

# TP555 - Inteligência Artificial e Machine Learning: *Redes Neurais Artificiais (Parte I)*



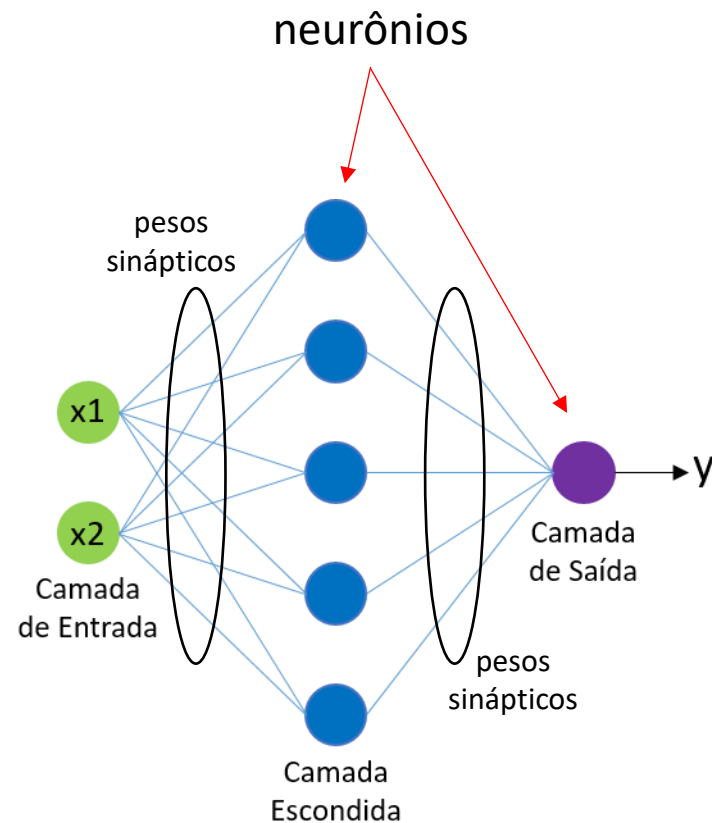
***Inatel***

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

# Introdução

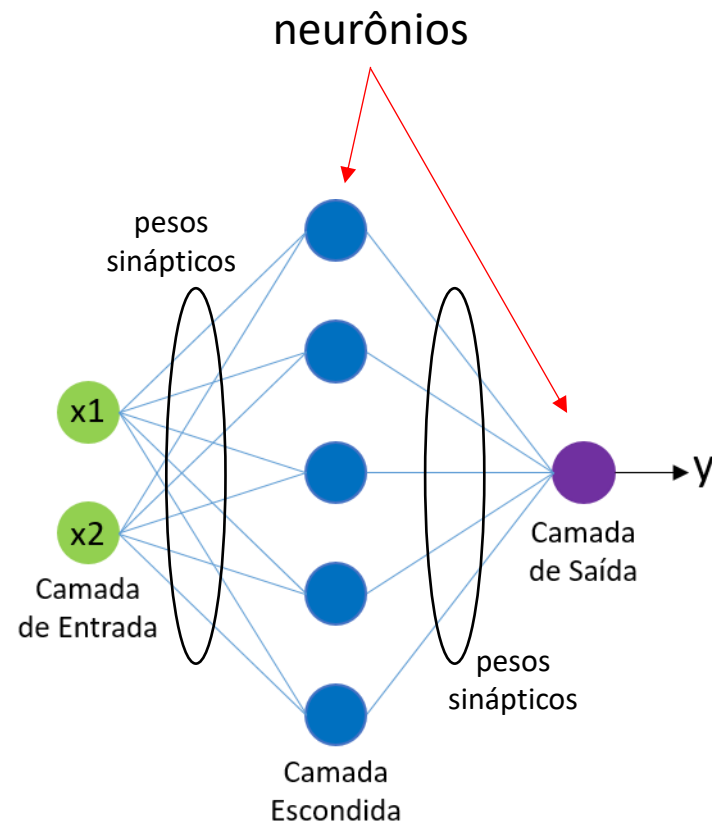
- A partir desta aula, entenderemos como as ideias que discutimos até agora serão úteis na construção de ***modelos matemáticos que aproximam a atividade de aprendizagem do cérebro.***
- Essas ideias que já discutimos, nos ajudarão a entender o funcionamento das ***redes neurais artificiais*** (RNAs).
- 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 grandes e 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),
  - Pilotar um veículo com pouca ou nenhuma intervenção humana,
  - Responder perguntas (e.g., ChatGPT).



Alexa



Google Assistant



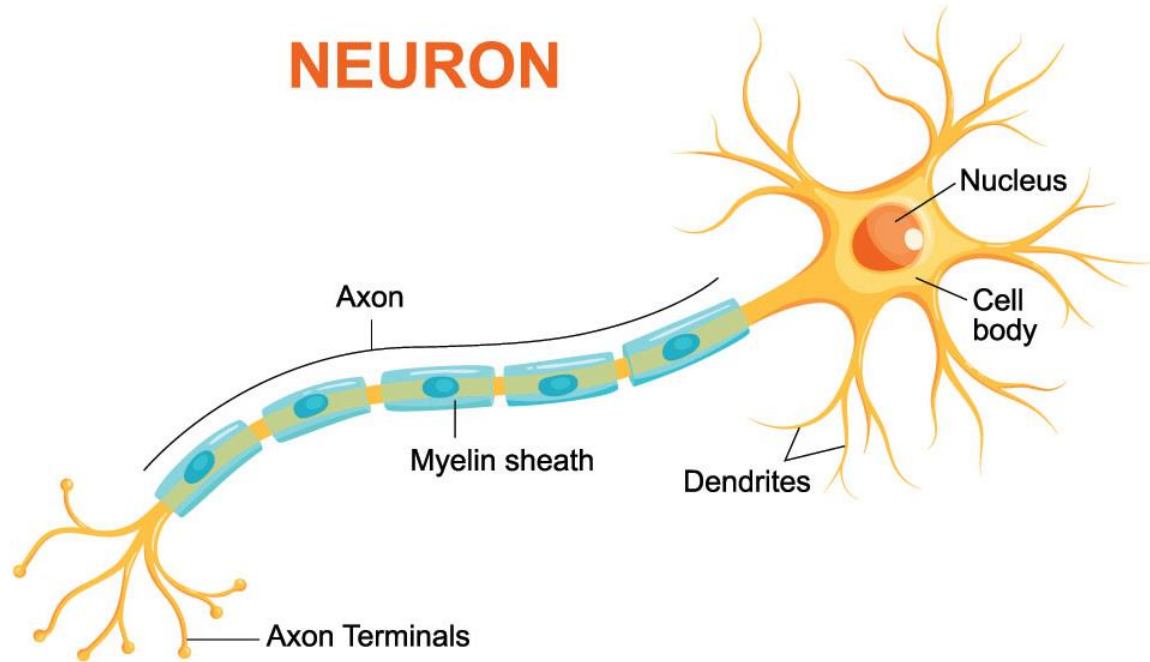
Siri



WAYMO

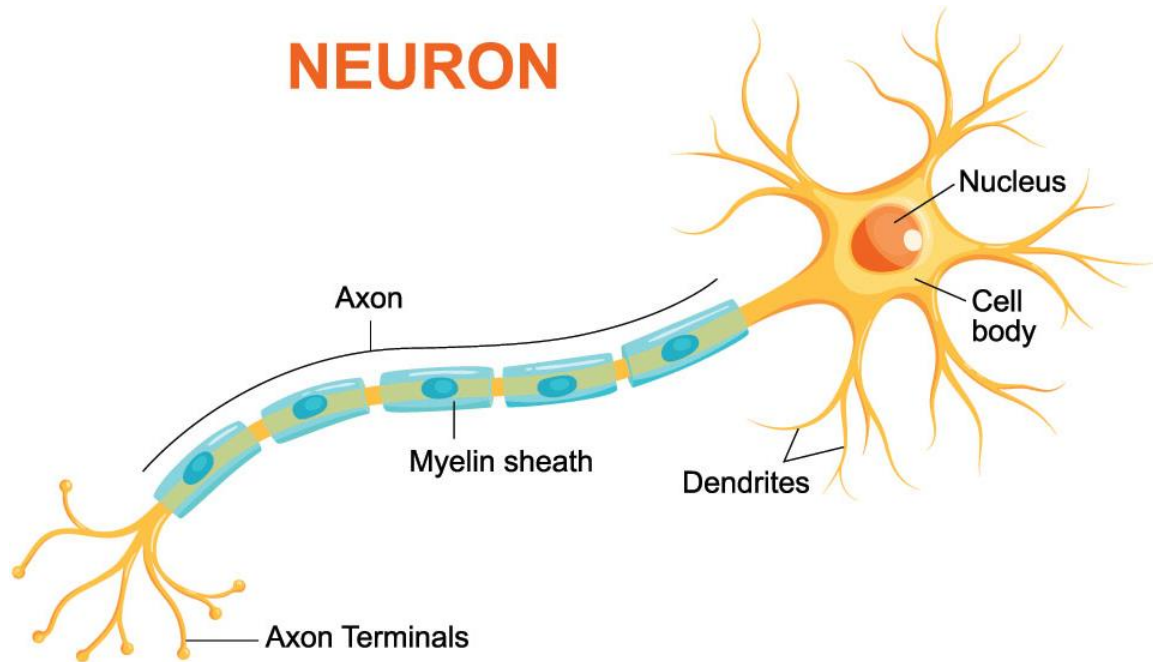


# Um pouco de contexto



- Os **neurônios** são células **eucariontes** que possuem **mecanismos eletroquímicos** característicos para transmissão de informações.
  - Células eucariontes possuem membrana celular, citoplasma e núcleo.
- Além da membrana celular, citoplasma e núcleo, os neurônios apresentam três partes importantes: os **dendritos**, o **axônio** e o **corpo celular (soma)**.

