Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Aprendizado Profundo

Redes Neurais Artificiais

Thais Gaudencio

Tiago Maritan

Inspiração Biológica

- Redes de neurônios biológicos (cérebros) são compostos por cerca de **86 bilhões de neurônios** conectados a outros muitos neurônios.

- Estima-se que existem mais de **500 trilhões de conexões entre neurônios** no cérebro humano.

Inspiração Biológica

- Do ponto de vista biológico, um **neurônio biológico** é uma unidade excitável que pode **processar e transmitir informação** via sinais elétricos e químicos.

- O **neurônio** é considerado o **principal componente do sistema nervoso**.

Inspiração Biológica

- Existem duas principais propriedades das redes neurais artificiais que seguem a ideia geral de como o cérebro funciona:
  - 1) A unidade básica é o **neurônio artificial**.
  - 2) **Neurônios artificiais** são modelados como neurônios biológicos do cérebro;
  - 3) Como os neurônios biológicos, eles são **estimulados por entradas.**

Inspiração Biológica

- RNAs são um modelo computacional que compartilha algumas propriedades com o cérebro animal:
  - Contém **muitas unidades simples** trabalhando em paralelo;
  - Os **pesos** entre as unidades são os principais **meios de armazenamento a longo prazo de informações em redes neurais**.
  - A **atualização dos pesos** é a principal maneira de a rede neural **aprender novas informações**.

Redes Neurais Artificiais

$$\mathbf{x} \cdot \mathbf{W} = \mathbf{y}$$

- **x** = matriz que corresponde ao dado de entrada (matriz de vetores de xn - um vetor por objeto);
- **W** = pesos da rede que são aprendidos durante o treinamento;
- **y** = vetor que corresponde ao rótulo ou resultados para cada exemplo do treinamento.

Redes Neurais Artificiais

$$x \cdot W = y$$

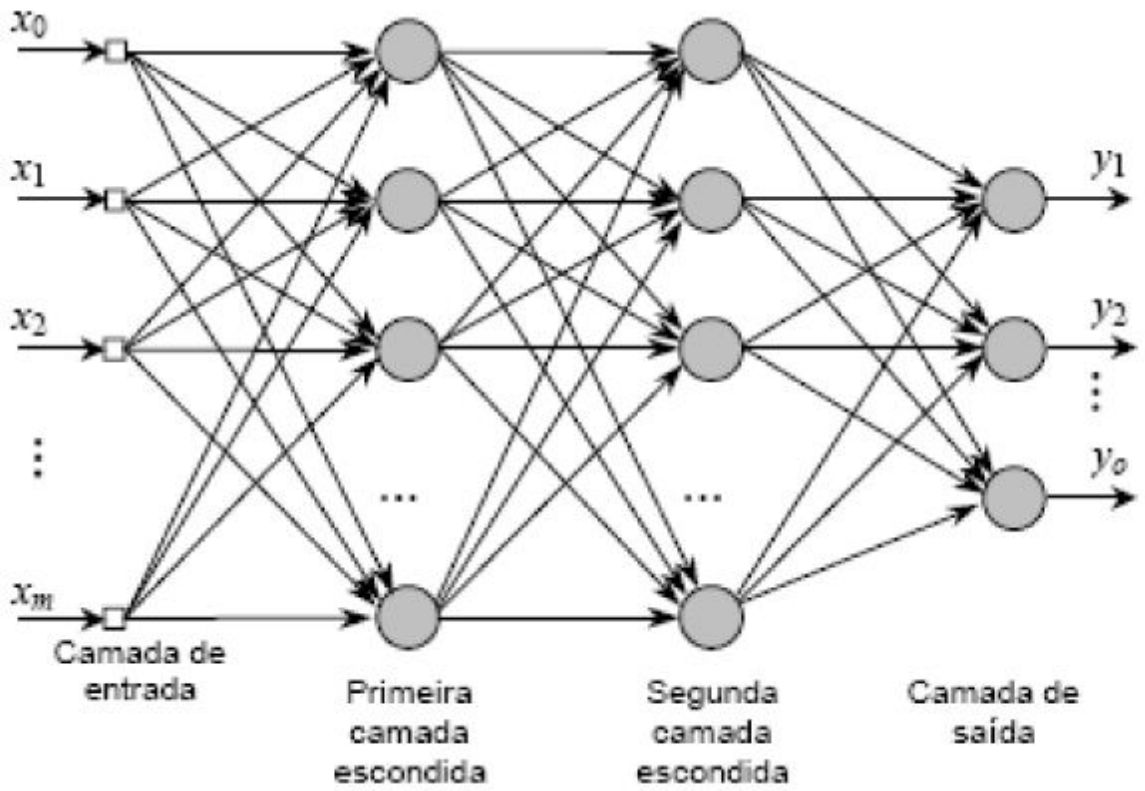
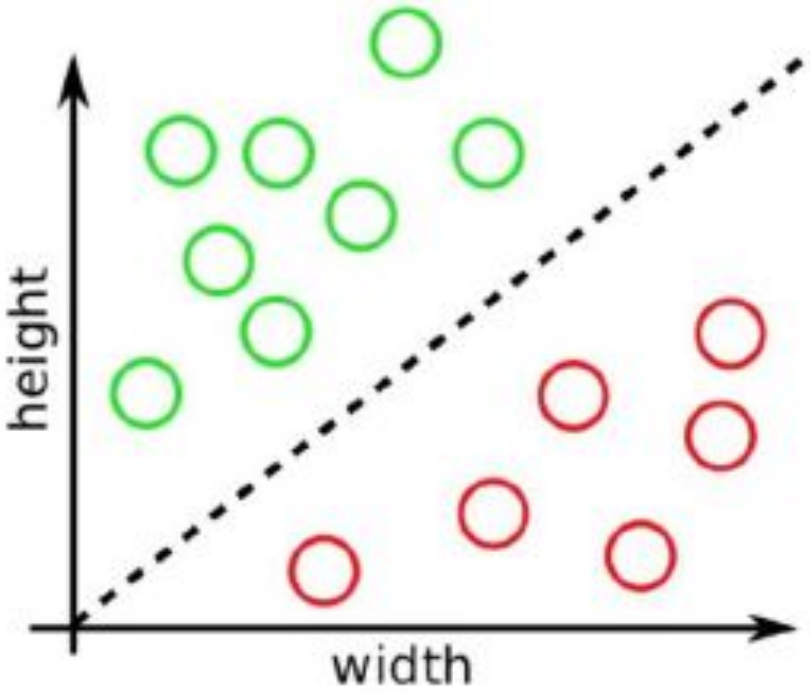


