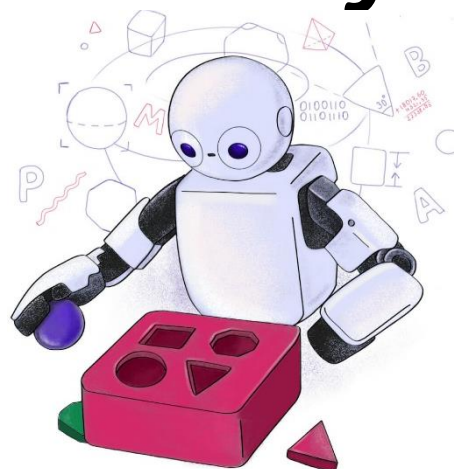


T320 - Introdução ao Aprendizado de Máquina II: *Redes Neurais Artificiais (Parte I)*



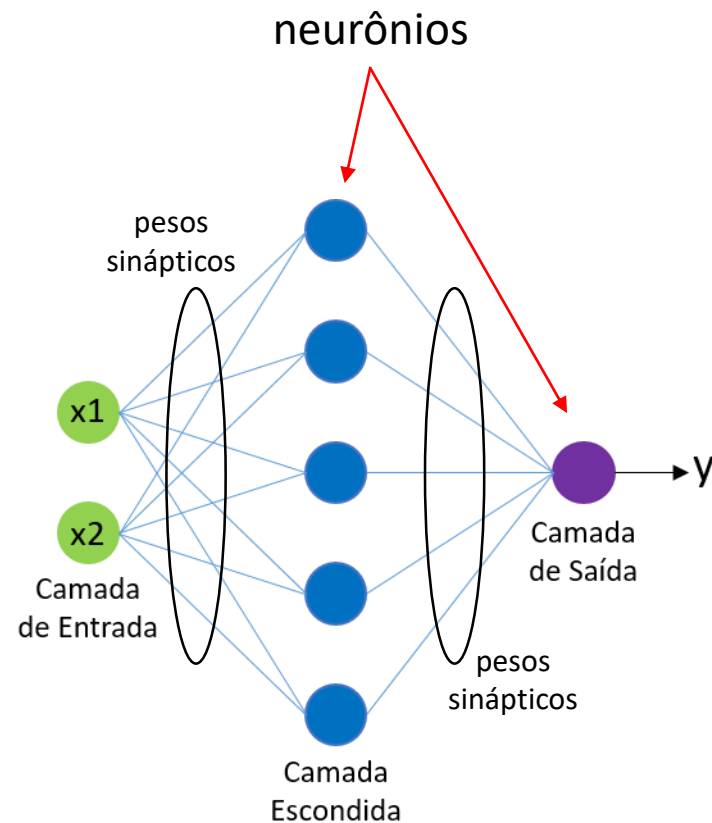
Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Introdução

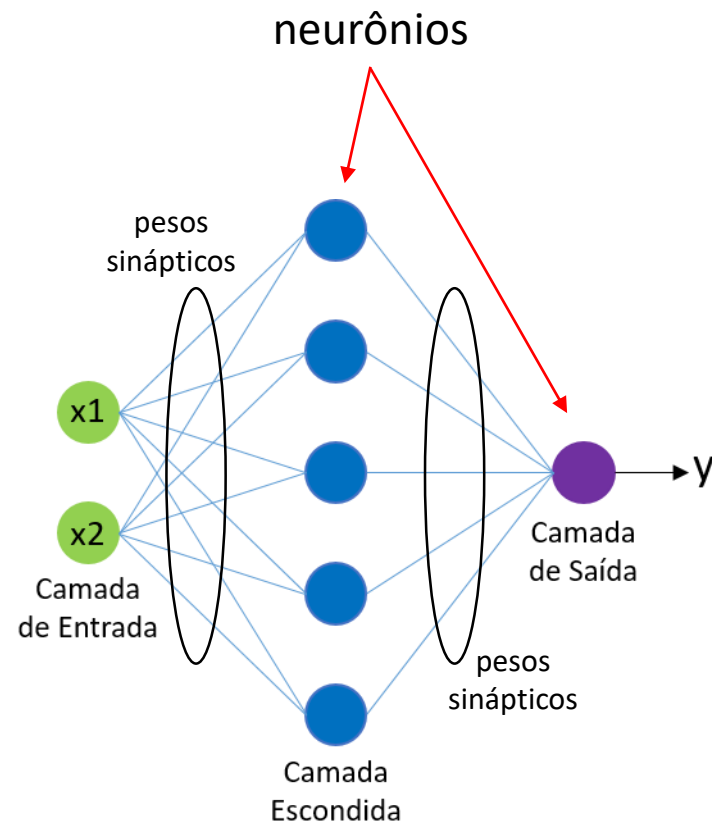
- A partir desta aula, entenderemos como as ideias que discutimos até agora serão úteis na construção de ***modelos matemáticos que aproximam a atividade de aprendizagem do cérebro.***
- Essas ideias que já discutimos, nos ajudarão a entender o funcionamento das ***redes neurais artificiais*** (RNAs).
- Redes neurais artificiais são uma das formas mais populares e efetivas para implementação de sistemas de aprendizado de máquina e mereceriam por si só uma disciplina em separado.
- Portanto, neste tópico, veremos uma breve visão geral sobre as RNAs.

Redes Neurais Artificiais



- **Redes neurais artificiais** são modelos computacionais *inspirados* pelo funcionamento do cérebro dos animais.
- Elas são capazes de realizar tarefas de aprendizado de máquina (e.g., regressão e classificação) com grande eficácia.

Redes Neurais Artificiais



- RNAs são geralmente apresentadas como ***sistemas de nós (unidades ou neurônios) interconectados***, que geram valores de saída, simulando o comportamento de ***redes neurais biológicas***.
- Esta primeira parte deste tópico, ***foca nos elementos básicos de construção de uma rede neural***, os ***nós*** ou ***neurônios***.

Algumas aplicações famosas

- RNAs são versáteis, poderosas e escalonáveis, tornando-as ideais para realizar tarefas altamente complexas de ***aprendizado de máquina***, como por exemplo:
 - Classificar bilhões de imagens (e.g., como o Google Images, Facebook, etc. fazem),
 - Serviços de reconhecimento de fala (e.g., a Siri da Apple, Alexa da Amazon e Google Assistant),
 - Recomendar vídeos que melhor se adequam ao comportamento de centenas de milhões de usuários todos os dias (e.g., YouTube, Netflix),
 - Dirigir um veículo com pouca ou nenhuma intervenção humana,
 - Responder perguntas (e.g., ChatGPT, DeepSeek).



Alexa



Google Assistant



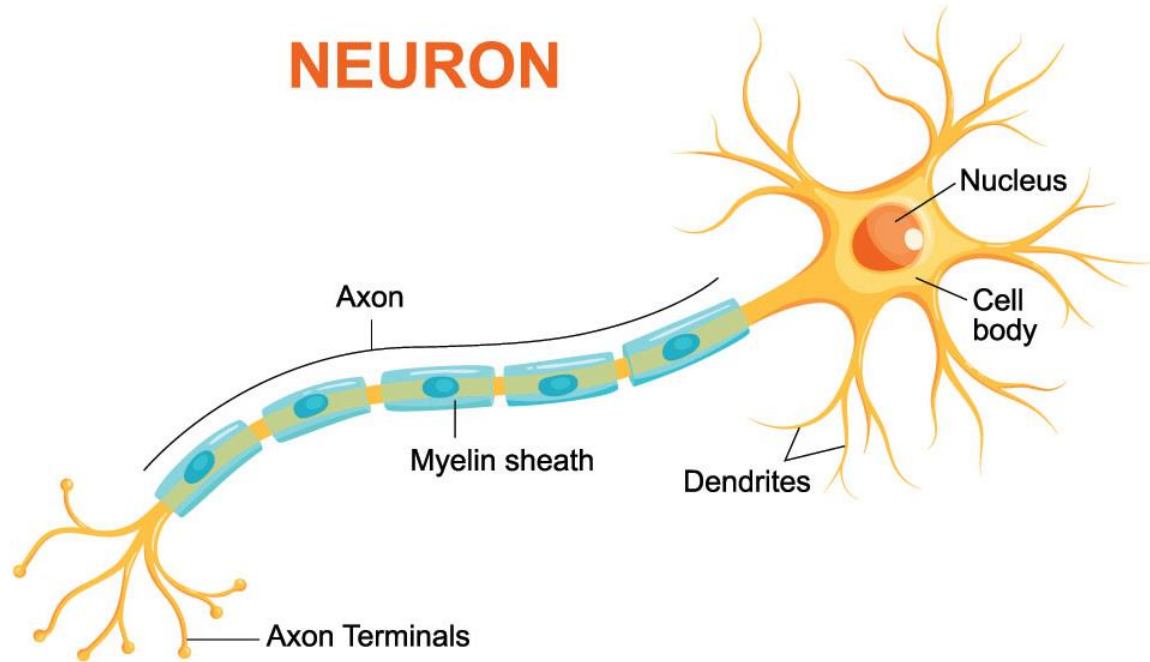
Siri



WAYMO



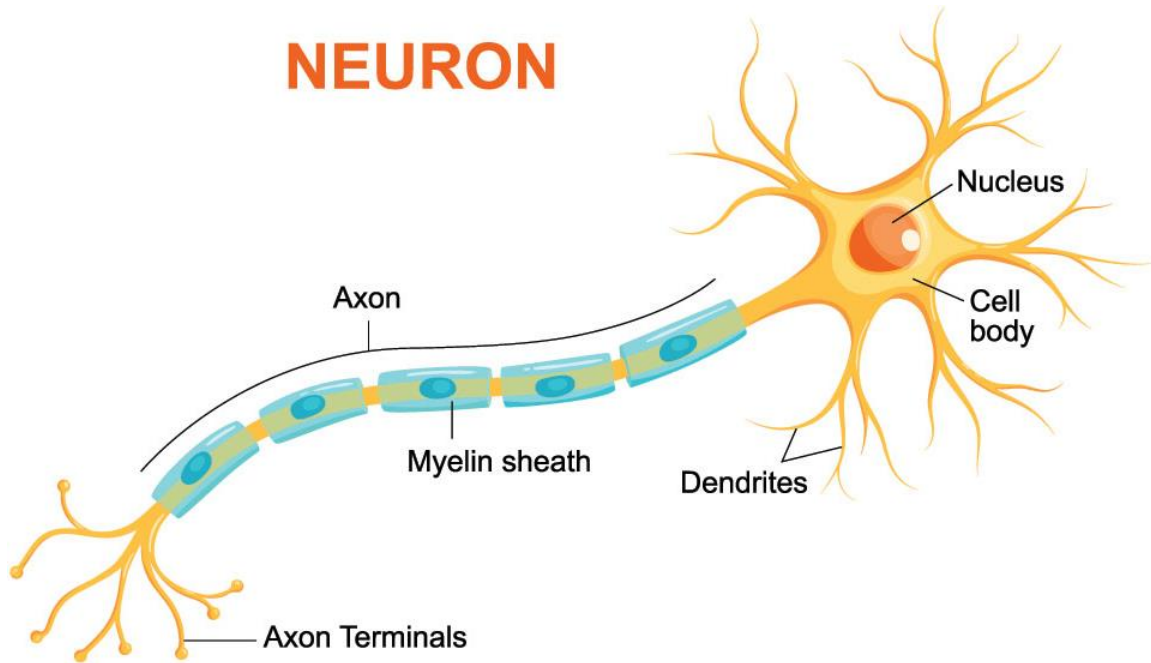
Um pouco de contexto



- Os ***neurônios*** são células ***eucariontes*** que possuem ***mecanismos eletroquímicos*** para transferência de informações entre eles.
- Eles apresentam três partes principais:
 - os ***dendritos***,
 - o ***axônio*** e
 - o ***corpo celular (soma)***.