# Um pouco de contexto

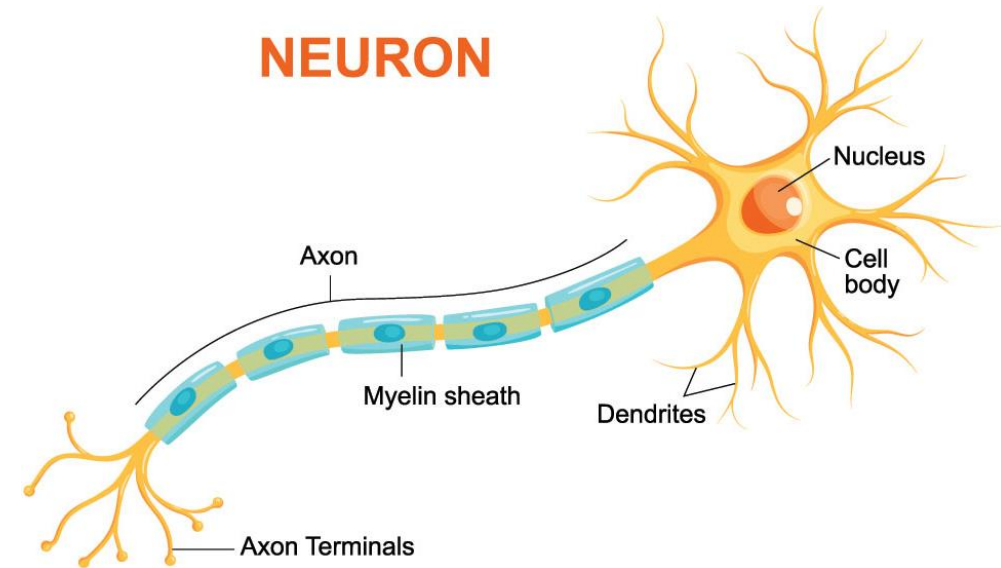


- Os **dendritos** são prolongamentos do neurônio que garantem a **recepção de estímulos** de outros neurônios, levando impulsos nervosos em direção ao **corpo celular**.
- O **axônio** é um prolongamento que garante o **envio de informação** (estímulos) **a outros neurônios** através de seus **terminais**.
- 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.
- Os **pontos de contato** entre os dendritos de um neurônio e os terminais do axônio de outro neurônio são chamados de **sinapses**.



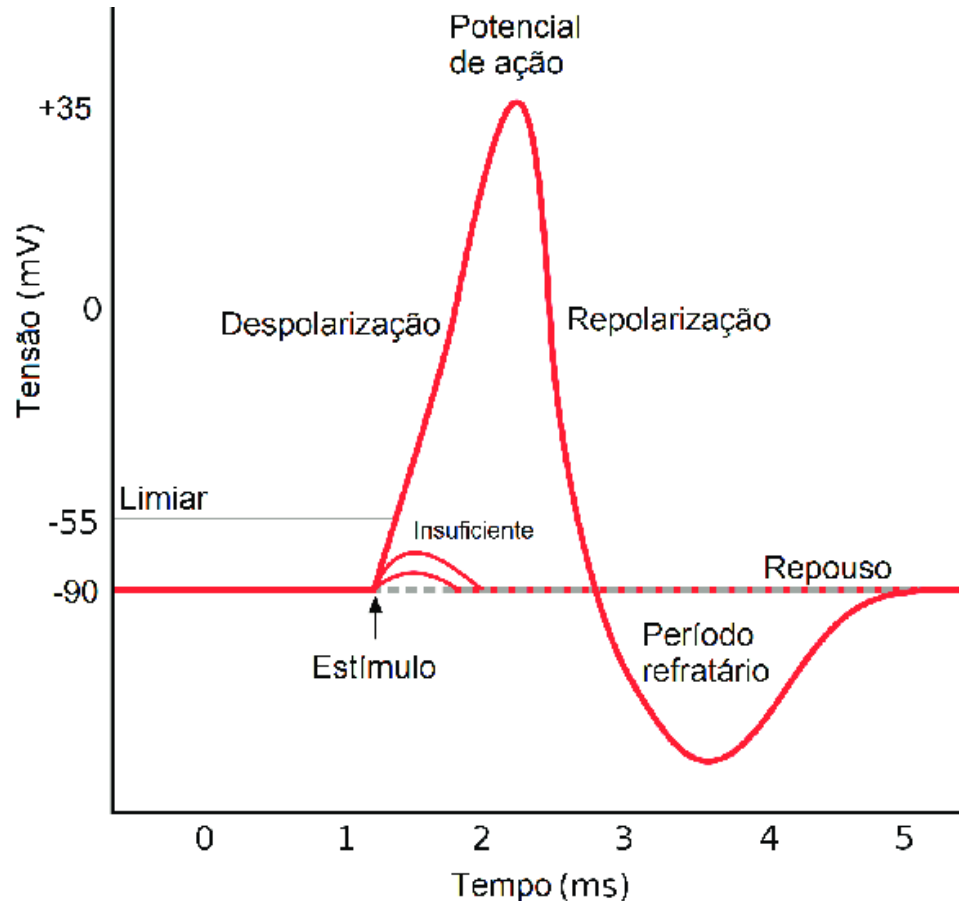


# Um pouco de contexto

- Ou seja, os neurônios se comunicam uns com os outros através das *sinapses*.
- Sinapses podem ser *químicas*, as mais comuns, ou *elétricas*, pouco comuns.
- A figura ao lado apresenta uma sinapse química.

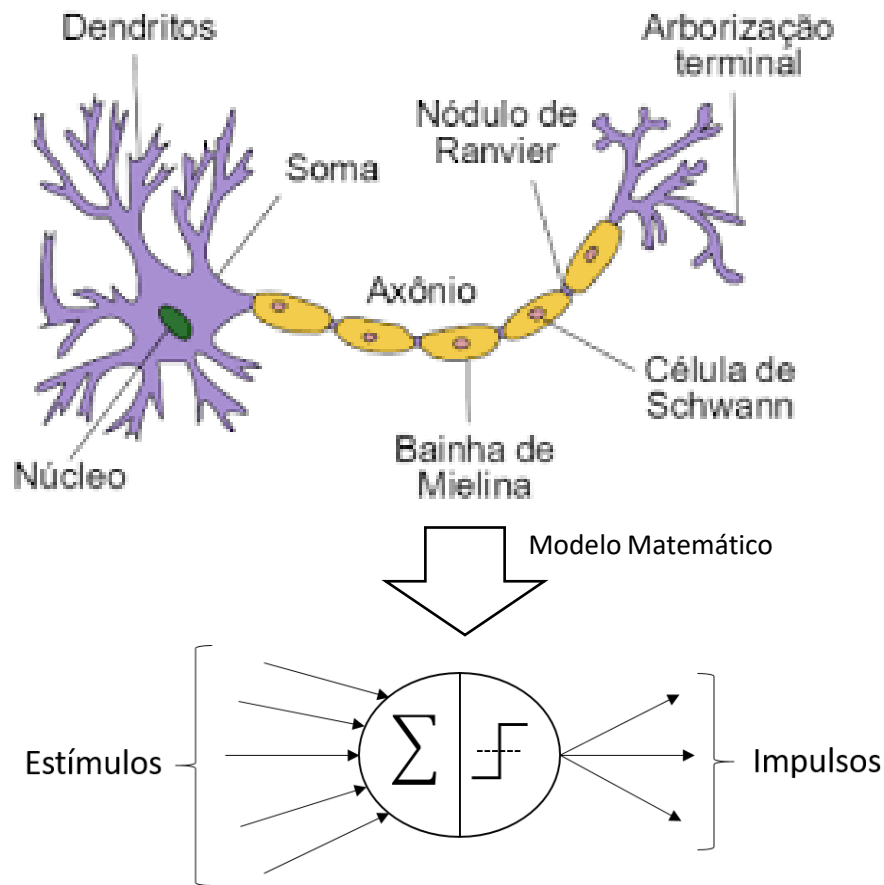


# Um pouco de contexto



- 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 das **sinapses** (químicas ou elétricas).
- 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



