Machine Learning

Dia 8 - Redes Neurais

ImageU - Grupo de Pesquisa em Machine Learning e Visão Computacional

https://imageu.github.io/

Curso de Verão 2022

Instituto de Matemática e Estatística - IME USP

Programa

1. Modelos Não-Lineares

2. Redes Neurais

Modelos Não-Lineares

Linear $\times$ não-linear

- Seja $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ uma observação do seu dataset
- Seja $\mathbf{w} = (w_1, w_2, \ldots, w_d) \in \mathbb{R}^d$, $b \in \mathbb{R}$ um conjunto de pesos
- Uma função linear tem a forma:

$$s = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

- Já funções não-lineares podem aparecer de várias formas:

$$s = w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 + b$$

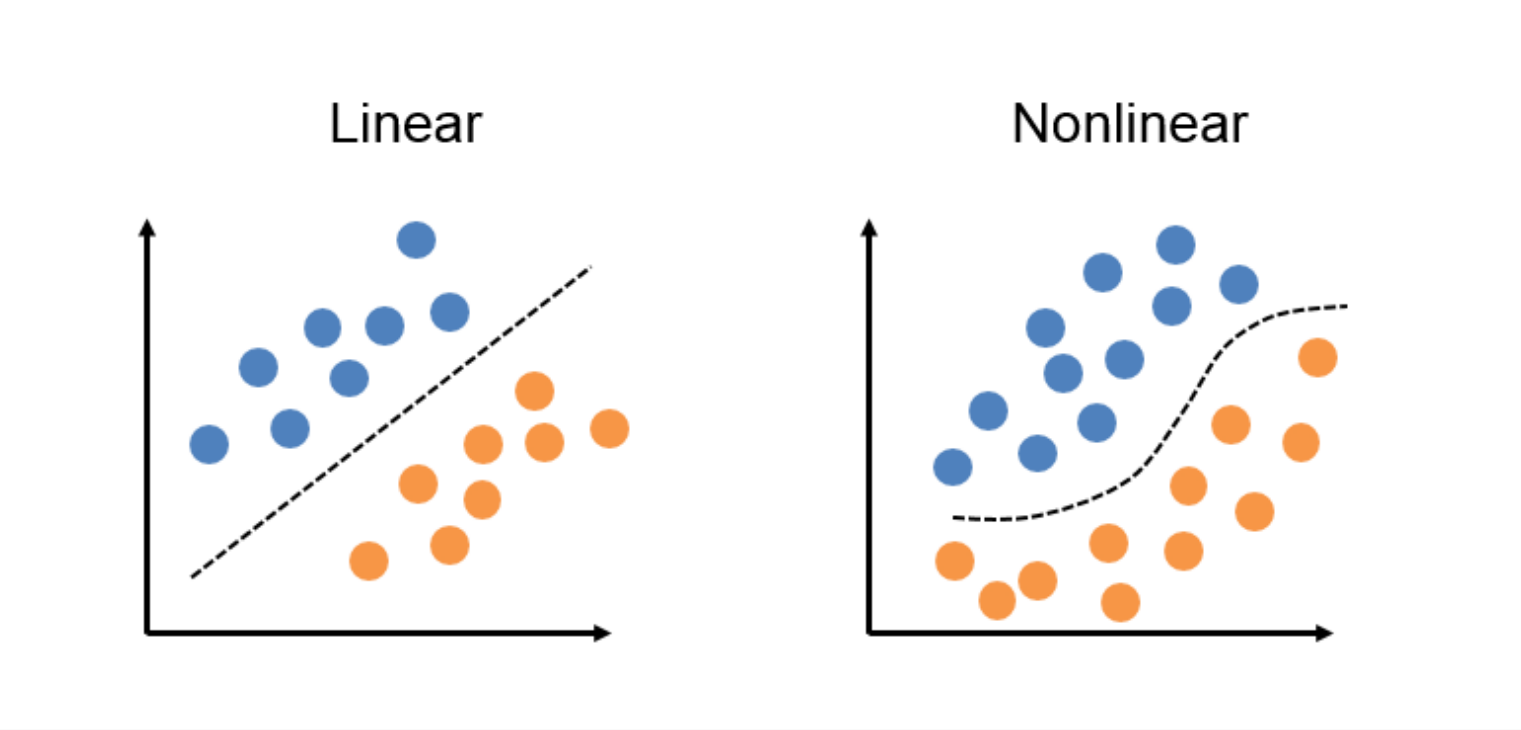
$$s = w_1 x_1^2 + w_2 x_2^2 + b$$

Linear $\times$ não-linear

- Uma função $s : \mathbb{R}^d \to \mathbb{R}$ qualquer pode ser usada como classificador
  - $s < 0 \implies$ classe $:=$ negativa
  - $s > 0 \implies$ classe $:=$ positiva
  - $s = 0 \implies$ fronteira de decisão

Linear × não-linear

- **Linear:** hiperplanos (retas no espaço 2D)
- **Não-linear:** superfícies complexas (ex.: polinomiais)



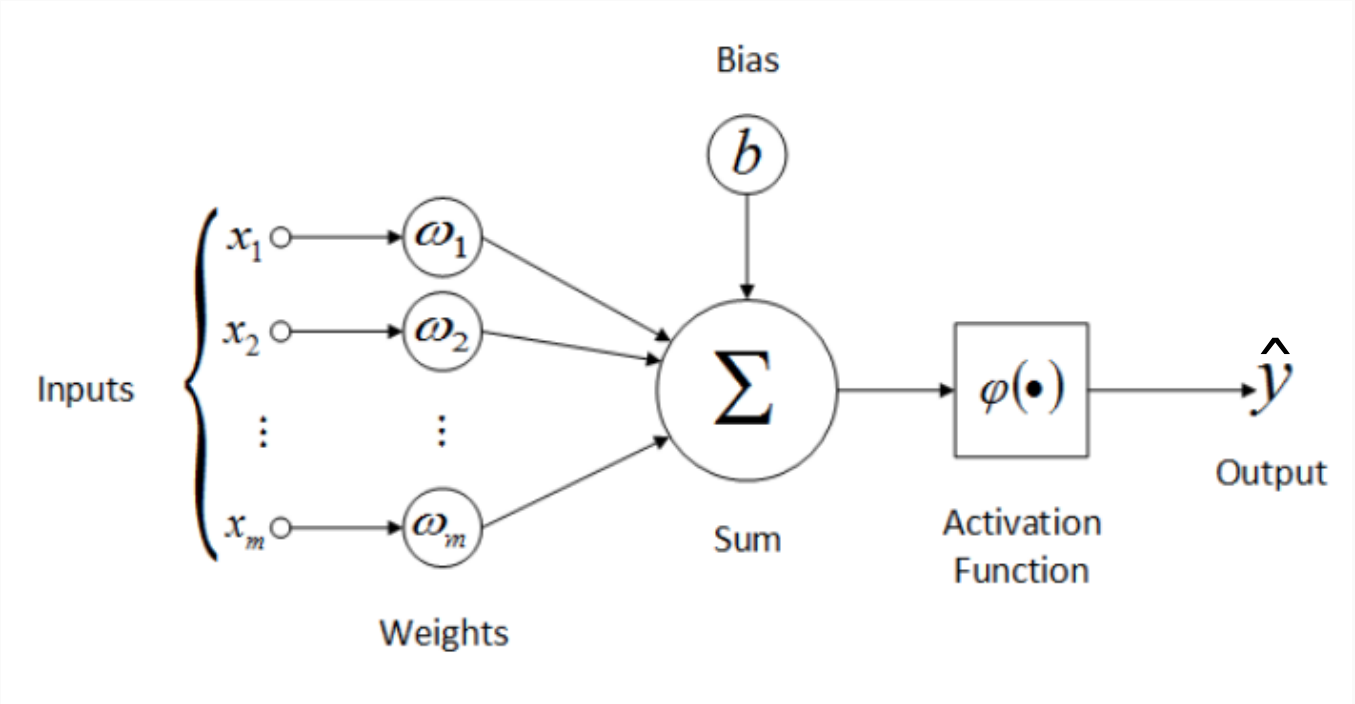
Fonte: https://jtsulliv.github.io/perceptron/

Revisão: Regressão Logística

- Classificador Binário
- Saída a ser estimada é 0 ou 1
- Saída gerada pode ser pensada como sendo $P(y = 1|\mathbf{x})$
- Função de custo clássica a ser otimizada: entropia cruzada
- Minimização da função de custo pode ser feita usando gradient descent

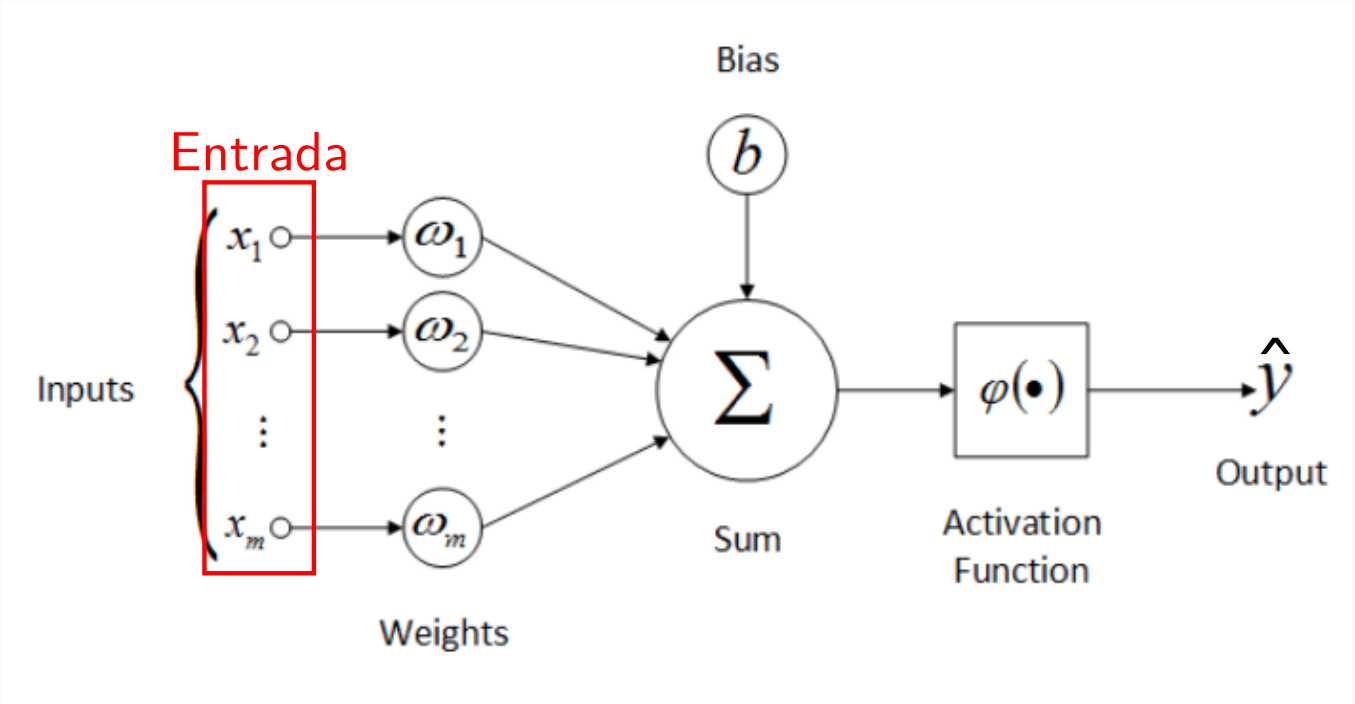
Revisão: Regressão Logística

- Diagrama de um classificador logístico:



