Regressão e redes neurais rasas

Juvenal J. Duarte

Regressão linear:
Revisão

Juvenal J. Duarte

Regressão linear multivariada

Quantidade de linhas / registros / amostras: m

Quantidade de colunas / atributos / characteristicas: n

Conjunto de entradas:

$$\vec{X}_{m,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

Conjunto de pesos:

$$\vec{B}_{n,1} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_n \end{bmatrix}$$

Bias: α

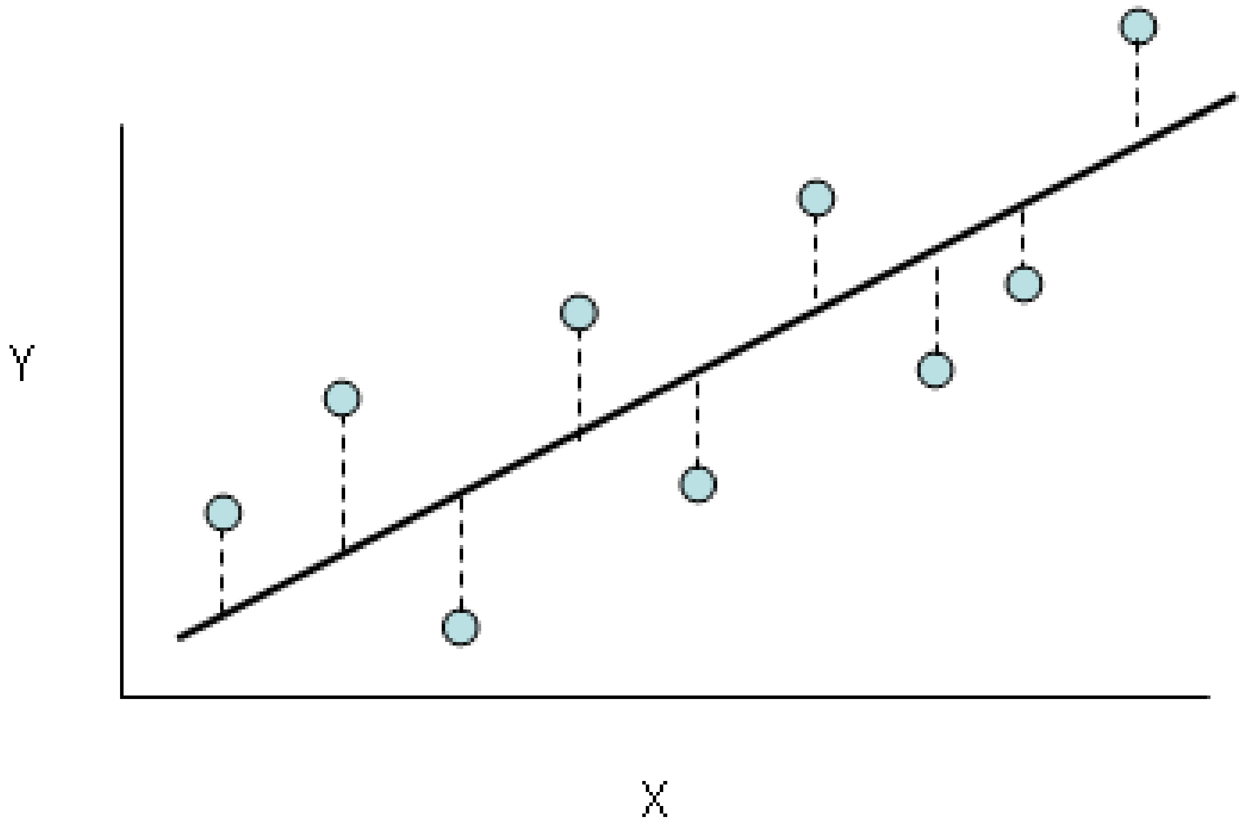
(1) Predição (formula vetorial):

$$\hat{y}_{m,1} = \vec{X}_{m,n}\,\vec{B}_{n,1} + \alpha$$

Regressão linear multivariada

Problema mapeado como regressão:

Entrada: Vetor de números reais.
Saída: Número real.



Regressão linear multivariada

1. *Calcular as predições para cada um dos exemplos no conjunto de teste, pela função $\hat{y}$.*

2. *Calcular cada uma das perdas individuais pela função de perda:*
$$loss(y, \hat{y}) = y - \hat{y}$$

3. *Calcular o custo do modelo dados os pesos escolhidos:*
$$J(B, \alpha) = \frac{1}{m} \sum_{i=1}^{m} loss(y, \hat{y})^2$$

4. *Calcular os gradientes em relação aos pesos $\frac{\partial J}{\partial B}$ e ao bias $\frac{\partial J}{\partial \alpha}$.*

5. *Atualizas os pesos e o bias de acordo com o gradiente e a taxa de aprendizado.*

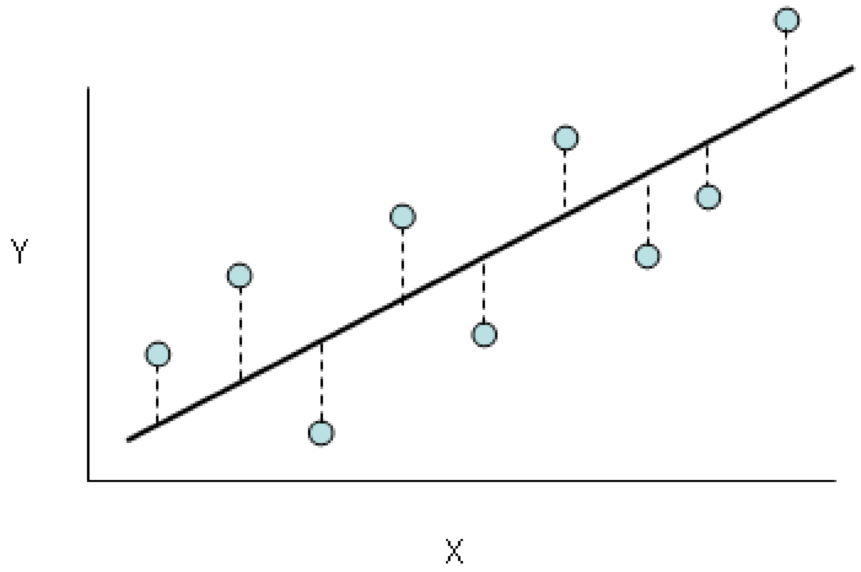
6. *Repetir o processo iterativamente até que as iterações se excedam.*

Regressão Logística Binária

Juvenal J. Duarte

Regressão / Classificação:

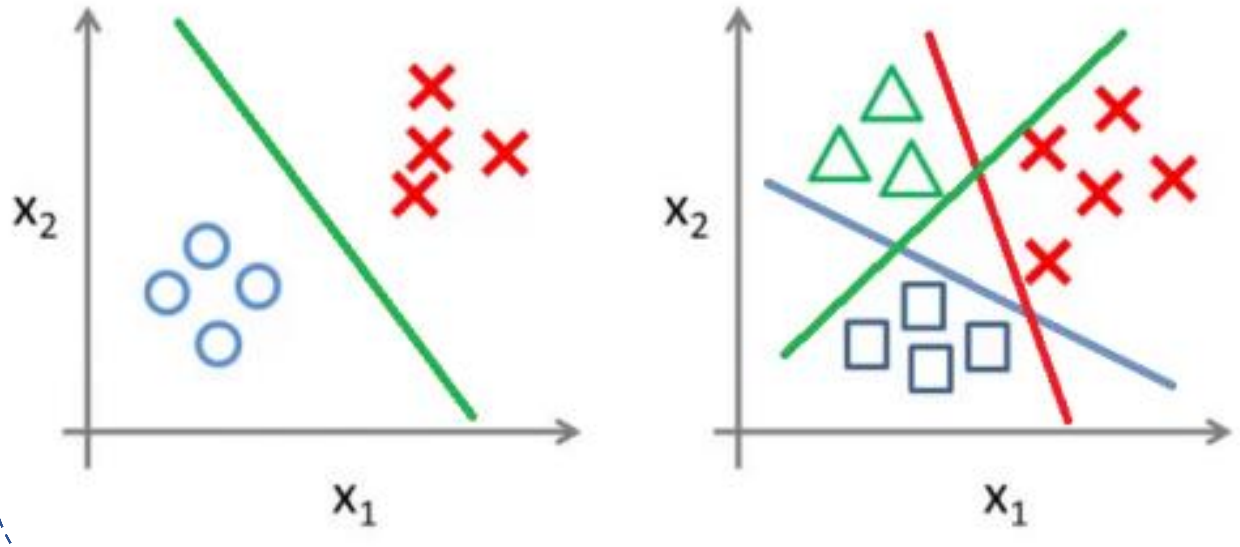
Regressor: $f: X \rightarrow y \mid y \in \mathbb{R}$