Figura 1 – Rede Neural Artificial Multicamadas.

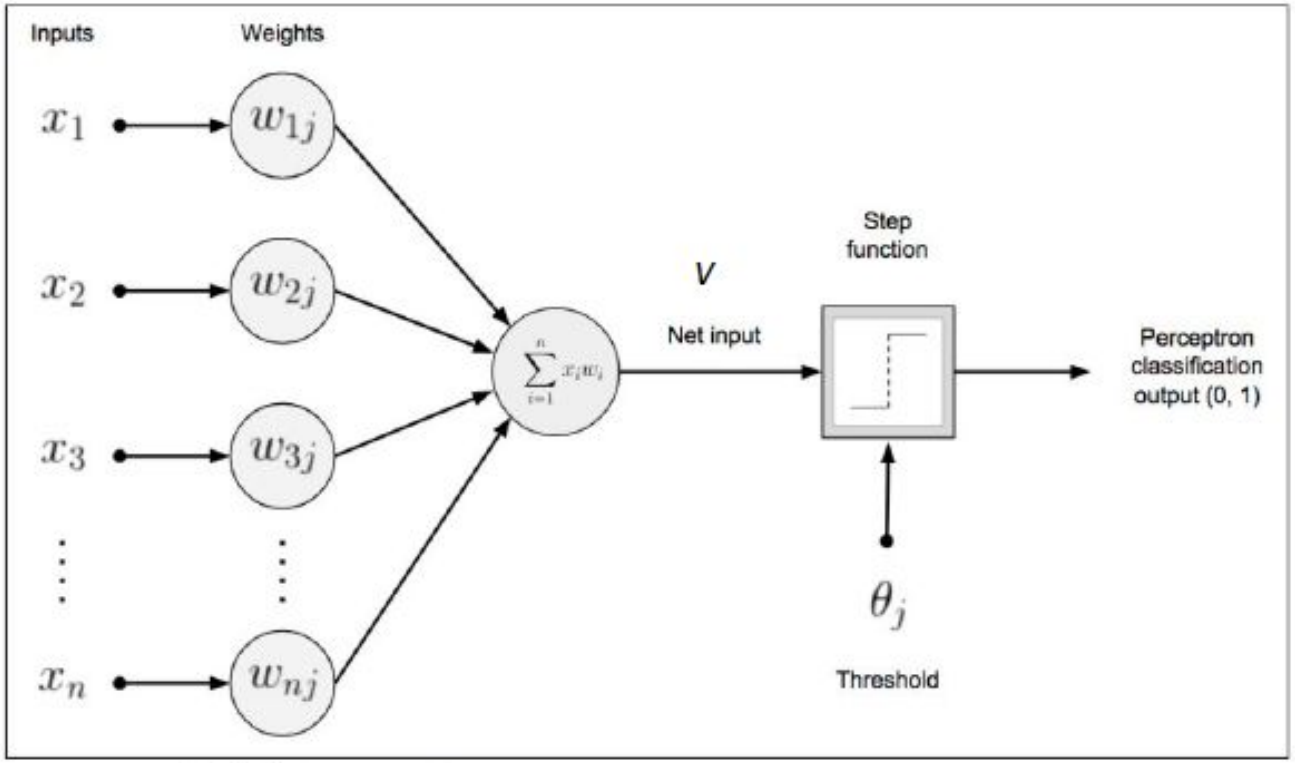
Redes Neurais

- Umas das primeiras redes neurais criadas foi o **Perceptron de Rosenblatt** (1958).
  - **Classificador binário linear**.