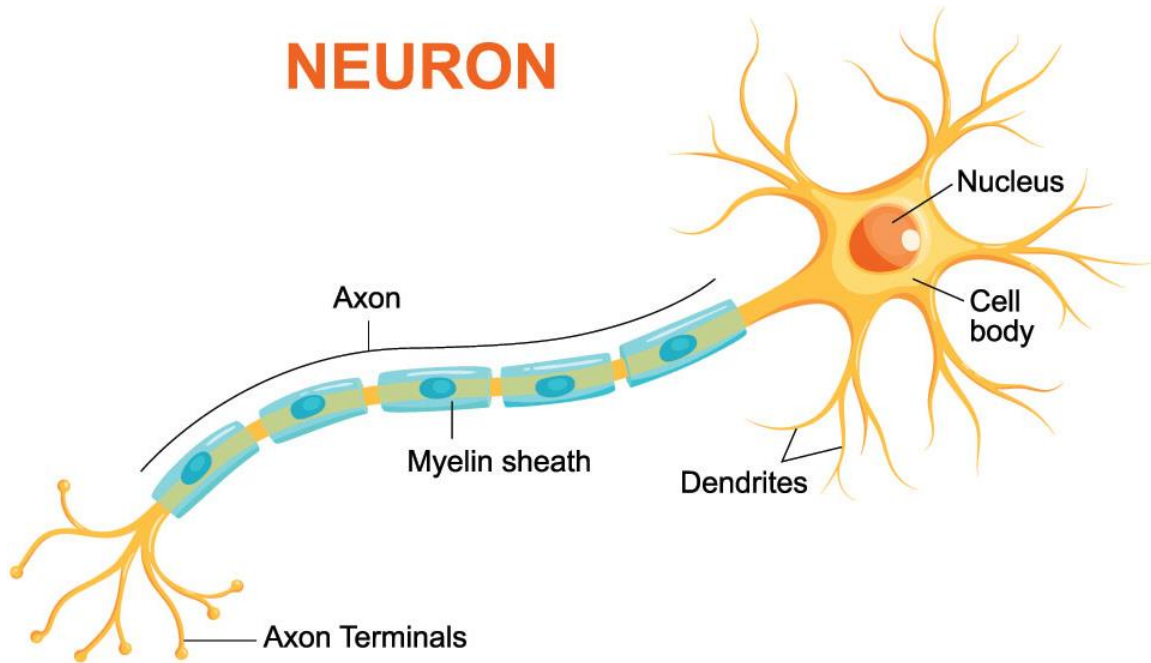
Um pouco de contexto

- Os **dendritos** são prolongamentos do neurônio que garantem a **recepção de estímulos** de outros neurônios, levando impulsos nervosos em direção ao **corpo celular**.



Um pouco de contexto

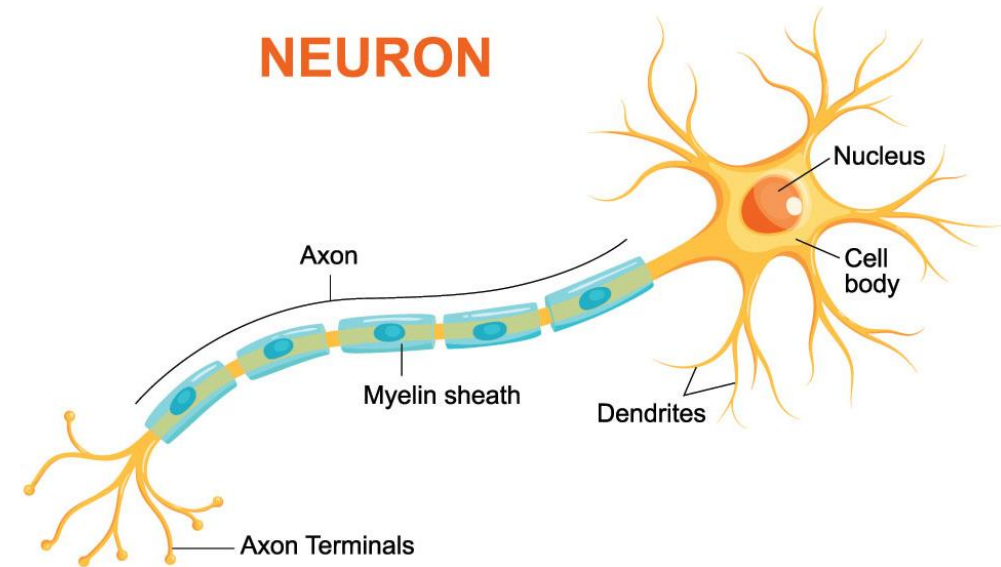
NEURON



- O ***axônio*** é um prolongamento que garante o envio de informação (estímulos) a outros neurônios através de seus ***terminais***.
- Cada neurônio possui apenas um axônio, o qual é, geralmente, mais longo que os dendritos.

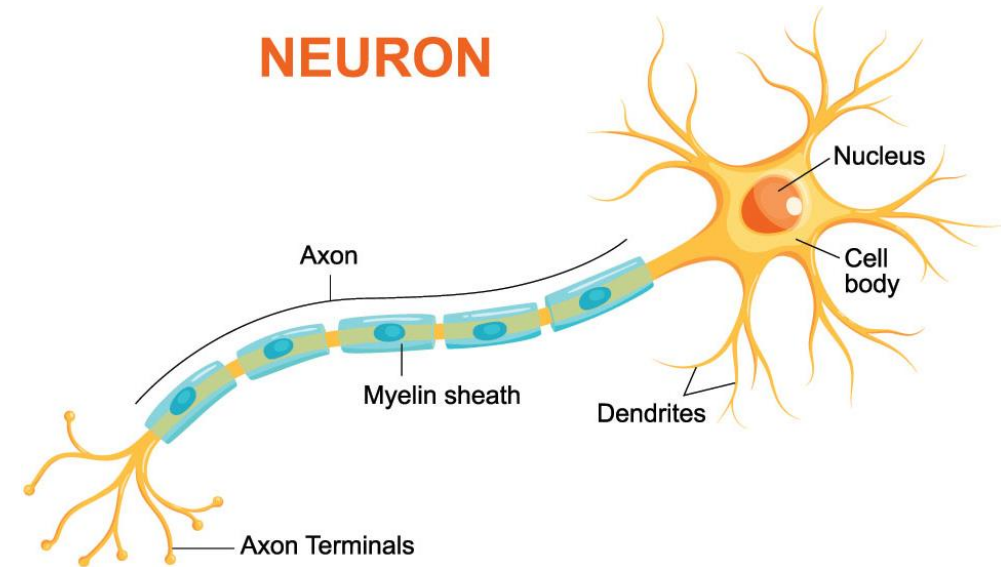
Um pouco de contexto

- O ***corpo celular*** (também conhecido como ***soma***) contém o núcleo do neurônio e é responsável por realizar a ***integração*** dos estímulos recebidos pelo neurônio através de seus dendritos.

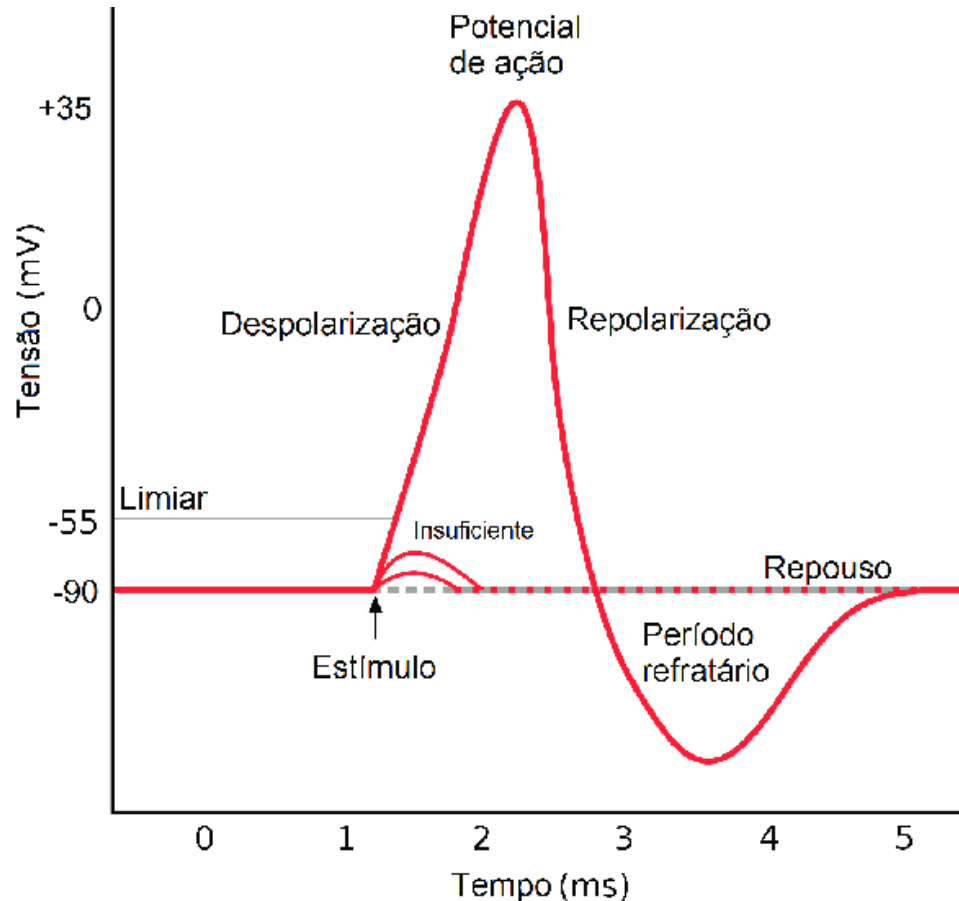


Um pouco de contexto

- Os pontos de contato entre os dendritos de um neurônio e os terminais do axônio de outro neurônio são chamados de ***sinapses***.
- A comunicações entre neurônios se dá através das ***sinapses***.
- Sinapses podem ser **químicas**, as mais comuns, ou **elétricas**, muito pouco comuns.
- A figura ao lado mostra uma sinapse química.

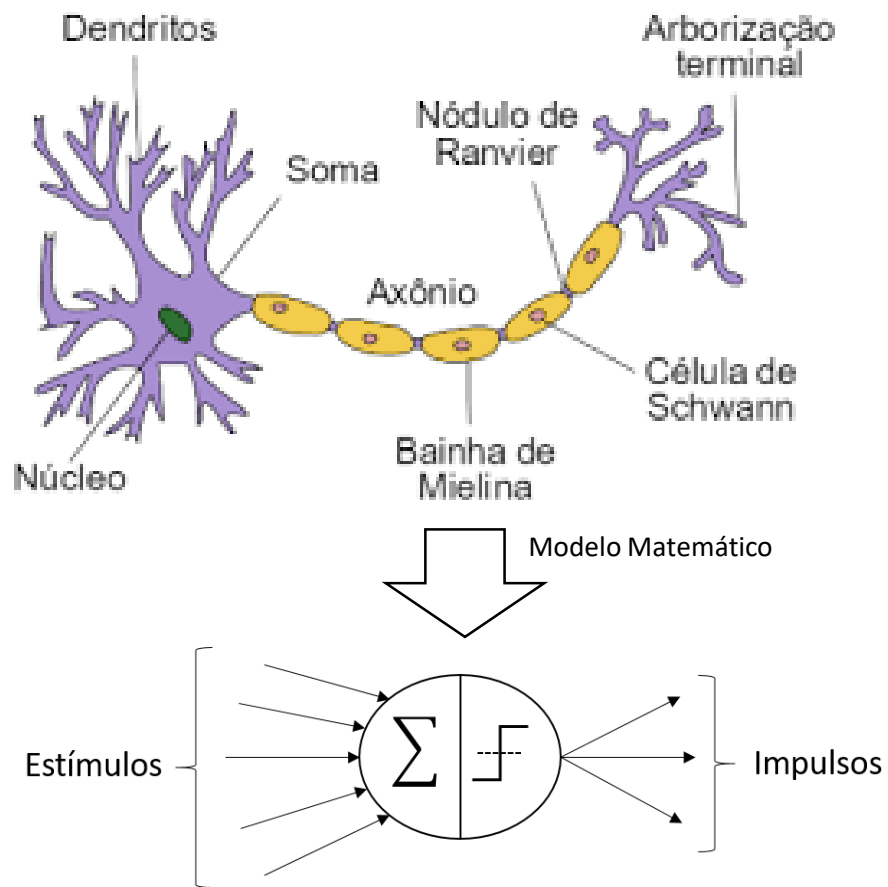


Um pouco de contexto



- Em termos bem simples, mas lembrando de que existem exceções, nós podemos simplificar o funcionamento do **neurônio** como:
 - O neurônio recebe estímulos elétricos, através dos **dendritos**.
 - Esses estímulos são somados no **corpo celular** (i.e., **soma**).
 - Se a soma dos estímulos exceder um certo **limiar de ativação**, o **neurônio** gera um pulso (ou **potencial de ação**) que é enviado pelos **terminais do axônio** a outros neurônios.