Funções de decisão: $f: X \rightarrow y \mid y \in \{c_1, c_2, \cdots, c_n\}$

Binária

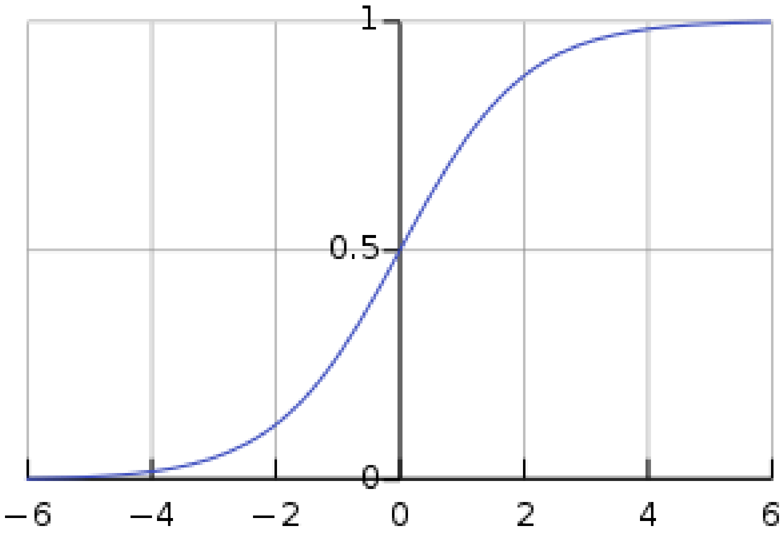
Multi-classe



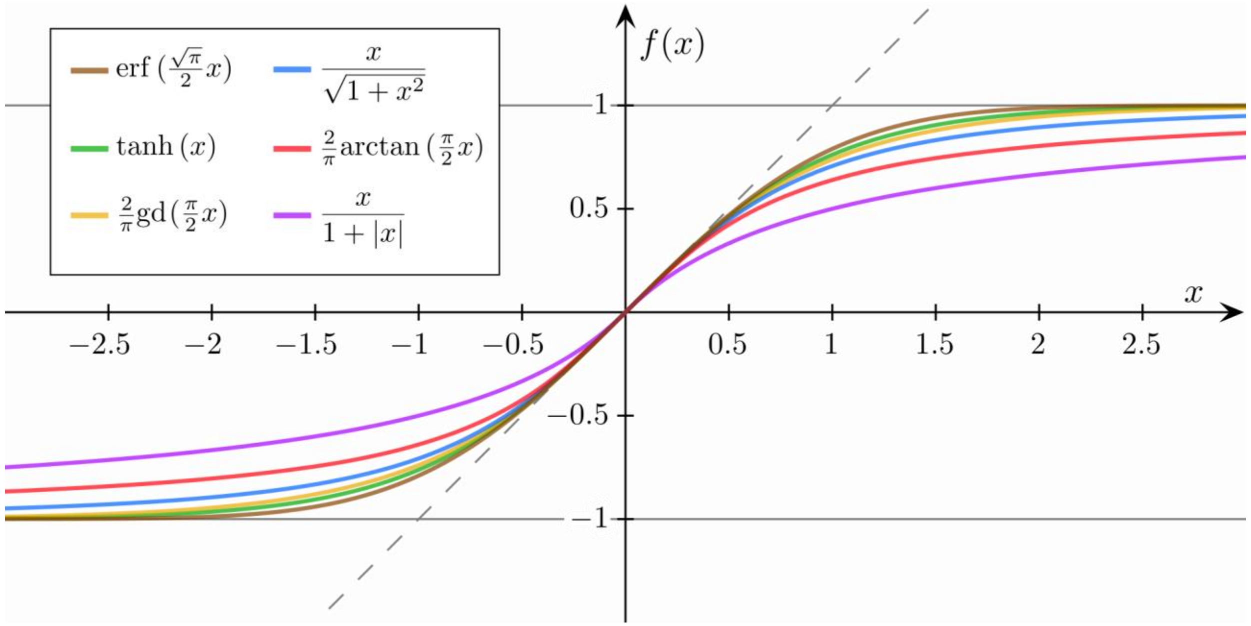
Função sigmoid

É uma função em formato de ‘S’, com domínio (entradas) em números reais e imagem (saídas) limitadas a um intervalo, normalmente $[0,1]$ ou $[-1,1]$.

FUNÇÃO LOGÍSTICA



OUTRAS FUNÇÕES SIGMOID

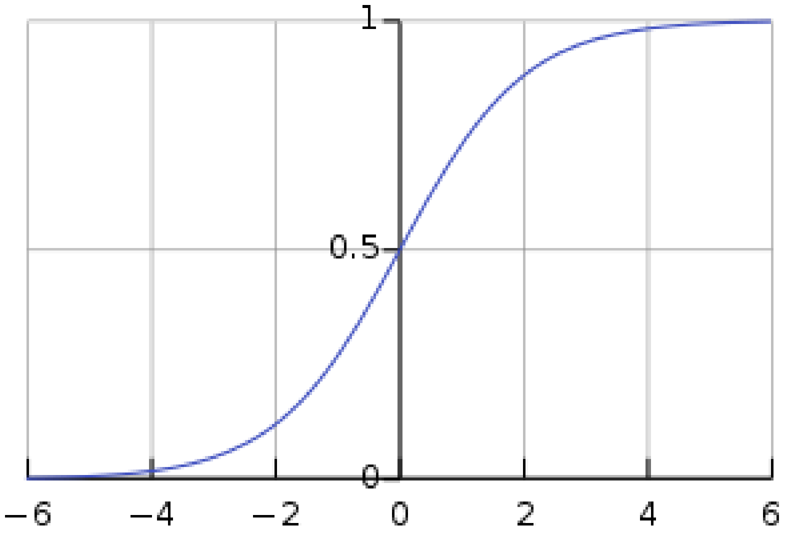


Função Logística

É uma função sigmoidal cujos valores de saída variam no intervalo $[0, L]$, sendo o valor mais comum $[0,1]$.

$$f(x) = \frac{L}{1 + e^{-kx}}$$

FUNÇÃO LOGÍSTICA



Onde:

L = valor máximo da função
e = número Euler, base do logarítmo natural
k = taxa de crescimento
x = entrada