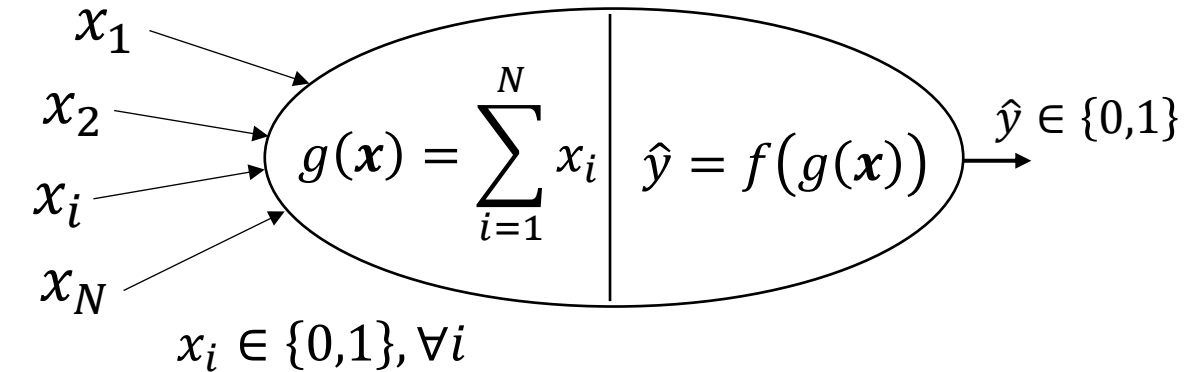
Walter Pitts e Warren McCulloch

- 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 forneceu *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

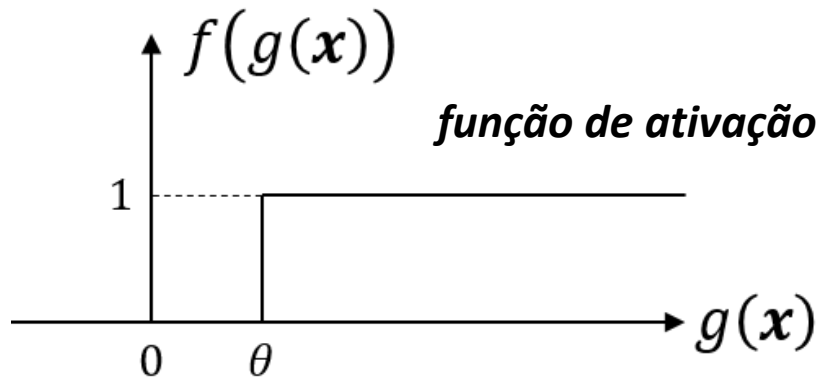


$$\hat{y} = f(g(x)) = \begin{cases} 1, & \text{se } g(x) \geq \theta \\ 0, & \text{se } g(x) < \theta \end{cases}$$

onde  $\theta$  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,  $\theta$** , da função de ativação  $f(\cdot)$ .
- O modelo estabelece algumas premissas apresentadas a seguir.

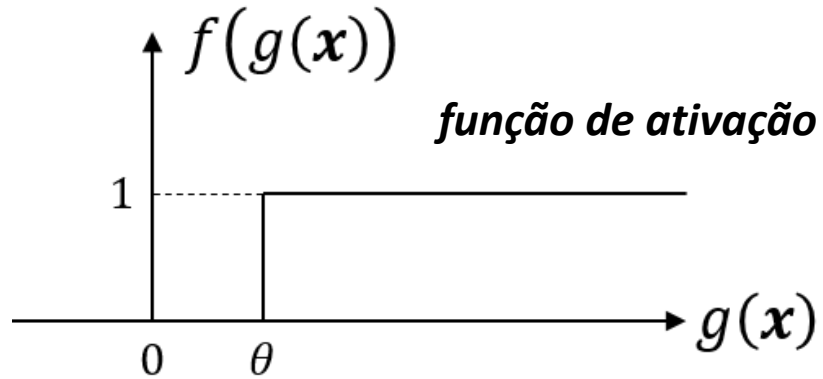
# Premissas do modelo de McCulloch e Pitts



- 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**,  $\theta$ .
- Um certo número de **sinapses** deve ser excitado para que o neurônio "dispare".



# O modelo de McCulloch e Pitts

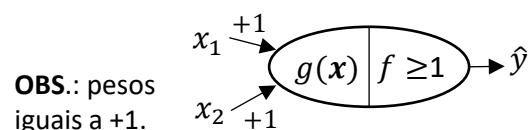
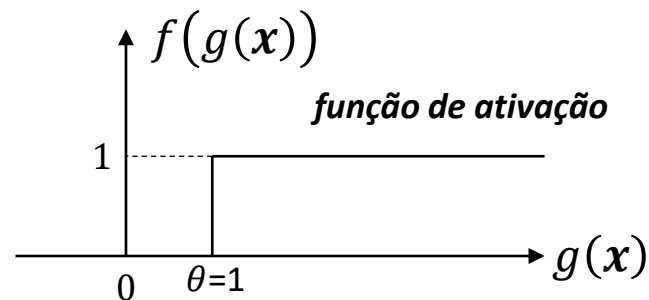


- Portanto, o modelo do **neurônio** de McCulloch e Pitts nada mais é do que um **classificador linear binário** com
  - **limiar de decisão rígido,**
  - **ponto de disparo variável,**
  - **pesos com magnitudes unitárias e**
  - **atributos booleanos.**
- Percebam que este **não é um modelo de aprendizado supervisionado.**
  - Vamos ver alguns exemplos com portas lógicas, os quais podem ser vistos como problemas de classificação binária.

# Exemplos de portas lógicas com o modelo M-P

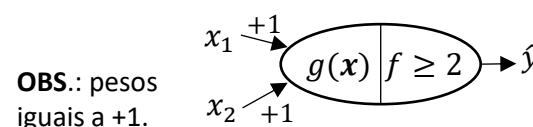
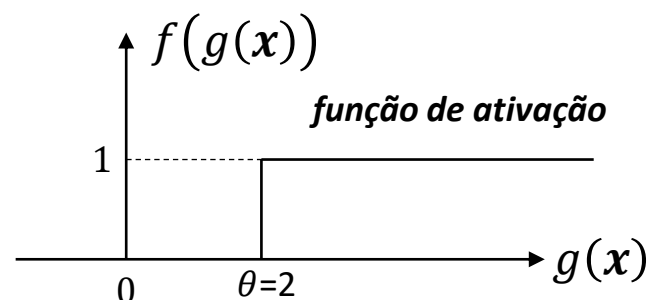
OR			
$x_1$	$x_2$	$g(x)$	$\hat{y}$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

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



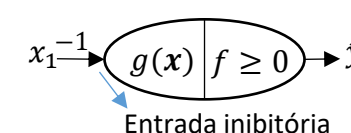
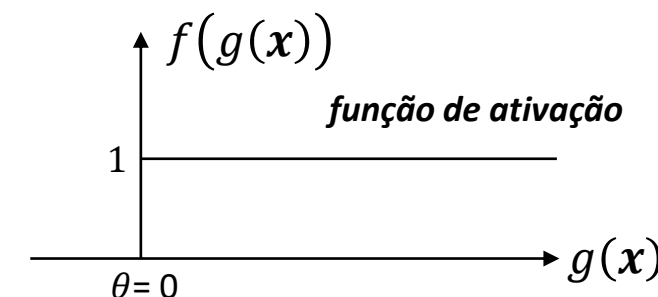
AND			
$x_1$	$x_2$	$g(x)$	$\hat{y}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

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



NOT			
$x_1$	$-x_1$	$g(x)$	$\hat{y}$
0	0	0	1
1	-1	-1	0

- Qual é o valor do **limiar de ativação**,  $\theta$ ?
- 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.

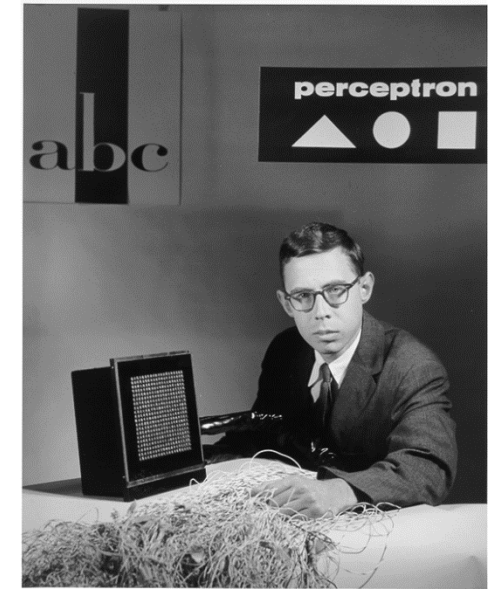
# Exemplos de portas lógicas com o modelo M-P

- Para casa:
  - Qual deve ser o valor do **limiar de ativação**,  $\theta$ , para a porta lógica XOR?
  - Dicas
    - Desenhem uma figura com os 4 pontos com cores diferentes representando as 2 classes.
    - Esse é um problema linearmente separável, ou seja, um hiperplano consegue separar as classes?

