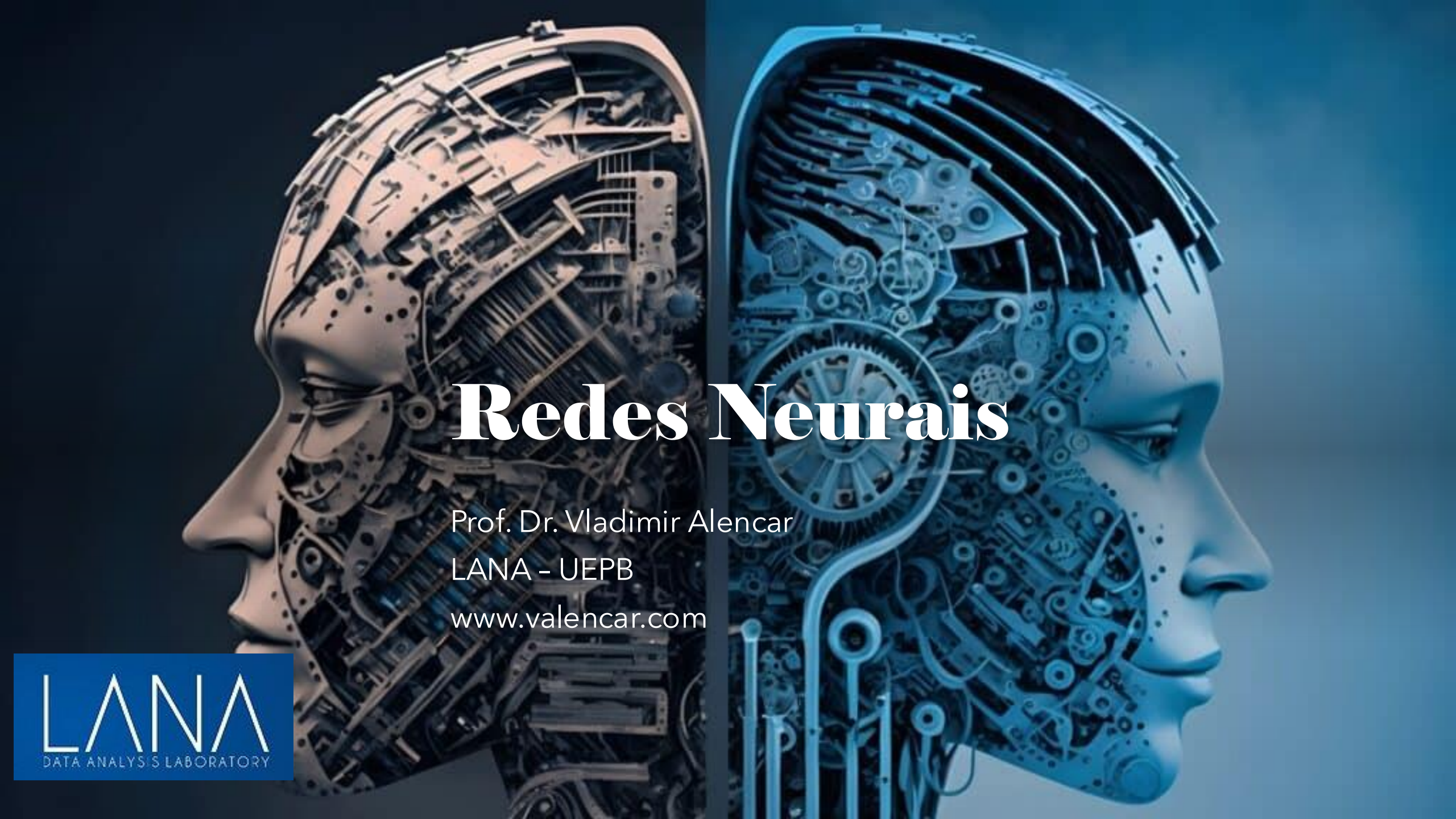


Redes Neurais

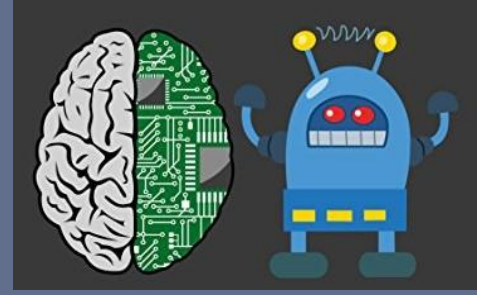
Prof. Dr. Vladimir Alencar

LANA – UEPB

www.valencar.com

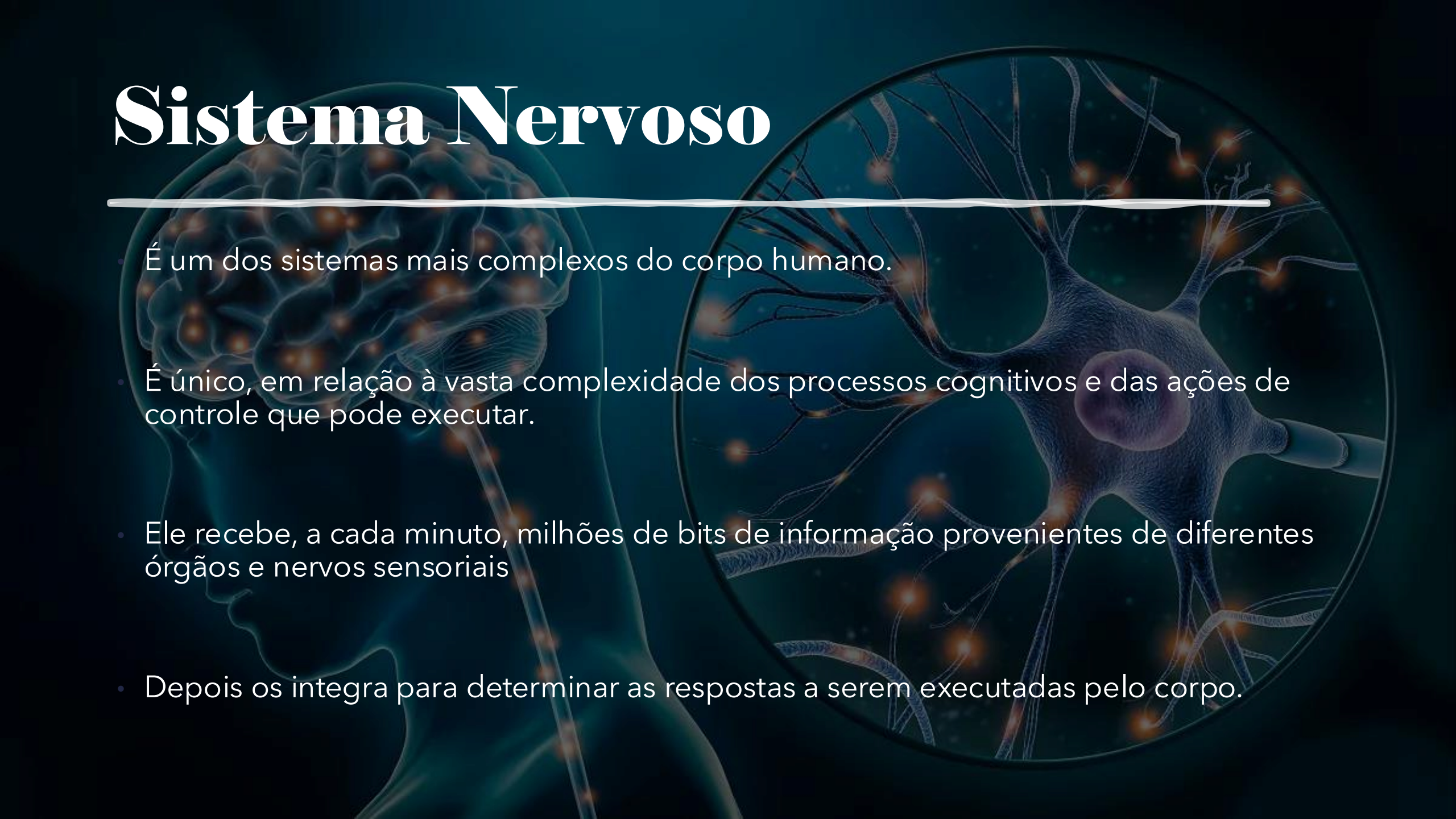
LANA
DATA ANALYSIS LABORATORY

Redes Neurais



- O grande responsável pelo processamento de informações e geração de respostas é o **cérebro humano**
- Tarefas cotidianas, como caminhar, pegar um objeto, envolve a ação de diversos componentes, como: memória, aprendizado e coordenação motora.
- A realização dessas tarefas se deve à nossa complexa estrutura biológica

Sistema Nervoso



- É um dos sistemas mais complexos do corpo humano.
- É único, em relação à vasta complexidade dos processos cognitivos e das ações de controle que pode executar.
- Ele recebe, a cada minuto, milhões de bits de informação provenientes de diferentes órgãos e nervos sensoriais
- Depois os integra para determinar as respostas a serem executadas pelo corpo.

Sistema Nervoso

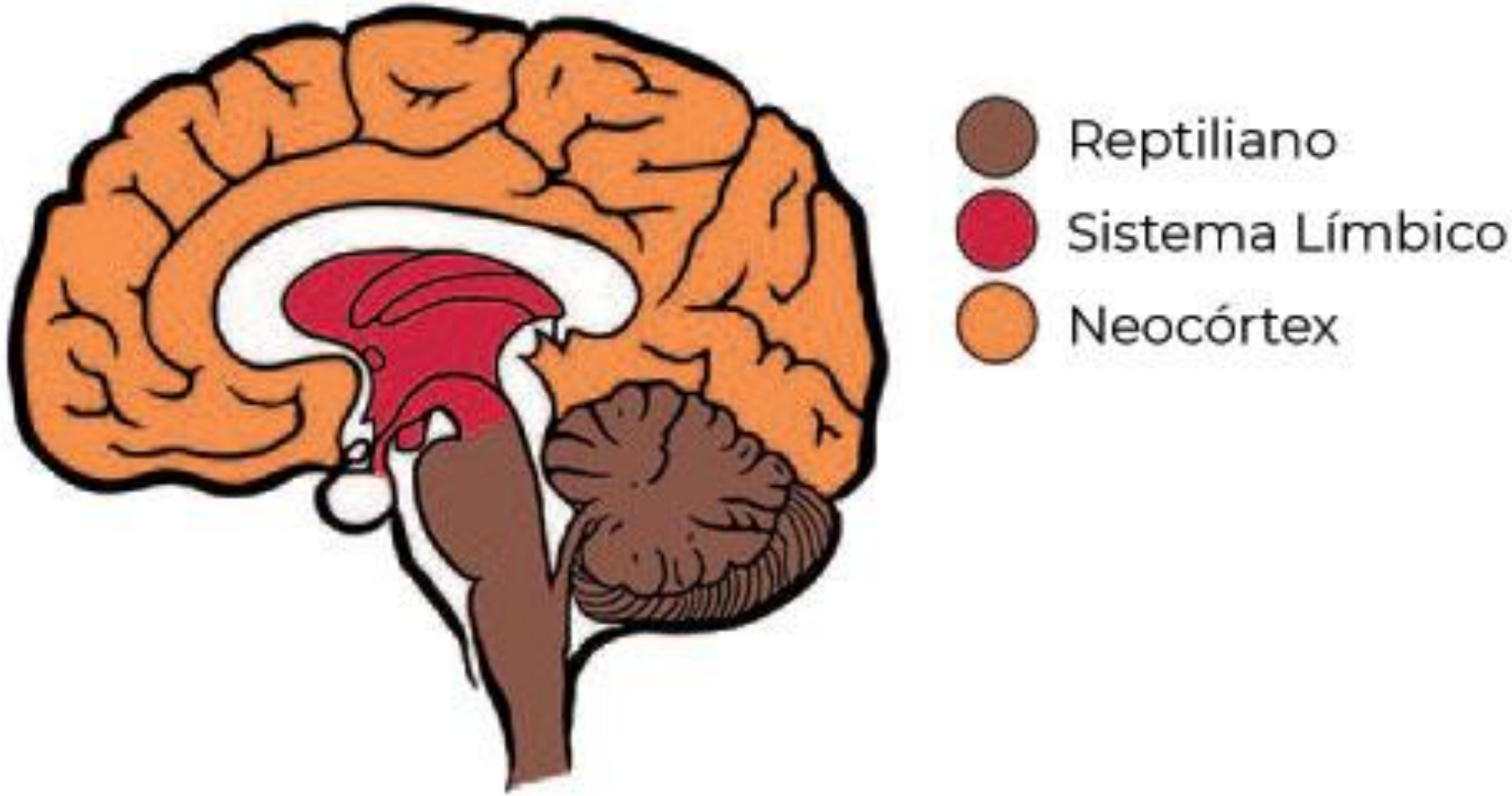
- **Sensitiva**
  - Através de órgãos sensorias como visão, audição, tato, olfato, paladar

- **Integradora**
  - Processamento dos sinais do organismo para o cérebro

- **Motora**
  - Através da ativação da Contração muscular ou da Secreção glandular de substâncias

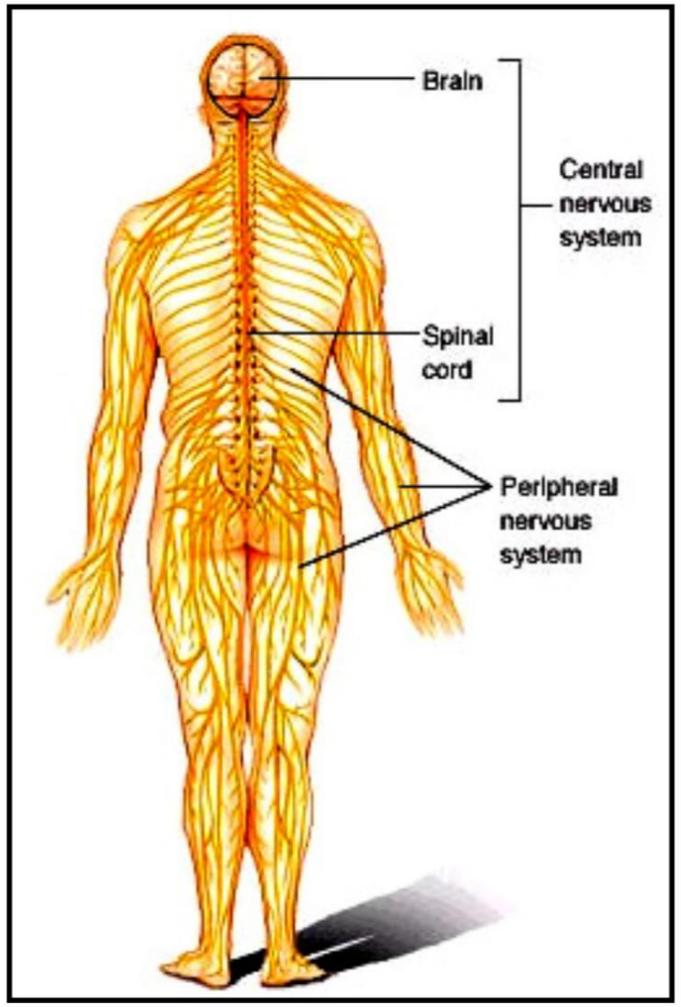
- Ex. Estímulo de calor no braço => Córtex Cerebral => Resposta => Reação Motora (contração do braço)

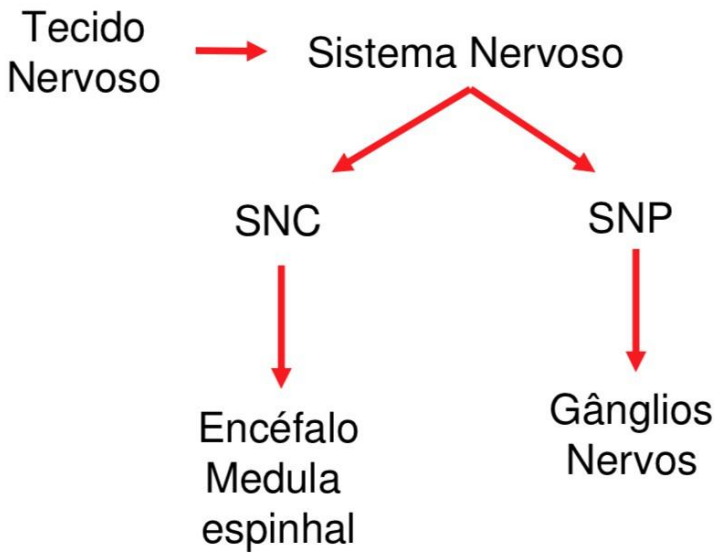
O Cérebro Trino de MacLean



Sistema Nervoso

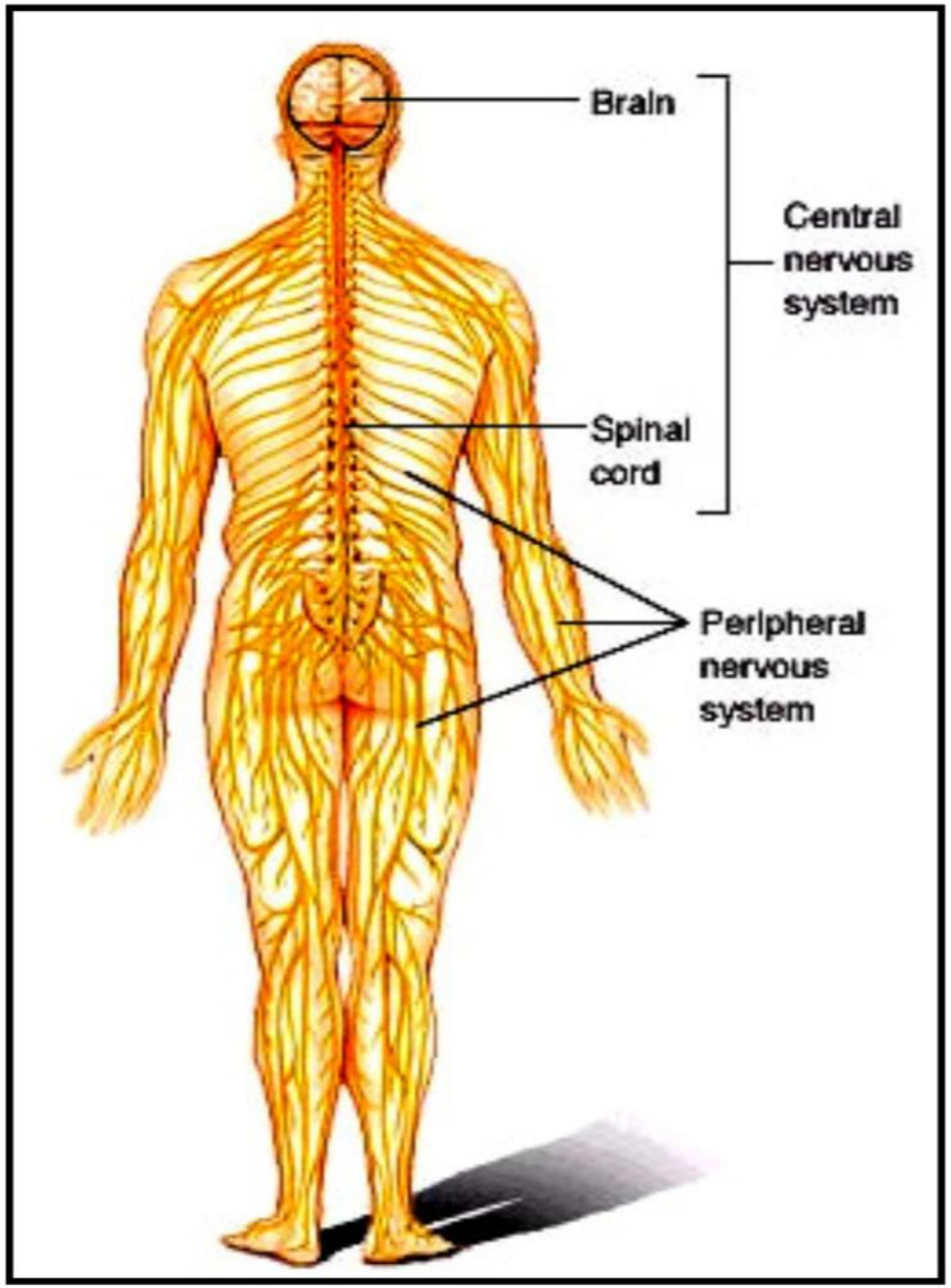
- O encéfalo é uma região pertencente ao sistema nervoso, e engloba o cérebro, tálamo, mesencéfalo, ponte cerebelo e bulbo.
- Pesa 1,4kg e sua função principal é processar e integrar as informações nervosas do corpo humano.
- O encéfalo se parece com um computador, porém, além do processamento lógico, ele é capaz de um desenvolvimento complexo, aprendizado, autoconciência, emoção e criatividade.
- A cada segundo, milhões de sinais químicos e elétricos passam pelo encéfalo e pela intricada rede de nervos do corpo





SNC – Sistema Nervoso Central Tareras mais complexas,
Memória, Secreção Glandular, Reação Motora

SNP – Sistema Nervoso Periférico
1. Somático (Controle Voluntário)
2. Autônomo (Reações Involuntárias – Coração, Respiração)

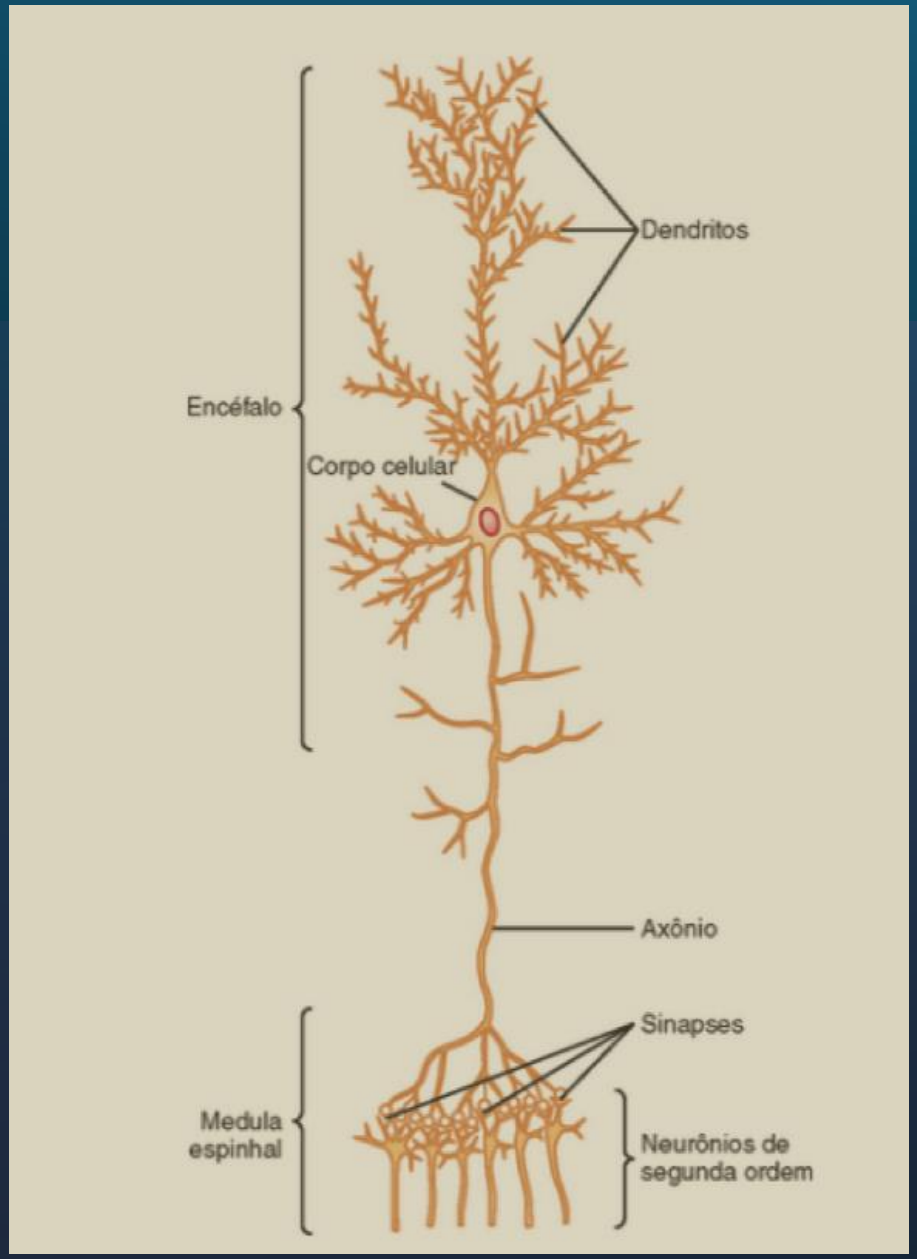


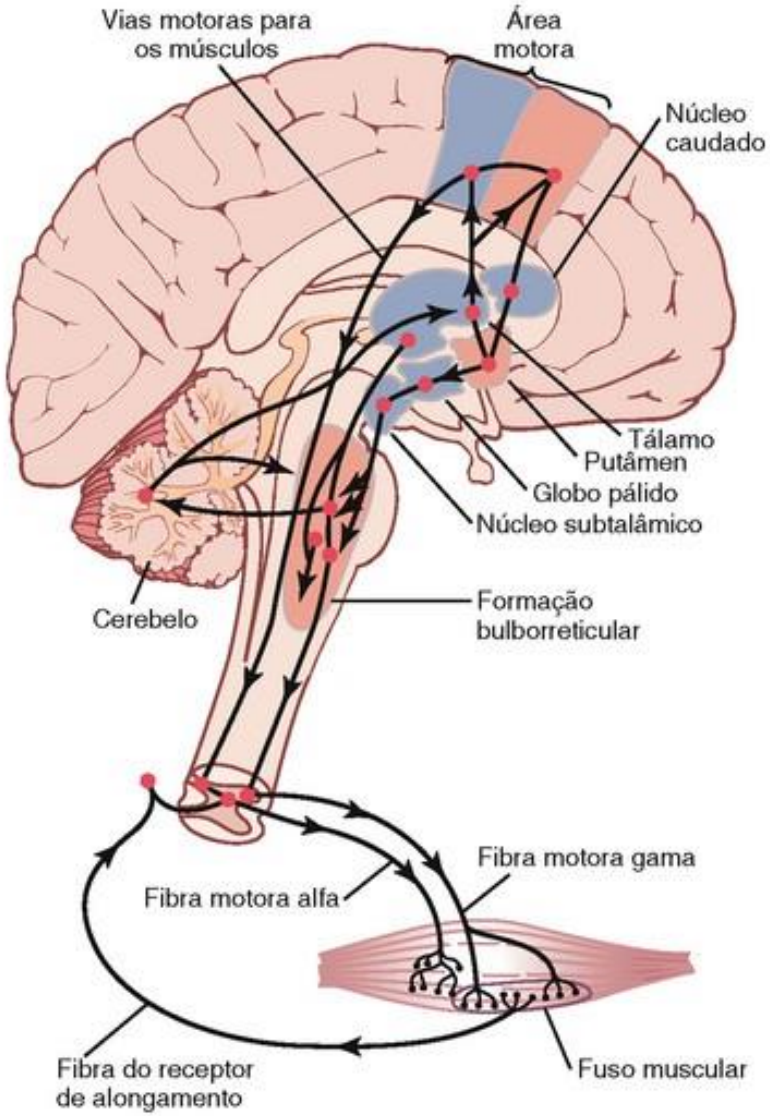
SNC – Sistema Nervoso Central
Tarefas mais complexas,
Memória, Secreção Glandular,
Reação Motora

SNP – Sistema Nervoso Periférico
1. Somático (Controle Voluntário)
2. Autônomo (Reações Involuntárias – Coração, Respiração)