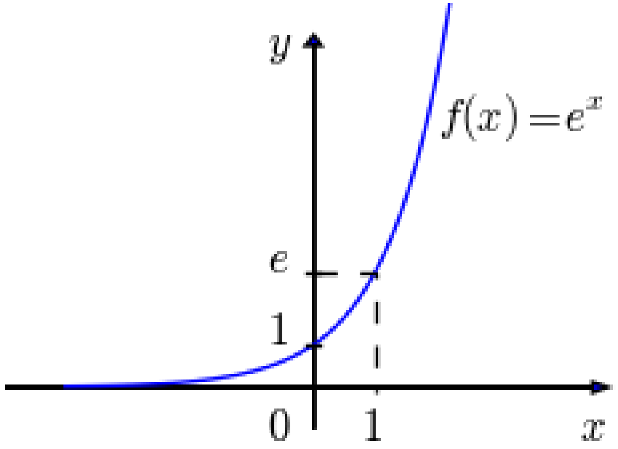
Função Logística

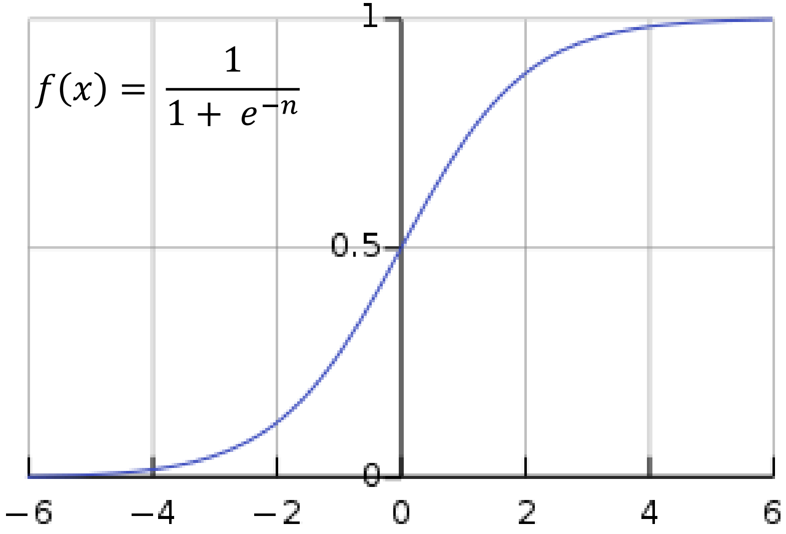
$$\lim_{n\to\infty} e^{-n} = ?$$

$$\lim_{n\to-\infty} e^{-n} = ?$$

$$\lim_{n\to\infty} \frac{1}{1+\ e^{-n}} = ?$$

$$\lim_{n\to-\infty} \frac{1}{1+\ e^{-n}} = ?$$





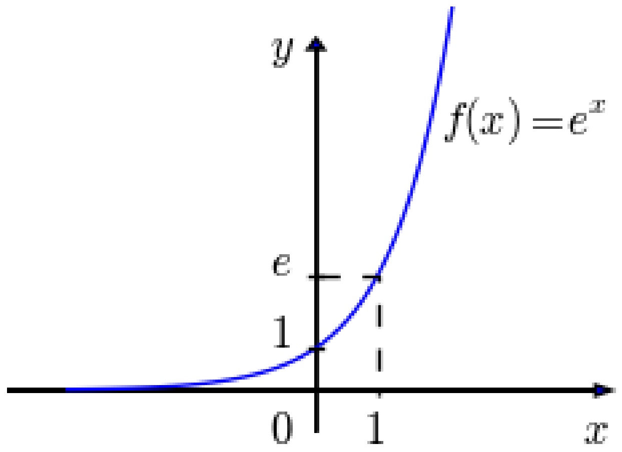
Função Logística

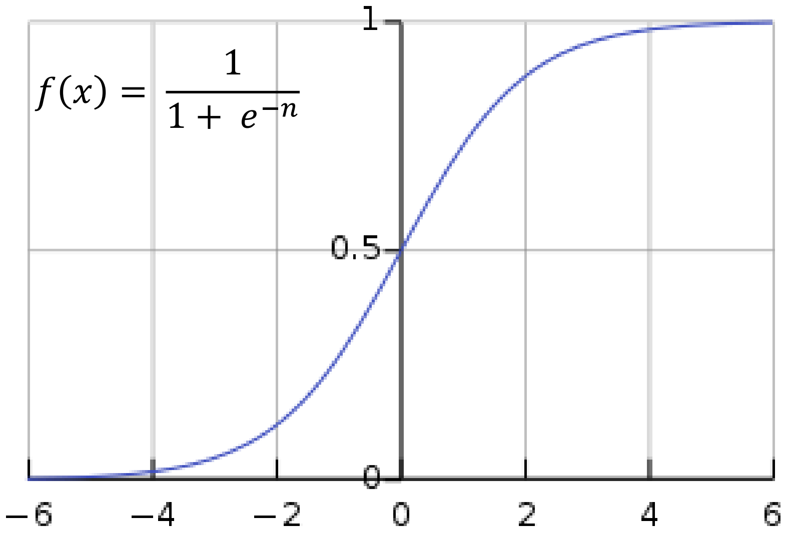
$$\lim_{n\to\infty} e^{-n} = 0$$

$$\lim_{n\to-\infty} e^{-n} = \infty$$

$$\lim_{n\to\infty} \frac{1}{1+ e^{-n}} =?$$

$$\lim_{n\to-\infty} \frac{1}{1+ e^{-n}} =?$$





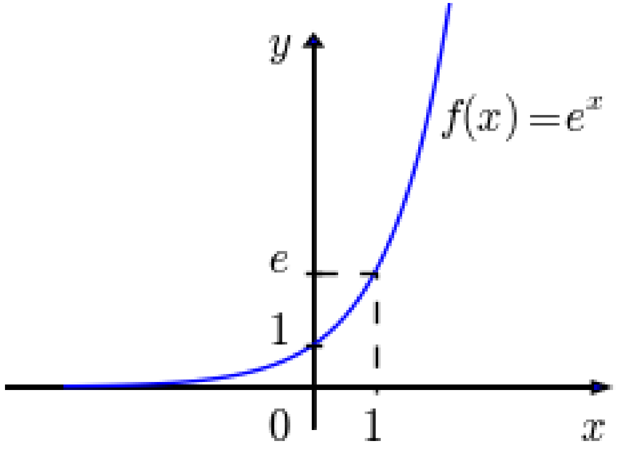
Função Logística

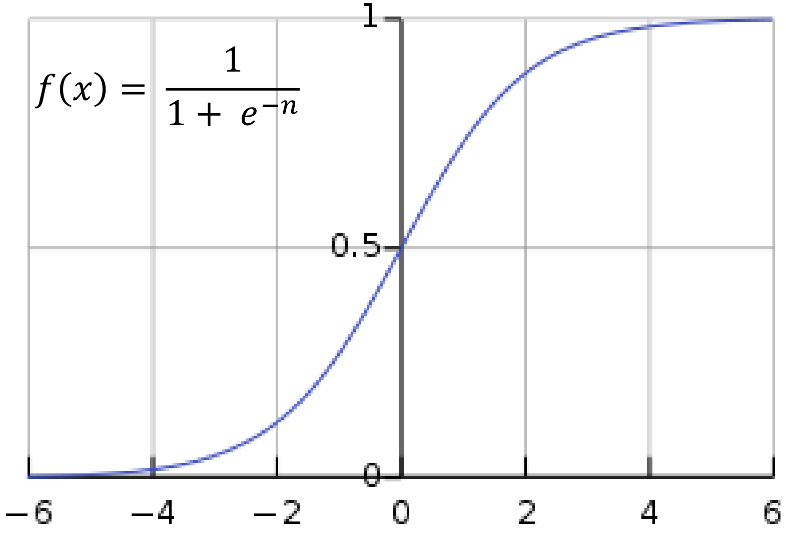
$$\lim_{n \to \infty} e^{-n} = 0$$

$$\lim_{n \to -\infty} e^{-n} = \infty$$

$$\lim_{n \to \infty} \frac{1}{1 + e^{-n}} = 1$$

$$\lim_{n \to -\infty} \frac{1}{1 + e^{-n}} = 0$$





Função Logística

$$\lim_{n\to\infty} e^{-n} = 0$$

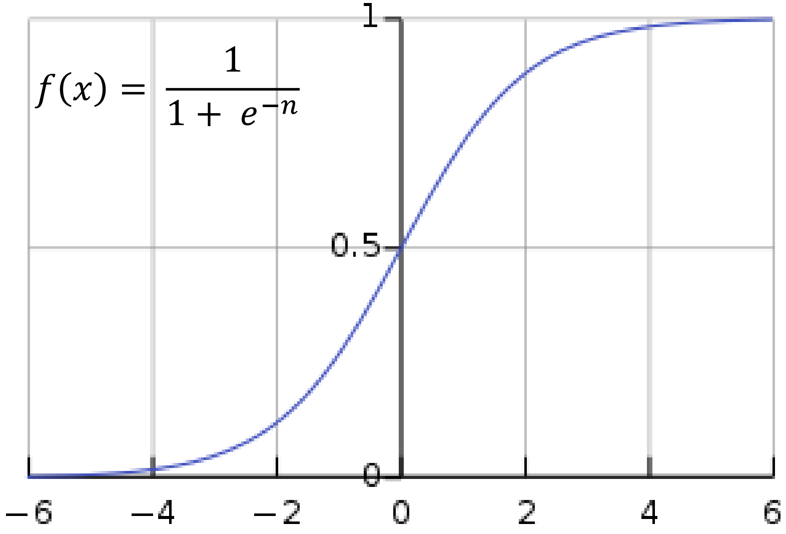
$$\lim_{n\to-\infty} e^{-n} = \infty$$

$$\lim_{n\to\infty} \frac{1}{1 + e^{-n}} = 1$$

$$\lim_{n\to-\infty} \frac{1}{1 + e^{-n}} = 0$$

Conclusão:

A função se aproxima de 0 pela esquerda, mas nunca chega. A função se aproxima de 1 pela direita, mas também nunca chega. Temos duas assíntotas horizontais.



Regressão logística multivariada

Quantidade de linhas / registros / amostras: m

Quantidade de colunas / atributos / characteristicas: n

Conjunto de entradas:

$$\vec{X}_{m,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

Conjunto de pesos:

$$\vec{B}_{n,1} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_n \end{bmatrix}$$

Bias: α

(1) Predição (formula vetorial):

$$\hat{y}_{m,1} = \frac{1}{1 + e^{-(\vec{X}_{m,n}\,\vec{B}_{n,1} + \alpha)}}$$

Regressão Logística: Função de custo

(2)

$$loss(i) = y_i - \hat{y}_i$$

(3) Erro Quadrático Médio

$$cost(\beta, \alpha) = J = \frac{1}{m}\sum_{i=1}^{m} loss(y_i, \hat{y}_i)^2$$

Regressão Logística: Função de custo

(2)

$$loss(i) = y_i - \hat{y}_i$$

(3) Erro Quadrático Médio

$$\cancel{cost(\beta, \alpha) = J = \frac{1}{m} \sum loss(y_i, \hat{y}_i)^2}$$

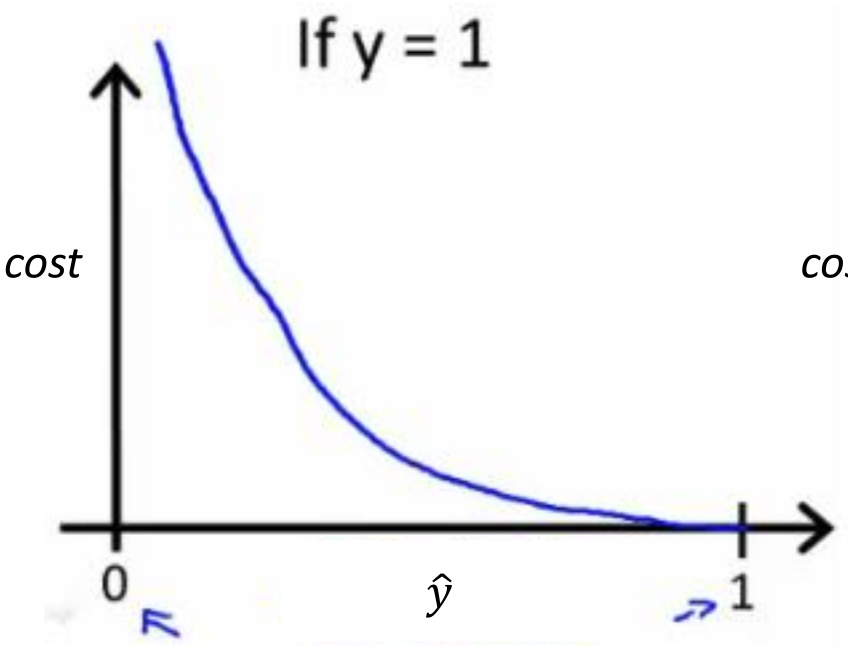
(3) Entropia Cruzada

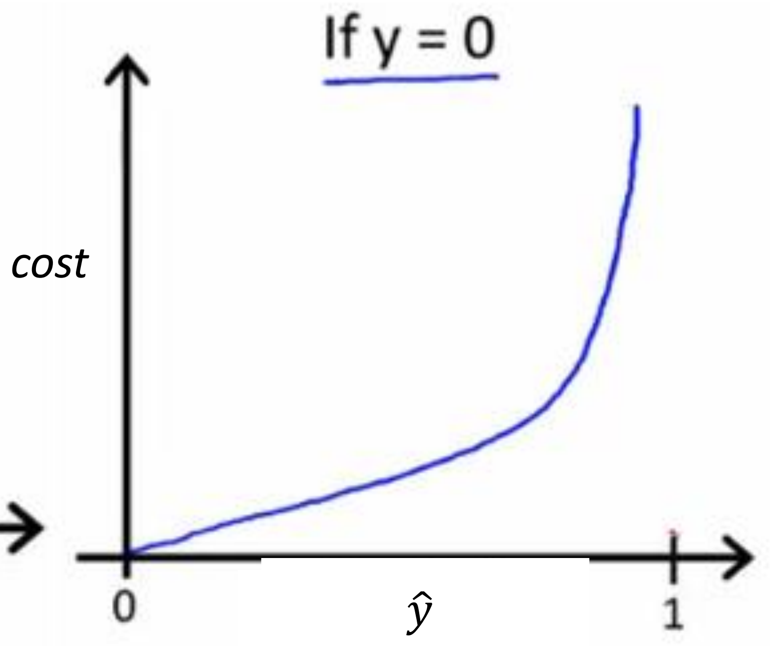
$$cost(\beta, \alpha) = J = \frac{1}{m} \sum_{i=1}^{m} -[y_i \log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

Regressão Logística: Função de custo

(3) Entropia Cruzada

$$cost(\beta, \alpha) = J = \frac{1}{m}\sum_{i=1}^{m} -[y_i \log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$





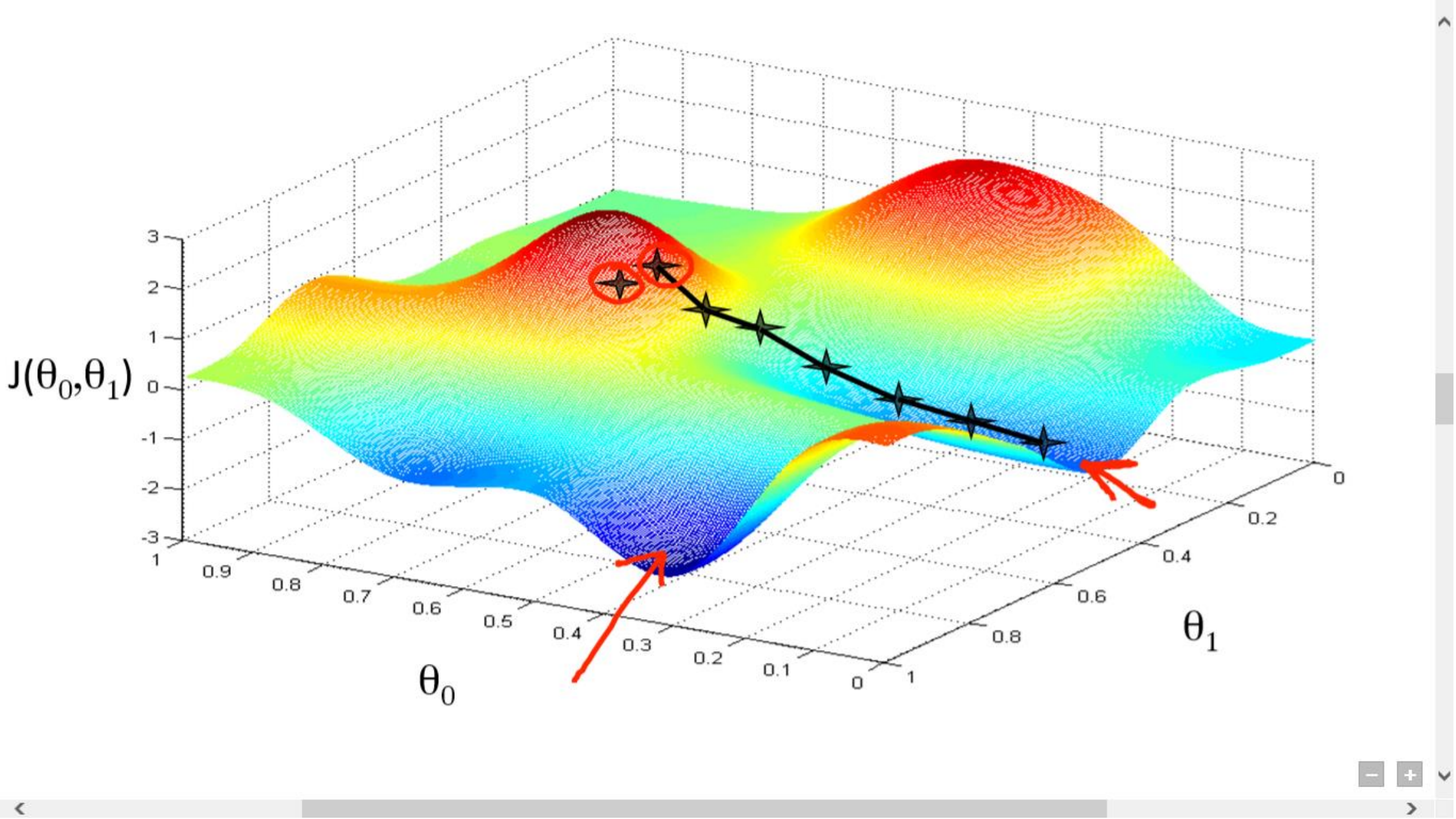
Regressão Logística: Função de custo

(3) Entropia Cruzada

$$cost(\beta,\alpha) = J = \frac{1}{m}\sum_{i=1}^{m} \underbrace{-[y_i \log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)]}$$

