

# Aprendizagem de Máquina

## Redes Neurais

Telmo de Menezes e Silva Filho

[tmfilho@gmail.com](mailto:tmfilho@gmail.com)/[telmo@de.ufpb.br](mailto:telmo@de.ufpb.br)

[www.de.ufpb.br](http://www.de.ufpb.br)

**UFPB**

 Departamento de  
**ESTATÍSTICA**

# Sumário

Inspiração Biológica

Histórico

Perceptron

Rede Multicamada Feed-Forward

Backpropagation

Para Terminar



# 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
- ▶ Mesmo as maiores redes neurais artificiais de hoje nem chegam perto de se aproximar desse número

# 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

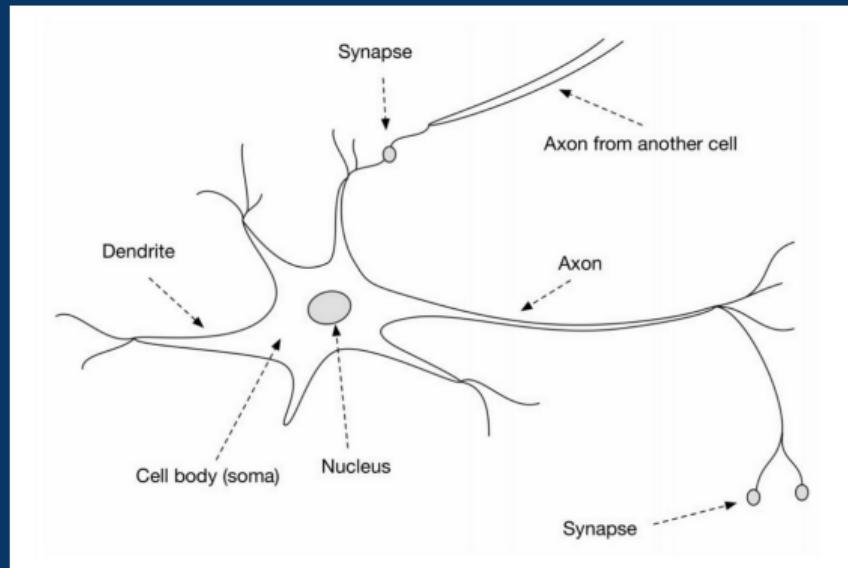
- Existem duas principais propriedades das redes neurais artificiais que seguem a ideia geral de como o cérebro funciona:
  1. A unidade mais básica de uma rede neural é o neurônio artificial
  2. Neurônios artificiais são modelados como neurônios biológicos do cérebro, e como os neurônios biológicos,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



# Inspiração Biológica

- ▶ As redes neurais são um modelo computacional que compartilha algumas propriedades com o cérebro animal em que muitas unidades simples estão trabalhando em paralelo com nenhuma unidade de controle centralizadora
- ▶ 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

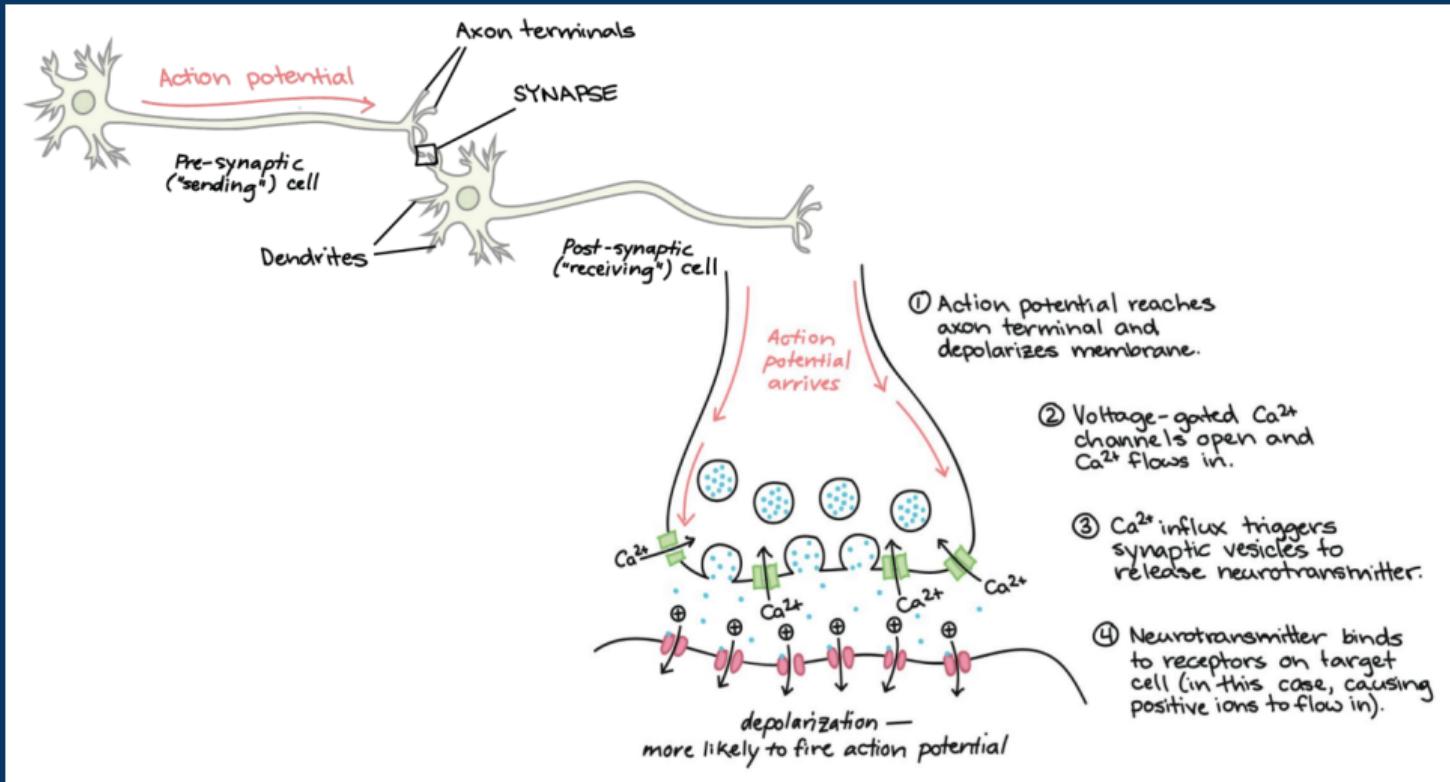
# Neurônio Biológico



- ▶ O neurônio é constituído por um soma (corpo celular) que tem muitos dendritos, mas apenas um axônio
- ▶ O axônio único pode se ramificar várias vezes
- ▶ Os dendritos são estruturas finas que surgem do corpo celular principal
- ▶ Os axônios são fibras nervosas com uma extensão celular especial que vem do corpo da célula



