# TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

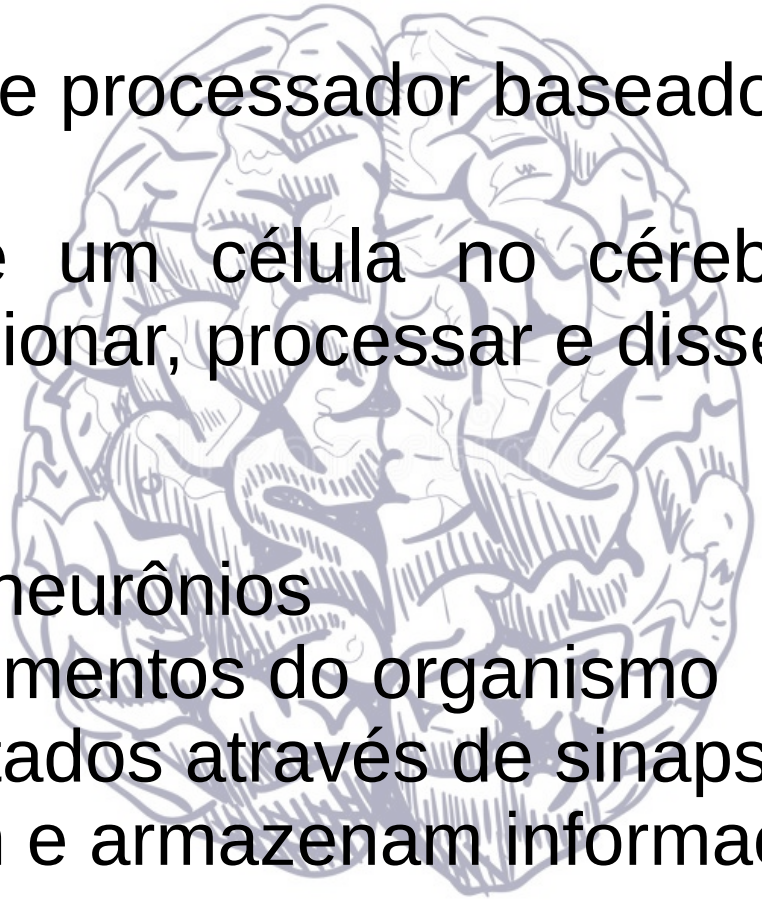
## Redes Neurais Artificiais

Professor Ricardo Kerschbaumer
ricardo.kerschbaumer@ifc.edu.br

http://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/

# Introdução

## O Cérebro humano

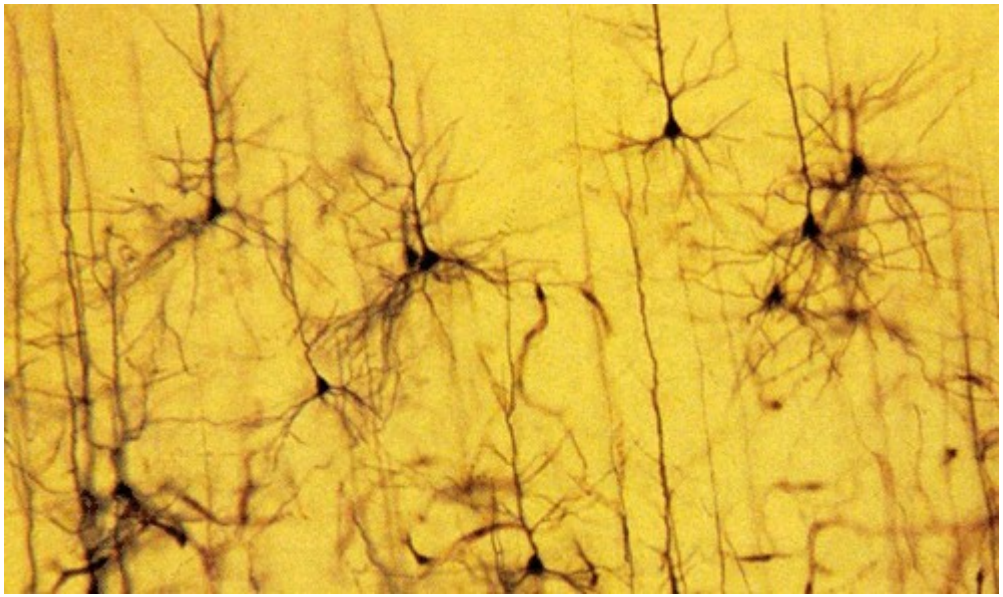


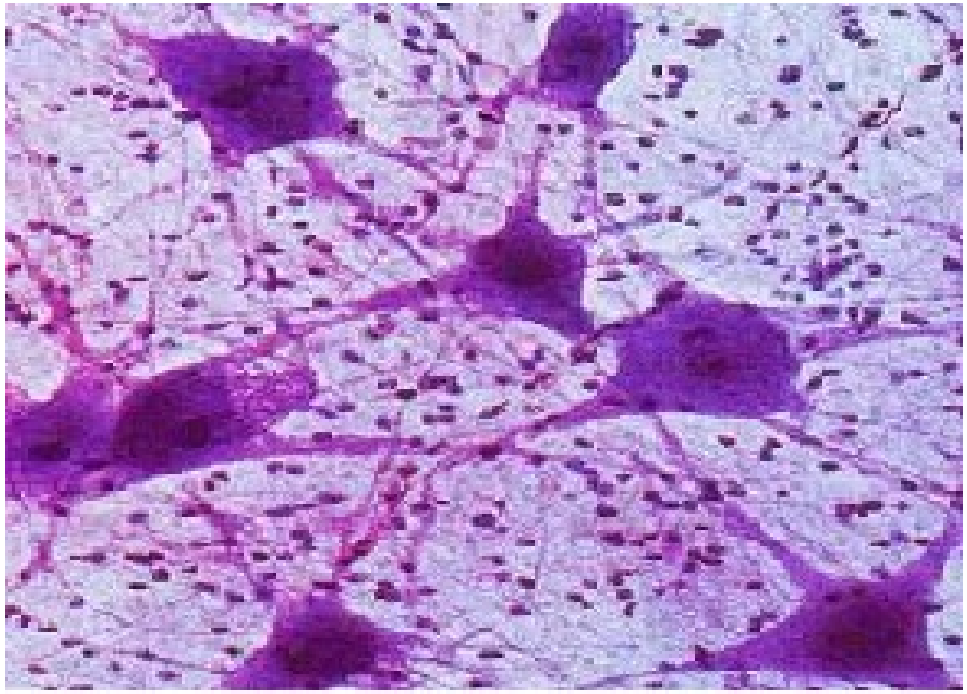
- Mais fascinante processador baseado em carbono
- O neurônio é um célula no cérebro cuja principal função é colecionar, processar e disseminar sinais elétricos
- 10 bilhões de neurônios
  - todos movimentos do organismo
  - são conectados através de sinapses
  - processam e armazenam informações

# Introdução

## Redes Neurais Naturais

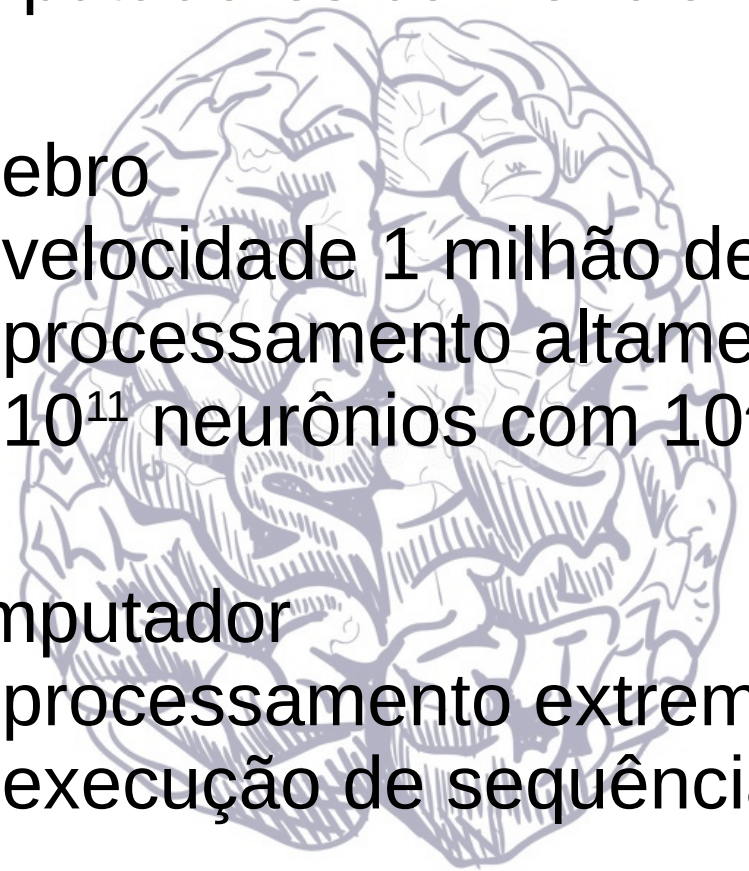
- O sistema nervoso é formado por um conjunto extremamente complexo de células, os neurônios
- O cérebro humano possui cerca de **$10^{11}$ neurônios** e mais de **$10^{14}$ sinapses**, possibilitando a formação de redes muito complexas

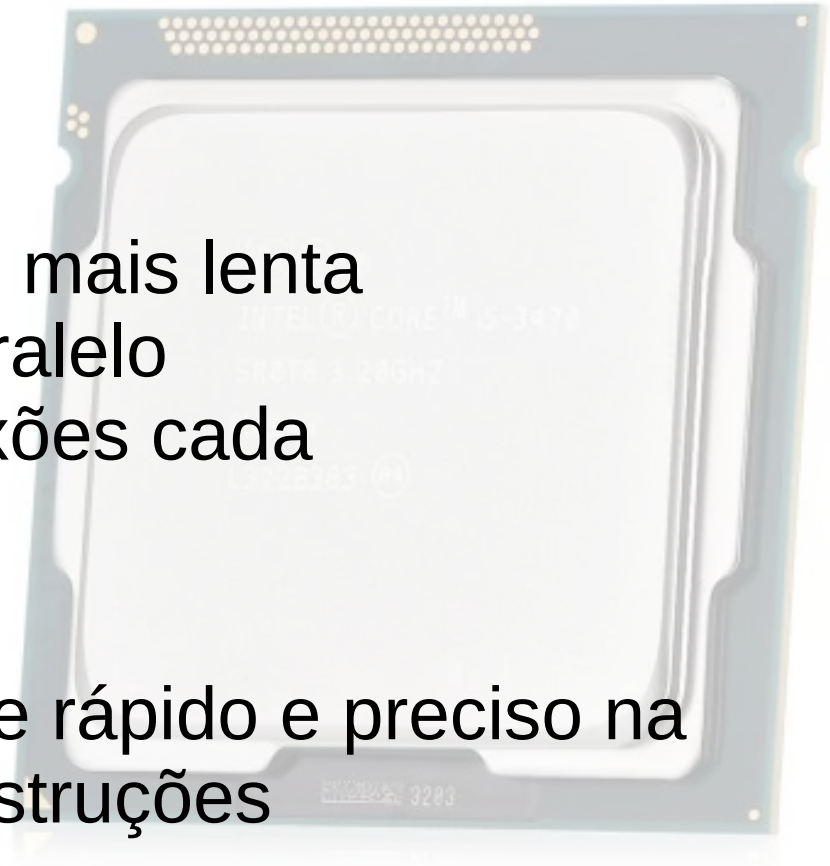




# Introdução

O cérebro processa informações de forma diferente dos computadores convencionais



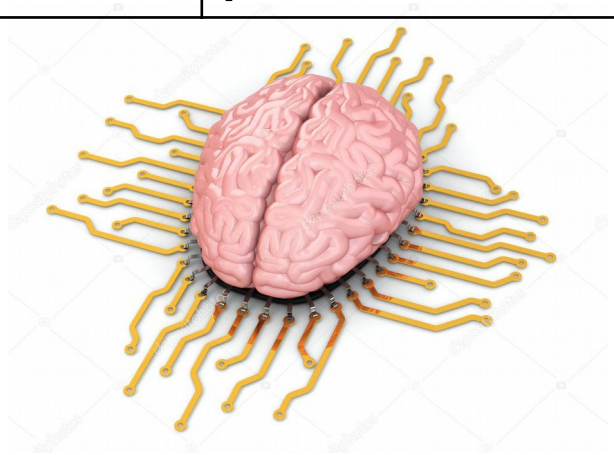


- Cérebro
  - velocidade 1 milhão de vezes mais lenta
  - processamento altamente paralelo
  - $10^{11}$ neurônios com $10^{4}$ conexões cada

- Computador
  - processamento extremamente rápido e preciso na
  - execução de sequência de instruções

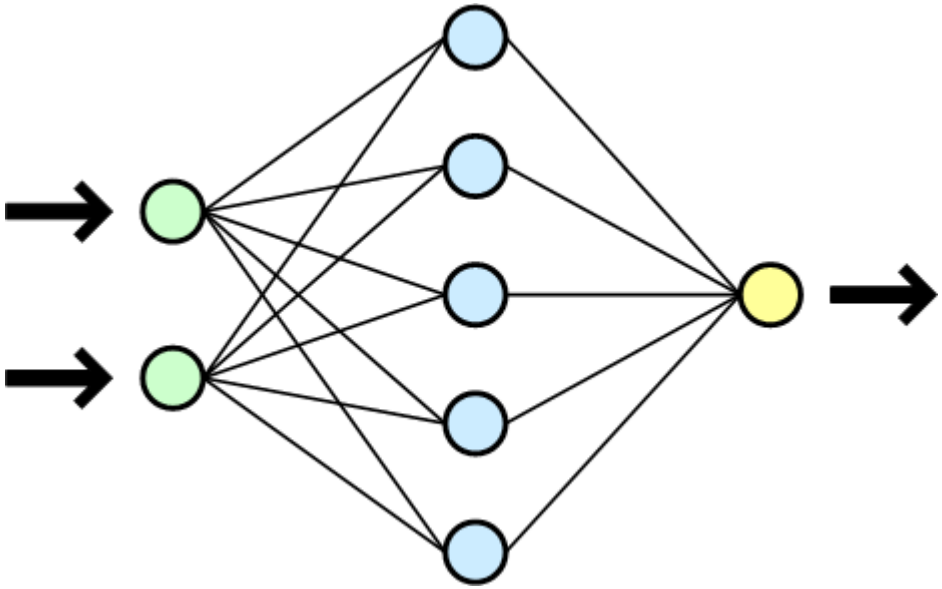
# Introdução

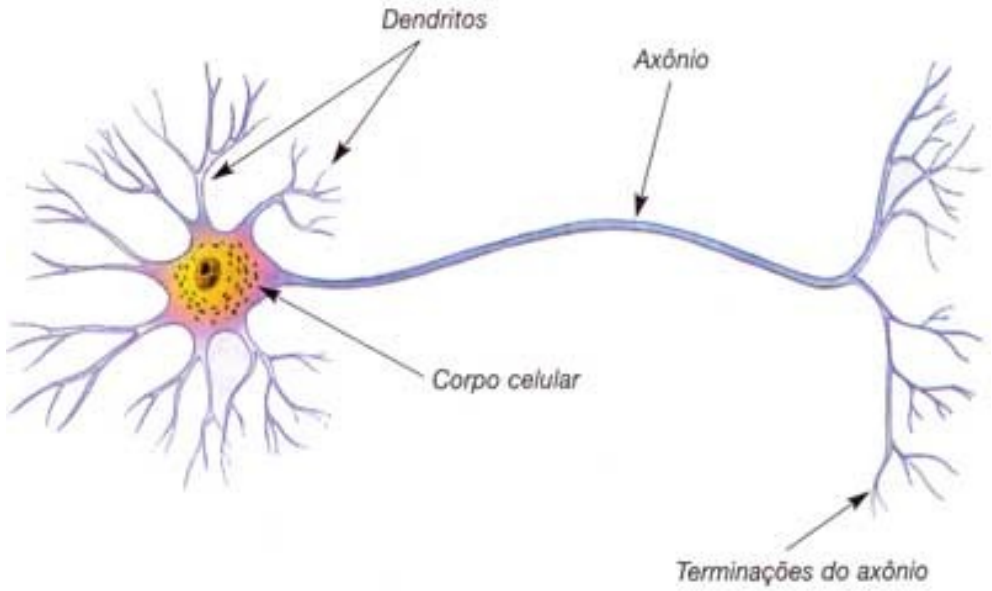
| **Computadores** | **Rede Neural** |
|---|---|
| Executa programas | Aprende |
| Executa operações lógicas | Executa operações não lógicas, transformações, comparações |
| Depende do modelo ou do programador | Descobre as relações ou regras dos dados e exemplos |
| Testa uma hipótese por vez | Testa todas as possibilidades em paralelo |



# Redes Neurais Artificiais

**Redes Neuronais Artificiais** (RNAs) são modelos computacionais inspirados pelo sistema nervoso central de um animal (em particular o cérebro) que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões (Wikipedia).