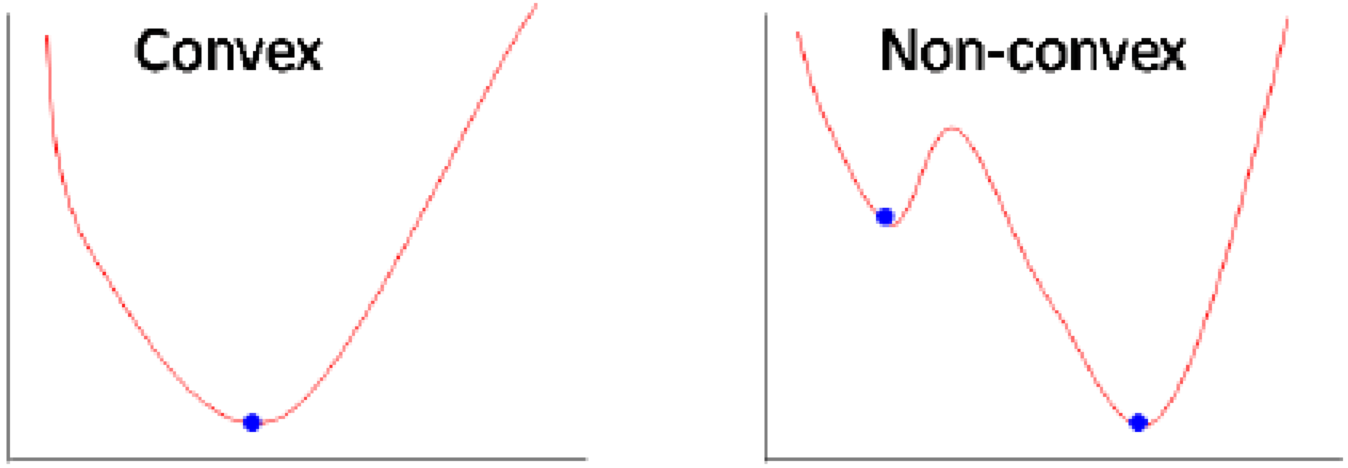
$$\begin{cases} -\log(\hat{y}). & ,se\ y = 1 \\ -\log(1-\hat{y}) & ,se\ y = 0 \end{cases}$$

Funções Convexas



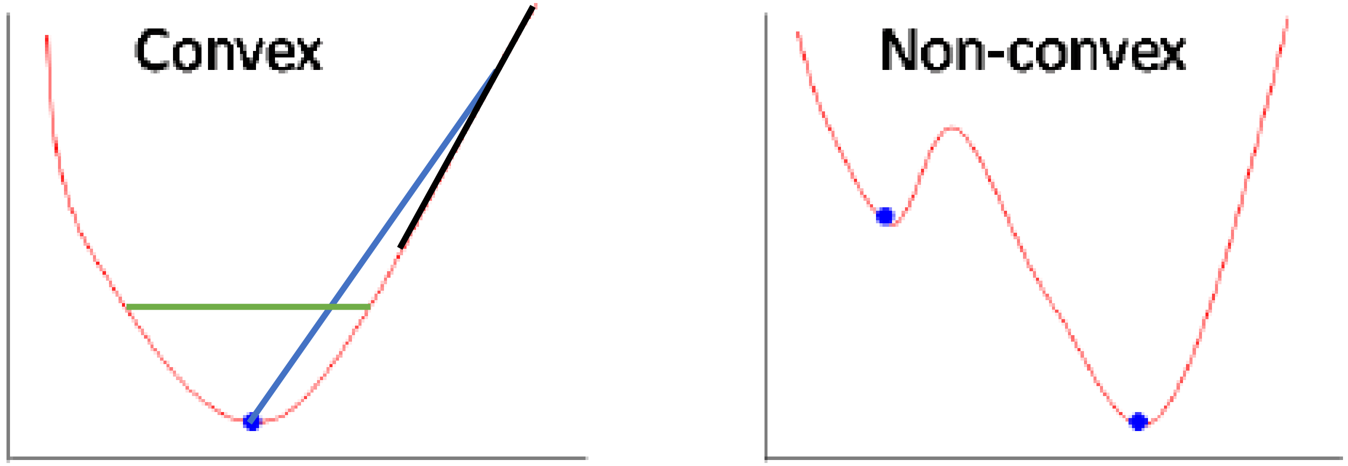
Funções Convexas

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.



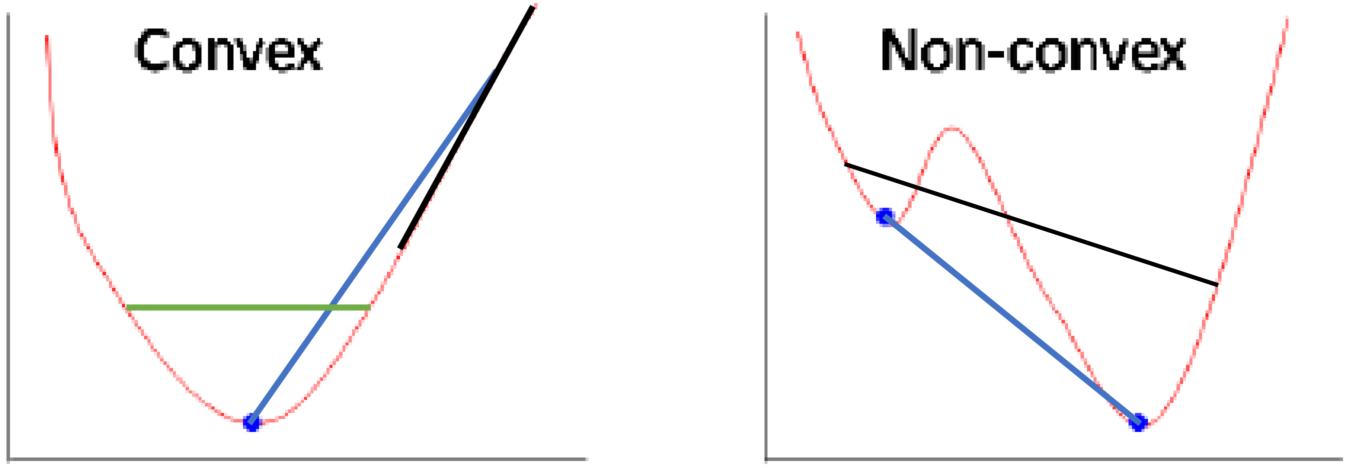
Funções Convexas

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.



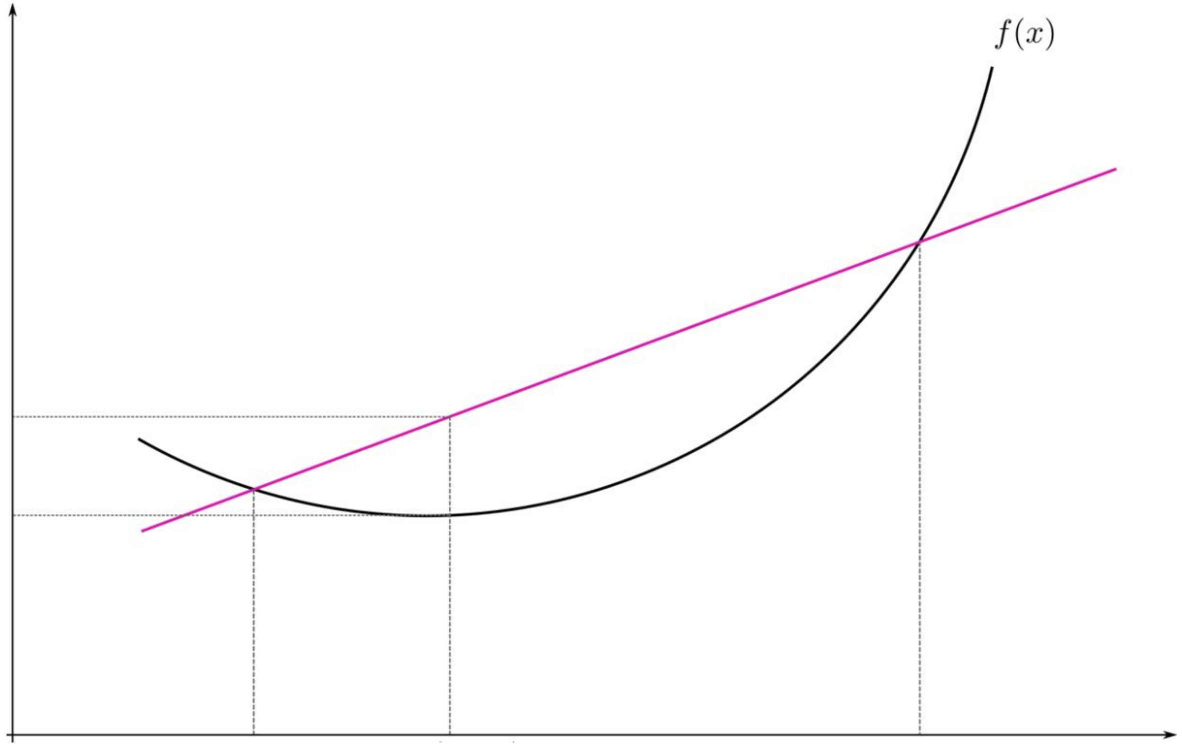
Funções Convexas

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.