Um pouco de contexto



- Um **neurônio** pode se conectar a até 20.000 outros **neurônios** através das **sinapses**.
- Os sinais são passados de **neurônio** para **neurônio** através de **reações eletroquímicas**.
- Do ponto de vista do nosso curso, o **neurônio** será considerado como um **sistema com várias entradas e uma ou mais saídas** onde a comunicação entre neurônios é feita através de **sinais elétricos**.

O modelo de McCulloch e Pitts



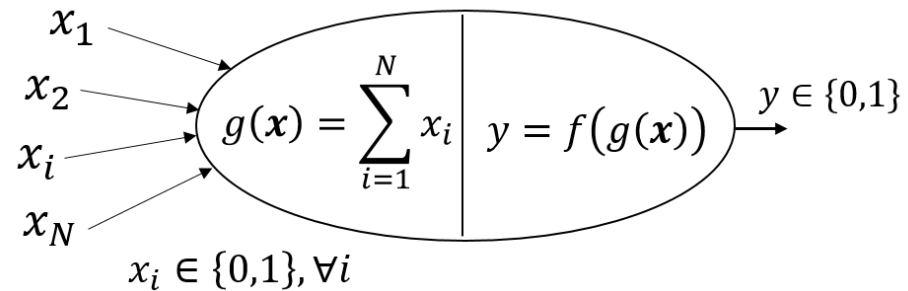
Walter Pitts e Warren McCulloch

- O final do século XIX e o início do século XX foram períodos fundamentais para o estabelecimento do conhecimento atual do sistema nervoso.
- De posse desse entendimento, em 1943, dois neurocientistas, Warren McCulloch e Walter Pitts apresentam em um artigo científico o primeiro **modelo computacional de um neurônio**.
- A partir desse modelo, foi possível estabelecer uma conexão entre o funcionamento de um neurônio e a ***lógica proposicional***.

O modelo de McCulloch e Pitts

- ***Lógica proposicional*** se baseia em ***proposições***.
 - Uma ***proposição*** é uma ***sentença declarativa*** ou ***afirmação***, ou seja, é uma sentença que faz uma ***afirmação sobre um fato***, podendo este ser verdadeiro ou falso.
- O artigo de McCulloch e Pitts fornece *insights* fundamentais sobre como a ***lógica proposicional pode ser processada por um neurônio***.
- Existe uma correspondência direta entre a lógica proposicional e a lógica Booleana.
 - Podemos pensar em uma ***sentença declarativa*** como sendo uma ***expressão Booleana***
 - $1 \text{ ou } 1 = 1$
 - $1 \text{ e } 0 = 0$
- A partir desta correspondência, a relação com a computação foi direta e natural.

O modelo de McCulloch e Pitts

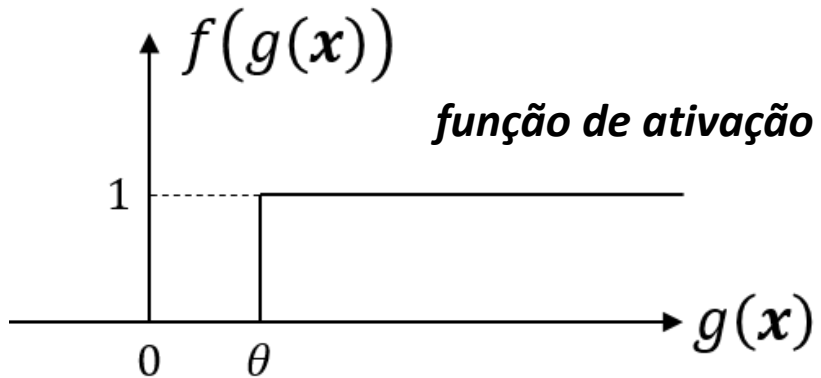


$$y = f(g(x)) = \begin{cases} 1, & \text{se } g(x) \geq \theta \\ 0, & \text{se } g(x) < \theta \end{cases}$$

onde θ é o **limiar de ativação**.

- A figura ao lado apresenta o modelo matemático do **neurônio** proposto por McCulloch e Pitts.
- Esse modelo é chamado de modelo de McCulloch e Pitts (M-P).
- Grosso modo, o **neurônio** é ativado (ou disparado) quando a **soma** de suas entradas, $g(x)$, excede o **limiar de ativação**, θ , da função de ativação $f(\cdot)$.
- O modelo estabelece algumas premissas apresentadas a seguir.

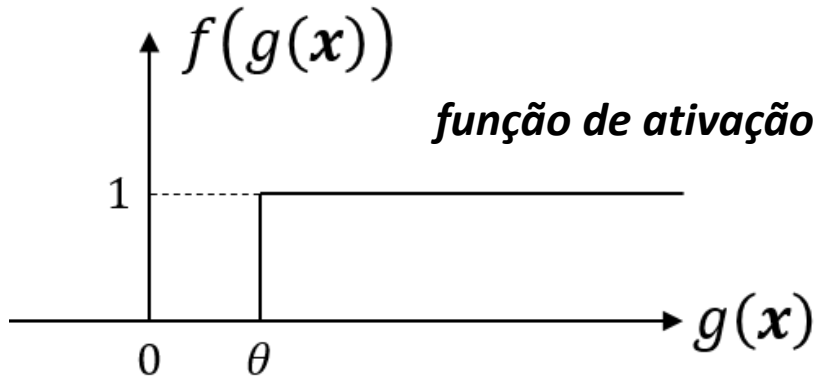
O modelo de McCulloch e Pitts



- As premissas desse modelo são:
 - Os valores das entradas, $x_i, \forall i$, ou também chamados de **sinapses**, são sempre valores **booleanos**, i.e., '0', ou '1'.
 - As entradas são multiplicadas por **pesos com magnitudes unitárias** (+/- 1) e somadas.
 - A atividade do **neurônio** é um processo do tipo "**tudo ou nada**", ou seja, um processo binário (0 ou 1).
 - Portanto, a **função de ativação** do neurônio é uma **função degrau** com **ponto de disparo variável**, dependente do **limiar de ativação**, θ .
 - Um certo número de **sinapses** deve ser excitado para que o neurônio "dispare".

O modelo de McCulloch e Pitts

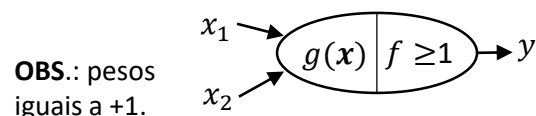
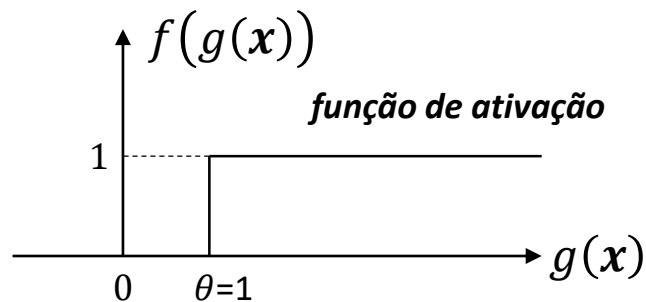
- Portanto, o modelo do ***neurônio*** de McCulloch e Pitts nada mais é do que um ***classificador linear*** com
 - ***limiar de decisão rígido,***
 - ***ponto de disparo variável,***
 - ***pesos com magnitudes unitárias e***
 - ***atributos booleanos.***



Exemplos de portas lógicas com o modelo M-P

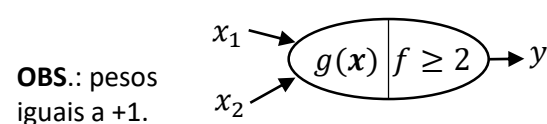
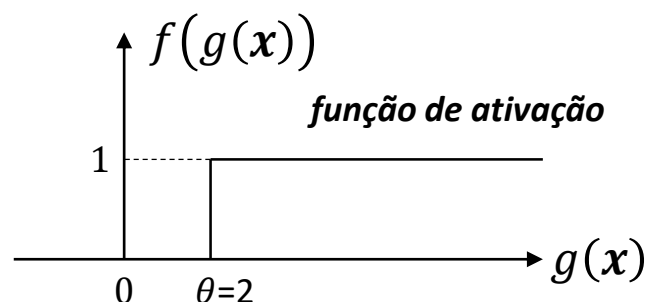
OR			
x_1	x_2	$g(x)$	y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

- Qual é o valor do **limiar de ativação**, θ ?
- Analisando-se $g(x)$, vemos que o disparo deve ocorrer quando $g(x) \geq 1$, portanto, $\theta = 1$.



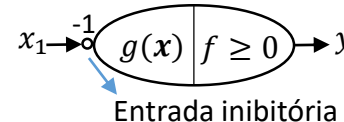
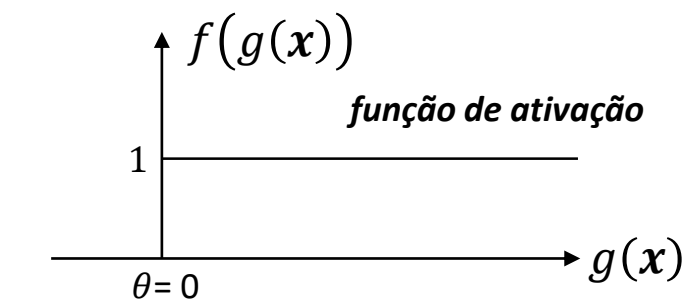
AND			
x_1	x_2	$g(x)$	y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

- Qual é o valor do **limiar de ativação**, θ ?
- Analisando-se $g(x)$, vemos que o disparo deve ocorrer quando $g(x) \geq 2$, portanto, $\theta = 2$.



NOT			
x_1	$-x_1$	$g(x)$	y
0	0	0	1
1	-1	-1	0

- Qual é o valor do **limiar de ativação**, θ ?
- Analisando-se x_1 , vemos que para o disparo ocorrer, seu valor deve ser **negado** (i.e., multiplicado por -1), e assim, o disparo ocorre quando $g(x) \geq 0$, portanto, $\theta = 0$.



OBS.: Entradas inibitórias são entradas que têm seus valores multiplicados por -1.

Todos esses exemplos podem ser interpretados como problemas de classificação.

Exemplos de portas lógicas com o modelo M-P

- Qual deve ser o valor do **limiar de ativação**, θ , para a porta lógica XOR?

Sem entradas inibitórias.

XOR			
x_1	x_2	$g(x)$	y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	0

$$0 < g(x) < 2$$

x_1 como entrada inibitória.

XOR			
$-x_1$	x_2	$g(x)$	y
0	0	0	0
0	1	1	1
-1	0	-1	1
-1	1	0	0

$$g(x) \leq -1 \text{ ou } g(x) \geq 1$$

x_2 como entrada inibitória.

XOR			
x_1	$-x_2$	$g(x)$	y
0	0	0	0
0	-1	-1	1
1	0	1	1
1	-1	0	0

$$g(x) \leq -1 \text{ ou } g(x) \geq 1$$

x_1 e x_2 como entradas inibitórias.

XOR			
$-x_1$	$-x_2$	$g(x)$	y
0	0	0	0
0	-1	-1	1
-1	0	-1	1
-1	-1	-2	0

$$-2 < g(x) < 0$$

- Resposta:** com um único modelo de M-P, não é possível encontrar um **limiar de ativação** que resolva este problema, pois como veremos adiante, este problema não é **linearmente separável**.
- O modelo de M-P só resolve problemas **linearmente separáveis**.