SNP Autônomo Simpático
Reações de tensão e estresse
Aumento da frequência cardíaca,
Aumento da pressão arterial,
Dilatação da pupila e brônquios, etc

SNP Autônomo Parassimpático
Controle em situação de calmaria
Funções digestivas, diminui a FC

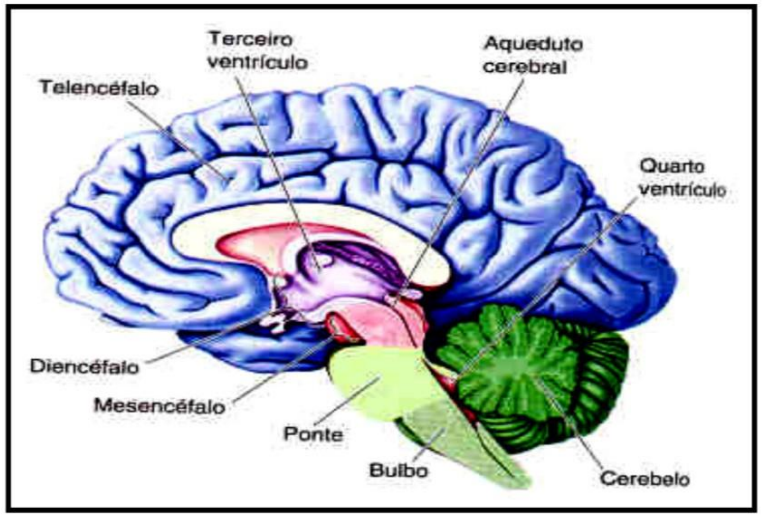

Dendritos
Encéfalo
Corpo celular
Axônio
Sinapses
Medula espinhal
Neurônios de segunda ordem

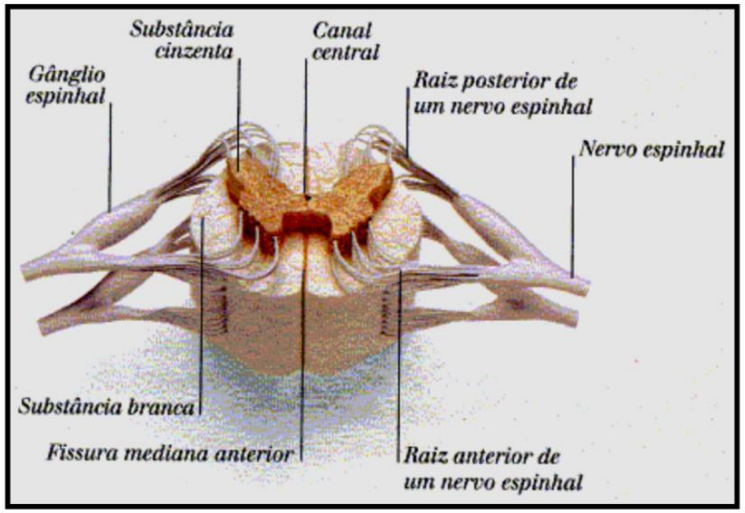

Vias motoras para os músculos
Área motora
Núcleo caudado
Tálamo
Putâmen
Globo pálido
Núcleo subtalâmico
Cerebelo
Formação bulborreticular
Fibra motora gama
Fibra motora alfa
Fibra do receptor de alongamento
Fuso muscular

Sistema Nervoso

➢ ***Divisão Anatômica***

Tecido Nervoso → Sistema Nervoso

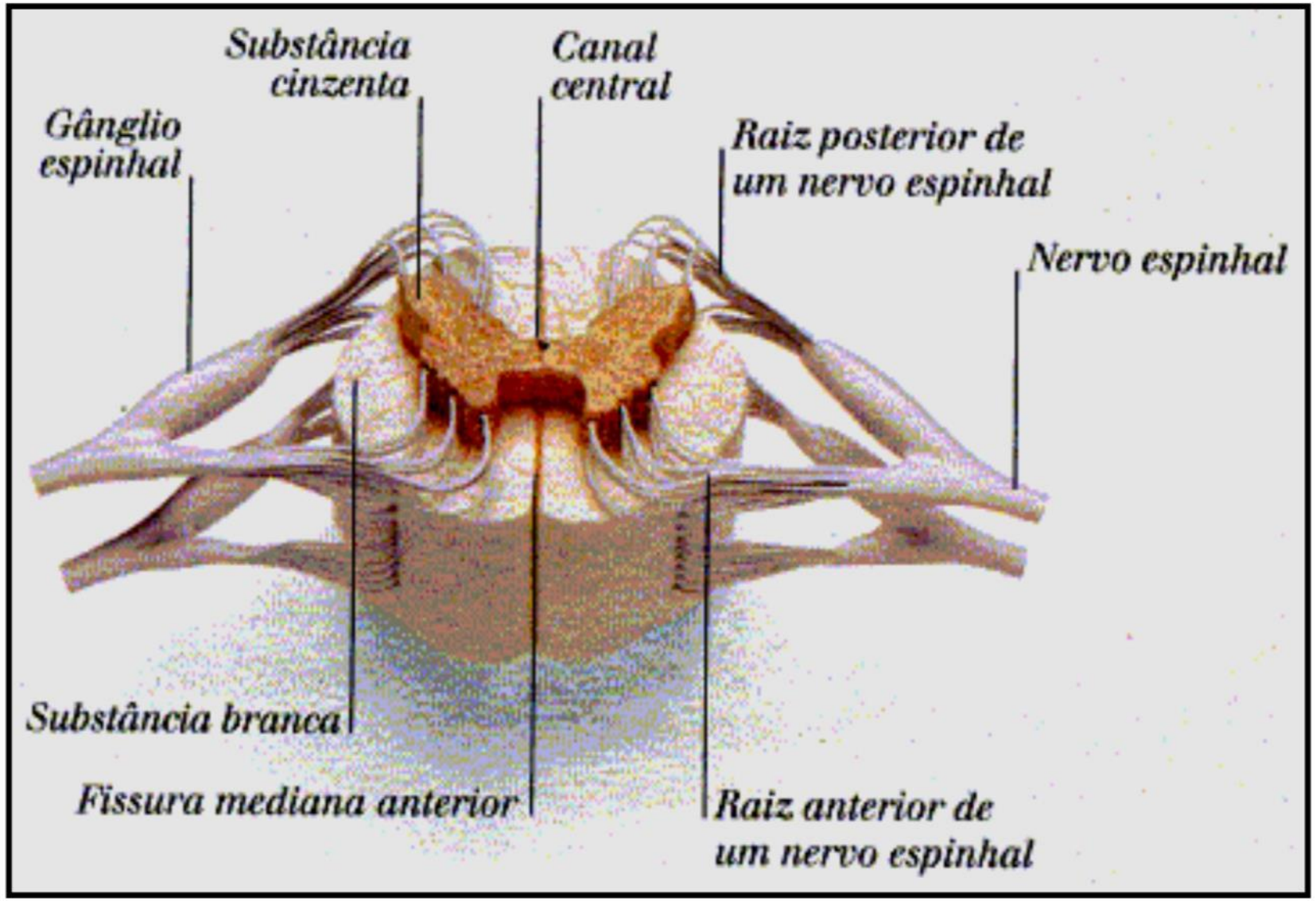




SNC → Encéfalo
- **Cérebro**
- **Cerebelo**
- **Ponte**
- **Bulbo**

SNC → Medula espinhal

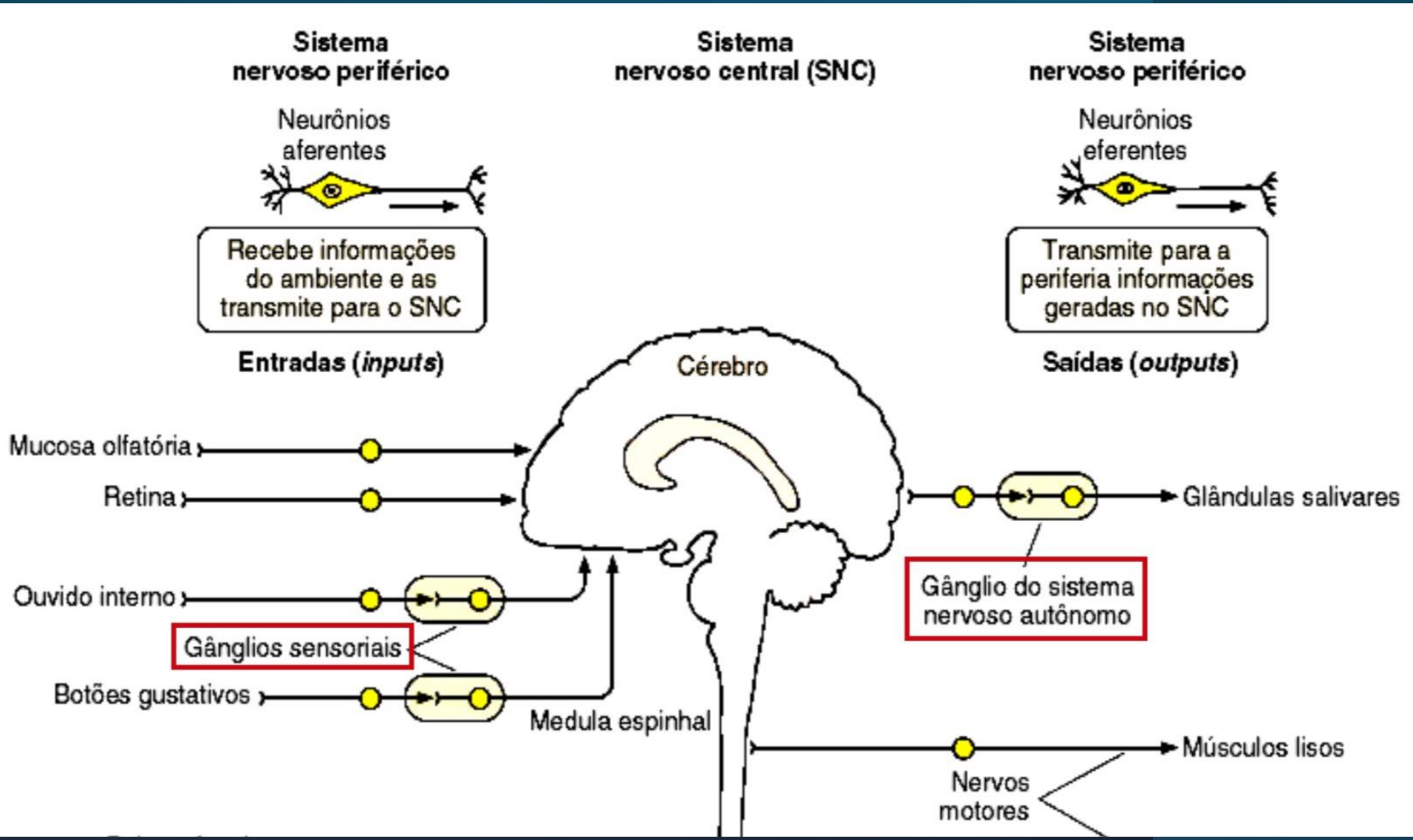
Nervoso



SNP

↓

Gânglios e nervos


Sistema nervoso periférico
Sistema nervoso central (SNC)
Sistema nervoso periférico
Neurônios aferentes
Neurônios eferentes
Recebe informações do ambiente e as transmite para o SNC
Transmite para a periferia informações geradas no SNC
Entradas (inputs)
Saídas (outputs)
Cérebro
Mucosa olfatória
Retina
Ouvido interno
Gânglios sensoriais
Botões gustativos
Medula espinhal
Glândulas salivares
Gânglio do sistema nervoso autônomo
Músculos lisos
Nervos motores

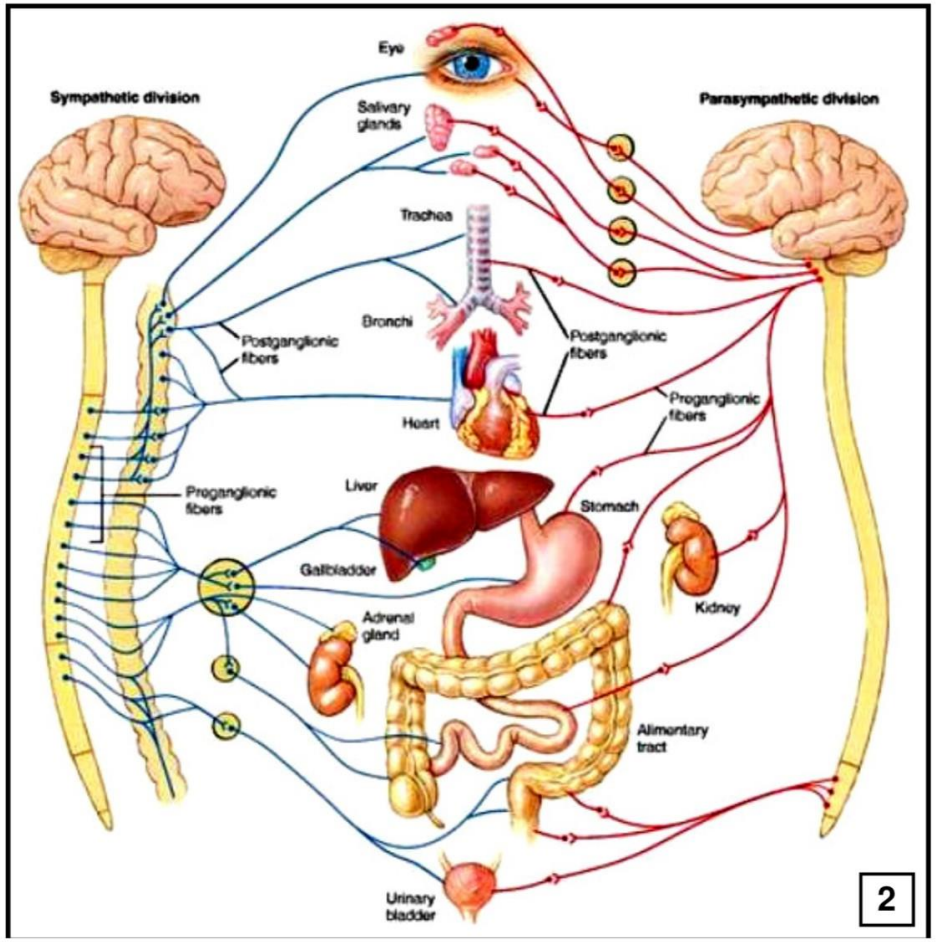
Gânglio

Glânglio é o aglomerado de corpos celulares de neurônios **fora** do Sistema Nervoso Central (SNC),

- ***Conjunto de pericários no SNP + células satélites***

Tipos:

1. Gânglios Sensitivos
2. Gânglios do SNA



Sistema Nervoso

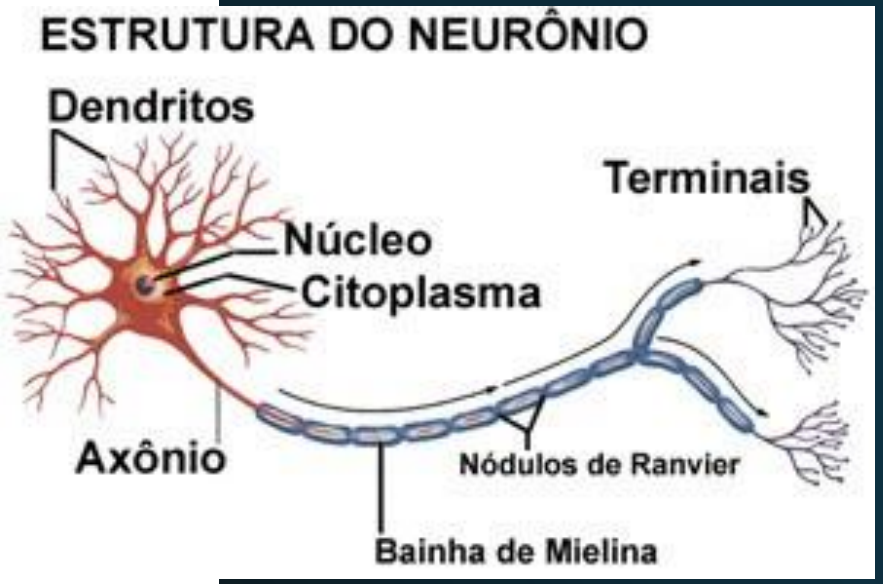
- Contém mais de 100 bilhões de neurôneos.
- Sinais **aferentes** chegam ao neurônio através de sinapses
- As sinapses estão localizadas nos dendritos neuronais
- Para diferentes tipos de neurônios, podem existir até 200.000 conexões sinápticas
- O sinal **eferente** desse neurônio trafega por axônio único
- O axônio tem muitas ramificações distintas que se dirigem para outras regiões do **Sistema nervoso** ou para a **periferia do corpo**

Sistema Nervoso - Funções

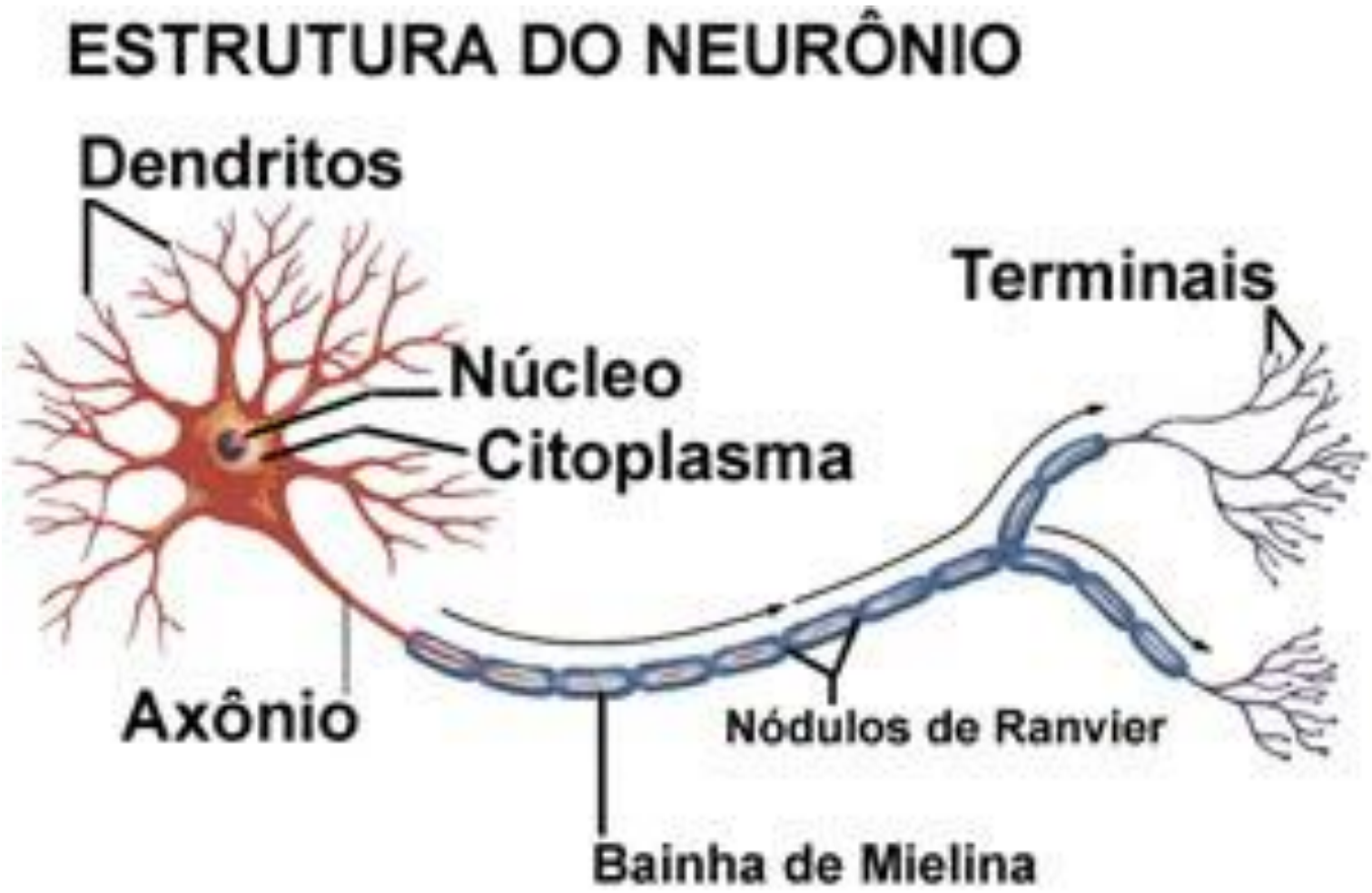
- **Sensitiva**
  - Através de órgãos sensorias como visão, audição, tato, olfato, paladar
- **Integradora**
  - Processamento dos sinais do organismo para o cérebro
- **Motora**
  - Através da ativação da Contração muscular ou da Secreção glandular de substâncias
- Ex. Estímulo de calor no braço -> Córtex Cerebral -> Resposta -> Reação Motora (contração do braço)

Sistema Nervoso – O Neurônio

- A célula nervosa apresenta-se munida de prolongamentos de vários tipos, e que dela fazem parte integrante.
- Ao conjunto formado pela célula e respectivos prolongamentos, dá-se o nome de **neurônio**.
- Distinguem-se neste, portanto, o corpo celular e os prolongamentos.



Sistema Nervoso – O Neurônio



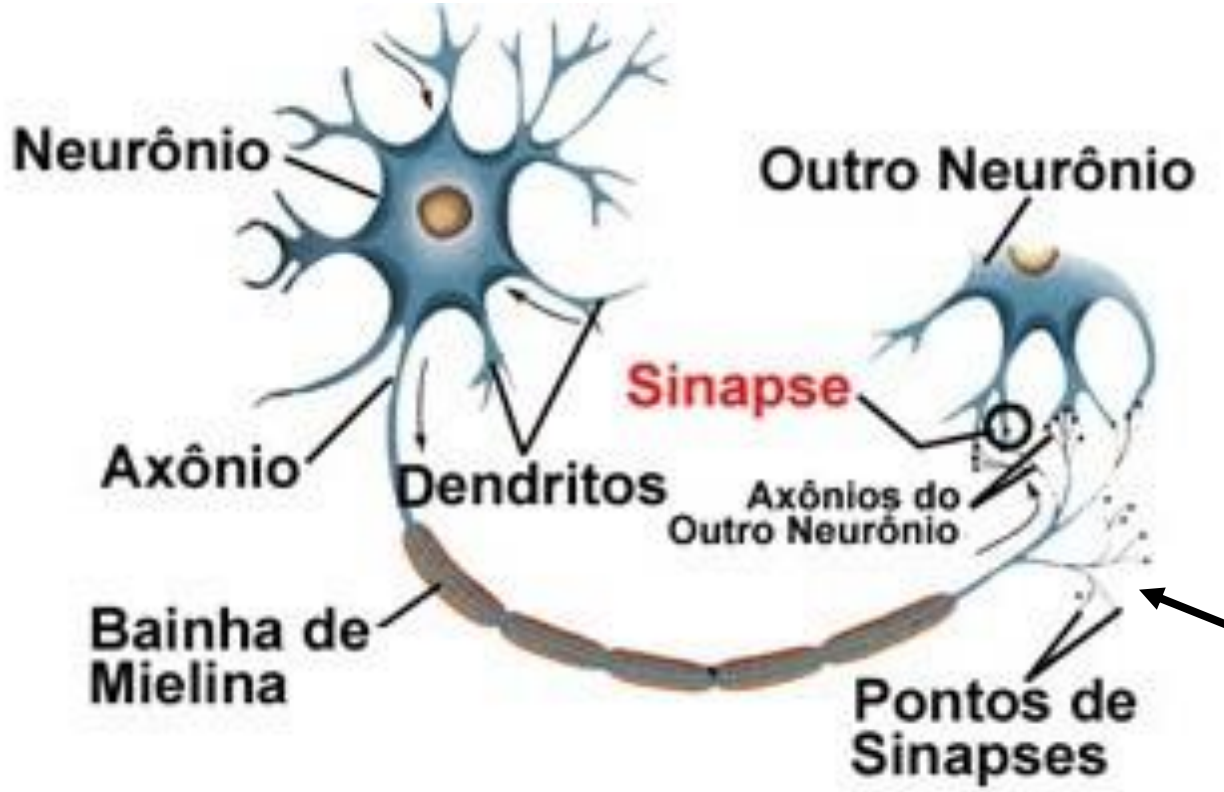
Sistema Nervoso – O Neurônio e Sinapses

Chama-se sinapse a articulação das terminações de um neurônio com as de outro, ou então com a fibra muscular ou com as células glandulares.

Há, portanto, sinapses interneurais, sinapses mioneurais e sinapses neuroglandulares.

As sinapses interneurais se fazem entre as ramificações dos prolongamentos eferentes de um neurônio e as dos prolongamentos aferentes de outro.