Funções Convexas

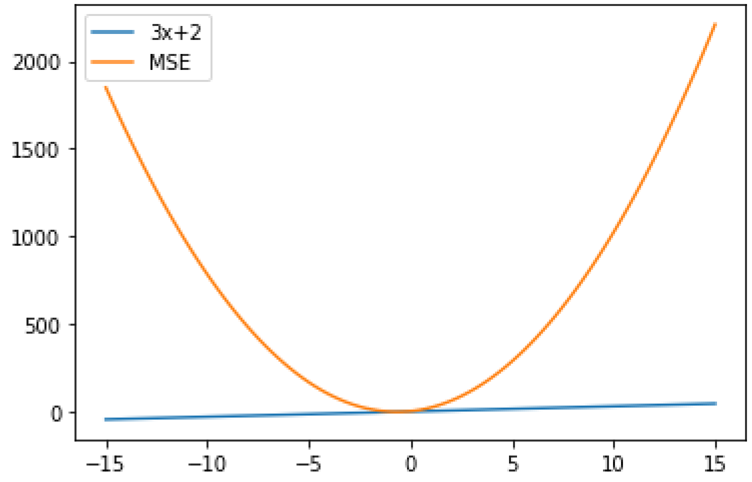
Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.



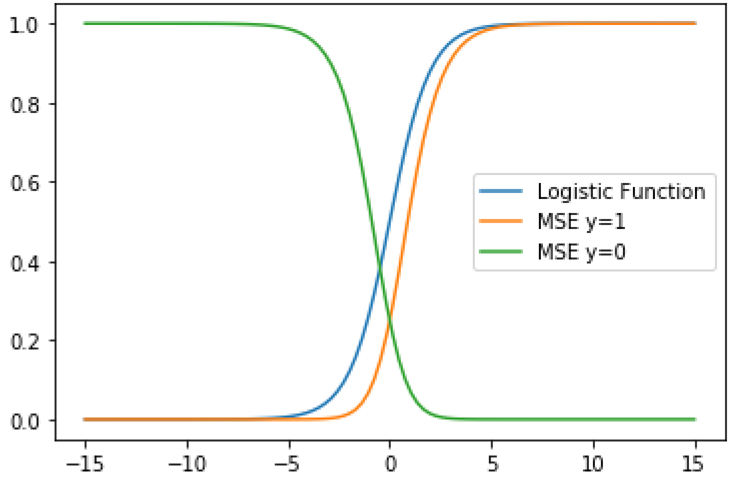
Convexidade de Funções de custo

Erros Quadráticos Médios (MSE) Vs Entropia Cruzada

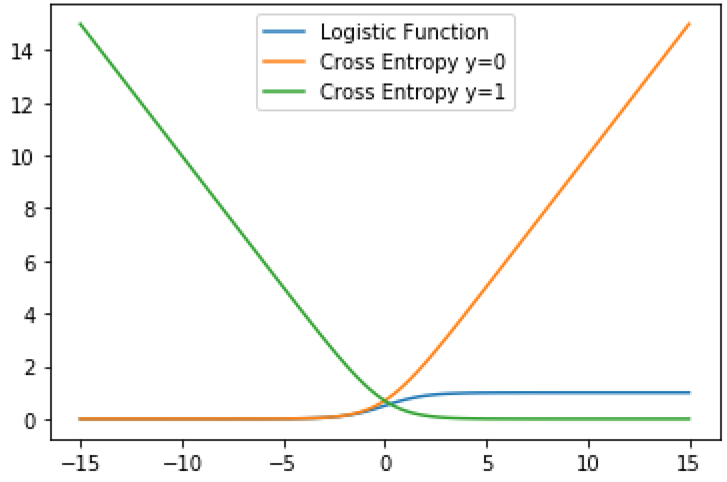
Reg. Linear & MSE



Reg. Logística & MSE



Reg. Logística & Entropia Cruzada



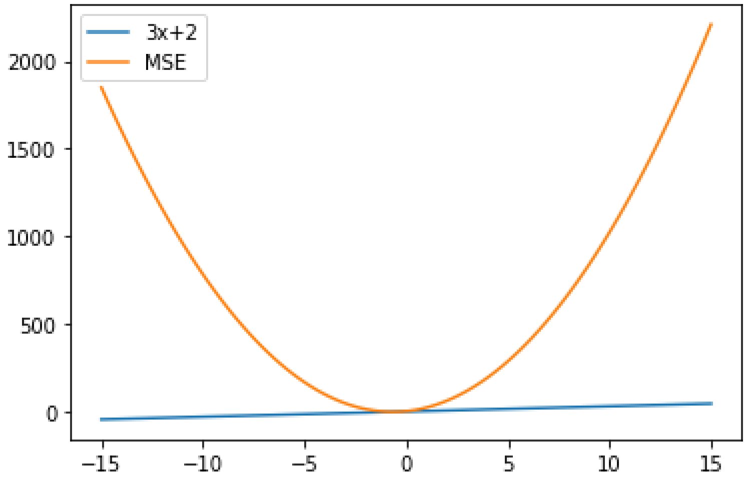
Convexidade de Funções de custo

Erros Quadráticos Médios (MSE) Vs Entropia Cruzada

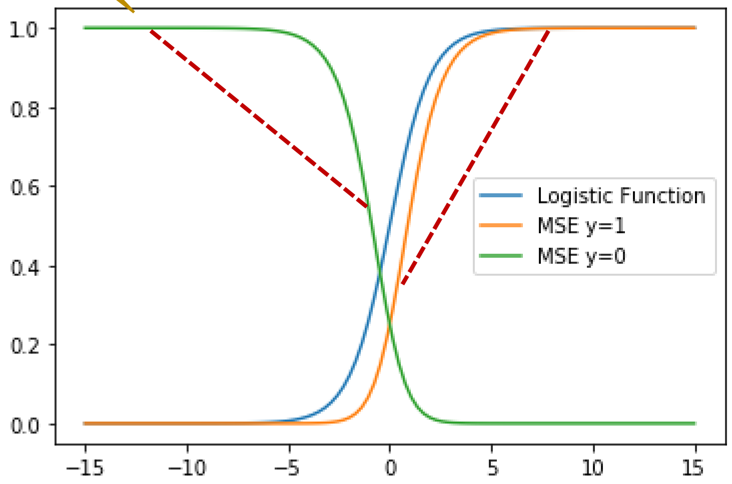
Não é convexa!

É convexa!

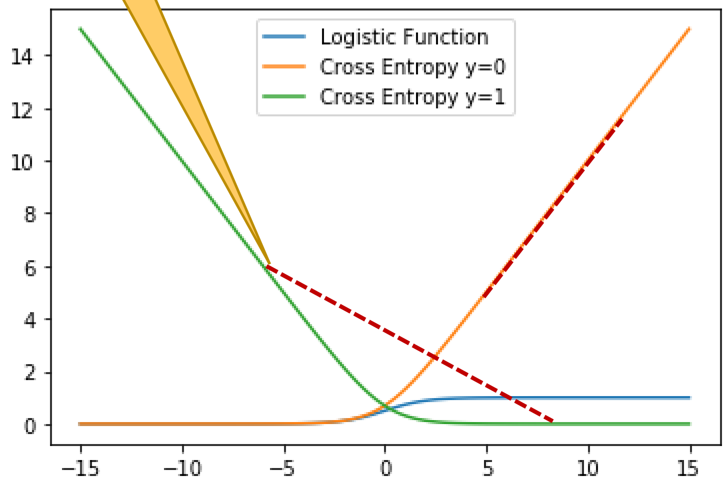
Reg. Linear & MSE



Reg. Logística & MSE



Reg. Logística & Entropia Cruzada



Sigmoides e Funções de custo

Exercício
Prático

Regressão logística: gradiente descendente

(4)

$$\frac{\partial J}{\partial \beta} = \frac{1}{m} \times transp(\vec{X}_{m,n}) \times [y_{m,1} - \hat{y}_{m,1}]$$

(3) Entropia Cruzada

$$cost(\beta, \alpha) = J = \frac{1}{m}\sum_{i=1}^{m} -[y_i \log(\hat{y}_i) + (1 - y_i)log(1 - \hat{y}_i)]$$

(5)

$$\frac{\partial J}{\partial \alpha} = \frac{1}{m} \times \sum [y_{m,1} - \hat{y}_{m,1}]$$

Hiper-Parametros

Learning Rate = Alpha:

Define a velocidade do aprendizado. Quando baixa, exige mais iteracoes ate a solucao otima, quando alta dificulta a convergencia do algoritmo.

Epochs:

Define quantas iteracoes de ajustes nos coeficientes serao executadas.

Ajuste dos pesos / coeficientes

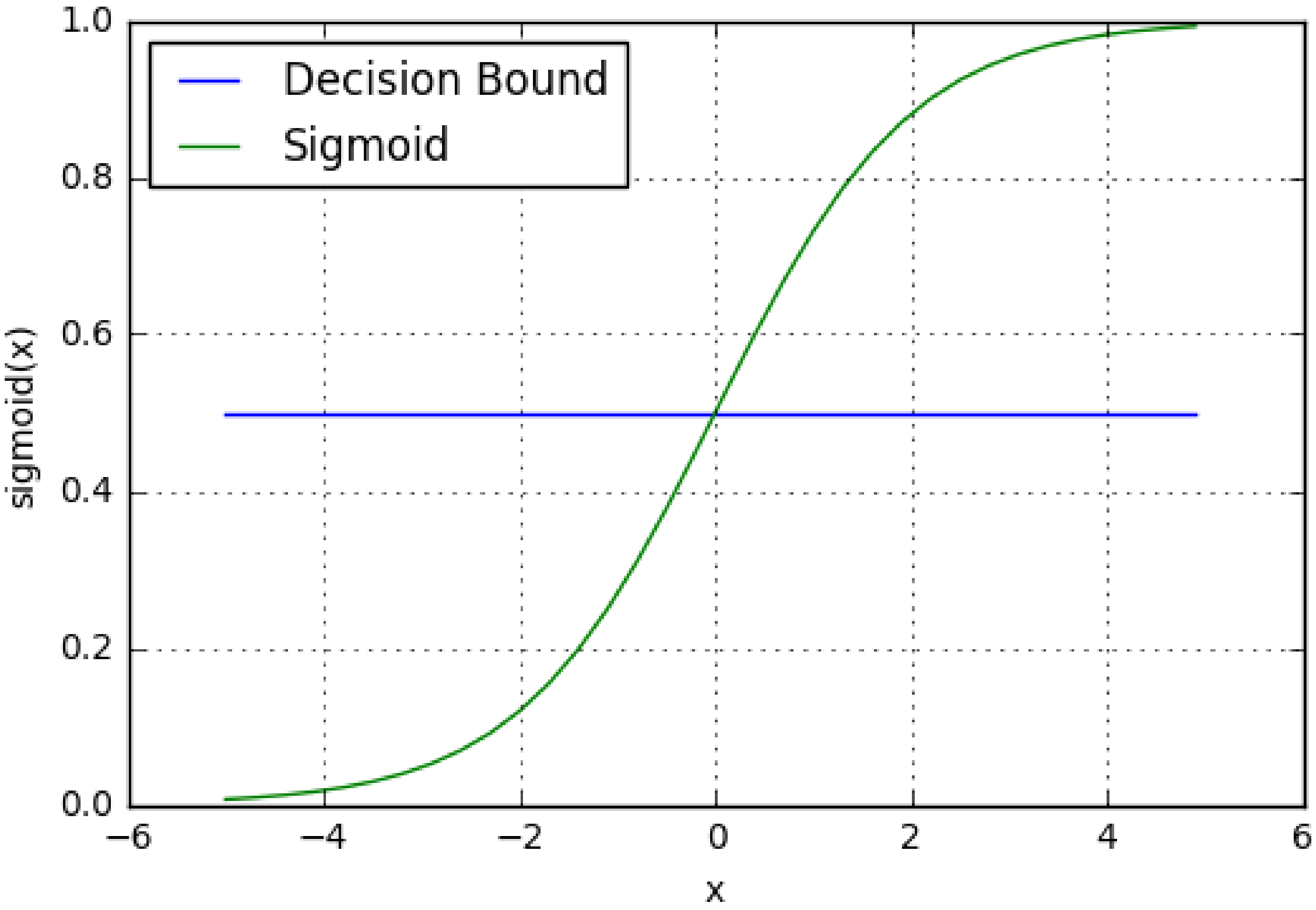
(6)

$$\vec{B}_{n,1} = \vec{B}_{n,1} - alpha \times \frac{\partial J}{\partial \beta}$$