Revisão: Regressão Logística

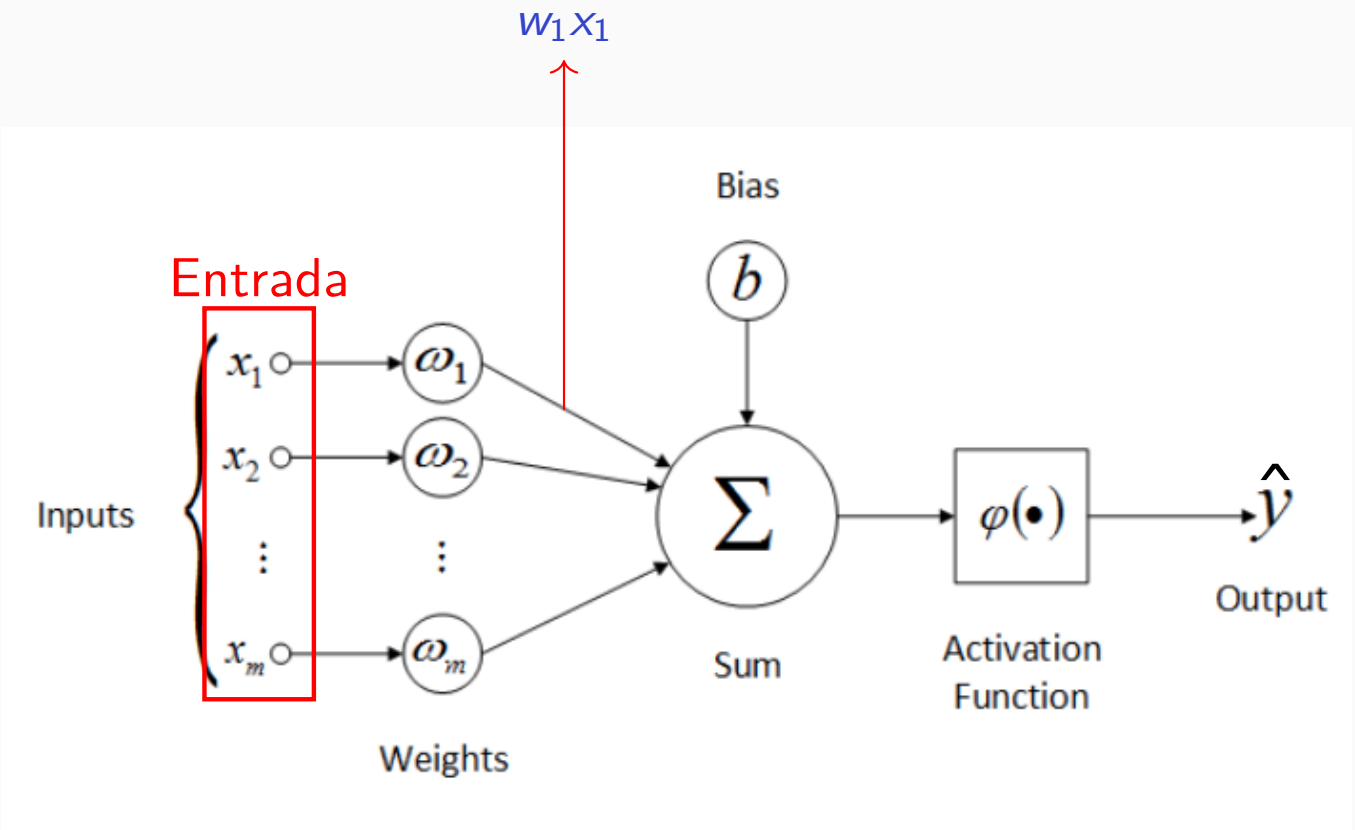
- Diagrama de um classificador logístico:



Forward pass $\Longrightarrow$

Revisão: Regressão Logística

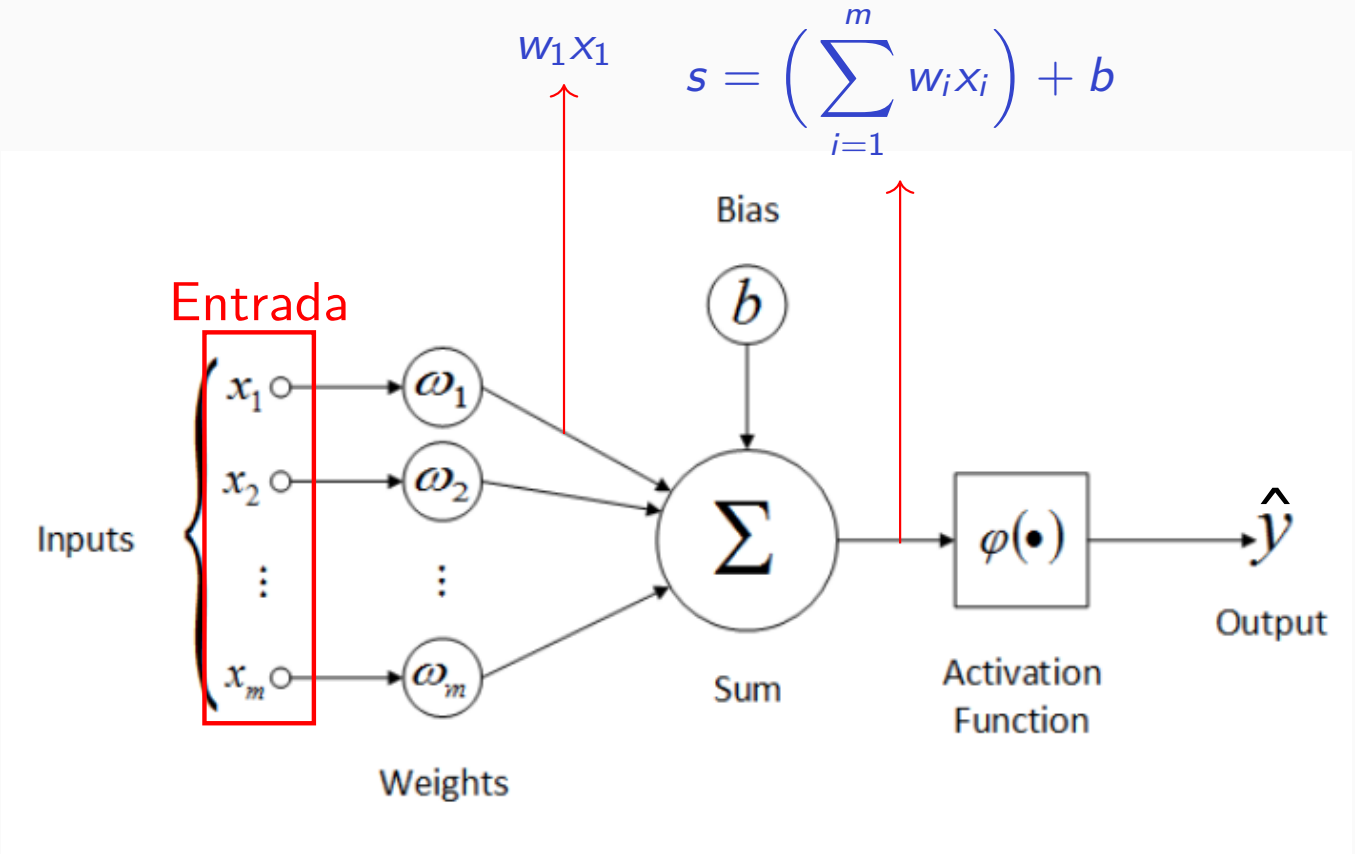
- Diagrama de um classificador logístico:



Forward pass $\Longrightarrow$

Revisão: Regressão Logística

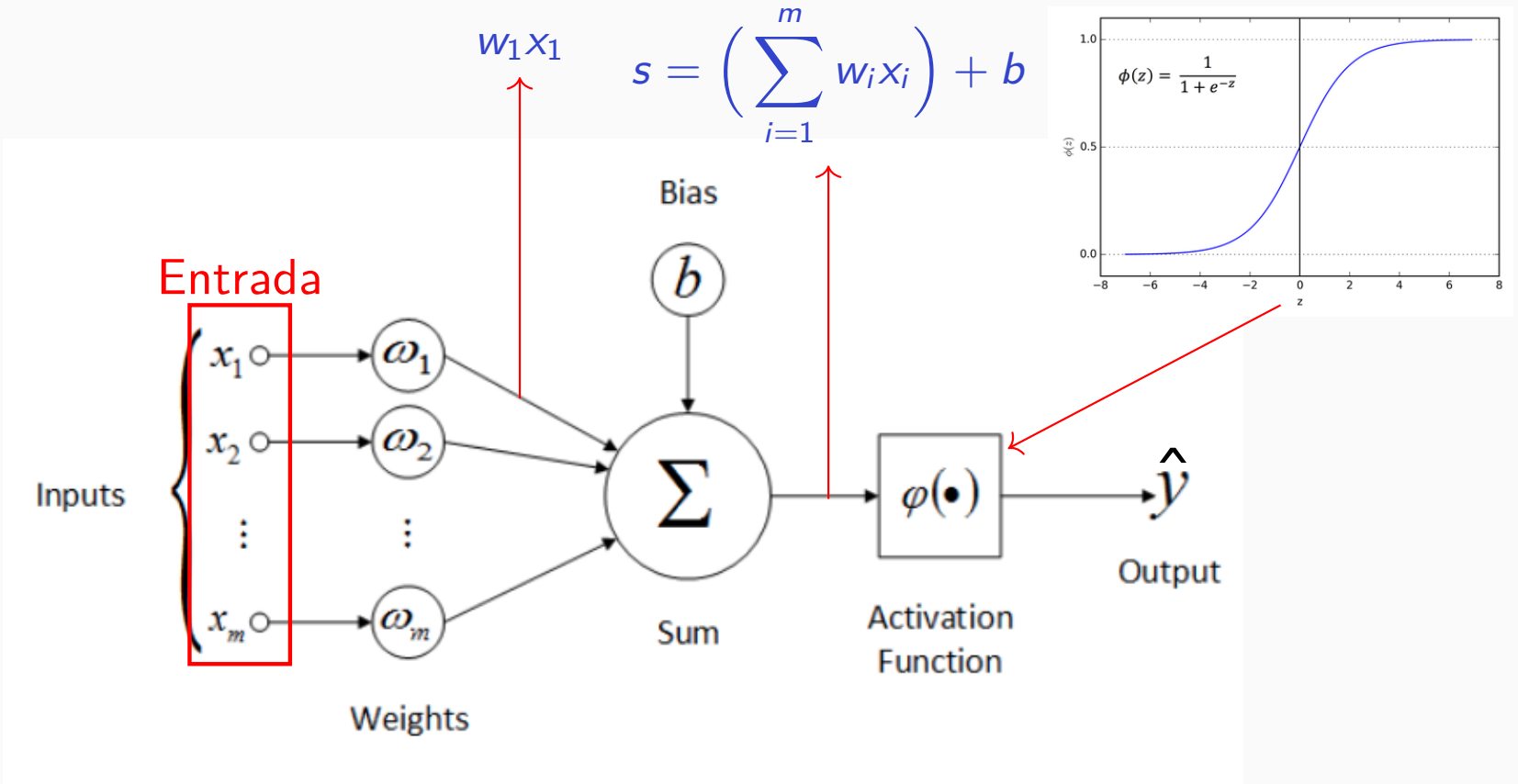
- Diagrama de um classificador logístico:



Forward pass $\Longrightarrow$

Revisão: Regressão Logística

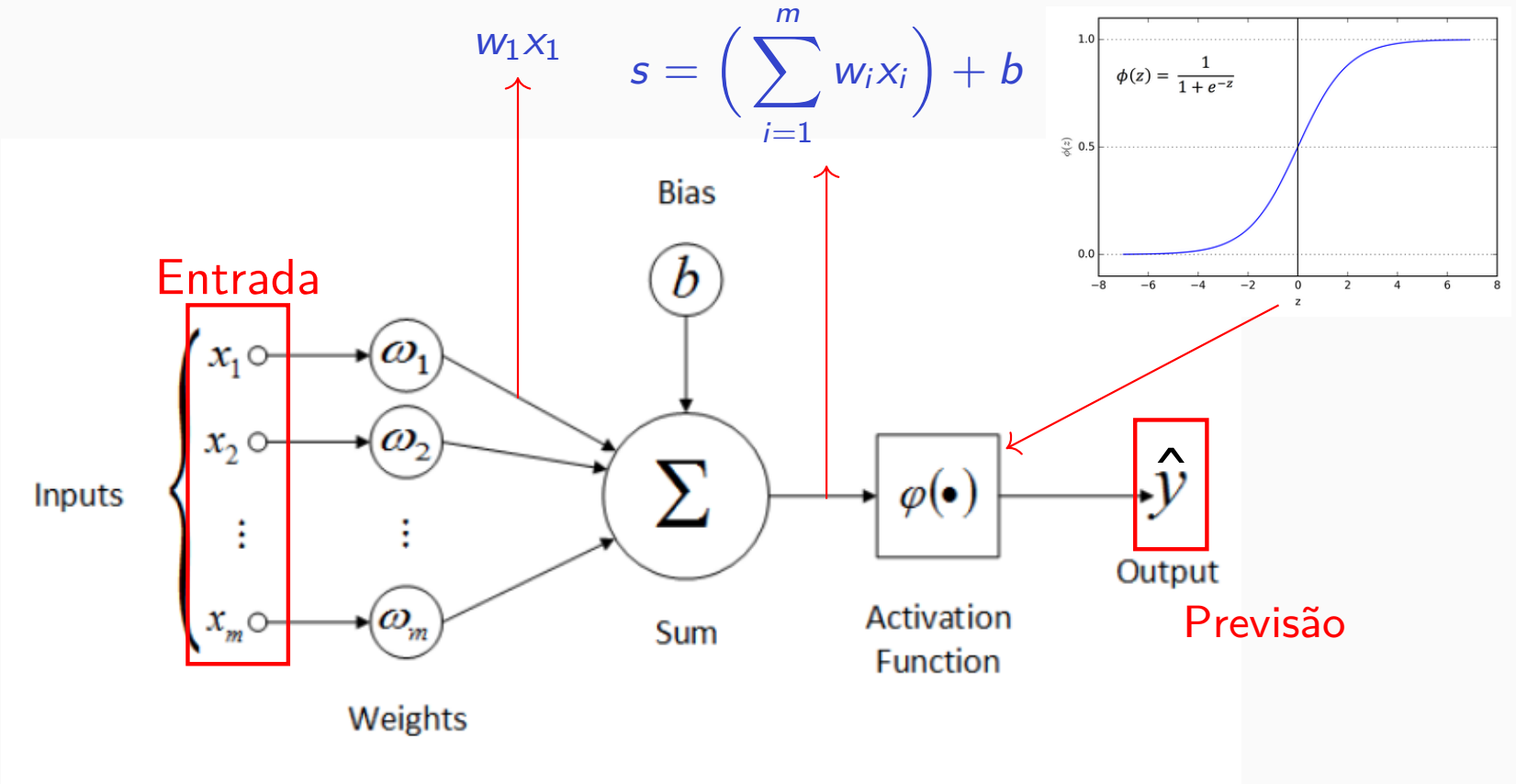
- Diagrama de um classificador logístico:



Forward pass ⟹

Revisão: Regressão Logística

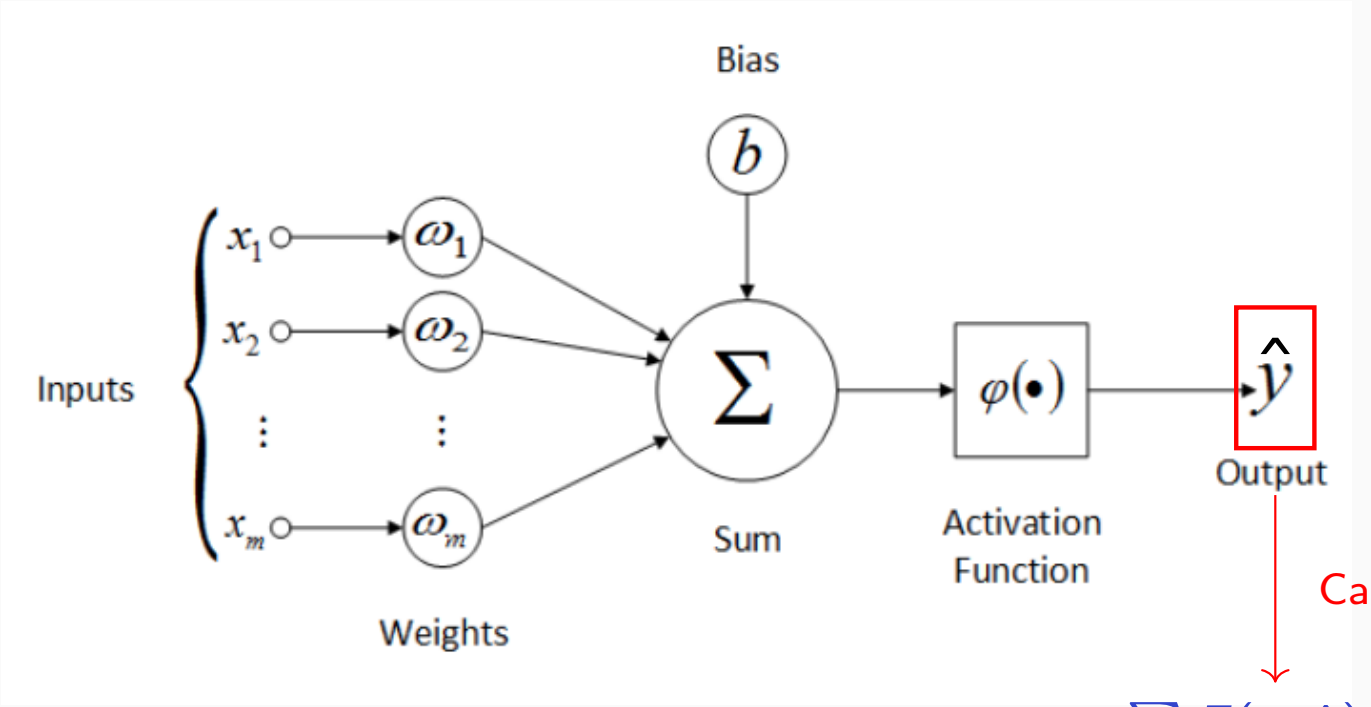
- Diagrama de um classificador logístico:



Forward pass ⟹

Revisão: Regressão Logística

- Diagrama de um classificador logístico:

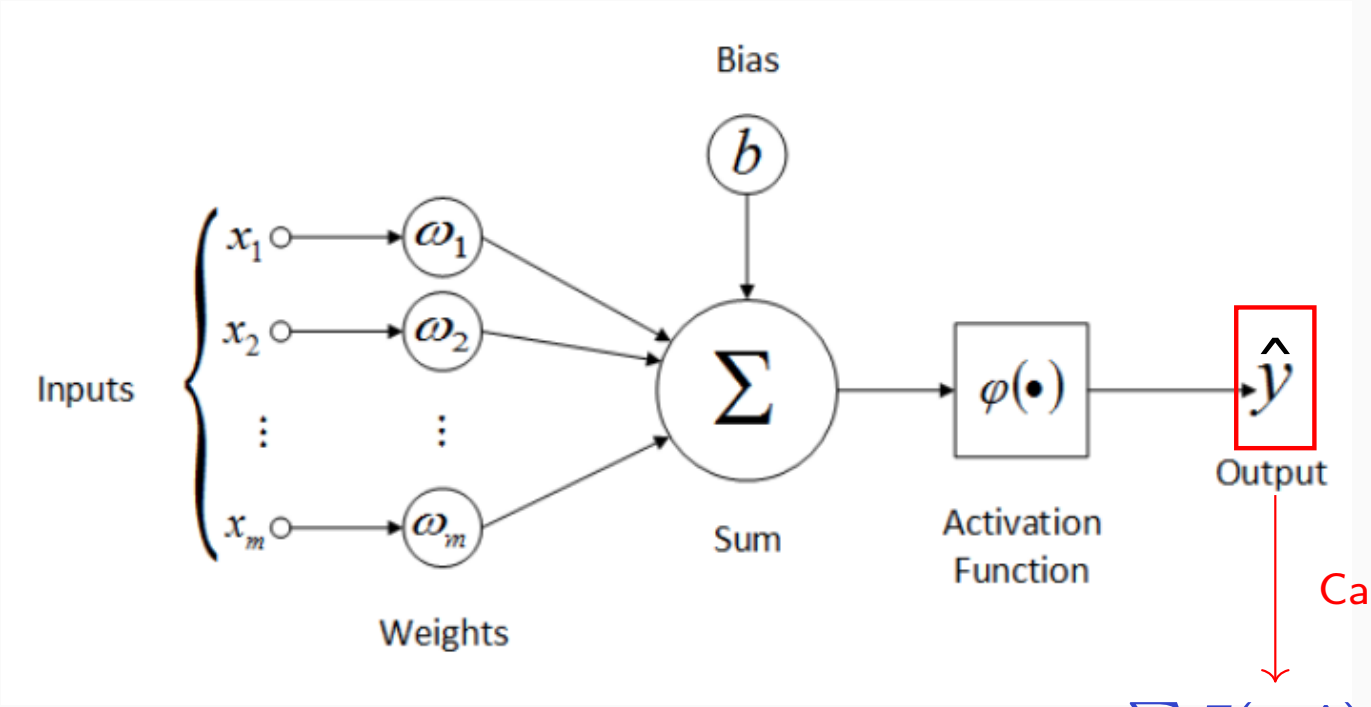


$$\sum E(y, \hat{y}) =$$

$$-\frac{1}{N} \sum_{n=1}^{N} \left[y^{(n)} \ln \hat{y}^{(n)} + (1 - y^{(n)}) \ln(1 - \hat{y}^{(n)}) \right]$$