Sistema Nervoso – O Neurônio e Sinapses



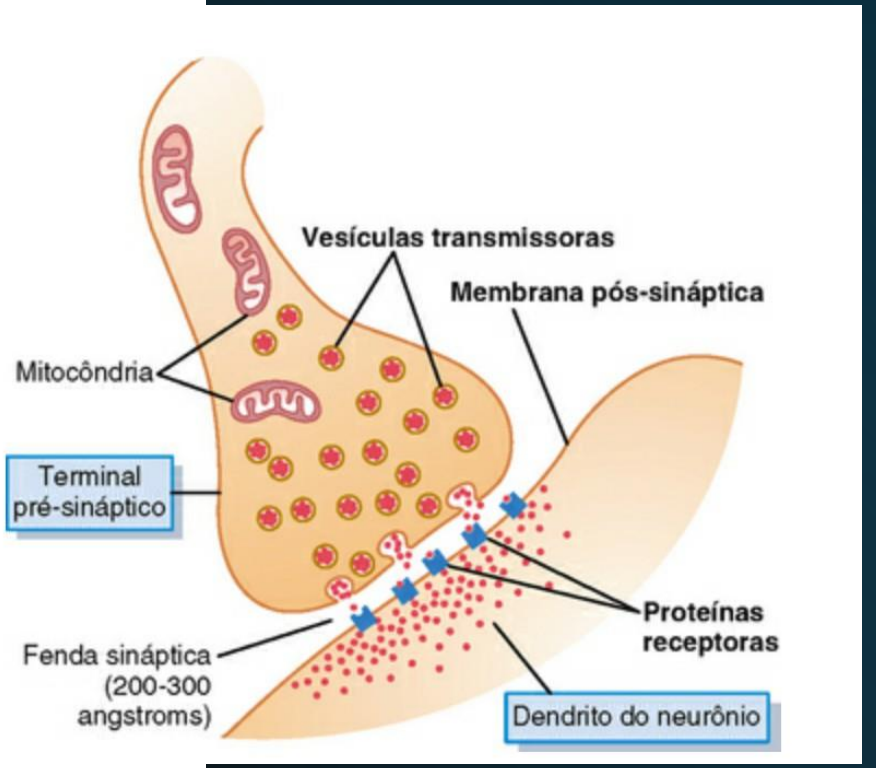
A transmissão do sinal é feita através de íons de sódio e potássio

O Neurônio e suas Sinapses

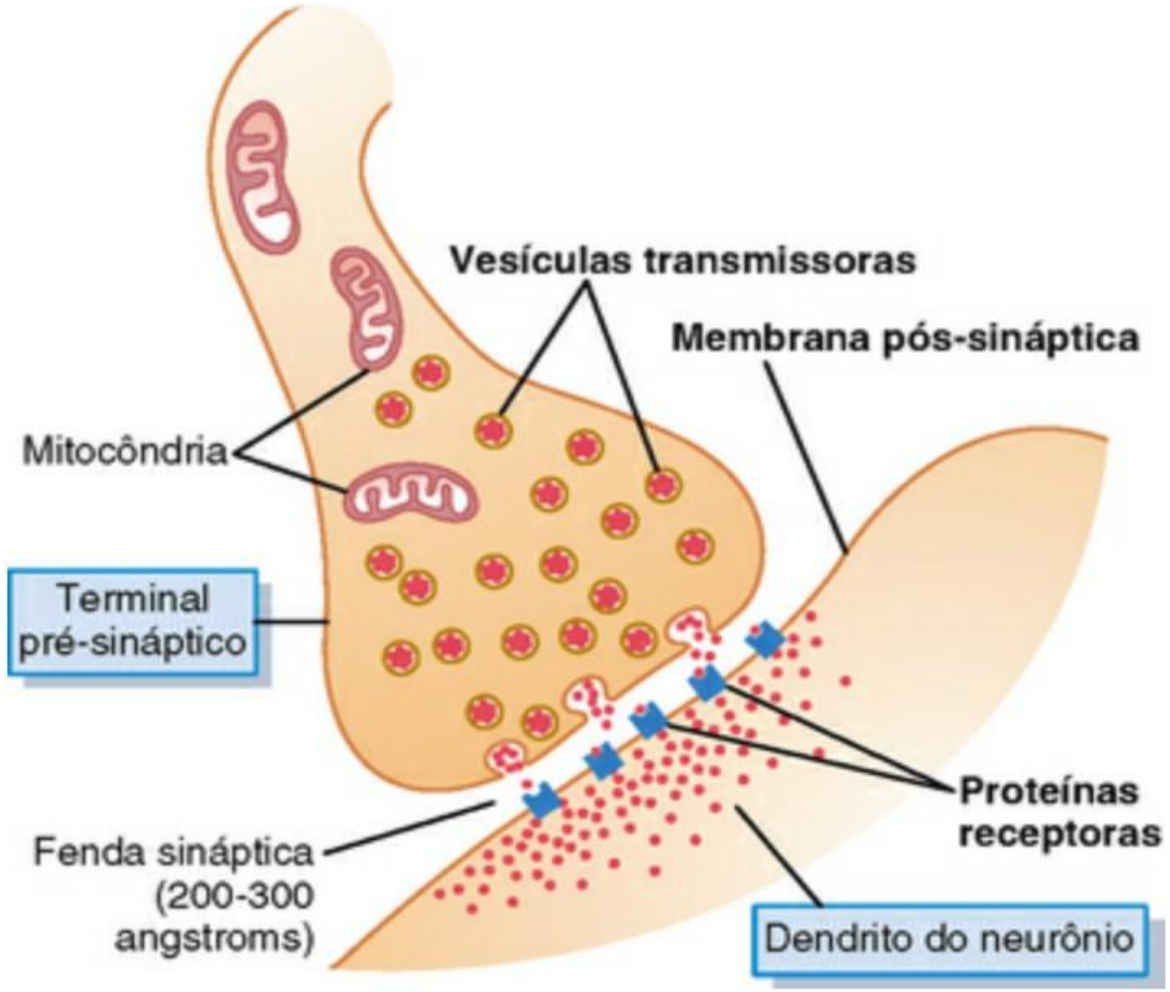
- Existem mais de **40 tipos** de Neurotransmissores (substâncias produzidas pelo neurônio para transmitir informação)

- **Neurotransmissores Excitatórios**
- Acetilcolina
- Dopamina
- Serotonina
- Histamina
- Glutamato

- **Neurotransmissores Inibitório**s
- GABA (ácido Gama)
- Aminoácido Glicina



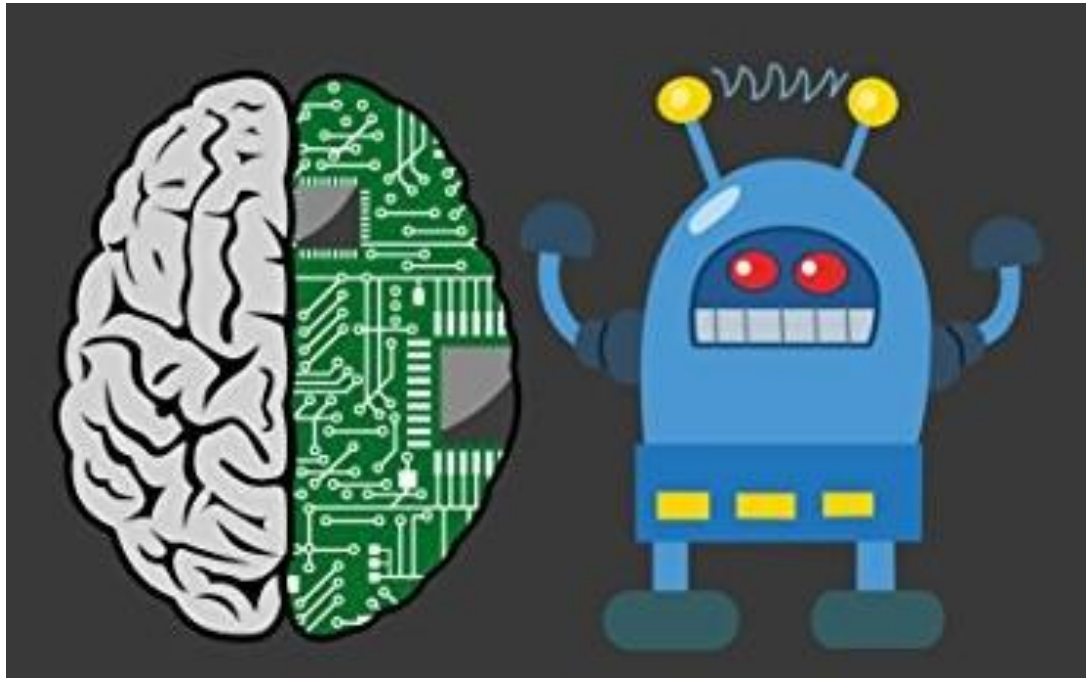
Sistema Nervoso – O Neurônio e Sinapses

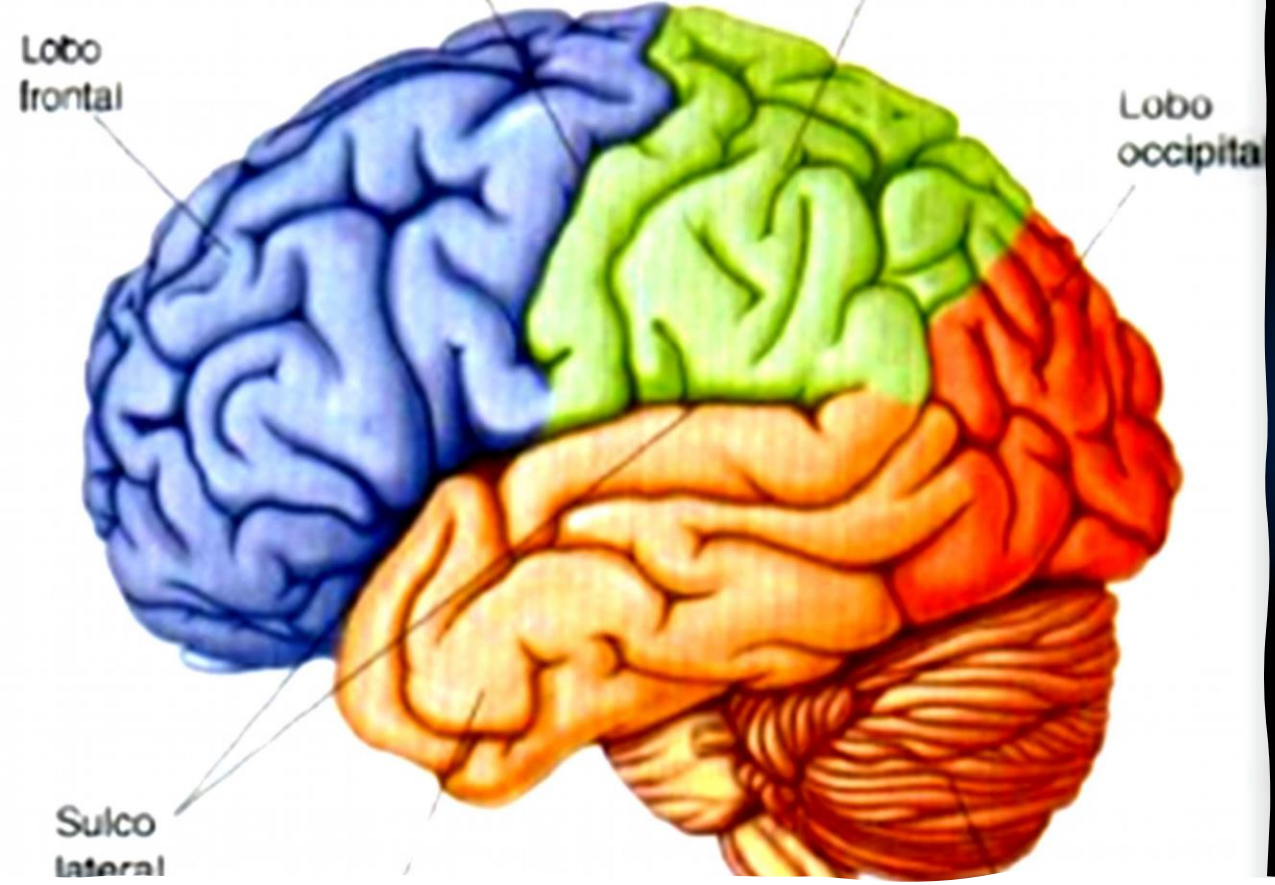


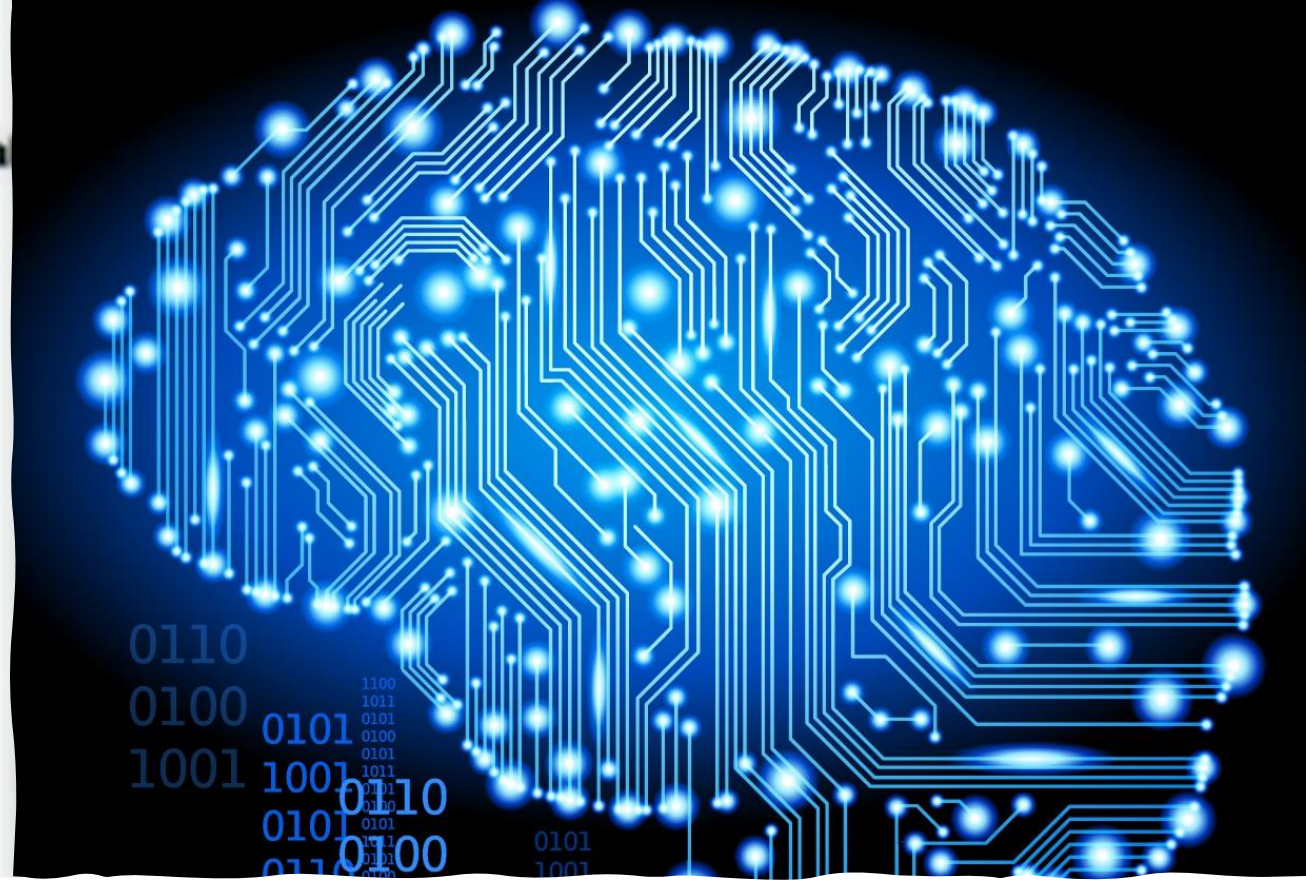
Neurônio x Redes Neurais Artificiais

- Assim como um cérebro usa uma rede de células interconectadas chamadas **neurônios** para criar um processador paralelo maciço,

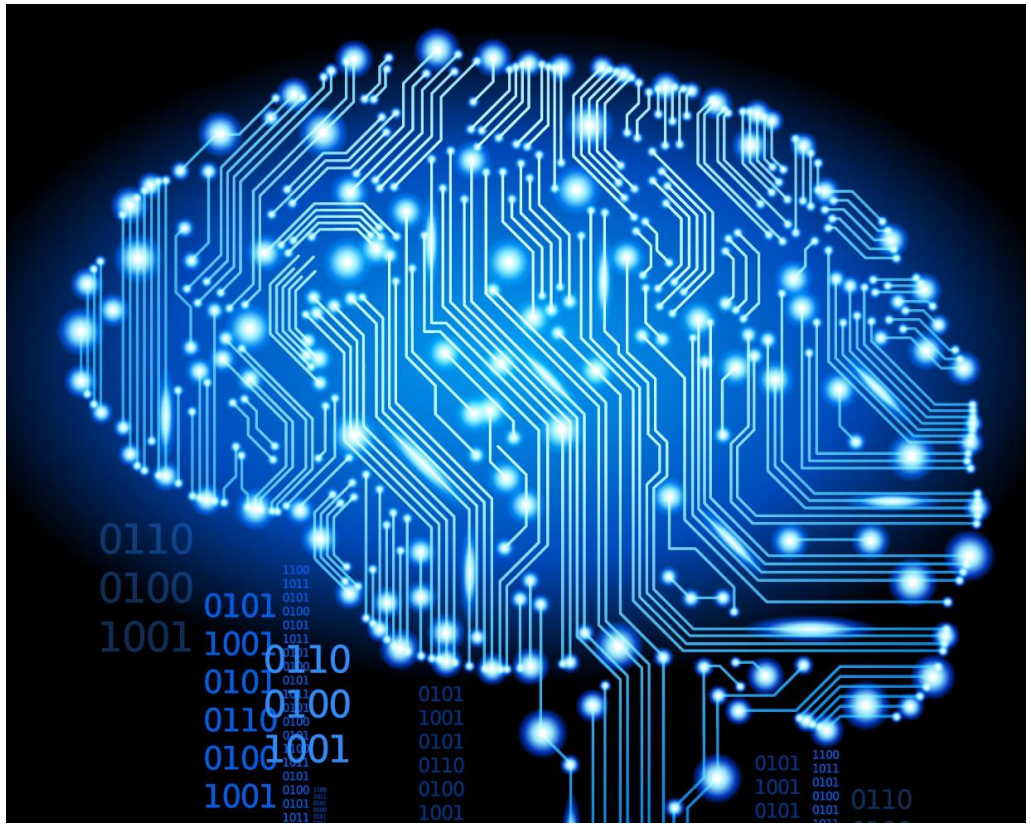
- a rede neural usa uma **rede de neurônios artificiais** para resolver problemas de aprendizagem

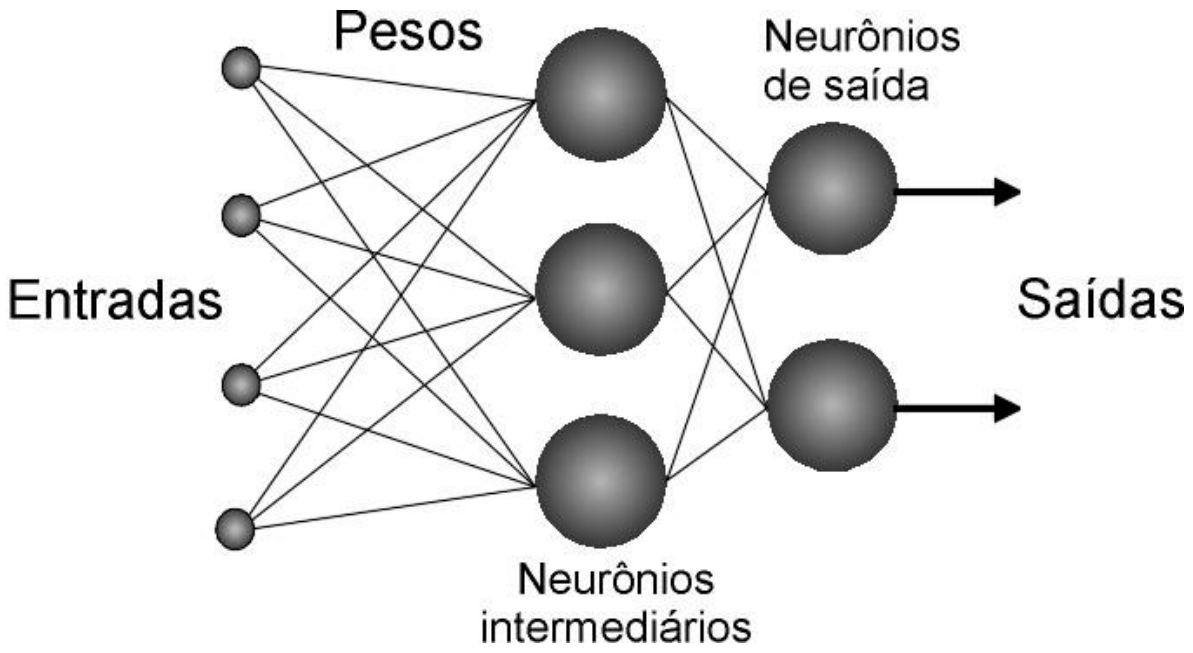






Sistema Nervoso x Redes Neurais Artificiais



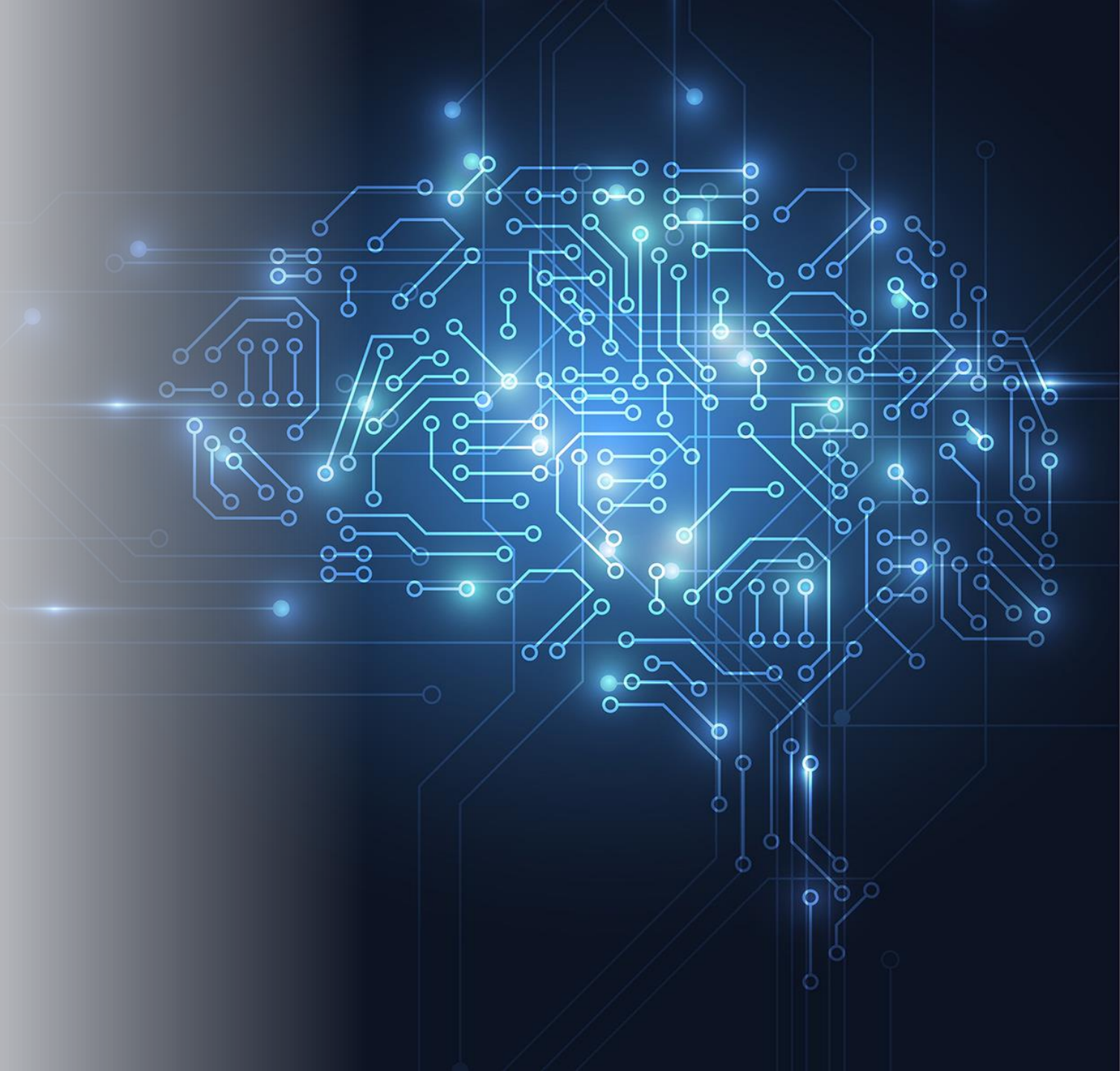


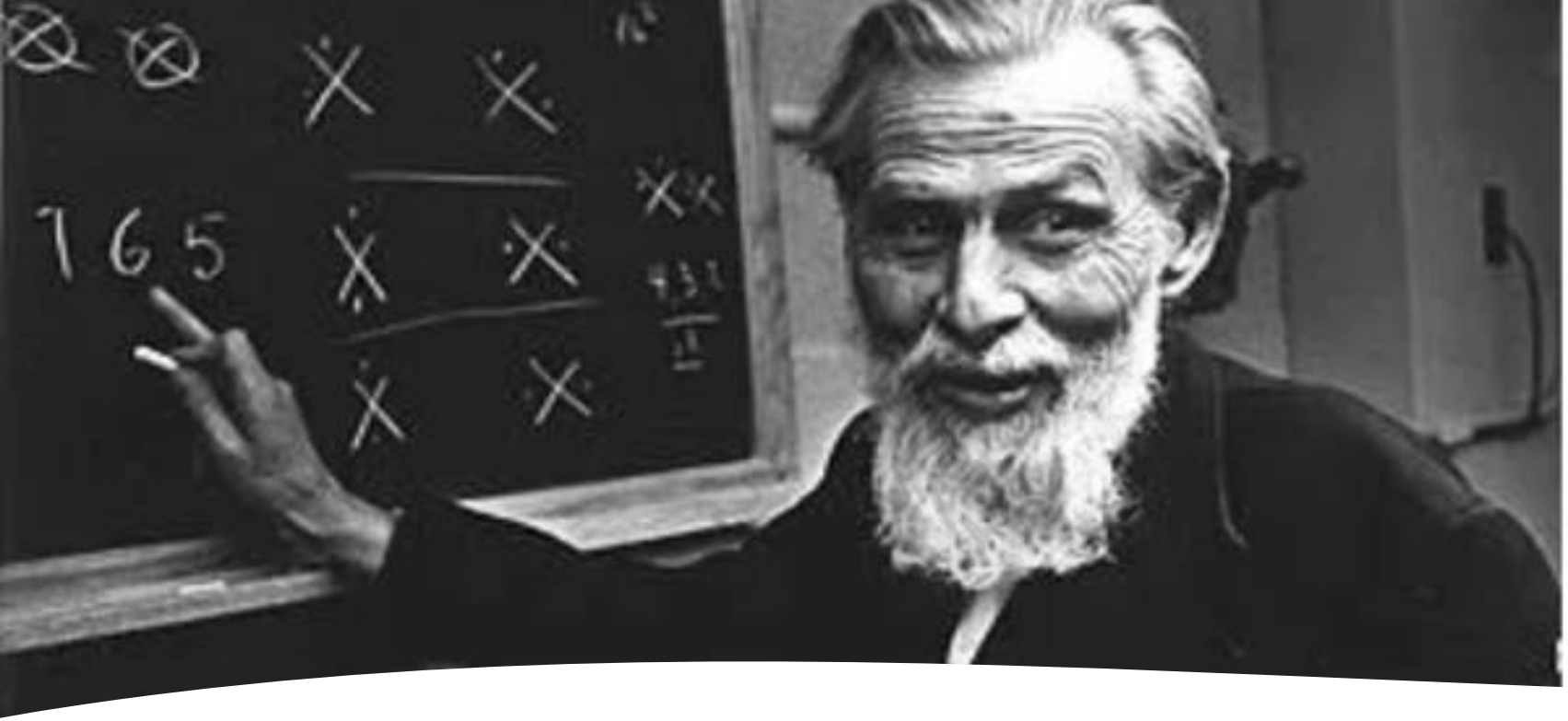
Sistema Nervoso x Redes Neurais Artificiais

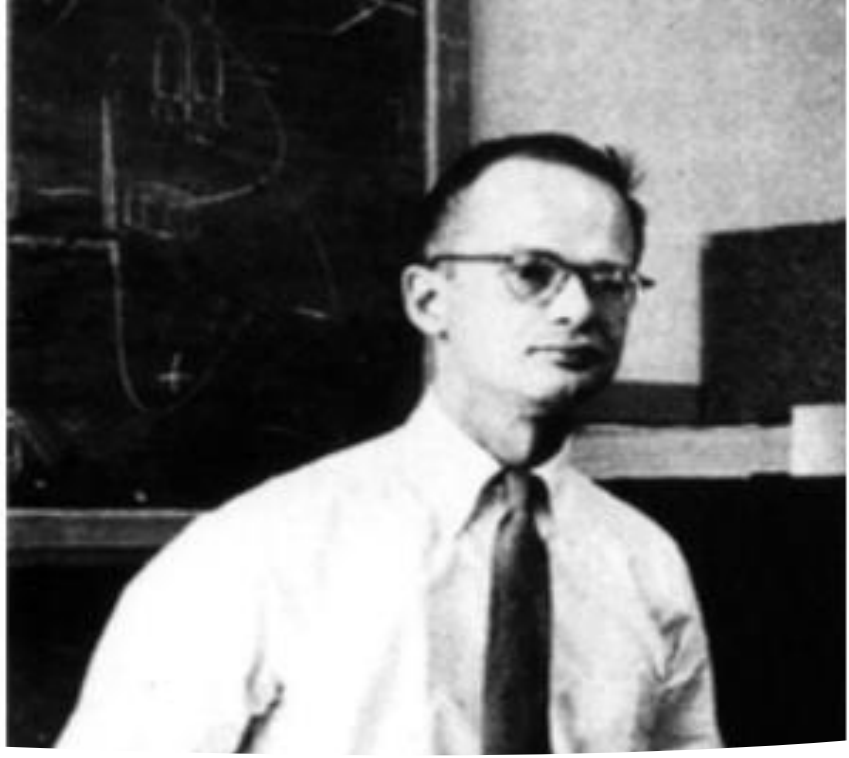
Redes Neurais Artificiais

- A partir dessas motivações, as Redes Neurais Artificiais (RNAs), tomou como inspiração a estrutura e o funcionamento do sistema nervoso.