(7)

$$\alpha = \alpha - alpha \times \frac{\partial J}{\partial \alpha}$$

Threshold de decisão



Regressão Logística Multiclasse

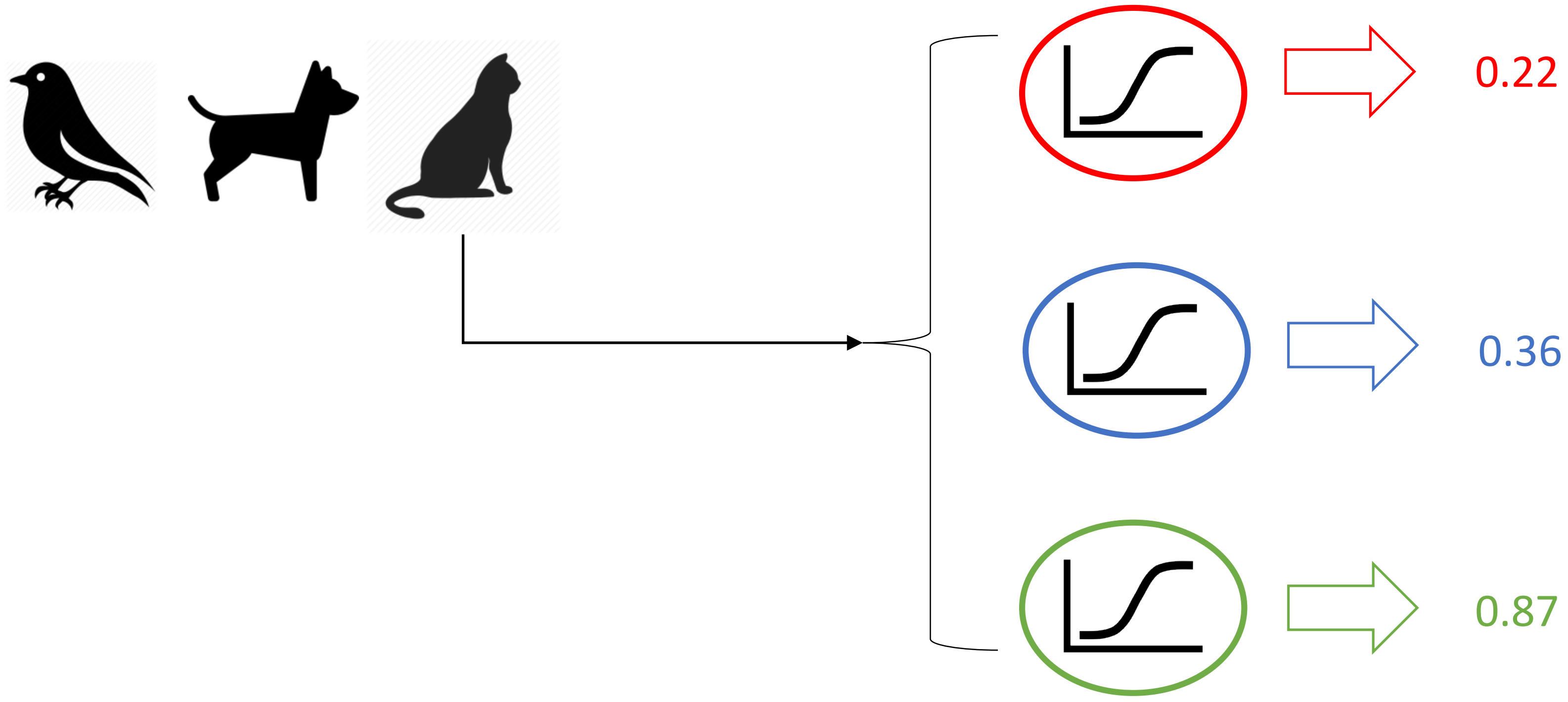
Juvenal J. Duarte

Abordagem para o problema

1. Para cada classe do problema, construa um classificador binário para recuperar a pontuação de cada registro em relação a esta classe.

2. Atribua a classe cuja ativação seja mais alta.

Abordagem para o problema: exemplo



Gradiente Descendente:

Batch, Mini Batch e Estocástico

Juvenal J. Duarte

Comparativo de abordagens

- ***Batch:*** A cada iteração/ época todos os registros de treino são avaliados antes de calcular o gradiente.

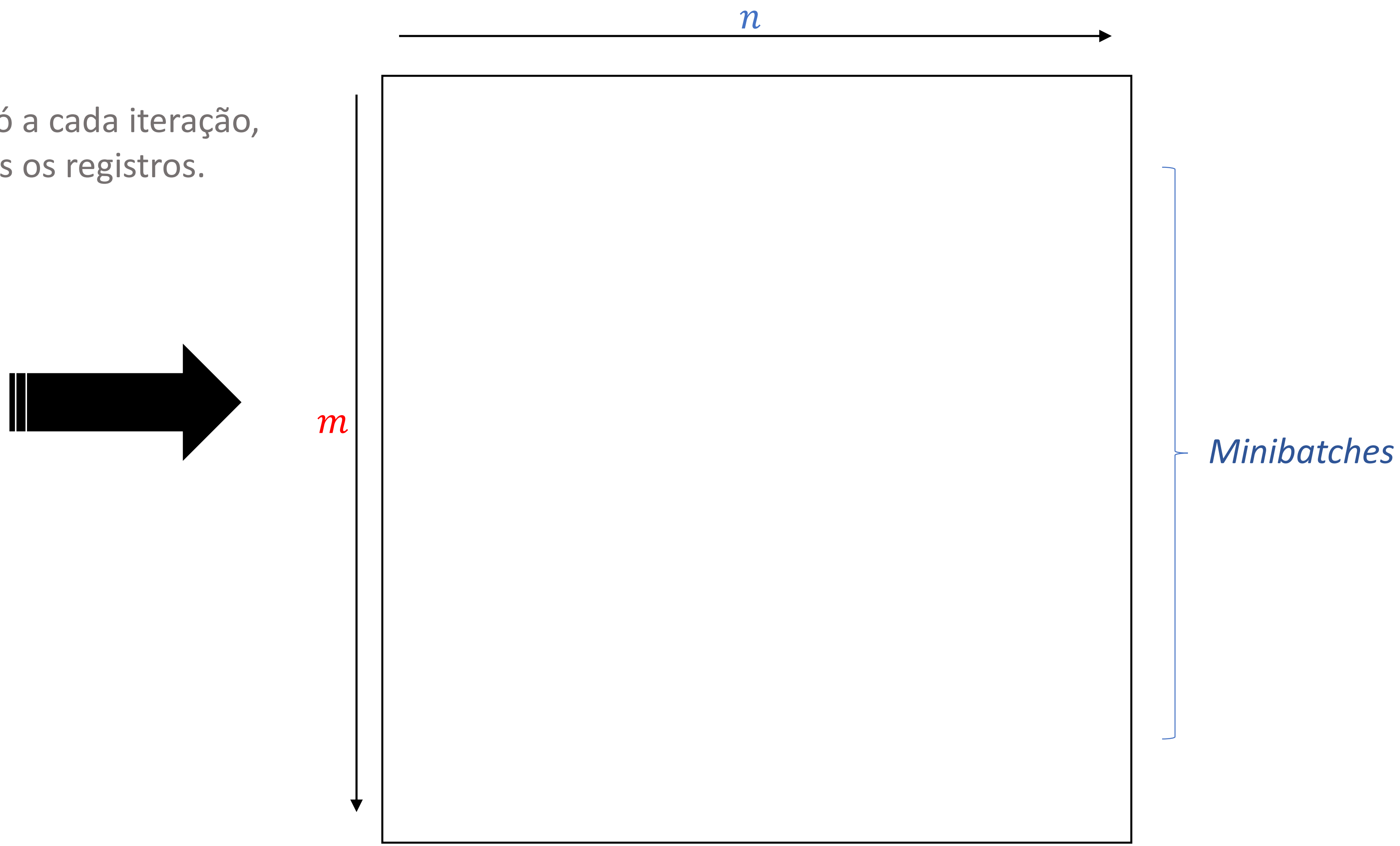
- ***Mini Batch:*** São divididos subgrupos de registros para avaliação do gradiente, com mais de um atualização por época.

- ***Estocástico:*** A cada época o gradiente é atualizado uma vez por registro.

Batch

Calcula o gradiente uma vez só a cada iteração, com o valor médio entre todos os registros.