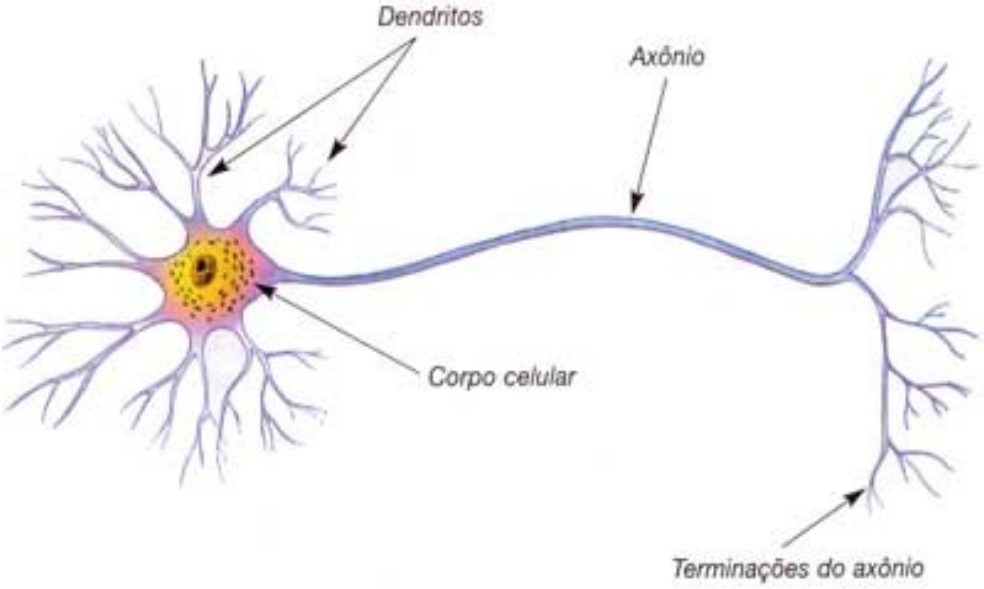


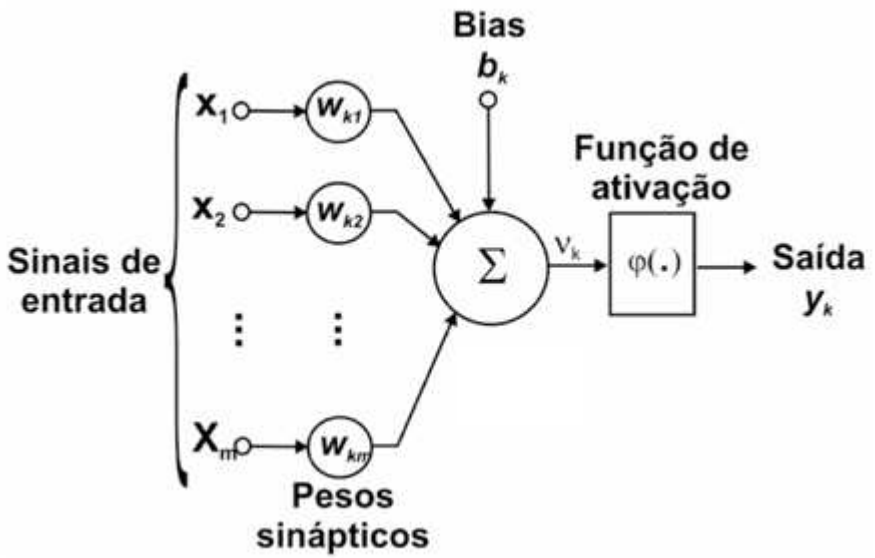


INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

# Redes Neurais Artificiais

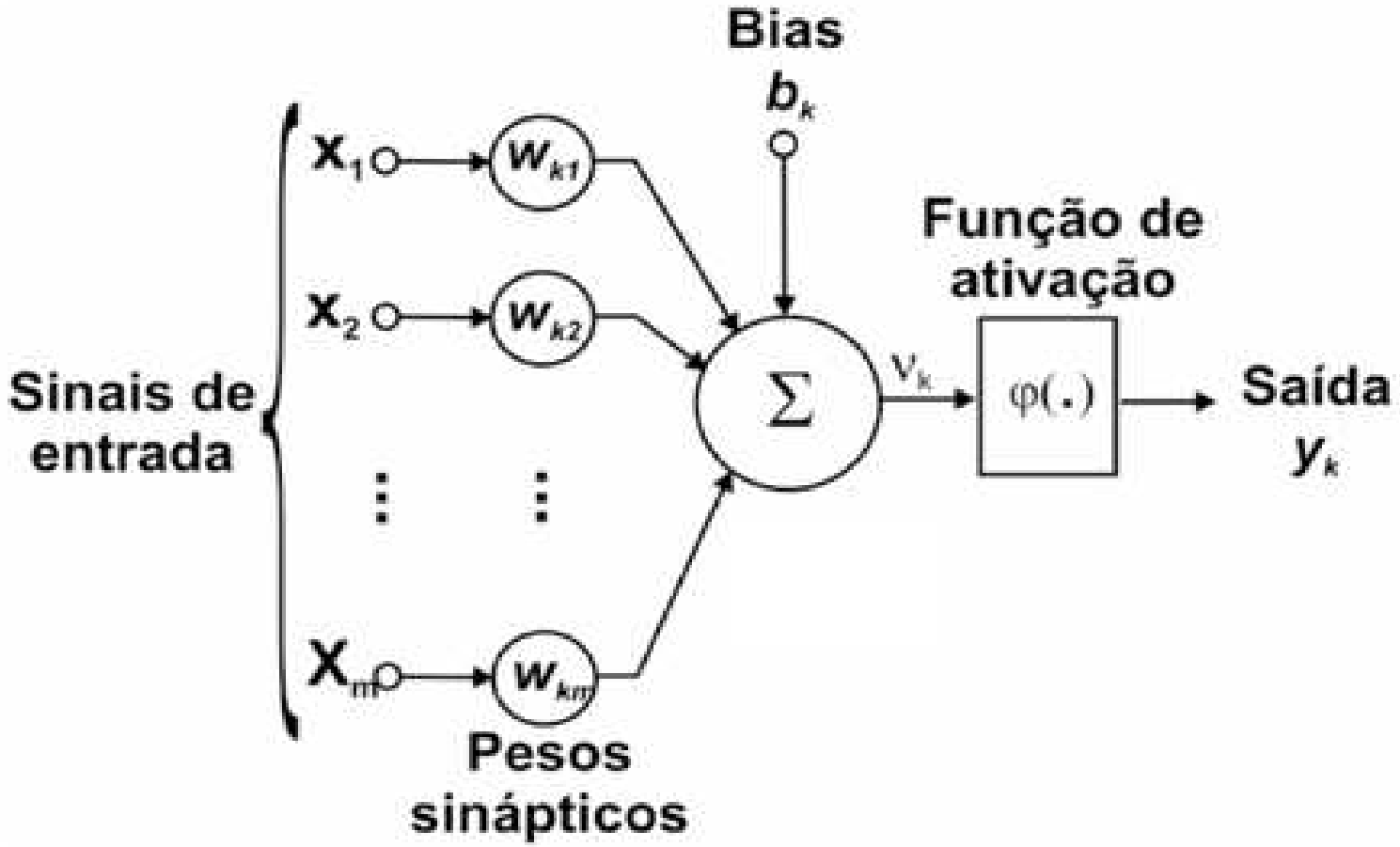
As Redes Neurais são compostas de uma coleção massivamente paralela de unidades de processamento (Neurônios) pequenas e simples, onde as interligações são responsáveis pela maior parte da “inteligência”.



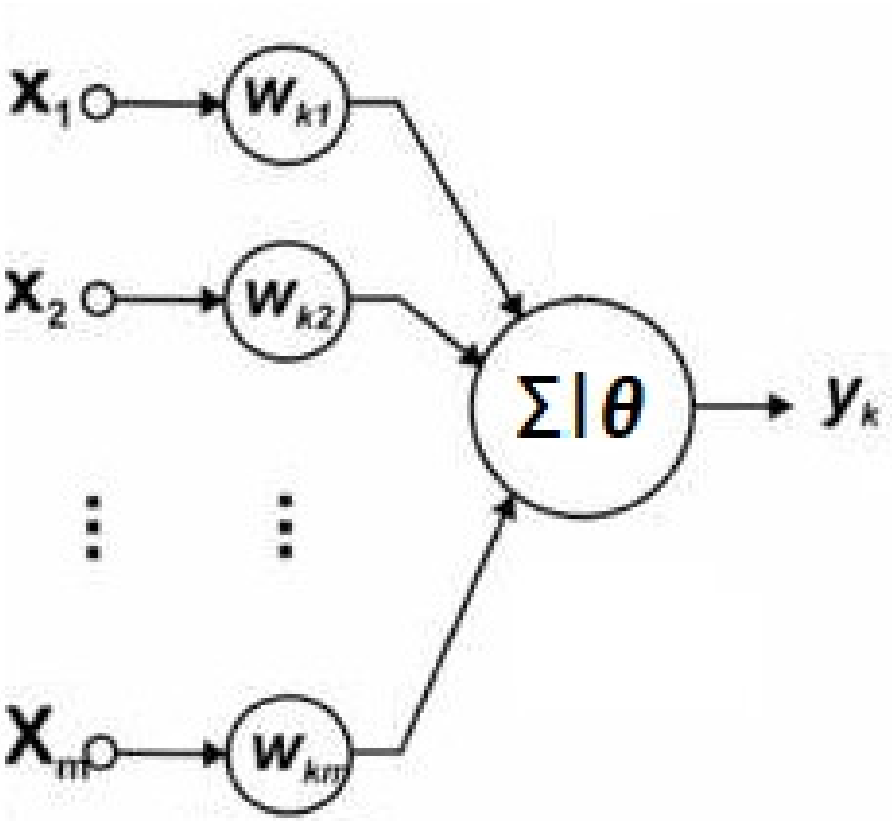
# Redes Neurais Artificiais

Os primeiros trabalhos na área datam de 1943: McCulloch e Pitts desenvolveram o primeiro modelo matemático do neurônio.



# Redes Neurais: Neurônio MCP

- Neste modelo os neurônios são unidades de processamento simples com uma função de ativação degrau (Limiar).

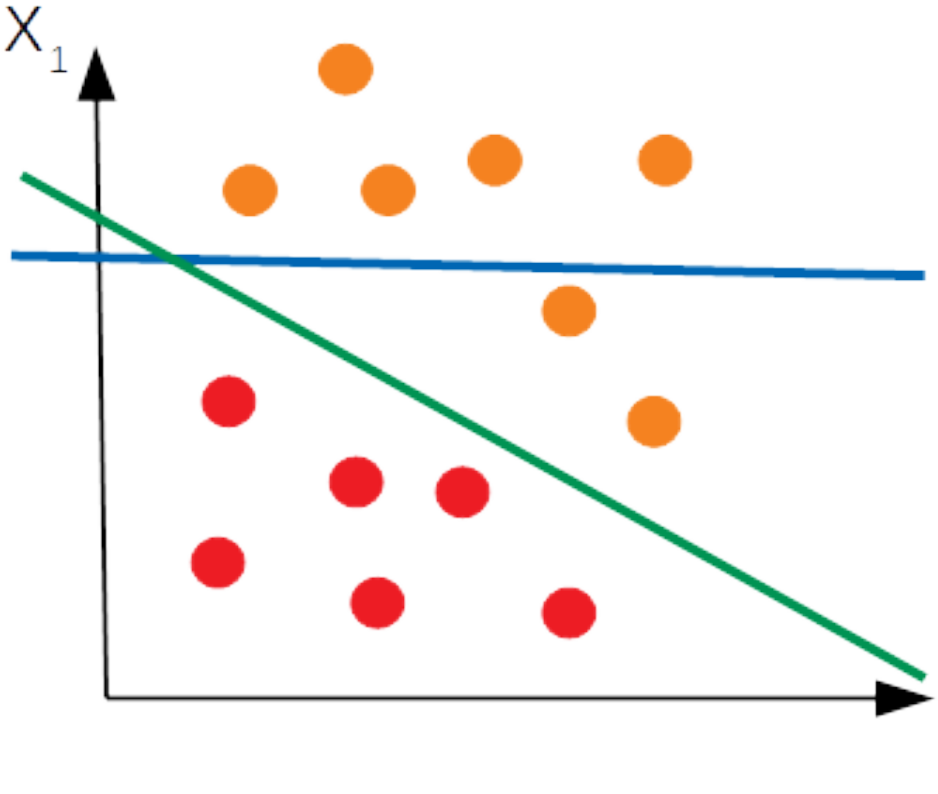


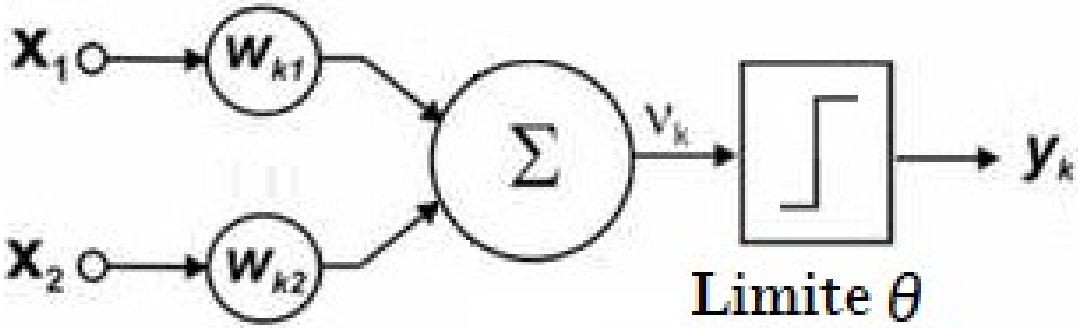

$$Y_k = 1 \; se \left( \sum_{i=1}^{n} w_{ki} x_i \right) \geqslant \theta$$

$$Y_k = 0 \; caso \; contrário$$

# Partições do espaço de busca

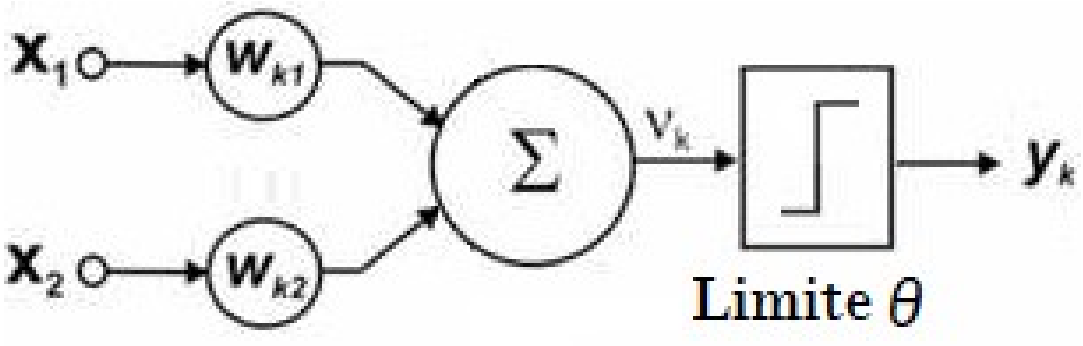
Neurônios com diferentes pesos em suas entradas e diferentes valores de limiar produzem diferentes partições no espaço de entradas.



# Partições do espaço de busca

Vamos analisar o caso particular de 2 entradas e 1 saída.




$$Y_k = 1 \; se \left( \sum_{i=1}^{n} w_{ki} x_i \right) \geqslant \theta$$

$$Y_k = 0 \; caso contrário$$

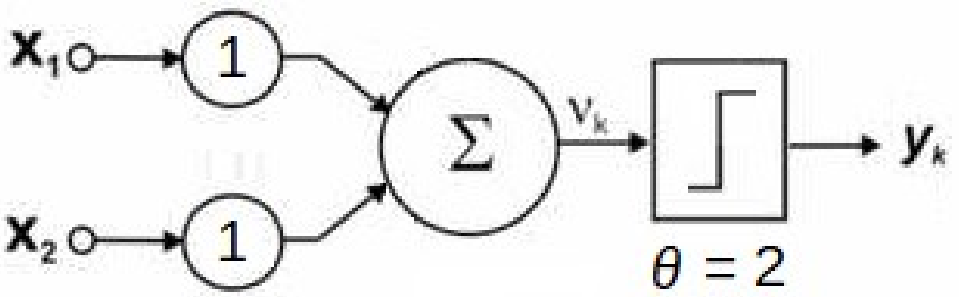
$$\Rightarrow$$

$$Y_k = 1 \; se \; W_{k1} X_1 + W_{k2} X_2 \geqslant \theta$$

$$Y_k = 0 \; caso contrário$$

# Partições do espaço de busca

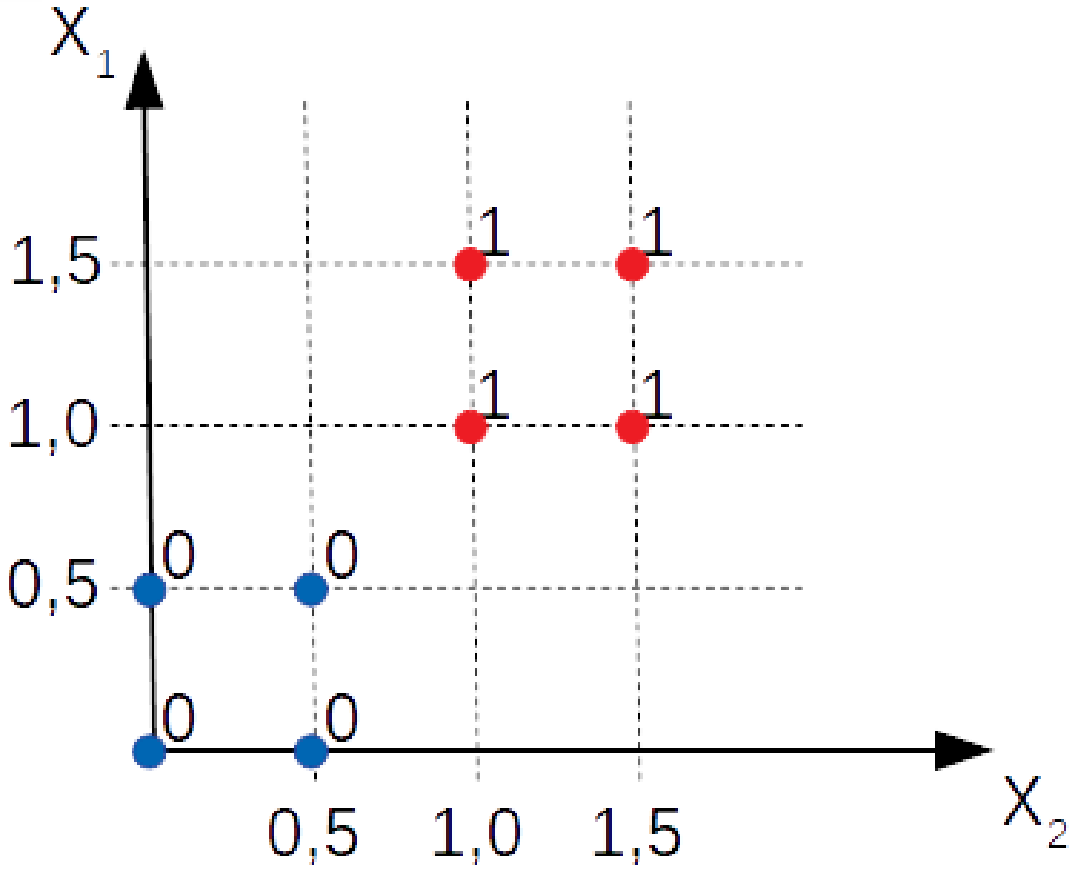
Considerando w1 = 1, w2 = 1 e Θ = 2 temos:




$$Y_k = 1 \; se \; 1 \cdot X_1 + 1 \cdot X_2 \geqslant 2$$

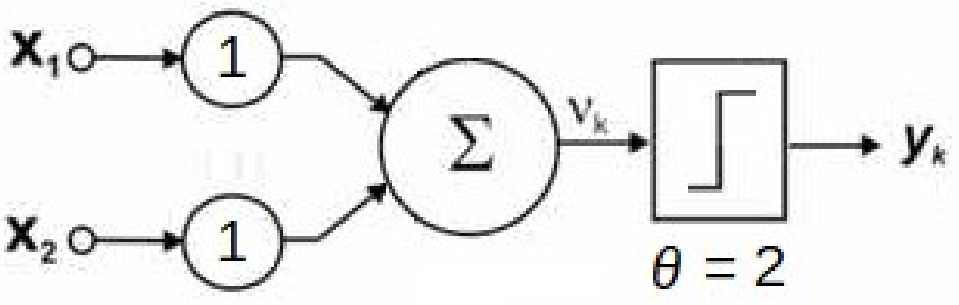
$$Y_k = 0 \; caso \; contrário$$

| $X_1$ | $X_2$ | $Y_k$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0,5 | 0 |
| 0,5 | 0 | 0 |
| 0,5 | 0,5 | 0 |
| 1 | 1 | 1 |
| 1 | 1,5 | 1 |
| 1,5 | 1 | 1 |
| 1,5 | 1,5 | 1 |