- As RNAs têm como objetivo simular a capacidade de aprendizado do cérebro humano na aquisição de conhecimento.







Redes Neurais

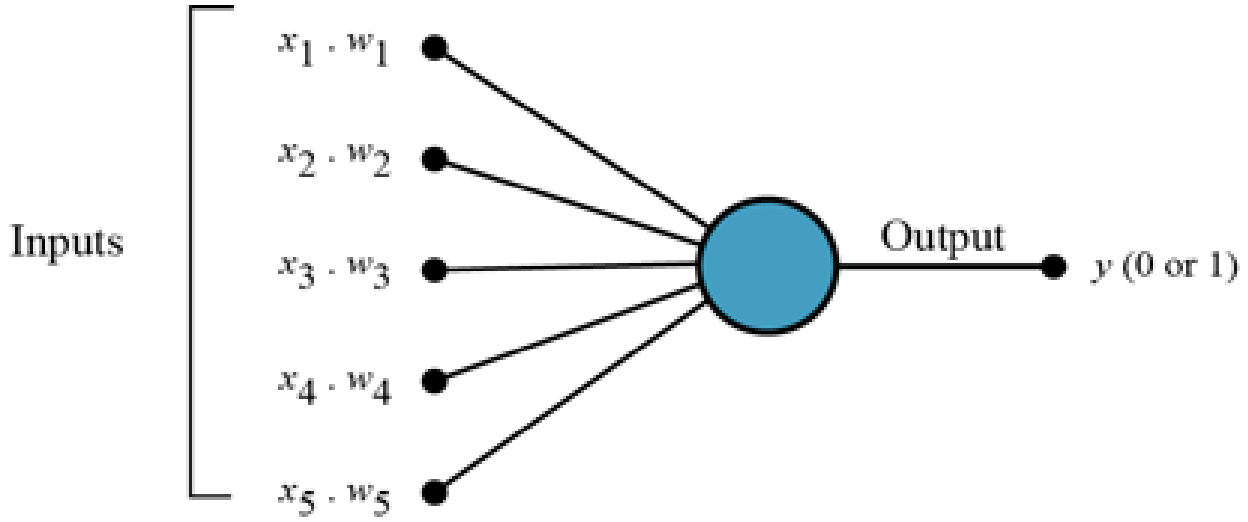
- A procura por sistemas computacionais e matemáticos do sistema nervoso teve início na década de 40 (apredizado).

- As redes neurais foram propostas pela primeira vez em 1943, pelo neurofisiologista Warren McCullough, da Universidade de Illinois, e Walter Pitts, da Universidade de Chicago, onde eles escreveram um artigo sobre como os neurônios podem funcionar

Redes Neurais

- Warren McCulloch e Walter Pitts criaram um modelo computacional para redes neurais baseadas em matemática e algoritmos denominados lógica de limiar (threshold logic).

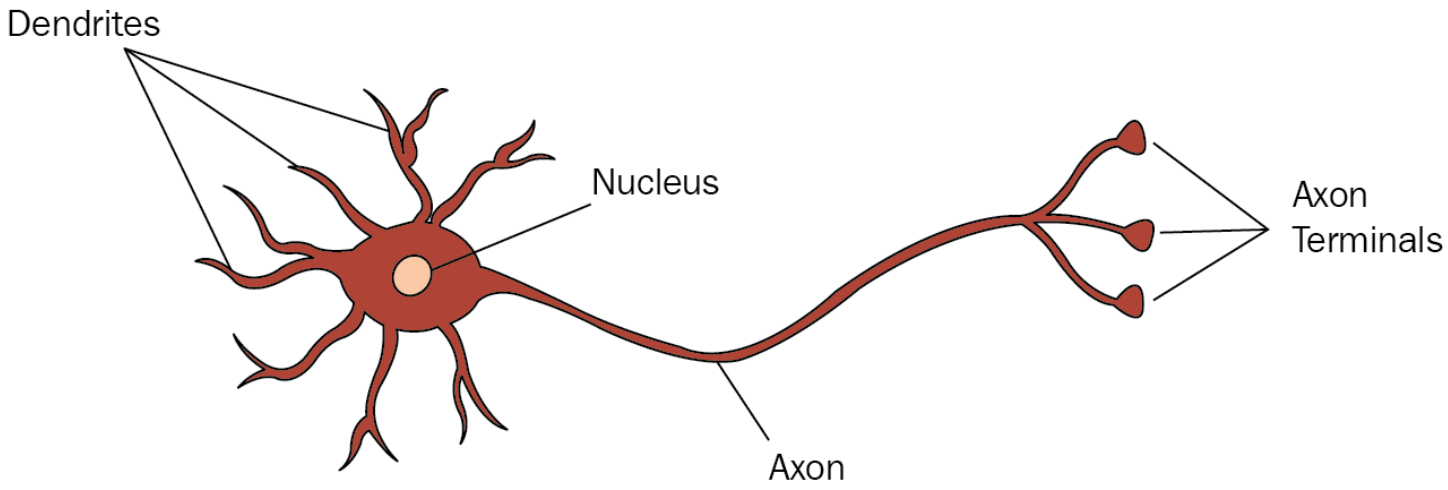
- Este modelo abriu o caminho para a pesquisa da rede neural dividida em duas abordagens: uma abordagem focada em processos biológicos no cérebro, enquanto a outra focada na aplicação de redes neurais à inteligência artificial.

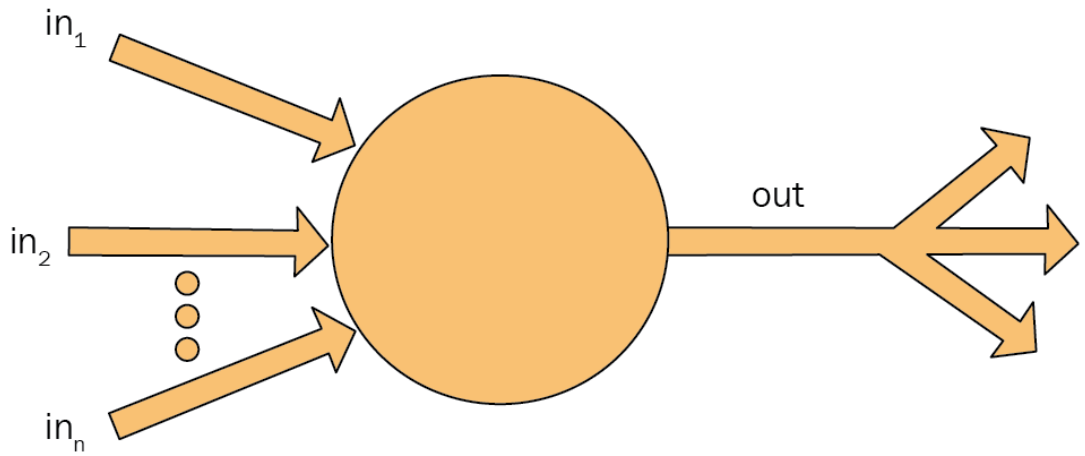


Redes Neurais

- A primeira rede neural treinável, o Perceptron, foi demonstrada pelo psicólogo da Cornell University, Frank Rosenblatt em 1957

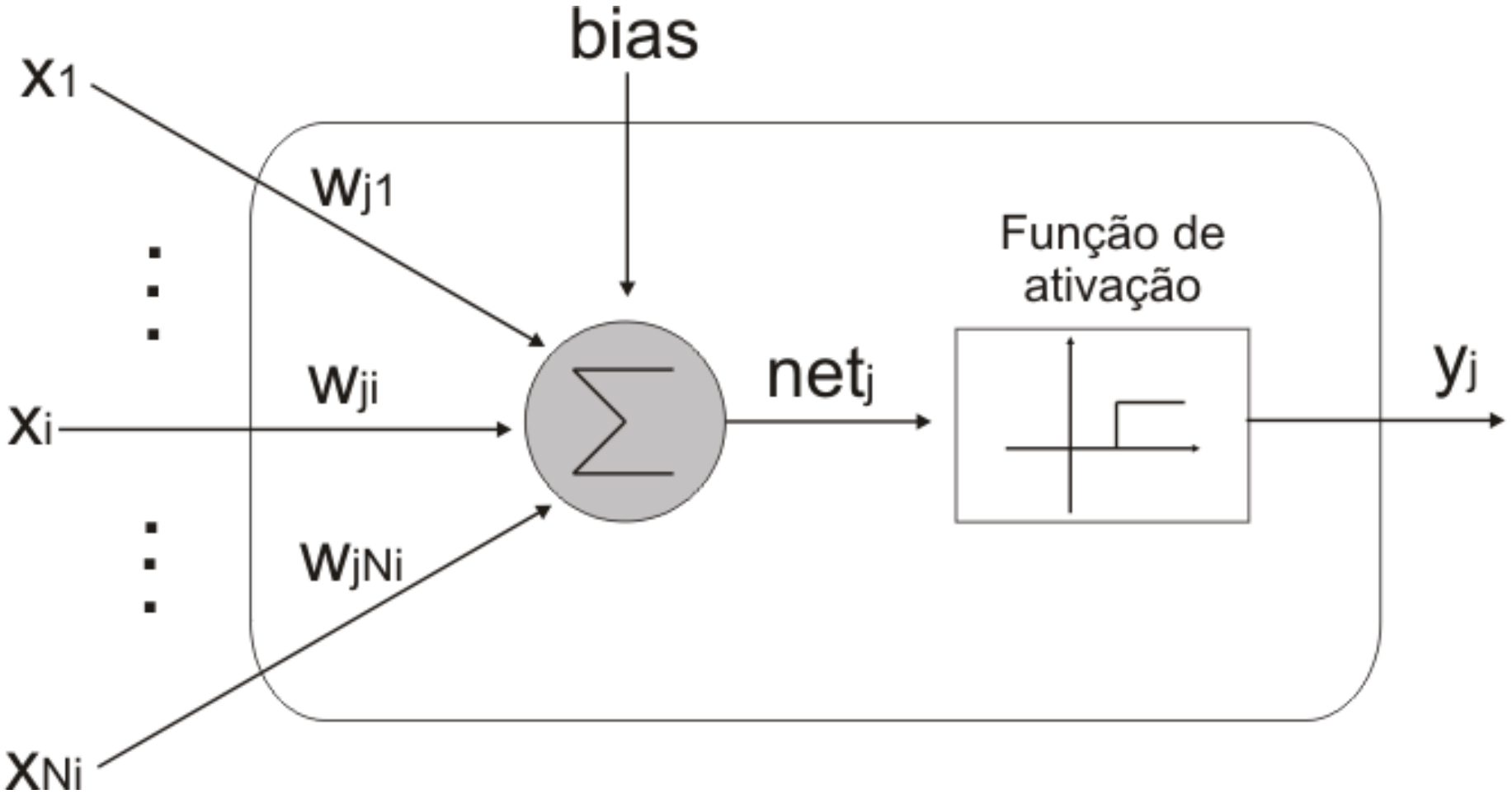
- O design do Perceptron era muito parecido com o da rede neural moderna, exceto que tinha apenas uma camada com pesos e limites ajustáveis, em que os pesos se situam entre a entrada e camada de saída





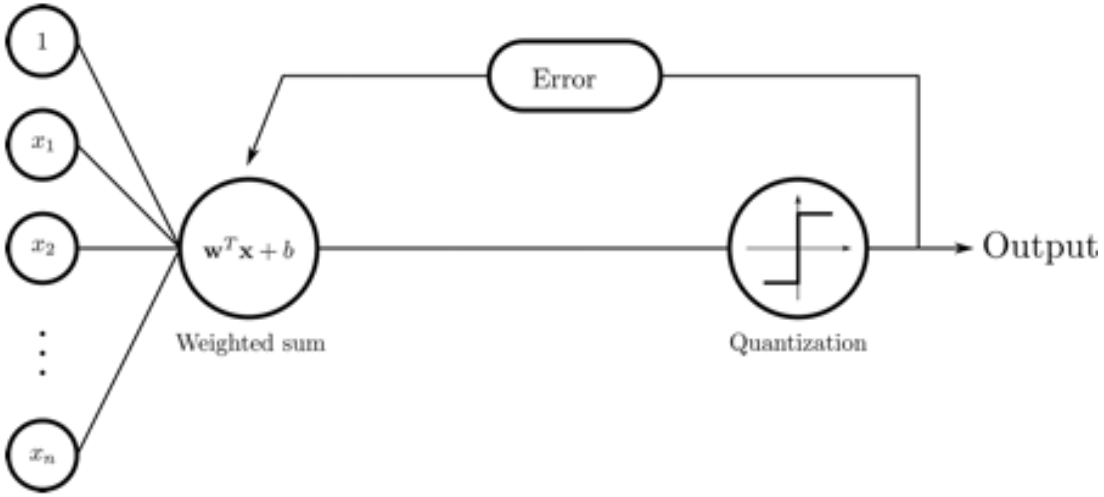
Redes Neurais

- Frank Rosenblat e a Máquina Perceptron Mark I, a primeira implementação do algoritmo perceptron, no Cornell Aeronautical Laboratory, 1957.
- Arquitetura: Perceptron de camada única com 400 neurôneos.



Perceptron (Neurônio artificial)

Perceptron de uma camada



$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$$

$$z = \mathbf{w}^T\mathbf{x} + b$$

$$\hat{y} = H(z)$$

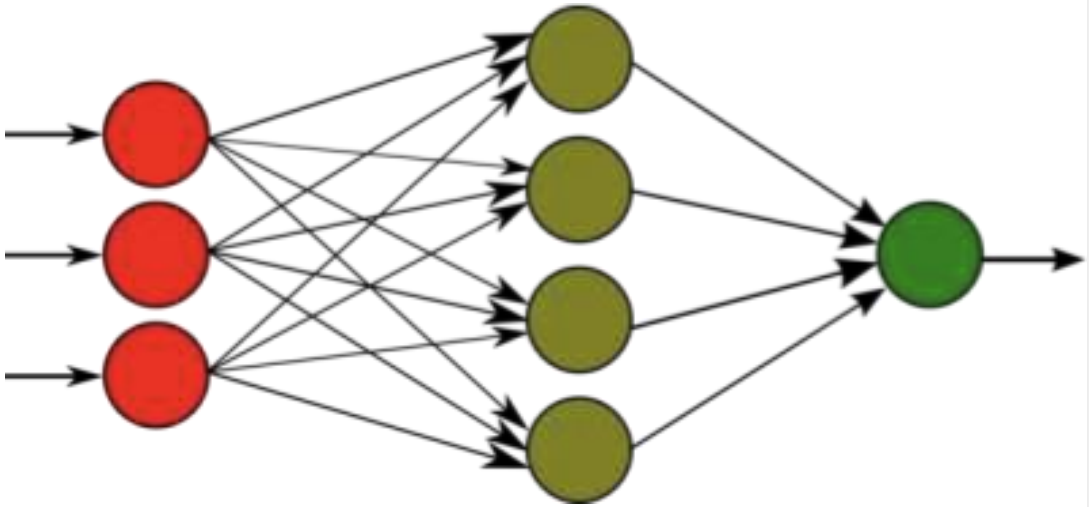
Redes Neurais

- Como o advento da **computação distribuída** e **processamento paralelo**, as redes neurais vem ganhando destaque atualmente, através da **Deep Learning** – Redes Neurais Profundas, com milhares de camadas internas.
- Aplicações:
- Reconhecimento de padrões, voz, escrita, imagens
- Problemas de Classificação (linear e não-linear)
- Previsão de séries temporais
- Automação de dispositivos inteligentes (Drones, Carros)
- Modelos sofisticados de padrões climáticos

Componentes Básicos das RNAs

- As RNAs são sistemas computacionais distribuídos compostos de unidades computacionais simples, densamente interconectadas.

Essas unidades são conhecidas como neurônios artificiais, que computam funções matemáticas

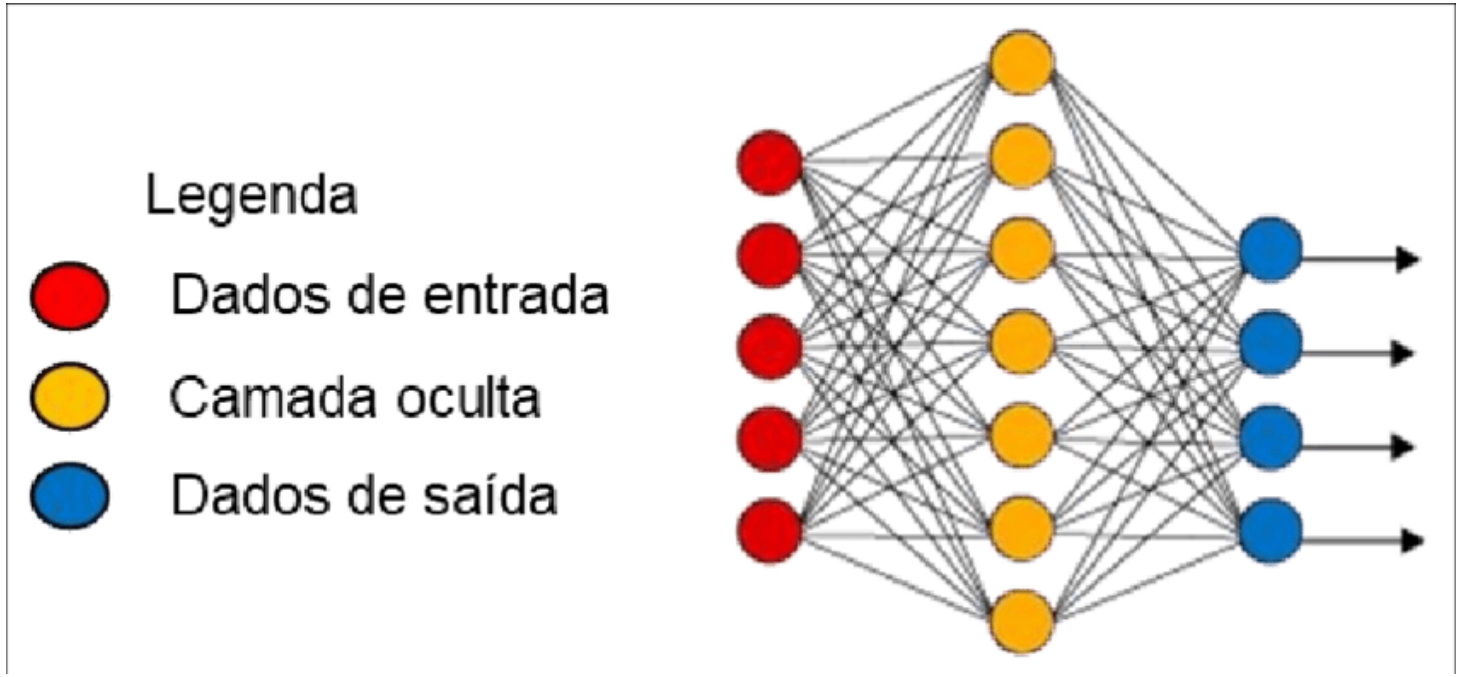


Componentes Básicos das RNAs

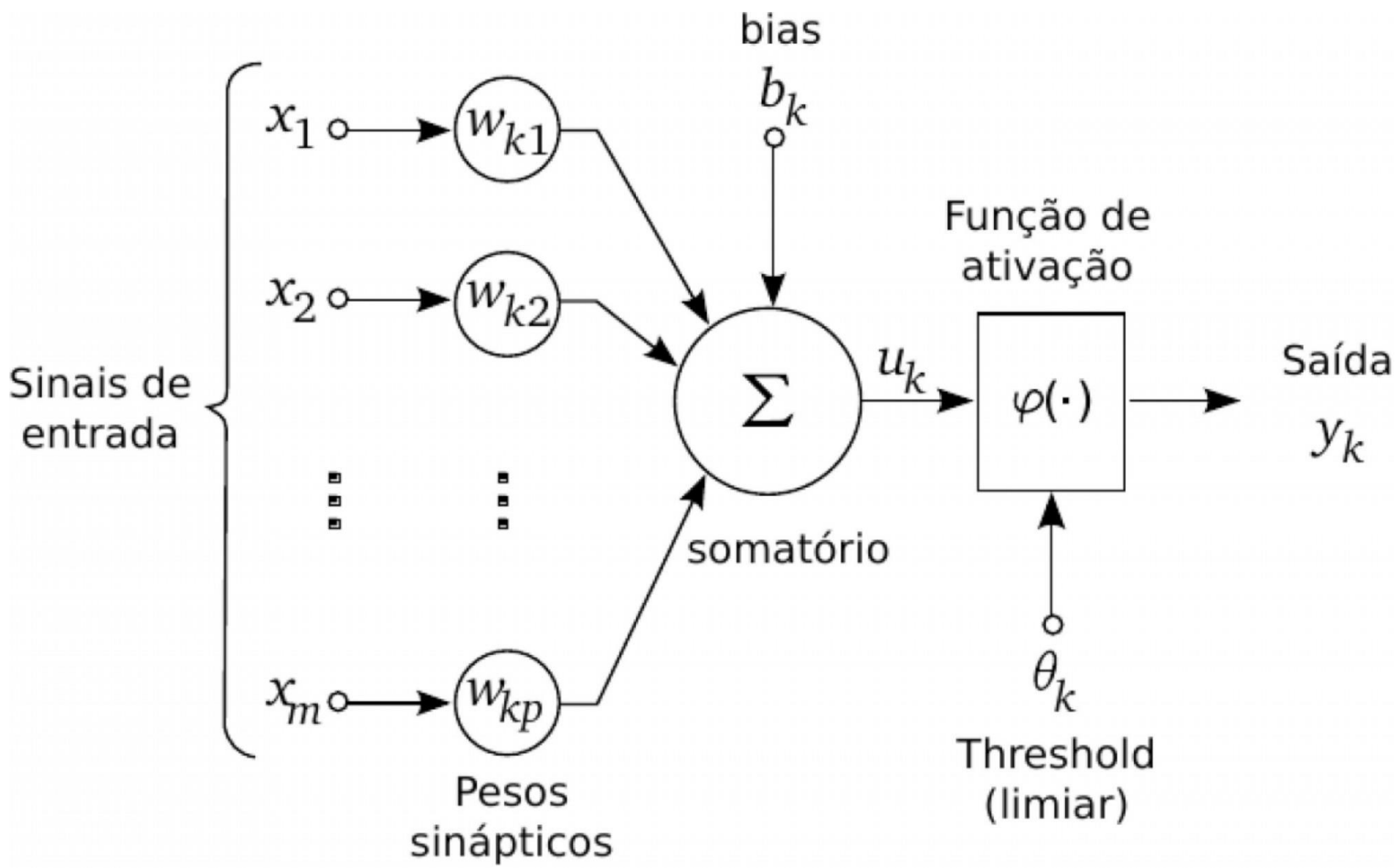
- As unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões.

- Essas conexões simulam sinapses biológicas, que possuem pesos associados , que ponderam a entrada recebida por cada neurônio da rede.

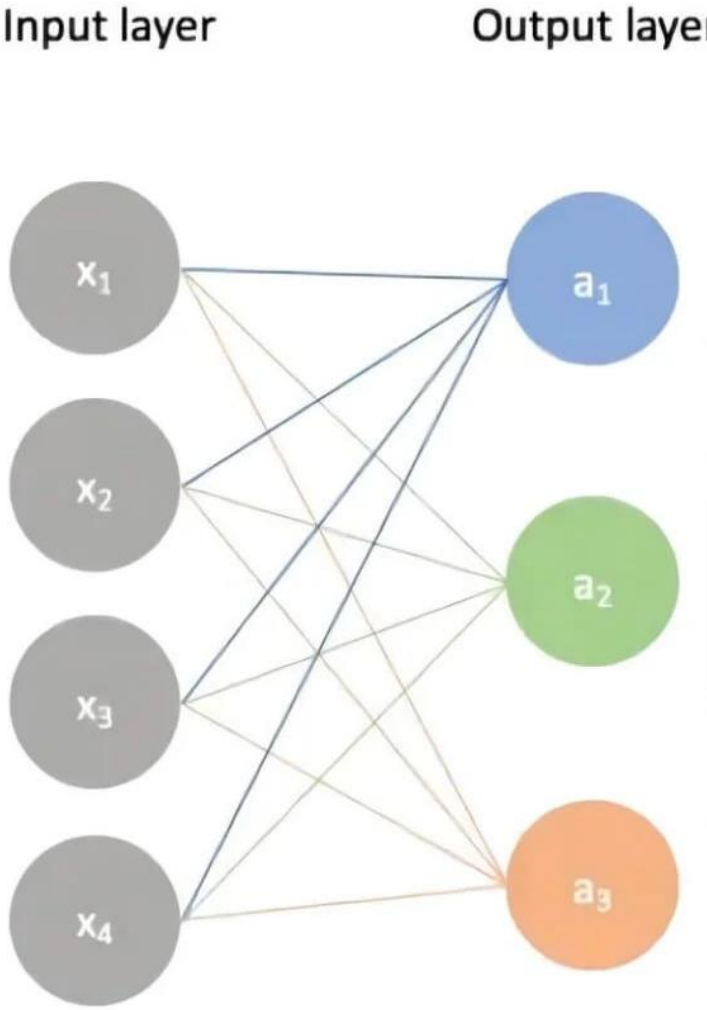
- Os pesos tem seus valores ajustados por um processo de aprendizagem que codifica o conhecimento adquirido.