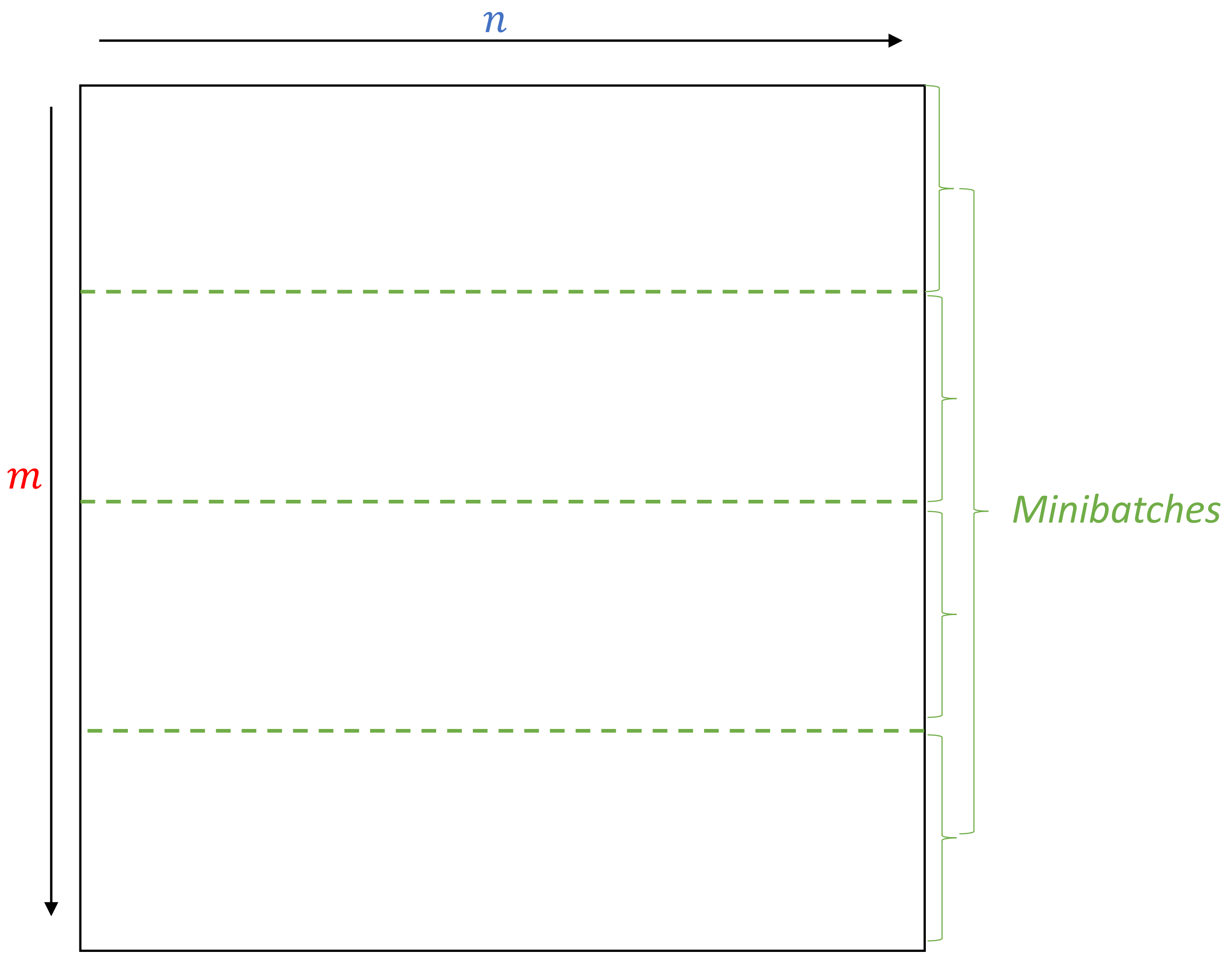
$$\vec{X}_{m,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$



Minibatches

Divide os dados em porções de tamanho fixo, atualiza os pesos com o gradiente calculado a partir desses exemplos.

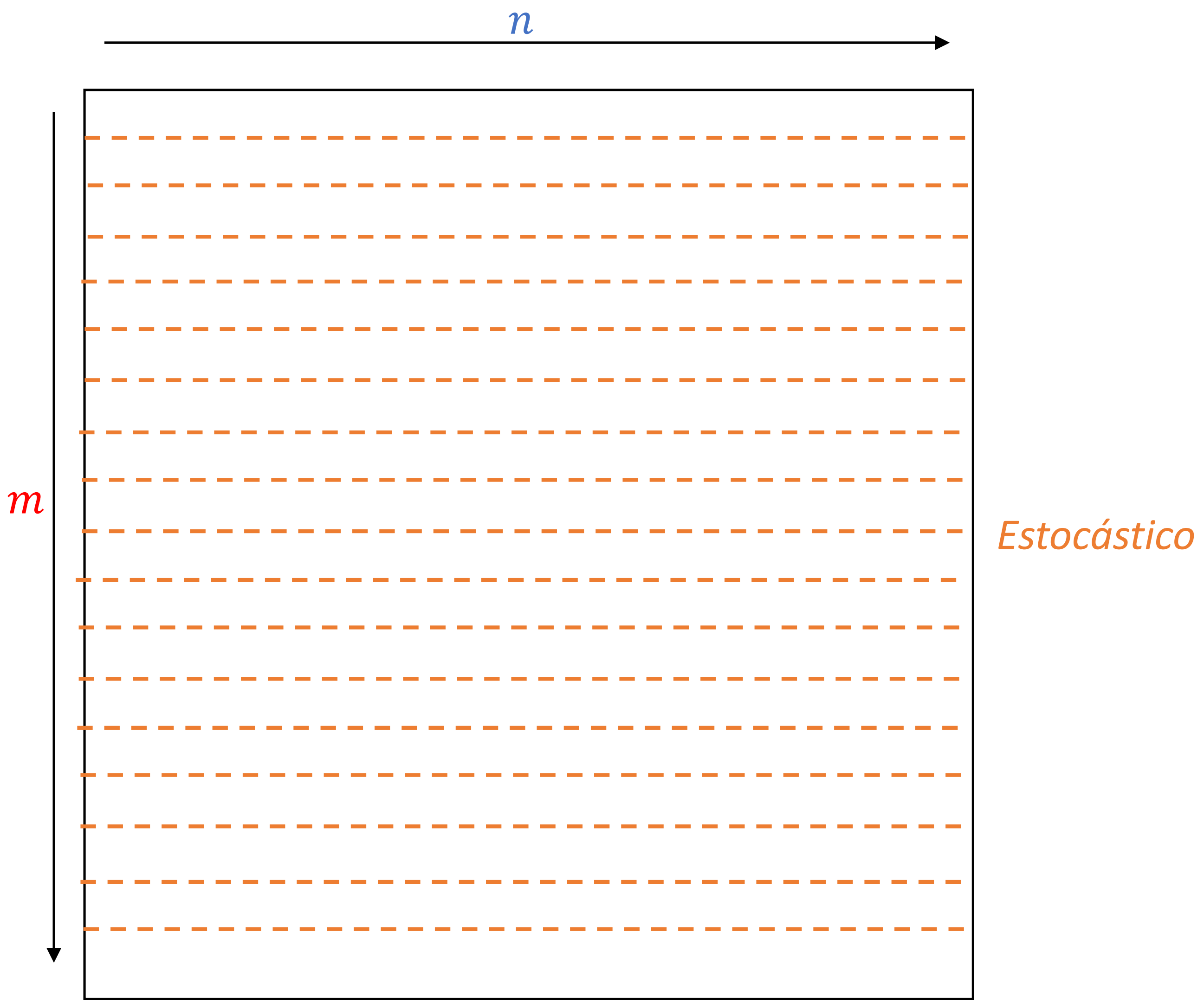
$$\vec{X}_{m,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$



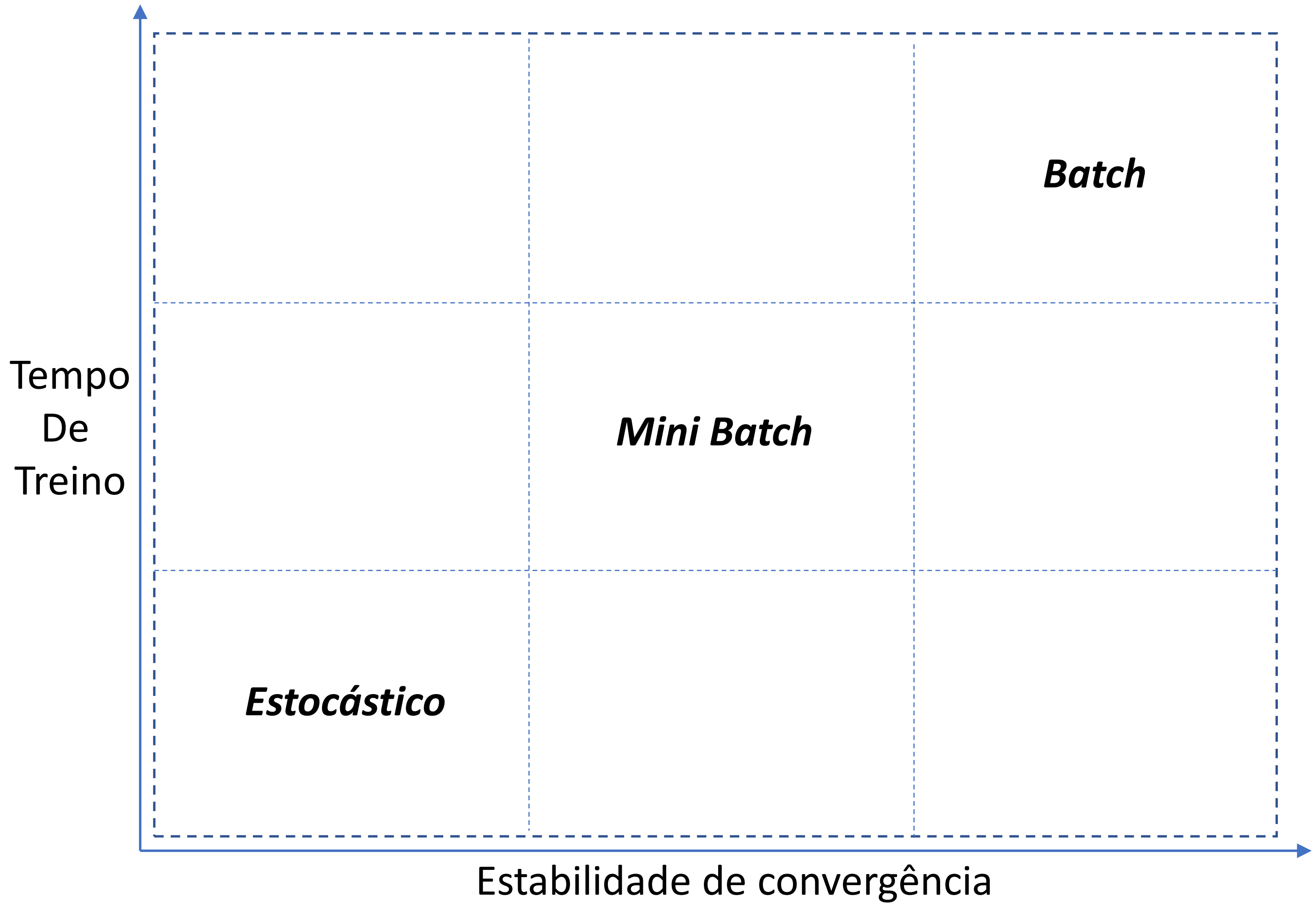
Estocástico

Calcula o gradiente e atualiza os pesos a para cada Registro.