Perceptron

Função de ativação de Heaviside: $f(v) = \begin{cases} 0 & v < 0 \\ 1 & v >= 0 \end{cases}$



Single-layer perceptron

Perceptron

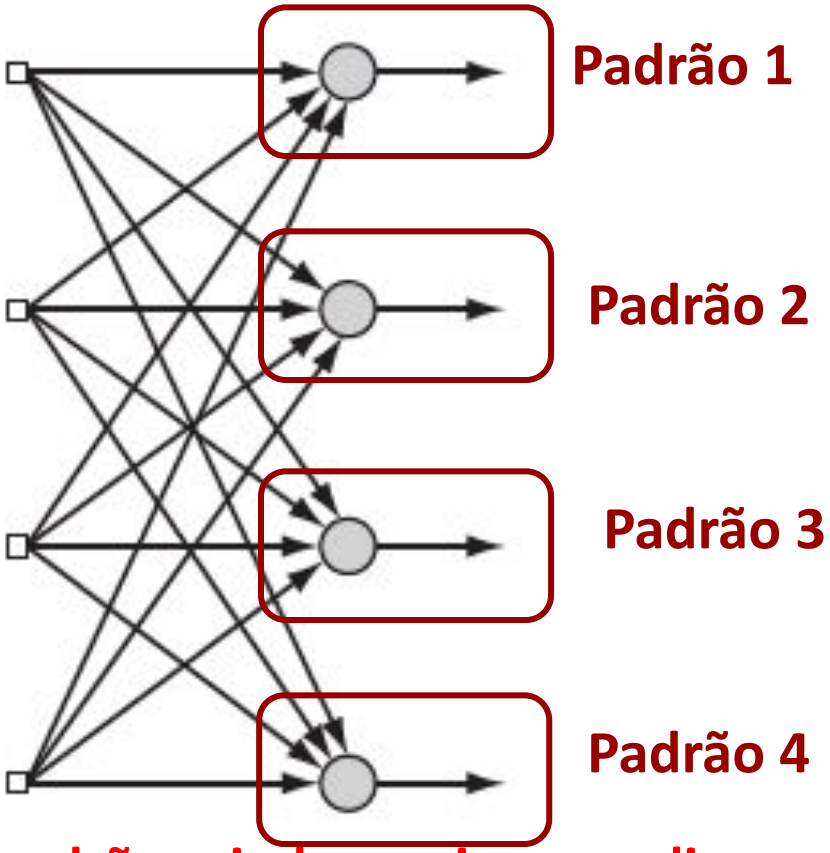
$$v = \sum_{i=1}^{n} x_i w_i$$

$$f(v) = \begin{cases} 0, & v < 0 \\ 1, & v \geq 0 \end{cases}$$

Ok, mas e se precisarmos classificar mais do que 2 padrões? Como resolver?

Perceptron de Camada Única

- Trabalhar com vários Perceptrons simultaneamente



Limitação: Os padrões ainda precisam ser linearmente separáveis;

E como funciona a Aprendizagem do Perceptron?

Algoritmo de Aprendizagem do Perceptron

- O algoritmo de aprendizagem do perceptron **altera os pesos no modelo** até que os registros de entrada estejam “corretamente classificados”.

- Geralmente os pesos são inicializados com pequenos valores aleatórios ou 0.0s no início do treinamento.

- O algoritmo de aprendizagem do perceptron considera cada registro de entrada e calcula a classificação de saída para verificar em relação a “classificação esperada” (rótulo).

Algoritmo de Aprendizagem do Perceptron

```
function rna-learning(training-examples)
    network <- initialize_weights(randomly)
    start loop
        for each example in training_examples do
            network_out = rna_output(network, example)
            example_err = actual_out - network_out
            Atualiza pesos usando a Regra Delta
        end for
    end loop quando todos os exemplos forem
    corretamente preditos ou critérios de parada
    forem atingidos
    return network
```

Algoritmo de Aprendizagem do Perceptron

- Para a correção do erro, os pesos da rede devem ser ajustados de forma a aproximar a saída real da saída desejada.