Tarefa

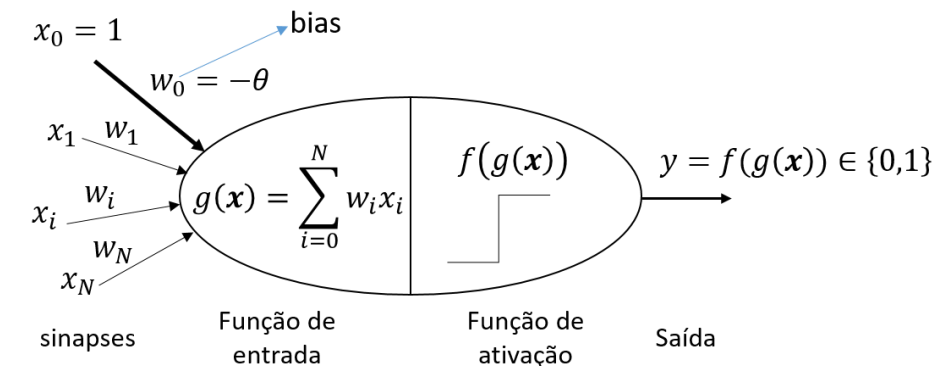
- **Quiz:** “*T320 - Quiz – Redes Neurais Artificiais (Parte I)*” que se encontra no MS Teams.

Perceptron

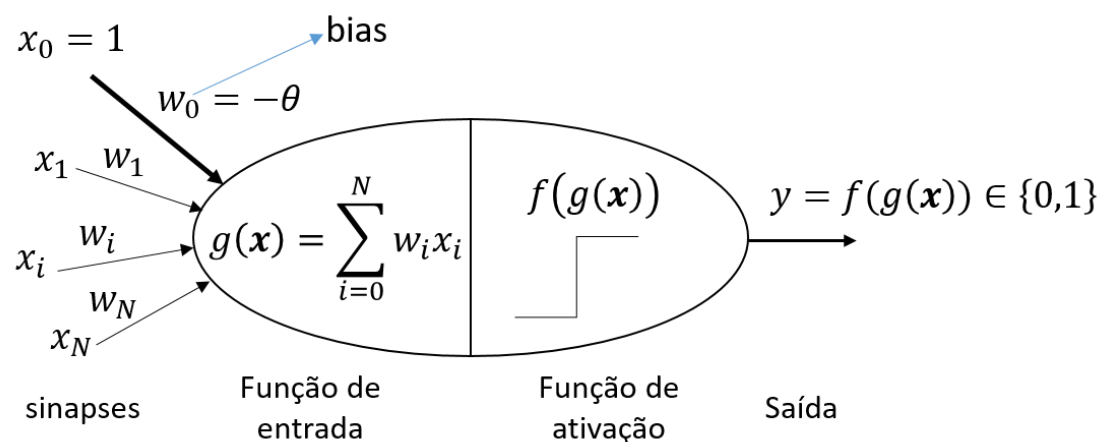
- Em 1958, Frank Rosenblatt, propôs um novo **modelo computacional mais geral** que o modelo do **neurônio** de McCulloch e Pitts.
- O modelo criado por ele foi chamado de **perceptron** e é mostrado na figura ao lado.
- O **perceptron** é um modelo para **aprendizado supervisionado** de **classificadores binários**, ou seja **problemas com duas classes**.
- Assim como o modelo de M-P, o **perceptron** só é capaz de classificar padrões **linearmente separáveis**.
- Ou seja, o **perceptron** também não resolve o problema da classificação XOR.



Frank Rosenblatt



Perceptron

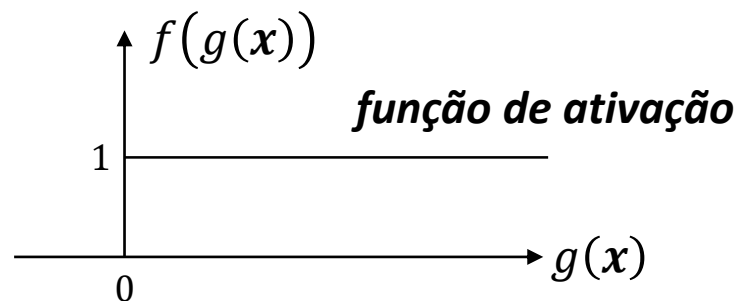


$$y = f(g(x)) = \begin{cases} 1, & \text{se } g(x) \geq 0 \\ 0, & \text{se } g(x) < 0 \end{cases}$$

Percebam que o **limiar de ativação**, θ , agora faz parte das entradas e é chamado de **peso de bias**.

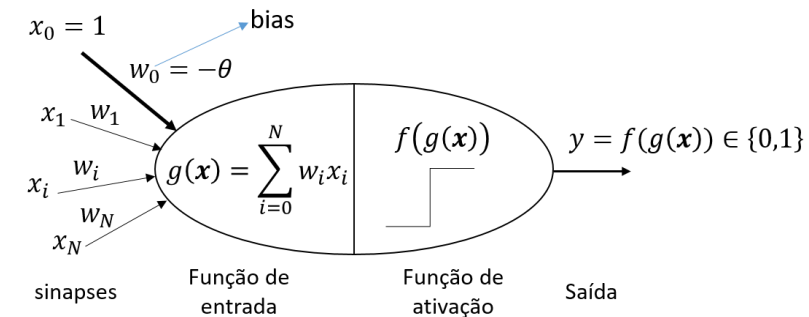
- Esse novo modelo supera algumas das limitações do modelo de M-P:

- Introdução do conceito de **pesos sinápticos com valores reais** para as entradas (ou **sinapses**).
 - Pesos dão uma medida de importância das sinapses (i.e., atributos).
- E um método para que o modelo **aprenda os pesos e o ponto de ativação**, que passa a ser um peso também.



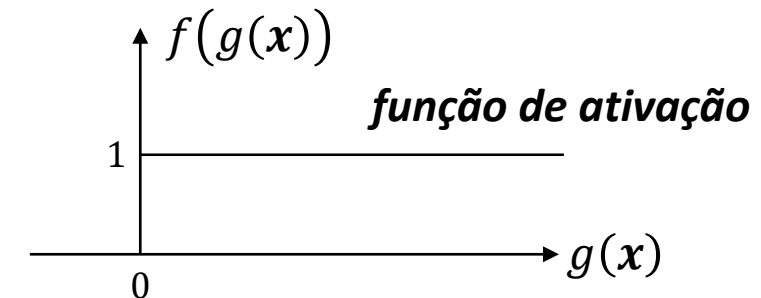
Perceptron

- Além disso, as entradas não são mais limitadas a valores booleanos, como no caso do modelo de M-P, **suportando entradas com valores reais**, o que torna este modelo mais útil e generalizado.
- Assim como no modelo de M-P, a **função de ativação** utilizada pelo **perceptron** também é a **função degrau** com a **diferença que a transição não mais depende do limiar de ativação, θ** .
- Ou seja, a transição ou ativação sempre ocorre em 0.



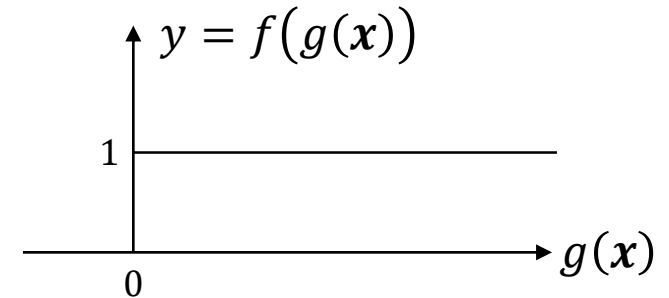
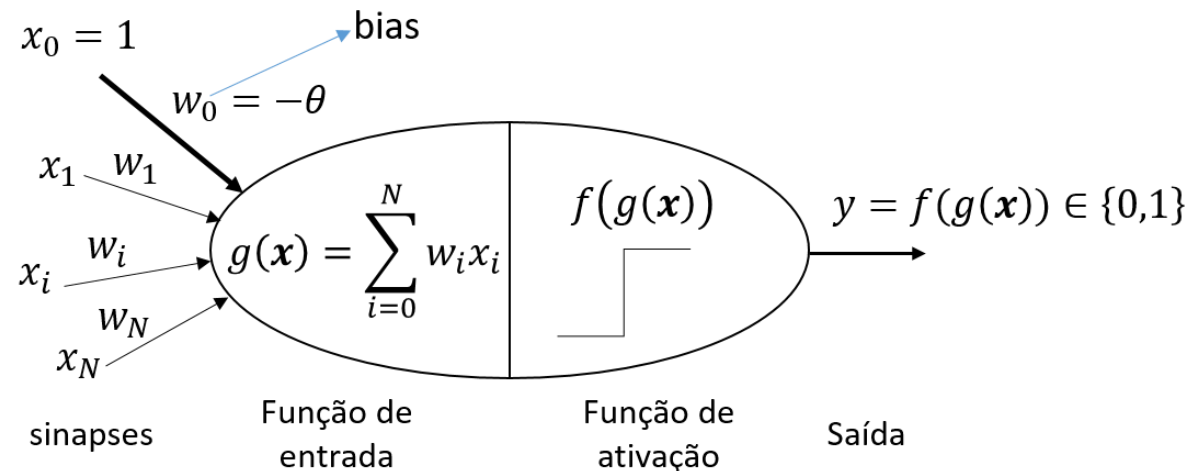
$$y = f(g(x)) = \begin{cases} 1, & \text{se } g(x) \geq 0 \\ 0, & \text{se } g(x) < 0 \end{cases}$$

Percebam que o **limiar de ativação**, θ , agora faz parte das entradas e é chamado de **bias**.



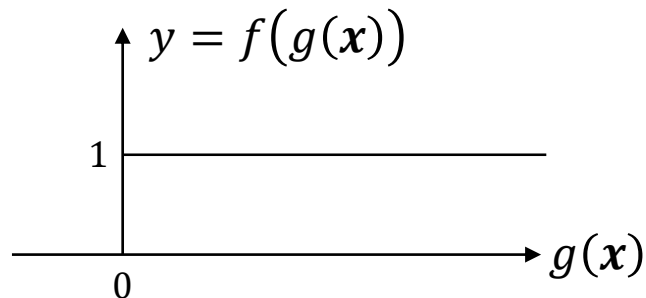
A ativação do Perceptron

- A ativação do **perceptron** é causada pela **combinação linear** dos **estímulos de entrada** em relação aos **pesos sinápticos**.
 - Se a **combinação linear** exceder o **limiar de ativação**, θ , o **disparo** ocorre.
 - Isso é expresso por uma **função de ativação** do tipo **degrau**.



A ativação do Perceptron

- Notem que a **função de ativação**, $f(\cdot)$, abaixo tem a transição para o valor 1 quando $g(x) = 0$ e o **limiar de ativação** é controlado, indiretamente, pelo valor do **peso de bias**, w_0 .
 - O **limiar de ativação** foi absorvido pela combinação linear, $g(x)$, e, portanto, podemos usar a **função de ativação com transição fixa em zero**, pois, agora, ajusta-se o limiar de ativação indiretamente através da atualização do peso w_0 .



Para que tenhamos a saída do perceptron, y , igual a 1, então

$$w_0 + \sum_{i=1}^N w_i x_i \geq 0 \quad \therefore \quad \sum_{i=1}^N w_i x_i \geq -w_0$$

Por exemplo

- Se $w_0 = 1$, $\sum_{i=1}^N w_i x_i \geq -1$.
- Se $w_0 = -1$, $\sum_{i=1}^N w_i x_i \geq 1$.