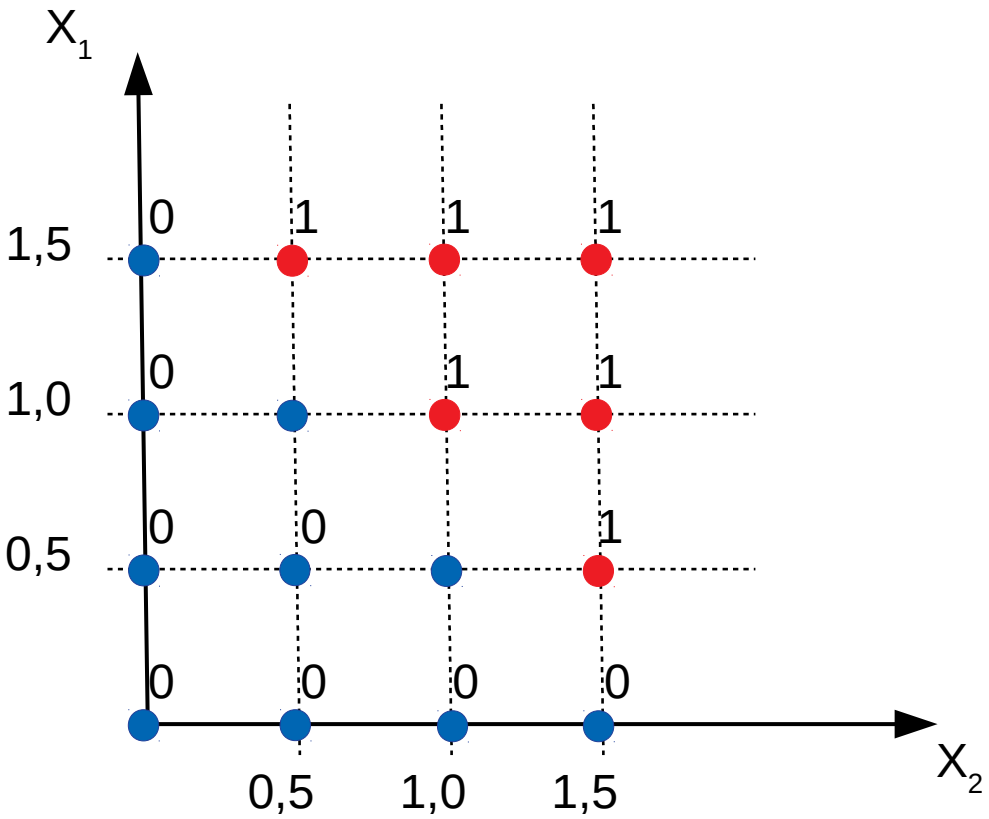
# Partições do espaço de busca

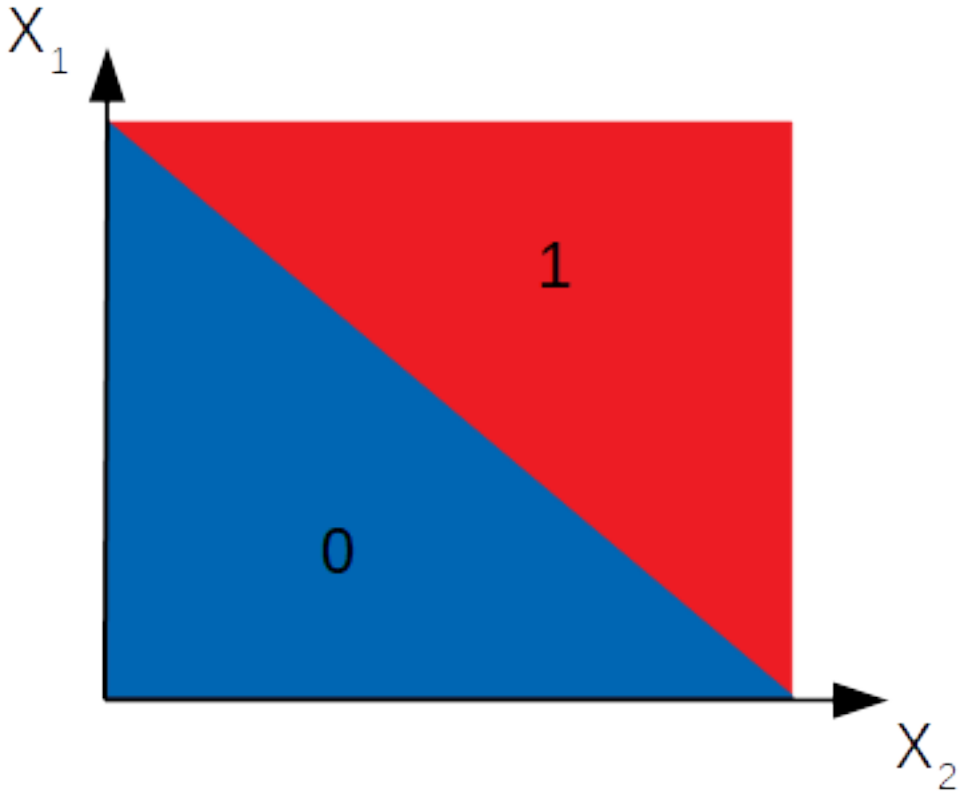
Se aumentarmos o número de entradas temos:




$$Y_k = 1 \; se \; 1 \cdot X_1 + 1 \cdot X_2 \geqslant 2$$

$$Y_k = 0 \; caso \; contrário$$



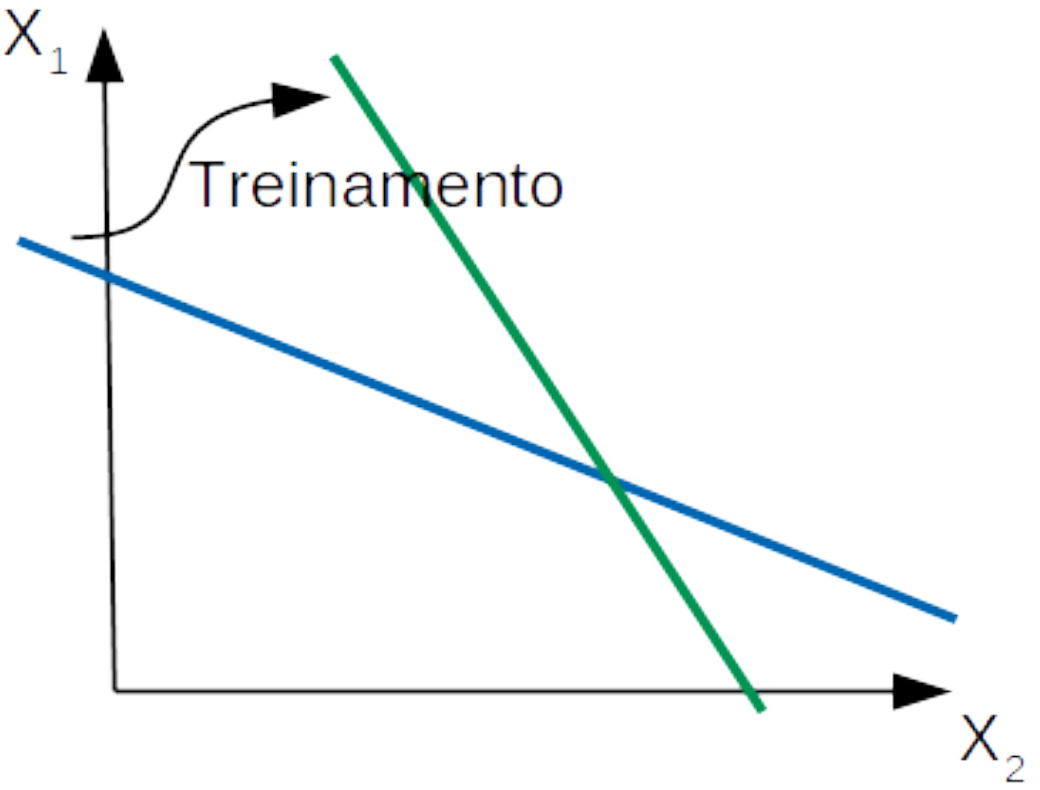
# Partições do espaço de busca

A saída (y) é uma função da combinação linear das entradas (x).

$$W_1 \cdot X_1 + W_2 \cdot X_2 = \theta$$

ou

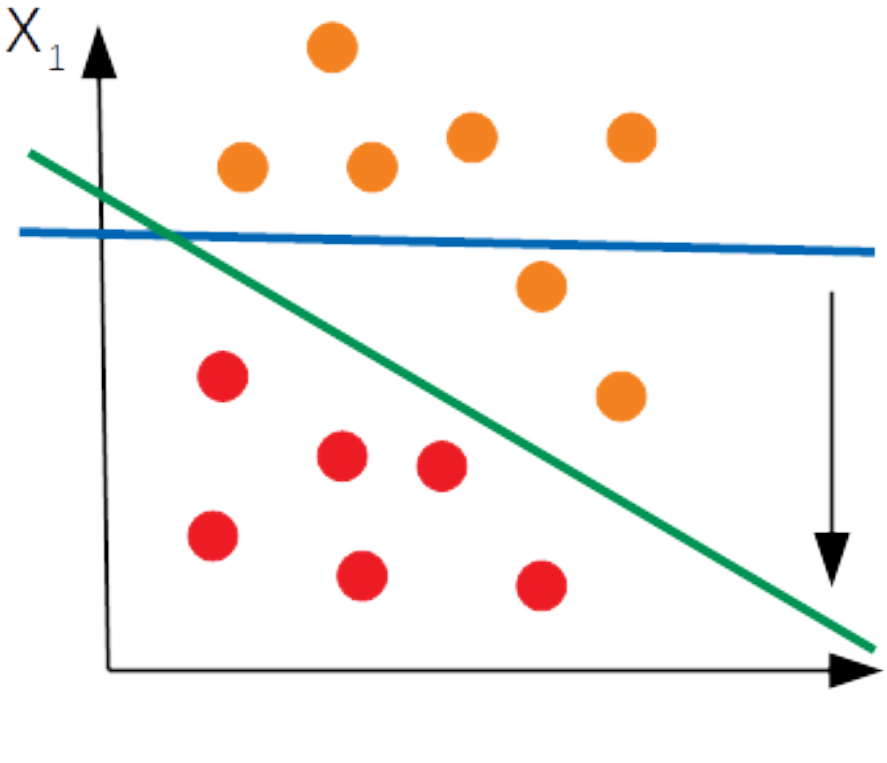
$$X_2 = -\left(\frac{W_1}{W_2}\right) \cdot X_1 + \left(\frac{\theta}{W_2}\right)$$




O treinamento de um neurônio e consequentemente de uma Rede Neural Artificial acontece através da alteração dos parâmetros w1, w2 e Θ.

# Problemas de classificação

Levando em consideração que a alteração dos parâmetros w e Θ (treinamento) modifica a posição da reta e portanto da partição no espaço de entrada é possível observar que a estrutura de uma Rede Neural Artificial é apropriada para resolver problemas de classificação.




$$Y_k = 1 \; se \; W_1 \cdot X_1 + W_2 \cdot X_2 \geq \theta$$
$$Y_k = 0 \; se \; W_1 \cdot X_1 + W_2 \cdot X_2 < \theta$$

Treinamento

$$Y_k = 1 \; se \; W_1^t \cdot X_1 + W_2^t \cdot X_2 \geq \theta^t$$
$$Y_k = 0 \; se \; W_1^t \cdot X_1 + W_2^t \cdot X_2 < \theta^t$$

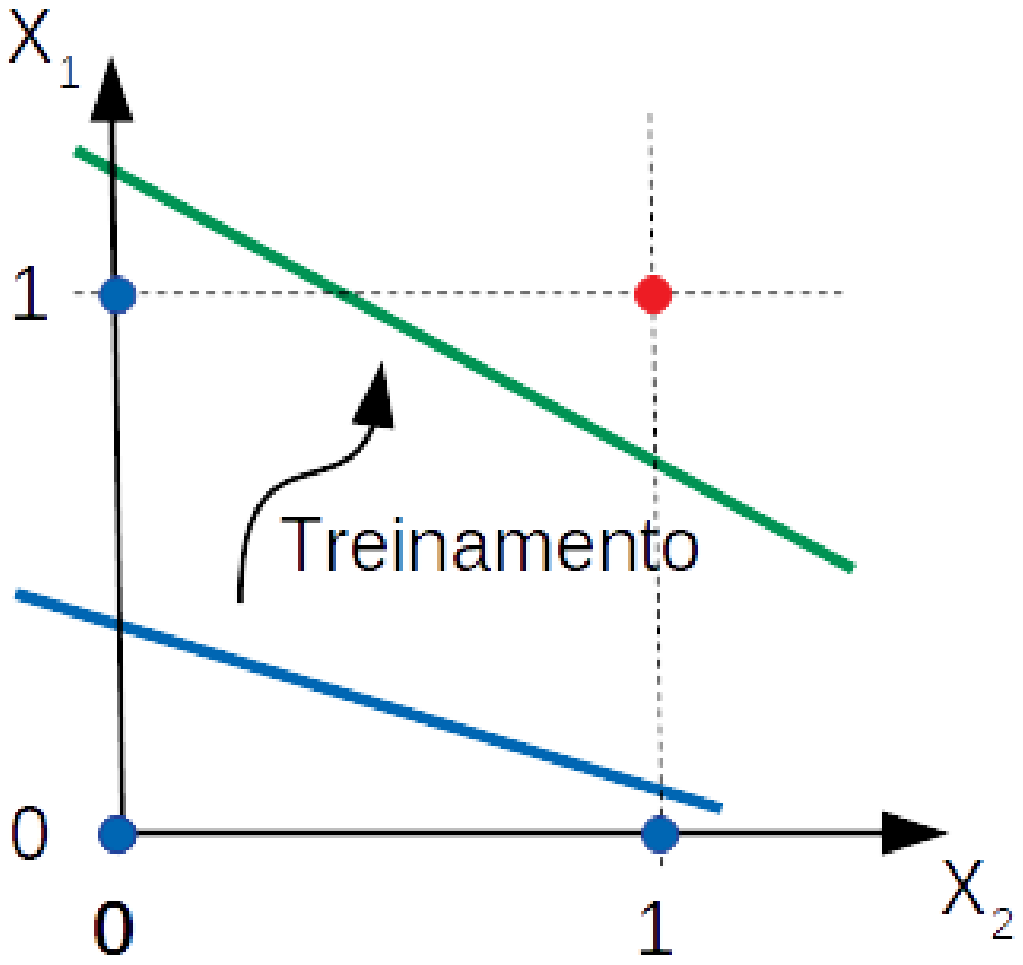
# Resolução de problemas

Devido a simplicidade apresentada para os neurônios este modelo ser apenas para a solução de problemas simples ou linearmente separáveis.

Exemplo:

AND Lógico

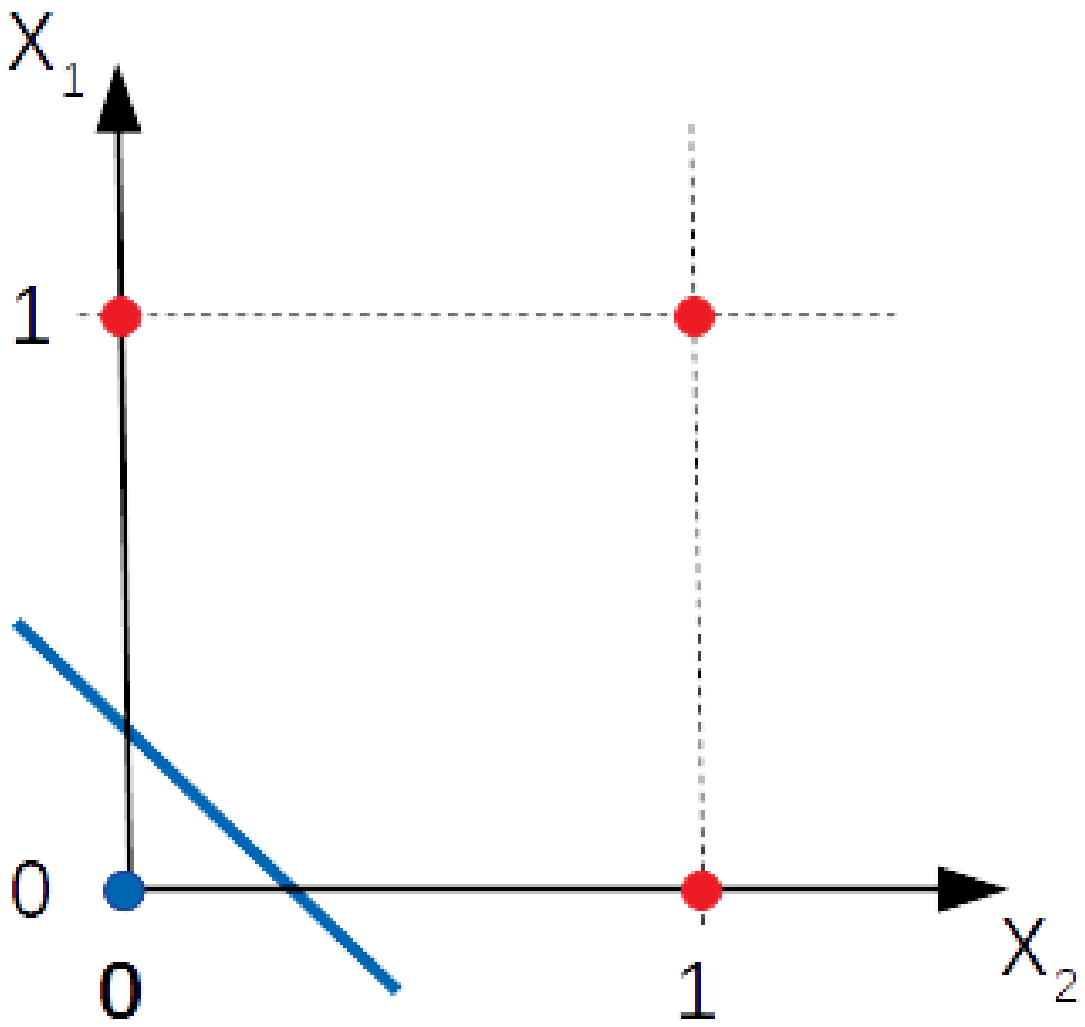
| $X_1$ | $X_2$ | $Y_k$ |
|---|---|---|
| 1 | 1 | **1** |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |



# Resolução de problemas

OR Lógico: Também é linearmente separável.

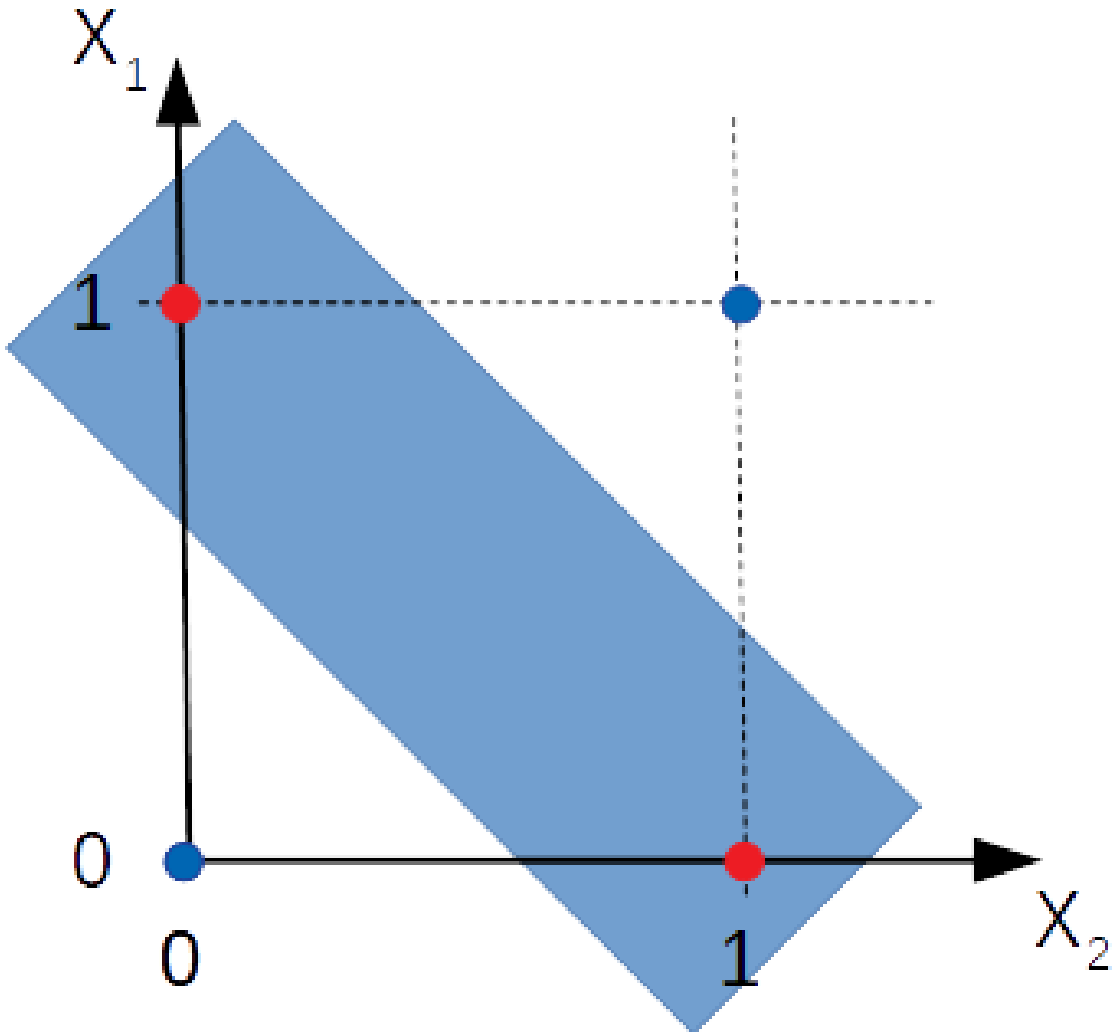
| $X_1$ | $X_2$ | $Y_k$ |
|---|---|---|
| 1 | 1 | **1** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 0 | 0 | 0 |