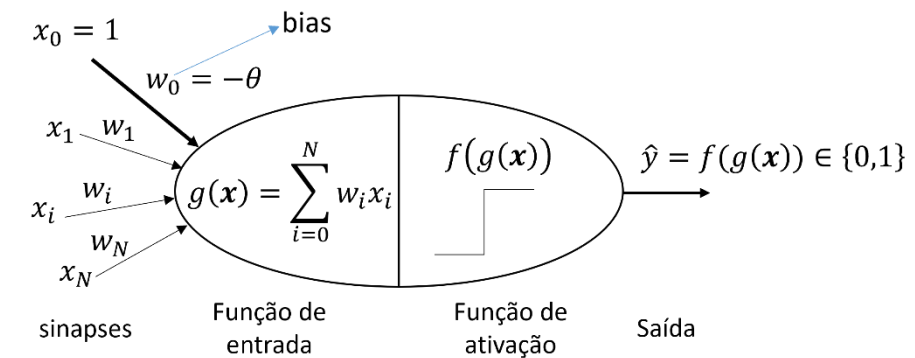
XOR		
$x_1$	$x_2$	$\hat{y}$
0	0	0
0	1	1
1	0	1
1	1	0

# Perceptron

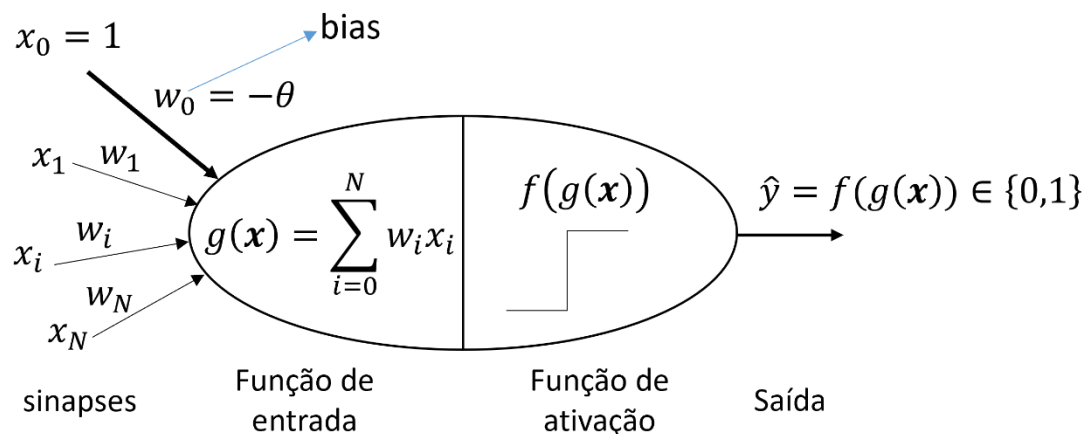
- Em 1958, Frank Rosenblatt, propôs um novo **modelo computacional mais geral** que o modelo do **neurônio** de McCulloch e Pitts.
- O modelo criado por ele foi chamado de **perceptron** e é mostrado na figura ao lado.
- O **perceptron** é um modelo para **aprendizado supervisionado** de **classificadores binários**.
- Assim como o modelo de M-P, por definição, o **perceptron** só é capaz de classificar padrões **linearmente separáveis**.



Frank Rosenblatt

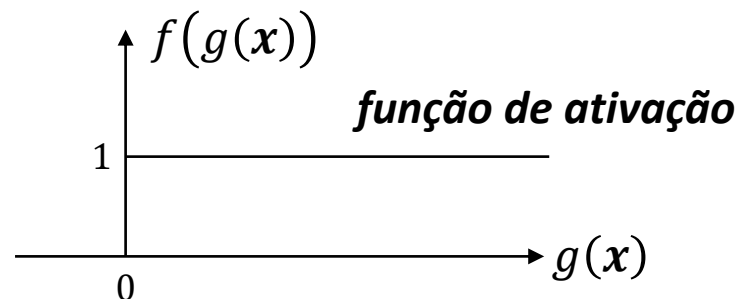


# Perceptron



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

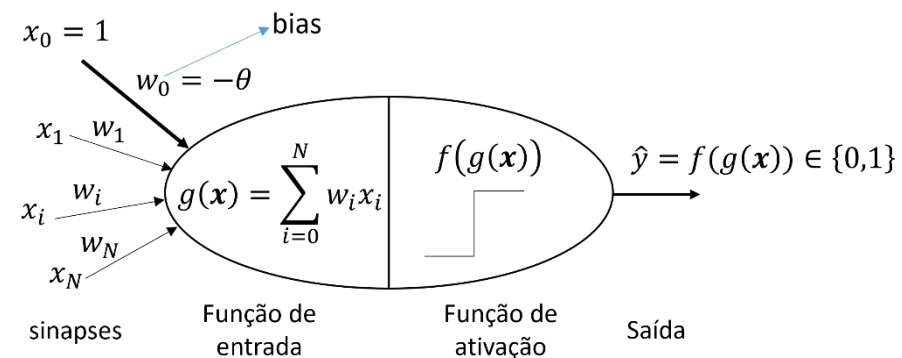
Percebam que o **limiar de ativação**,  $\theta$ , agora faz parte das entradas e é chamado de **peso de bias**.



- Esse novo modelo **supera algumas das limitações** do modelo de M-P:
  - Introdução de **pesos sinápticos com valores reais** para as entradas (i.e., **sinapses**).
    - Pesos dão uma medida de importância dos sinapses (i.e., atributos).
  - É um método para que o modelo **aprenda os pesos e o ponto de ativação**, que passa a ser um peso também.

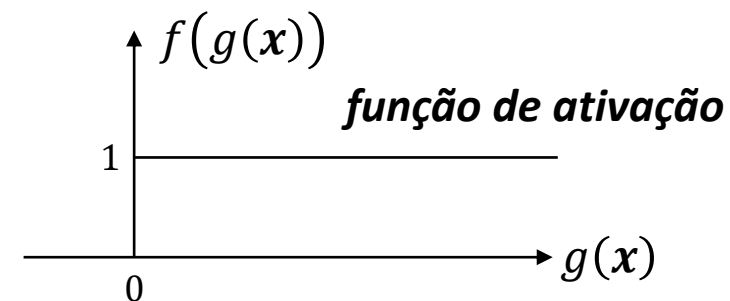
# Perceptron

- 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.
- Porém, assim como no modelo de M-P, a **função de ativação** utilizada pelo **perceptron** também é a **função degrau** com a **diferença que aqui ela não mais depende do limiar de ativação,  $\theta$** .
- Ou seja, a **transição** ou **ativação** sempre ocorre quando  $g(x) = 0$ .



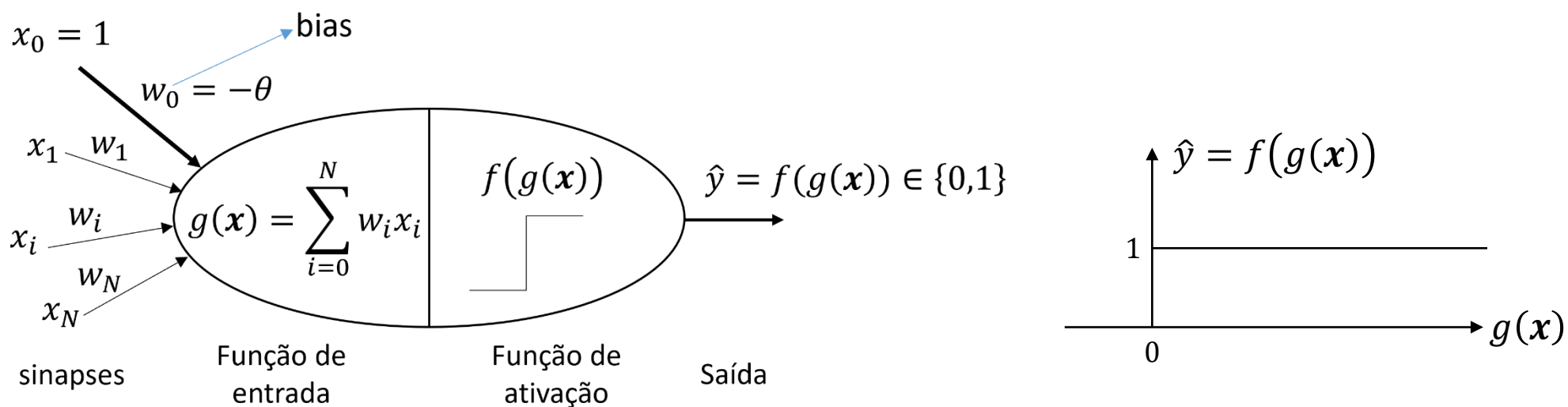
$$\hat{y} = f(g(x)) = \begin{cases} 1, & \text{se } g(x) \geq 0 \\ 0, & \text{se } g(x) < 0 \end{cases}$$