$$\vec{X}_{m,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

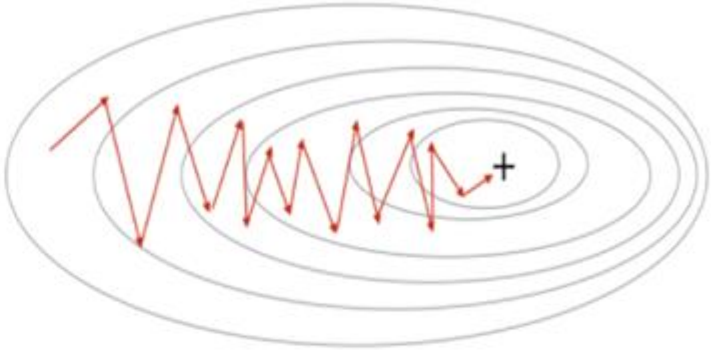


Comparativo de abordagens

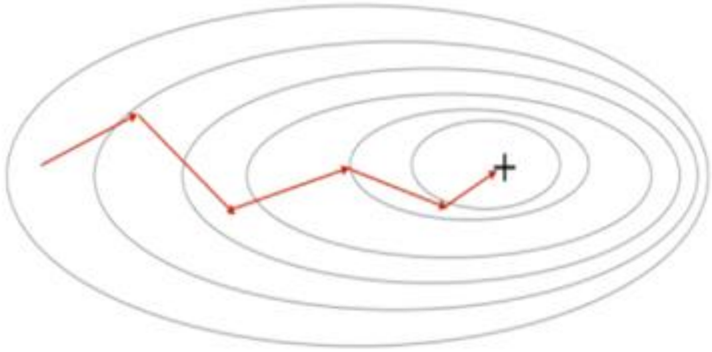


Comparativo de abordagens

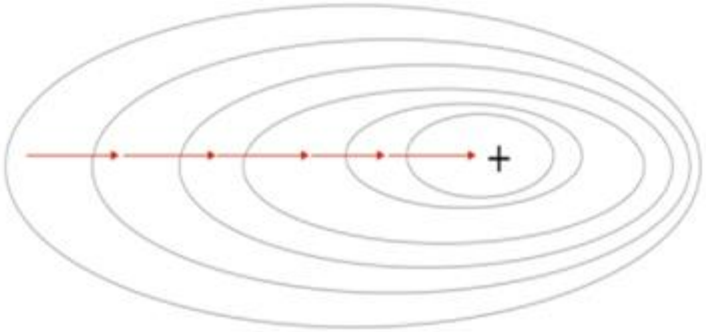
Estocástico



Mini Batch



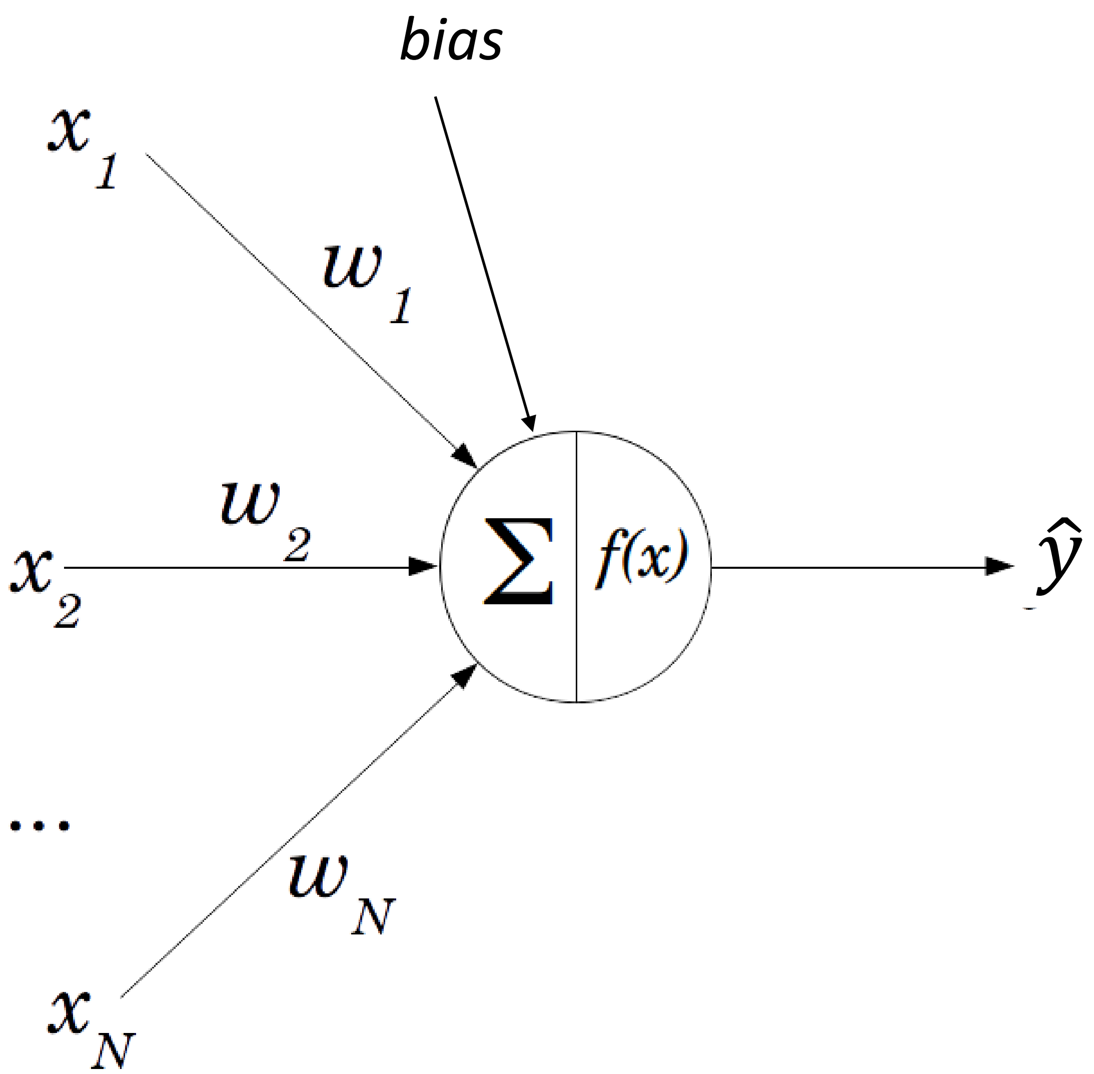
Batch



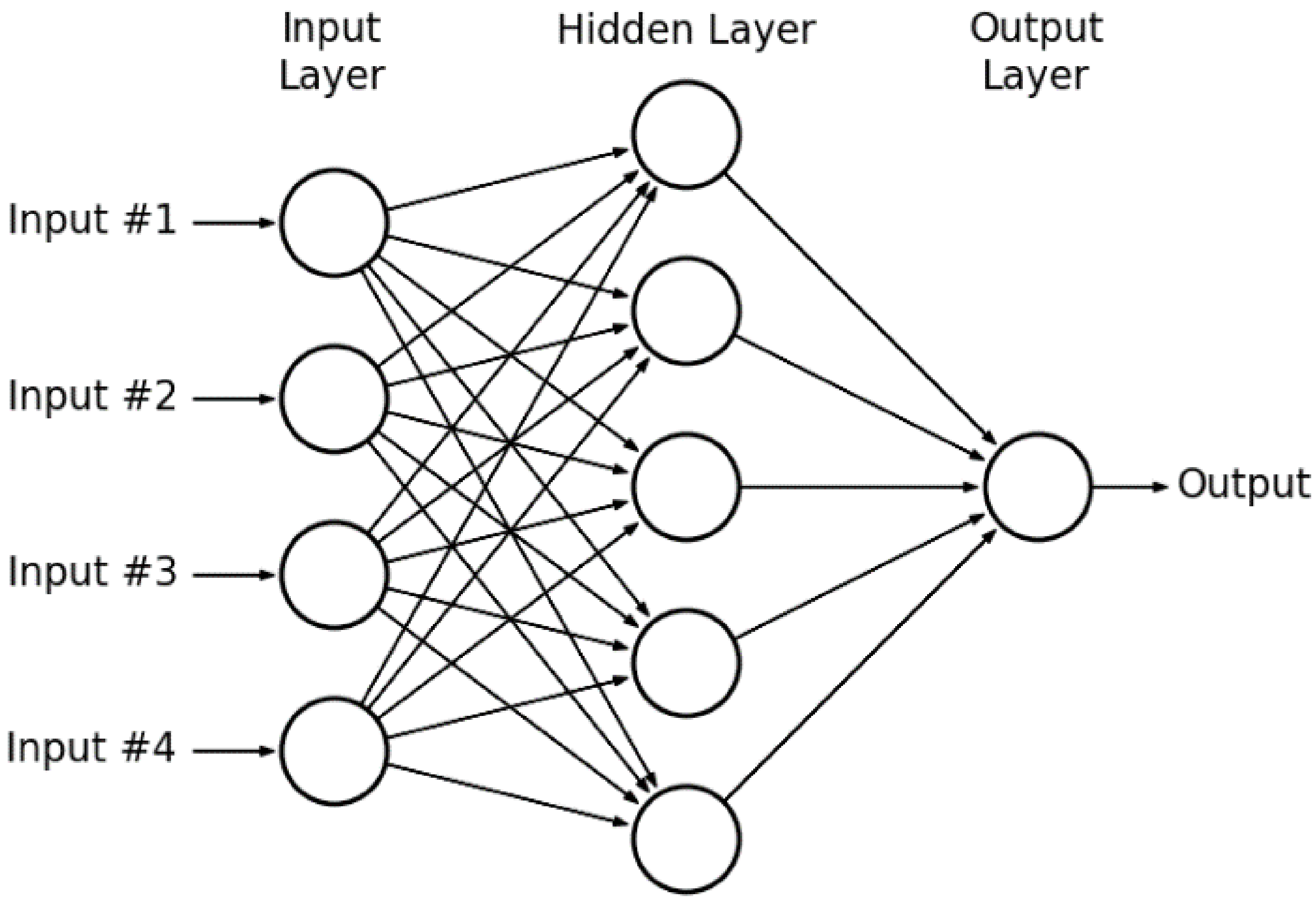
Multilayer Perceptron (MLP): Introdução

Juvenal J. Duarte

O perceptron



O Multilayer Perceptron



O Multilayer Perceptron: definições

Quais as variáveis que temos por neurônio?

$\vec{A}^{[i-1]}$: entradas vindas dos neurônios da camada anterior. Quando $i = 0$, representa os valores de cada um dos atributos de entrada.

$a_k^{[i]}$: saída do neurônio k na camada i.

$\overrightarrow{W}_k^{[i]}$: vetor de pesos para cada uma das saidas camada $i - 1$, usado no neurônio k.

$b_k^{[i]}$: bias usado no neurônio k, camada i.

O Multilayer Perceptron: definições

Quais as variáveis que temos por neurônio?

$\vec{A}^{[i-1]}$: $\vec{X}$

$a_k^{[i]}$: $\hat{y}$

$\overrightarrow{W}_k^{[i]}$: $\vec{\beta}$

$b_k^{[i]}$: α

O Multilayer Perceptron