# Sinapse



# Neurônio Biológico

- ▶ As **sinapses** são a junção de conexão entre axônio e dendrito. A maioria das sinapses enviam sinais do axônio de um neurônio para o dendrito de outro neurônio
- ▶ Os **dendritos** têm fibras que se ramificam do soma em uma rede espessa ao redor da célula nervosa
  - ▶ Dendritos permitem que a célula receba sinais de neurônios vizinhos conectados

# Neurônio Biológico

- ▶ Os **axônios** são as únicas fibras longas que se estendem a partir do soma principal
  - ▶ Estendem-se distâncias mais longas do que os dendritos e medem geralmente 1 centímetro de comprimento (100 vezes o diâmetro do soma)
- ▶ Os neurônios são capazes de enviar pulsos eletroquímicos através de mudanças de tensão entre membranas gerando potencial de ação
  - ▶ Este sinal viaja ao longo do axônio da célula e ativa conexões sinápticas com outros neurônios



# Neurônio Biológico

- ▶ As sinapses que aumentam o potencial são consideradas excitatórias, e as que diminuem o potencial são consideradas inibitórias
- ▶ Os neurônios também podem formar novas conexões ao longo do tempo e até mesmo migrar
- ▶ Esses mecanismos combinados a mudança de conexão conduzem o processo de aprendizagem no cérebro biológico

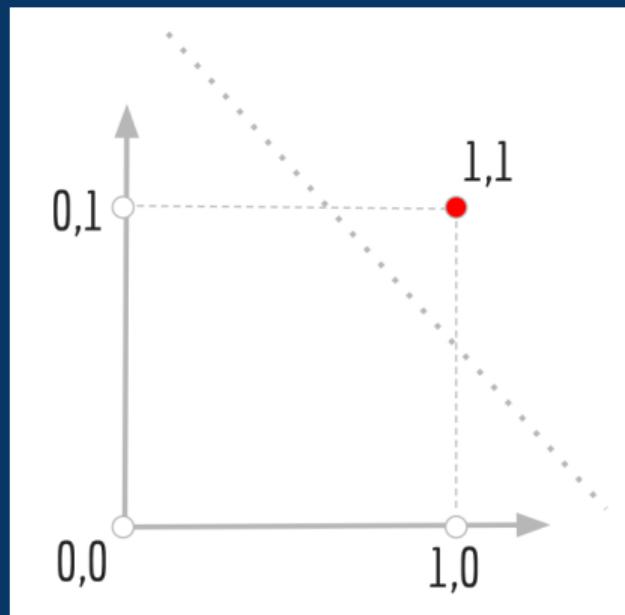


# Histórico



# Histórico

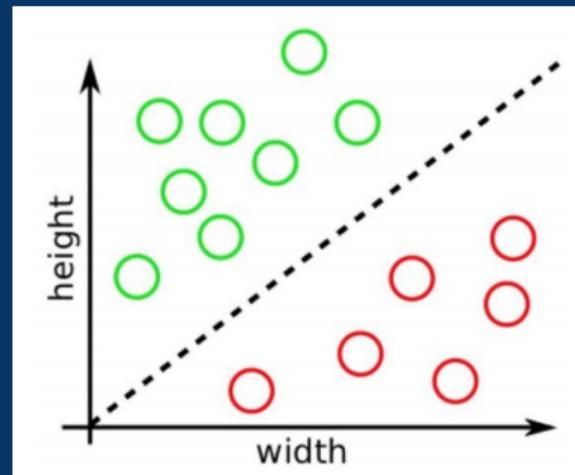
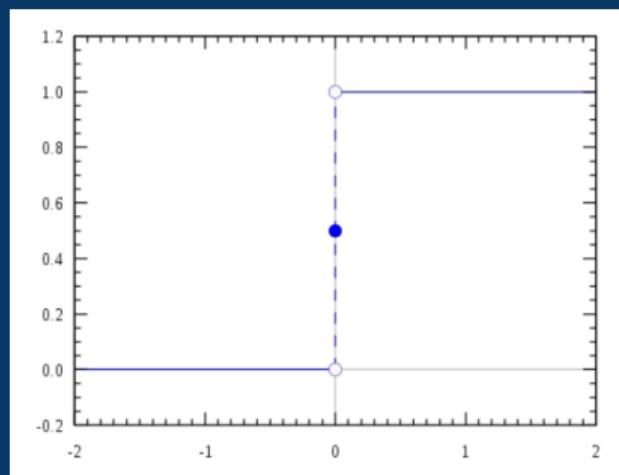
- O primeiro modelo de neurônio artificial foi a Unidade Lógica do Limiar (TLU) proposta por McCulloch e Pitts em 1943, que poderia aprender a função lógica AND e OR



$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

# Histórico

- ▶ O perceptron foi inventado em 1957 no Laboratório Aeronáutico Cornell por Frank Rosenblatt
- ▶ O perceptron é um modelo linear usado para classificação binária
- ▶ No campo das redes neurais, o perceptron é considerado um neurônio artificial usando o passo de Heaviside como função de ativação.