- De acordo com a **Regra Delta**, tal ajuste dependerá do:
  - Erro calculado => **e(n) = d(n) - y(n)**
  - Valor do estímulo de entrada que é "transmitido" pelo peso a ser ajustado => **x(n)**
  - Taxa de aprendizado => η
    - Cautela com que a curva de erros é percorrida

Regra Delta

- Para um dado exemplo de treinamento *n*:

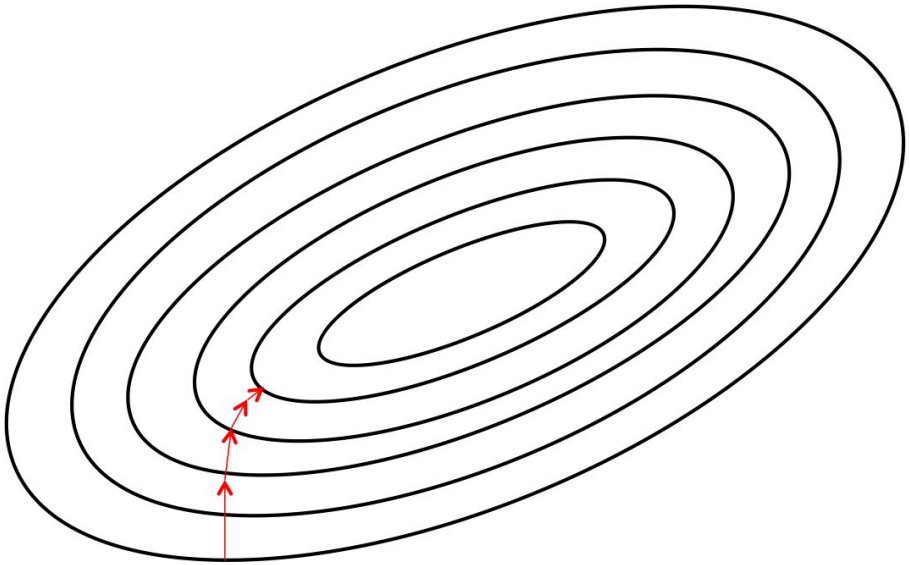
$$\Delta w(n) = \boldsymbol{\eta e(n) x(n)}$$

- O valor do peso atualizado será:

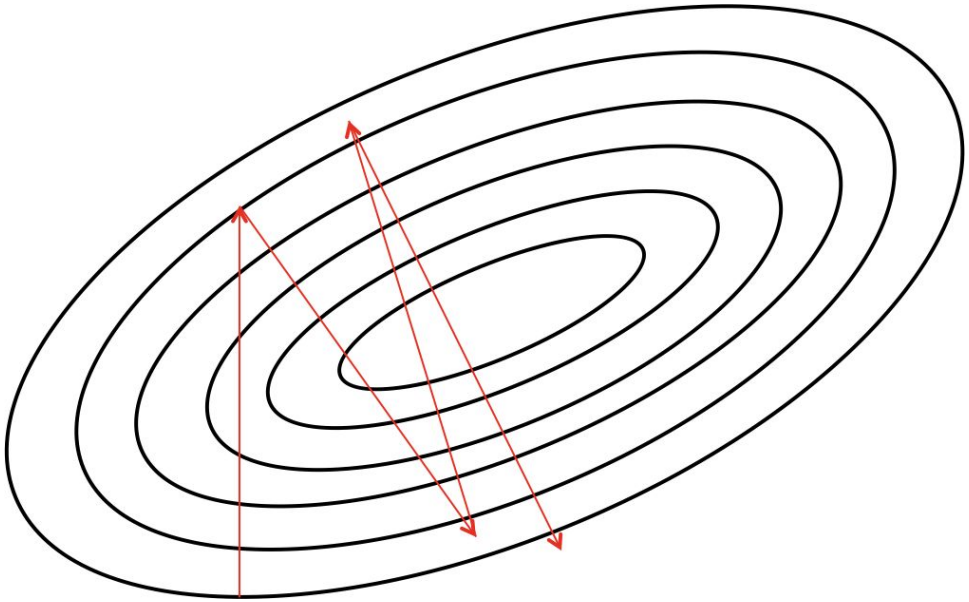
$$w(n+1) = w(n) + \Delta w(n)$$

Taxa de Aprendizagem (η)

η muito pequeno
Aprendizagem e convergência lenta



η muito grande
Rede instável



Redes Neurais de Múltiplas Camadas

Redes Neurais de Múltiplas Camadas

- Rede neural com **várias camadas de Neurônios**
  - **Permite resolver problemas não lineares e mais complexos**
  - **"Aproximadores de função universais";**