Revisão: Regressão Logística

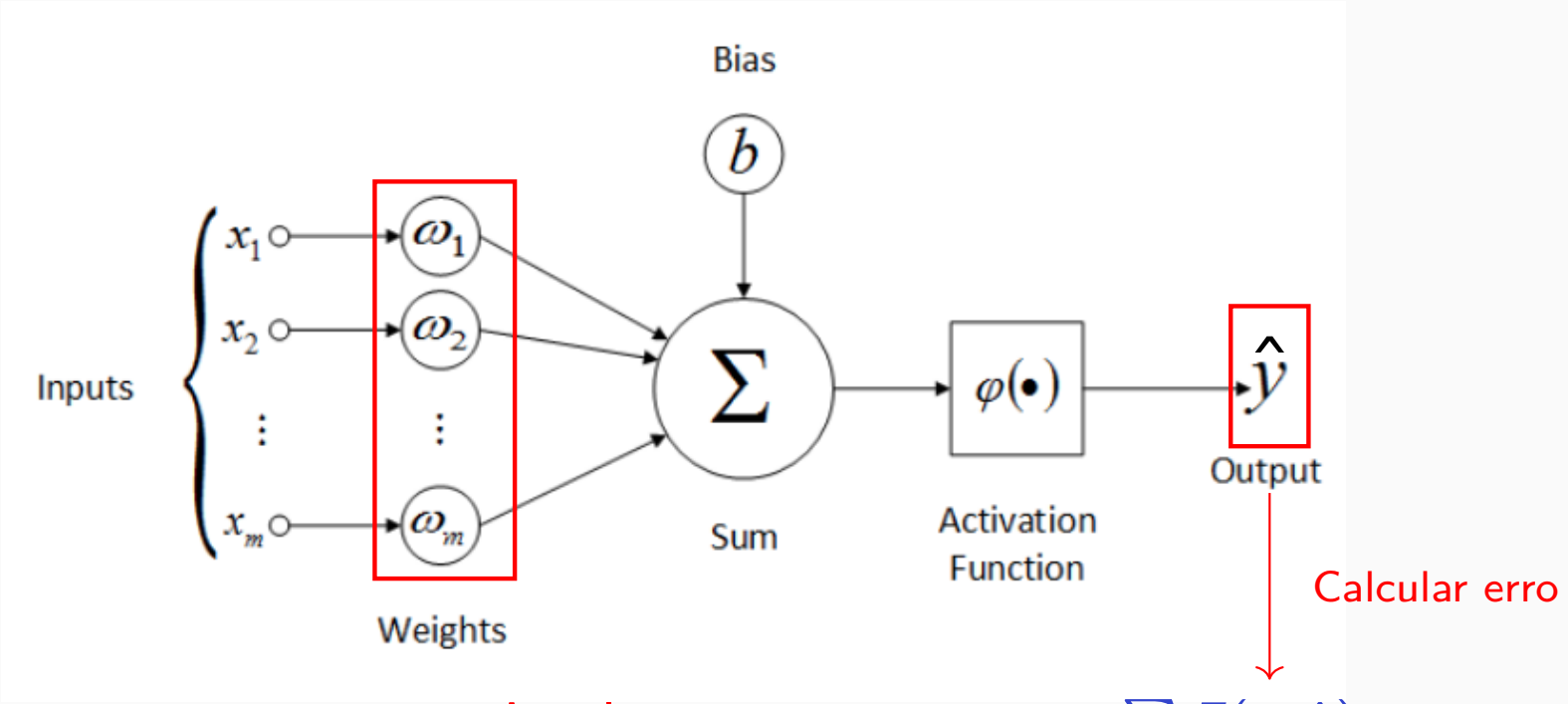
- Diagrama de um classificador logístico:



Backward pass $\Longleftarrow$

Revisão: Regressão Logística

- Diagrama de um classificador logístico:



Backward pass $\Longleftarrow$

Revisão: Regressão Logística

- Entropia Cruzada:

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}\ln\hat{y}^{(n)} + (1-y^{(n)})\ln(1-\hat{y}^{(n)})\right]$$

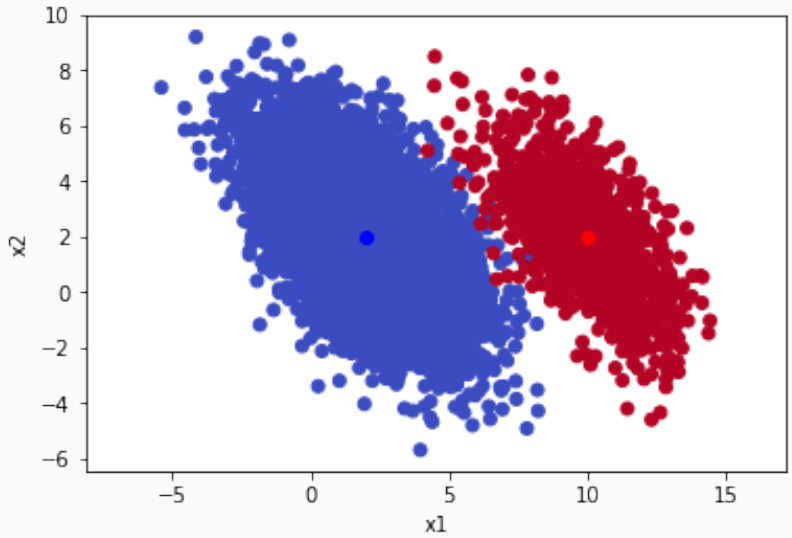
- Gradient Descent em linhas gerais:
  1. Chutar um valor para $\mathbf{w}$
  2. Calcular o gradiente de J no ponto $\mathbf{w}$ ("direção de maior inclinação")
  3. Alterar $\mathbf{w}$ no sentido oposto ao do vetor gradiente
  4. Repetir passos (2)-(3)

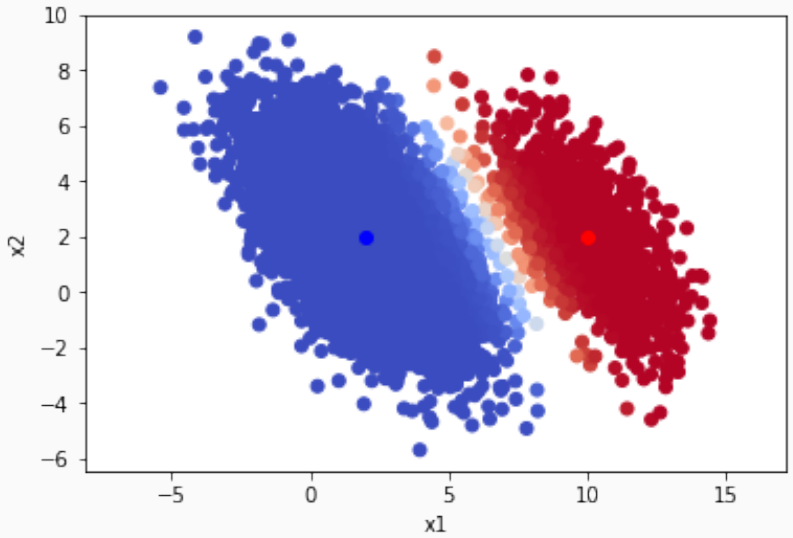
Revisão: Regressão Logística

- Regressão logística gera uma superfície de decisão linear:

$$s = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$
$$\hat{y} = \hat{P}(y = 1|\mathbf{x}) = \frac{1}{1+e^{-s}} \in [0, 1]$$

- Alterando-se **w** (supondo **x** fixo), pode-se fazer $\hat{y}$ variar entre 0 e 1.





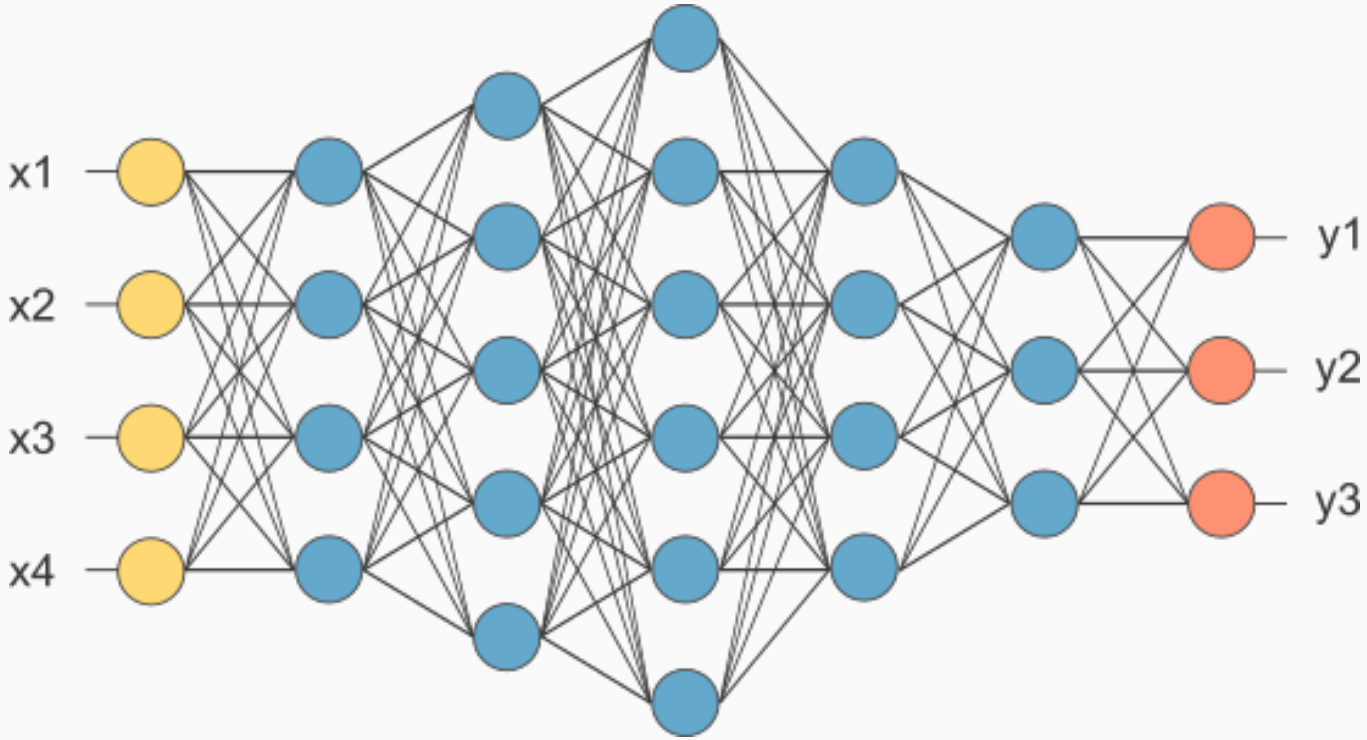
Redes Neurais

Redes Neurais

- Modelo para criarmos classificadores ou regressores não-lineares
- Podem ser compostas de unidades de processamento simples, por exemplo, regressores logísticos