Neurônio Artificial



Neurônio Artificial



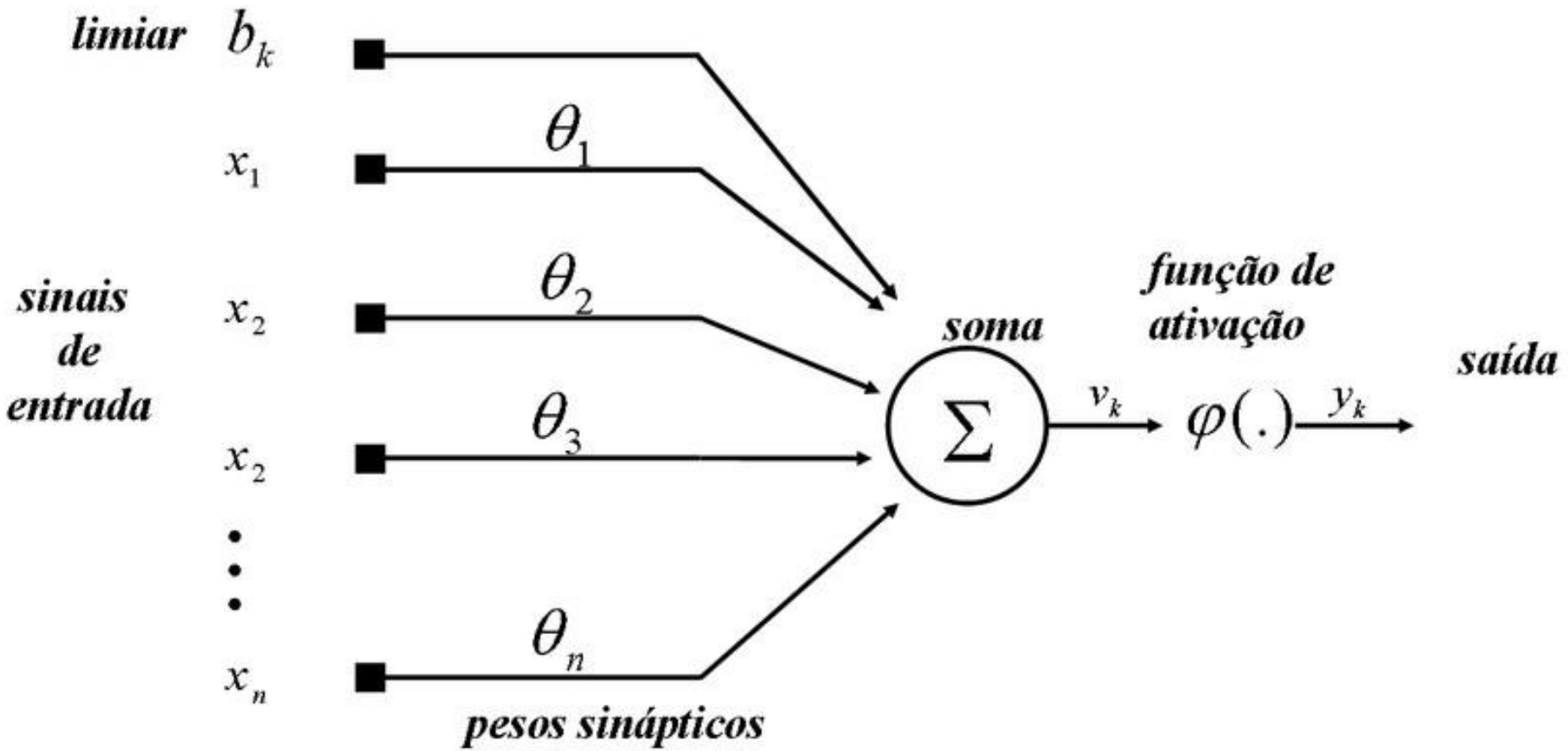
$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Arquitetura

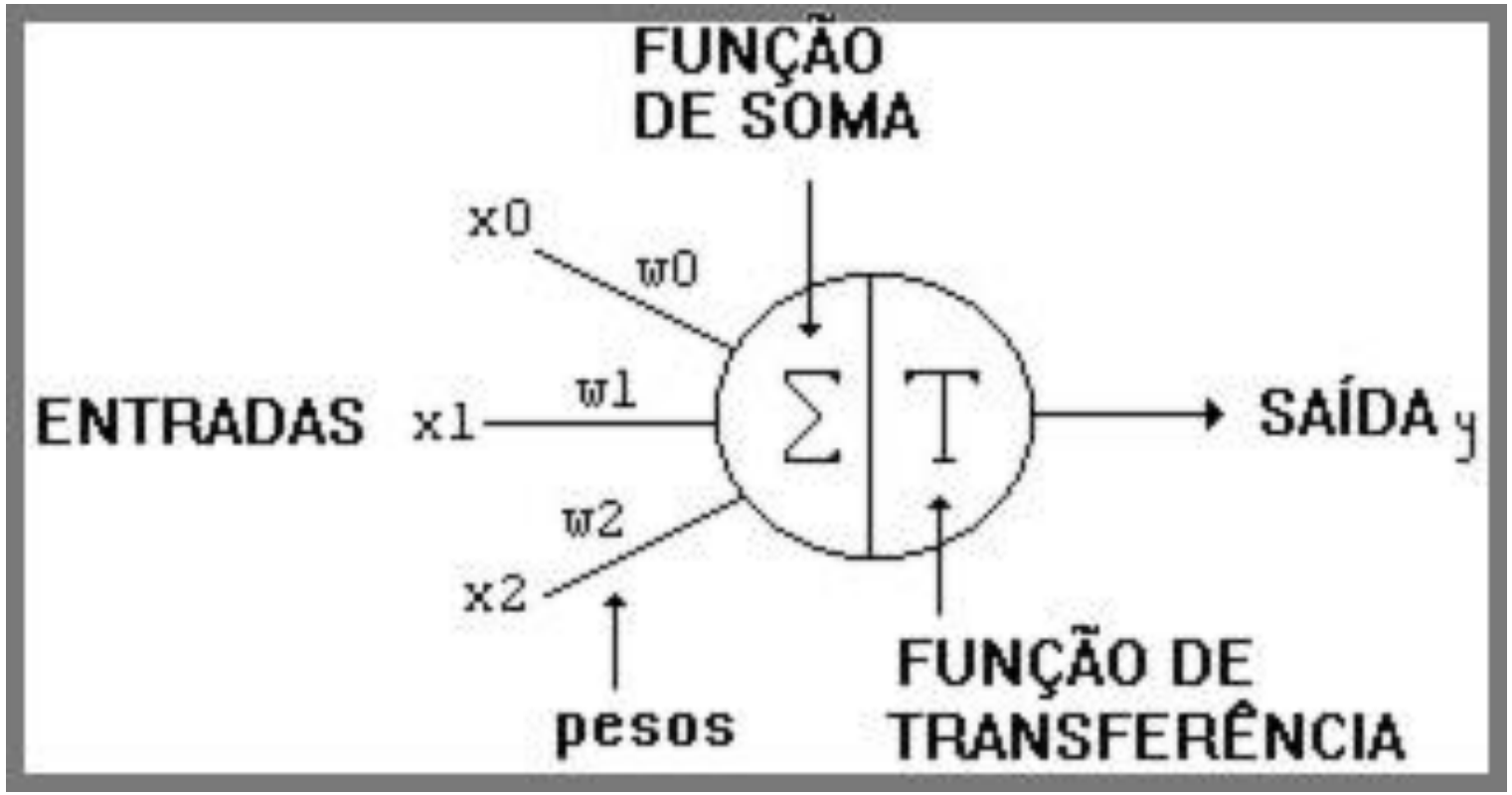
- O neurônio é a unidade de processamento fundamental de um RNA.
- Cada terminal de entrada do neurônio, recebe um valor
- Os valores recebidos são ponderados e combinados por uma função matemática f(a)
- Os valores de entrada são multiplicados pelos seus pesos sinápticos e somados
- Depois o valor Total da soma é apresentado a uma função de ativação, que retornará a resposta do neurônio.

Neurônio Artificial

(Bias)



Neurônio Artificial



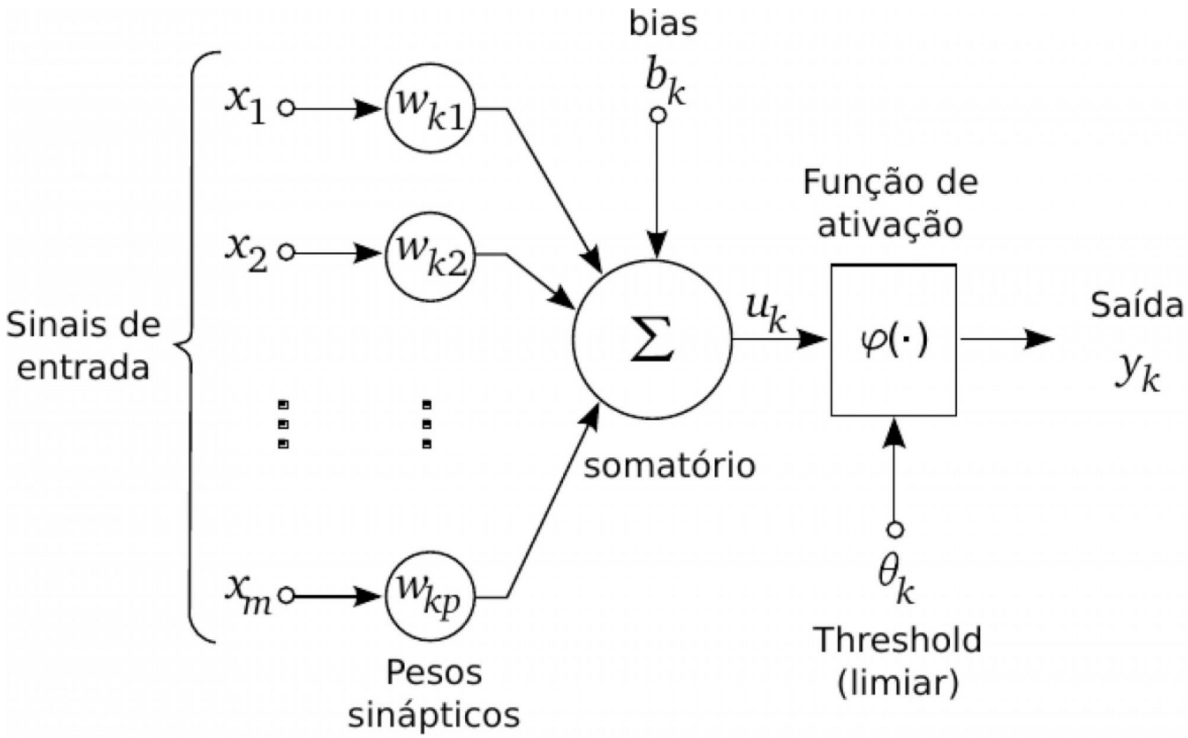
Neurônio Artificial

Além das entradas X, também se prevê uma entrada Extra, chamada de Bias, Polariazação ou Viés.

A primeira parte do processamento é:

$$S = \sum_{i=1}^{n} x_i . w_i + b_k$$

- São as entradas
- São os pesos associados às entradas
- é o Bias ou polarização

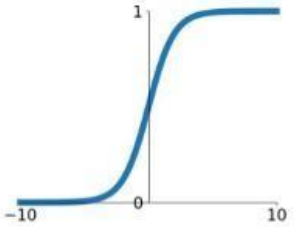


Neurônio Artificial

Activation Functions

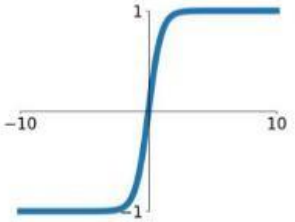
Sigmoid

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



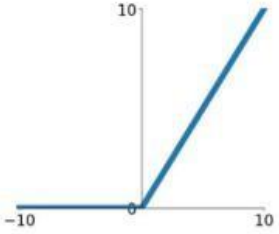
tanh

$$\tanh(x)$$



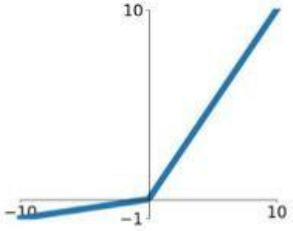
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

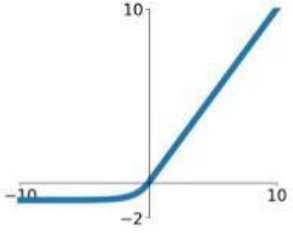


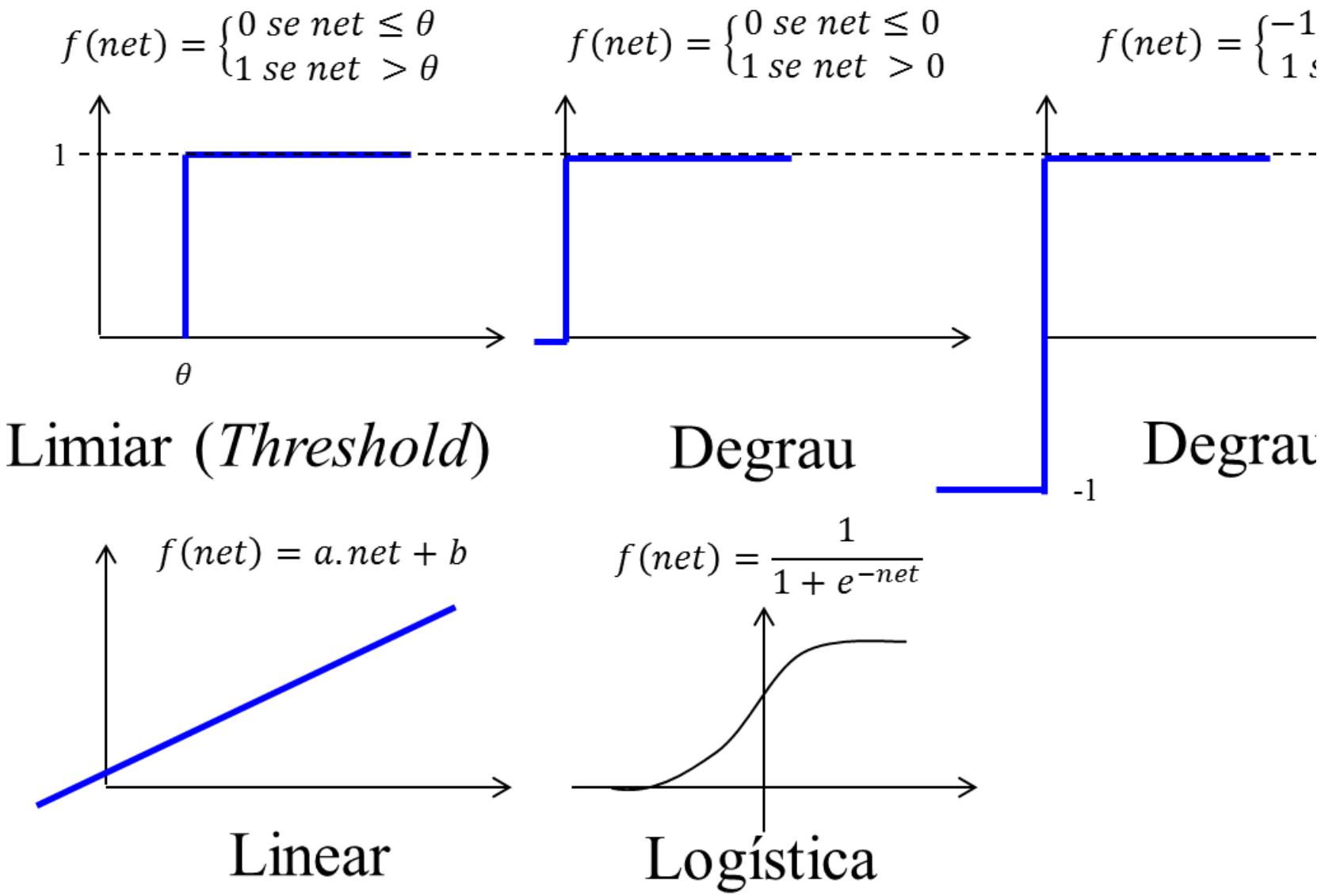
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





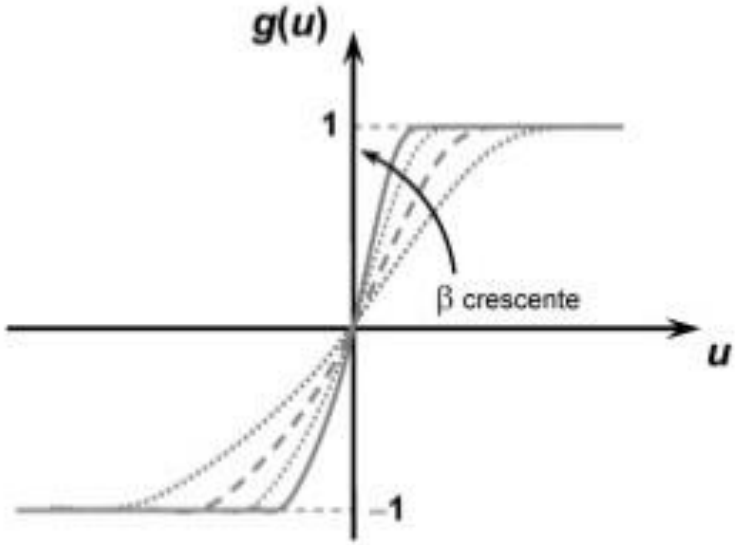
Neurônio Artificial

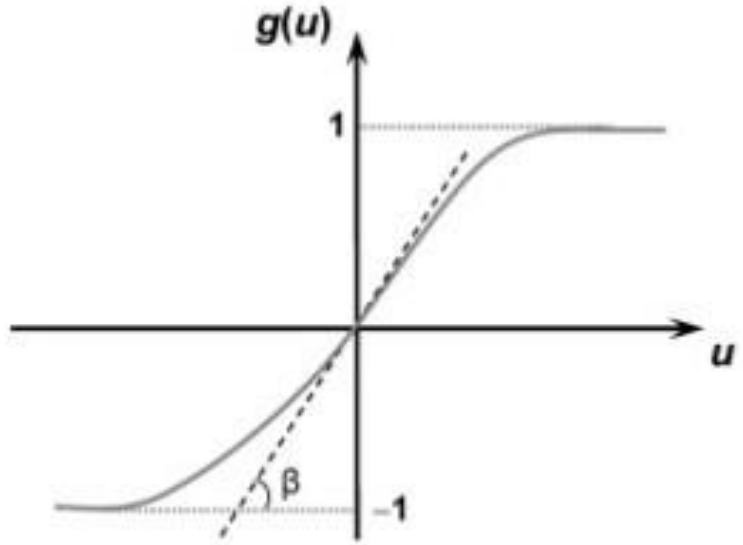
- Para a função de ativação ou função de transferência , existem várias possibilidades, como a função limiar, degrau, logística, etc

Neurônio Artificial

- Função de ativação ou função de transferência
- - número de Euler (= 2.718281...).
- - Constante de inclinação betha

- Função tangente hiperbólica $\longrightarrow$ $g(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}}$





Neurônio Artificial

Funções de Ativação

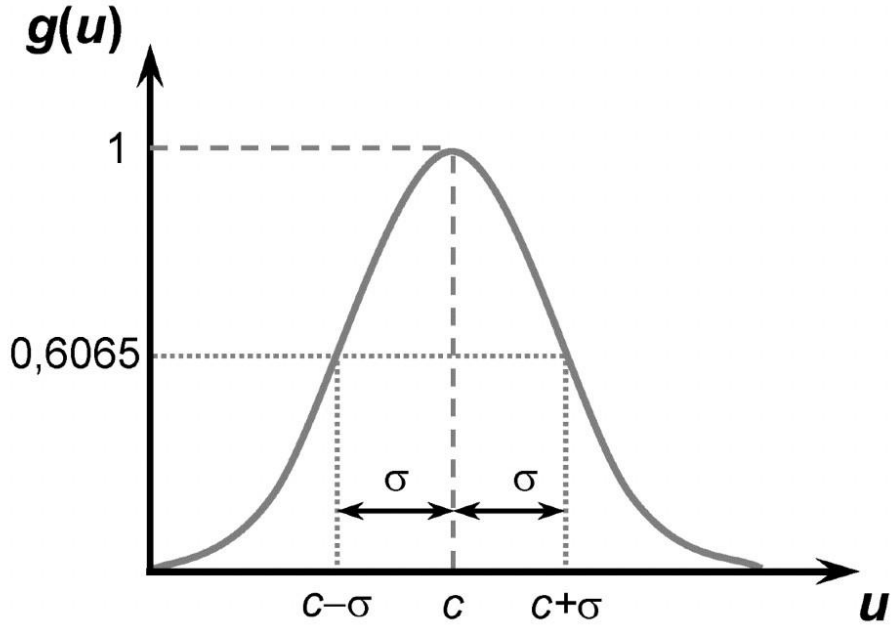
- Função Gaussiana

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}}$$

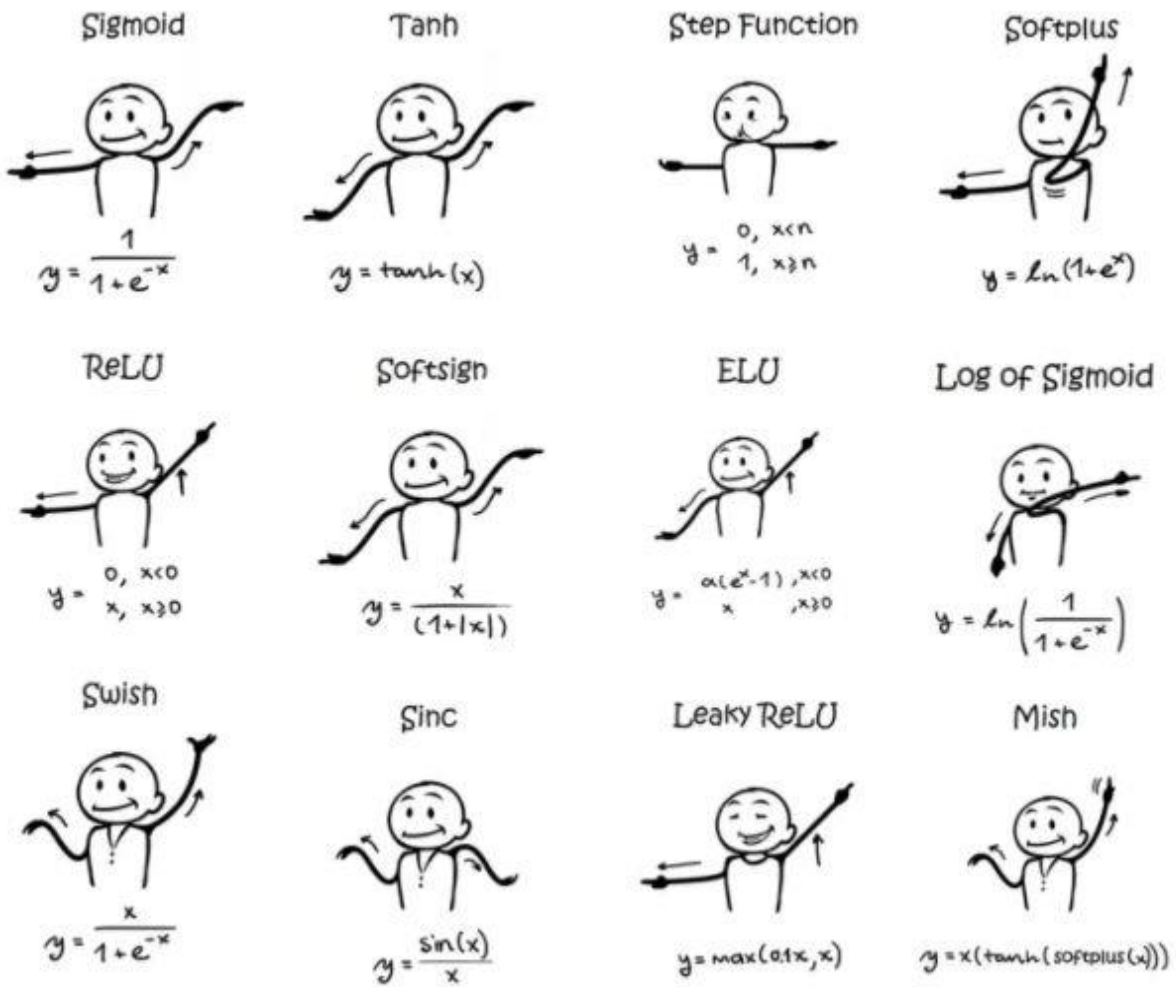
$e = 2{,}718281 \Rightarrow Número\ de\ Euler$

$\sigma \Rightarrow Desvio\ Padrão$

$c \Rightarrow Centro\ da\ Função\ Gaussiana$

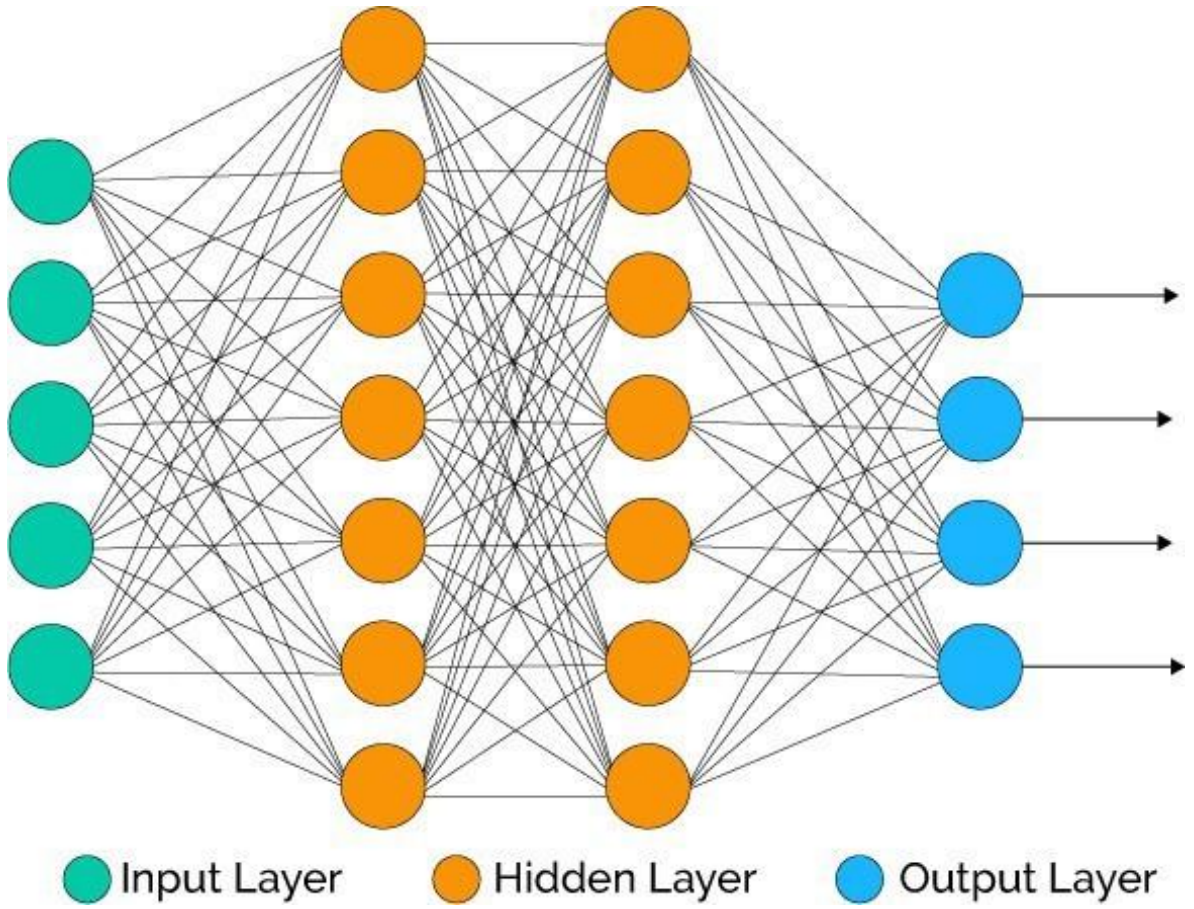


Funcão de Ativação

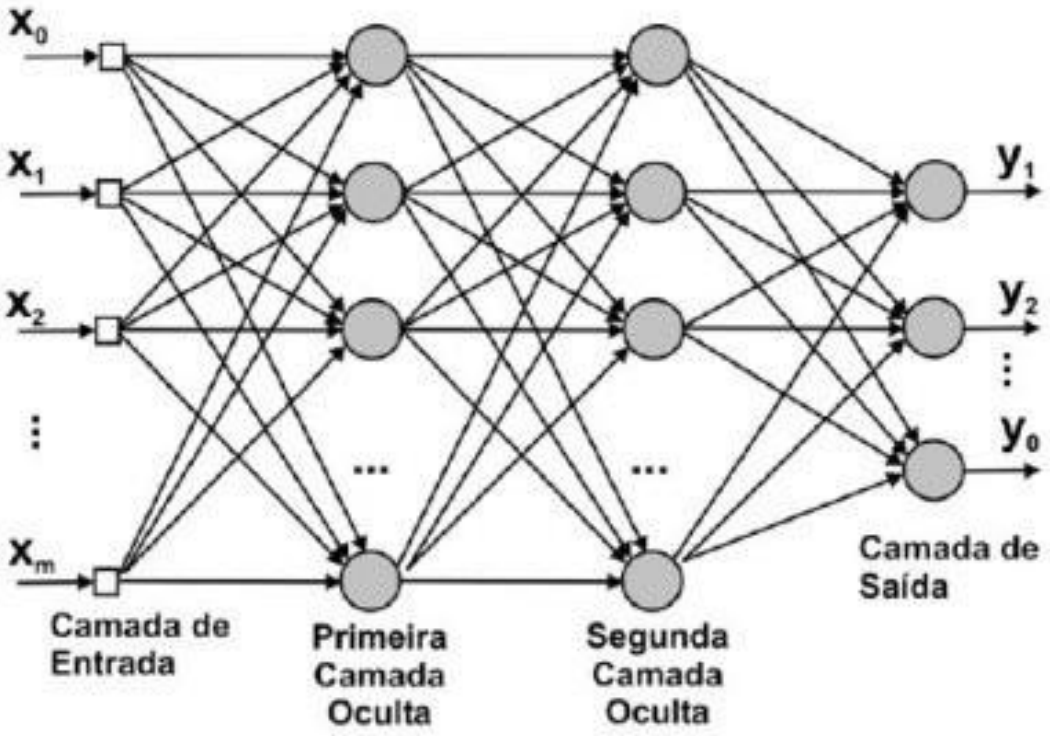


Arquitetura

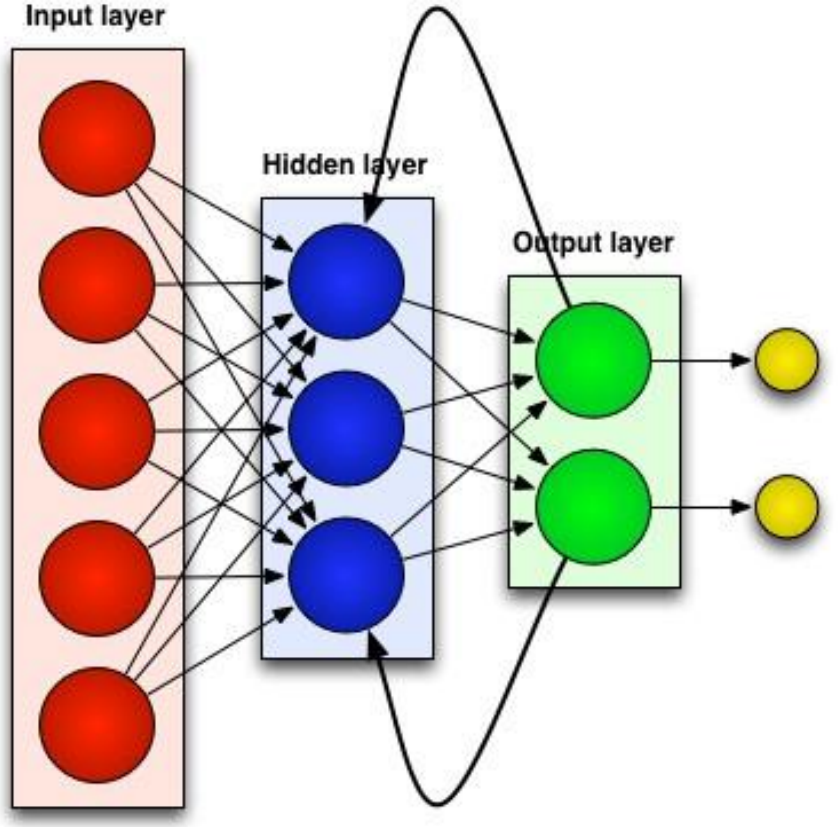
Em uma Rede Neural, os neurônios podem estar dispostos em uma ou mais camadas



Arquitetura



Rede Recorrente



Arquitetura

- A informação em uma rede neural geralmente flui da camada de entrada da rede para os neurônios da camada de saída (rede feedforward).