# Histórico

- ▶ O Mark I Perceptron foi projetado para reconhecimento de imagem para fins militares pela marinha dos EUA
- ▶ O Mark I Perceptron tinha 400 fotocélulas ligadas a neurônios artificiais na máquina e os pesos foram implementados por potenciômetros
- ▶ As atualizações de peso foram fisicamente executado por motores elétricos

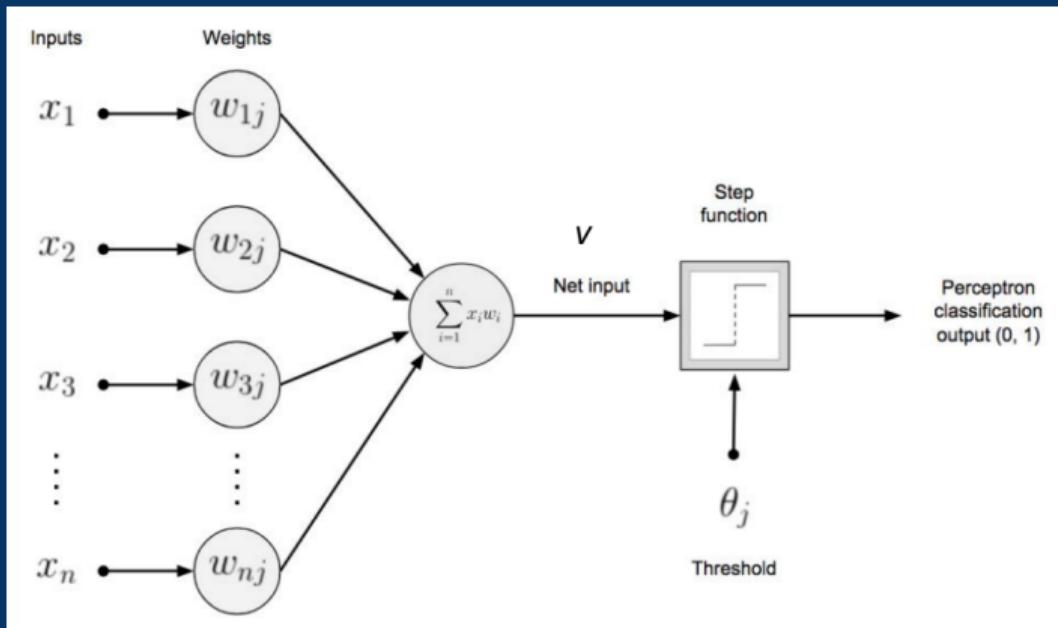
# Perceptron



# Perceptron

- ▶ O perceptron é um classificador binário linear com uma relação de entrada / saída simples
- ▶ Estamos somando  $n$  entradas vezes seus pesos associados e, em seguida, enviando essa “entrada de rede” para uma função de ativação com um limiar definido
- ▶ Tipicamente com perceptrons, esta é uma função de ativação de Heaviside com um valor limiar de 0,5
- ▶ Esta função produzirá um valor real binário simples (0 ou 1), dependendo da entrada

# Perceptron



# Perceptron

- ▶ O algoritmo de aprendizagem do perceptron muda os pesos no modelo do perceptron até que todos os registros de entrada estejam corretamente classificados
- ▶ O algoritmo não terminará se a entrada de aprendizagem não é linearmente separável
- ▶ O algoritmo de aprendizagem do perceptron inicializa o vetor de peso 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 atual



# Algoritmo de Aprendizagem do Perceptron

- ▶ Se a classificação for correta, nenhuma alteração de peso será feita
- ▶ Se a classificação for incorreta, os pesos são ajustados
- ▶ Os pesos são atualizados para os exemplos de treinamento em uma forma de “aprendizagem online”
- ▶ Este ciclo continua até que todos os exemplos de entrada estejam corretamente classificados
- ▶ Se o conjunto de dados não for linearmente separável, o algoritmo de treinamento não terminará



# A Função XOR

Um perceptron básico (de camada única) não pode resolver o problema de lógica XOR, ilustrando uma limitação inicial do modelo perceptron.

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

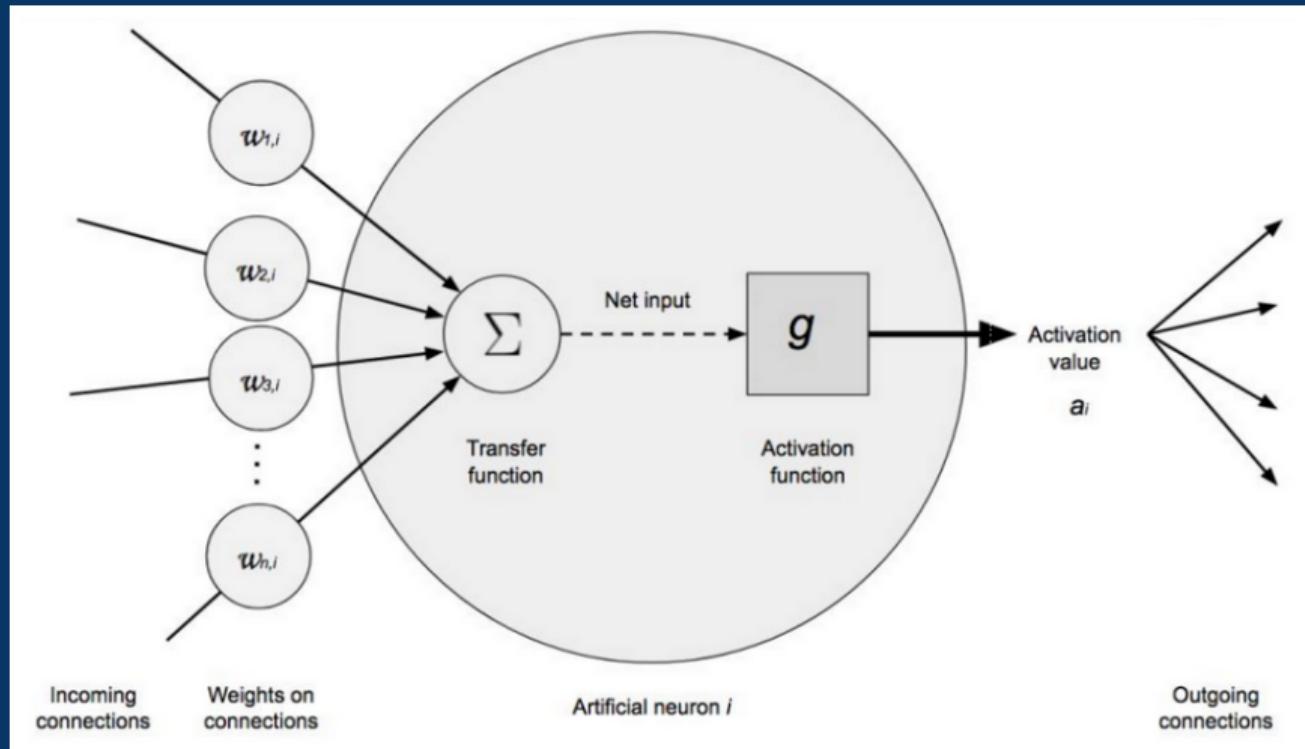


# Rede Multicamada Feed-Forward

# Rede Multicamada Feed-Forward

- ▶ Uma rede neural com uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída
- ▶ Cada camada tem um ou mais neurônios
- ▶ Estes neurônios artificiais são semelhantes aos seus precursores de perceptron, mas têm diferentes funções de ativação dependendo do propósito específico da camada na rede