# Resolução de problemas

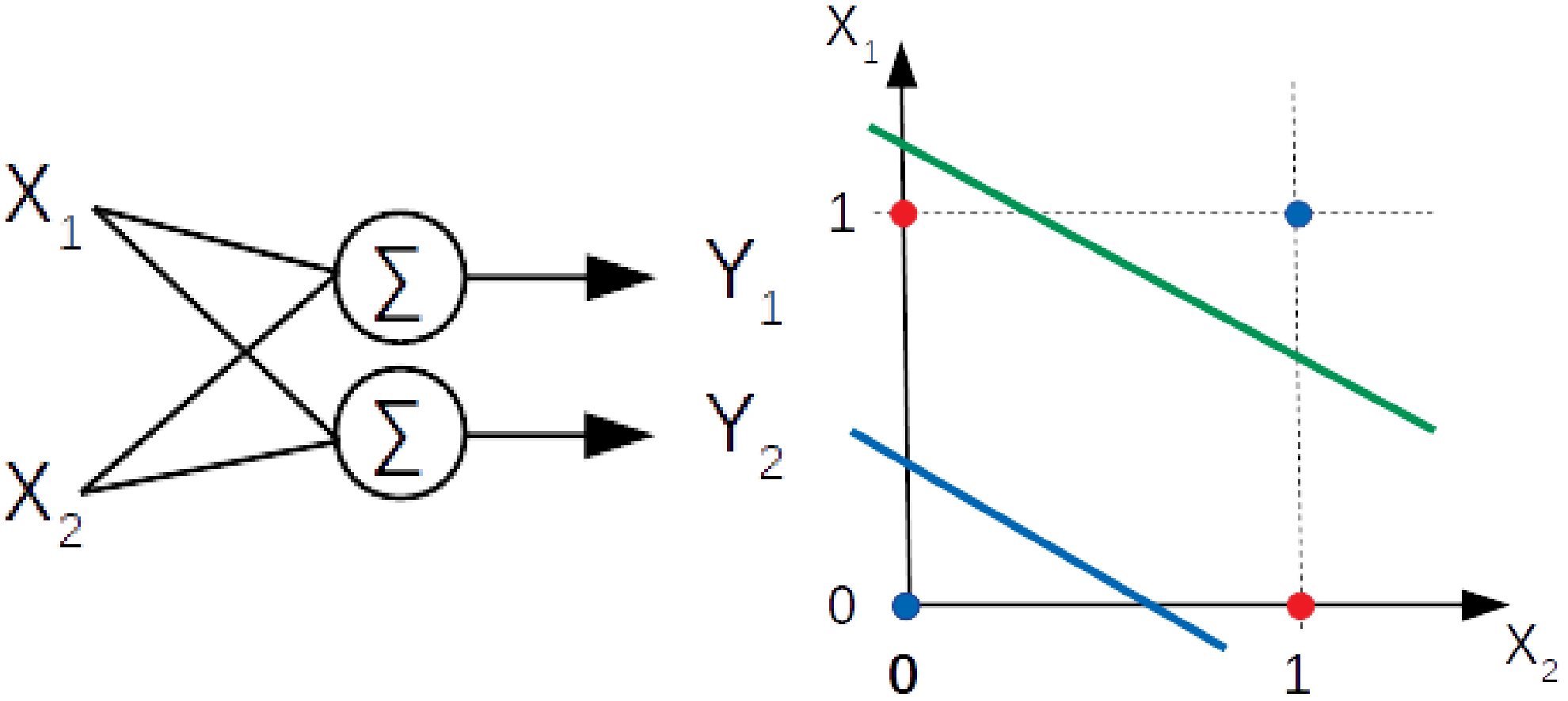
XOR Lógico: Não é linearmente separável.

| $X_1$ | $X_2$ | $Y_k$ |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 0 | 0 | 0 |



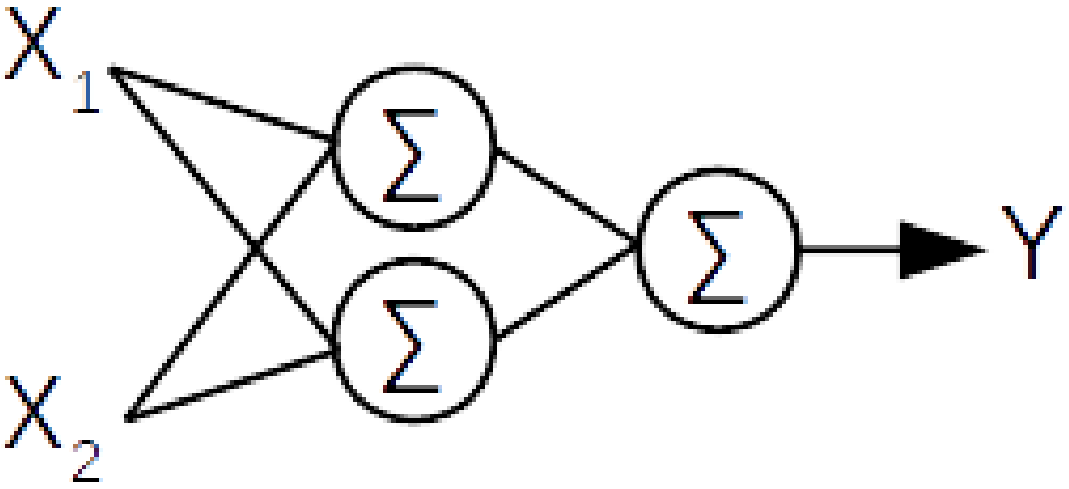
# Resolução de problemas

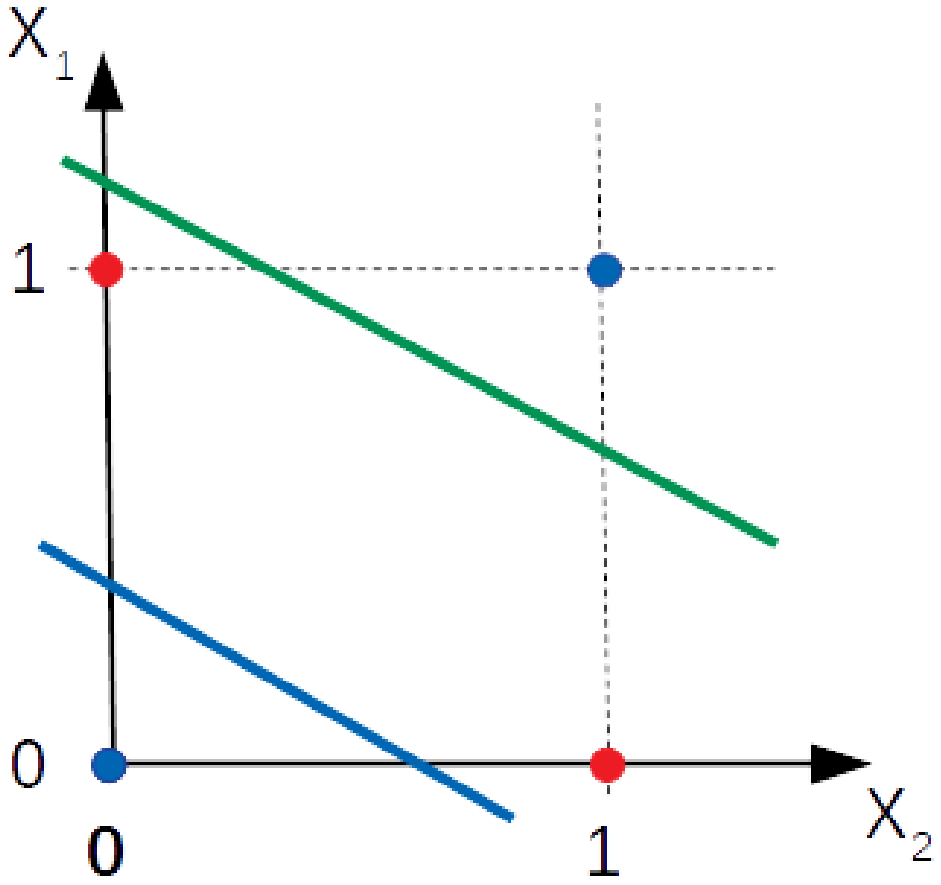
- Como resolver problemas que não são linearmente separáveis?
- Aumentar o número de neurônios resolve? Não!



# Resolução de problemas

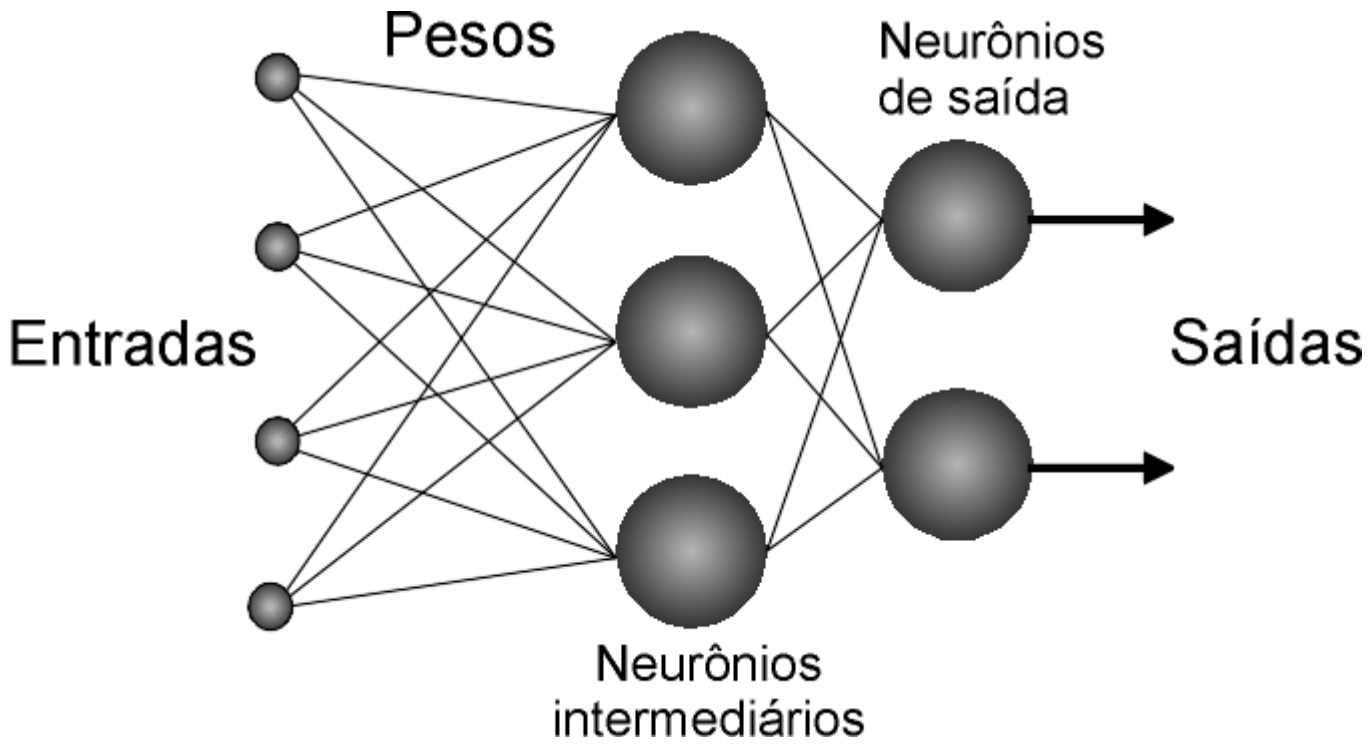
A solução é aumentar o número de camadas de neurônios.



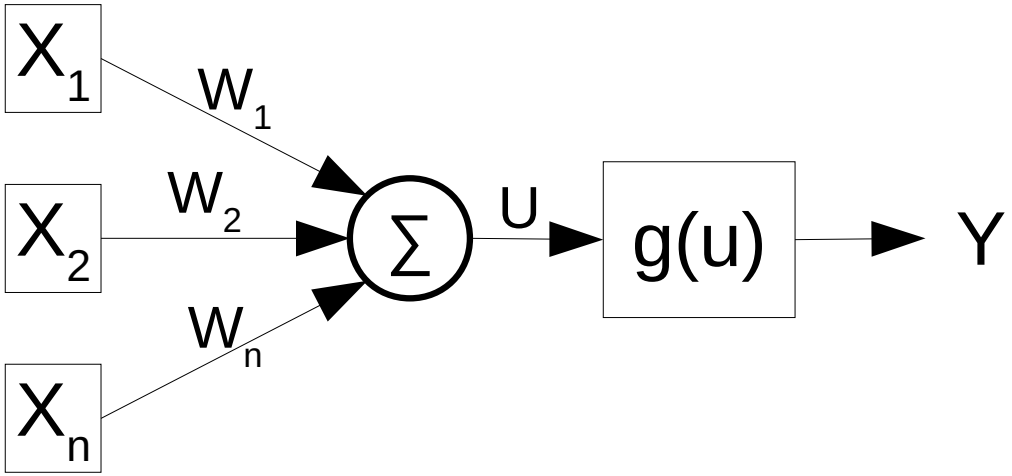
# Redes Neurais Artificiais modernas

As redes neurais mais modernas são compostas por múltiplas camadas de neurônios e possibilitam a resolução de problemas não linearmente separáveis.



# Funções de ativação

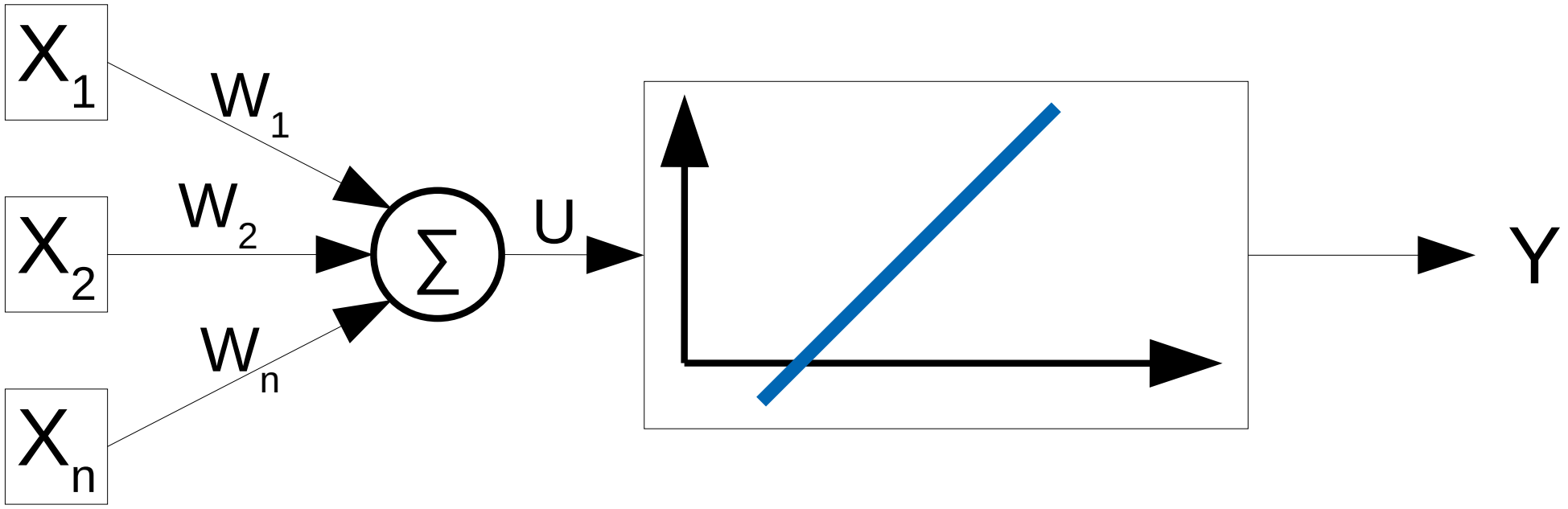
Para o modelo de neurônio a seguir temos:




- Potencial de ativação: $u = \sum_{i-1}^{n} W_i \cdot X_i$
- Função de ativação: g( ). (função degrau nos exemplos anteriores)
- Assim: $Y = g(u) = g\left( \sum_{i-1}^{n} W_i \cdot X_i \right)$

# Funções de ativação

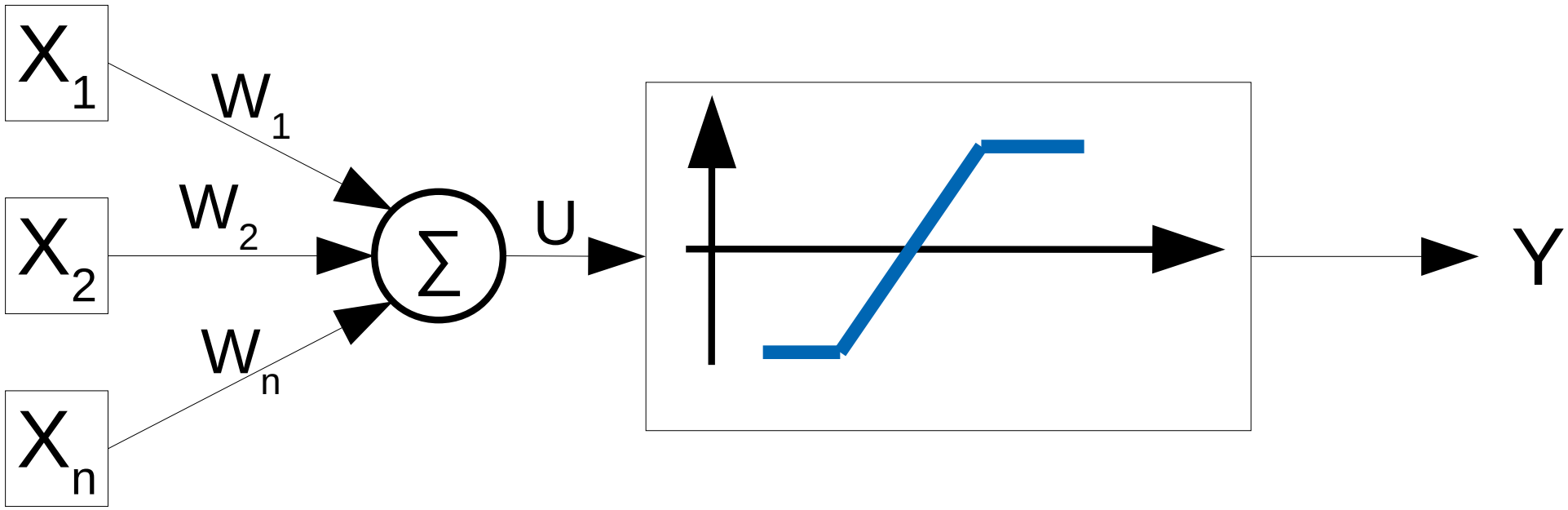
## Função Linear




$$Y = \gamma\left(\sum_{i-1}^{n} W_i \cdot X_i\right) = \gamma\, u$$

# Funções de ativação

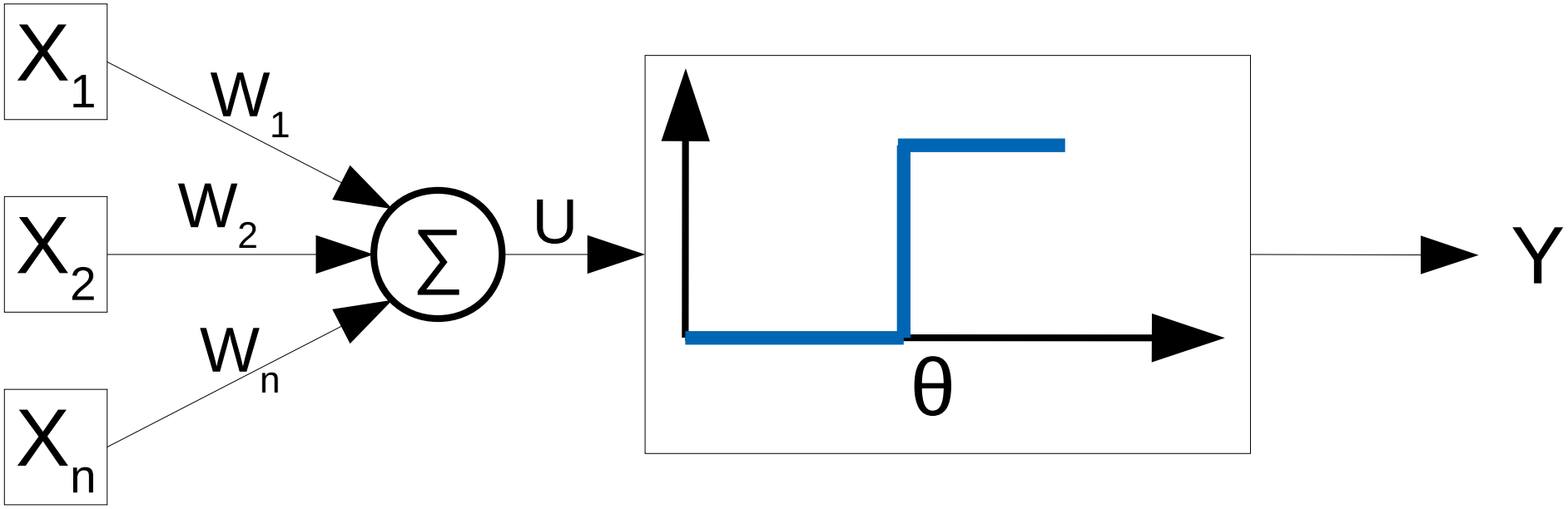
## Função Rampa




$$Y = \begin{cases} +1 & se\ (u \geq 1) \\ u & se\ (|u| < 1) \\ -1 & se\ (u \leq -1) \end{cases} \quad \text{onde,} \quad u = \left( \sum_{i-1}^{n} W_i \cdot X_i \right)$$