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



# Perceptron

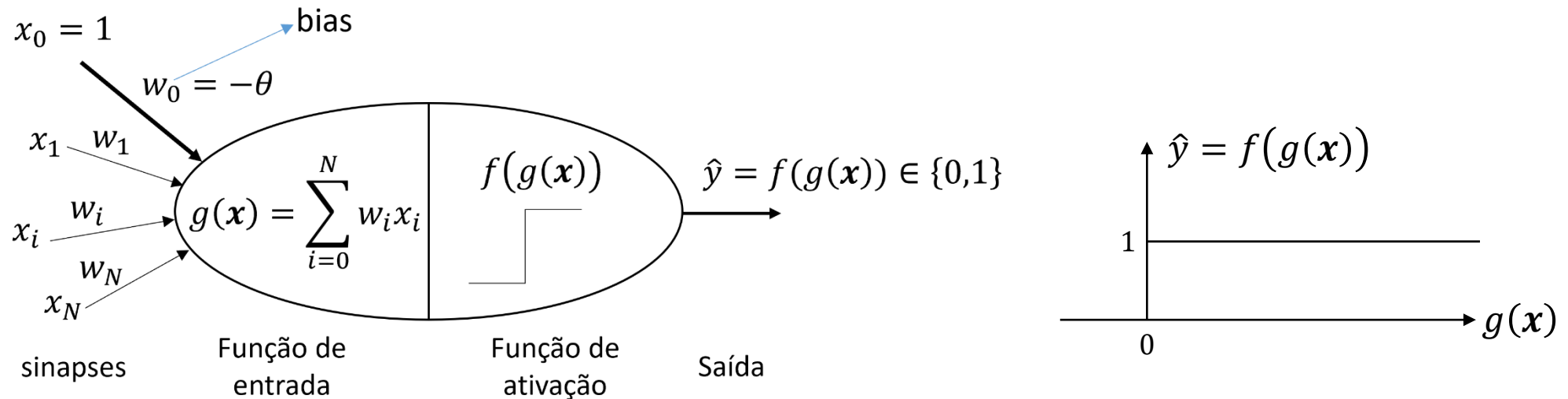
- A ativação do **perceptron** é causada pela **combinação linear** dos **estímulos de entrada** em relação aos **pesos sinápticos**.
- Se a **combinação linear** **exceder** o **limiar de ativação**,  $\theta$ , o **disparo ocorre**.
- Isso é expresso por uma **função de ativação** do tipo **degrau**.





# Perceptron

- Notem que a **função de ativação**,  $f(\cdot)$ , tem a transição para o valor 1 quando  $g(\mathbf{x}) = 0$ , ou seja, o **limiar de ativação não é variável**.
- **Mas então como podemos ter limiares de ativação diferentes para resolver diferentes tipos de problemas de classificação?**



# Perceptron

- No caso do perceptron, o *limiar de ativação é controlado pelo valor do peso de bias*,  $w_0$ , fazendo com que a transição sempre seja em  $g(\mathbf{x}) = 0$ .
- Ao incorporar o *limiar de ativação* à combinação linear,  $g(\mathbf{x})$ , podemos usar uma *função de ativação com transição fixa em zero*, pois, agora, ajusta-se o *limiar de ativação* através do peso  $w_0$ .
- O peso  $w_0$  (i.e., o limiar de ativação,  $\theta$ ) é incorporado ao modelo para que seu valor seja aprendido durante o treinamento junto com os outros pesos.

# Perceptron

- Para que tenhamos a saída do perceptron,  $\hat{y}$ , igual a 1, então

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^N w_i x_i \geq 0.$$

- Portanto, reescrevendo a equação acima, temos

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

para que haja a ativação do perceptron.

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

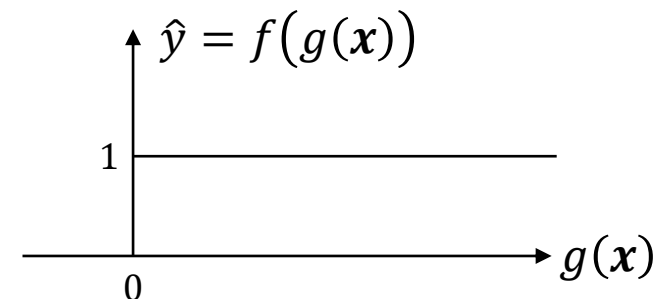
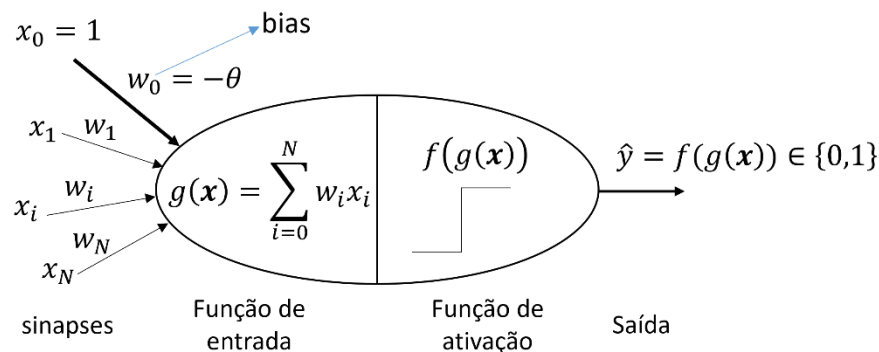
# Perceptron

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

$$g(\mathbf{x}) = \sum_{i=0}^N w_i x_i,$$

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

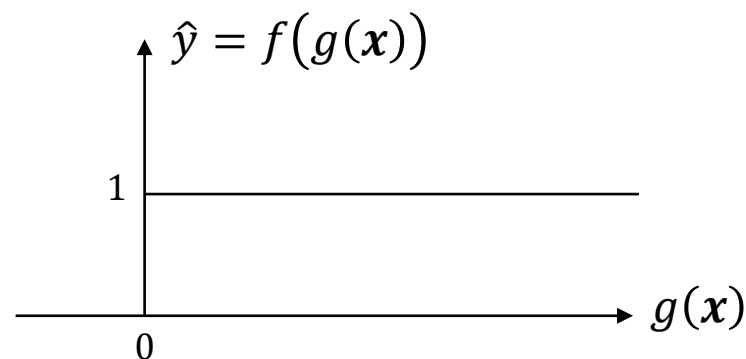
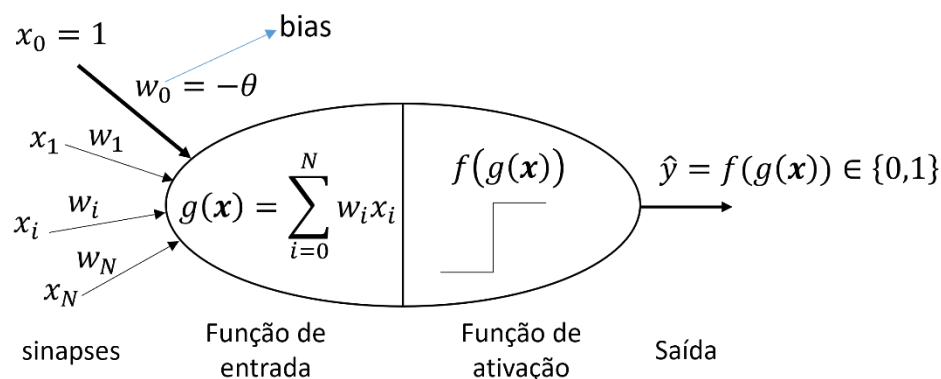
- Portanto, como já sabemos, este tipo de função dá origem a um **classificador binário** onde as **classes são separadas** por uma **superfície de separação linear**.



***Como os pesos são aprendidos?***

# 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, podemos usar 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 e somente 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

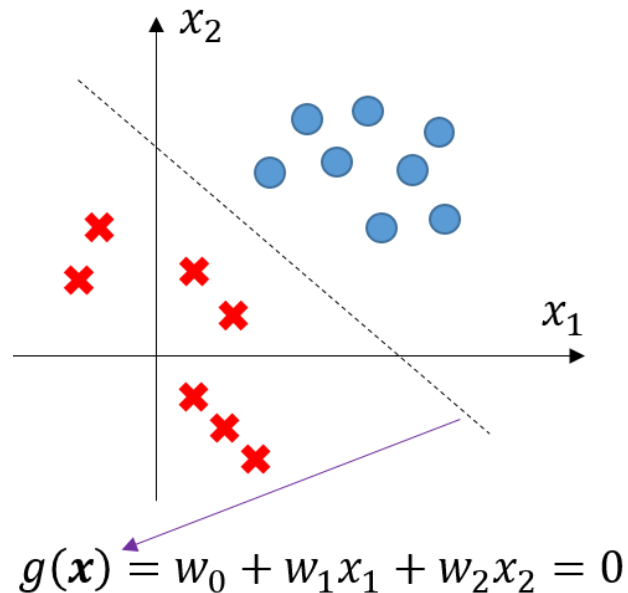
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - \hat{y})\mathbf{x},$$

onde  $\mathbf{w}$  é o vetor de pesos,  $\alpha$  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.

- Percebam que a equação acima é idêntica a da atualização do gradiente descendente estocástico, mas ela só tem 3 possibilidades.



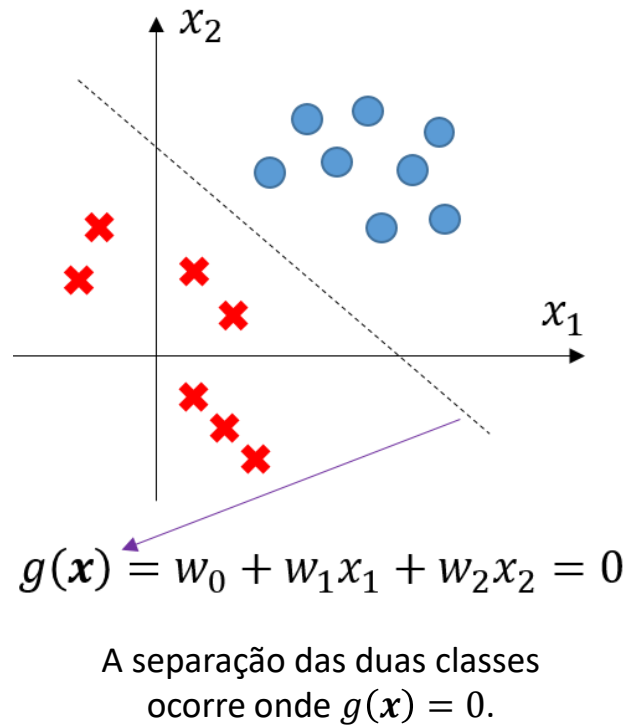
# Perceptron



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

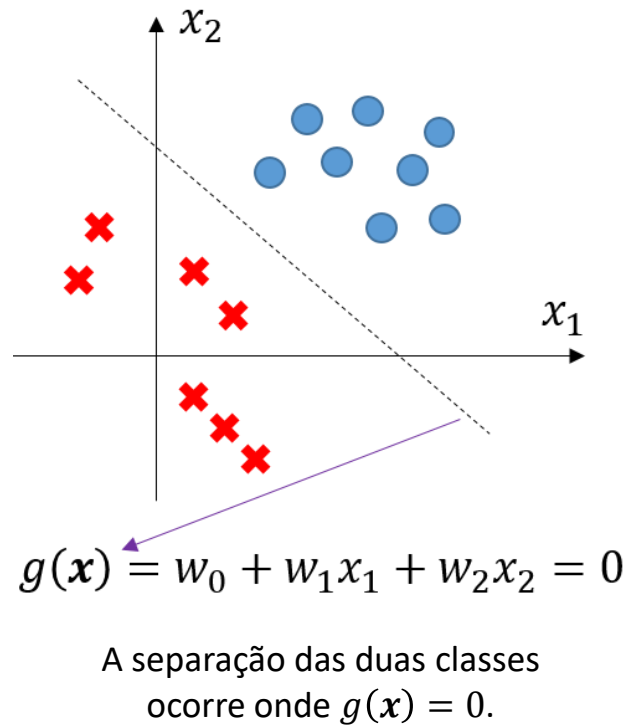
- Como percebemos, o **perceptron** é **idêntico** ao **classificador binário com limiar de decisão rígido**.
- Por definição, o **perceptron** sempre utiliza **superfícies de separação lineares**, ou seja, sempre teremos  $g(\mathbf{x})$  como sendo a equação de um **hiperplano**.

# Perceptron



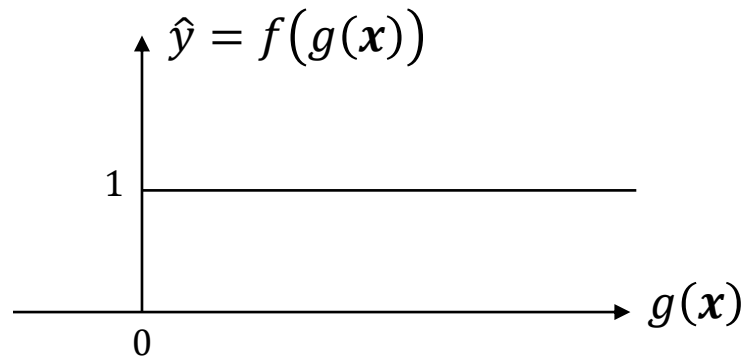
- Portanto, teoricamente, **sem transformação dos atributos**, um **único perceptron** só é capaz de **classificar** dados que sejam **linearmente separáveis** (ou seja, separáveis por um **hiperplano**).
- A figura ao lado ilustra isso para um caso bidimensional.
- A separação das duas classes ocorre onde  $g(\mathbf{x}) = 0$ .

# Perceptron



- 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.

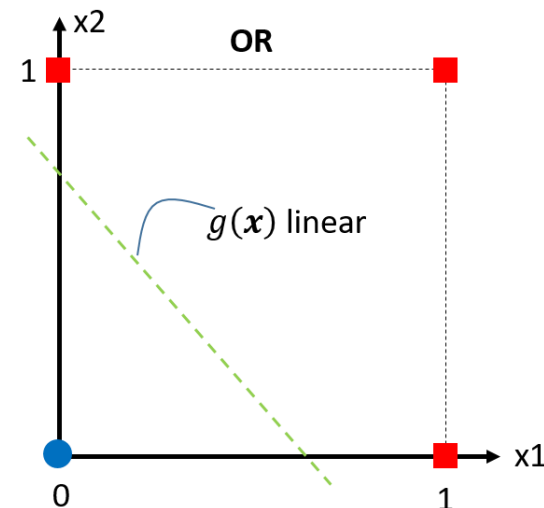
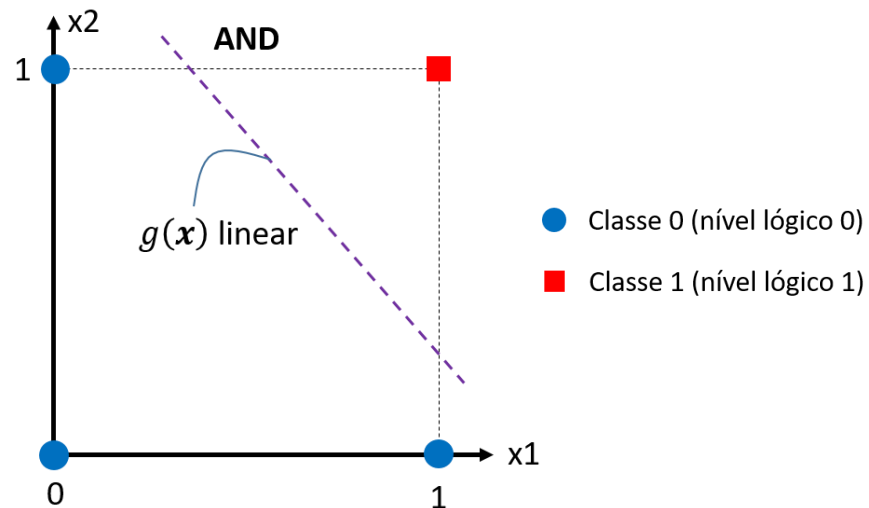
# Perceptron



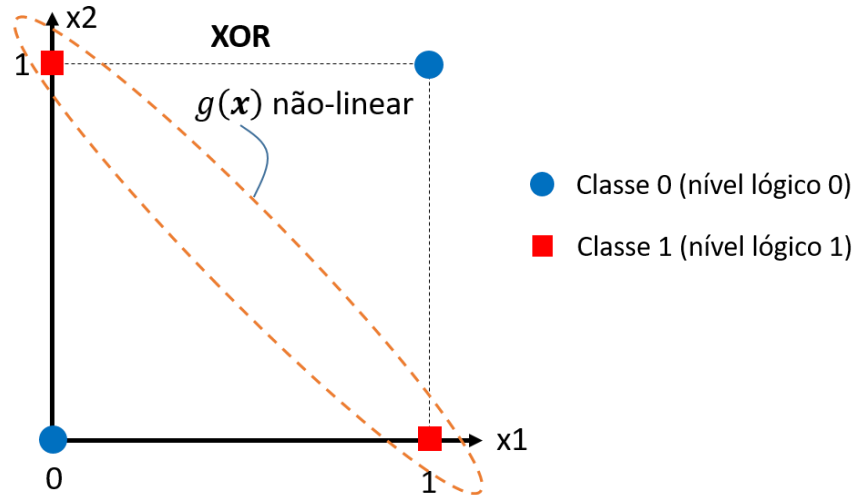
- Observem que, ao contrário do **regressor logístico**, o **perceptron não produz em sua saída a probabilidade da classe positiva**, em vez disso, ele faz **predições rígidas**, i.e., **confiantes**, em uma das duas classes.
- Essa é uma das razões para se preferir a **regressão logística** ao invés do **perceptron**.
- Além disso, para se **treinar perceptrons em conjunto** (i.e., uma **rede neural**), é necessário que as funções de ativação dos perceptrons sejam deriváveis.