# Rede Multicamada Feed-Forward



# Pesos da Rede Neural

- ▶ Os pesos nas conexões em uma rede neural são coeficientes que escalam (amplificam ou minimizam) o sinal de entrada para um determinado neurônio na rede
- ▶ Nas representações comuns das redes neurais, estas são as linhas / setas que vão de um ponto a outro
- ▶ As conexões são comumente denotadas como  $w$  em representações matemáticas de redes neurais



# Viés (Bias)

- ▶ O bias corresponde a valores escalares adicionados à entrada para garantir que pelo menos alguns nós por camada sejam ativados independentemente da força do sinal
- ▶ O bias permite que a aprendizagem aconteça em caso de sinal baixo
  - ▶ Permite que a rede tente novas interpretações ou comportamentos
- ▶ O bias geralmente é representado como  $b$  e, como os pesos, é modificado ao longo do processo de aprendizagem



# Função de Ativação

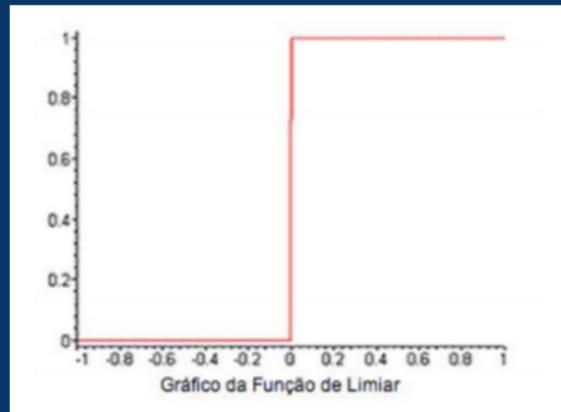
- ▶ Restringe a amplitude da saída de um neurônio
- ▶ É também referida como função restritiva
- ▶ Limita o intervalo permissível de amplitude do sinal de saída a um valor finito
- ▶ O intervalo normalizado da amplitude de saída de um neurônio é escrito como o intervalo unitário fechado  $[0, 1]$  ou alternativamente  $[-1, 1]$
- ▶ Quando um neurônio artificial passa em um valor diferente de zero para outro neurônio artificial, é dito ser ativado



# Função de Ativação

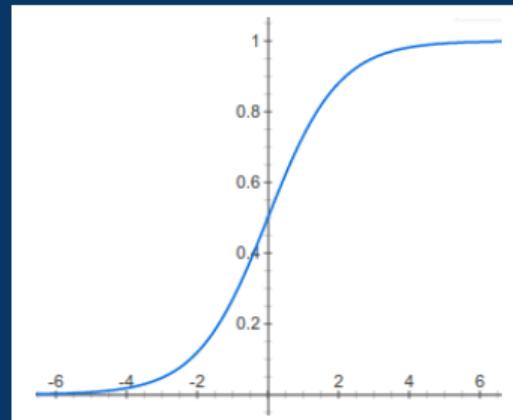
Função de Limiar (Degrau)

$$\begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases}$$



Função Sísmoide

$$f(v) = \frac{1}{1 + e^{(-av)}}$$



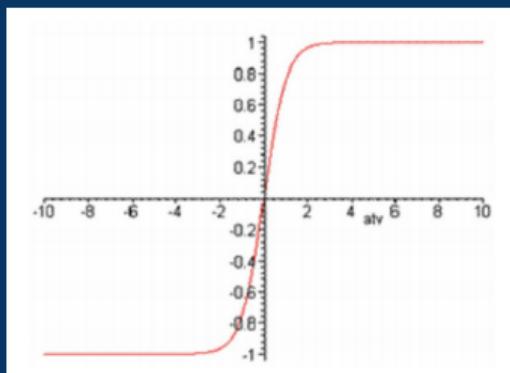
# Função de Ativação

- Tangente Hiperbólica: Como a Função Logística, também possui forma de “s”, assumindo valores entre 1 e -1, sendo representada por:

$$f(v) = a \frac{e^{(bv)} - e^{(-bv)}}{e^{(bv)} + e^{(-bv)}}$$

Onde:

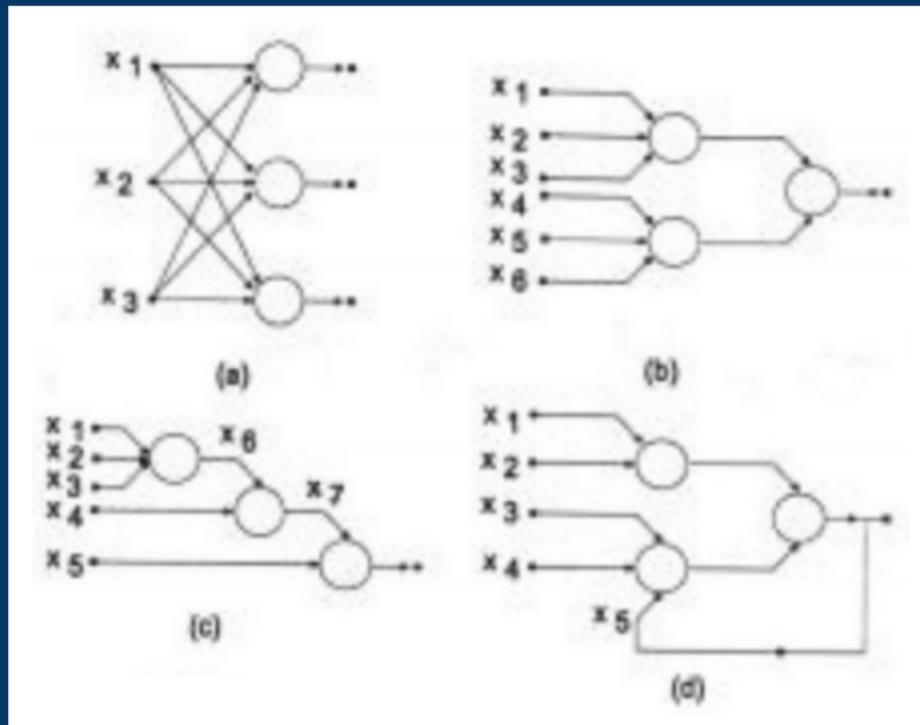
- $a$  define os limites inferiores e superiores ( $a = 1$  no gráfico)
- $b$  é o parâmetro de inclinação da curva