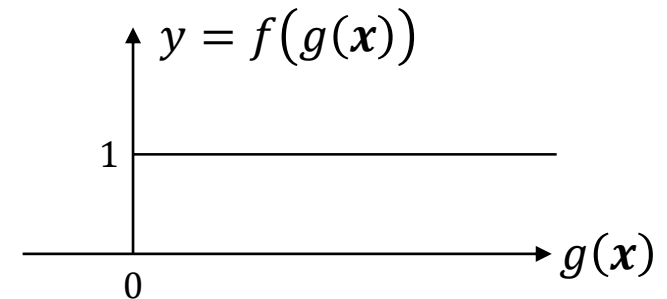
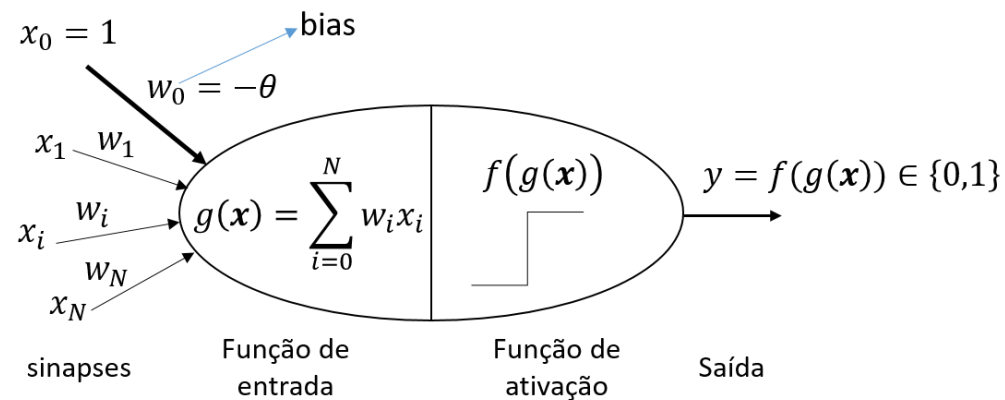
A ativação do Perceptron

- Como podemos ver, a **função discriminante**, $g(\mathbf{x})$, do **perceptron** tem a forma de um **hiperplano**

$$g(\mathbf{x}) = \sum_{i=0}^N w_i x_i, \quad \text{combinação linear das entradas}$$

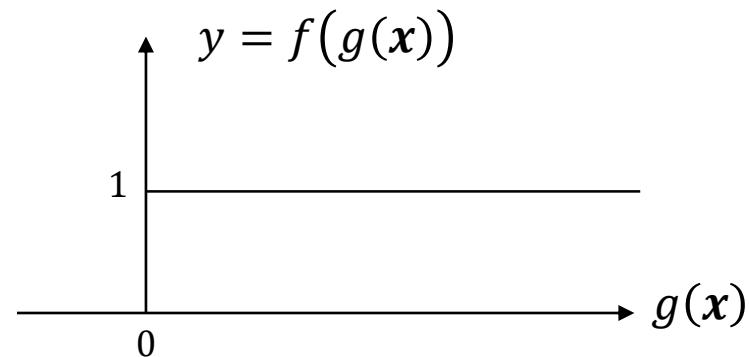
onde x_0 é o atributo de bias com valor constante igual a 1.

- Portanto, como já sabemos, este tipo de função dá origem a um **classificador binário** onde as classes são separadas por uma **superfície de separação linear**.




Regra de aprendizado do perceptron

- Devido ao fato da **função degrau**, $f(g(x))$, ter derivada igual a zero em todos os pontos, exceto em $g(x) = 0$, onde ela é indefinida, nós não podemos utilizar o **gradiente descendente**.
- Entretanto, como aprendemos anteriormente, usamos a **regra de aprendizado do perceptron** para treinar o modelo.
- É uma regra **simples e intuitiva** para atualização dos pesos do modelo.



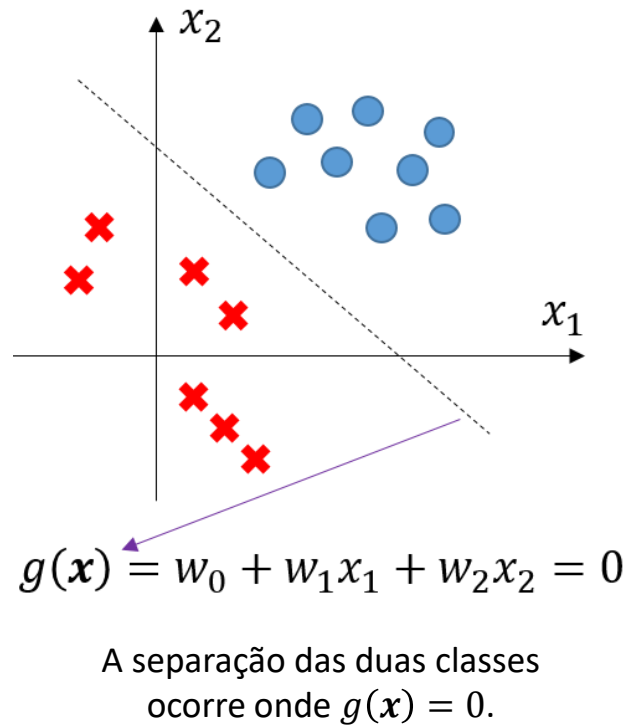
Regra de aprendizado do perceptron

- No caso do perceptron, onde $g(\mathbf{x})$, por definição, é um **hiperplano**, a regra converge para uma solução perfeita se as classes forem **linearmente separáveis**:
 - Classes **suficientemente espaçadas** e que podem ser separadas por um **hiperplano**.
- A **equação de atualização dos pesos** é definida como
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - \hat{y})\mathbf{x},$$


Equação idêntica a da atualização do gradiente descendente estocástico.

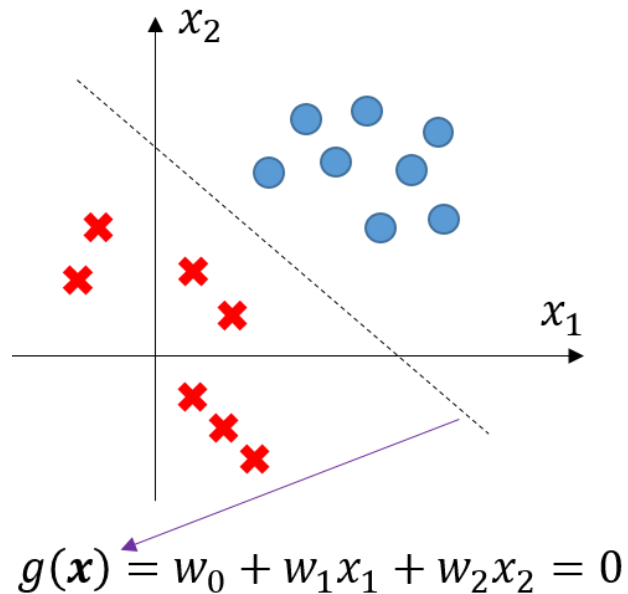
onde \mathbf{w} é o vetor de pesos, α é o passo de aprendizagem, y é o valor de saída esperado, \hat{y} é a saída do modelo, i.e., $f(g(\mathbf{x}))$, e \mathbf{x} é o vetor de atributos.

Perceptron



- Como percebemos, o **perceptron** é idêntico ao **classificador binário com limiar de decisão rígido**.
- Por definição, o **perceptron** sempre utiliza **superfícies de separação lineares**, ou seja, sempre teremos $g(\mathbf{x})$ como sendo a equação de um **hiperplano**.
- Portanto, teoricamente, **sem transformação dos atributos**, um **único perceptron** só é capaz de **classificar** dados que sejam **linearmente separáveis** (ou seja, separáveis por um **hiperplano**).

Perceptron

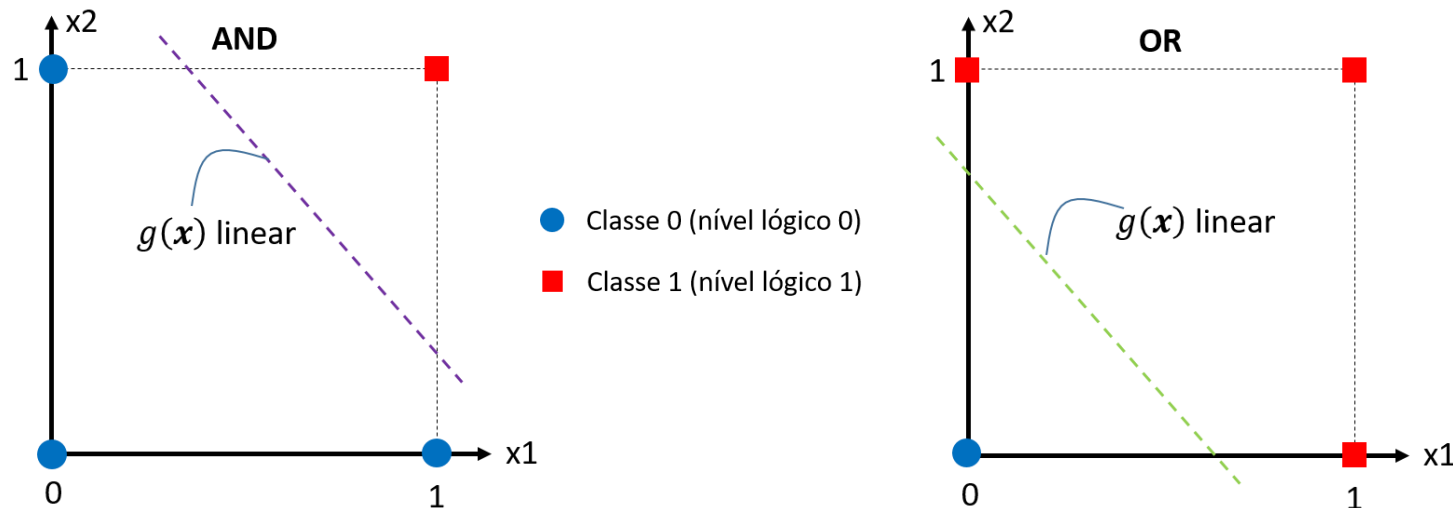


A separação das duas classes
ocorre onde $g(\mathbf{x}) = 0$.

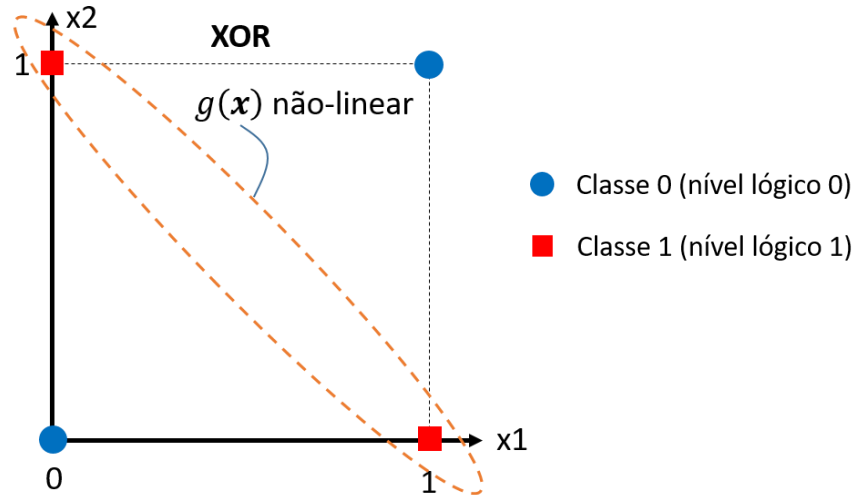
- A figura ao lado ilustra isso para um caso bidimensional.
- Entretanto, como veremos na sequência, podemos ***combinar os resultados de vários perceptrons*** para criar ***superfícies de separação*** que separem dados que não sejam linearmente separáveis sem a necessidade de ***transformar os atributos***.
- Ou seja, não precisamos usar funções discriminantes, $g(\mathbf{x})$, com outros formatos (e.g., polinômios) que não sejam o de um hiperplano.

Perceptron

- Por serem ***linearmente separáveis***, as lógicas AND e OR podem ser separadas por um único perceptron.
- As figuras abaixo demonstram que uma simples reta consegue separar os dados das duas lógicas.