# Exemplo: Perceptron com SciKit-Learn

- 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 e o problema da lógica XOR



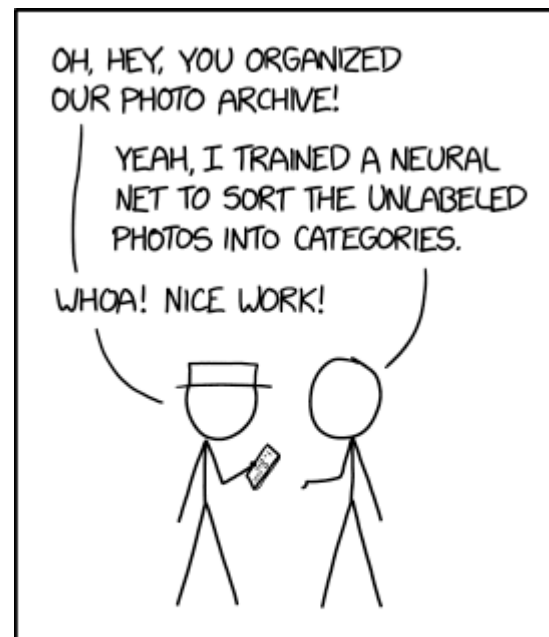
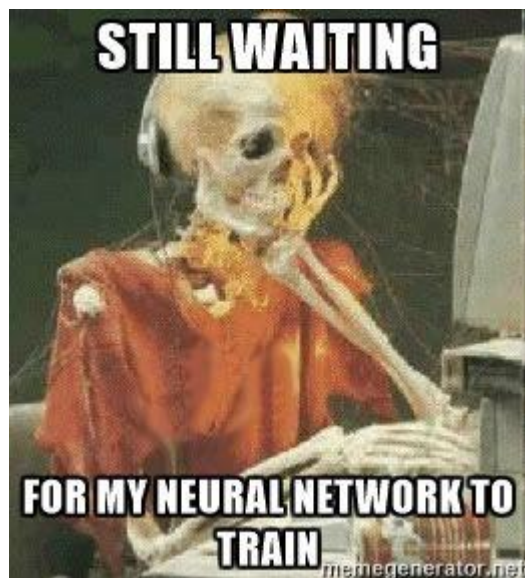
- Porém, a lógica XOR **não é linearmente separável** e necessita de uma **superfície de separação não-linear**.
- Vejam na figura ao lado que são necessárias **no mínimo duas retas paralelas**.
- **A separação da lógica XOR pode ser obtida combinando-se o resultado alguns perceptrons**, o que resultará em uma superfície de separação não linear.
- Qual o número mínimo de perceptrons?

# Avisos

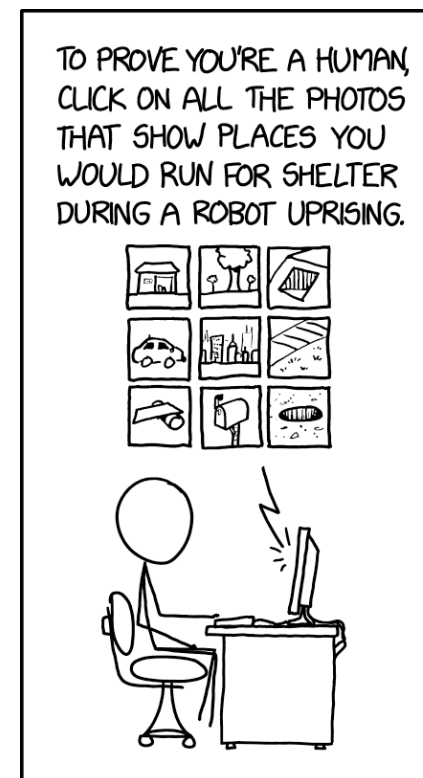
- Vocês já podem fazer os exercícios da lista #10.



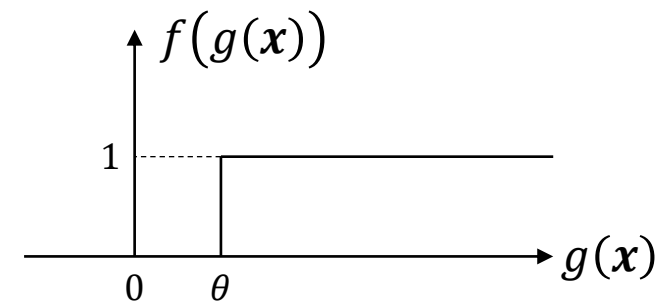
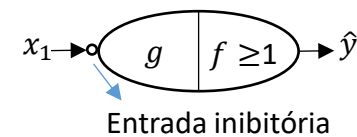
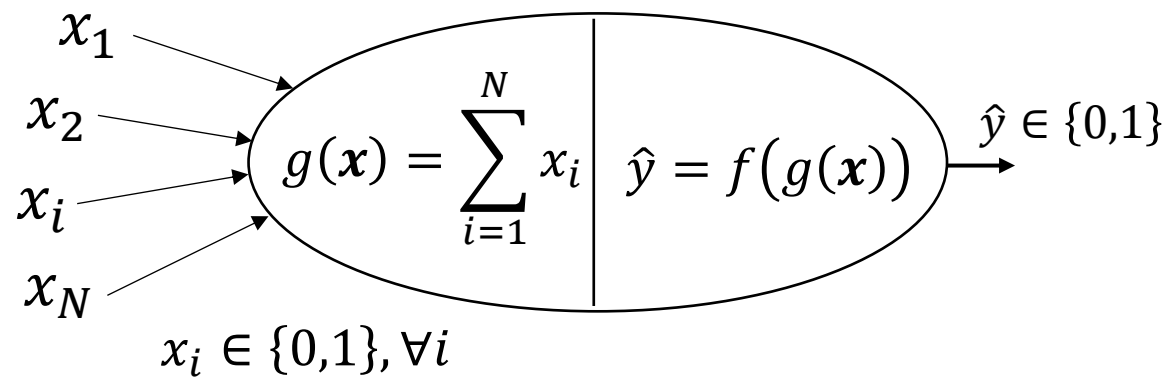
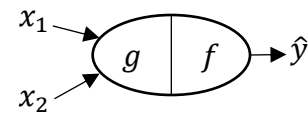
Obrigado!