# Funções de ativação

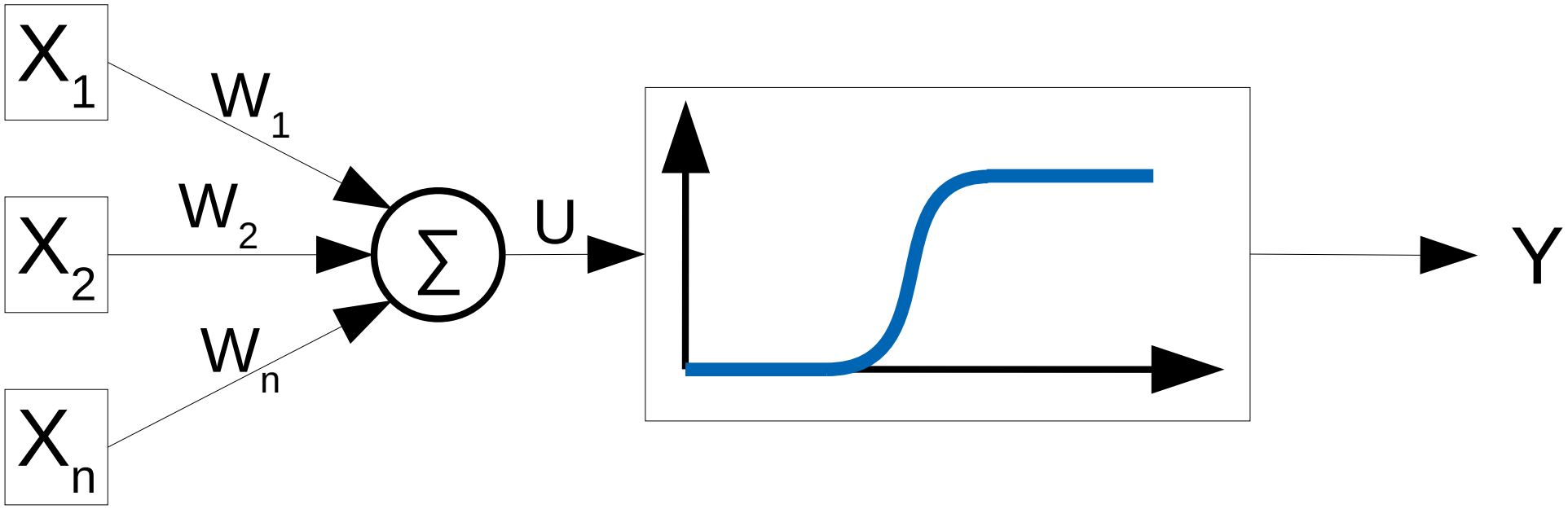
## Função Degrau




$$Y = \begin{cases} 1 & se\,(u \geq \theta) \\ 0 & se\,(u < \theta) \end{cases} \qquad \text{onde,} \qquad u = \left( \sum_{i-1}^{n} W_i \cdot X_i \right)$$

# Funções de ativação

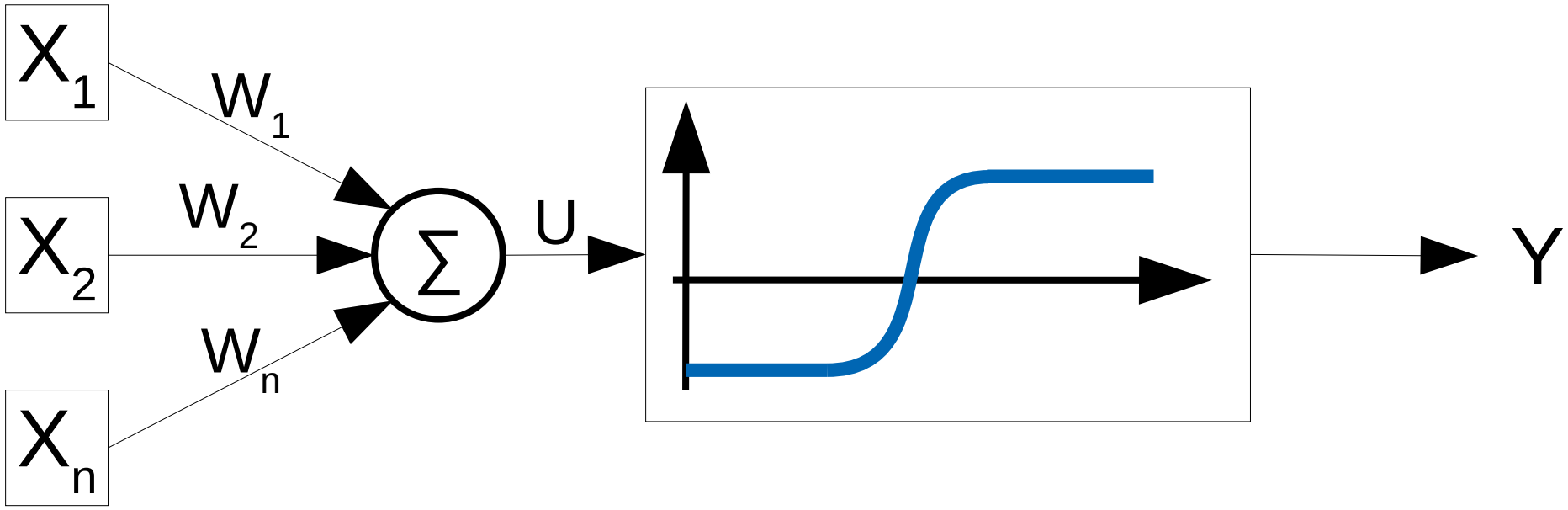
## Função Sigmoidal




$$Y = \frac{1}{1+e^{-u}} \qquad onde, u = \left( \sum_{i-1}^{n} W_i \cdot X_i \right)$$

# Funções de ativação

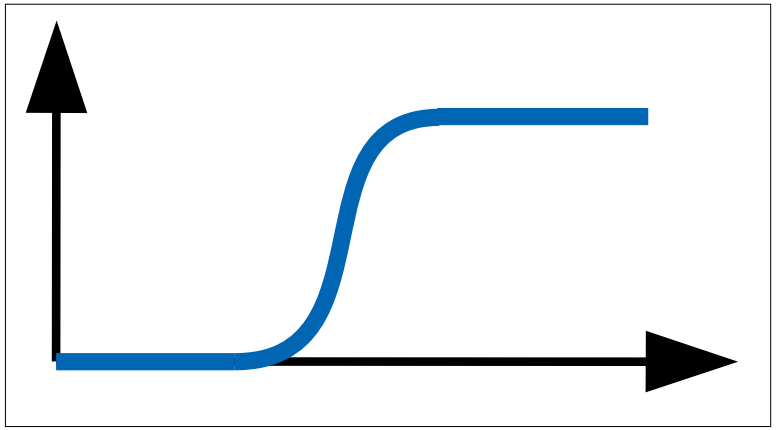
## Função Tangente Hiperbólica

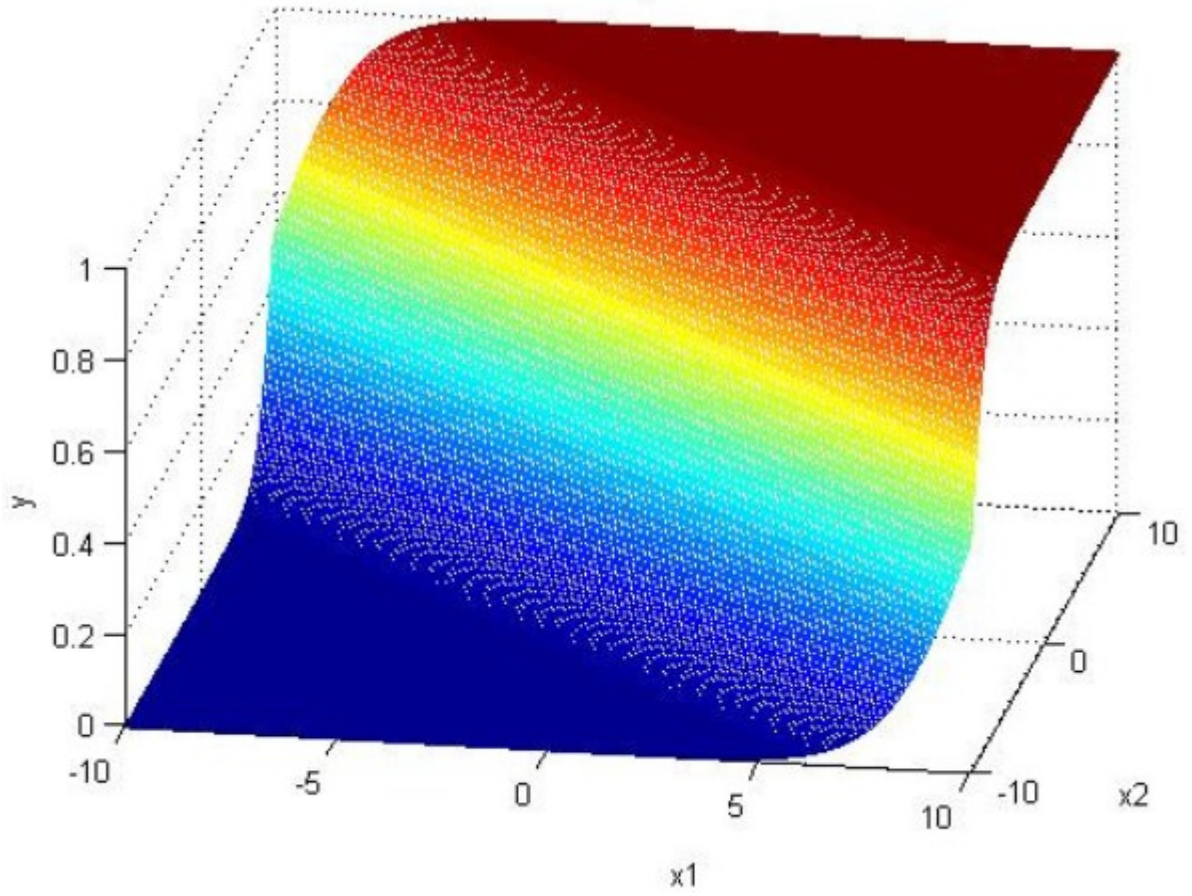



$$Y = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \qquad onde, u = \left( \sum_{i-1}^{n} W_i \cdot X_i \right)$$

# Funções de ativação

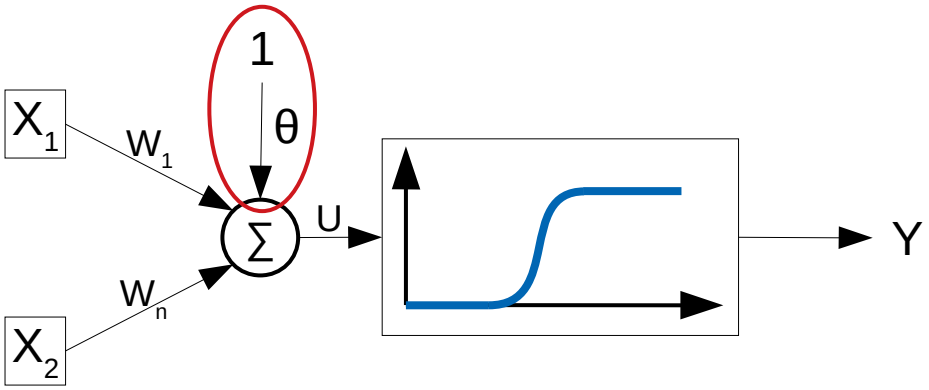
Mapeamento de entrada para saída em um neurônio com função de ativação sigmoidal.



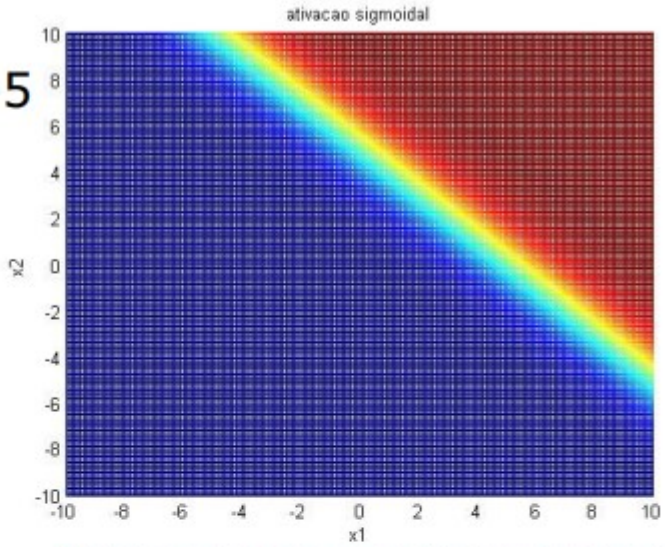


# Funções de ativação

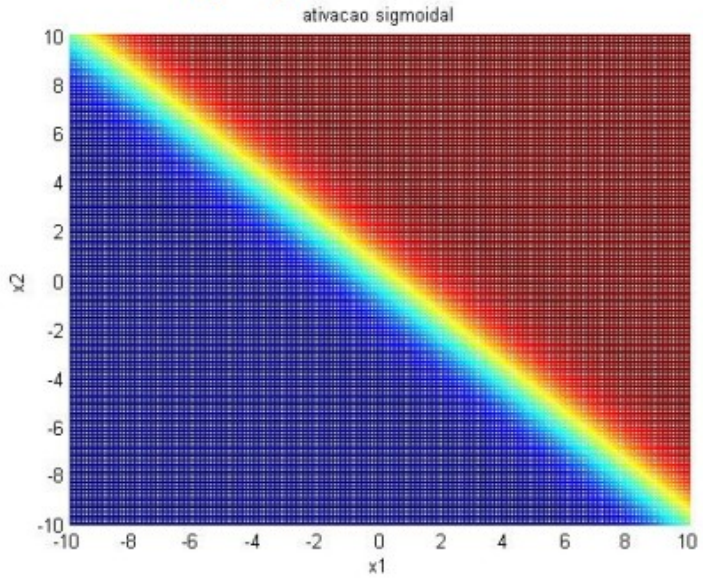
Uma entrada especial chamada **Bias** é utilizada para ajustar o limiar de atuação.



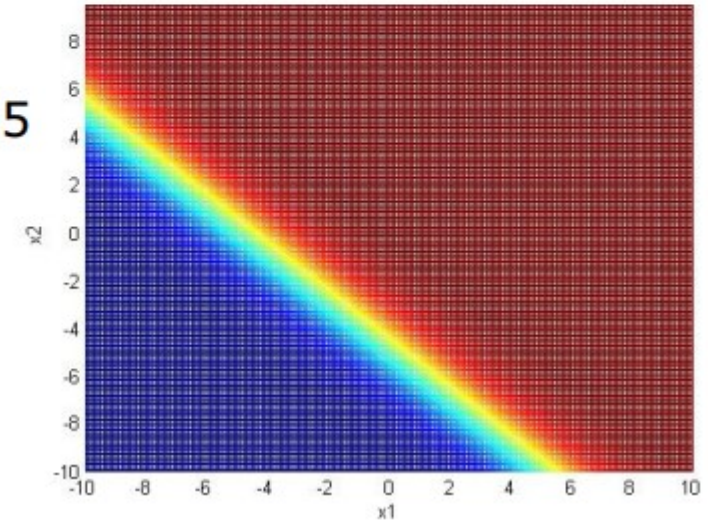

Θ =-5




Θ =0




Θ =+5



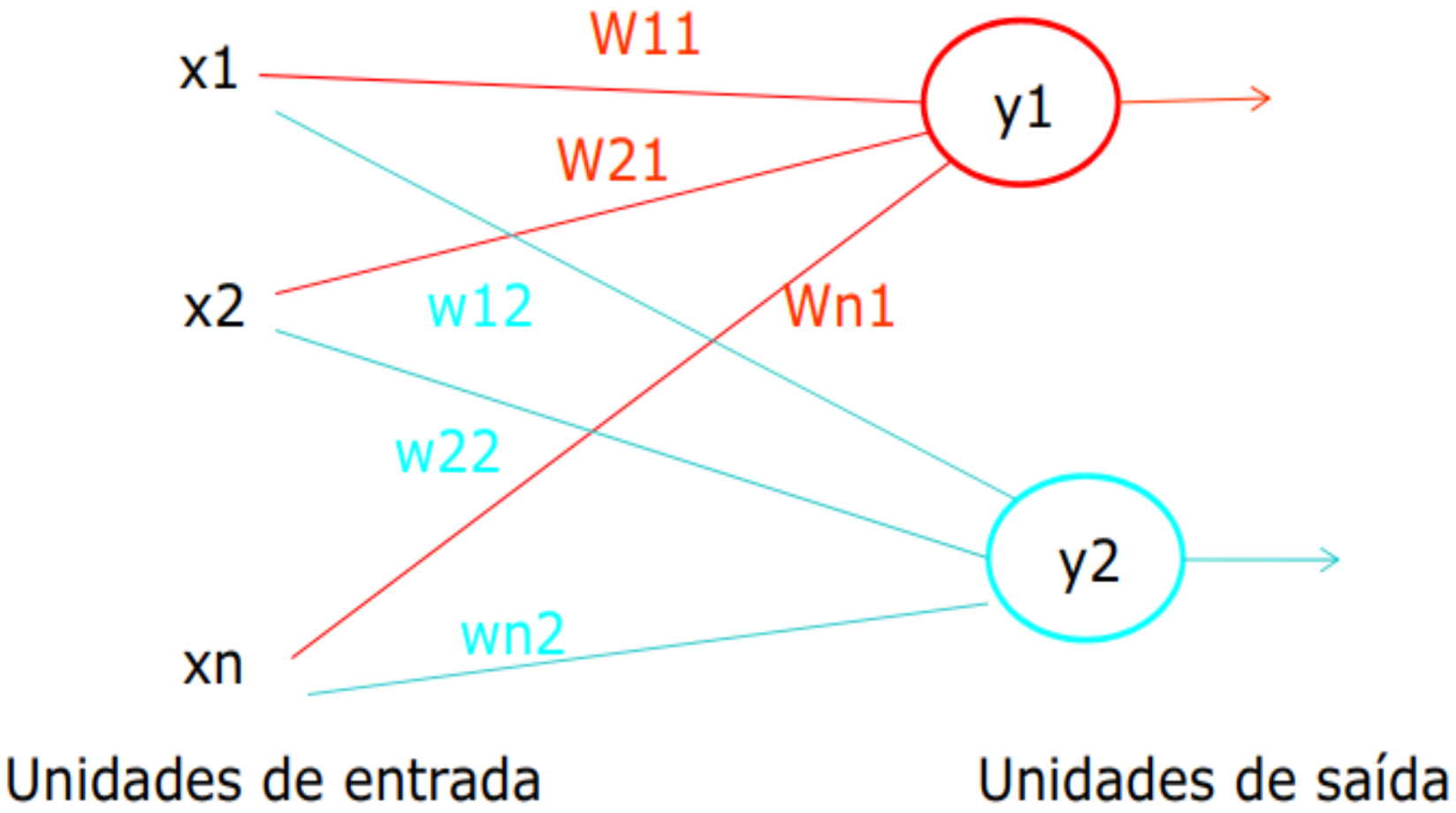
# Topologias da Redes Neurais

**Parâmetros que definem a arquitetura:**

- Número de camadas
- Número de neurônios em cada camada
- Tipo de conexão entre os neurônios
  - Feedforward
  - Feedback
- Conectividade da rede
  - Parcialmente conectada
  - Completamente conectada