Redes Neurais

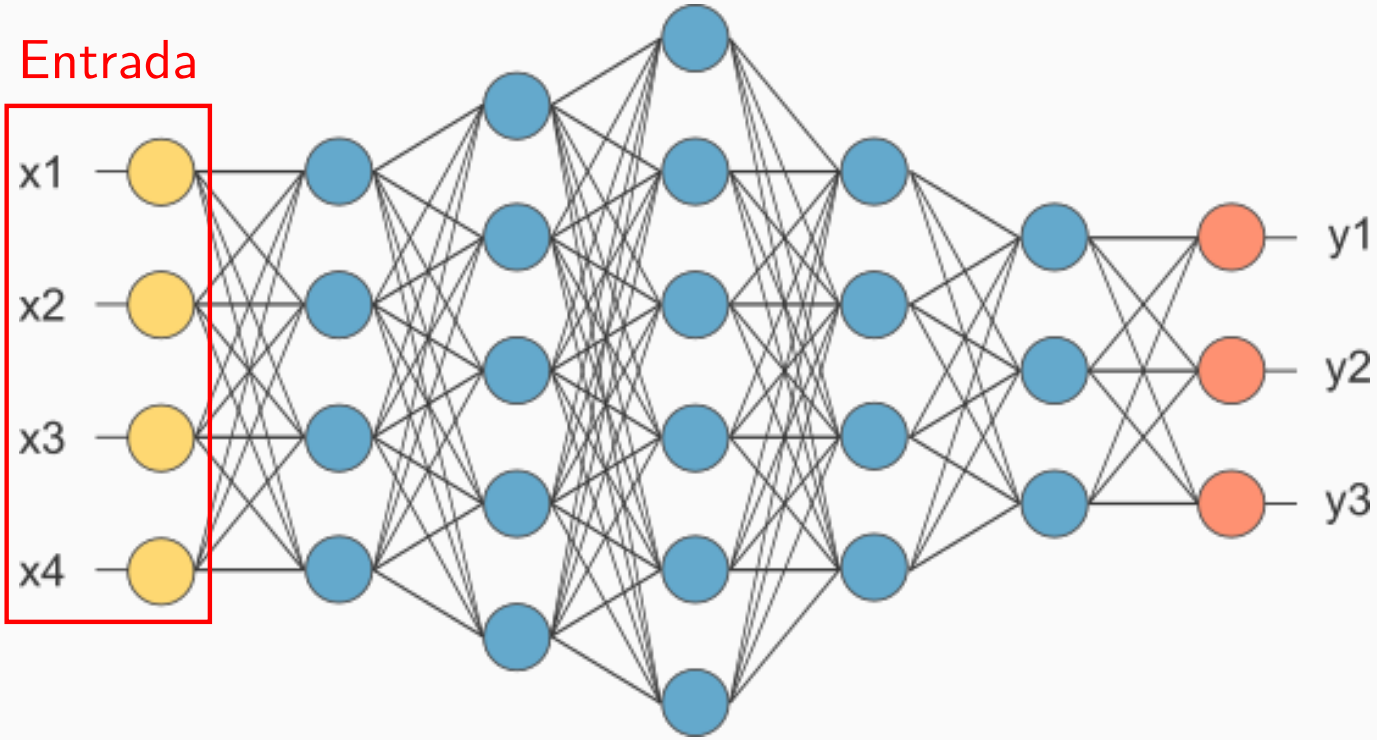
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

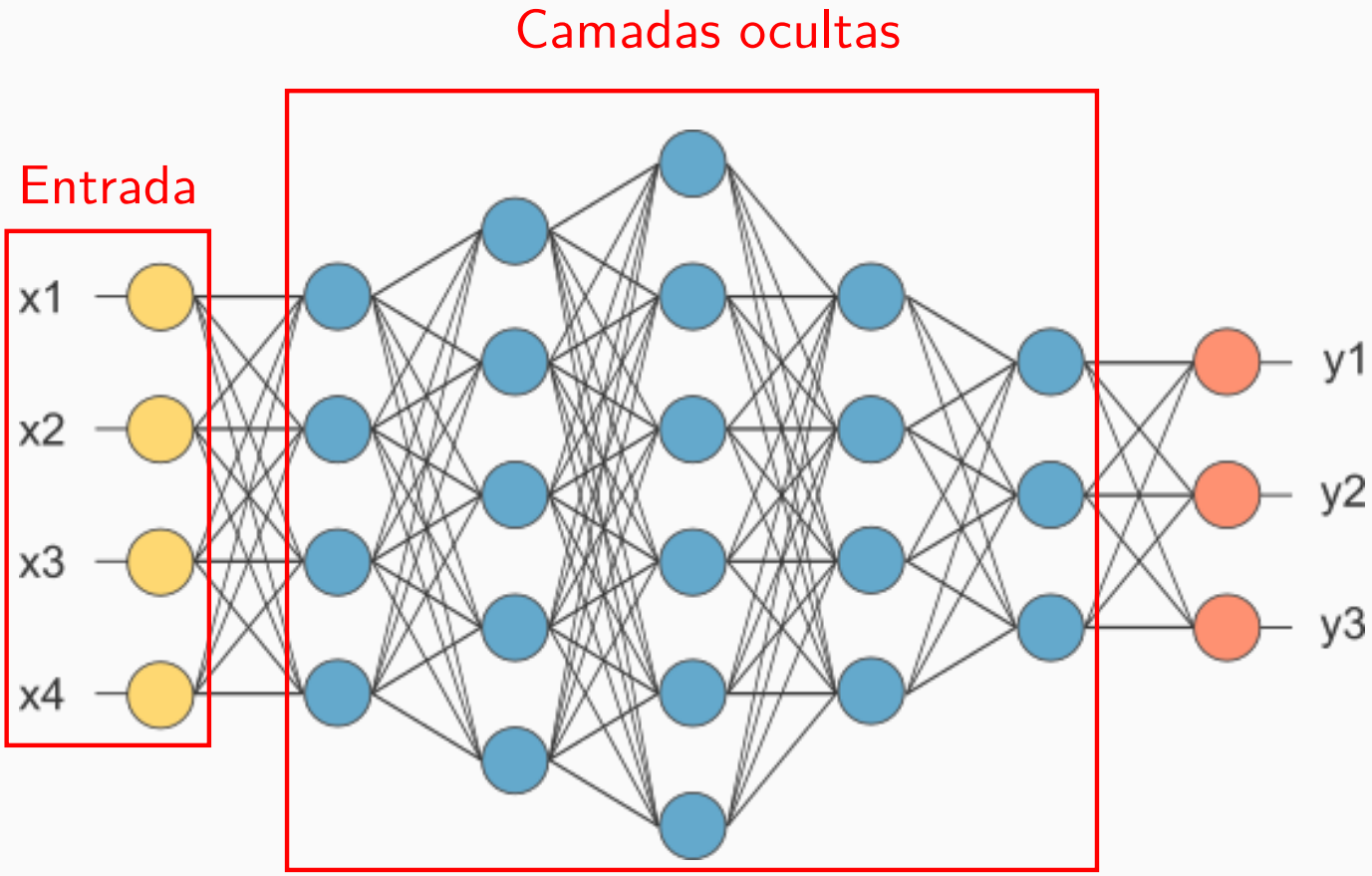
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

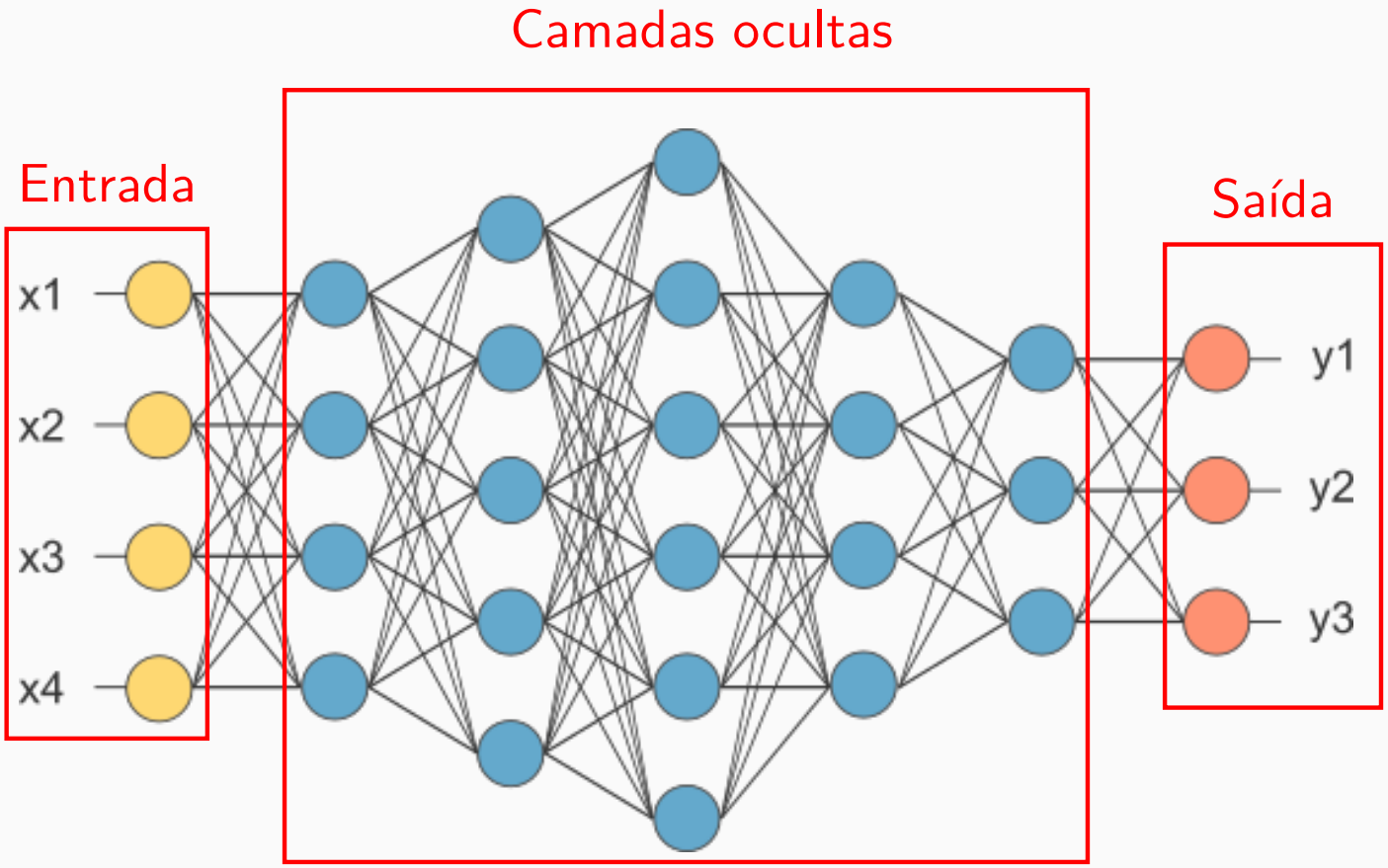
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

