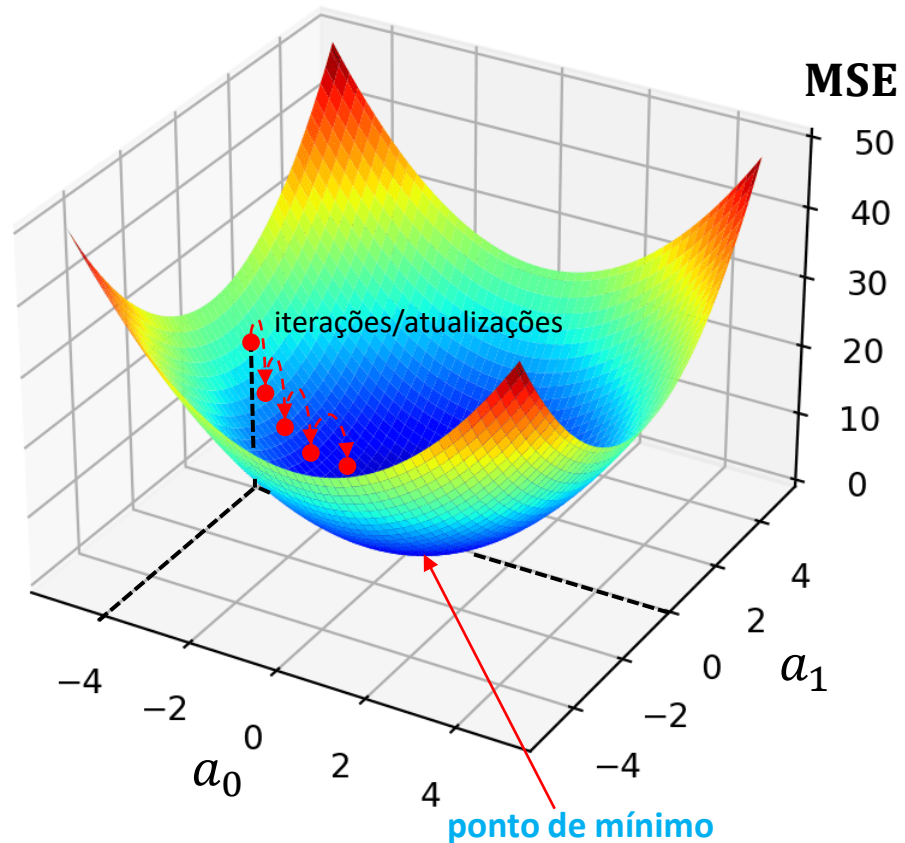
- Esse **processo de busca** que tem como **objetivo** encontrar o **ponto** que **minimiza o erro** é chamado de **otimização** e, no contexto de ML, é conhecido como **treinamento do modelo**.
  - Treinamento é o processo de **atualizar o modelo** para obter previsões melhores a cada **iteração**.

# Otimização (treinamento) do modelo



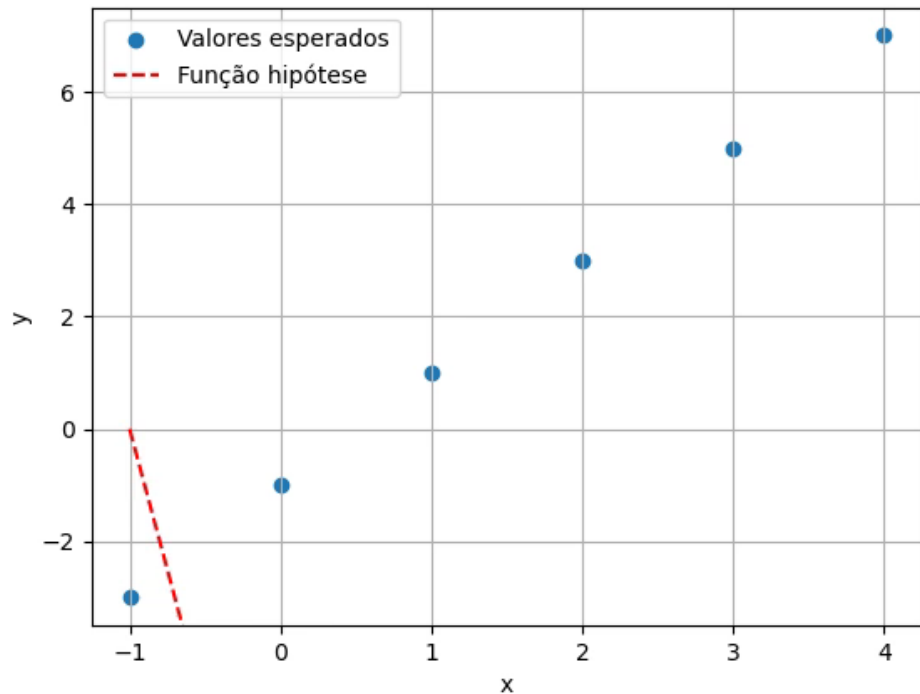
- O **treinamento**/atualização do modelo se **baseia** na **informação obtida a partir da função de erro**.
- Como veremos, essa informação **aponta para a direção do ponto de mínimo** (i.e., conjunto de pesos ótimo).

# Gradiente descendente



- Como veremos em breve, de posse dessa informação trazida pelo erro, o algoritmo de otimização, conhecido como *gradiente descendente (GD)*, consegue *caminhar iterativamente* através da *superfície de erro* até o *ponto de mínimo* (i.e., conjunto de pesos ótimo).

# Gradiente descendente



- A figura ao lado ilustra o processo iterativo de otimização realizado pelo GD.
  - A partir de um **ponto inicial** (i.e., uma **suposição aleatória** de pesos), o algoritmo **extrai a informação de direção do ponto de mínimo a partir da função de erro** e a usa para **atualizar o ponto atual** (i.e., otimizar a suposição atual).
  - Em geral, o **novo ponto** (i.e., conjunto de pesos atualizado) resulta em um **erro menor do que o anterior**.
  - Esse processo é repetido a partir do novo ponto até que algum critério de parada seja atingido.

# Próximos passos

- Nesse tópico nós vimos o que é o **erro** (ou perda) e definimos uma forma de medi-lo.
- Agora, o próximo passo envolverá a **definição de um processo que utilize a informação do erro para gerar o próximo palpite** (ou suposição) de forma que ele **seja melhor do que o atual**.
- Esse processo é chamado de **otimização** e será abordado em breve.
- Mas primeiro, vamos dar uma olhada em um **exemplo** que nos dará uma ideia **sobre como funciona esse processo de otimização**.



[Explorando a função de erro](#)

# Atividades

- Quiz: “***TP557 - Medindo a precisão de um modelo de ML***”.
- Exercício: [Encontre os pesos da função hipótese](#).

Perguntas?



Obrigado!