Perceptron

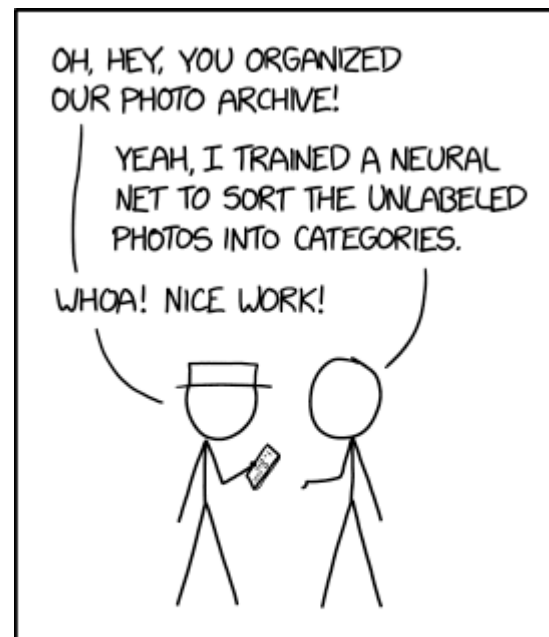
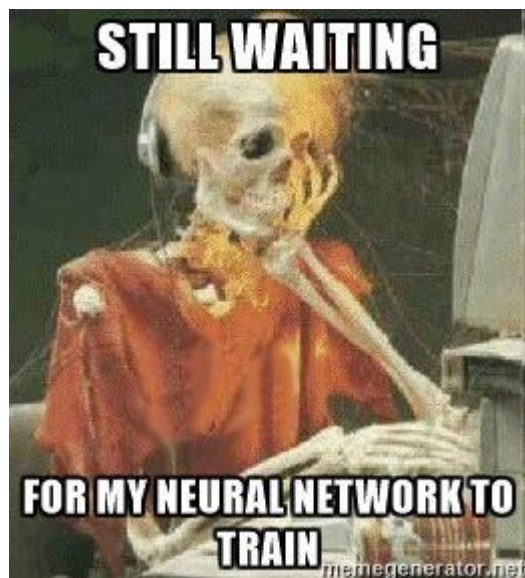


- Porém, a lógica XOR *não é linearmente separável* e necessita de uma *superfície de separação não-linear*.
- Vejam na figura abaixo que são necessárias *no mínimo duas retas paralelas*.
- Como veremos, a *separação da lógica XOR pode ser obtida combinando-se o resultado de dois perceptrons* (i.e., dois classificadores lineares), que resultará em uma superfície de separação não linear.

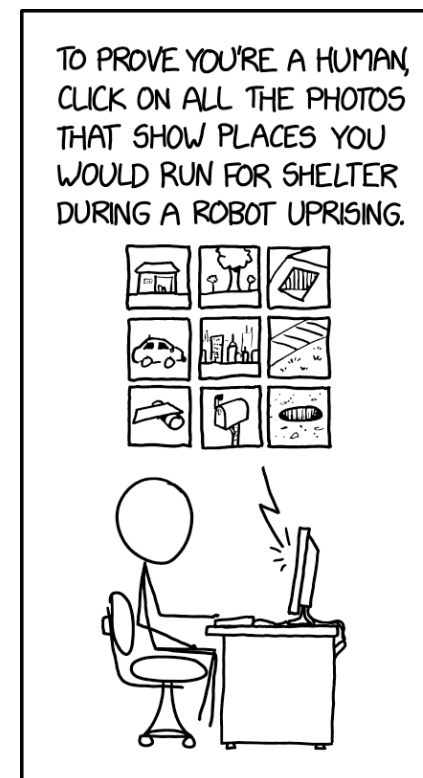
Tarefas

- **Quiz:** “*T320 - Quiz – Redes Neurais Artificiais (Parte II)*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #6](#).
 - Pode ser baixado do MS Teams ou do GitHub.
 - Pode ser respondido através do link acima (na nuvem) ou localmente.
 - [Instruções para resolução e entrega dos laboratórios](#).

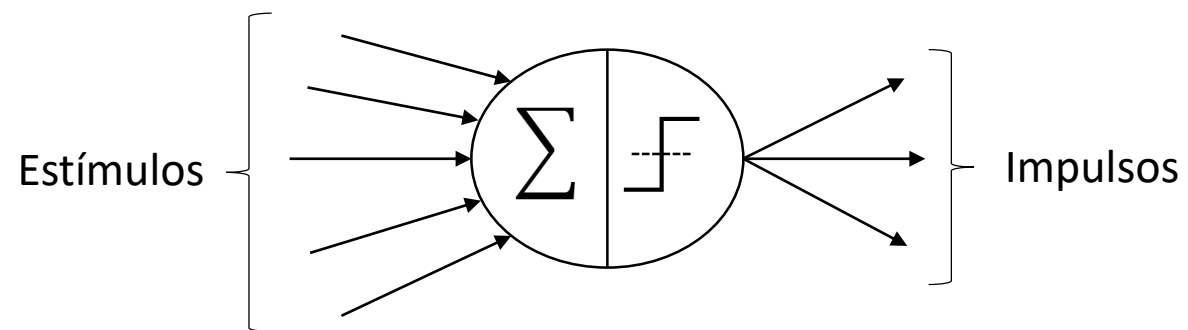
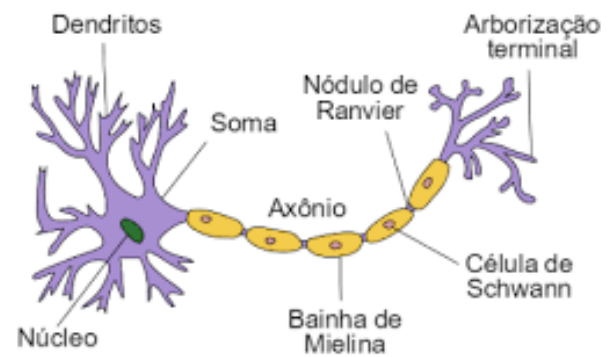
Obrigado!

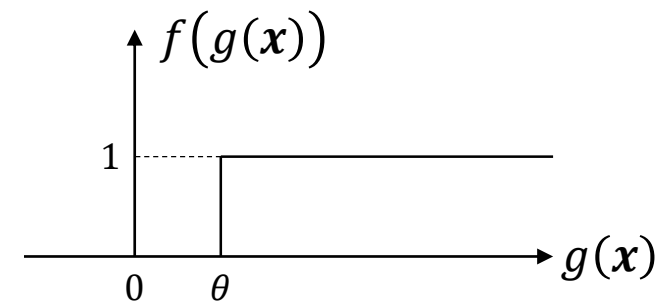
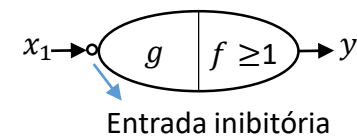
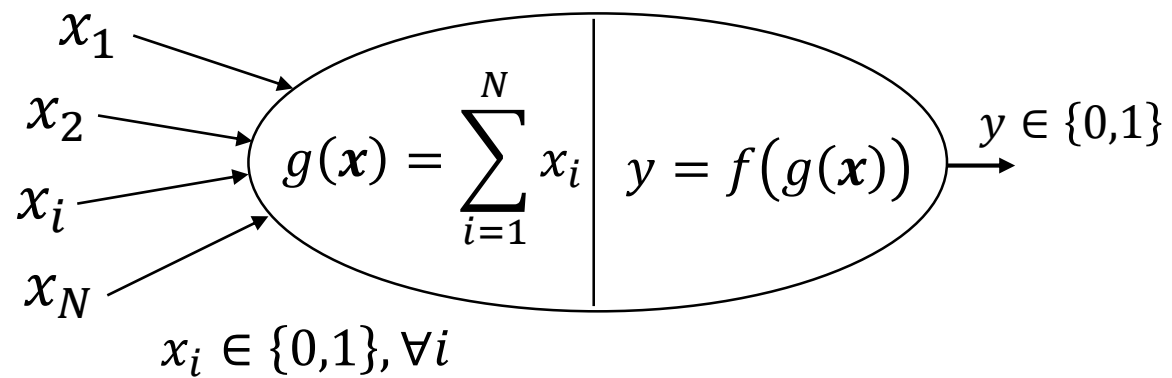
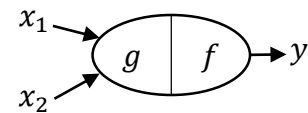


ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND,
YOU CAN TECHNICALLY SAY YOU
TRAINED A NEURAL NET TO DO IT.