# Arquiteturas de RNAs

- ▶ Fazem parte da definição da arquitetura, os seguintes parâmetros:
  - ▶ Número de camadas da rede
  - ▶ Número de neurônios em cada camada
  - ▶ Tipo de conexão entre os neurônios
  - ▶ Topologia da rede

# Arquiteturas de RNAs



# Redes Neurais de Múltiplas Camadas

- ▶ Redes de uma só camada resolvem apenas problemas linearmente separáveis
- ▶ A solução de problemas não linearmente separáveis passa pelo uso de redes com uma ou mais camadas intermediárias
- ▶ Uma rede com uma camada intermediária pode implementar qualquer função contínua
- ▶ A utilização de duas camadas intermediárias permite a aproximação de qualquer função
- ▶ Dependendo da distribuição dos dados, a rede pode convergir para um mínimo local ou pode demorar demais para encontrar a solução desejada

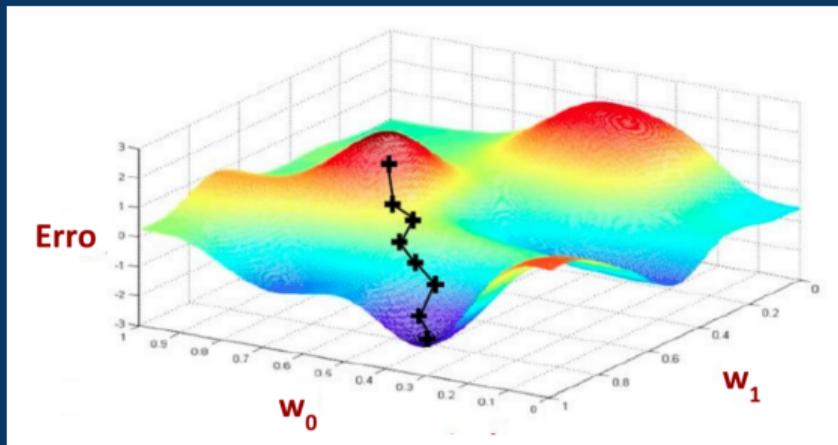


# Backpropagation



# Treinamento de Redes Neurais

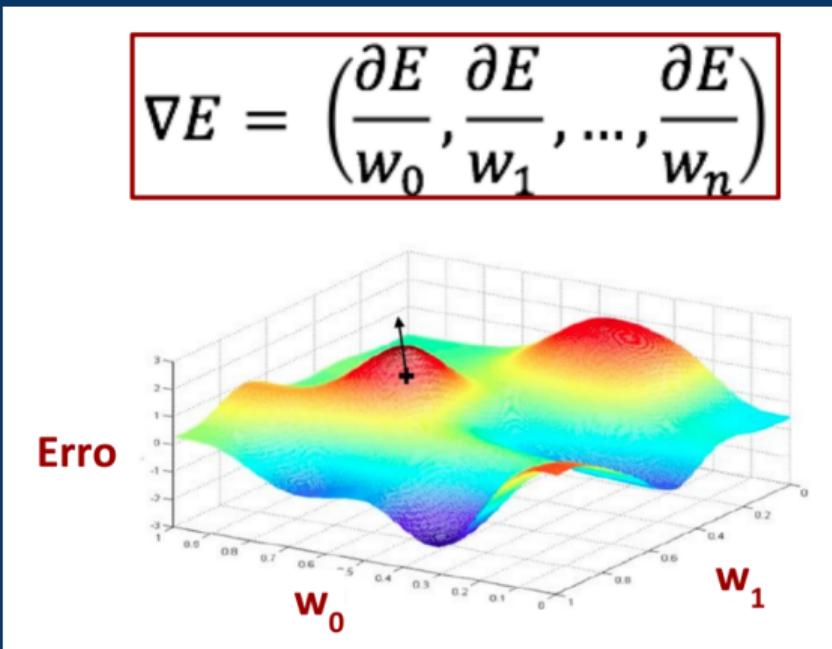
- ▶ Problema de otimização: encontrar a matriz de pesos sinápticos que minimiza o erro  $E$  entre a saída obtida pela rede e a saída desejada
- ▶ O algoritmo clássico de treinamento (Gradiente Descendente) atualiza os pesos na direção de descida do gradiente



# Gradiente

- O gradiente de uma função indica a direção de maior incremento em sua grandeza a partir de um ponto

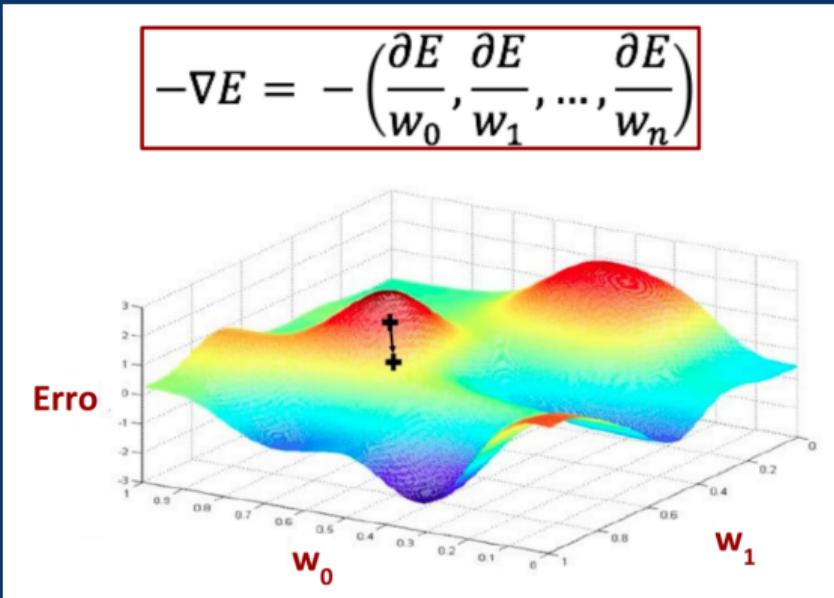
$$\nabla E = \left( \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right)$$



