



Instituto Tecnológico de Aeronáutica

---

# **Introdução às Redes Neurais e aos Grandes Modelos de Linguagem**

**Prof. Mauri Aparecido de Oliveira**

---

**CURSO – Extensivo**

**PO-249**

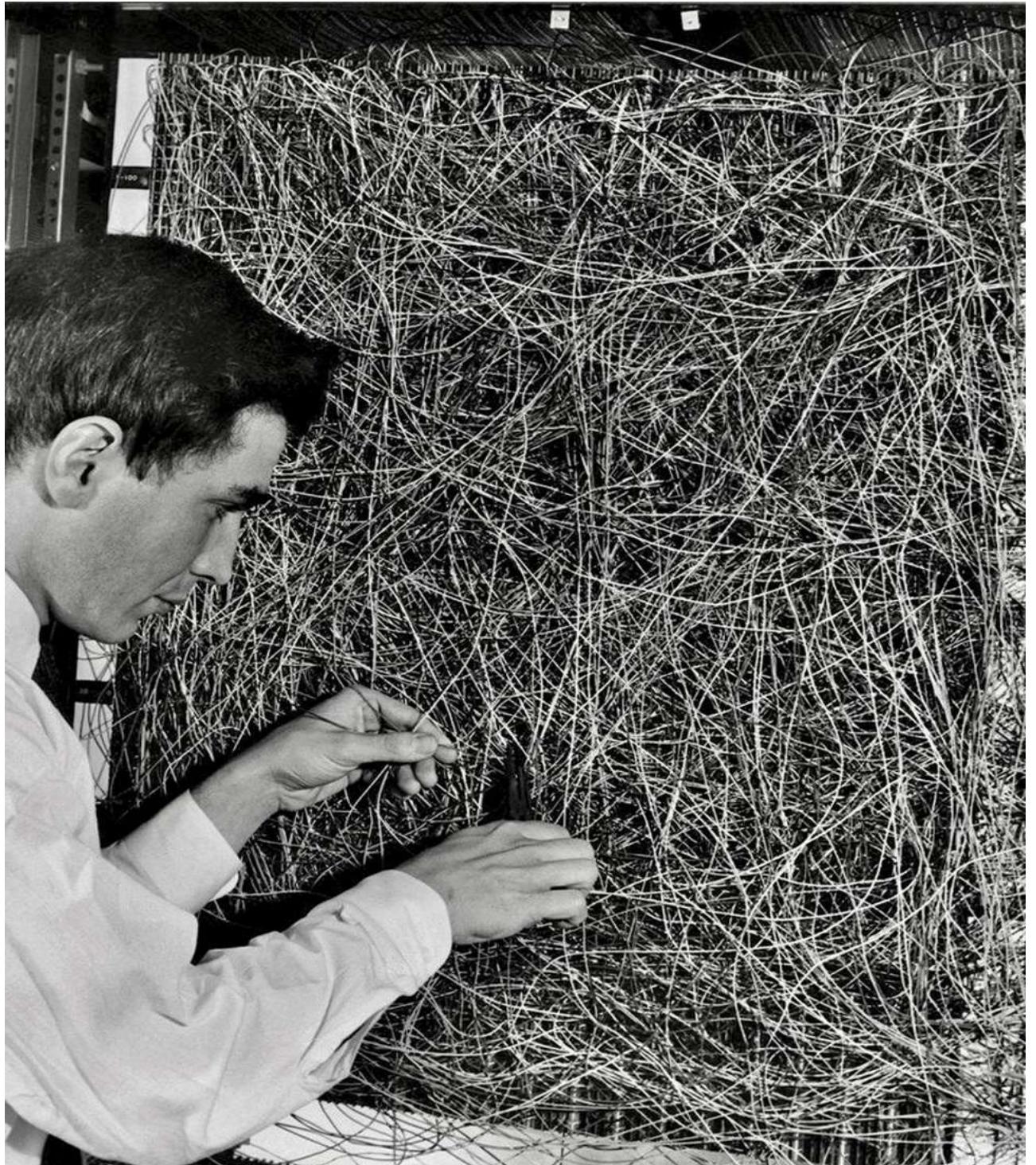


**Perceptron**

ai

# PERCEPTRON

**Frank Rosenblatt**  
com um computador Mark I  
Perceptron em 1960



Os neurônios processam e transmitem informação. Para enviar e receber informação dos diferentes tipos de células, os neurônios possuem conexões especializadas para realizar estas tarefas.

Apesar da forma dos neurônios naturais variar em algum sentido eles são compostos de maneira geral por quatro regiões específicas responsáveis por realizar funções distintas, estas partes são: o corpo celular, os dendritos, o axônio e os terminais sinápticos. O perceptron é a representação mais simples de uma rede neural, de tal forma que incorpora as partes básicas do neurônio em sua arquitetura.

# Santiago Ramón y Cajal: O primeiro a mapear o cérebro humano



O neurocientista, patologista e artista espanhol Santiago Ramón y Cajal (1852-1934) era fascinado pelo cérebro. Suas ilustrações complexas, belas e precisas do funcionamento interno do cérebro ainda são usadas na neurociência para demonstrar a arquitetura neural subjacente à memória e ao pensamento humano.