- Para redes multicamadas, eles podem apresentar ou não a retroalimentação (feedback).

- As conexões de retroalimentação permitem que um neurônio receba em seus terminais de entrada, a saída de um neurônio da camada posterior (ou da mesma camada).

Redes Neurais - Apredizado

- O aprendizado é dado pelo **ajuste de parâmetros**, que são os **pesos** associados às conexões de rede que fazem com que o modelo obtenha melhor desempenho (acurácia)
- Existem vários algoritmos que tem um conjunto de regras bem definidas que especificam quando e como deve ser alterado o valor de cada peso.
- Estes algoritmos estão divididos em 4 grupos: Correção de erro, Hebbiano, Competitivo e Boltwzmann.

Apredizado por correção de erro

- Geralmente utilizados em aprendizado supervisionado
- Eles procuram ajustar os pesos da RNA de forma a reduzir os erros cometidos pela rede

Redes Perceptron

- Foi a primeira RNA implementada
- É uma rede simples, apresentando apenas uma camada de neurônios
- Apresenta uma boa acurácia preditiva em diversos problemas de classificação

Redes Perceptron

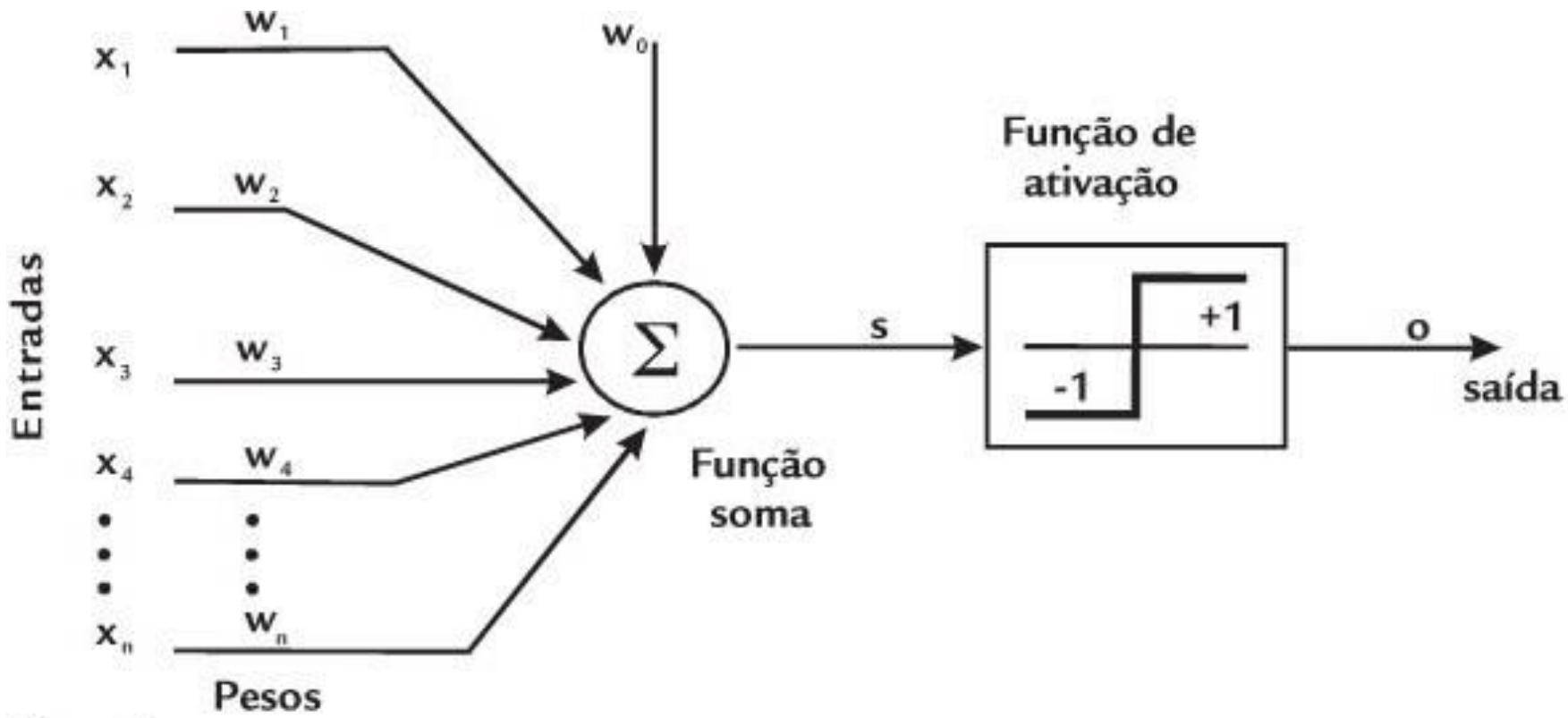


Figura 1

Modelo de um neurônio perceptron de Rosenblatt. Fonte: Adaptado de Medeiros (2006, p. 3).

Redes Perceptron

- É treinada por um algoritmo supervisionado de correção de erro
- Usa a função de ativação tipo limiar
- Durante seu treinamento, para um objeto , os pesos são ajustados de acordo com a equação:

$$Erro = y\_observado - y\_previsto$$

$$W = W\_anterior + Taxa\_Aprendizado * Xi * erro * f\,'\,(x)$$

- Em que é o peso da j-ésima conexão de entrada no instante de tempo ,
- é a taxa de aprendizado, que define a magnitude do ajuste feito no valor de cada peso.

Esse magnitude vai definir a velocidade de convergência

- é o valor do j-ésimo atributo do vetor de entrada ,
- é a saída esperada pela rede num instante de tempo e é a saída desejada para a rede.

Redes Perceptron

Erro = y_observado - y_previsto

W = W_anterior + Taxa_Aprendizado * Xi * erro * f ' (x)

- **-Aprendizado Baseado em Erros Binários:**
- O algoritmo ajusta os pesos apenas quando a saída do Perceptron está incorreta.
- Ele calcula o erro como a diferença entre a saída prevista e a saída real, mas **usa o erro diretamente para corrigir os pesos**.

Algoritmo de treinamento das Redes Perceptron

Entrada: Conjunto de treinamento $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1,...n\}$

Saída: Rede Perceptron com pesos ajustados

Iniciar pesos da rede com valores baixos

repita

  para cada $\mathbf{x}_i$ **faça**

    Calcular valor da saída produzida pela rede $\hat{f}(\mathbf{x}_i)$

    erro $e = y_i - \hat{f}(\mathbf{x}_i)$

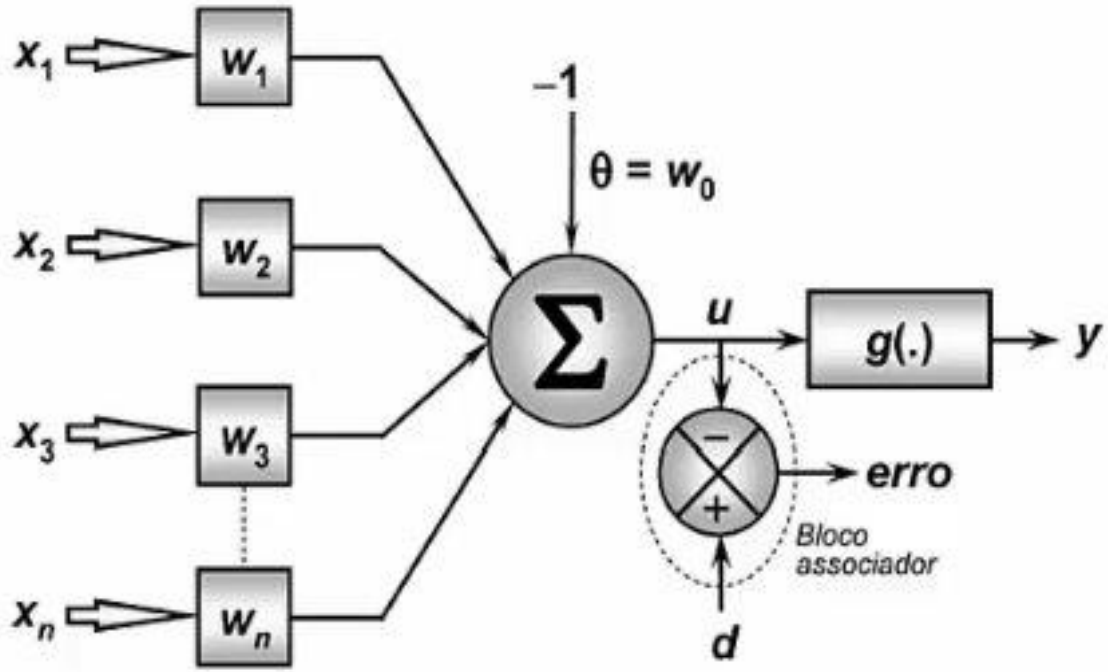
    se $e > 0$ **então**

      Ajustar pesos do neurônio $\mathbf{w}(t+1) = w(t) + \eta e \mathbf{x}(t)$

até que erro = 0 (ou erro < ε)

Redes ADALINE – Regra Delta

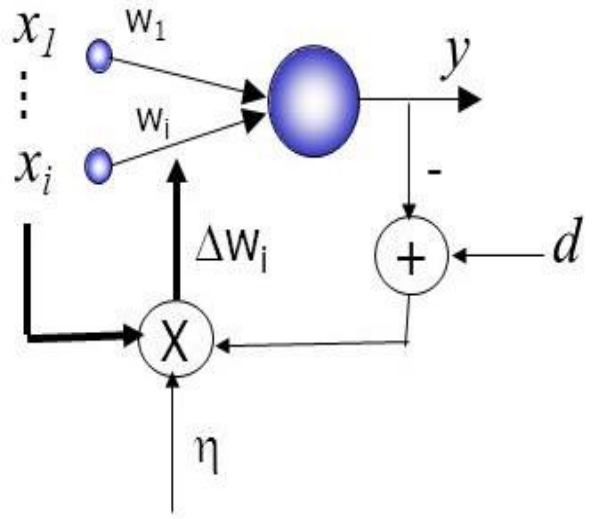
- A rede Adaline (Adaptive Linear Element) proposta por Widrow e Hoff (1960) tem a mesma estrutura do Perceptron, diferenciando apenas no algoritmo de treinamento.
- Enquanto o Perceptron ajusta os pesos somente quando um padrão é classificado incorretamente,
- o Adaline utiliza a Regra Delta para minimizar o erro médio (MSE) após cada padrão ser apresentado, ajustando os pesos proporcionalmente ao erro



Redes ADALINE – Regra Delta

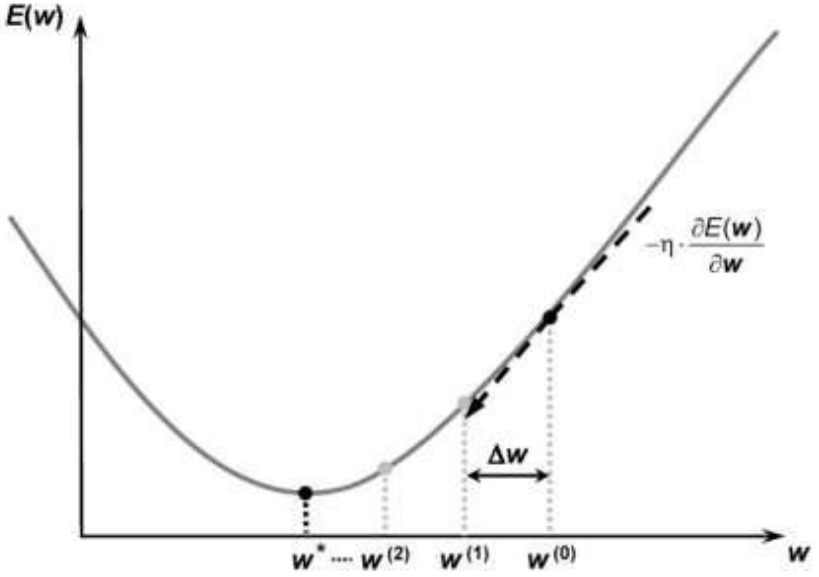
- Enquanto o Perceptron ajusta os pesos somente quando um padrão é classificado incorretamente, o Adaline utiliza a Regra Delta para minimizar o erro médio (MSE) após cada padrão ser apresentado, ajustando os pesos proporcionalmente ao erro

Treinamento: Regra Delta ou LMS



$$\Delta w_i = \eta . x_i . (d - y)$$

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



Redes ADALINE – Regra Delta

$$E = \frac{1}{2M}\sum_{p=1}^{M}\left(d^p - S^p\right)^2 = \frac{1}{2M}\sum_{p=1}^{M}\left[d^p - \sum_{i=0}^{N}\omega_i x_i^p\right]^2$$

- Inicialmente, atribui-se aos pesos valores aleatórios e, com eles, apresenta-se um conjunto de sinais de entrada e calcula-se a resposta da rede. Então, comparam-se os valores calculados com os valores desejados (treinamento supervisionado).

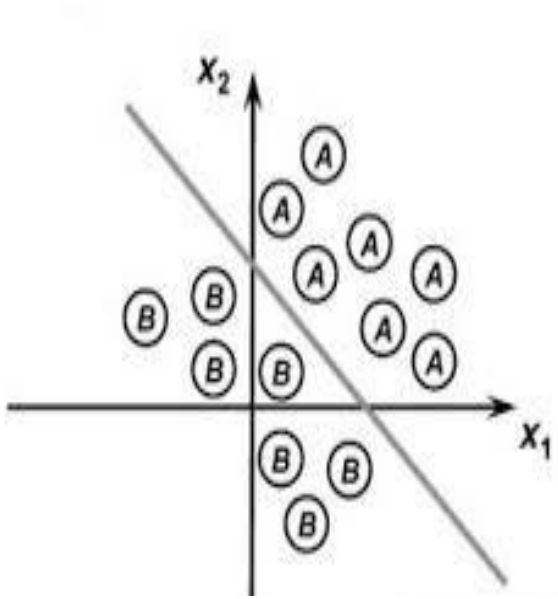
- **Método de Aprendizado:** A atualização dos pesos é feita por meio do método do **gradiente descendente**, que ***minimiza o erro médio quadrático***.

Redes ADALINE

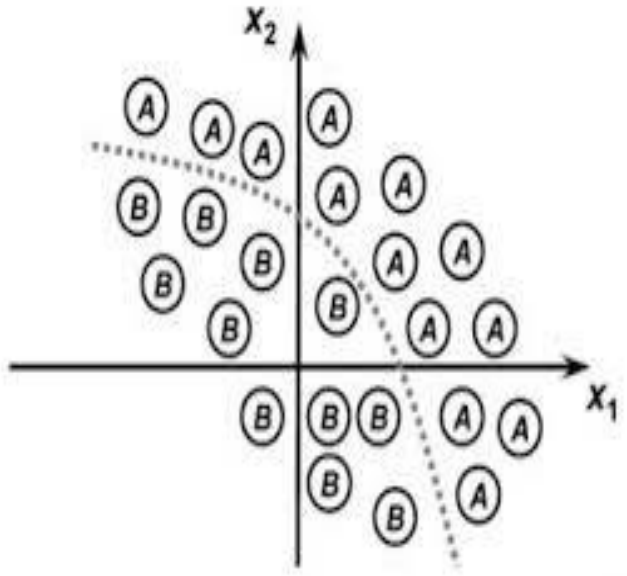
- A rede ADALINE utiliza uma função de ativação linear, levando assim, a magnitude do erro em consideração na hora de ajustar os pesos da rede (regra Delta)
- Para isto, utiliza uma regra de ajuste denominada Regra Delta, que é a diferença entre os valores desejados e a saída produzida.
- É definido como sendo contínuo.
- São comumente usados em problemas supervisionados de regressão.
- Em problemas de classificação, as saídas dos neurônios devem ser discretizadas.

Limitações de Perceptron e ADALINE

- Elas só conseguem classificar apenas objetos que são linearmente separáveis.



Objetos linearmente separáveis

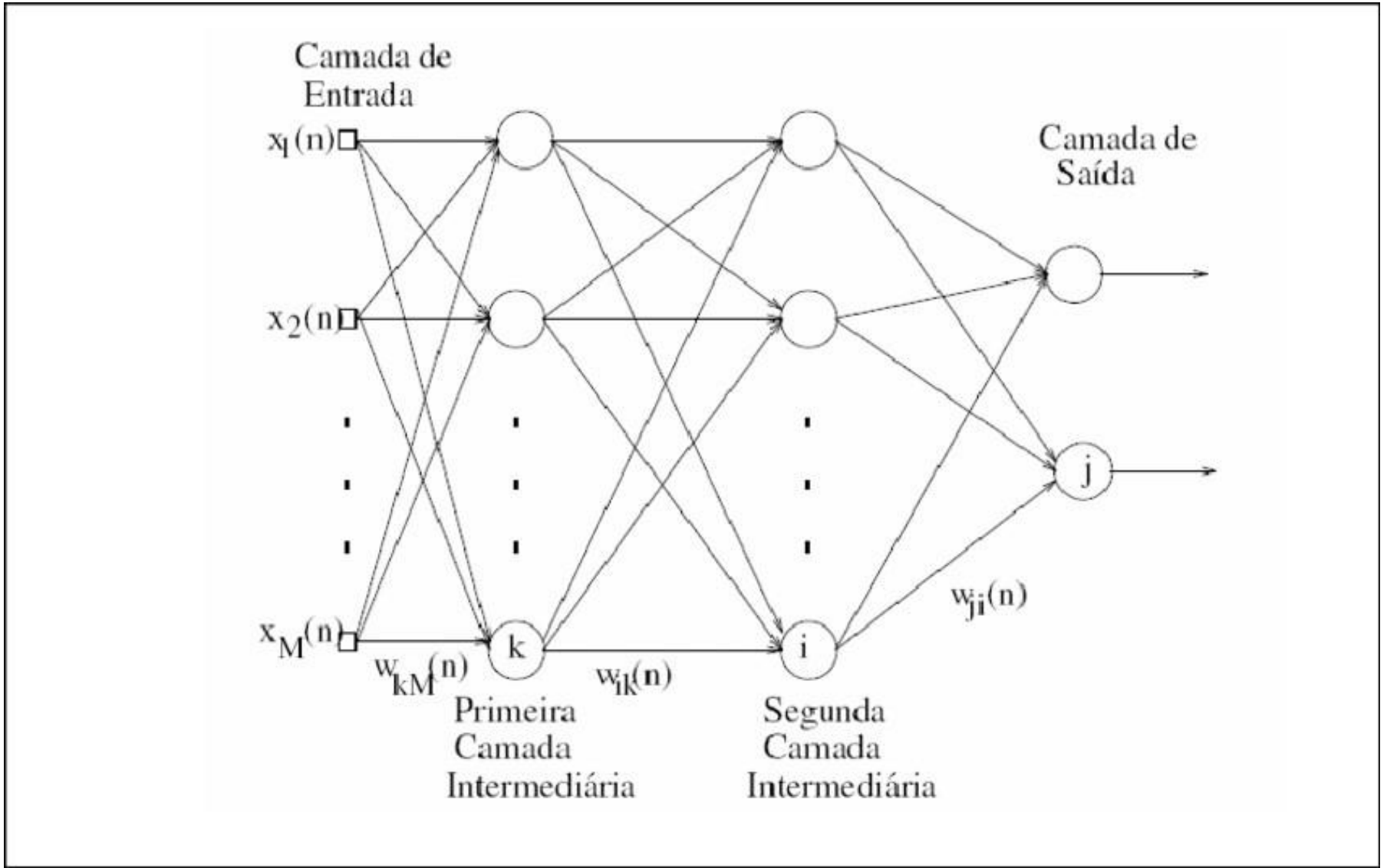


Separação não-linear

Perceptron Multicamadas

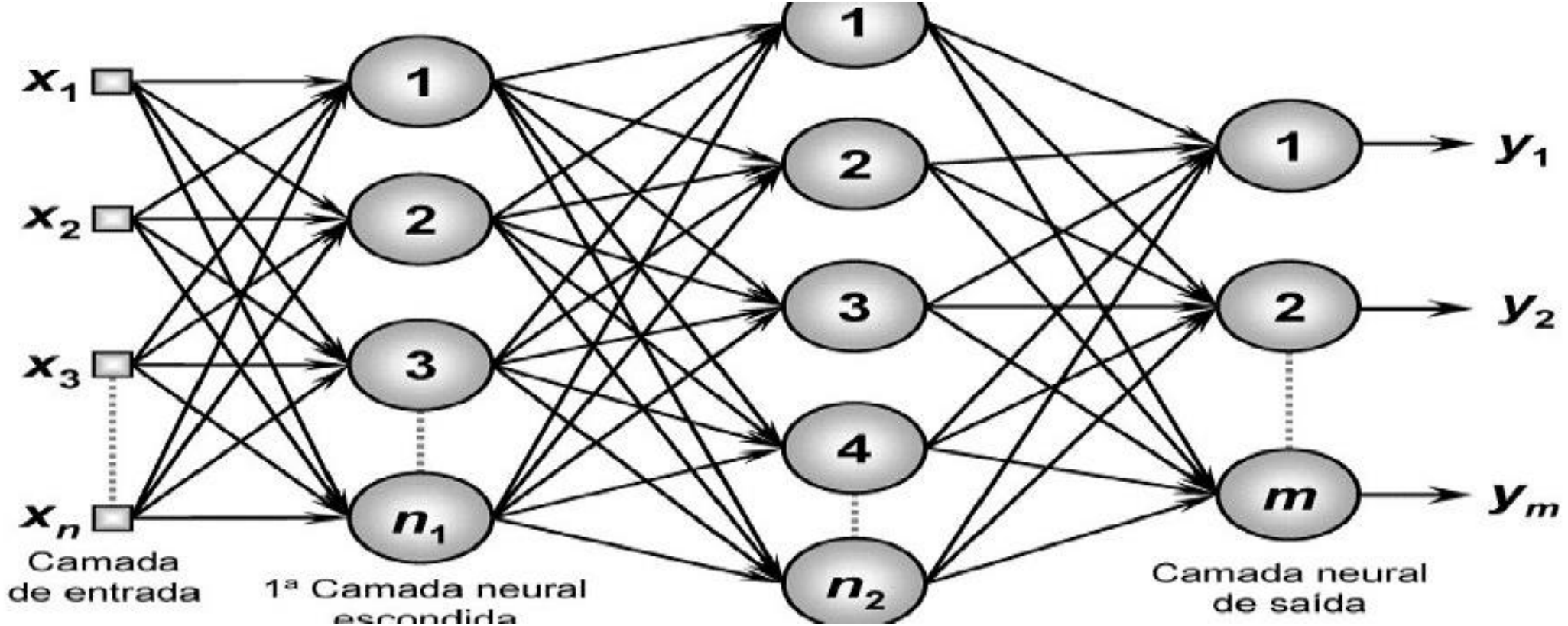
- Resolve o problema da separação não-linear
- A rede pode implementar qualquer função contínua
- As redes perceptron multicamadas (*MLP-multilayer perceptron*) apresentam uma ou mais camadas intermediárias (ou ocultas) de neurônios e uma camada de saída.

Perceptron Multicamadas



Perceptron Multicamadas

- Utilizam nas camadas intermediárias, funções de ativação não lineares, como a função sigmoidal e a função tangente hiperbólica.
- Em uma rede MLP, cada neurônio realiza uma função específica
- A função implementada por um neurônio de uma dada camada, é uma combinação das funções realizadas pelos neurônios da camada anterior que estão conectados a ele.
- É a combinação das funções desempenhadas por cada neurônio da rede que define a função associada à RNA como um todo.