# Gradiente

- ▶ A direção oposta ao gradiente indica o maior declive

$$-\nabla E = -\left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n}\right)$$



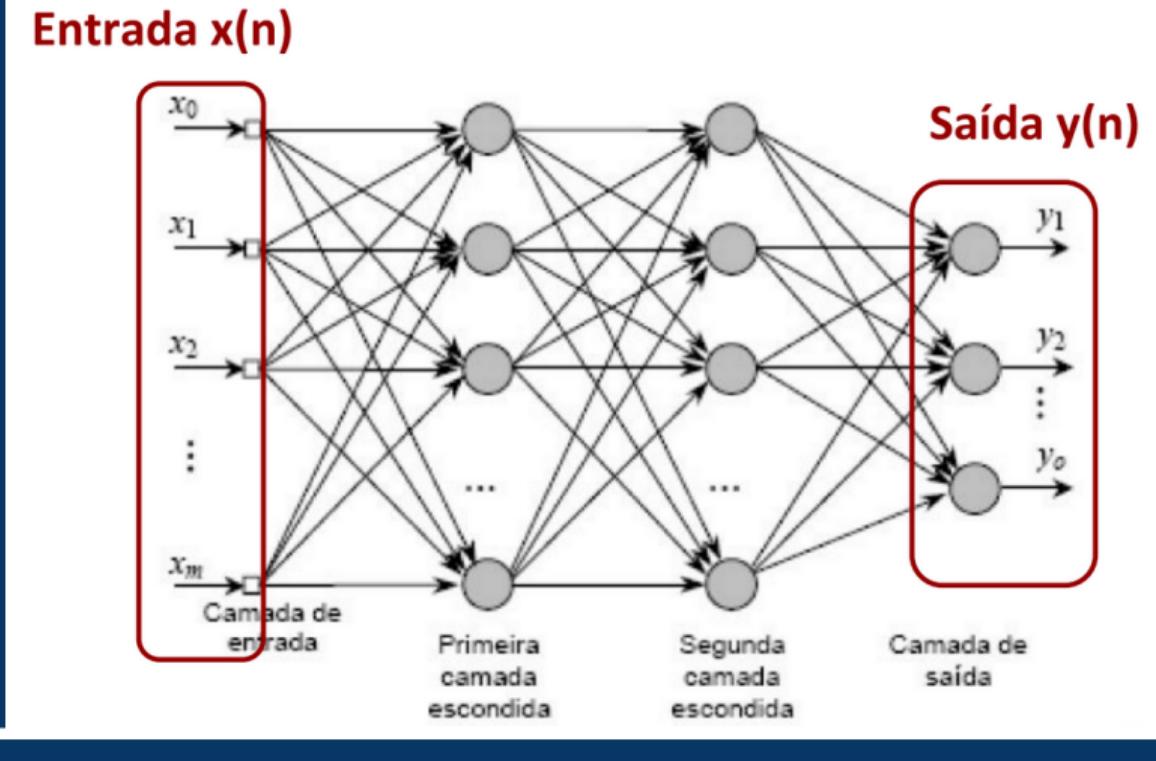
# Algumas Notações

- ▶ Erro na saída do neurônio  $j$  para o exemplo  $n$ :  $e_j(n) = d_j(n) - y_j(n)$
- ▶ Erro instantâneo para todos os neurônios da camada de saída  $C$  para um exemplo  $n$ :  $E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$
- ▶ Erro médio quadrático para todos os exemplos de treinamento:  
$$E_{med} = \frac{1}{N} \sum_{n=1}^N E(n)$$



# Backpropagation

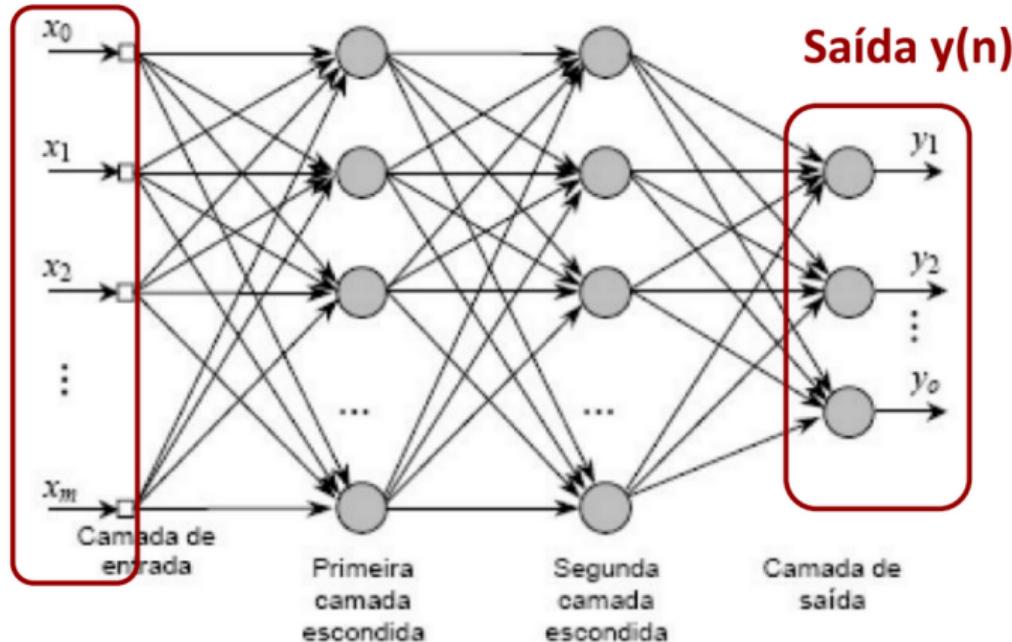
- ▶ Considere a rede de múltiplas camadas a seguir