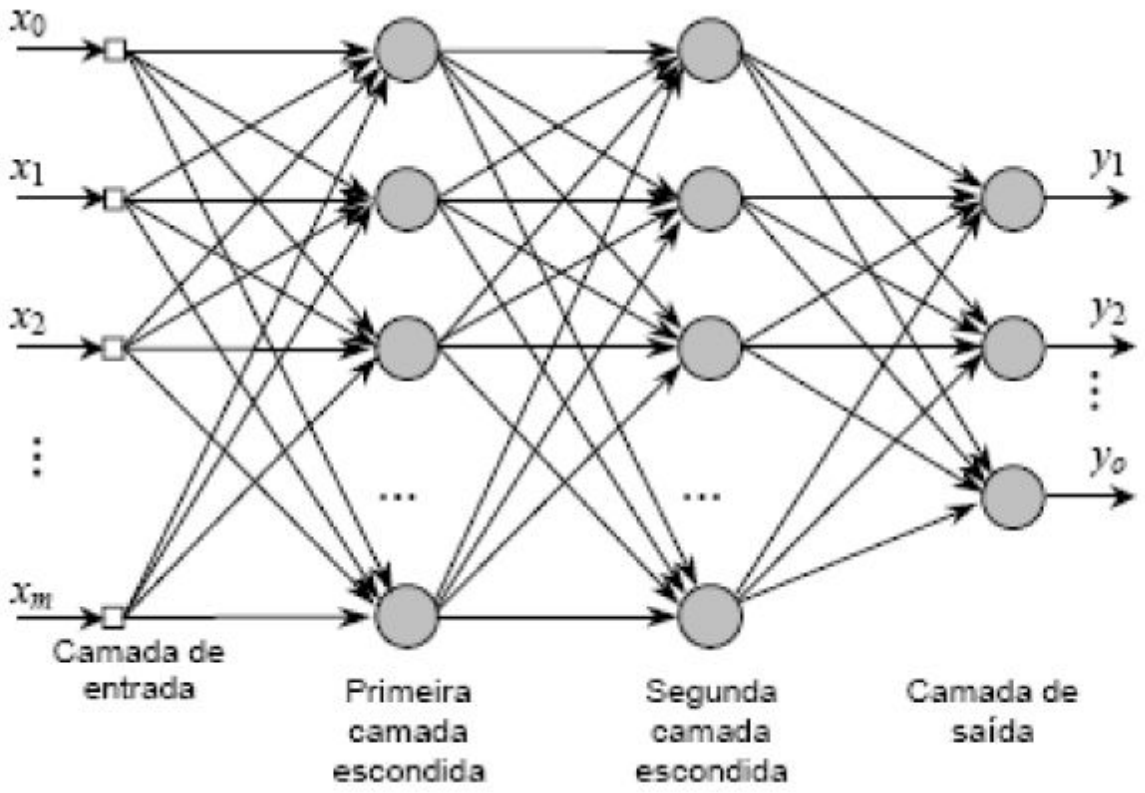


Figura 1 – Rede Neural Artificial Multicamadas.