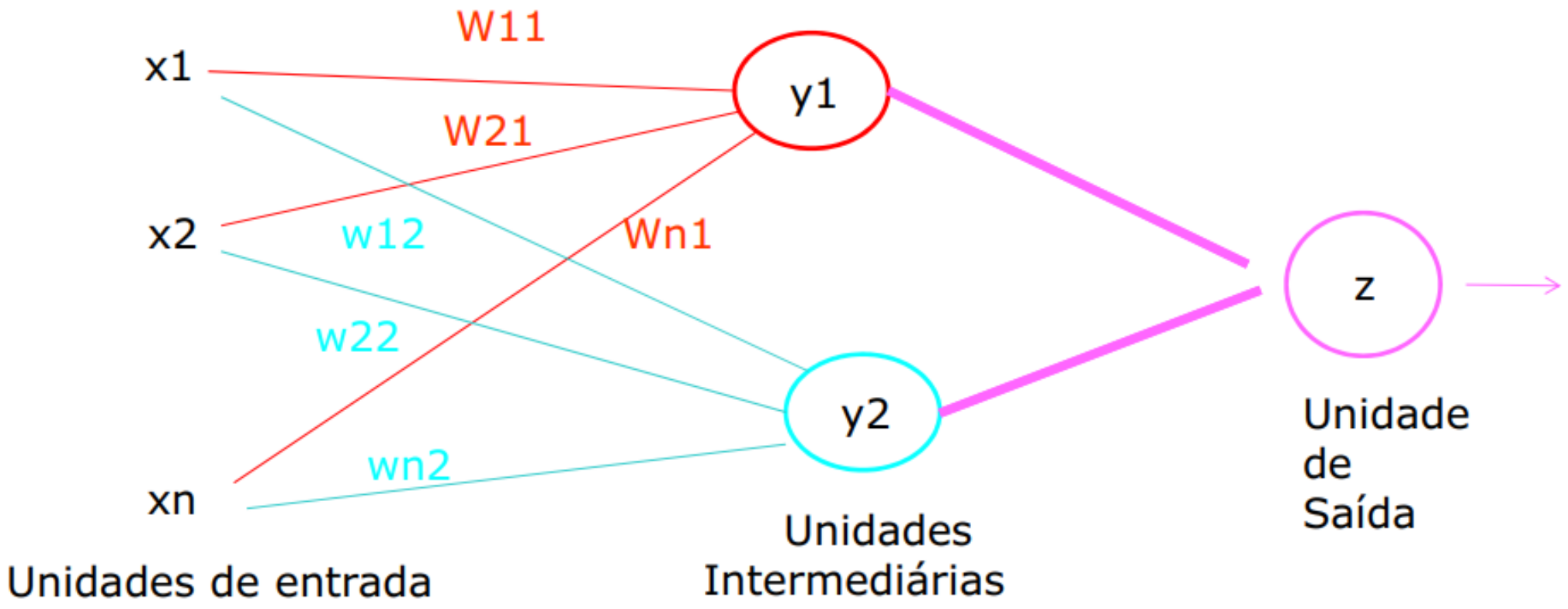
# Topologias da Redes Neurais

Redes de camada única **(Perceptron)**

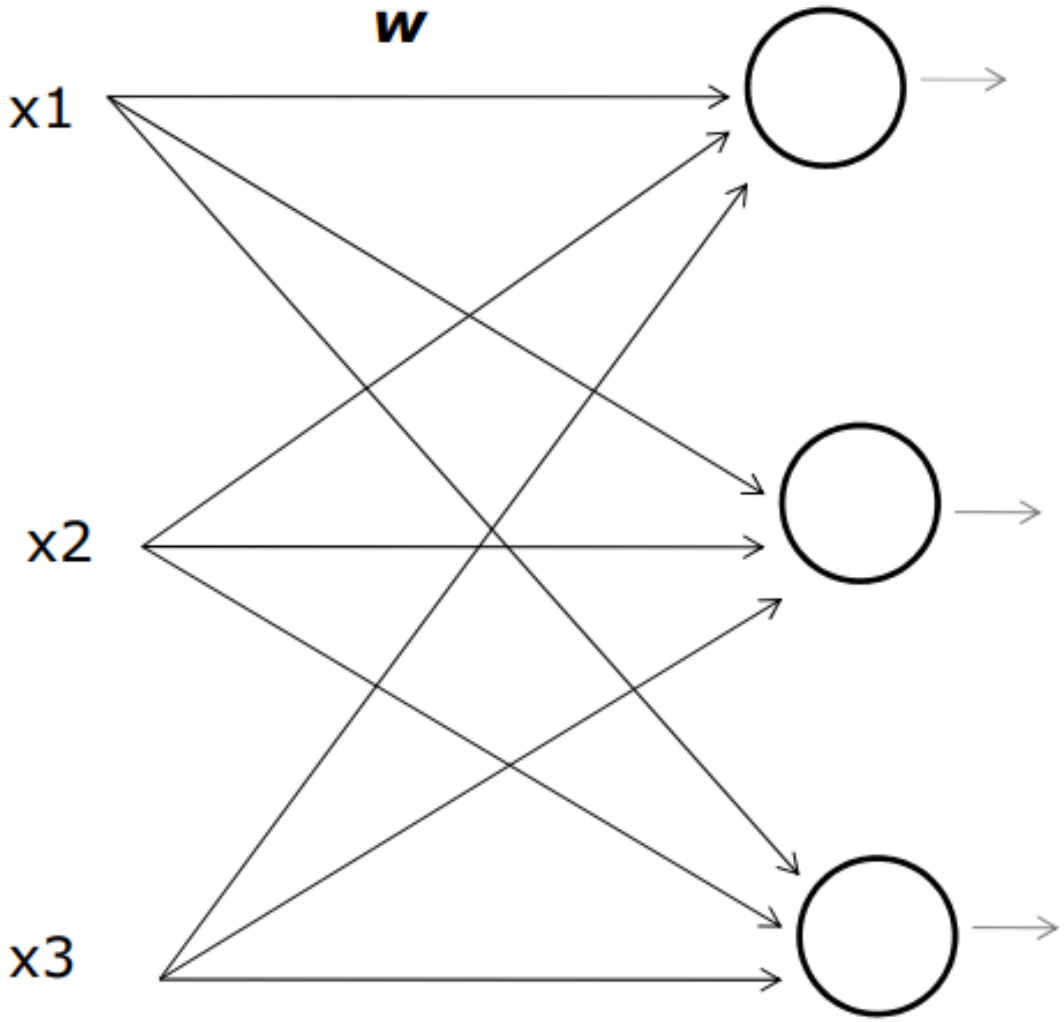


# Topologias da Redes Neurais

## Rede de Múltiplas Camadas
## **(Multilayer Perceptron – MLP)**



# Topologias da Redes Neurais



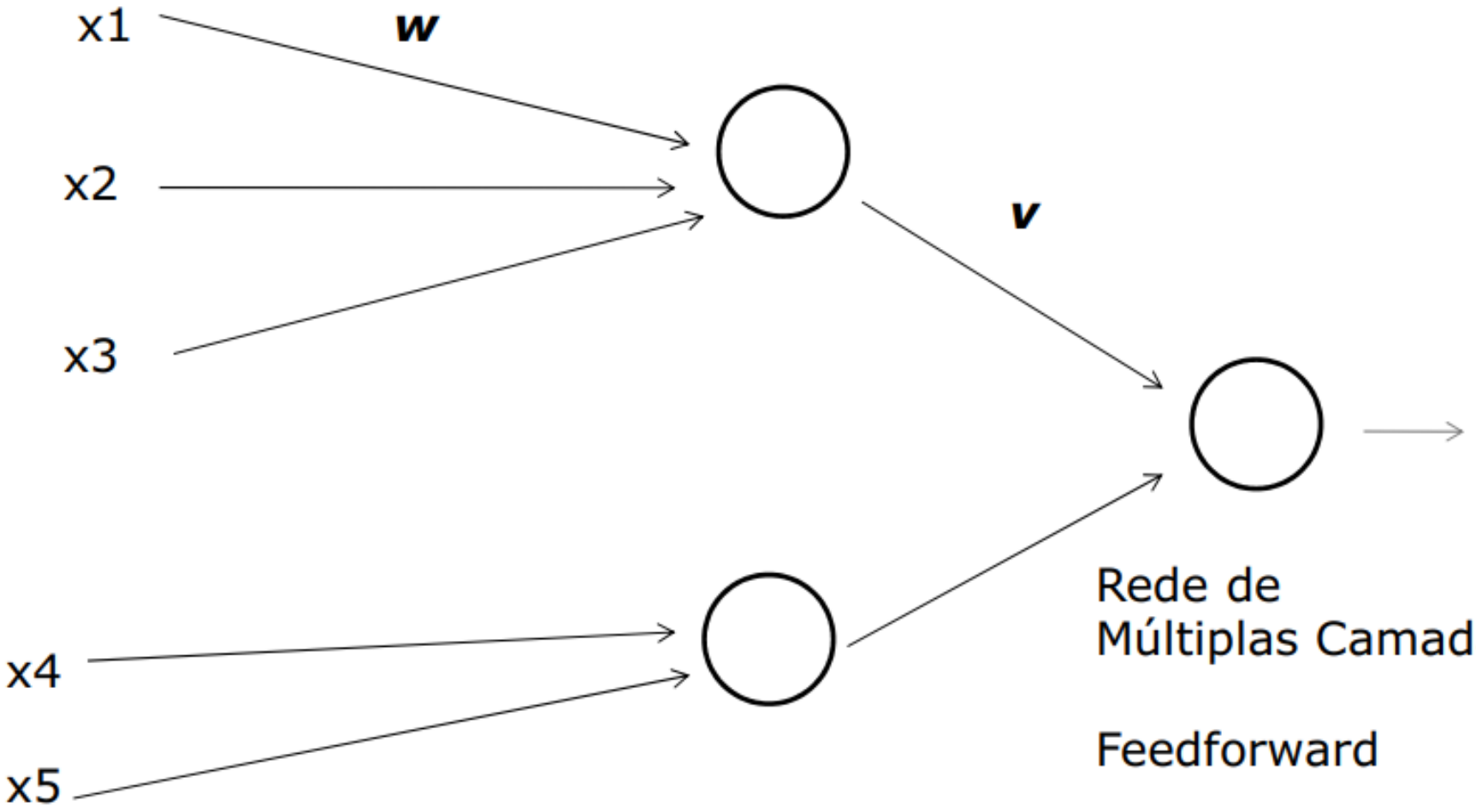

Rede de
Camada Única

Feedforward

Complemente
Conectada

Exemplo:
**Perceptron**

# Topologias da Redes Neurais

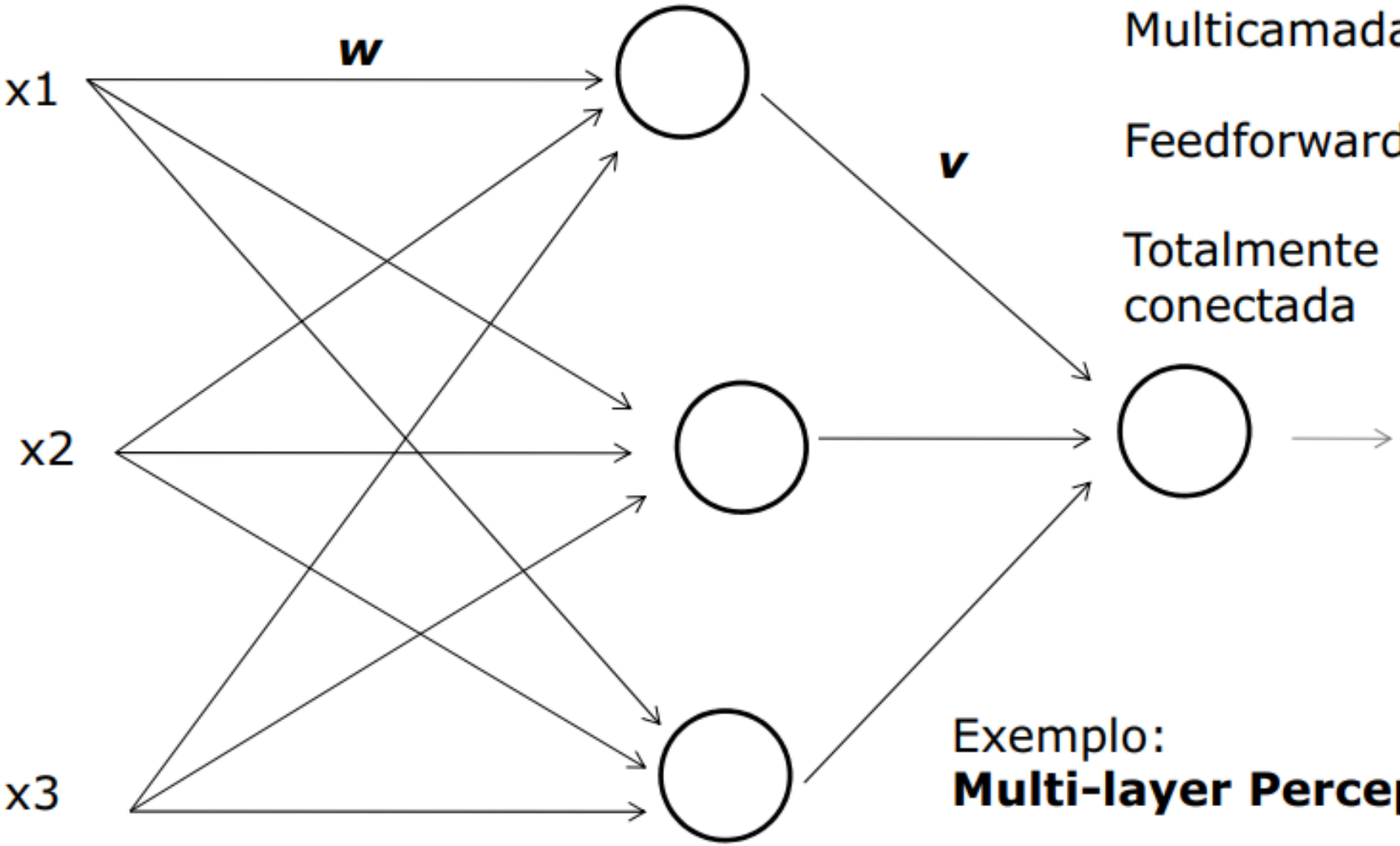



Rede de
Múltiplas Camadas

Feedforward

Parcialmente Concetada

# Topologias da Redes Neurais



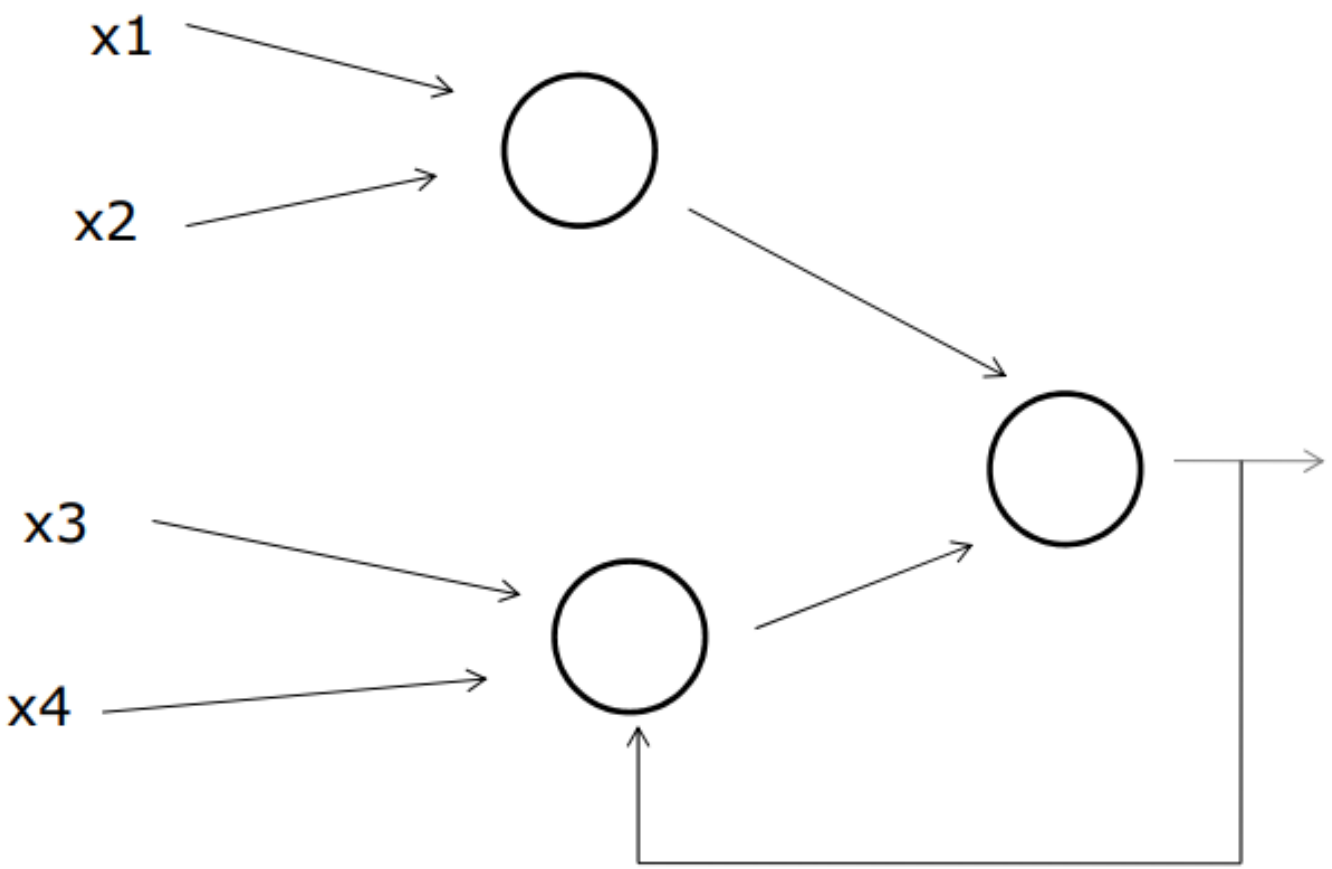

Rede Multicamadas

Feedforward

Totalmente conectada

Exemplo:
**Multi-layer Perceptron**

# Topologias da Redes Neurais

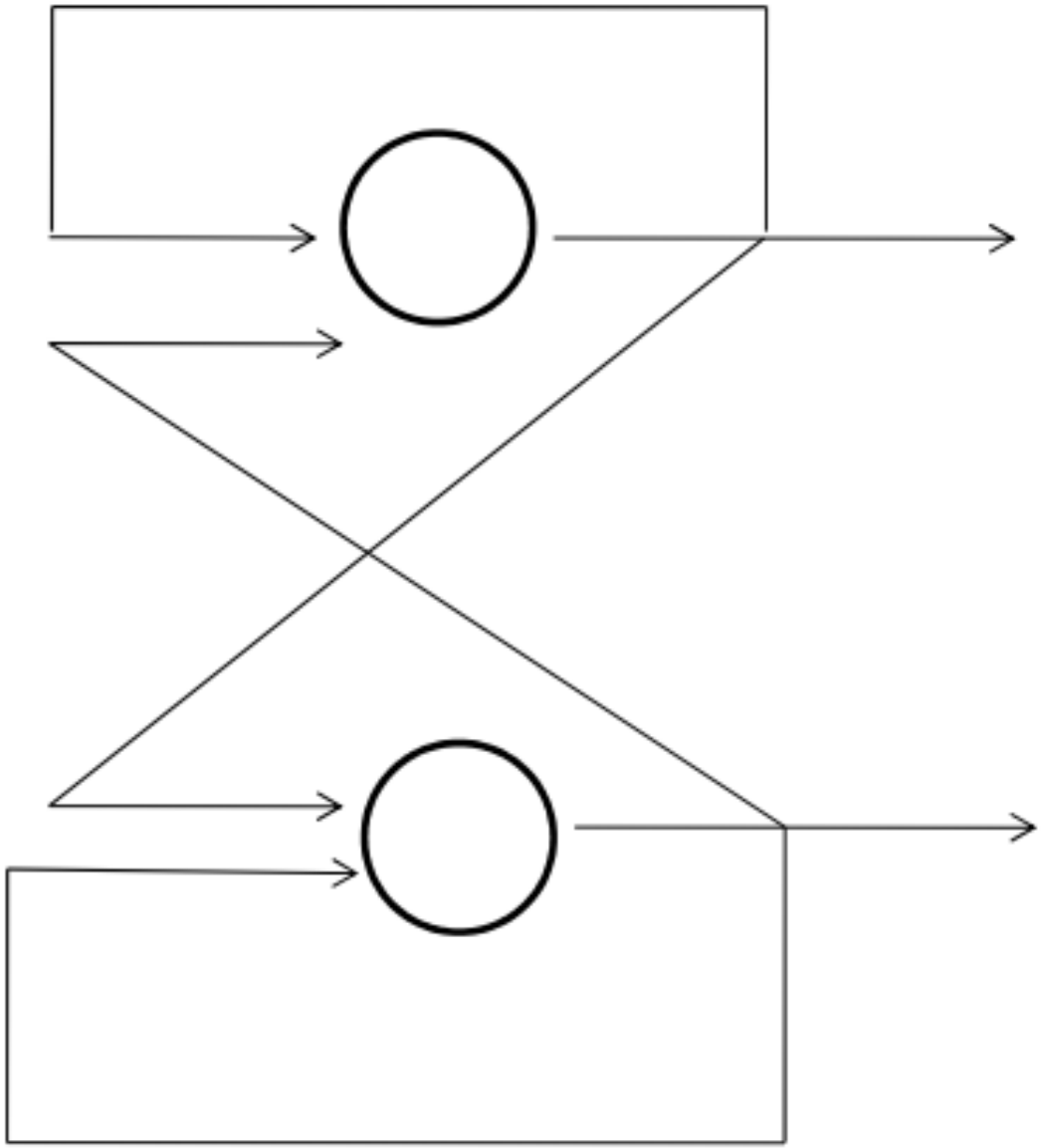



Rede de
Múltiplas Camadas

Recorrente

Parcialmente
Conectada

# Topologias da Redes Neurais



Rede de
Camada Única

Recorrente

Completamente
Conectada

# Treinamento das Redes Neurais

- Para realizar o treinamento das redes neurais é feito o ajuste dos parâmetros da rede.
- O treinamento é feito de forma iterativa, ajustando os pesos das conexões.
- O conhecimento adquirido pela rede fica armazenado nos pesos das conexões.
- Em outras palavras, aprendizagem em RNA é o processo de modificar os valores de pesos e do limiar.

# Treinamento das Redes Neurais

## Algoritmo de retropropagação

- O algoritmo de retropropagação ou back-propagation é aplicado nas RNAs cíclica (feedforward) com uma ou mais camadas intermediárias .
- Utiliza um método do gradiente descendente por correção de erro.
- Uma função de custo é minimizada realizando-se iterativamente ajustes nos pesos sinápticos de acordo com o erro quadrático acumulado para todos os padrões do conjunto de treinamento

# Treinamento das Redes Neurais

Para aplicar o algoritmo de retropropagação é necessário:

- Um conjunto de padrões de treinamento com dados de entrada e saída desejada
- Um valor para a taxa de aprendizado
- Um critério que finalize o algoritmo (por nº de ciclos, épocas ou por erro)