# Backpropagation

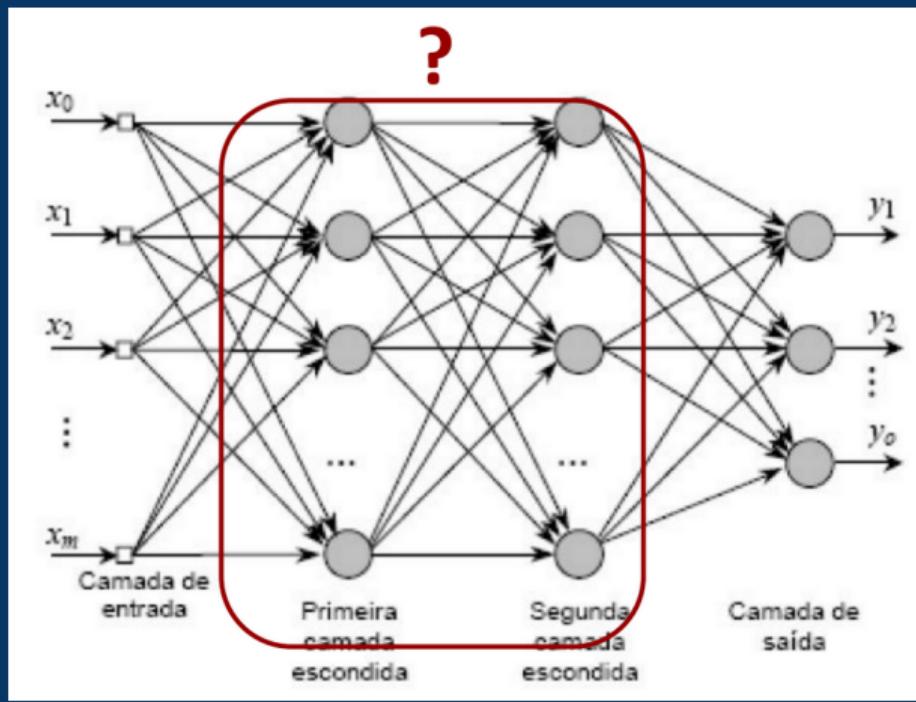
► **Problema:** o sinal de erro só pode ser calculado na camada de saída

Entrada  $x(n)$



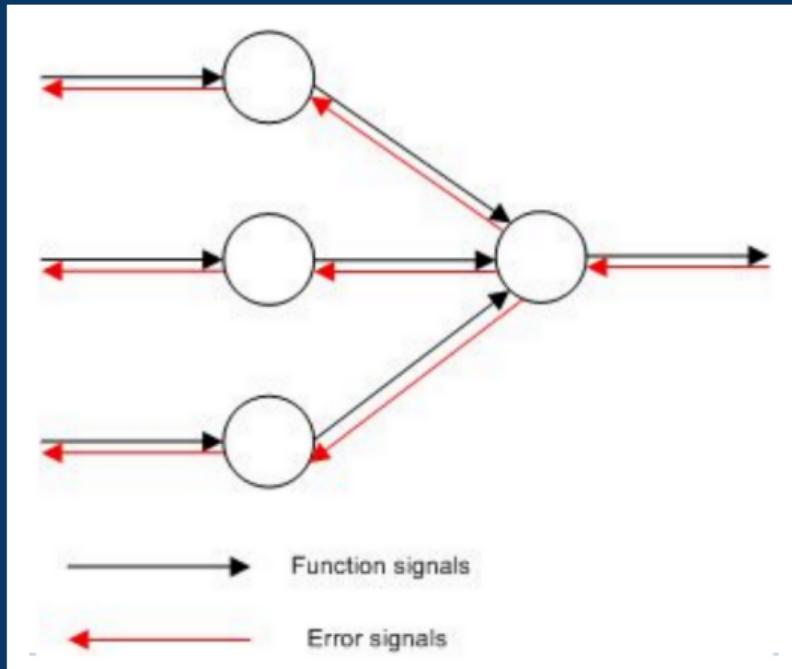
# Backpropagation

- ▶ Então como calcular o erro e ajustar os pesos das camadas escondidas?



# Backpropagation

- **Solução:** sinal de erro se origina em um neurônio da camada de saída e se propaga para trás camada por camada



# Backpropagation – Regra Delta

- A correção dos pesos é definida pela regra delta

$$\Delta w_{ij}(n) = \eta \delta_j(n) y_i(n)$$

- $\Delta w_{ij}(n)$  – valor de ajuste a ser acrescido ao peso  $w_{ij}$
- $j$  – é o neurônio atual
- $i$  – é um neurônio da camada anterior
- $\eta$  – é uma constante positiva que determina a taxa de aprendizagem
- $\delta_j(n)$  é o gradiente local e varia dependendo se  $j$  é oculto ou de saída