x1

x2

w1 = 0.8 e w2 = 1.1, w0 = 5

v = x1w1 + x2w2 = **0.8x1 + 1.1x2 + 5**

f(v) = 0 ou 1

Redes Neurais de Múltiplas Camadas

- Rede neural com **várias camadas de Neurônios**

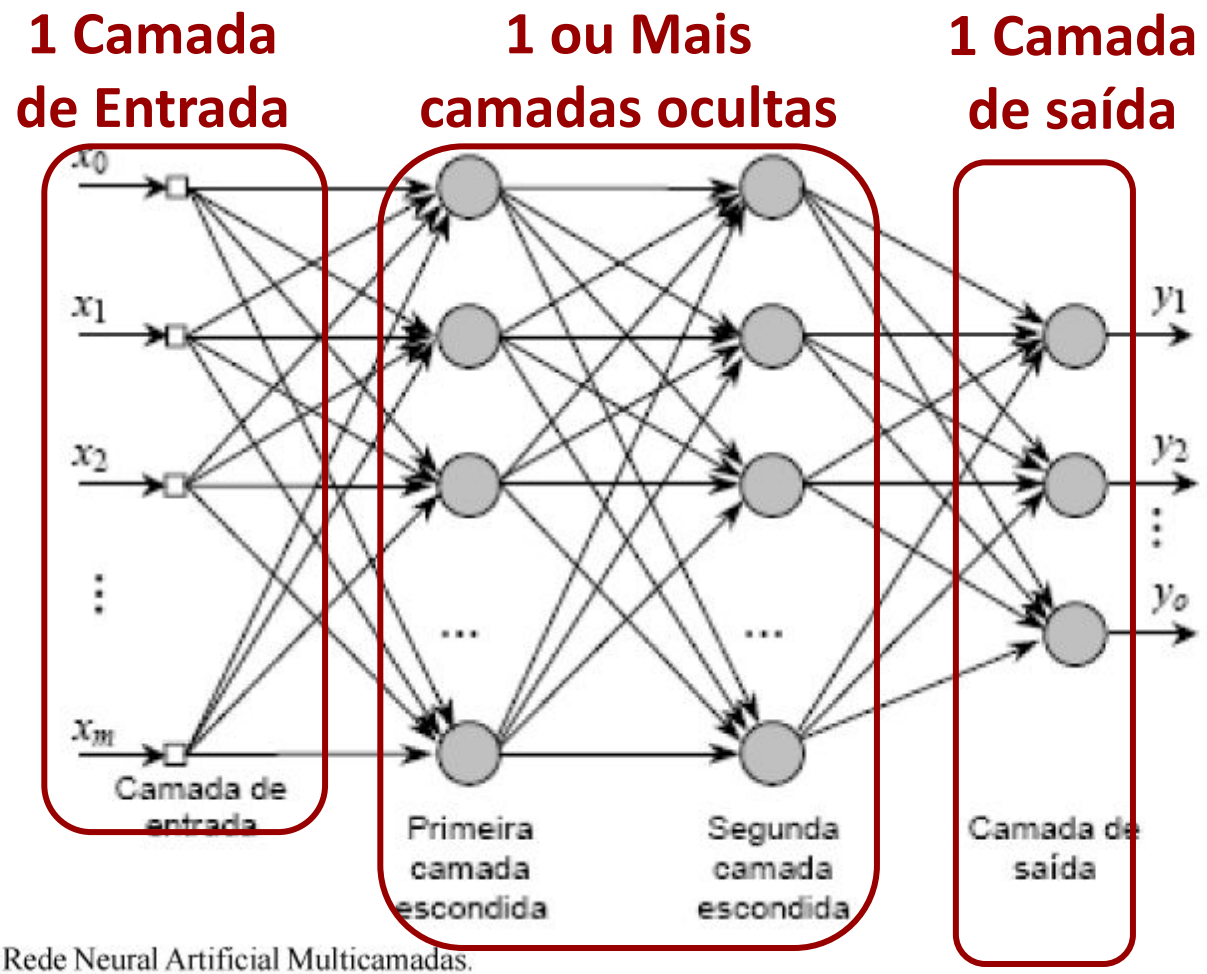
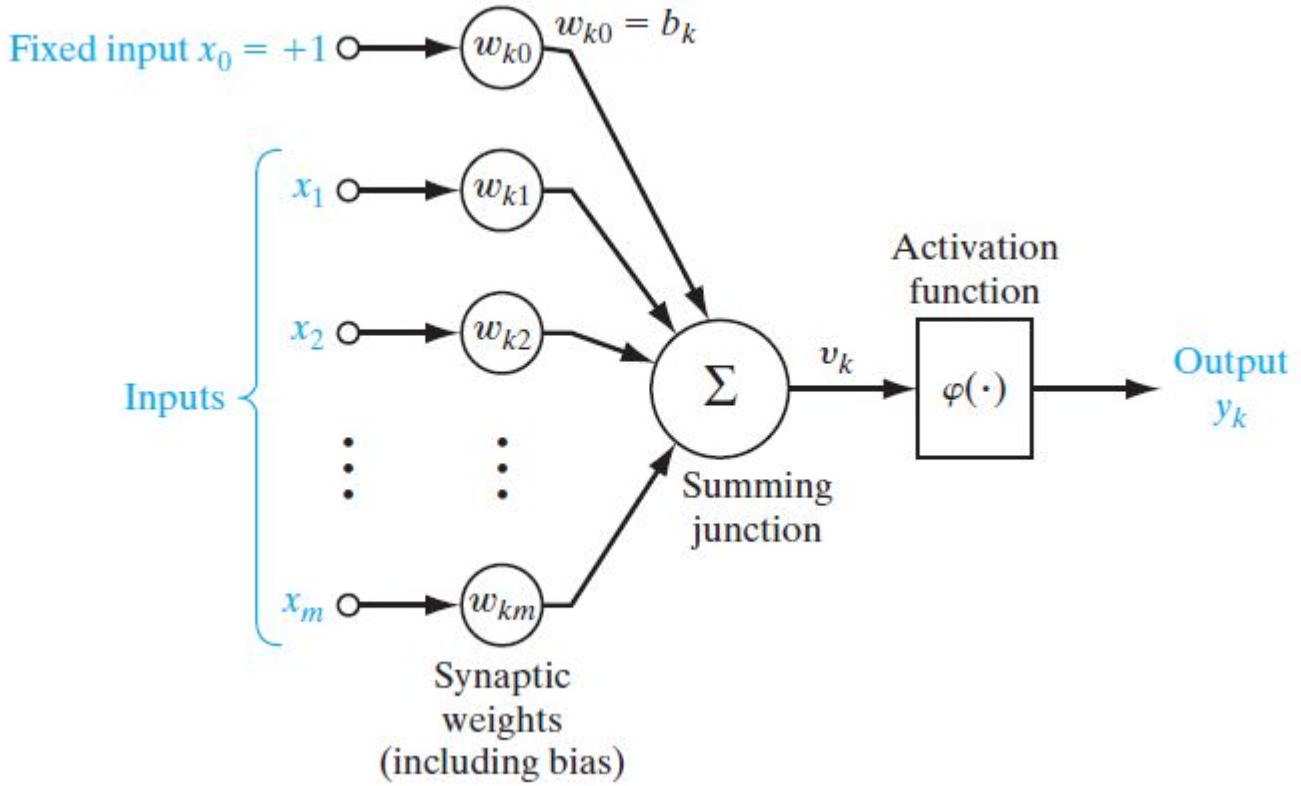


Figura 1 – Rede Neural Artificial Multicamadas.