# List of Nobel laureates in Physiology or Medicine

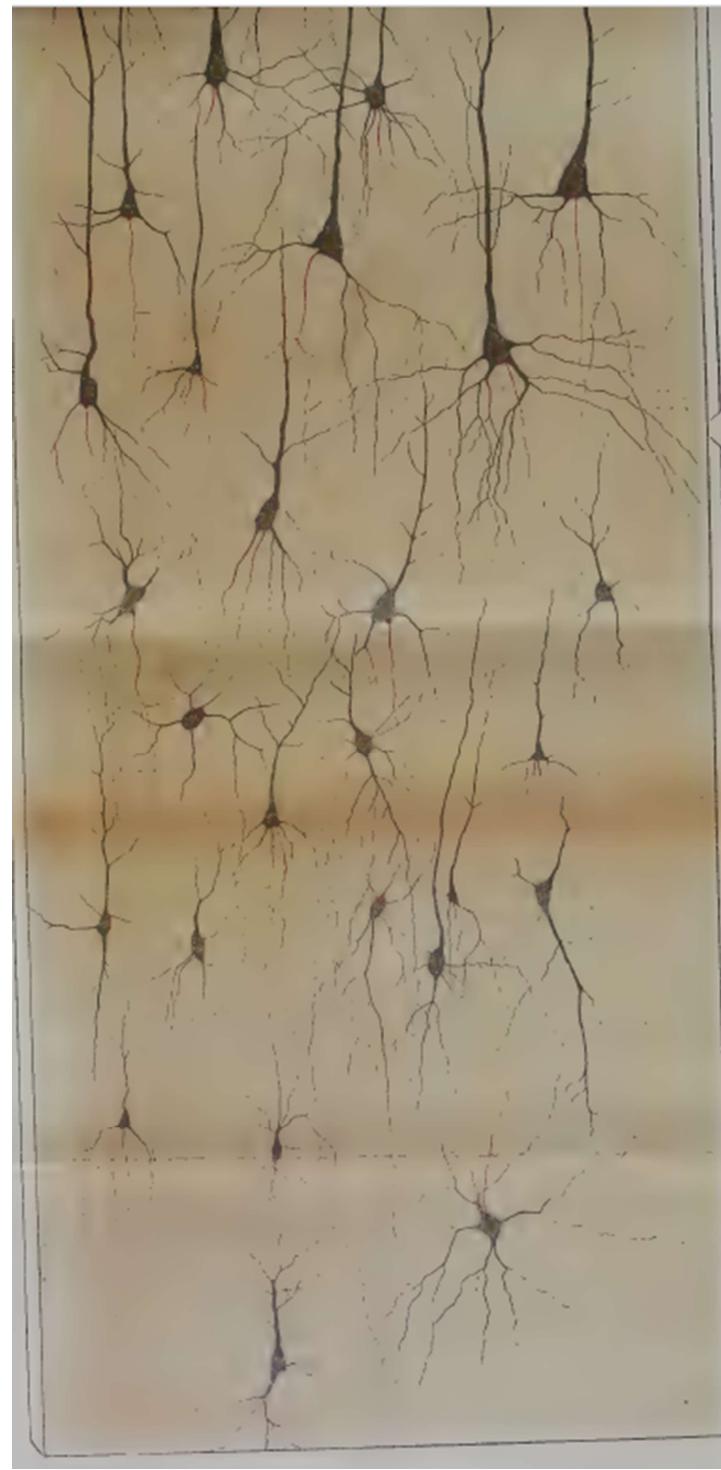
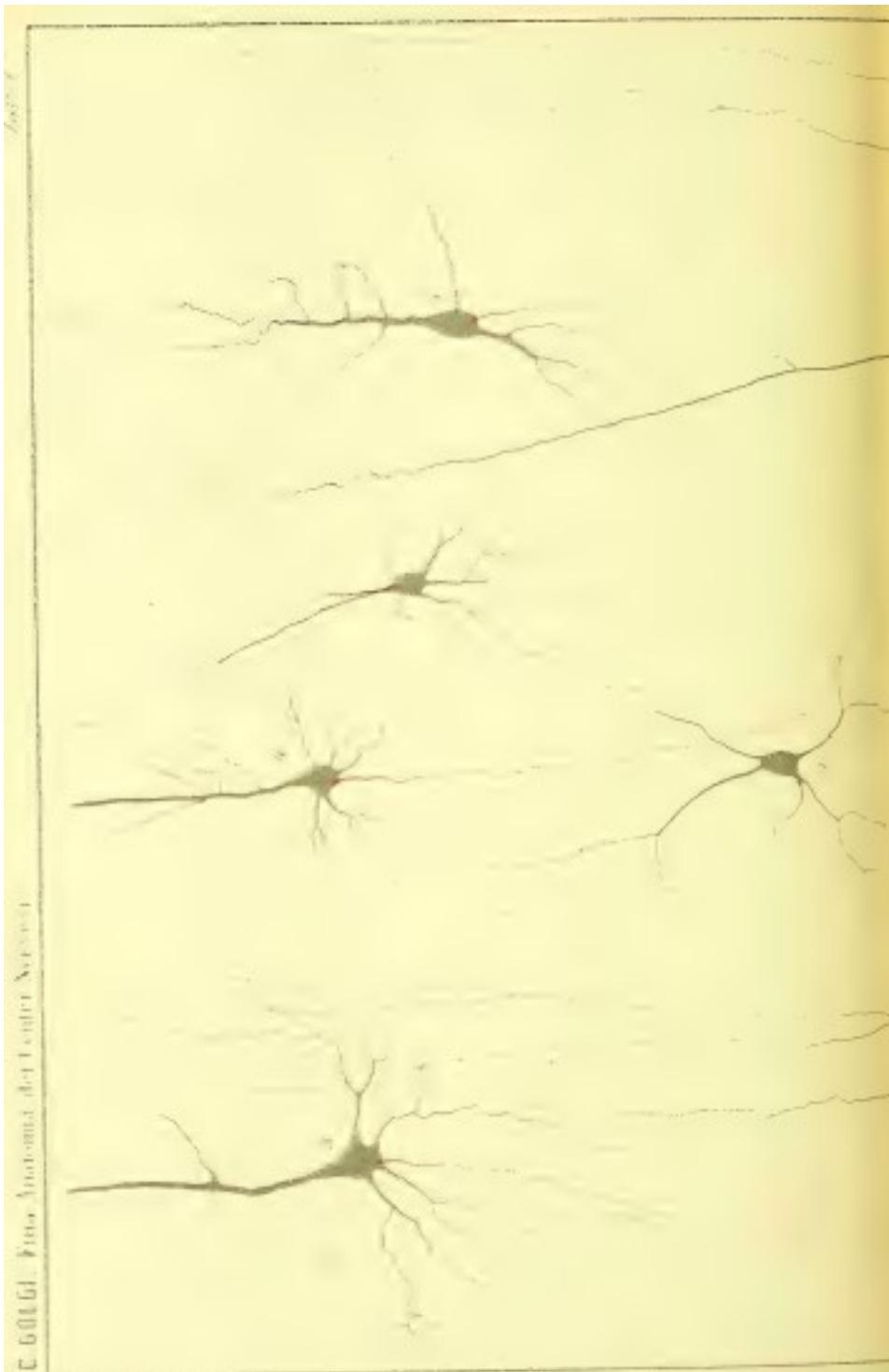
Year	Image	Laureate <sup>[A]</sup>	Nationality <sup>[B]</sup>	Rationale <sup>[C]</sup>
1901		Emil von Behring (1854–1917)	 Germany	"for his work on <a href="#">serum therapy</a> , especially its application against <a href="#">diphtheria</a> , by which he has opened a new road in the domain of medical science and thereby placed in the hands of the physician a victorious weapon against illness and deaths"
1902		Sir Ronald Ross (1857–1932)	 United Kingdom	"for his work on <a href="#">malaria</a> , by which he has shown how it enters the organism and thereby has laid the foundation for successful research on this disease and methods of combating it"
1903		Niels Ryberg Finsen (1860–1904)	 Faroe Islands	"[for] his contribution to the treatment of diseases, especially <a href="#">lupus vulgaris</a> , with concentrated light radiation, whereby he has opened a new avenue for medical science"
1904		Ivan Pavlov (1849–1936)	 Russia	"in recognition of his work on the physiology of <a href="#">digestion</a> , through which knowledge on vital aspects of the subject has been transformed and enlarged"
1905		Robert Koch (1843–1910)	 Germany	"for his investigations and discoveries in relation to <a href="#">tuberculosis</a> "

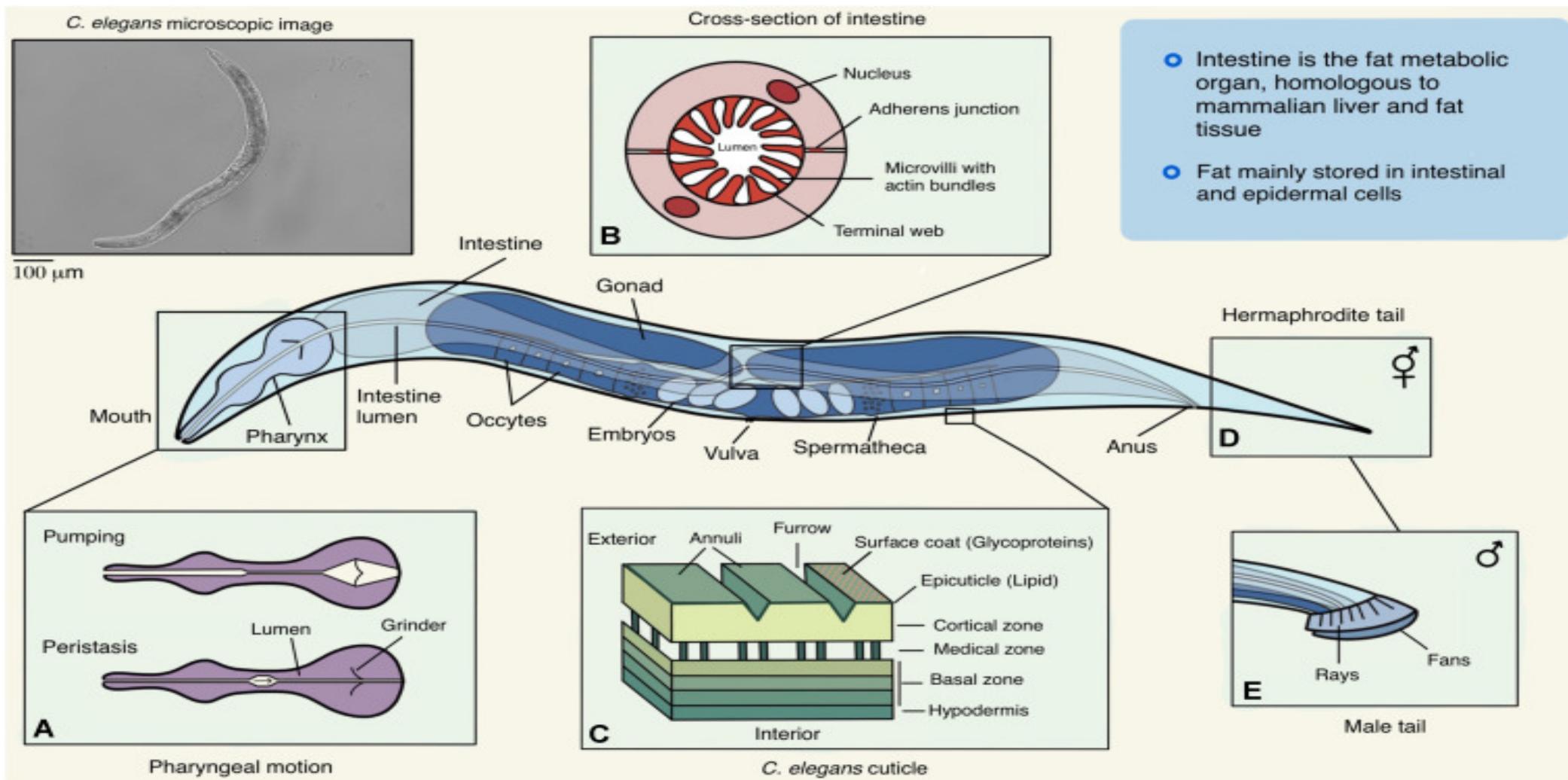
1906		Camillo Golgi (1843–1926)	 Italy	'in recognition of their work on the structure of the <a href="#">nervous system</a> '
		Santiago Ramón y Cajal (1852–1934)	 Spain	

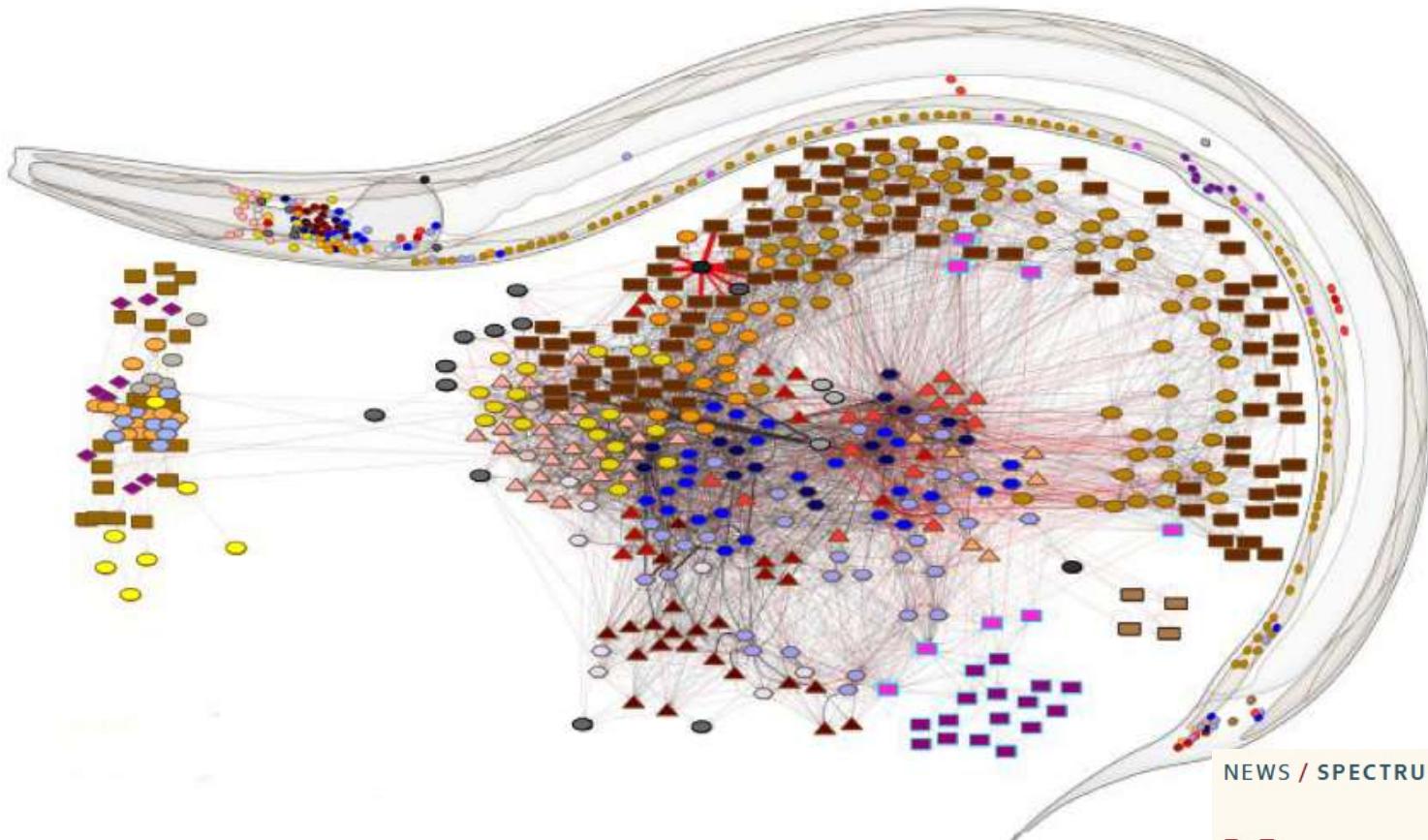
SULLA FINA ANATOMIA  
DEGLI  
**ORGANI CENTRALI**  
DEL  
**SISTEMA NERVOSO**

**STUDI**  
DI  
**CAMILLO GOLGI**  
Professore di Patologia generale e Istologia nell'Università di Pavia  
(CON 24 TAVOLE)

**ULRICO HOEPLI**  
EDITORE-LIBRAIO  
NAPOLI                    MILANO                    PISA  
—  
1886.







NEWS / SPECTRUM

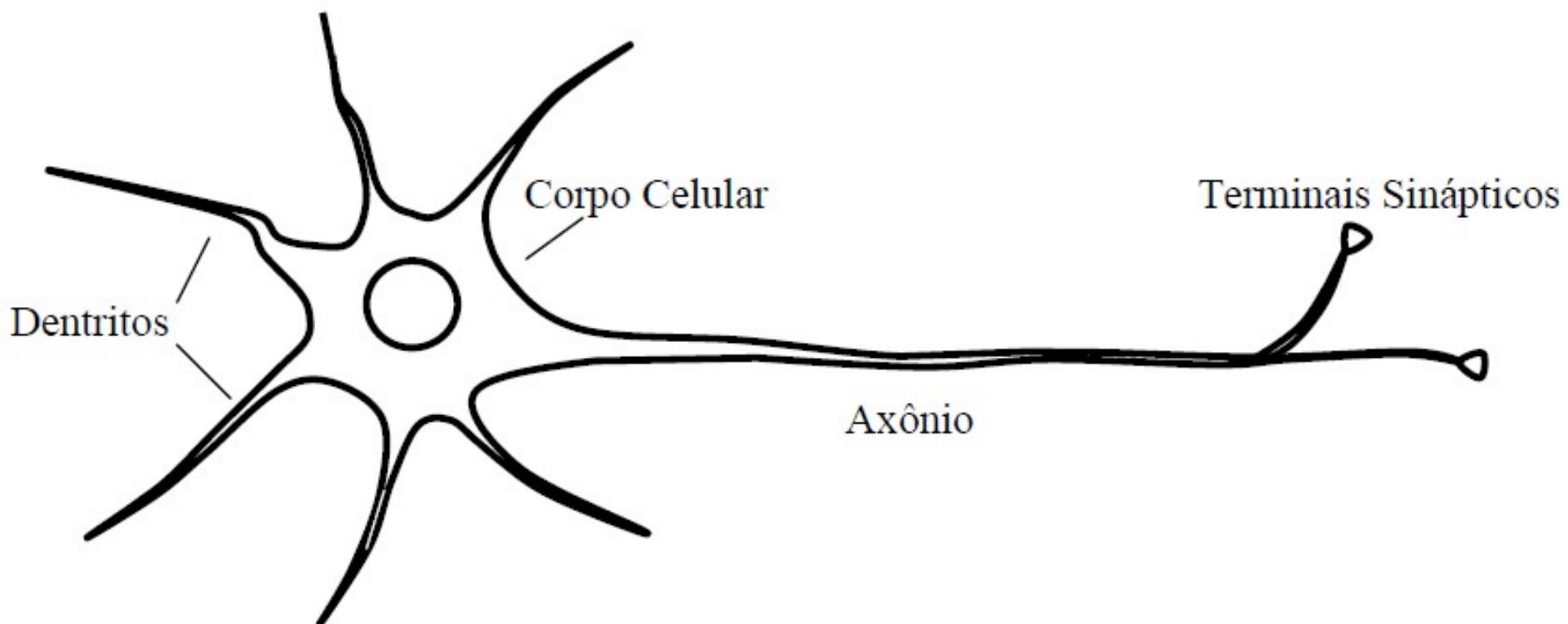
## New maps of neuronal connections reveal roundworms' wiring

Two new maps show the entire nervous system of the adult roundworm *Caenorhabditis elegans*.

BY POLINA POROTSKAYA  
30 AUGUST 2019 | 2 MIN READ

## Arquitetura do Perceptron

Para construir o algoritmo de treinamento do perceptron iniciamos com a elaboração de uma representação simplificada das principais partes de um neurônio. Ou seja, temos que os sinais de entrada do neurônio são obtidos a partir dos dendritos, esses sinais são processados no corpo celular (também chamado de soma), o resultado do processamento é conduzido através do axônio até os terminais sinápticos que representam as saídas que transmitem a informação. A Figura-1 mostra as partes principais de um neurônio natural.



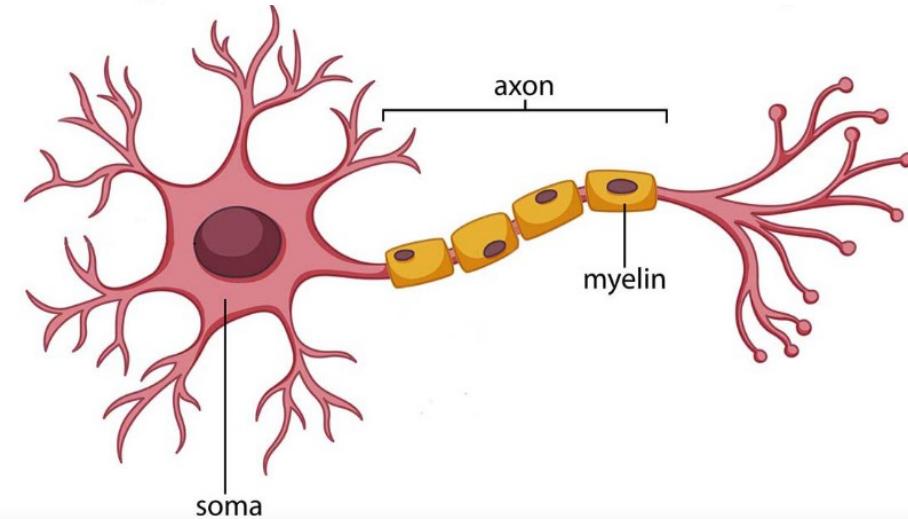
**Figura-1** Principais Componentes de um Neurônio Natural.

No corpo celular está contido o núcleo onde acontece a síntese de praticamente toda proteína do neurônio, o axônio é a parte especializada na condução dos sinais elétricos, em particular de um impulso elétrico denominado potencial de ação que chega a se propagar a velocidades acima de 100 metros por segundo. Os dendritos estendem-se para fora do corpo celular e recebem os sinais dos terminais dos axônios de outros neurônios, esses sinais são transmitidos para o interior do neurônio no corpo celular. O cérebro humano tem aproximadamente 1500 gramas, possui em torno de 86 bilhões de neurônios, mais de 10 trilhões de sinapses e estima-se que realize entre  $10^{13}$  a  $10^{16}$  operações lógicas por segundo.

Um modelo de neurônio artificial pode ser criado a partir da organização do neurônio descrito na Figura–1. Como os neurônios realizam várias conexões, esse modelo ou representação do neurônio é chamado de arquitetura da rede neural, sendo que o perceptron possui a arquitetura mais simples. A Figura–2 mostra a representação do perceptron, basicamente consiste de um único neurônio possuindo um conjunto de pesos ajustáveis e cuja tarefa principal é classificar os dados em duas classes. Um neurônio artificial também pode ser caracterizado como possuindo quatro elementos principais: entradas, pesos, função de ativação e saída. Na Figura–2 é apresentada a

# What is the brain?

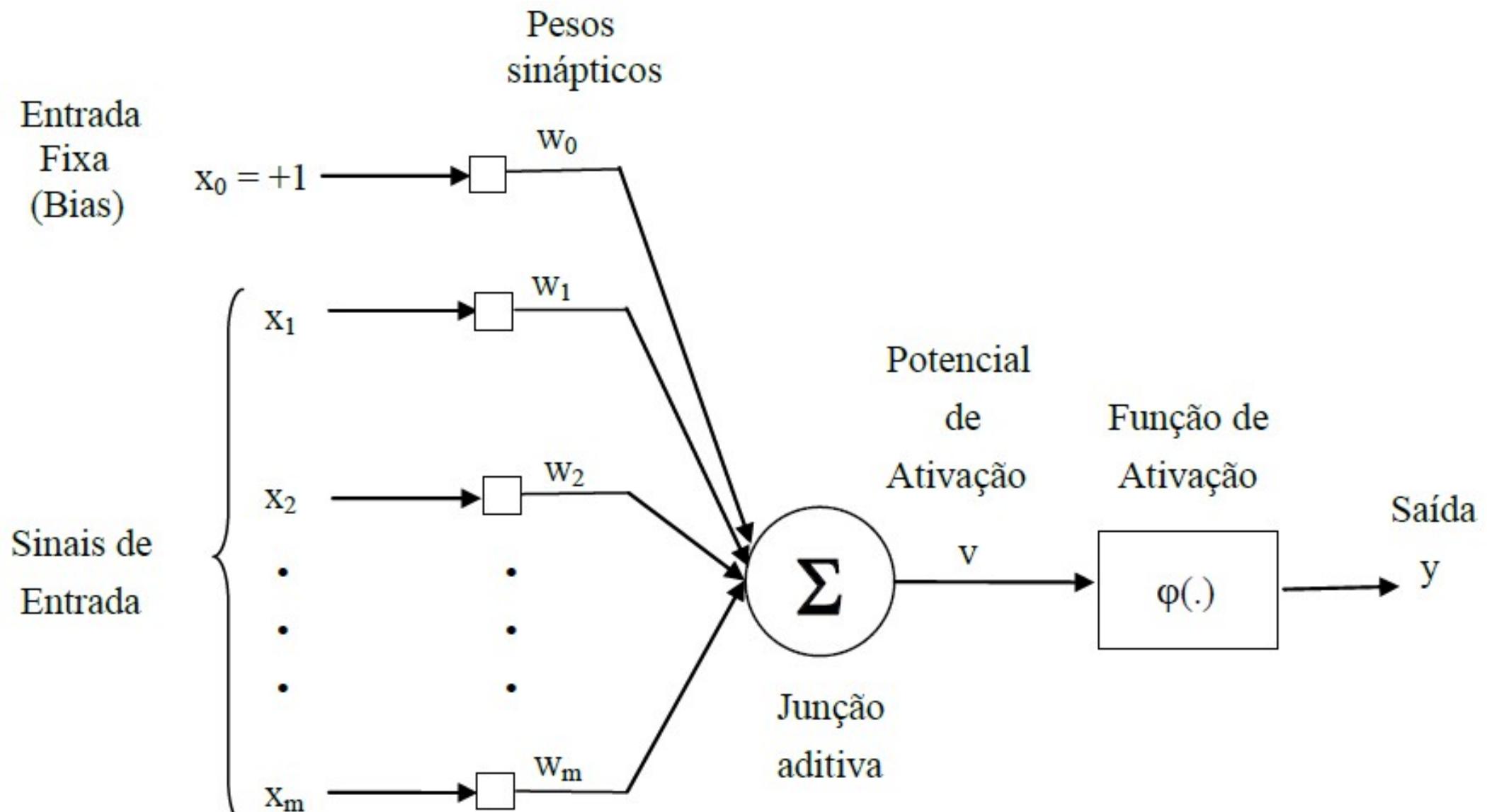
The brain is a complex organ that controls thought, memory, emotion, touch, motor skills, vision, breathing, temperature, hunger and every process that regulates our body. Together, the brain and spinal cord that extends from it make up the central nervous system, or CNS.



<https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain>

Um modelo de neurônio artificial pode ser criado a partir da organização do neurônio descrito na Figura–1. Como os neurônios realizam várias conexões, esse modelo ou representação do neurônio é chamado de arquitetura da rede neural, sendo que o perceptron possui a arquitetura mais simples. A Figura–2 mostra a representação do perceptron, basicamente consiste de um único neurônio possuindo um conjunto de pesos ajustáveis e cuja tarefa principal é classificar os dados em duas classes. Um neurônio artificial também pode ser caracterizado como possuindo quatro elementos principais: entradas, pesos, função de ativação e saída.

Na Figura-2 é apresentada a configuração detalhada do perceptron onde temos: (i) um conjunto de conexões, ou sinapses, nesse caso temos um neurônio que se conecta com  $m$  outros neurônios recebendo as entradas  $x_1, x_2, \dots, x_m$ , (ii) uma entrada fixa, denominada  $x_0$ , que desempenha papel fundamental no ajuste do modelo e para o sucesso do processo de aprendizagem, na Figura-2 o bias é representado por  $w_0$  e seu valor é multiplicado por uma entrada fixa igual a +1, (iii) um conjunto de pesos, representados por  $w_0, w_1, \dots, w_m$ , onde cada peso pondera um sinal de entrada do neurônio, cada peso  $w_1, w_2, \dots, w_m$  representa a força da sinapse realizada, com  $w_0$  sendo a ponderação do bias, (iv) um somador ou integrador que soma os sinais de entrada depois de terem sido multiplicados por seus respectivos pesos, a soma dada por  $v = \sum_{i=0}^m w_i x_i$  ou  $v = \sum_{i=1}^m w_i x_i + w_0$  recebe o nome de campo local induzido ou potencial de ativação do neurônio, (v) uma função de ativação, ou função de transferência, que limita o valor de saída do neurônio, tipicamente a saída é limitada ao intervalo  $[0, 1]$  ou alternativamente  $[-1, 1]$ , dependendo do tipo de função de ativação que seja utilizada; (vi) e  $y$  é o sinal de saída do neurônio.

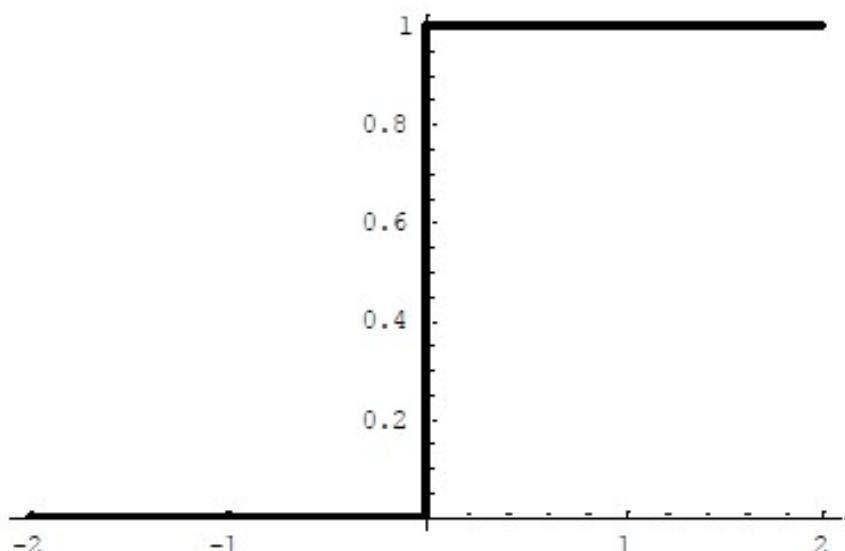


Figura–2 Modelo de um neurônio artificial.

O perceptron mostrado na Figura–2 recebe o nome de neurônio de McCulloch-Pitts. Nesse neurônio é calculada a soma ponderada  $v$  de todas as  $m$  entradas. Em seguida, é aplicada a função de ativação que pode ser do tipo degrau de Heaviside, de acordo com a equação (1), o resultado  $\varphi(v)$  gera uma saída que é então comparada a um valor desejado produzindo um erro.

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases} \quad (1)$$

A função de ativação,  $\varphi(\cdot)$ , define a saída de um neurônio. No caso do treinamento do neurônio de McCulloch-Pitts além da função degrau de Heaviside descrita pela equação (1) e com gráfico mostrado na Figura–3, também pode ser utilizada a função sinal que retorna +1 se  $v \geq 0$  e -1 se  $v < 0$ .



Figura–3 Função degrau de Heaviside



Em 1958, Frank Rosenblatt publicou o trabalho *The perceptron: A probabilistic model for information storage and organization in the brain* no periódico Psychological Review apresentando o modelo perceptron e a regra de aprendizagem associada. O trabalho inicia com uma consideração seguida por três questões:

*Se quisermos finalmente entender a capacidade dos organismos superiores para o reconhecimento, generalização, recordação e pensamento perceptivo, devemos primeiro ter respostas a três questões fundamentais: 1. Como a informação sobre o mundo físico é sentida, ou detectada, pelo sistema biológico? 2. Em que forma a informação é armazenada, ou lembrada? 3. Como a informação contida no armazenamento, ou na memória, influencia o reconhecimento e o comportamento?*

Logo a seguir Rosenblatt escreve:

*A teoria a ser apresentada aqui assume a posição empirista, ou "conexionista" com relação a essas questões. A teoria foi desenvolvida para um sistema nervoso hipotético, ou máquina, chamado perceptron. O perceptron é projetado para ilustrar algumas das propriedades fundamentais dos sistemas inteligentes em geral, sem se tornar muito profundamente enredado nas condições especiais, e frequentemente desconhecidas, que mantêm-se para organismos biológicos particulares. A analogia entre o perceptron e os sistemas biológicos deve ser prontamente evidente para o leitor.*

## Treinamento do Perceptron

Um perceptron pode ser utilizado para classificar padrões de entrada em duas classes. Sendo que a principal tarefa a ser realizada no treinamento do perceptron em seu processo de aprendizagem constitui-se em ajustar o vetor de pesos  $\mathbf{w}$  utilizando uma regra de aprendizagem adaptativa, denominada algoritmo de convergência do perceptron. Igualando o potencial de ativação do neurônio a zero tem-se a equação (2) que gera uma partição que separa os valores de entrada em duas regiões, ou classes.

$$v = \sum_{i=1}^m w_i x_i + w_0 = 0 . \quad (2)$$

A equação (2) algumas vezes também é escrita como

$$v = \sum_{i=1}^m w_i x_i + b = 0 ,$$

onde  $b$  representa o bias.

De tal forma que, se existem duas classes  $C_1$  e  $C_2$  que são linearmente separáveis (ou seja, elas residem em lados opostos de uma linha reta ou, em geral, um hiperplano), então existe um vetor de pesos  $\mathbf{w}$  com as seguintes propriedades

$$\sum_{i=1}^m w_i x_i + w_0 \geq 0 , \quad \text{para todo vetor de entrada } x \text{ pertencente à classe } C_1$$

$$\sum_{i=1}^m w_i x_i + w_0 < 0 , \quad \text{para todo vetor de entrada } x \text{ pertencente à classe } C_2 .$$



No processo de treinamento do perceptron é necessário estipular qual o valor de saída desejado. Este valor será comparado com a saída gerada pelo perceptron e a diferença entre os dois será um erro denotado por  $e$ . O valor desejado será denotado por  $d$ . O processo de aprendizagem será interrompido quando o valor desejado for alcançado ou o valor de saída do neurônio estiver tão próximo do desejado quanto o analista aceite. Claramente o erro é uma função do vetor de pesos  $\mathbf{w}$ , sendo que o sinal de erro na saída do neurônio na iteração  $k$  é definido por

$$e(k) = d(k) - y(k). \quad (3)$$

A construção do algoritmo de treinamento do perceptron utiliza o método de minimização do erro quadrado, para isso temos que o erro total é obtido somando-se todos os erros do neurônio, assim a função de erro total, escrita como  $\varepsilon(\mathbf{w})$ , é dada por

$$\varepsilon(\mathbf{w}) = \frac{1}{2} \sum_k e^2(k), \quad (4)$$

Como o erro é uma função do peso, utiliza-se um processo iterativo para alcançar o valor desejado. Essa abordagem, ou esse método de procura faz uso do algoritmo de descida do gradiente. Ou seja, a variação nos valores dos pesos sinápticos é obtida considerando o sentido contrário do vetor gradiente de erro multiplicado por um passo previamente definido pelo analista. Mais formalmente essa ideia pode ser escrita como

$$\Delta \mathbf{w} = -\eta \nabla \varepsilon(\mathbf{w}). \quad (5)$$

A equação (5) pode ser reescrita considerando cada iteração do processo de aproximação da seguinte forma

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \frac{\partial \varepsilon(\mathbf{w}(k))}{\partial \mathbf{w}(k)}. \quad (6)$$



A equação (6) é conhecida como regra delta ou regra de Widrow-Hoff. Em 1960, Bernard Widrow e Marcian Edward Ted Hoff Jr. no trabalho *Adaptive switching circuits* introduziram um novo algoritmo de aprendizado e o utilizaram para treinar redes neurais lineares adaptativas, que eram semelhantes em estrutura e capacidade ao perceptron de Rosenblatt. Uma vez que a saída antecipada é especificada, a utilização da regra delta é considerada treinamento supervisado. Nesse trabalho de 1960, Widrow e Hoff descrevem que padrões linearmente separáveis puros e versões ruidosas deles são prontamente classificados por um único neurônio. Padrões não linearmente separáveis puros e seus equivalentes ruidosos também podem ser separados por um único neurônio, mas o desempenho absoluto pode ser melhorado e a generalidade do esquema de classificação pode ser bastante aumentada usando mais de um neurônio. Sendo que um ruído pode ser definido como um sinal indesejado que interfere com a comunicação ou medição de outro sinal.

Na equação (6) temos que a derivada de  $\varepsilon(\mathbf{w})$  com relação ao vetor de pesos  $\mathbf{w}$  pode ser obtida considerando as relações construídas anteriormente, assim temos

$$\frac{\partial \varepsilon(\mathbf{w}(k))}{\partial \mathbf{w}(k)} = \frac{\partial \varepsilon(\mathbf{w}(k))}{\partial y(k)} \frac{\partial y(k)}{\partial \mathbf{w}(k)},$$

onde

$$\begin{aligned} \frac{\partial \varepsilon(\mathbf{w}(k))}{\partial y(k)} &= \frac{\partial}{\partial y(k)} \left[ \frac{1}{2} \sum_k e^2(k) \right], \\ &= \frac{1}{2} \frac{\partial}{\partial y(k)} \sum_k [d(k) - y(k)]^2, \\ &= \frac{1}{2} \cdot 2 \cdot [d(k) - y(k)](-1), \\ &= -[d(k) - y(k)], \end{aligned}$$



e como no perceptron o neurônio é um separador linear, a saída na iteração  $k$  é igual ao campo local

induzido  $\sum_k w(k)x(k) + w_0(k)$  que em notação vetorial é dado por  $\mathbf{x}'(k)\mathbf{w}(k)$ , onde temos

$$\begin{aligned} \frac{\partial y(k)}{\partial \mathbf{w}(k)} &= \frac{\partial}{\partial \mathbf{w}(k)} [\mathbf{x}'(k)\mathbf{w}(k)], \\ &= \mathbf{x}(k). \end{aligned}$$

Portanto, na iteração  $k$ , o gradiente do erro  $\nabla \varepsilon(\mathbf{w})$  com relação ao vetor de pesos  $\mathbf{w}$  é obtido tomando-se o valor negativo do produto do erro na iteração  $k$  pelo vetor de valores de entrada nessa iteração. Aplicando esses resultados na equação (6) temos

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta(-[d(k) - y(k)])\mathbf{x}(k),$$

ou

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \mathbf{x}(k) e(k). \quad (7)$$

O tamanho do passo  $\eta$  no processo de aprendizagem do perceptron recebe o nome de **taxa de aprendizagem** e é importante no processo de convergência. Valores típicos para a taxa de aprendizagem situam-se no intervalo  $0 < \eta \leq 1$ . O número de vezes que todos os vetores de treinamento são utilizados para atualizar os pesos uma vez é denominado de época. O algoritmo de treinamento da rede neural perceptron é descrito a seguir.

### Algoritmo de aprendizagem do perceptron

1. Iniciar os pesos com valores aleatórios pequenos
2. Atribuir uma taxa de aprendizagem
3. Atribuir valor de erro total aceitável
4. Para cada dado de treinamento
  - Calcular o valor de saída da rede
  - Calcular o valor do erro
  - Calcular o gradiente
5. Para cada valor de entrada
  - Calcular o novo valor de peso
  - Atualizar os pesos
  - Calcular os erros
6. Calcular o erro total da rede para o conjunto de treinamento dado
7. Se o erro total da rede for maior do que o erro total aceitável, volte para o passo 4. Se o processo de aprendizagem tiver sido repetido por um número máximo de épocas sem ter convergido para um erro total aceitável, então finalize.



## EXEMPLO

Aplique o algoritmo de aprendizagem do perceptron para o caso do operador lógico E.

As entradas são dadas por  $x_0$ ,  $x_1$  e  $x_2$ , com pesos  $w_0$  (bias),  $w_1$  e  $w_2$ , respectivamente, sendo que  $y$  é a saída que deverá, de acordo com as entradas, ser igual ao valor desejado  $d$  mostrado na tabela

abaixo.

$x_0$	$x_1$	$x_2$	$x_1 \wedge x_2$	$d$
1	1	1	1	1
1	1	0	0	0
1	0	1	0	0
1	0	0	0	0

A rede neural para realizar essa tarefa de aprendizagem pode ser representada como mostrado na Figura-6.

Serão atribuídos os seguintes valores iniciais para os pesos e para a taxa de aprendizagem:

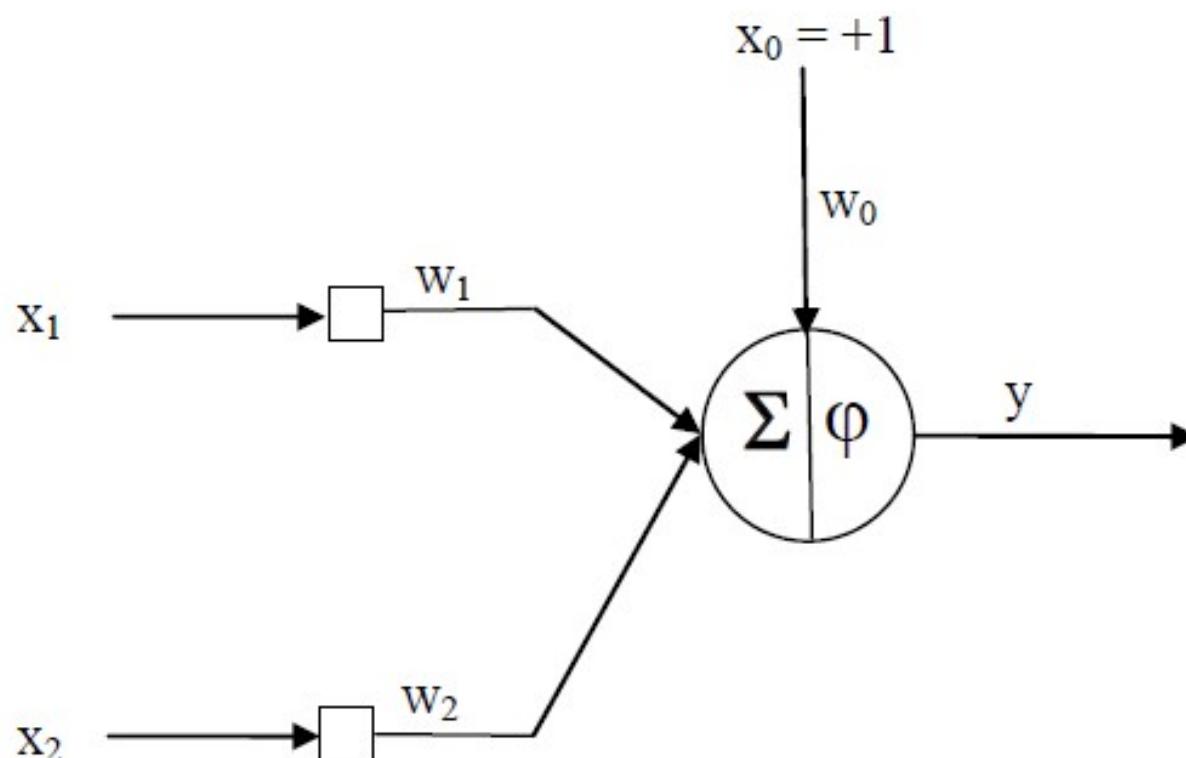
$$w_0 = 0, w_1 = 0, w_2 = 0 \text{ e } \eta = 0,5 .$$

Serão atribuídos os seguintes valores iniciais para os pesos e para a taxa de aprendizagem:

$$w_0 = 0, w_1 = 0, w_2 = 0 \text{ e } \eta = 0,5.$$

Será utilizada a função de ativação (ou função de transferência)

$$\varphi(v) = \begin{cases} 1, & \text{se } v > 0 \\ 0, & \text{se } v \leq 0 \end{cases}$$



**Figura–6** Rede neural perceptron para aprendizagem do operador lógico E.



## Primeiro Ciclo de Treinamento

Nesse caso, vamos parar o processo de aprendizagem quando o erro for zero, ou seja, quando todos os valores de entrada gerarem a saída adequada. Dessa forma, o primeiro conjunto de entrada é constituído pelos valores  $x_0$ ,  $x_1$  e  $x_2$ , respectivamente iguais a 1, 1 e 1, o que gera a saída

$$\begin{aligned}y &= \varphi(x_0w_0 + x_1w_1 + x_2w_2), \\&= \varphi(1.0 + 1.0 + 1.0), \\&= \varphi(0), \\&= 0,\end{aligned}$$

onde temos que a saída da rede neural é diferente do valor desejado,  $y \neq d$ . A saída é diferente do valor desejado e o erro é diferente de zero, o que implica que os pesos deverão ser alterados. Como a saída é diferente do valor desejado, deve-se proceder a atualização dos pesos sinápticos aplicando o resultado da equação (6), de onde são obtidos os novos valores dos pesos. No caso do peso  $w_0$ , a atualização é realizada fazendo

$$w_0(1) = w_0(0) + \eta(d(0) - y(0))x_0(0). \quad (8)$$

Apesar de perder um pouco de rigor em termos de notação, vamos reescrever a equação (8) como

$$w_0 = w_0 + \eta(d - y)x_0,$$

que resulta em

$$w_0 = 0 + 0,5(1 - 0)(1) = 0,5.$$

Para os pesos  $w_1$  e  $w_2$ , temos

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(1 - 0)(1) = 0,5,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0 + 0,5(1 - 0)(1) = 0,5,$$

Esse novo conjunto de pesos onde  $w_0 = 0,5$ ,  $w_1 = 0,5$  e  $w_2 = 0,5$  é utilizado para o próximo conjunto de entrada.

A saída é diferente do valor desejado, então há necessidade de alteração dos pesos.  
Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = 0,5 + 0,5(0 - 1)(1) = 0,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0,5 + 0,5(0 - 1)(1) = 0,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(0 - 1)(0) = 0,5.$$

Terceira entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 1$ , produz

$$y = \varphi(1.0 + 0.0 + 1.(0,5)) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = 0 + 0,5(0 - 1)(1) = -0,5,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(0 - 1)(0) = 0,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(0 - 1)(1) = 0.$$

Quarta entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 0$ , produz

$$y = \varphi(1.(-0,5) + 0.0 + 0.0) = \varphi(-0,5) = 0 \rightarrow y = d.$$

Apresentados todos os valores de entrada e atualizados os pesos, têm-se, então, que transcorreu uma época. Porém, apesar da quarta entrada junto com os pesos calculados produzir um valor desejado, é necessário verificar se esse conjunto de pesos é adequado para os dados das outras entradas, o que implica em iniciar um segundo ciclo de treinamento.



## Segundo Ciclo de Treinamento

Primeira entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 1$ , produz

$$y = \varphi(1.(-0,5) + 1.0 + 1.0) = \varphi(-0,5) = 0 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -0,5 + 0,5(1 - 0)(1) = 0,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(1 - 0)(1) = 0,5,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0 + 0,5(1 - 0)(1) = 0,5.$$

Segunda entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 0$ , produz

$$y = \varphi(1.0 + 1.(0,5) + 0.(0,5)) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = 0 + 0,5(0 - 1)(1) = -0,5,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0,5 + 0,5(0 - 1)(1) = 0,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(0 - 1)(0) = 0,5.$$

Terceira entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 1$ , produz

$$y = \varphi(1.(-0,5) + 0.0 + 1.(0,5)) = \varphi(0) = 0 \rightarrow y = d.$$

Quarta entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 0$ , produz

$$y = \varphi(1.(-0,5) + 0.0 + 0.(0,5)) = \varphi(-0,5) = 0 \rightarrow y = d.$$

### Terceiro Ciclo de Treinamento

Primeira entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 1$ , produz

$$y = \varphi(1.(-0,5) + 1.0 + 1.(0,5)) = \varphi(0) = 0 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -0,5 + 0,5(1 - 0)(1) = 0,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(1 - 0)(1) = 0,5,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(1 - 0)(1) = 1.$$

Segunda entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 0$ , produz

$$y = \varphi(1.0 + 1.(0,5) + 0.1) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = 0 + 0,5(0 - 1)(1) = -0,5,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0,5 + 0,5(0 - 1)(1) = 0,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 1 + 0,5(0 - 1)(0) = 1.$$

Terceira entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 1$ , produz

$$y = \varphi(1.(-0,5) + 0.0 + 1.1) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -0,5 + 0,5(0 - 1)(1) = -1,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(0 - 1)(0) = 0,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 1 + 0,5(0 - 1)(1) = 0,5.$$

Quarta entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 0$ , produz

$$y = \varphi(1.(-1) + 0.0 + 0.(0,5)) = \varphi(-1) = 0 \rightarrow y = d$$

## Quarto Ciclo de Treinamento

Primeira entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 1$ , produz

$$y = \varphi(1 \cdot (-1) + 1 \cdot 0 + 1 \cdot (0,5)) = \varphi(-0,5) = 0 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -1 + 0,5(1 - 0)(1) = -0,5,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0 + 0,5(1 - 0)(1) = 0,5,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(1 - 0)(1) = 1.$$

Segunda entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 0$ , produz

$$y = \varphi(1 \cdot (-0,5) + 1 \cdot (0,5) + 0,1) = \varphi(0) = 0 \rightarrow y = d.$$

Terceira entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 1$ , produz

$$y = \varphi(1 \cdot (-0,5) + 0 \cdot (0,5) + 1 \cdot 1) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -0,5 + 0,5(0 - 1)(1) = -1,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0,5 + 0,5(0 - 1)(0) = 0,5,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 1 + 0,5(0 - 1)(1) = 0,5.$$

Quarta entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 0$ , produz

$$y = \varphi(1 \cdot (-1) + 0 \cdot (0,5) + 0 \cdot (0,5)) = \varphi(-1) = 0 \rightarrow y = d.$$

## Quinto Ciclo de Treinamento

Primeira entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 1$ , produz

$$y = \varphi(1 \cdot (-1) + 1 \cdot (0,5) + 1 \cdot (0,5)) = \varphi(0) = 0 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -1 + 0,5(1 - 0)(1) = -0,5,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 0,5 + 0,5(1 - 0)(1) = 1,$$

$$w_2 = w_2 + \eta(d - y)x_2 = 0,5 + 0,5(1 - 0)(1) = 1.$$

Segunda entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 0$ , produz

$$y = \varphi(1 \cdot (-0,5) + 1 \cdot 1 + 0,1) = \varphi(0,5) = 1 \rightarrow y \neq d.$$

Atualização dos pesos

$$w_0 = w_0 + \eta(d - y)x_0 = -0,5 + 0,5(0 - 1)(1) = -1,$$

$$w_1 = w_1 + \eta(d - y)x_1 = 1 + 0,5(0 - 1)(1) = 0,5$$

$$w_2 = w_2 + \eta(d - y)x_2 = 1 + 0,5(0 - 1)(0) = 1.$$

Terceira entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 1$ , produz

$$y = \varphi(1 \cdot (-1) + 0 \cdot (0,5) + 1 \cdot 1) = \varphi(0) = 0 \rightarrow y = d.$$

Quarta entrada,  $x_0 = 1$ ,  $x_1 = 0$  e  $x_2 = 0$ , produz

$$y = \varphi(1 \cdot (-1) + 0 \cdot (0,5) + 0 \cdot 1) = \varphi(-1) = 0 \rightarrow y = d.$$

## Sexto Ciclo de Treinamento

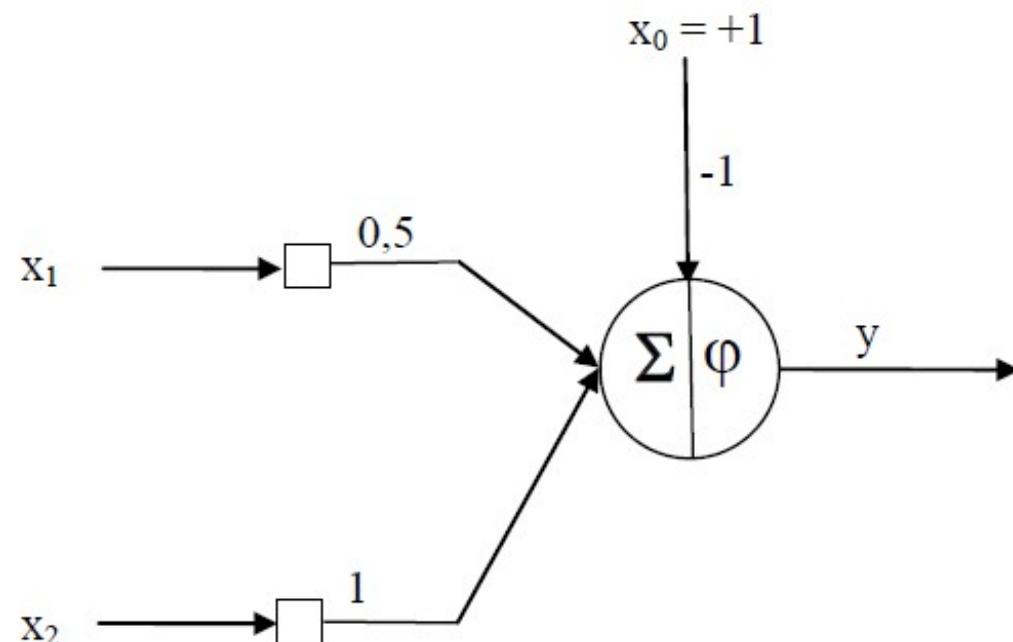
Primeira entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 1$ , produz

$$y = \varphi(1 \cdot (-1) + 1 \cdot (0,5) + 1 \cdot 1) = \varphi(0,5) = 1 \rightarrow y = d.$$

Segunda entrada,  $x_0 = 1$ ,  $x_1 = 1$  e  $x_2 = 0$ , produz

$$y = \varphi(1 \cdot (-1) + 1 \cdot (0,5) + 0 \cdot 1) = \varphi(-0,5) = 0 \rightarrow y = d.$$

Portanto, temos que o conjunto de pesos dados por  $w_0 = -1$ ,  $w_1 = 0,5$  e  $w_2 = 1$  é adequado para que a rede neural reproduza o operador lógico e. A configuração final da rede é mostrada na Figura-7.



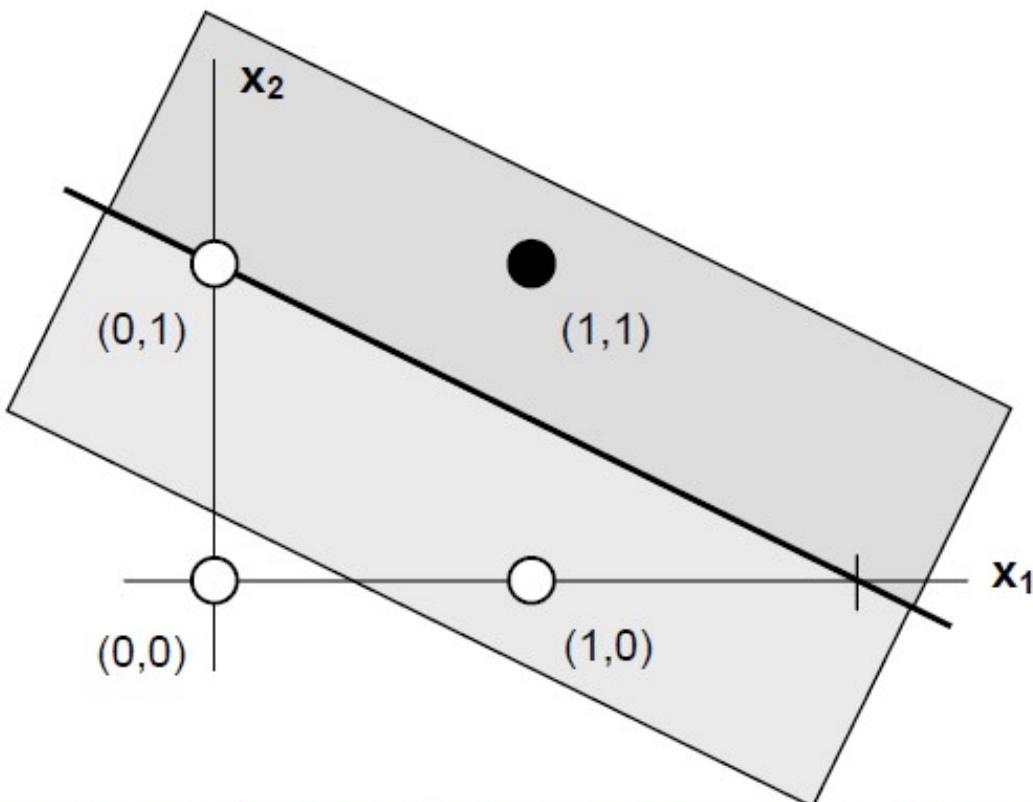
**Figura-7** Rede neural perceptron para aprendizagem do operador lógico E.

Para os valores de pesos obtidos e igualando o potencial de ativação do neurônio a zero tem-se que o perceptron separa os dados em duas regiões como mostrado na Figura–8.

Ou seja,

$$0,5x_1 + x_2 - 1 = 0 ,$$

$$x_2 = 1 - 0,5x_1 .$$



**Figura–8** Separação linear gerada pelo perceptron para o operador lógico E.

A tabela a seguir resume os valores encontrados durante o treinamento da rede. Na primeira coluna tem-se a quantidade de épocas que foram necessárias para a tarefa de aprendizagem, a seguir têm-se os valores dos sinais de entrada utilizados e depois as saídas que se desejam.

Época	Entradas			Valor Desejado $d$	Pesos Iniciais			Saída Atual $y$	Erro $e$	Pesos Finais		
	$x_0$	$x_1$	$x_2$		$w_0$	$w_1$	$w_2$			$w_0$	$w_1$	$w_2$
1	1	1	1	1	0	0	0	0	1	0,5	0,5	0,5
	1	1	0	0	0,5	0,5	0,5	1	-1	0	0	0,5
	1	0	1	0	0	0	0,5	1	-1	-0,5	0	0
	1	0	0	0	-0,5	0	0	0	0	-0,5	0	0
2	1	1	1	1	-0,5	0	0	0	1	0	0,5	0,5
	1	1	0	0	0	0,5	0,5	1	-1	-0,5	0	0,5
	1	0	1	0	-0,5	0	0,5	0	0	-0,5	0	0,5
	1	0	0	0	-0,5	0	0,5	0	0	-0,5	0	0,5
3	1	1	1	1	-0,5	0	0,5	0	1	0	0,5	1
	1	1	0	0	0	0,5	1	1	-1	-0,5	0	1
	1	0	1	0	-0,5	0	1	1	-1	-1	0	0,5
	1	0	0	0	-1	0	0,5	0	0	-1	0	0,5
4	1	1	1	1	-1	0	0,5	0	1	-0,5	0,5	1
	1	1	0	0	-0,5	0,5	1	0	0	-0,5	0,5	1
	1	0	1	0	-0,5	0,5	1	1	-1	-1	0,5	0,5
	1	0	0	0	-1	0,5	0,5	0	0	-1	0,5	0,5
5	1	1	1	1	-1	0,5	0,5	0	1	-0,5	1	1
	1	1	0	0	-0,5	1	1	1	-1	-1	0,5	1
	1	0	1	0	-1	0,5	1	0	0	-1	0,5	1
	1	0	0	0	-1	0,5	1	0	0	-1	0,5	1
6	1	1	1	1	-1	0,5	1	1	0	-1	0,5	1
	1	1	0	0	-1	0,5	1	0	0	-1	0,5	1
	1	0	1	0	-1	0,5	1	0	0	-1	0,5	1
	1	0	0	0	-1	0,5	1	0	0	-1	0,5	1



A rede neural inicia o treinamento a partir de um vetor de pesos previamente estabelecido que gera uma saída da rede que é comparada com o valor desejado, isso produz um erro. O objetivo é obter erro zero para cada padrão de entrada que é apresentado à rede. As atualizações dos pesos são efetuadas toda vez em que o erro for diferente de zero, nesse caso foram necessárias seis épocas para que o perceptron aprendesse a lógica do operador lógico E, os pesos finais são apresentados nas últimas três colunas da tabela.



```
✓ 0s
▶ import numpy as np

def perceptron_and_delta(learning_rate=0.5, max_epochs=100):
    """
    Treina um perceptron para o operador lógico AND utilizando a Regra Delta (Widrow-Hoff).

    Parâmetros:
        learning_rate (float): taxa de aprendizado ( $\eta$ ).
        max_epochs (int): número máximo de épocas de treinamento.

    Retorna:
        numpy.ndarray: vetor de pesos finais ( $w_0, w_1, w_2$ ), incluindo o bias.
    """
    # Conjunto de treinamento com bias: cada linha é [x0, x1, x2]
    X = np.array([
        [1, 0, 0], # x0=1, x1=0, x2=0
        [1, 0, 1], # x0=1, x1=0, x2=1
        [1, 1, 0], # x0=1, x1=1, x2=0
        [1, 1, 1] # x0=1, x1=1, x2=1
    ])
    # Alvos correspondentes para a porta AND
    targets = np.array([0, 0, 0, 1])

    # Inicialização dos pesos ( $w_0, w_1, w_2$ ) em zero
    weights = np.zeros(X.shape[1])
```

```
for epoch in range(max_epochs):
    errors = 0
    # Percorre todos os padrões de treinamento
    for x, t in zip(X, targets):
        # Calcula a soma ponderada (entrada líquida)
        net = np.dot(weights, x)
        # Determina a saída y pela função degrau
        y = step_function(net)
        # Calcula o erro
        error = t - y
        # Atualiza pesos somente se houve erro
        if error != 0:
            weights += learning_rate * error * x
            errors += 1
    # Se não houve erros em toda a época, o treinamento converge
    if errors == 0:
        break
return weights

if __name__ == "__main__":
    final_weights = perceptron_and_delta(learning_rate=0.5)
    print("Pesos finais para a porta AND:", final_weights)
```