# Backpropagation – Gradiente Local

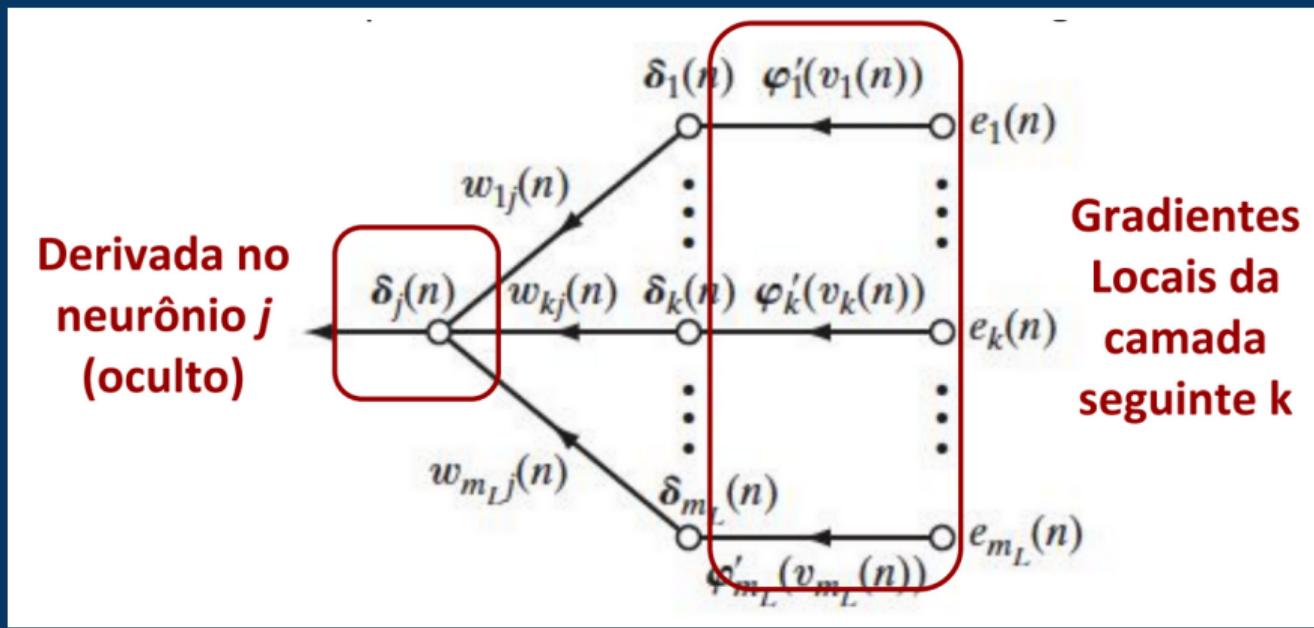
- ▶ Para neurônios da camada de saída, o gradiente local é dado pelo produto do sinal de erro pela derivada da função de ativação do neurônio

$$\delta_j(n) = e_j(n)\varphi'(v_j(n))$$



# Backpropagation – Gradiente Local

- Para neurônios da camada oculta, o gradiente local é dado pelo produto da derivada da função de ativação do neurônio pela soma ponderada dos gradientes locais da camada seguinte



# Backpropagation – Gradiente Local

- ▶ Para neurônios da camada oculta, o gradiente local é dado pelo produto da derivada da função de ativação do neurônio pela soma ponderada dos gradientes locais da camada seguinte

$$\delta_j(n) = \varphi'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$



# Funções de Ativação e Derivadas

- Sigmoide logística

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

- Tangente hiperbólica

$$\varphi(v) = \frac{1 - e^{-v}}{1 + e^{-v}}$$
$$\varphi'(v) = \frac{1}{2}(1 - \varphi^2(v))$$

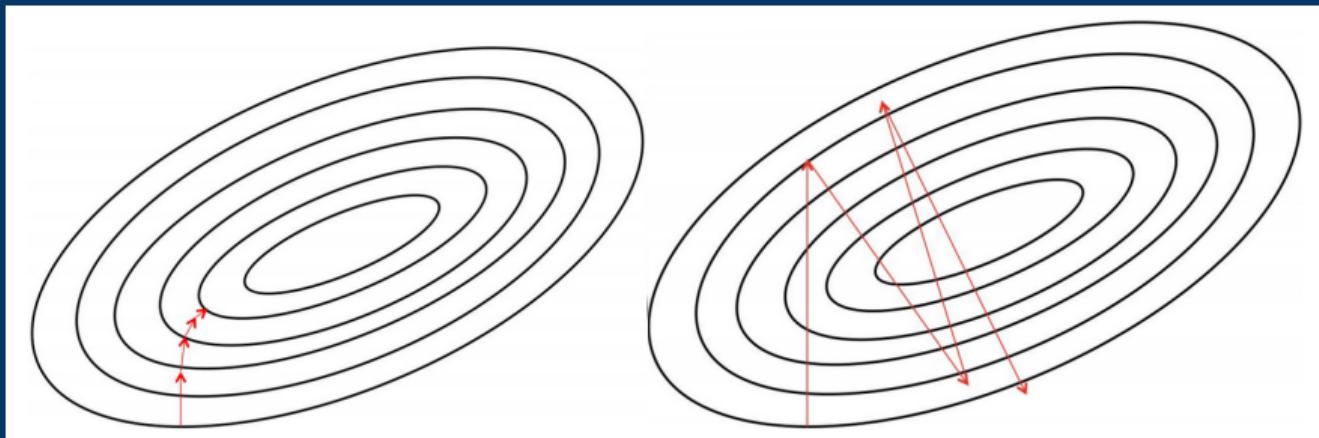
- ReLU

$$\varphi(v) = \max(0, v)$$
$$\varphi'(v) = \begin{cases} 1 & \text{se } v > 0 \\ 0 & \text{c.c} \end{cases}$$



# Backpropagation – Taxa de Aprendizagem

- ▶ Uma  $\eta$  muito pequena pode levar a uma aprendizagem muito lenta
- ▶ Por outro lado,  $\eta$  muito alta causar muito instabilidade no treinamento



## Para Terminar

- ▶ Redes neurais são capazes de gerar modelos poderosos, com alto grau de sucesso e aplicáveis às mais variadas tarefas
- ▶ No entanto, escolher a arquitetura correta e os hiper-parâmetros corretos para um dado problema pode ser uma tarefa difícil
- ▶ Hoje em dia, é comum reutilizar modelos que foram pré-treinados em tarefas semelhantes
- ▶ A interpretação dos seus resultados ainda é um problema em aberto
- ▶ Veremos mais à frente que as redes costumam ser muito confiantes nas suas previsões e, portanto, não costumam ser bons estimadores de probabilidades



# Sugestão de Atividade

- ▶ Brinque com o <https://playground.tensorflow.org/>
- ▶ Implemente uma rede neural que pode ter qualquer número de camadas/neurônios por camada/função de ativação e use os scripts de geração de conjuntos da aula passada para avaliar seus modelos

FIM



---

# Aprendizagem de Máquina

## Redes Neurais

Telmo de Menezes e Silva Filho

[tmfilho@gmail.com](mailto:tmfilho@gmail.com)/[telmo@de.ufpb.br](mailto:telmo@de.ufpb.br)

[www.de.ufpb.br](http://www.de.ufpb.br)

**UFPB**

 Departamento de  
**ESTATÍSTICA**