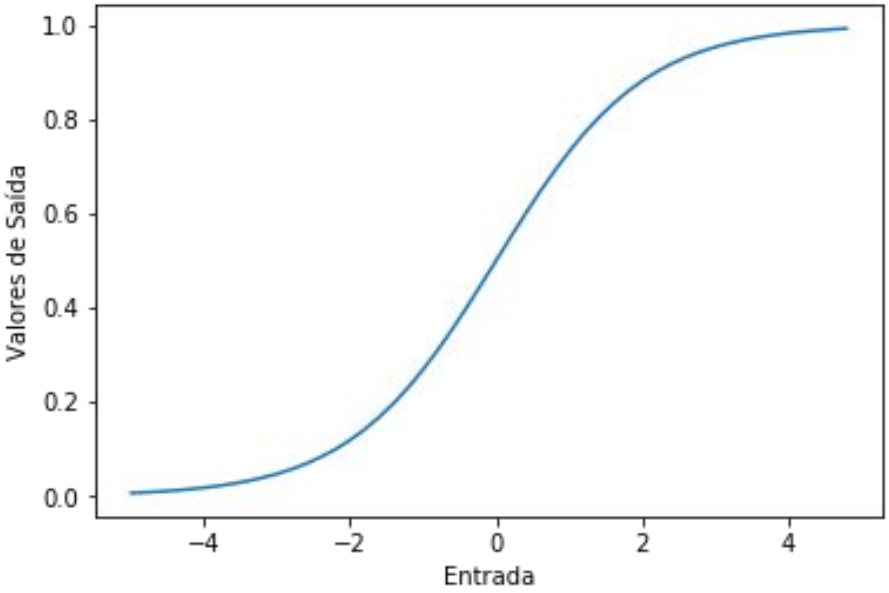
Redes Neurais de Múltiplas Camadas

- Neurônios artificiais são semelhantes ao Perceptron, mas têm diferentes **funções de ativação**



Funções de Ativação

- Sigmóide (Logística):



Função:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

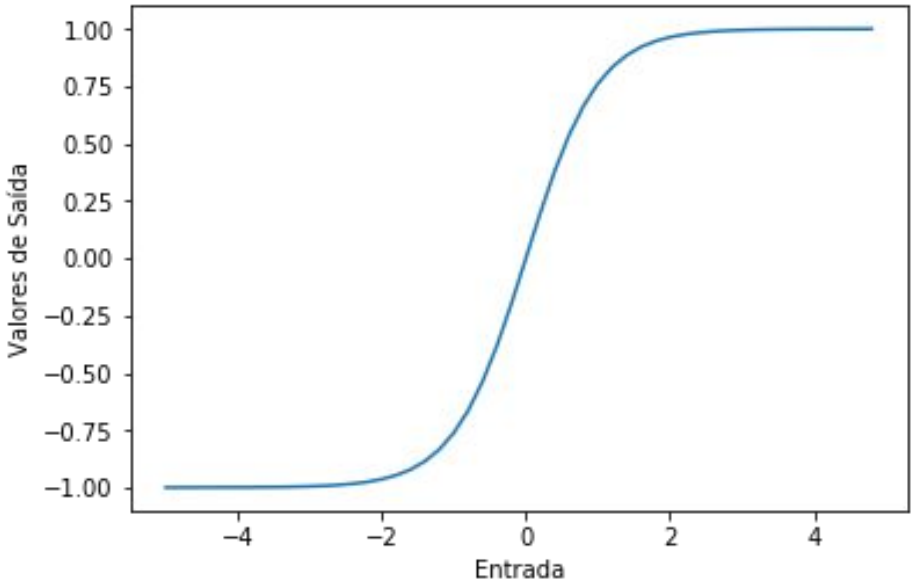
Derivada da Função:

$$\varphi'(x) = \varphi(x)(1 - \varphi(x))$$

Gera valores entre 0 e 1

Funções de Ativação

- Tangente Hiperbólica:



Função:

$$\varphi(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

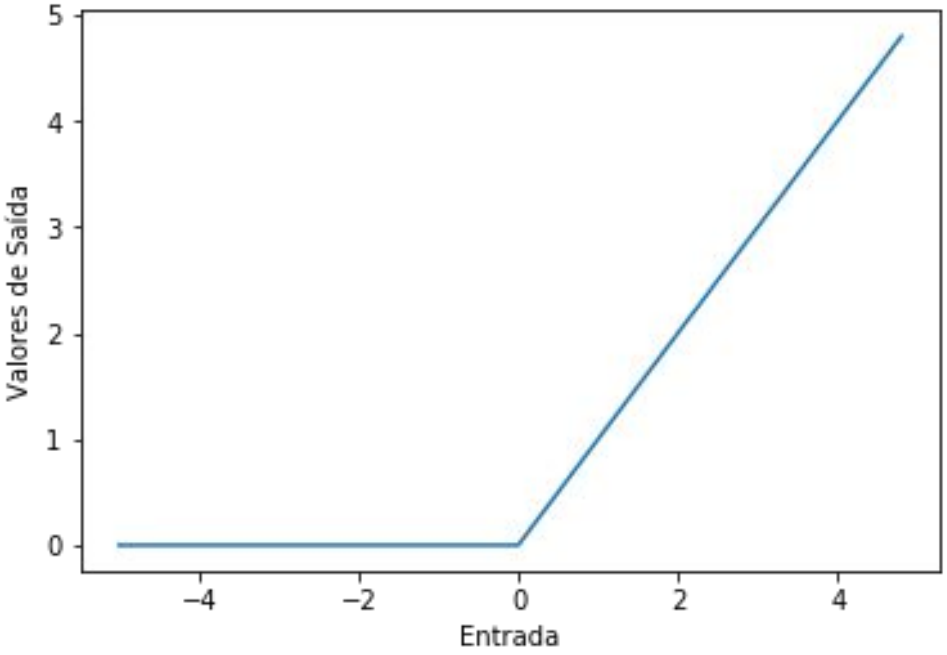
Derivada da Função:

$$\varphi'(x) = \frac{1}{2}(1 - \varphi^2(x))$$

Gera valores entre -1 e 1

Funções de Ativação

- ReLU:



Função:

$$\varphi(x) = \max(0, x)$$

Derivada da Função:

$$\varphi'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & c.c \end{cases}$$

Obs: Maior parte das redes DL usam ReLU nas camadas ocultas, porque o treinamento fica mais rápido, e atenua o problema do desaparecimento do gradiente (gradiente ≃0)

Redes Neurais de Multiplas Camadas

- **Teorema de Aproximação Universal:**
  - Uma rede neural com **uma única camada oculta** que contém um número finito de neurônios pode aproximar funções contínuas em subconjuntos compactos de R^n, com pressupostos mínimos de função de ativação

  - Em 1989, George Cybenko provou uma das primeiras versões do teorema para funções de ativação sigmóide.

    "*This combination of results demonstrates that any continuous function can be uniformly approximated by a continuous neural network having only one internal, hidden layer and with an arbitrary continuous sigmoidal nonlinearity (Theorem 2).*"

Cybenko, G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314

Teorema da Aproximação Universal

- Suponha que $\varphi(.)$ seja uma função contínua não constante, limitada e monotonamente crescente;
- Suponha que I_{m0} represente um hipercubo unitário $[0,1]^{m0}$ de dimensão m_0
- O espaço de funções contínuas em I_{m0} é representado por $C(I_{m0})$.
- Dada qualquer função $f \ni C(I_{m0})$ e $\epsilon > 0$, existe um conjunto de constantes α_i, b_i e w_{ij}, onde $i = 1,..., m_1$, e $j = 1,..., m_0$, tal que podemos definir

$$F(x_1, \dots, x_{m0}) = \sum_{i=1}^{m_1} \alpha_i \, \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right)$$

- Como uma **realização aproximada da função *f(.)***, isto é:

$$|F(x_1, \dots, x_{m0}) - f(x_1, \dots, x_{m0})| < \epsilon$$

Teorema da Aproximação Universal

- Suponha que ***φ(.)*** seja uma função contínua não constante, limitada e monotonamente crescente;

Função de ativação sigmóide atende a esses critérios

Representa a saída de um perceptron de múltiplas camadas com m_0 entradas, uma camada oculta contendo m_1 neurônios

$$F(x_1, \ldots, x_{m0}) = \sum_{i=1}^{m_1} \alpha_i \, \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right)$$

Neurônio oculto *i* tem pesos $w_{i1},\ldots,w_{m0}$, e bias b_i

$\alpha_i,\ldots,\alpha_{m1}$ são os pesos sinápticos da camada de saída.

Teorema da Aproximação Universal

- Resumindo**: <u>uma única camada oculta</u>** é suficiente para um **<u>Perceptron de Múltiplas Camadas (MLP)</u>** computar uma aproximação **ϵ** uniforme para:
  - Um conjunto de treinamento representado da seguinte forma:
    - Entrada: $x_1,...,x_{m0}$; Saída desejada: $f(x_1,...,x_{m0})$

- Contudo, teorema não diz que a única camada oculta é ótima no sentido de tempo de aprendizagem, facilidade de implementação, ou generalização.

Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Aprendizado Profundo

Redes Neurais Artificiais

Thais Gaudencio

Tiago Maritan