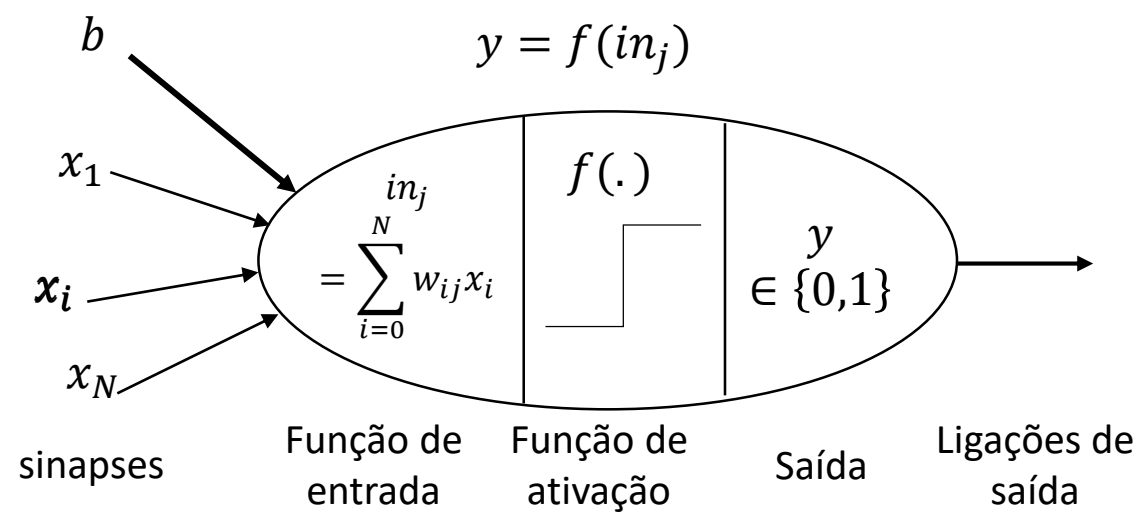
Figuras

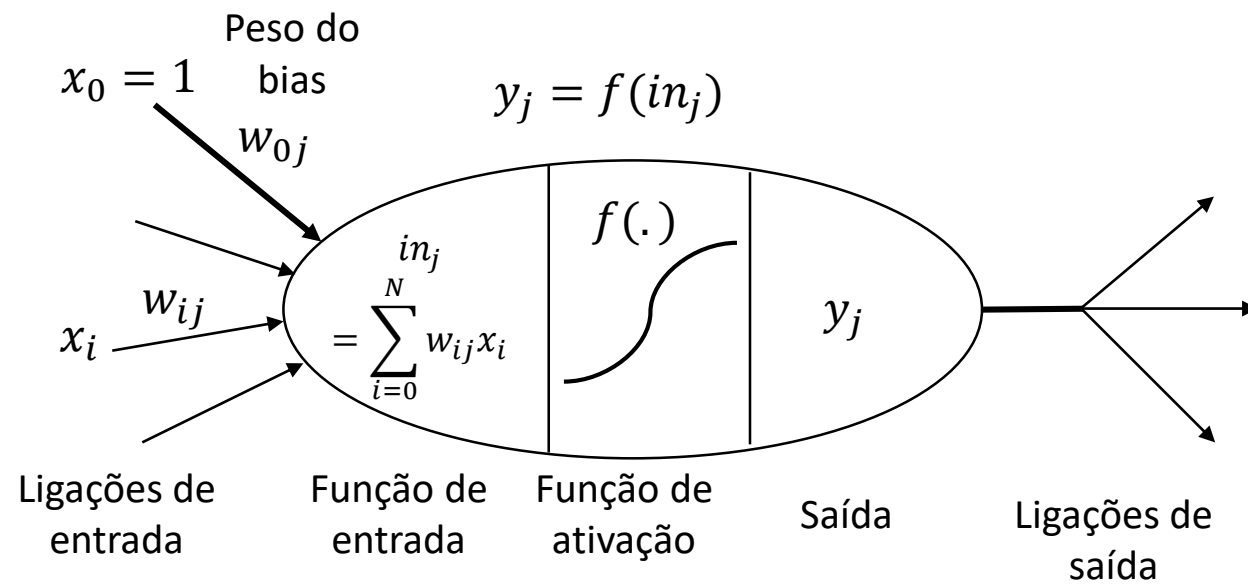


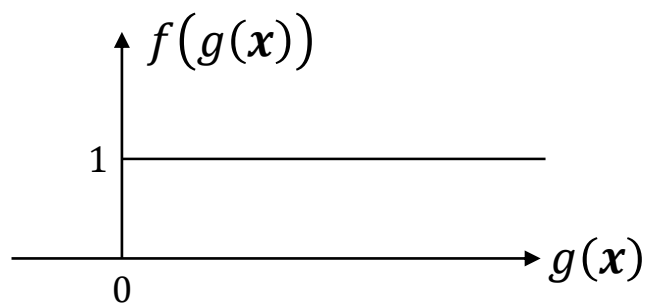
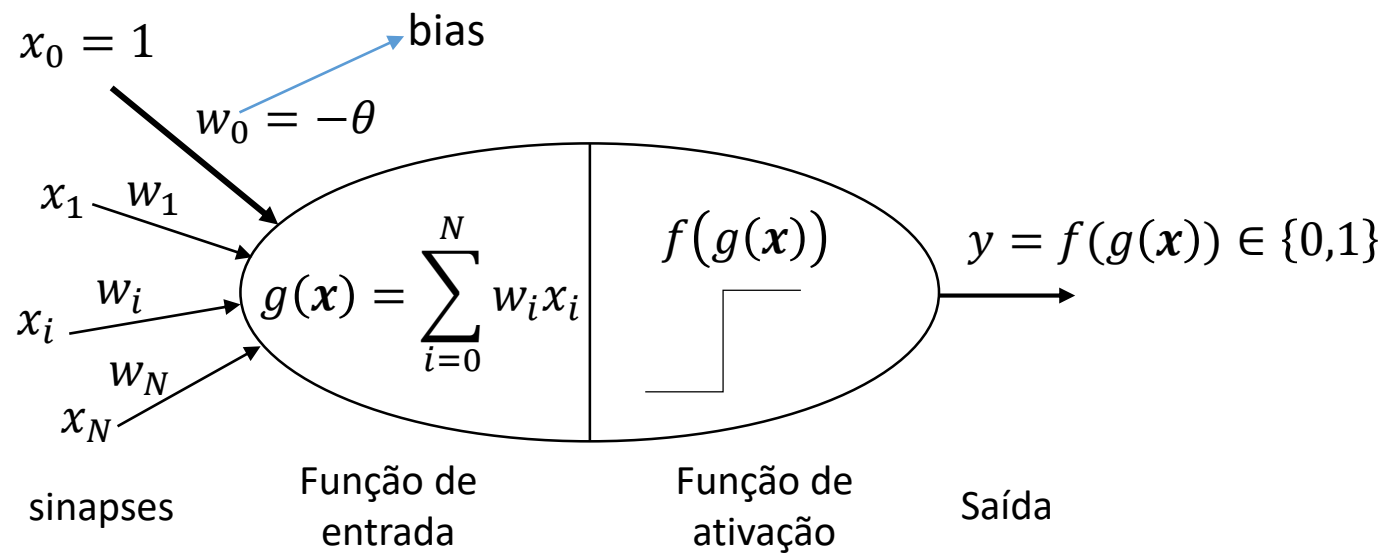


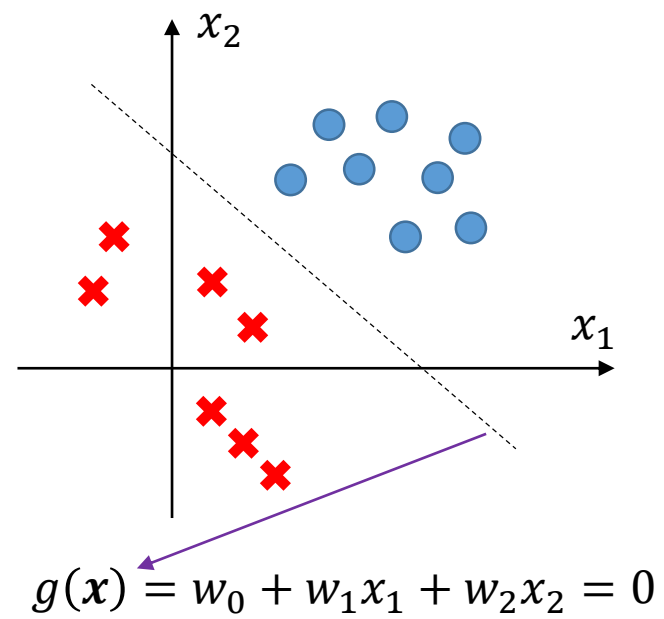
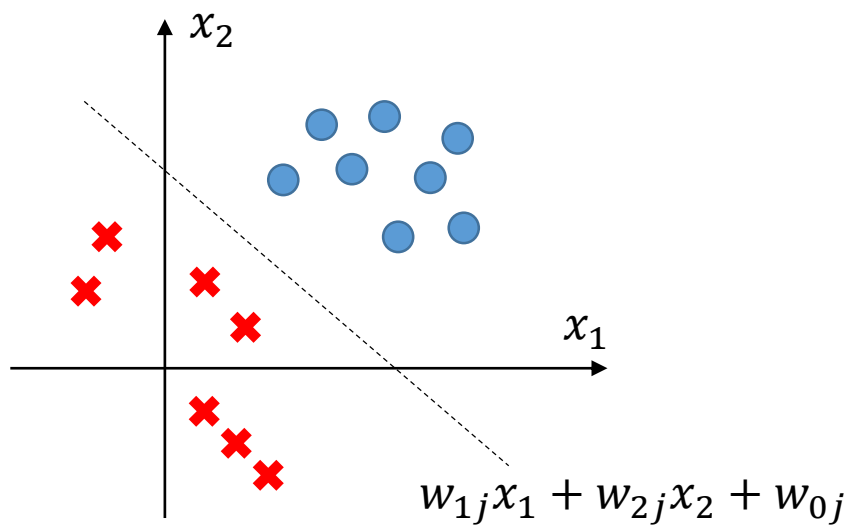
$$y = f(g(\mathbf{x})) = \begin{cases} 1 & \text{se } g(\mathbf{x}) \geq \theta \\ 0 & \text{se } g(\mathbf{x}) < \theta \end{cases}$$

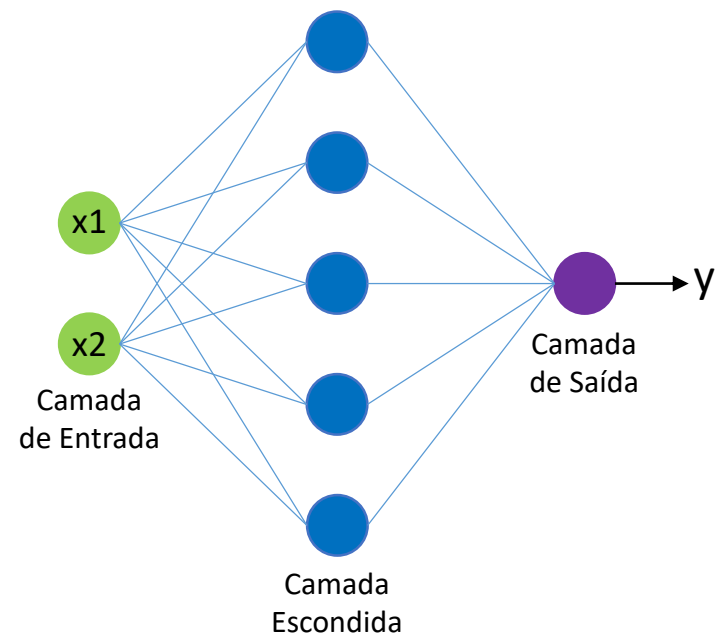
onde θ é o limiar de decisão.

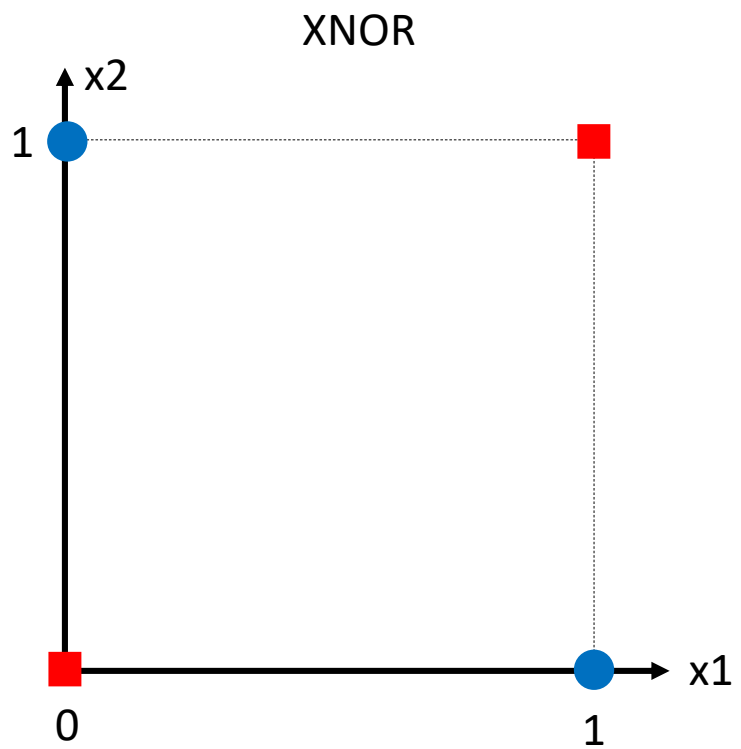






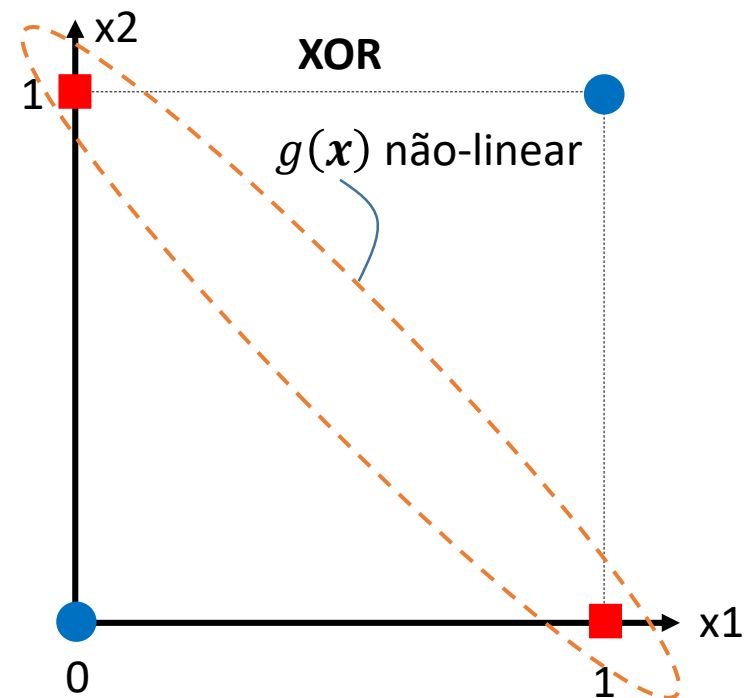
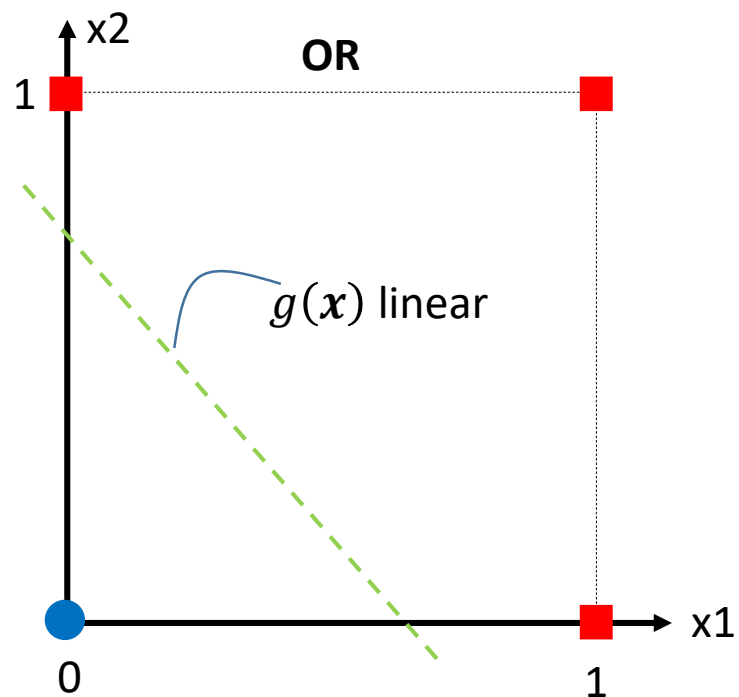
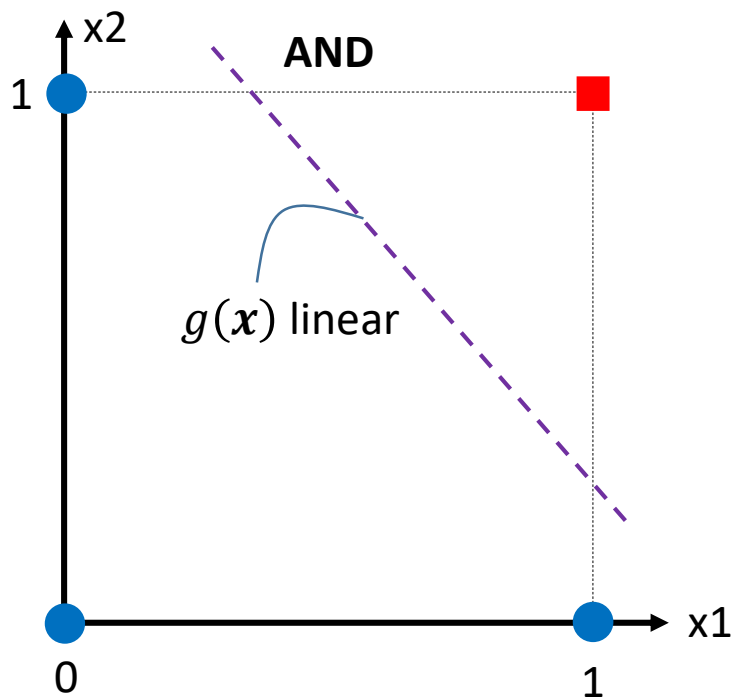