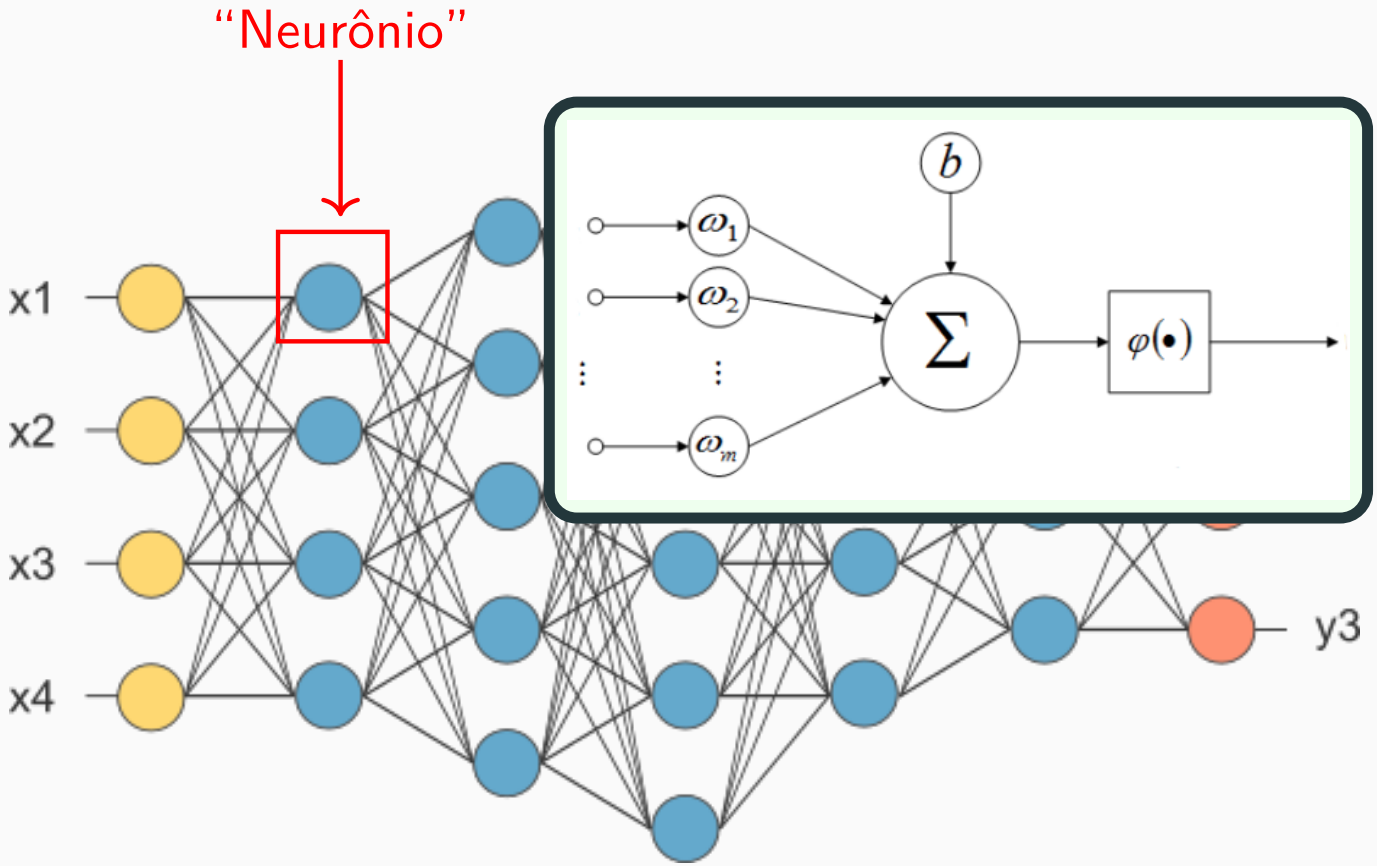
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

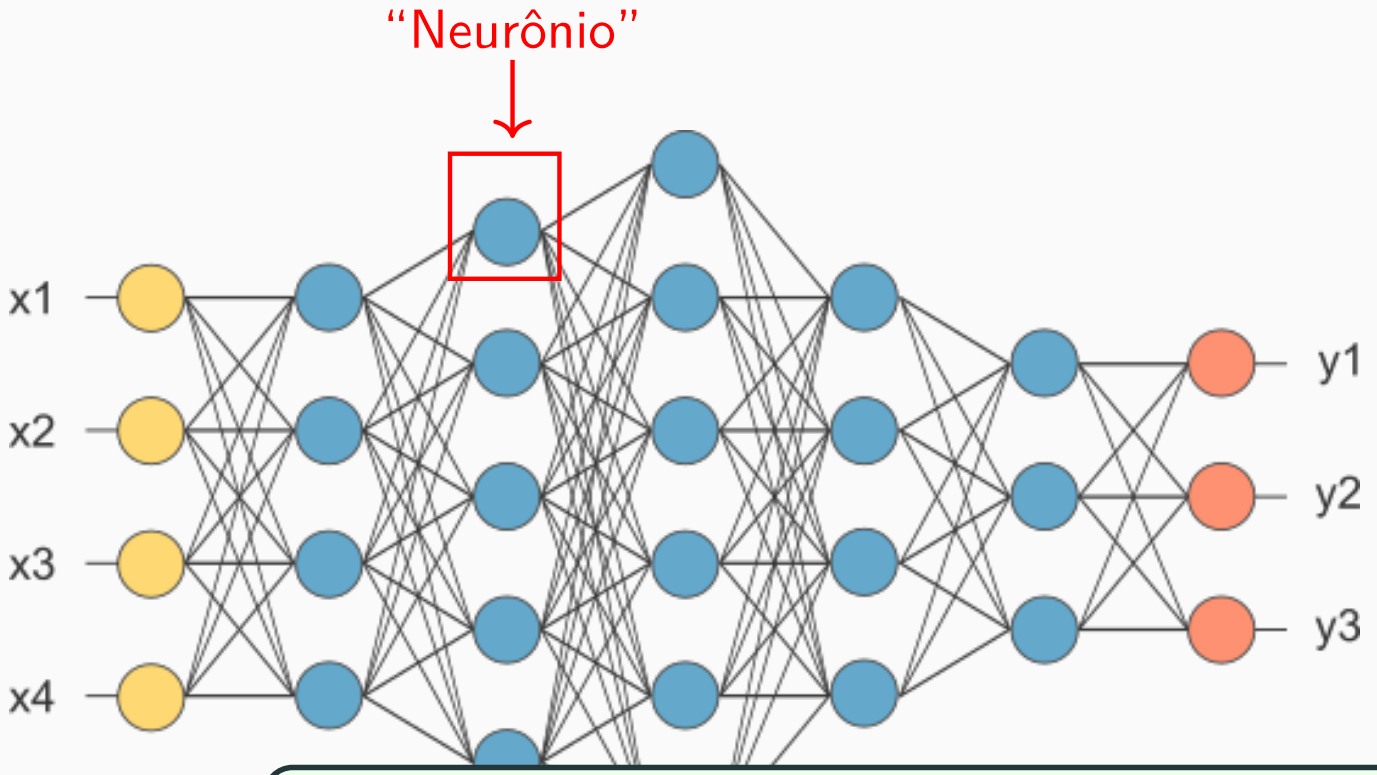
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:

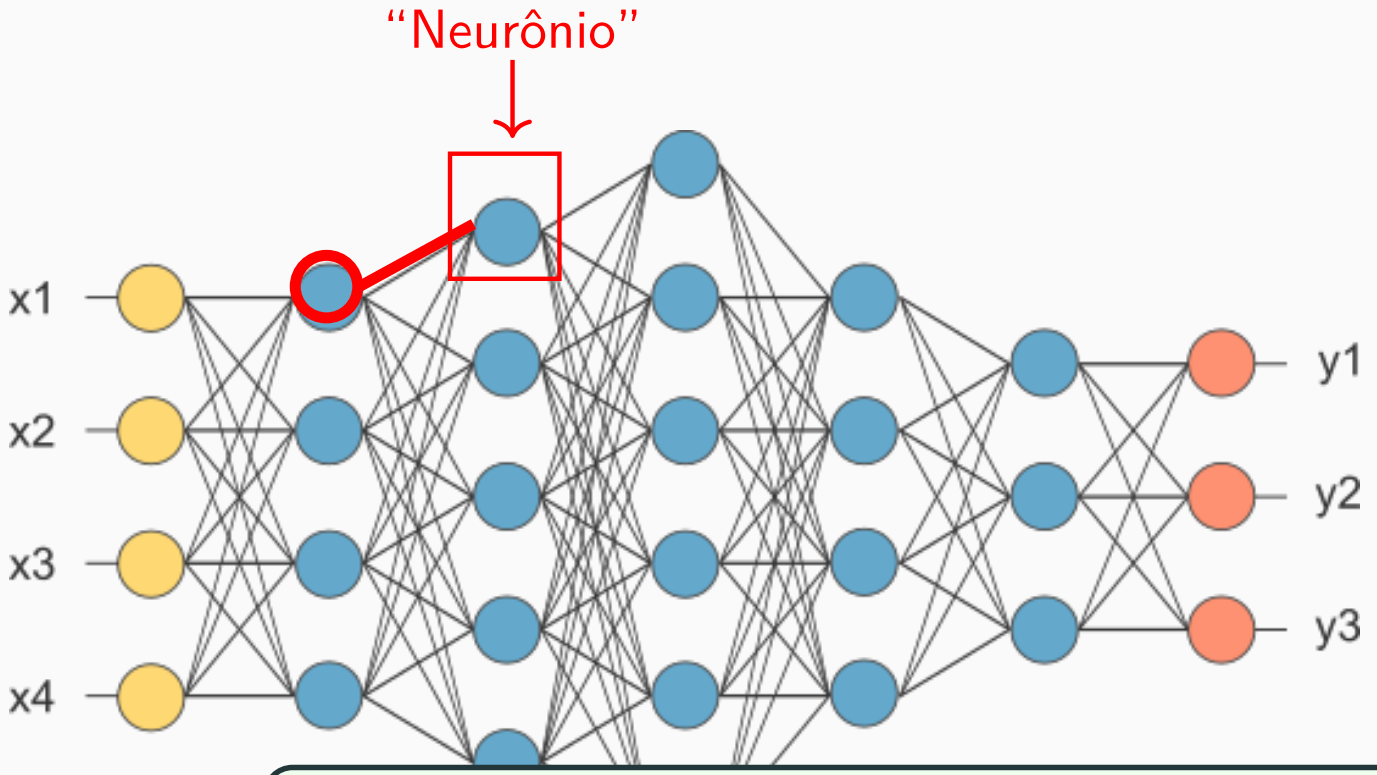


$$\varphi\Big(w_1\varphi(s_1) + w_2\varphi(s_2) + w_3\varphi(s_3) + w_4\varphi(s_4)\Big)$$

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:

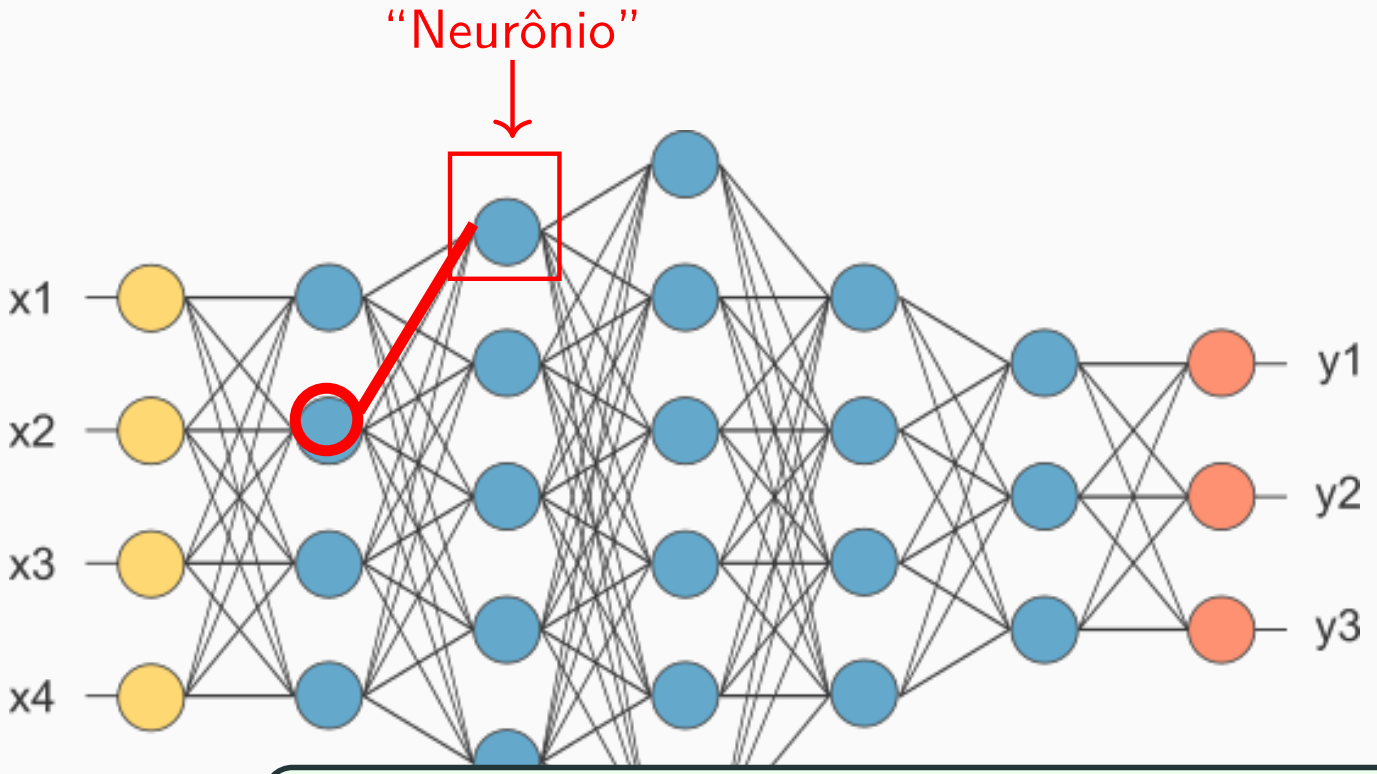


$$\varphi\Big(w_1\varphi(s_1) + w_2\varphi(s_2) + w_3\varphi(s_3) + w_4\varphi(s_4)\Big)$$

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:

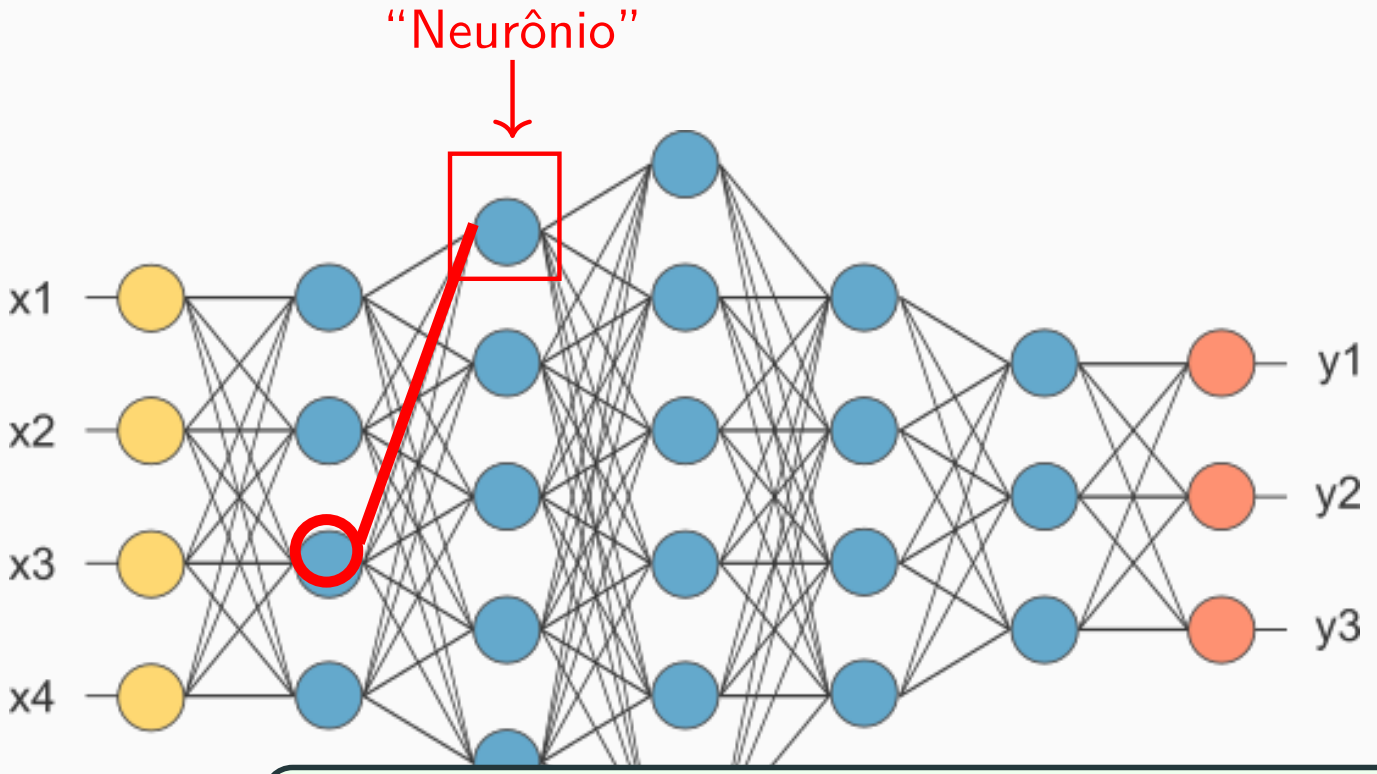


$$\varphi\Big(w_1\varphi(s_1) + w_2\varphi(s_2) + w_3\varphi(s_3) + w_4\varphi(s_4)\Big)$$

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:

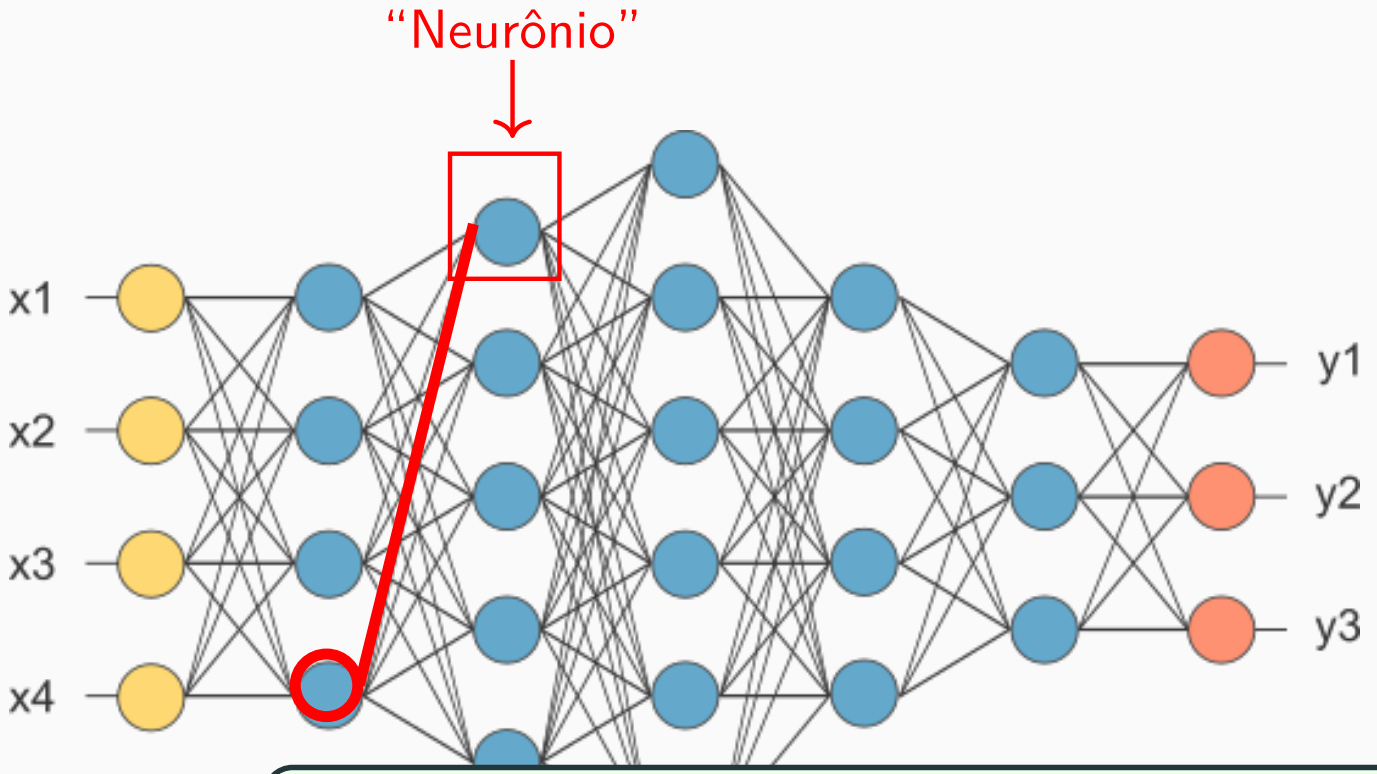


$$\varphi\Big(w_1\varphi(s_1) + w_2\varphi(s_2) + w_3\varphi(s_3) + w_4\varphi(s_4)\Big)$$

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:

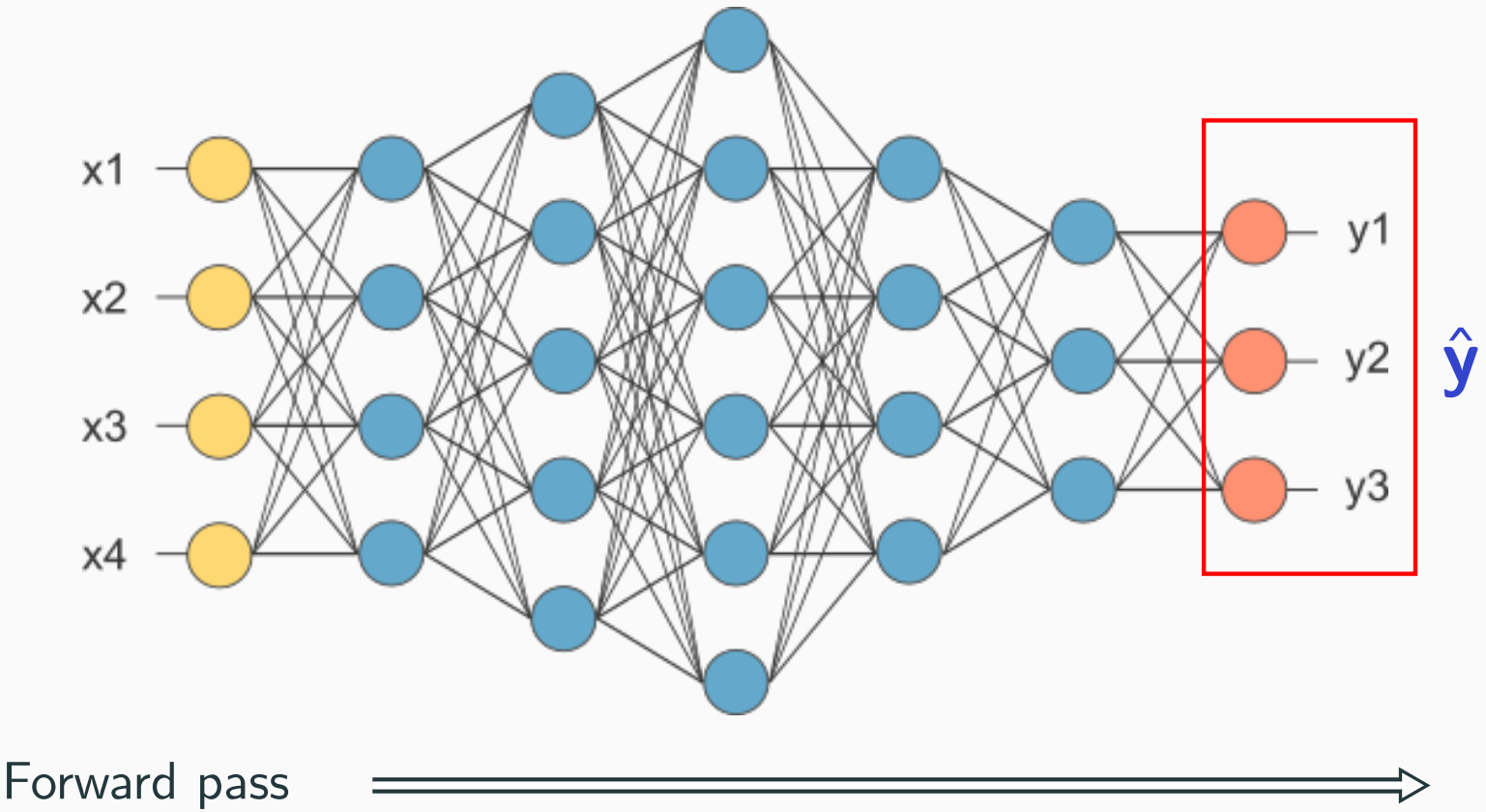


$$\varphi\Big(w_1\varphi(s_1) + w_2\varphi(s_2) + w_3\varphi(s_3) + w_4\varphi(s_4)\Big)$$

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

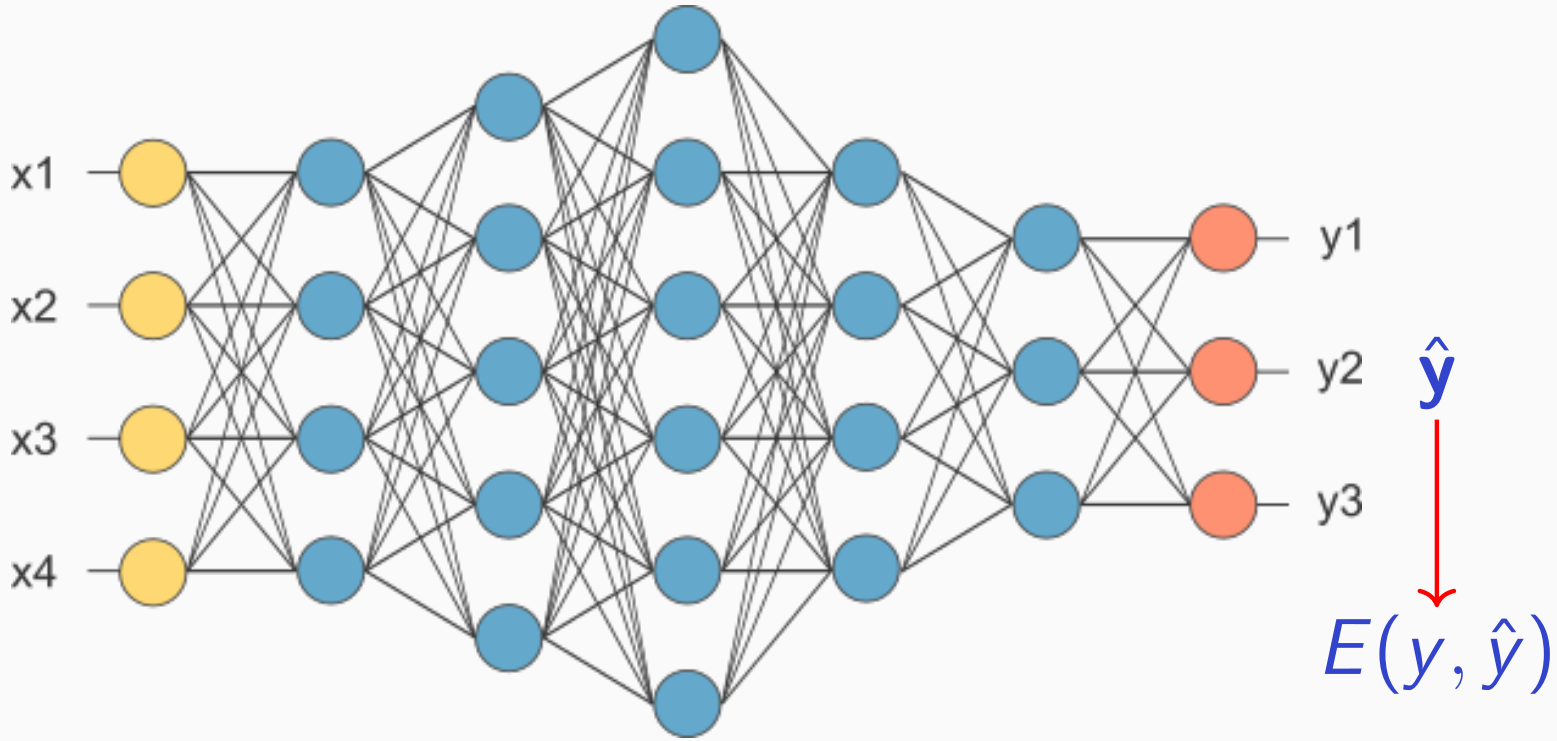
- Diagrama de uma rede neural típica:



Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Redes Neurais

- Diagrama de uma rede neural típica:



Backward pass ⟸

Fonte: https://www.kaggle.com/shokhan/neural-network-to-predict-dota-2-winner/comments

Treinamento de Redes Neurais

- Ideia geral:
  - **Saída esperada** (target): t_k
  - **Forward pass**: predições z_k
  - **Erro** entre saída esperada e predita: $e(t_k - z_k)$
  - A **função custo** depende de z_k
  - z_k, por sua vez, é uma **composição de funções**
  - (que indiretamente depende das entradas $x_1, \ldots, x_d$)
  - O **gradiente** da função custo com respeito aos pesos w pode ser calculado aplicando-se a regra da cadeia

Treinamento de Redes Neurais

- Na prática, precisamos pensar em:
  - Inicialização dos pesos
  - Taxa de Aprendizado
  - Número de iterações
  - Tamanho de batch (Stochastic Gradient Descent)
  - Arquitetura da rede (quantidade de camadas e de unidades por camada)

Próxima Aula: Grid Search e Seleção de Atributos