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

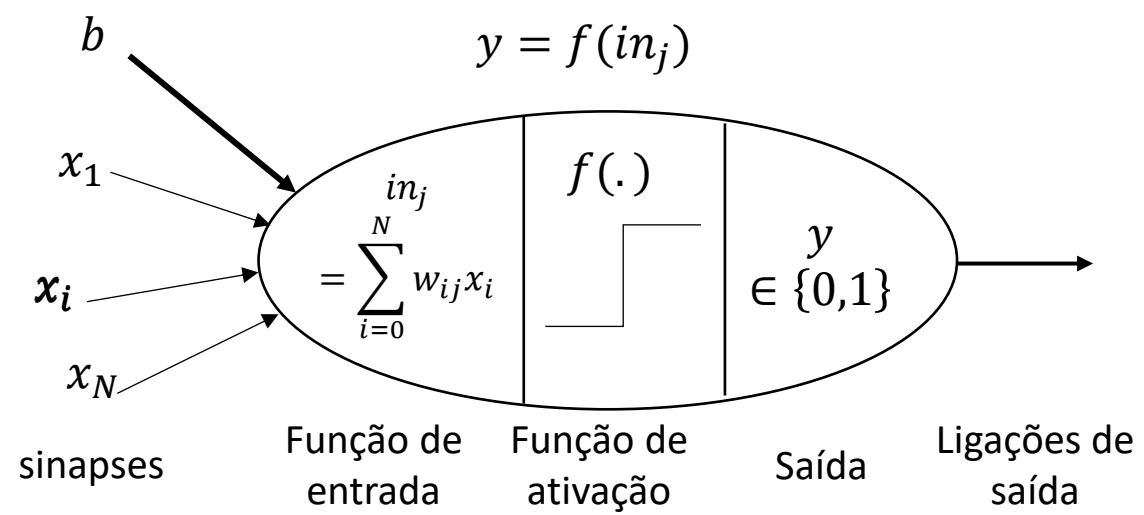


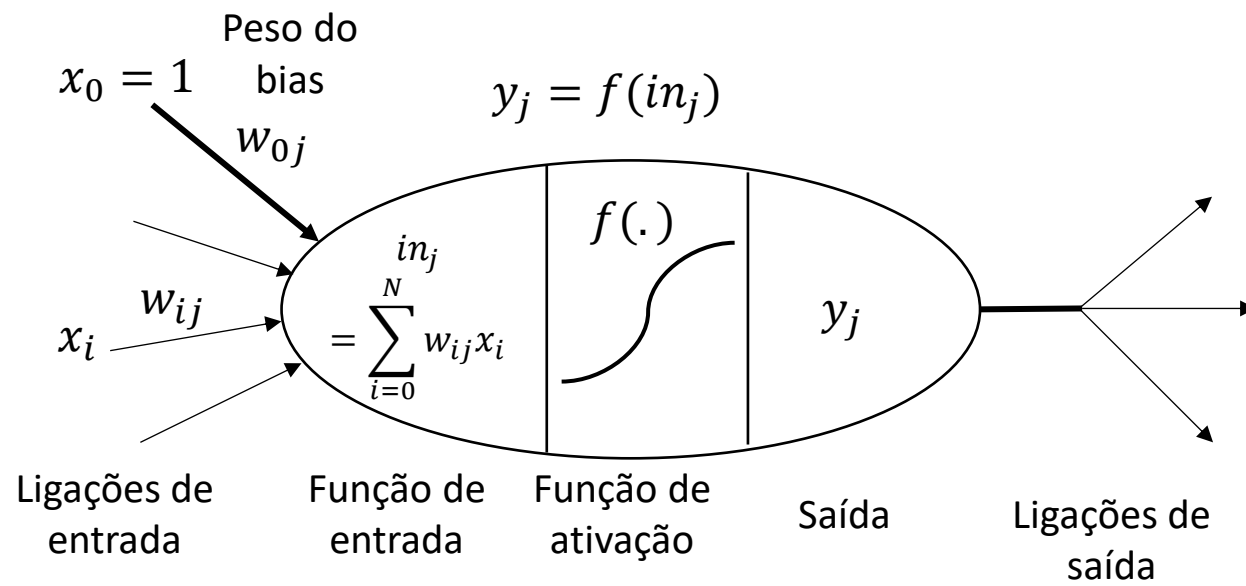
Figuras

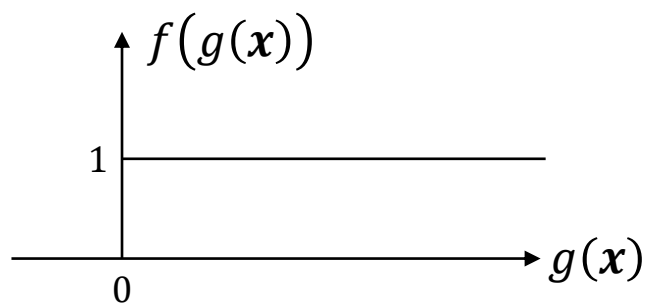
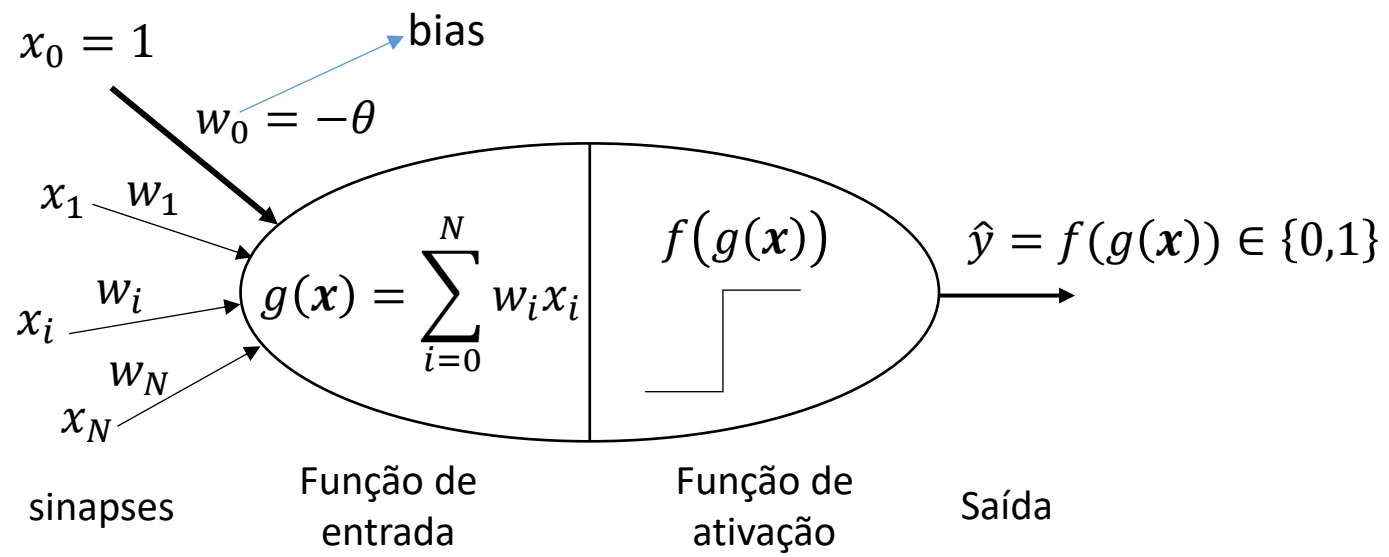


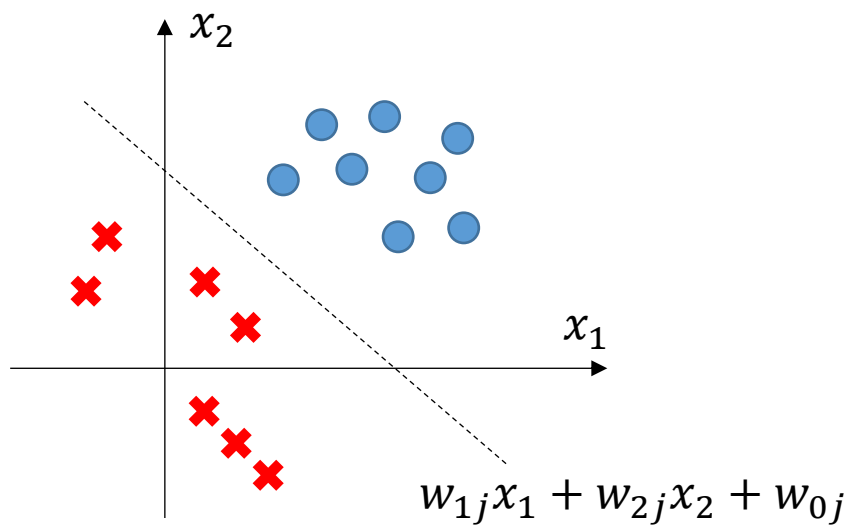
$$\hat{y} = f(g(\mathbf{x})) = \begin{cases} 1 & \text{se } g(\mathbf{x}) \geq \theta \\ 0 & \text{se } g(\mathbf{x}) < \theta \end{cases}$$

onde  $\theta$  é o limiar de decisão.

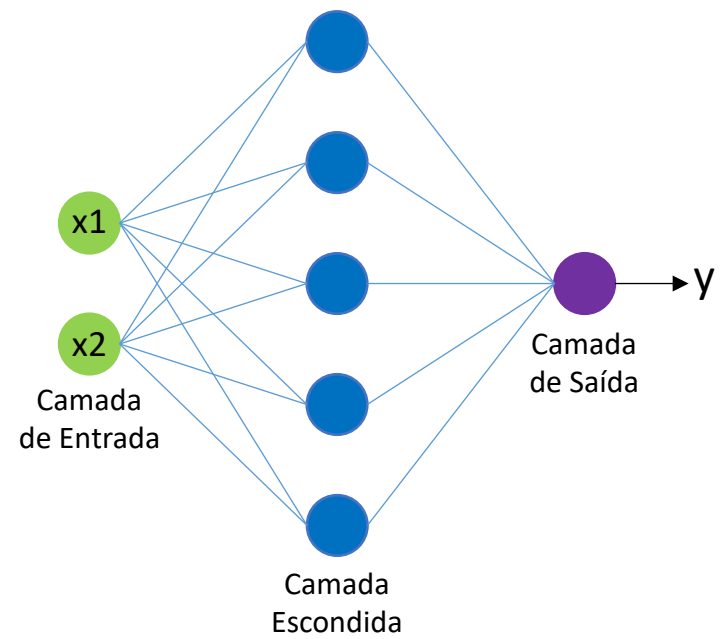


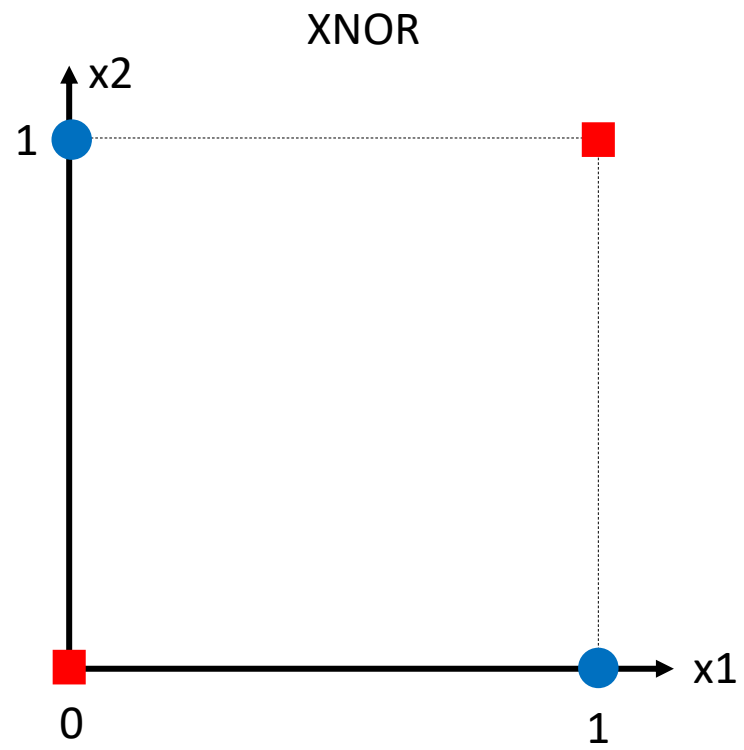












● Classe 0 (nível lógico 0)

■ Classe 1 (nível lógico 1)