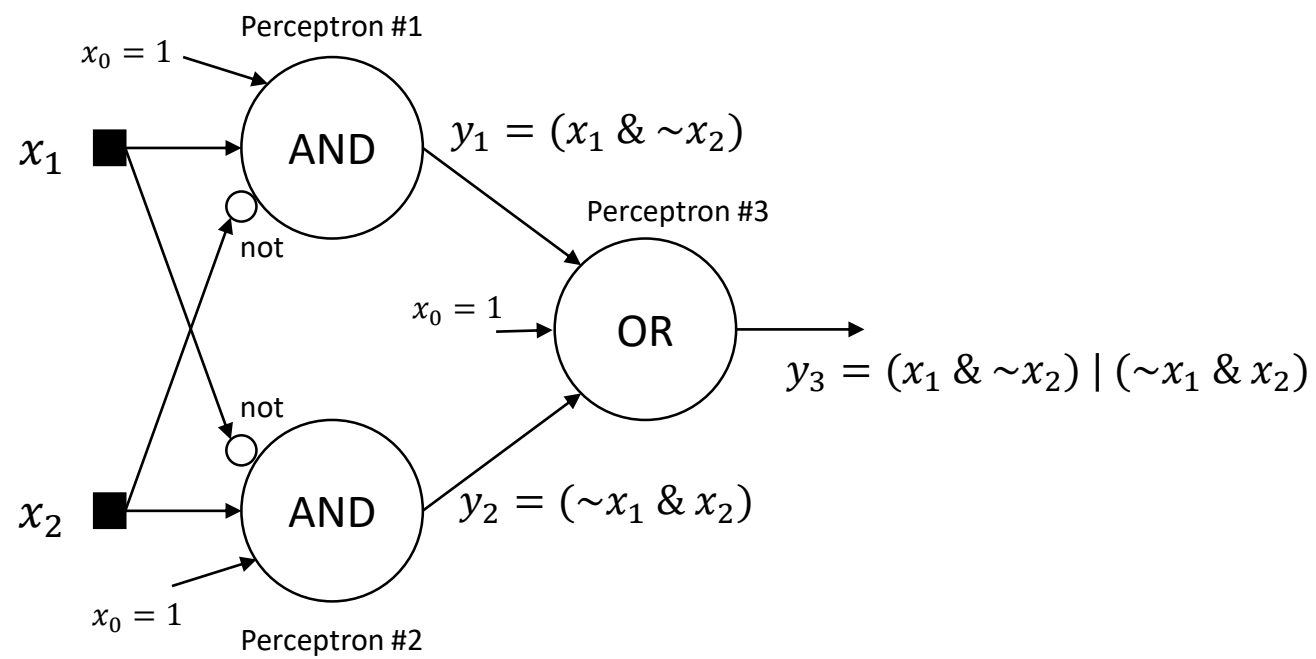
● Classe 0 (nível lógico 0)

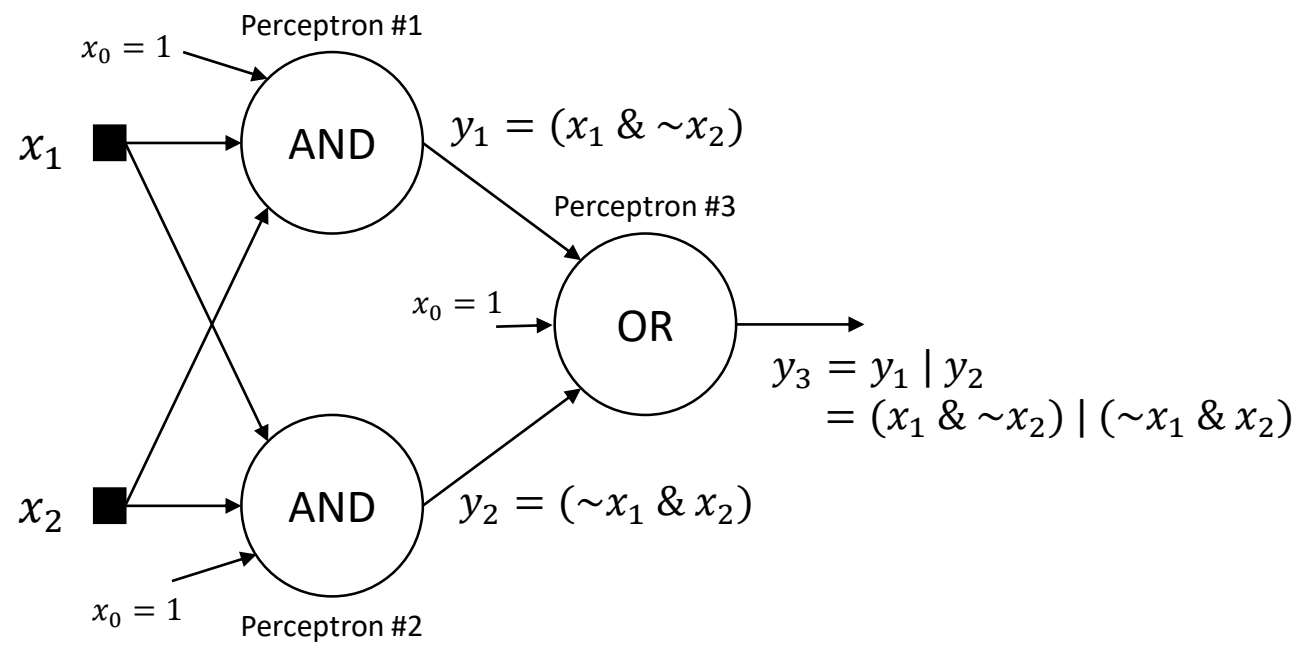
■ Classe 1 (nível lógico 1)

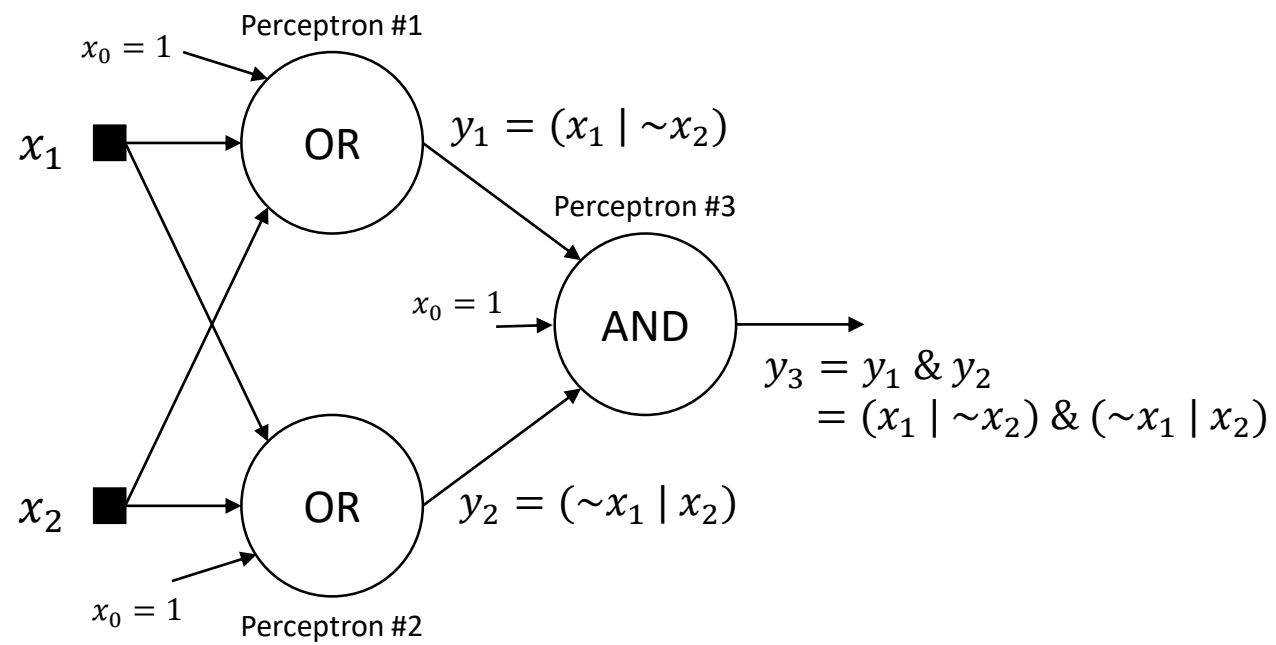


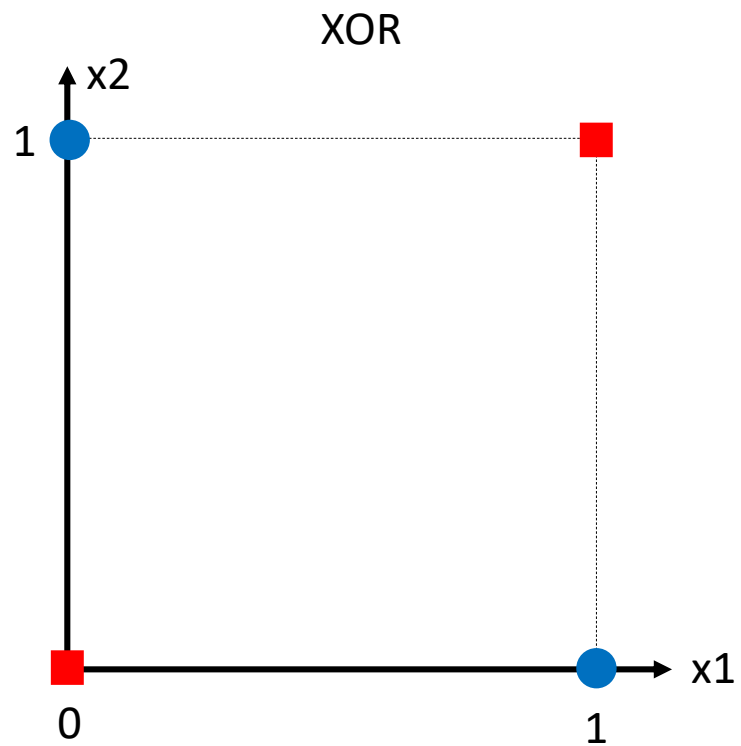
● Classe 0 (nível lógico 0)

■ Classe 1 (nível lógico 1)









● Classe 1 (nível lógico 1)

■ Classe 0 (nível lógico 0)