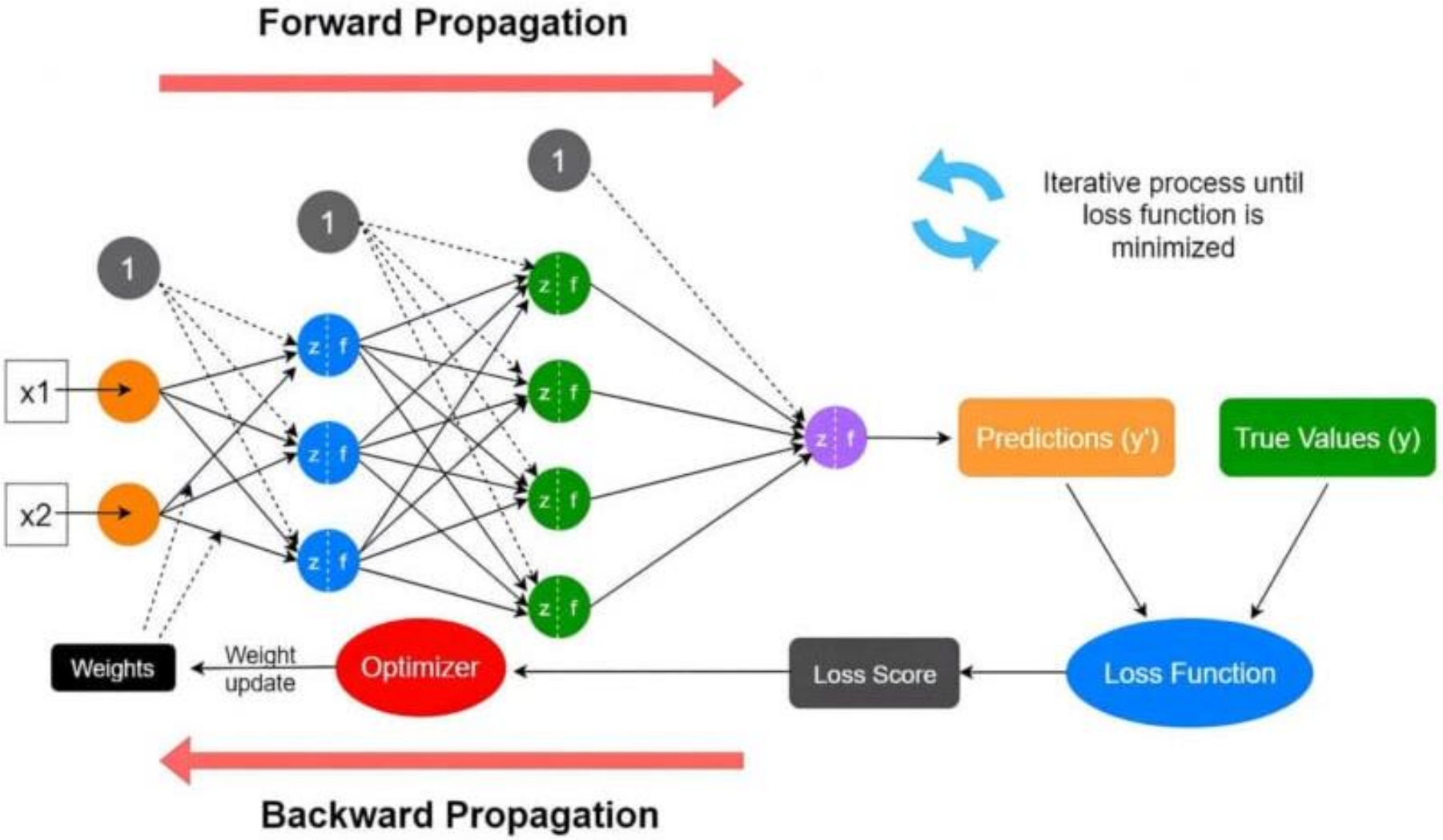
Perceptron Multicamadas

- Os valores gerados pelos neurônios de saída para um dado objeto de entrada, podem ser representados por um vetor,
- em que m é o número de neurônios da camada de saída (número de classes do problema)

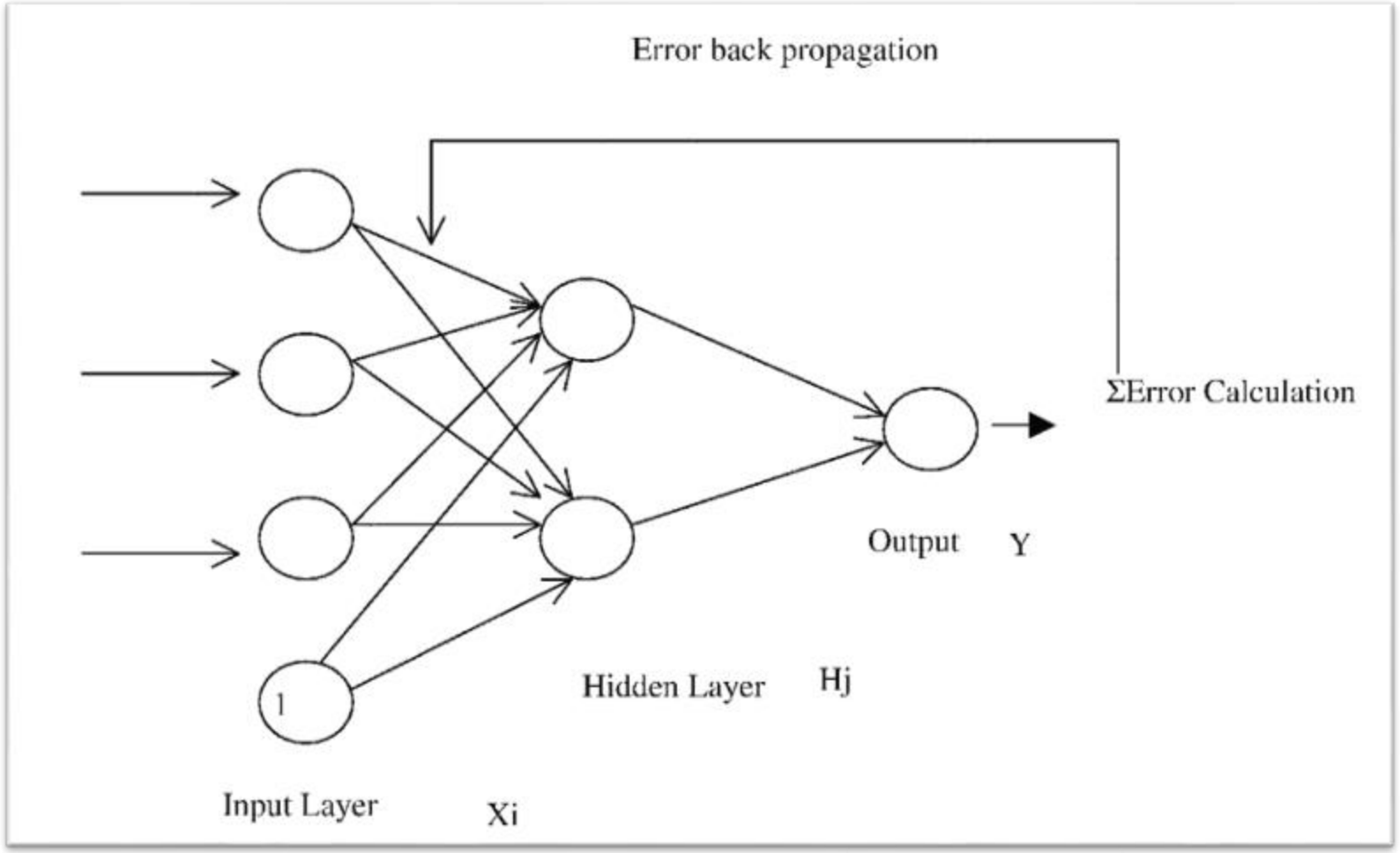
Algoritmo BackPropagation

- É baseado na regra Delta utilizada na rede ADALINE.
- Ele é constituído da iteração de duas fases:
- Uma fase para frente (forward)
- E um fase para trás (backward)

Algoritmo BackPropagation



Algoritmo BackPropagation



Algoritmo BackPropagation

- Na fase forward, cada objeto de entrada é apresentado à rede.
- O objeto é primeiramente recebido por cada um dos neurônios da primeira camada intermediária da rede, quando é ponderado pelo peso associado a suas conexões de entrada correspondentes.
- Cada neurônio nessa camada aplica uma função de ativação a sua entrada total e produz um valor de saída, que é utilizado como valor de entrada pelos neurônios da camada seguinte.
- Esse processo continua até que os neurônios da camada de saída produzam, cada um, o seu valor de saída.
- A diferença entres os valores de saída produzidos e desejados para cada neurônio da camada de saída, indica o erro cometido pela rede para o objeto apresentado.

Algoritmo Backpropagation

- O Valor do erro de cada neurônio da camada de saída é então utilizado na fase backward, para ajustar seus pesos de entrada.
- O ajuste prossegue da camada de saída até a primeira camada intermediária.
- $$w_{jl}(t+1) = w_{jl}(t) + \eta x_i^j \delta_l$$

  Nessa equação representa o peso entre um neurônio l e o j-ésimo neurônio da camada anterior
- x_i^j indica a entrada recebida por esse neurônio
- δ_l indica o erro associado ao l-ésimo neurônio

Algoritmo Backpropagation

O algoritmo propõe uma maneira de estimar o erro dos neurônios das camadas intermediárias, utilizando os erros observados nos neurônios da camada posterior.
A forma de calcular o erro:

$$\delta_l = \begin{cases} f'_a e_l, & e_l \in c_{saida} \\ f'_a \Sigma w_{lk} \delta_k, & e_l \in c_{interna} \end{cases}$$

w_{lk} é o l-ésimo neurônio

f_a' é a derivada parcial da função de ativação do neurônio

δ_l é o erro quadrático cometido pelo neurônio de saída quando sua resposta é comparada à desejada

Algoritmo Backpropagation

- O algoritmo propõe uma maneira de estimar o erro dos neurônios das camadas intermediárias, utilizando os erros observados nos neurônios da camada posterior.
- A forma de calcular o erro:

$$e_l = \frac{1}{2}\sum_{q=1}^{k}\left(y_q - \hat{f}_q\right)^2$$

- A derivada parcial define o ajuste dos pesos, utilizando o gradiente descendente da função de ativação.
- Essa derivada mede a contribuição de cada peso no erro da rede para a classificação de um dado objeto x.

Algoritmo Backpropagation

Entrada: X1, X2, ..., Xn (atributos)

Saída: Rede MLP com valores dos pesos ajustados

Inicializar: pesos = valores aleatórios, erro_total = 0

Repita

Para cada objeto xi do conjunto de treinamento:

Para cada camada de Rede:

Para cada Neurônio:

calcular o valor da saída produzida pelo neurônio

Calcular erro_parcial = y – y_prev

Para cada camada de Rede, a partir da camada de saída :

Para cada neurônio da camada atual:

Ajustar pesos do neurônio $w_{jl}(t+1) = w_{jl}(t) + \eta x_i^j \delta_l$

Calcular erro_total = erro_total + erro_parcial

Até erro_total < e

Algoritmo Backpropagation - Ajuste de hiperparâmetros

Momentum α

Incorpora as propriedades da atualização de pesos anterior e faz com que os pesos continuem sendo atualizados na mesma direção mesmo quando o erro diminui.

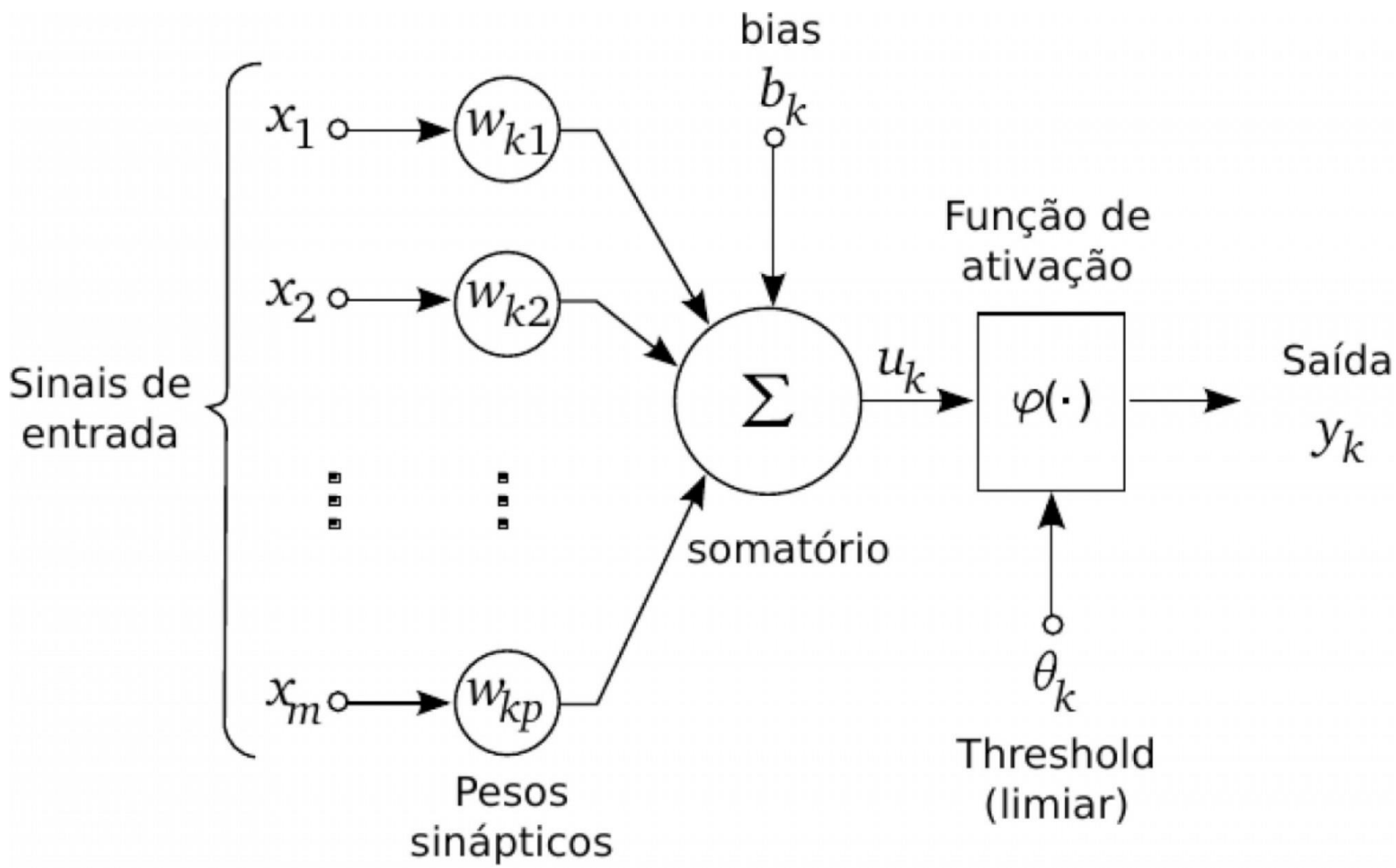
Learning Rate Decay η

Learning rate decay, é usado para diminuir o valor da learning rate conforme os erros diminuem.

Backpropagation – Ajuste de Hiperparâmetros

- O valor da taxa de aprendizado tem uma forte influência no tempo necessário à convergência da rede.
- Se a taxa de aprendizado for muito pequena, muitos ciclos podem ser necessários para induzir um bom modelo
- Se a taxa de aprendizado for elevada, pode provocar oscilações que dificultam a convergência.
- Uma possível solução para amenizar esse problema é a introdução do ***momentum***, que quantifica o grau de importância de variação de peso do ciclo anterior ao ciclo atual.

Neurônio Artificial



Redes Perceptron

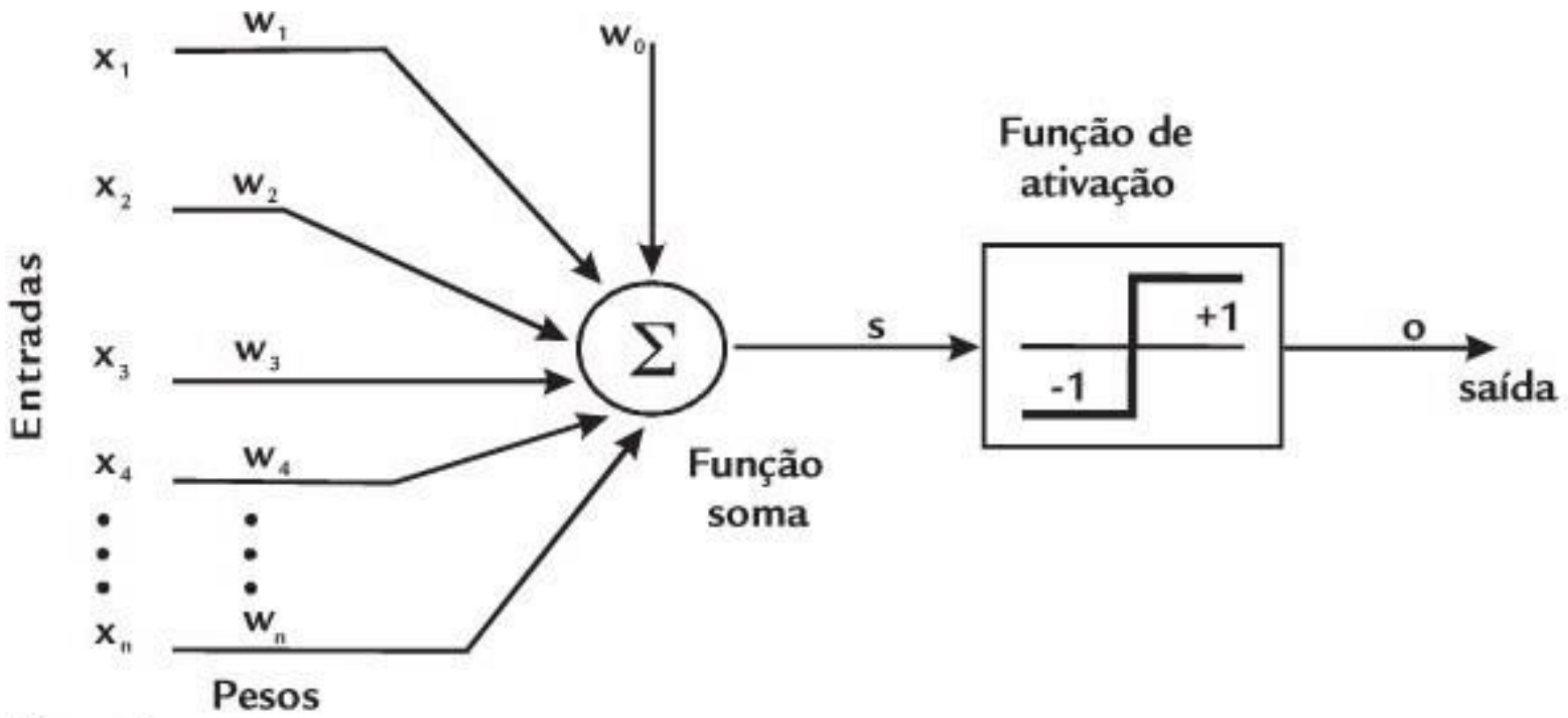
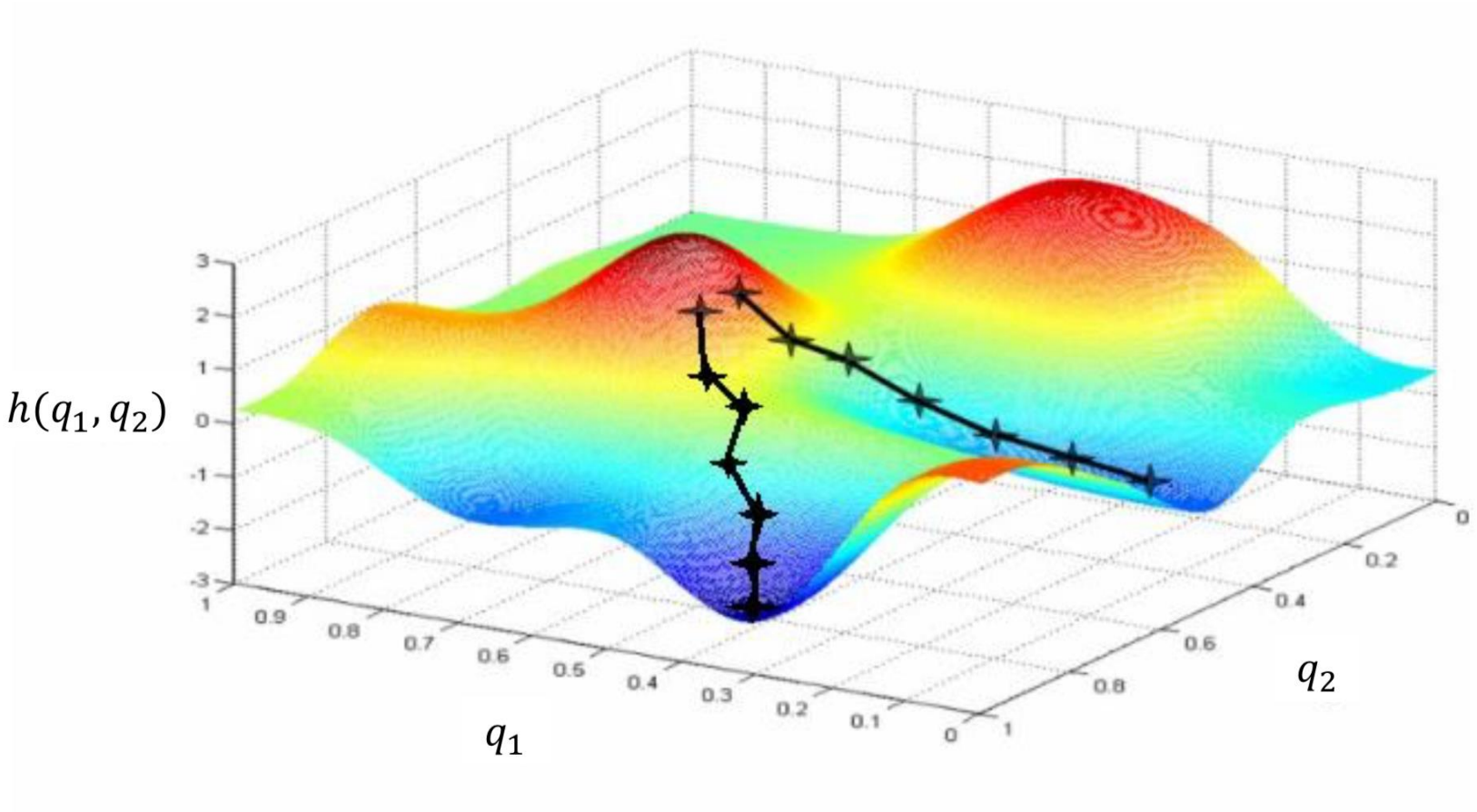


Figura 1

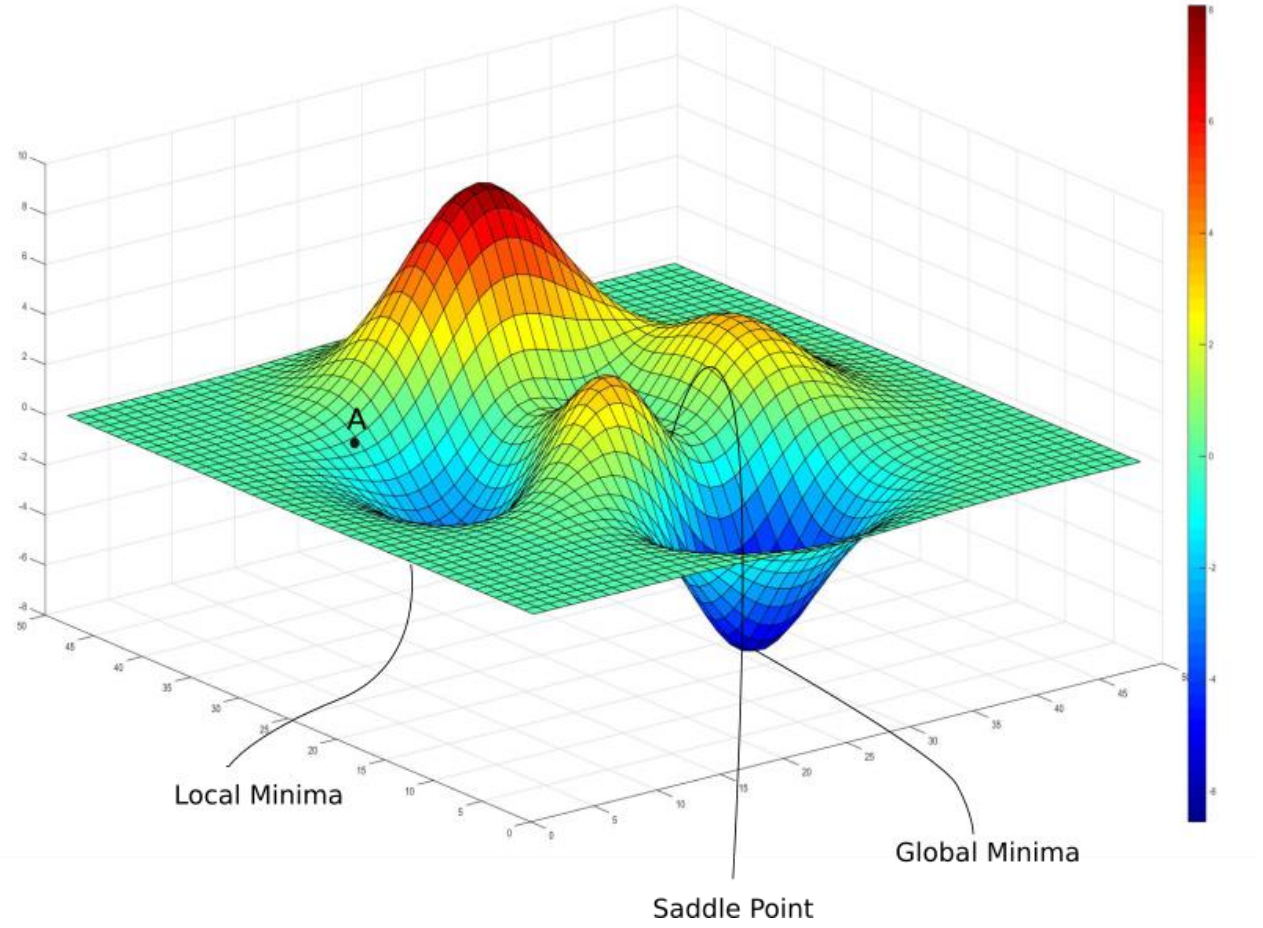
Modelo de um neurônio perceptron de Rosenblatt. Fonte: Adaptado de Medeiros (2006, p. 3).