- Uma metodologia para atualizar os pesos ($\Delta w$)
- Uma função de transferência não-linear
- Valores de pesos iniciais

# Treinamento das Redes Neurais

Com o algoritmo de aprendizado por retropropagação a rede basicamente aprende um conjunto pré-definido de pares de exemplos (entrada e saída) em ciclos de propagação e adaptação.

Toda vez que um padrão de entrada é aplicado como um estímulo aos elementos da primeira camada da rede, ele é propagado por cada uma das outras camadas até que a saída seja gerada.

Este padrão de saída é então comparado com a saída desejada e um sinal de erro é calculado para cada elemento de saída.

# Treinamento das Redes Neurais

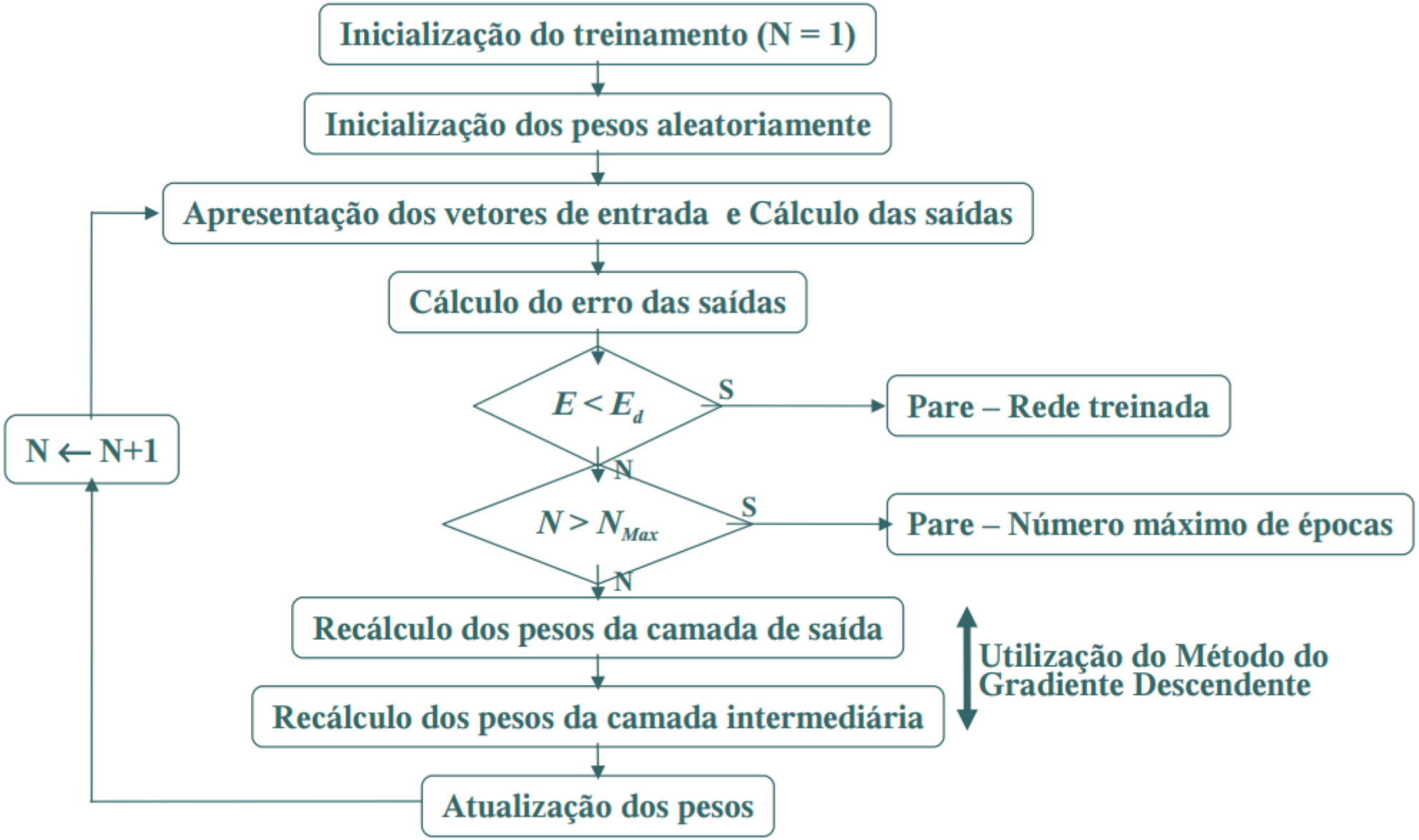
Então o sinal de erro é retropropagado da camada de saída para as camadas intermediárias.

Porém, cada elemento da camada intermediária recebe apenas uma porção do sinal de erro total.

Este processo se repete, camada por camada, até que cada elemento da rede receba um sinal de erro relativo ao erro total .

Baseado no sinal de erro recebido, os pesos das conexões são, então, atualizados de modo a fazer a rede convergir.

# Treinamento das Redes Neurais



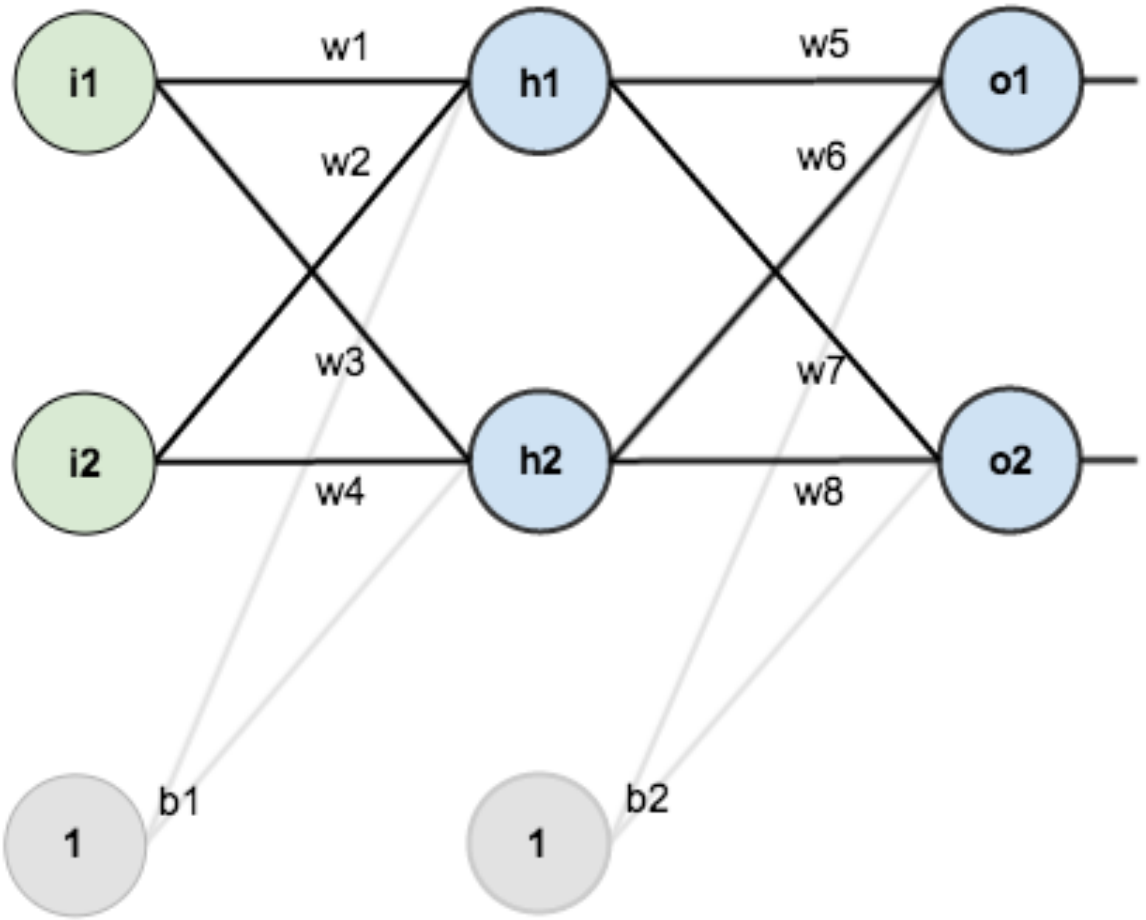
# Treinamento das Redes Neurais

## Critérios de parada

- Finalizar o treinamento após n ciclos
- Finalizar o treinamento após o erro quadrático médio
- Ficar abaixo de uma constante α
- Finalizar o treinamento quando a porcentagem de classificações corretas estiver acima de uma constante α (mais indicado para saídas binárias)
- Combinação dos métodos acima

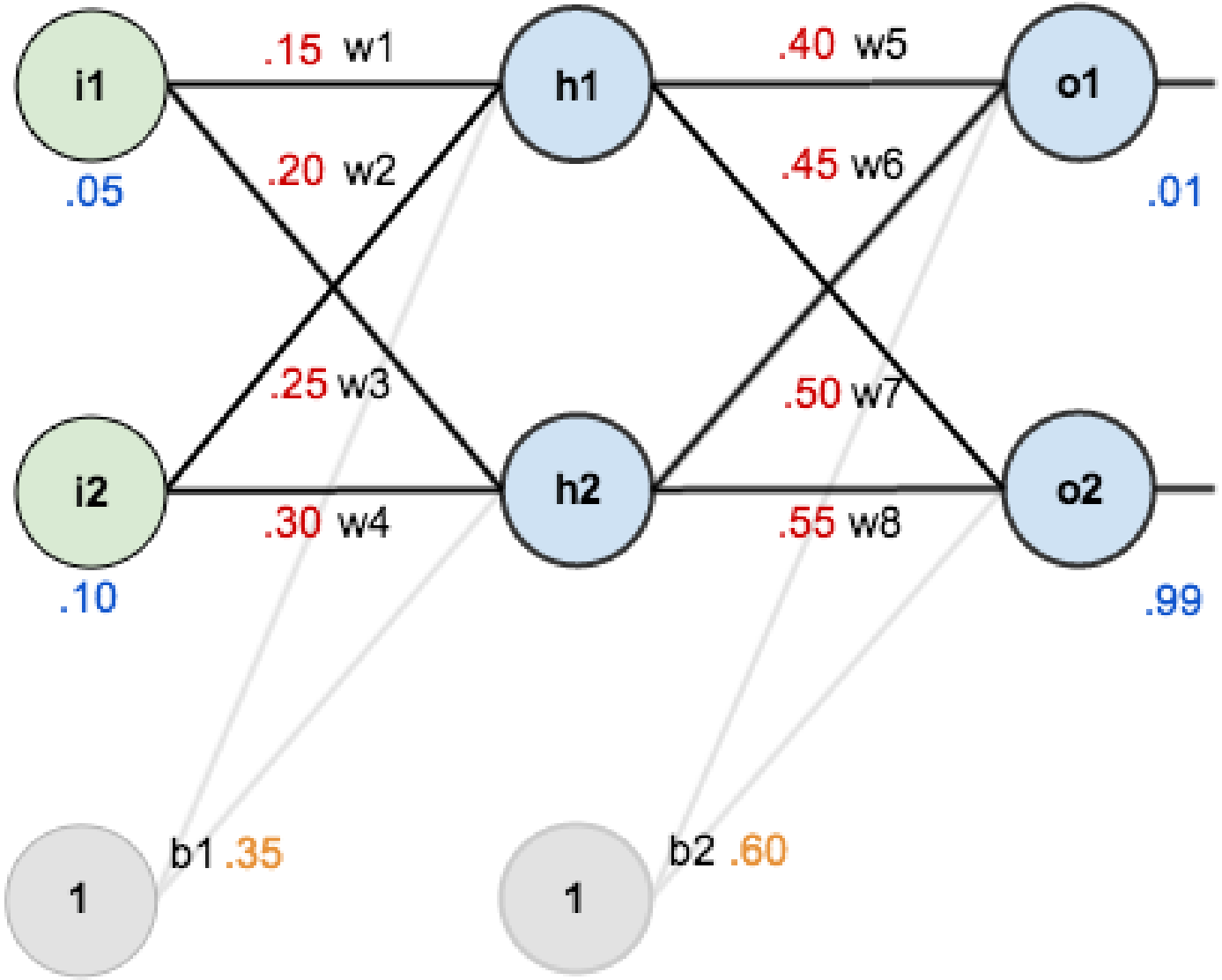
# Treinamento das Redes Neurais

## Exemplo Backpropagation

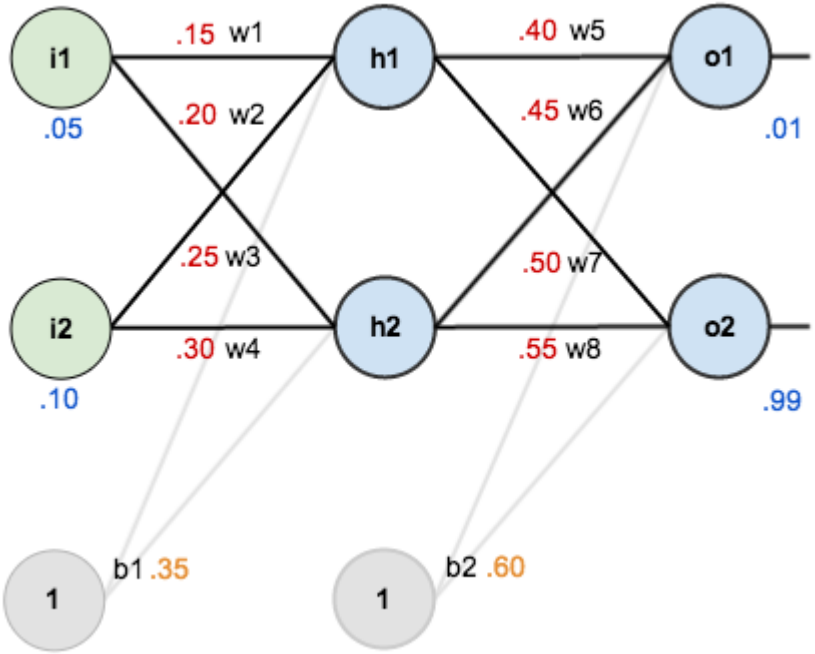



Adaptado de Matt Mazur (https://mattmazur.com)

# Treinamento das Redes Neurais



# Treinamento das Redes Neurais




Calculando a saída para as entradas 0.05 and 0.10.

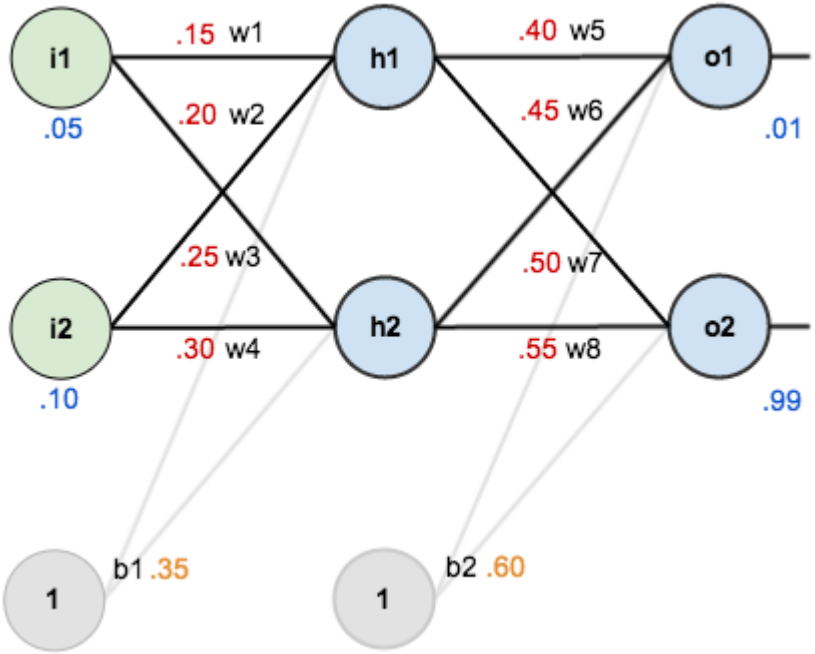
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

$$out_{h2} = 0.596884378$$

# Treinamento das Redes Neurais




Calculando a saída para as entradas 0.05 and 0.10.