Backpropagation – Convergência do algoritmo

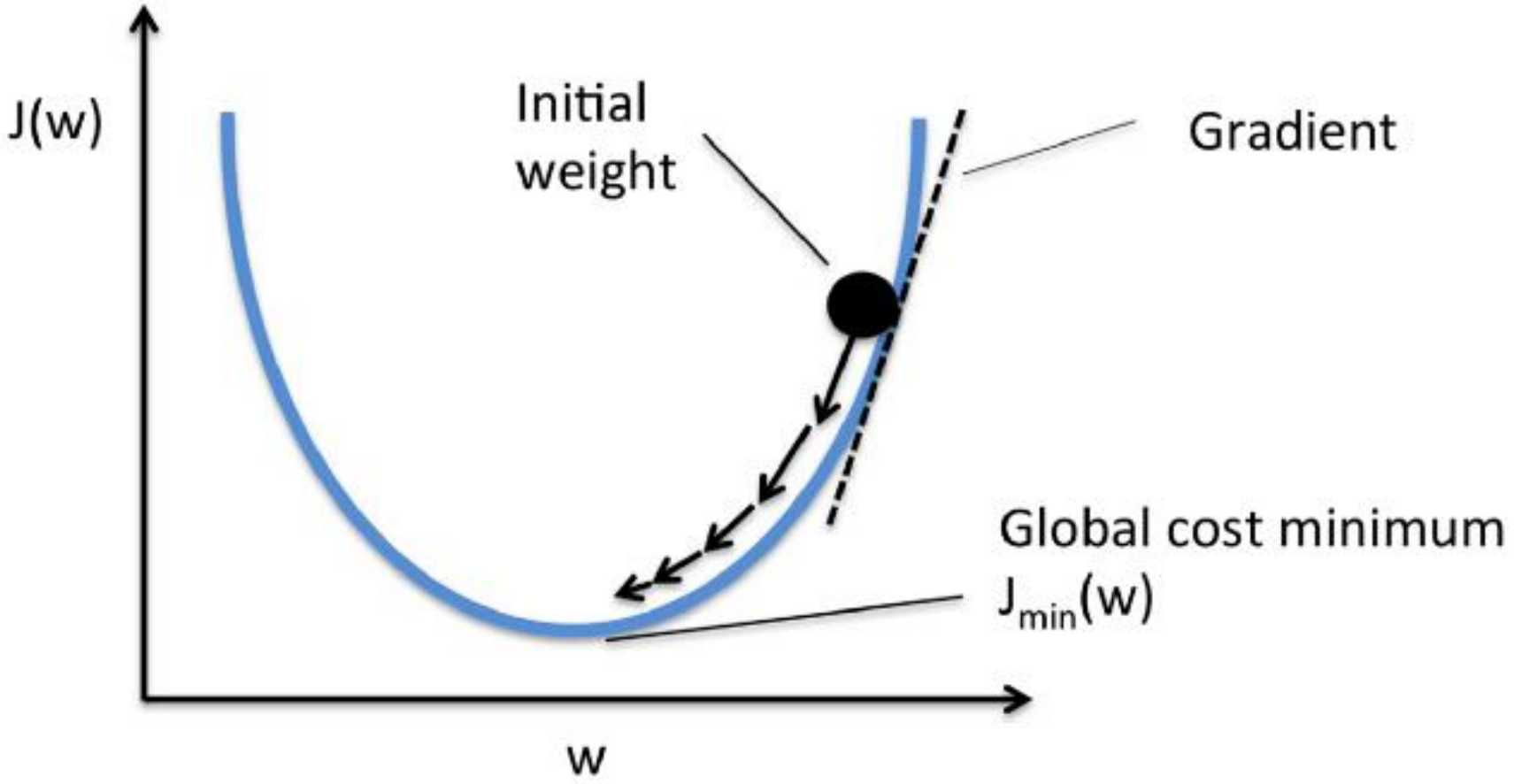
Passo =Taxa de Aprendizagem = Learning Rate (0.001)



Backpropagation – Convergência do algoritmo

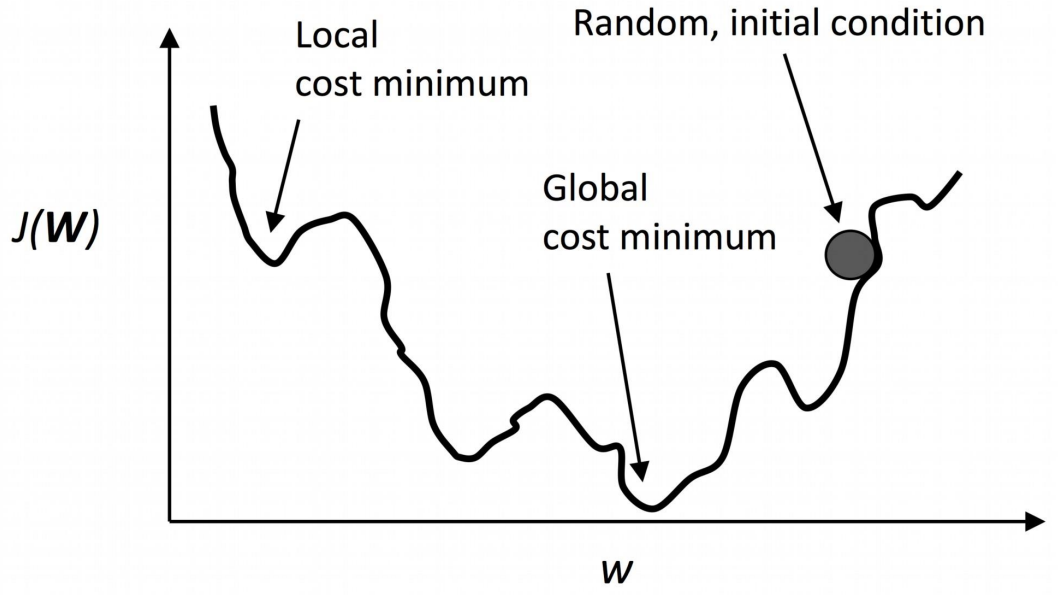


Backpropagation – Convergência do algoritmo

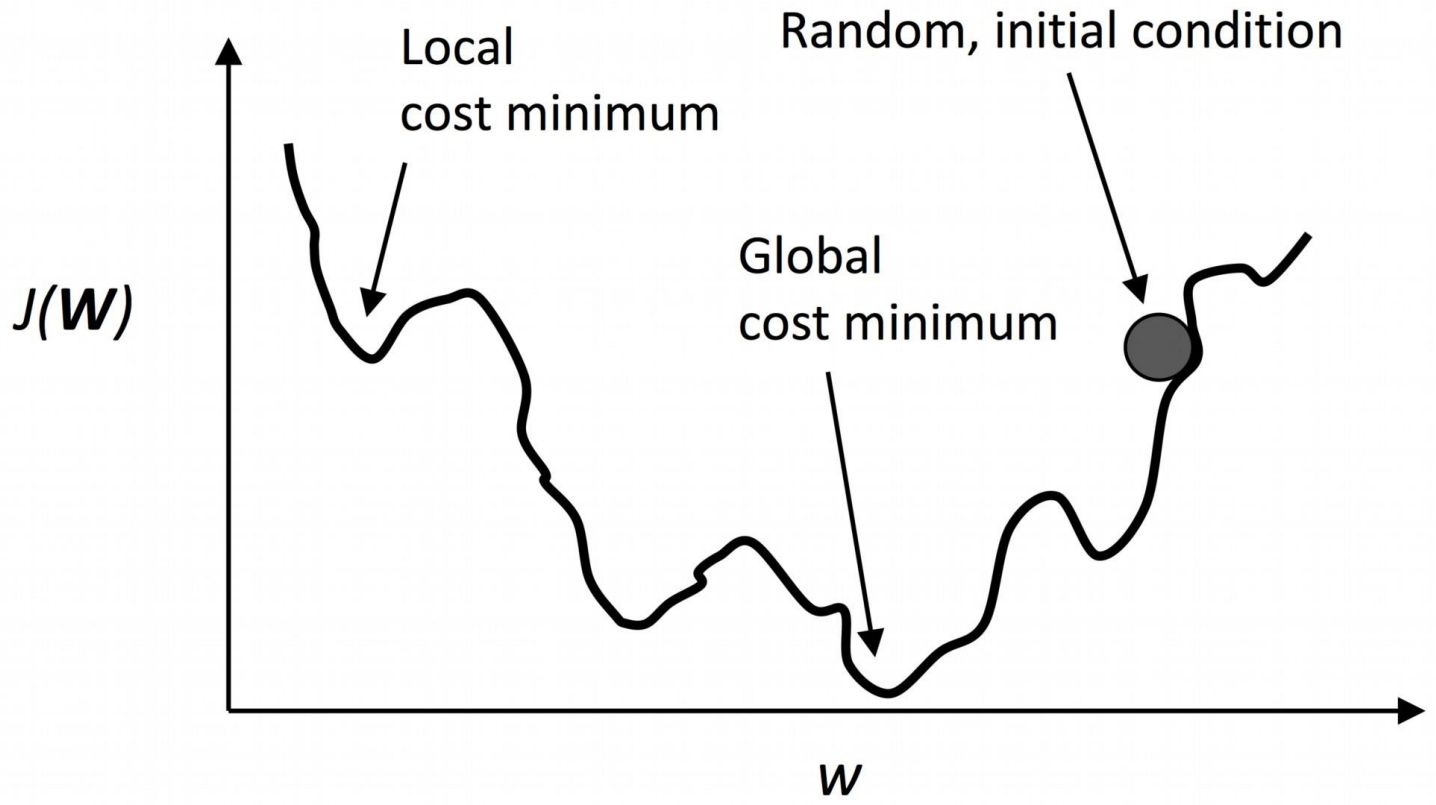


Backpropagation – Convergência do algoritmo

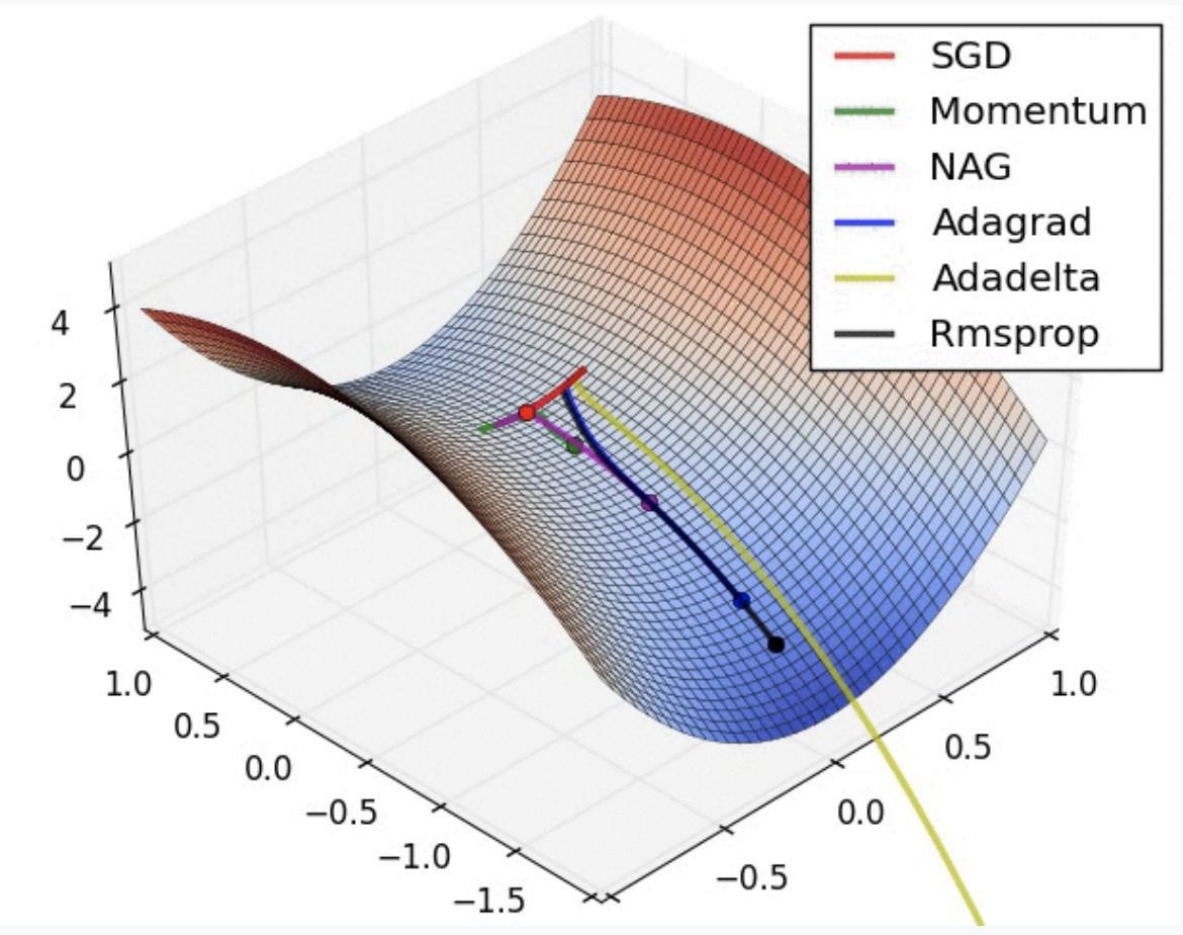
- A superfície de erro minimizada no backpropagation, apresenta regiões de mínimos locais e de mínimo global para problemas complexos.
- O objetivo do treinamento é atingir o mínimo global
- Entretanto, é possível que o treinamento com backpropagation fique numa região de mínimo local, fazendo com que a rede possua uma baixa acurácia preditiva.



Backpropagation – Convergência do algoritmo



Otimizadores
Cálculo de Mínimos Globais



Algoritmos de Otimização – Redes Neurais/Deep Learning

Gradiente Descendente (Gradient Descent)

- Batch Gradient Descent
- Stochastic Gradient Descent (SGD)
- Mini-batch Gradient Descent

Momentum

Nesterov Accelerated Gradient (NAG)

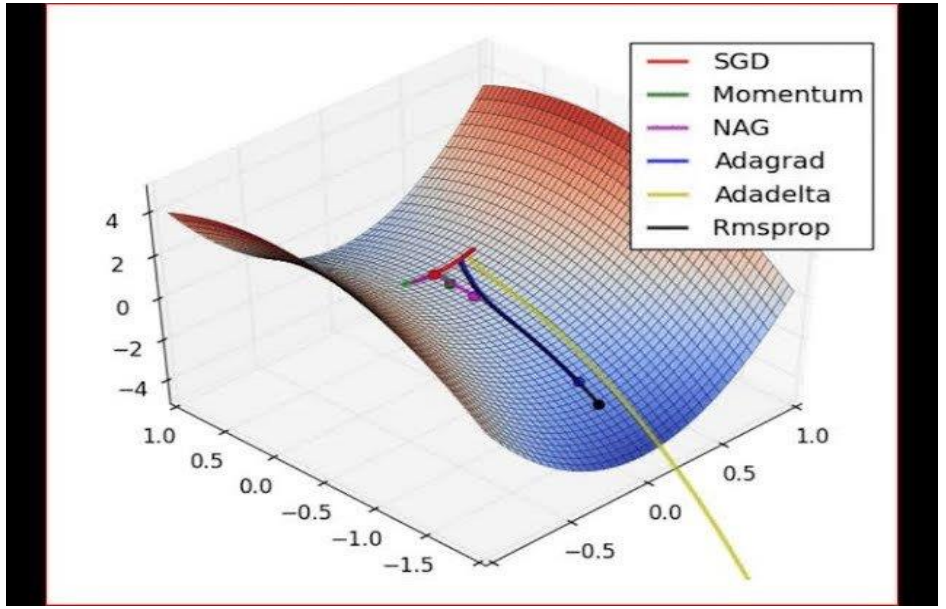
Adagrad (Adaptive Gradient Algorithm)

RMSprop (Root Mean Square Propagation)

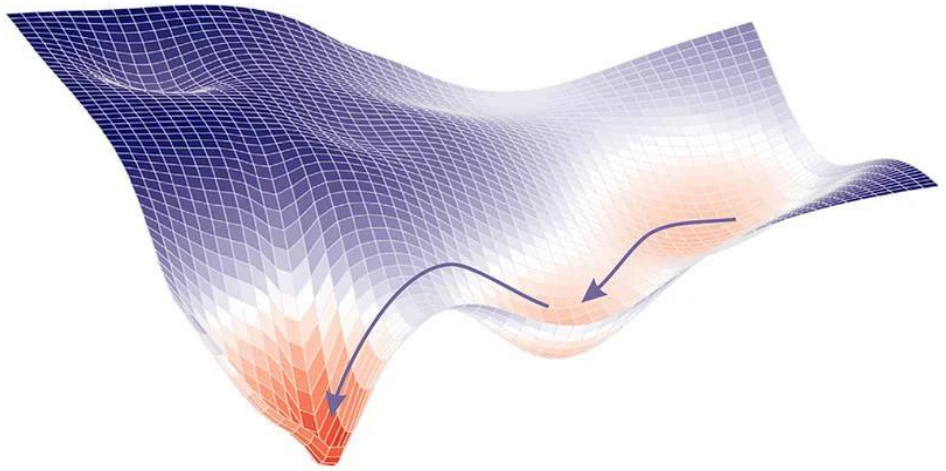
Adam (Adaptive Moment Estimation)

AdaDelta

Nadam (Nesterov-accelerated Adaptive Moment Estimation)



1. Gradiente Descendente (Gradient Descent)



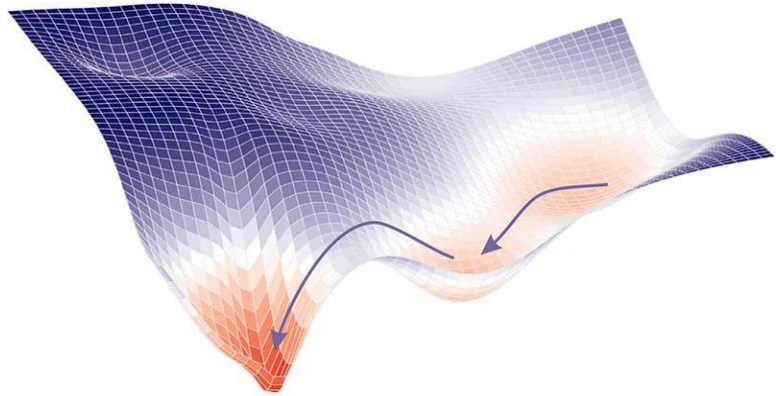
O **gradiente descendente** é o algoritmo de otimização mais básico e amplamente utilizado para treinar redes neurais. O objetivo é minimizar a função de perda ajustando os pesos da rede na direção oposta ao gradiente da função de custo.

Batch Gradient Descent: Calcula o gradiente de toda a função de perda para todo o conjunto de dados e, em seguida, atualiza os pesos.

Stochastic Gradient Descent (SGD): Atualiza os pesos para cada amostra de treinamento individual, o que pode acelerar a convergência, mas introduz mais variação.

Mini-batch Gradient Descent: Uma combinação dos dois, onde o gradiente é calculado para pequenos lotes (mini-batches) de amostras, tornando o treinamento mais eficiente.

2. Momentum



O **Momentum** é uma extensão do gradiente descendente que acelera a convergência em direções consistentes e reduz as oscilações.

Ele introduz uma "velocidade" que é acumulada em cada iteração, permitindo que o algoritmo supere áreas planas da superfície de perda.

Vantagem: Acelera o treinamento e ajuda o modelo a escapar de mínimos locais.

3. RMSprop (Root Mean Square Propagation)

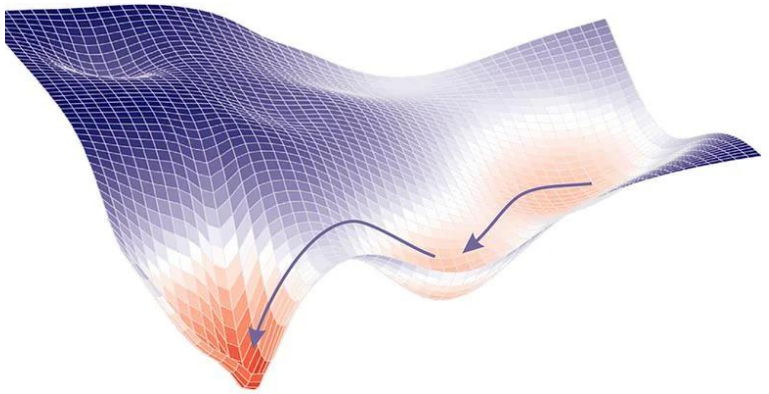
RMSprop ajusta o passo de atualização do gradiente dividindo pelo valor médio quadrado do gradiente.

Isso evita grandes oscilações nos pesos e acelera a convergência.

Vantagem: Adaptativo ao tamanho do passo para cada peso, o que o torna eficiente para redes neurais profundas.

4. **Adagrad (Adaptive Gradient Algorithm)**

Adagrad ajusta a taxa de aprendizado individualmente para cada parâmetro com base na soma acumulada dos quadrados dos gradientes anteriores. Isso significa que parâmetros que raramente mudam recebem uma taxa de aprendizado maior.



- **Vantagem**: Útil para lidar com gradientes esparsos e problemas em que certos parâmetros têm variação maior que outros.
- **Desvantagem**: A taxa de aprendizado pode se tornar muito pequena com o tempo, retardando a convergência.

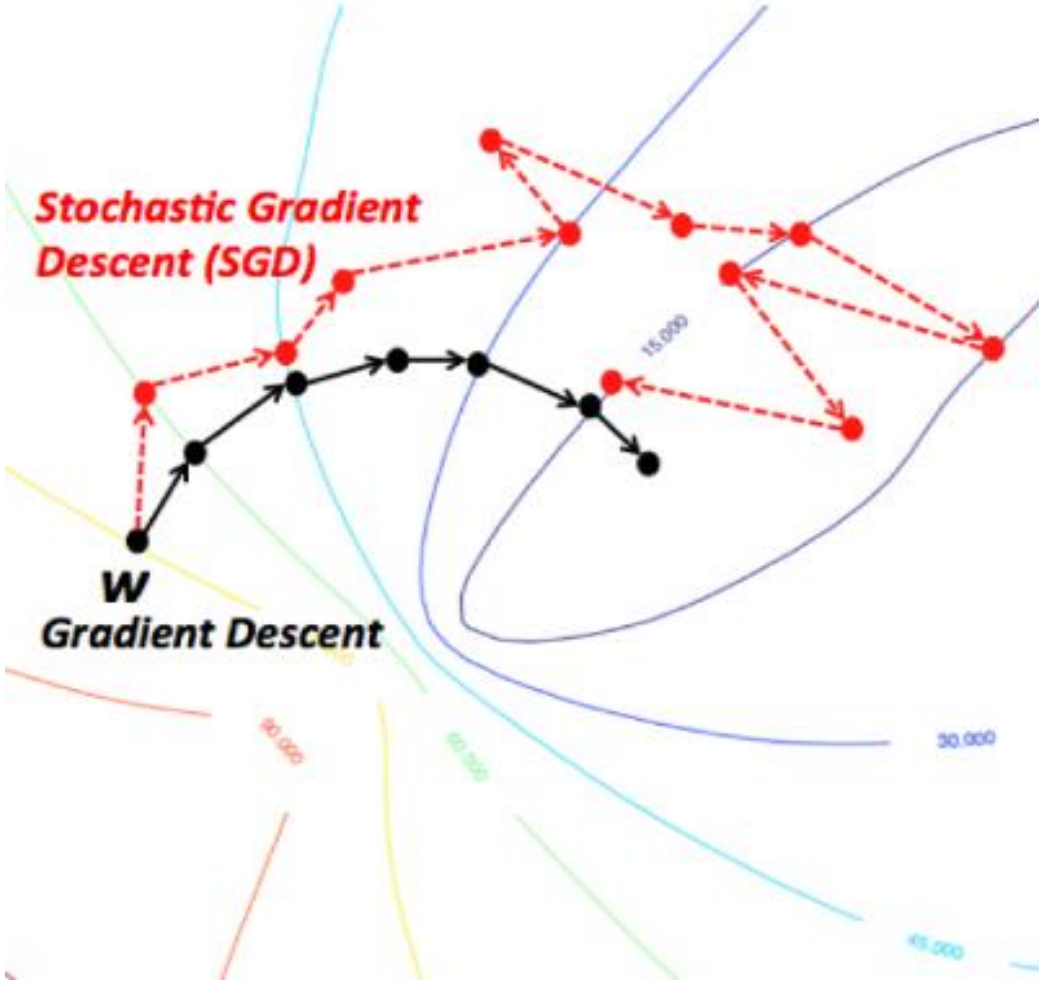
5. **Adam (Adaptive Moment Estimation)**

Adam combina as ideias de **Momentum** e **RMSprop**, armazenando médias em movimento do gradiente e do quadrado do gradiente, adaptando a taxa de aprendizado de forma eficiente para cada peso.

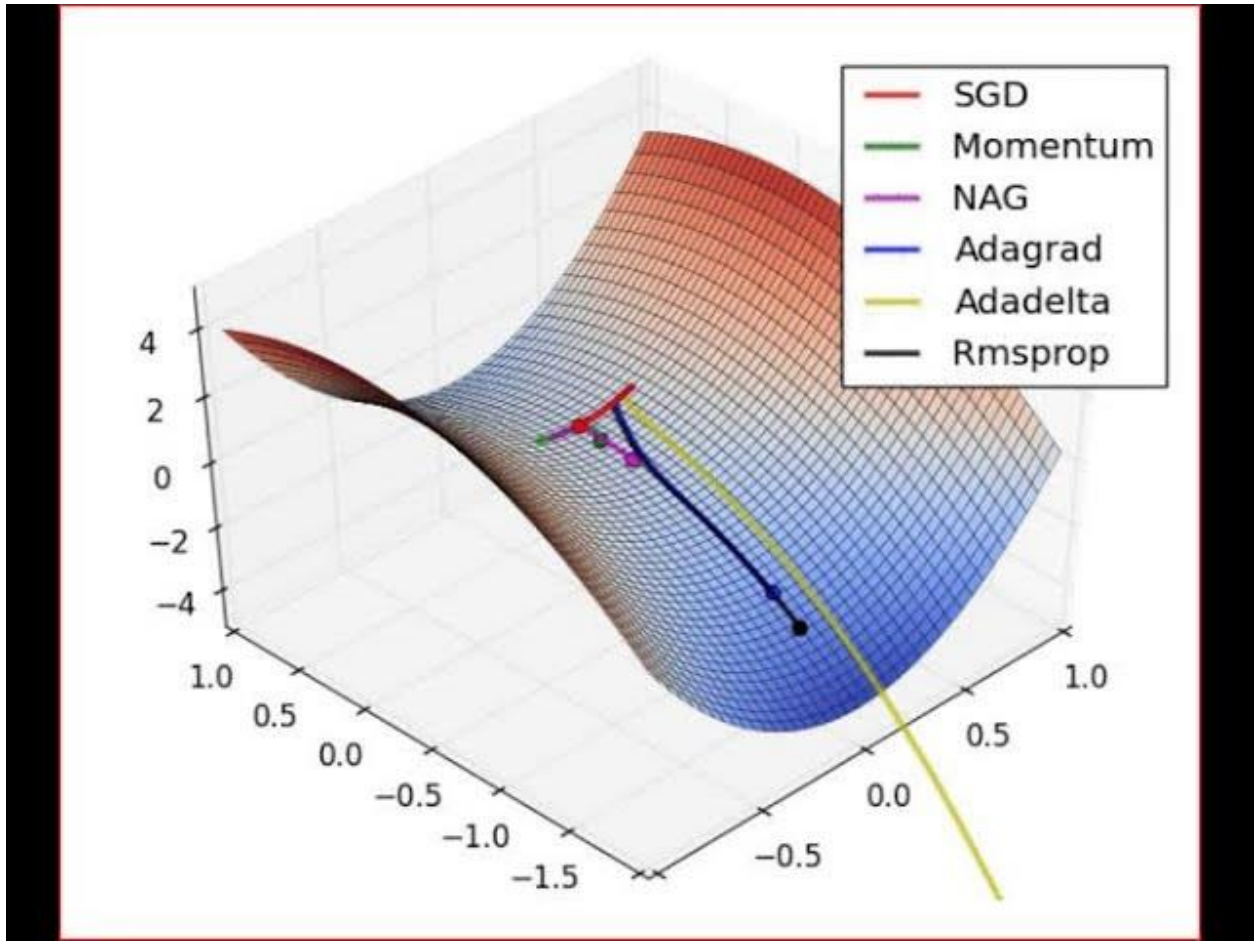
- **Vantagem**: Um dos algoritmos de otimização mais populares para deep learning, devido à sua capacidade de convergir rapidamente e lidar bem com grandes conjuntos de dados.
- **Desvantagem**: Embora seja amplamente utilizado, pode não funcionar bem em todos os problemas.

.

Gradient Descend x Stochastic Gradiente



Algoritmos de Otimização – Redes Neurais/Deep Learning



Gradient Descend x Stochastic Gradiente