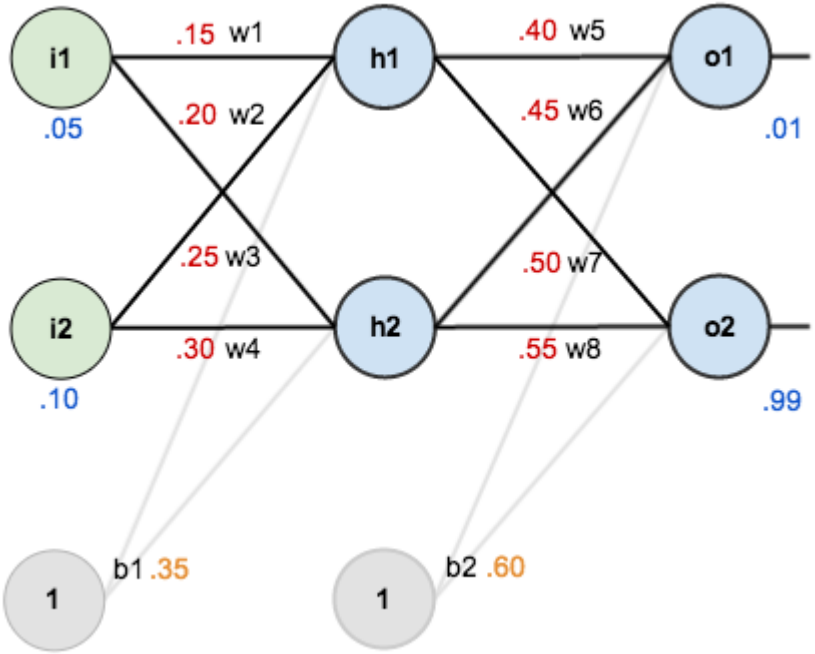
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$out_{o2} = 0.772928465$$

# Treinamento das Redes Neurais




Calculando o erro total

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Considerando que o1 deveria ser 0.01 e a saída atual é 0.75136507,temos:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

# Treinamento das Redes Neurais

**Realizando a retropropagação**

Considerando por exemplo o peso w5, desejamos saber quanto uma mudança neste peso afeta o erro total. Ou:

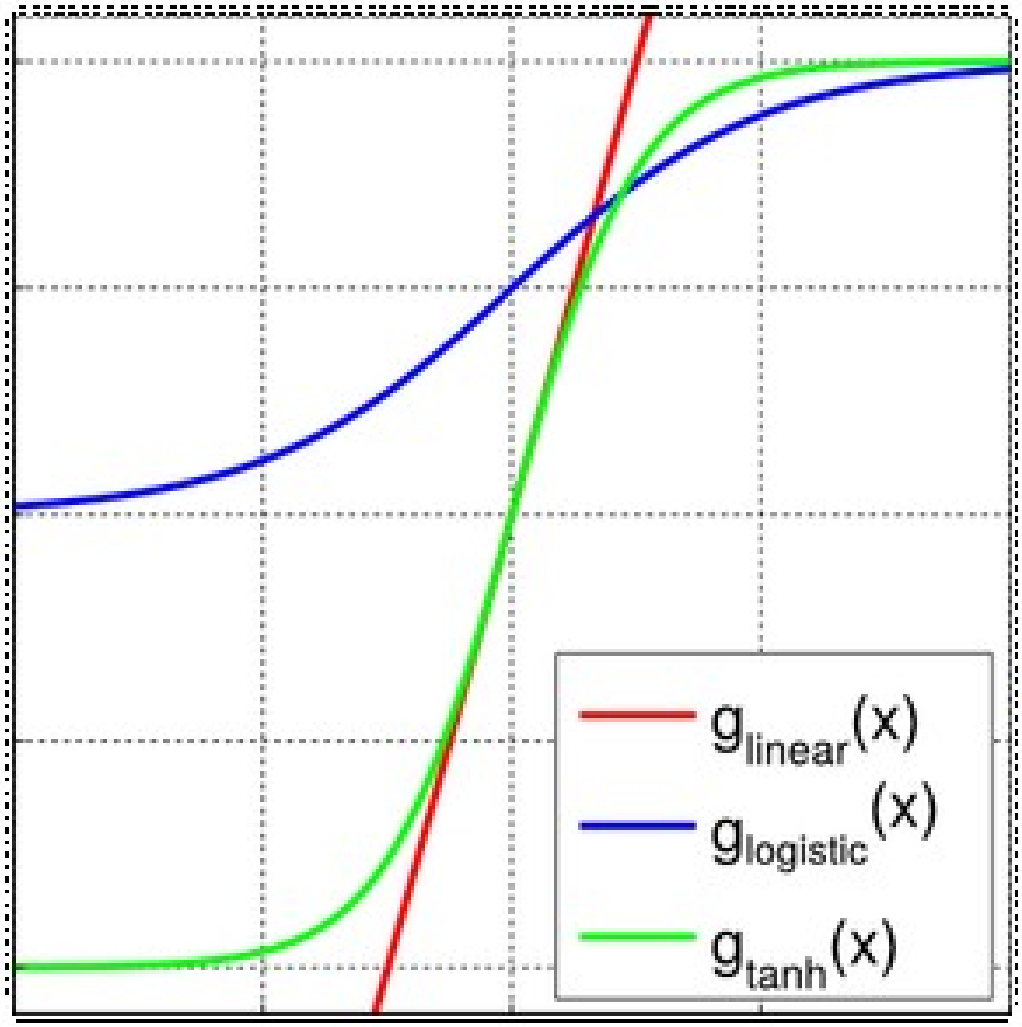
$$\frac{\partial E_{total}}{\partial w_5}$$

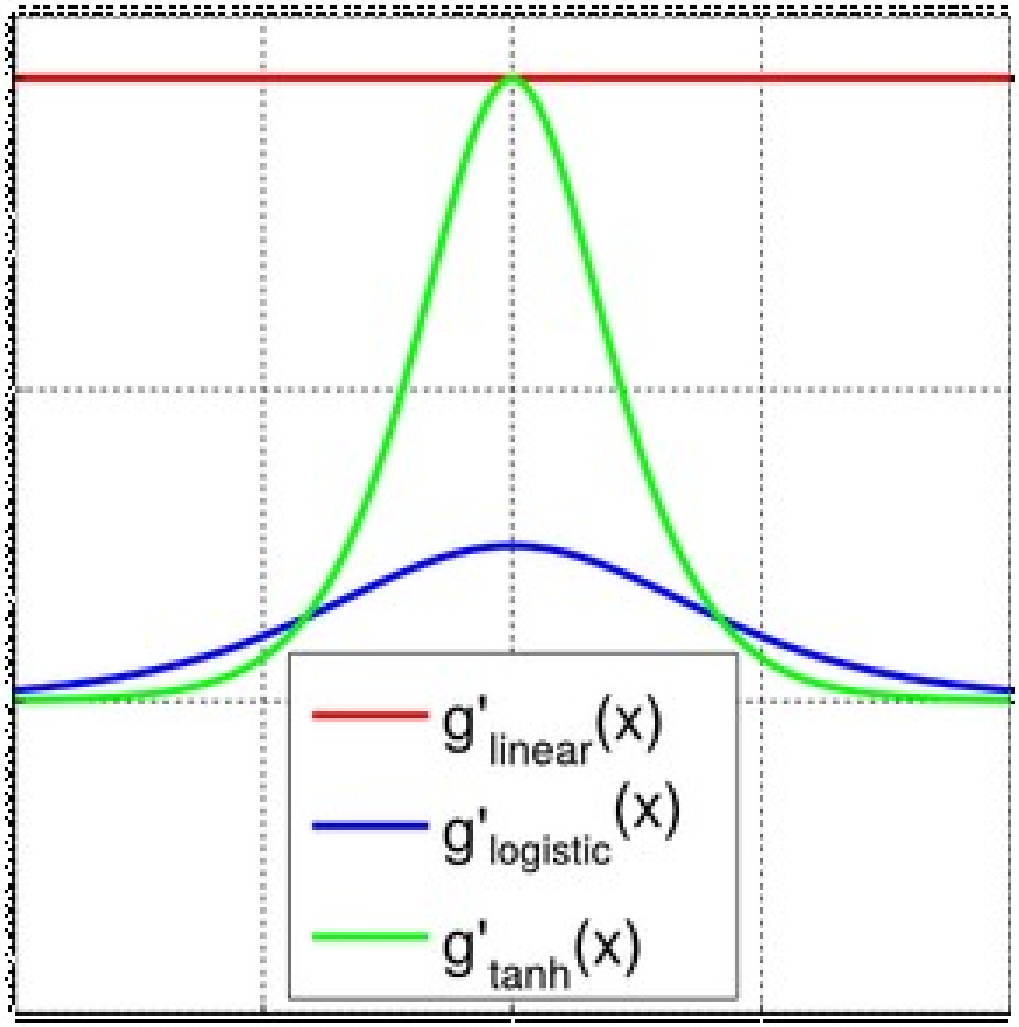
Esta razão é obtida por:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

# Treinamento das Redes Neurais

Derivadas das funções de ativação.

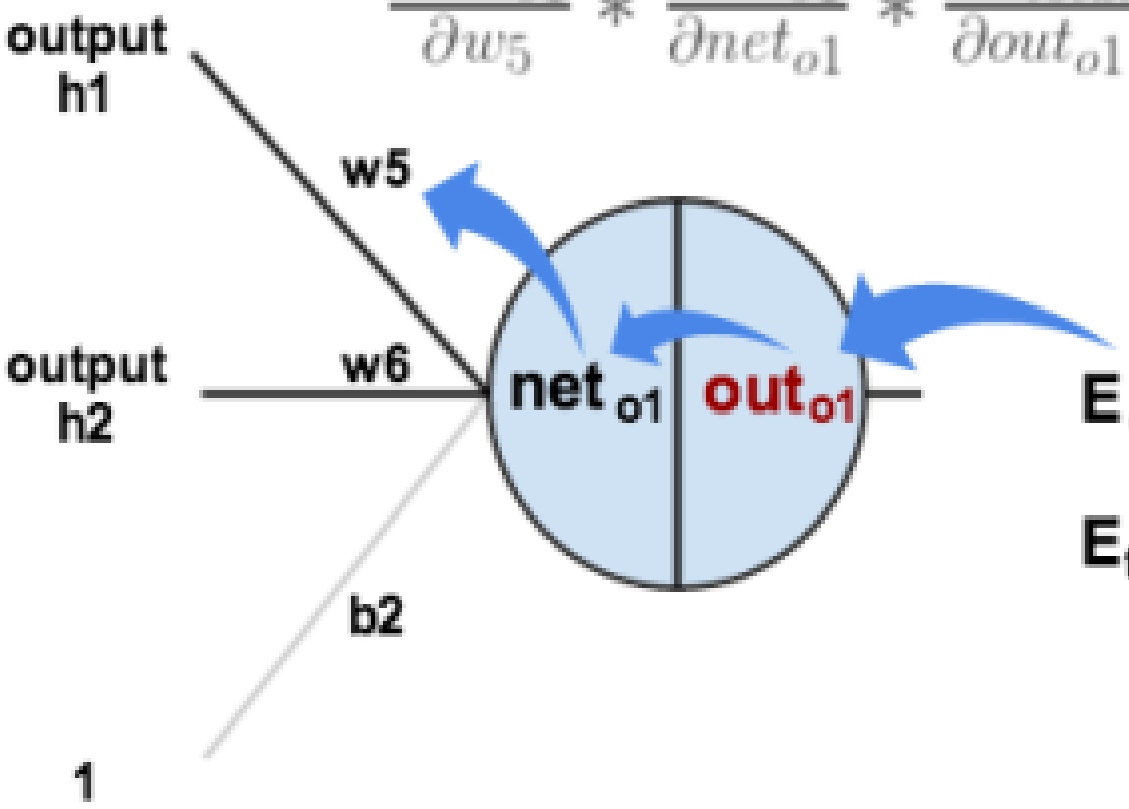


# Treinamento das Redes Neurais

Ou seja:

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$




$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$$

$$E_{total} = E_{o1} + E_{o2}$$

# Treinamento das Redes Neurais

Calculando cada um dos termos:

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$out_{o1} = \frac{1}{1+\epsilon^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

# Treinamento das Redes Neurais

Juntando tudo:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

Em uma forma combinada temos:

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

Para reduzir o erro é necessário subtrair este valor do peso atual desta sinapse, considerando uma taxa de aprendizado η (neste exemplo η = 0,5).

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

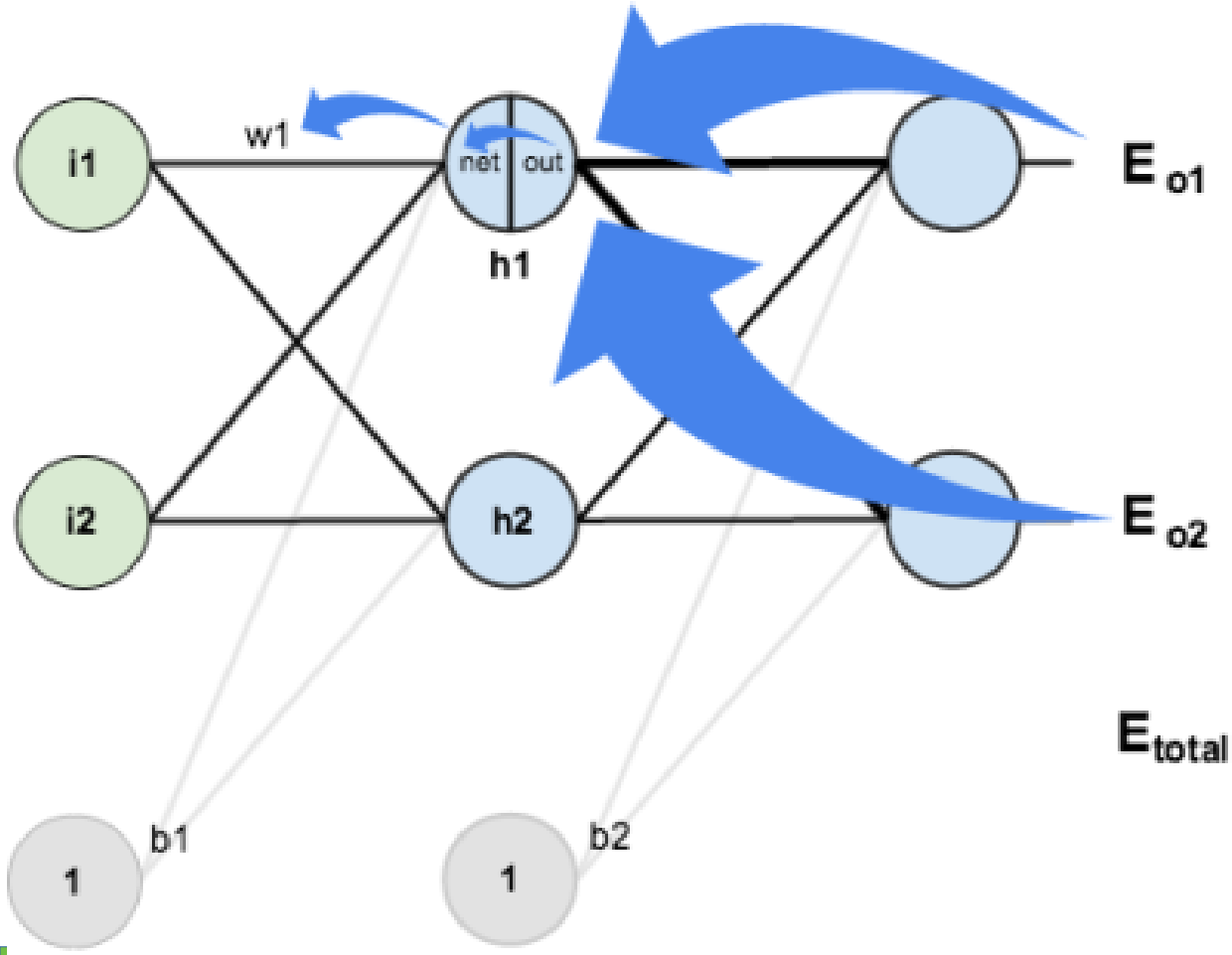
$$w_6^+ = 0.408666186 \quad w_7^+ = 0.511301270 \quad w_8^+ = 0.561370121$$

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

# Treinamento das Redes Neurais

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$




$E_{o1}$

$E_{o2}$

$E_{total} = E_{o1} + E_{o2}$

Antes de atualizar os pesos da camada de saída é necessário calcular os novos pesos para a camada oculta (w1, w2, w3 e w4).

# Treinamento das Redes Neurais

Como cada neurônio da camada oculta contribui para a saída de vários neurônios da camada de saída a forma de calcular o novo peso é diferente.

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

Usando os valores calculados anteriormente:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

# Treinamento das Redes Neurais

Considerando que $\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$

Temos:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

De forma similar:

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

Portanto:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

# Treinamento das Redes Neurais

Agora é necessário obter: $\frac{\partial out_{h1}}{\partial net_{h1}}$ e então $\frac{\partial net_{h1}}{\partial w}$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

Calculando para h1, assim como calculado para a saída temos:

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

# Treinamento das Redes Neurais

Juntado tudo temos:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

Agora é possível atualizar w1:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Similarmente:

$$w_2^+ = 0.19956143 \quad w_3^+ = 0.24975114 \quad w_4^+ = 0.29950229$$

# Treinamento das Redes Neurais

Por fim, atualizamos todos os pesos.

Quando aplicamos as entradas de **0,05** e **0,1** originalmente, o erro na rede foi de **0,298371109.**

Após essa primeira rodada de retropropagação, o erro total caiu para **0,291027924**.

Pode não parecer muito, mas depois de repetir este processo 10.000 vezes, por exemplo, o erro cai para 0.0000351085.

Neste ponto, quando avançamos 0,05 e 0,1, os dois neurônios gerados geram **0,015912196** (contra 0,01) e **0,984065734** (contra 0,99).

# Aplicações das Redes Neurais

- Diagnóstico: Médico, Falhas de Sistemas etc.
- Previsão de Séries Temporais: Cotações da Bolsa de Valores, Dados Econômicos, Metereologia etc.
- Processamento de Linguagem Natural
- Sistemas de Controle e Automação
- Reconhecimento e Síntese de Voz
- Processamento de Sinais e Imagens: Radar, Sensores, Imagens de satélite etc.
- Reconhecimento ótico de caracteres (OCR)
- Controle de processos industriais
- Piloto automático
- Reprodução da fala

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

# Vantágens das Redes Neurais

- Aplicações em Sistemas Adaptativos
- Aplicadas em tarefas onde temos bases de exemplos disponíveis, realizando a aquisição automática de conhecimentos
- Associação de padrões de entradas e saída
- Classificação de padrões de forma supervisionada ou não
- Aproximação de funções desconhecidas através de amostras destas funções
- Trabalhar com dados aproximados, incompletos e inexatos
- Paralelismo, generalização, robustez
- “Tarefas complexas realizadas por seres humanos”

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

# Limitações das Redes Neurais

- Composição e construção de conhecimentos estruturados
- Dificuldade de explicitação dos conhecimentos adquiridos
- Dificuldade para definir a estrutura da rede, seus parâmetros e a base de dados
- Falta de garantia de uma convergência do algoritmo para uma solução ótima

# Exemplos

https://www.youtube.com/watch?v=RcgfwZ-FFBI

https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi