Redes Neurais Artificiais

Curso Machine Learning e Data Science com Python

Redes Neurais Artificiais

- Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no sistema nervoso central que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões.
- O aprendizado de máquina (machine learning) é uma subárea da Inteligência Artificial dedicado ao desenvolvimento de algoritmos e técnicas que permitam o computador aprender algo, ou seja, aperfeiçoar o seu desempenho em alguma tarefa.
- Exemplos da aplicação de redes neurais artificiais:
  - Reconhecimento de voz
  - Reconhecimento de escrita manual

Redes Neurais Artificiais

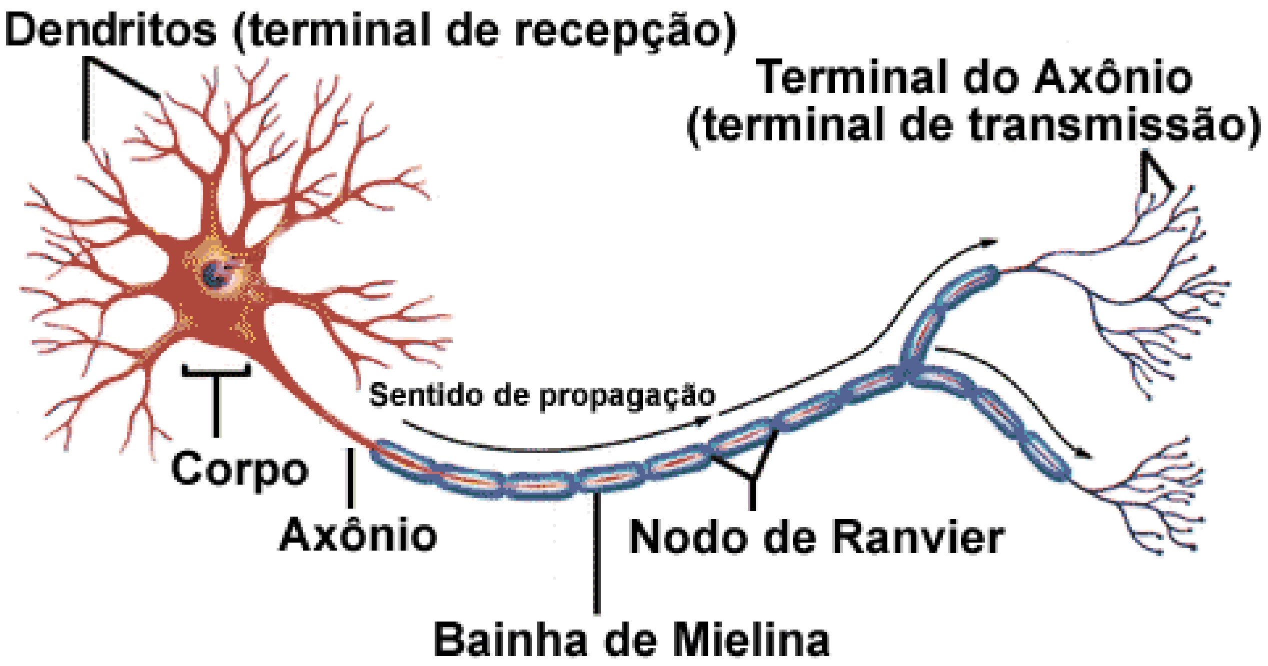
- O sistema nervoso é formado por um conjunto bastante complexo de neurônios.
- Nos neurônios, a comunicação é realizada através de impulsos.
- Quando um impulso é recebido, o neurônio processa e dispara um segundo impulso que produz uma substância neurotransmissora do qual flui do corpo celular para o axônio.
- Ao contrário das redes neurais artificiais, as redes neurais naturais não transmitem sinais negativos e não são uniformes.

Redes Neurais Artificiais

- Tem-se um grande desejo de se criar máquinas que operem independentemente do controle humano.
- Máquinas inspiradas na biologia podem coordenar diversos graus de liberdade durante a execução de tarefas complicadas sem que tenham que desenvolver um modelo matemático específico.
- Existem vários tipos de redes artificiais:
    - Rede Perceptron
    - Rede Adalaine

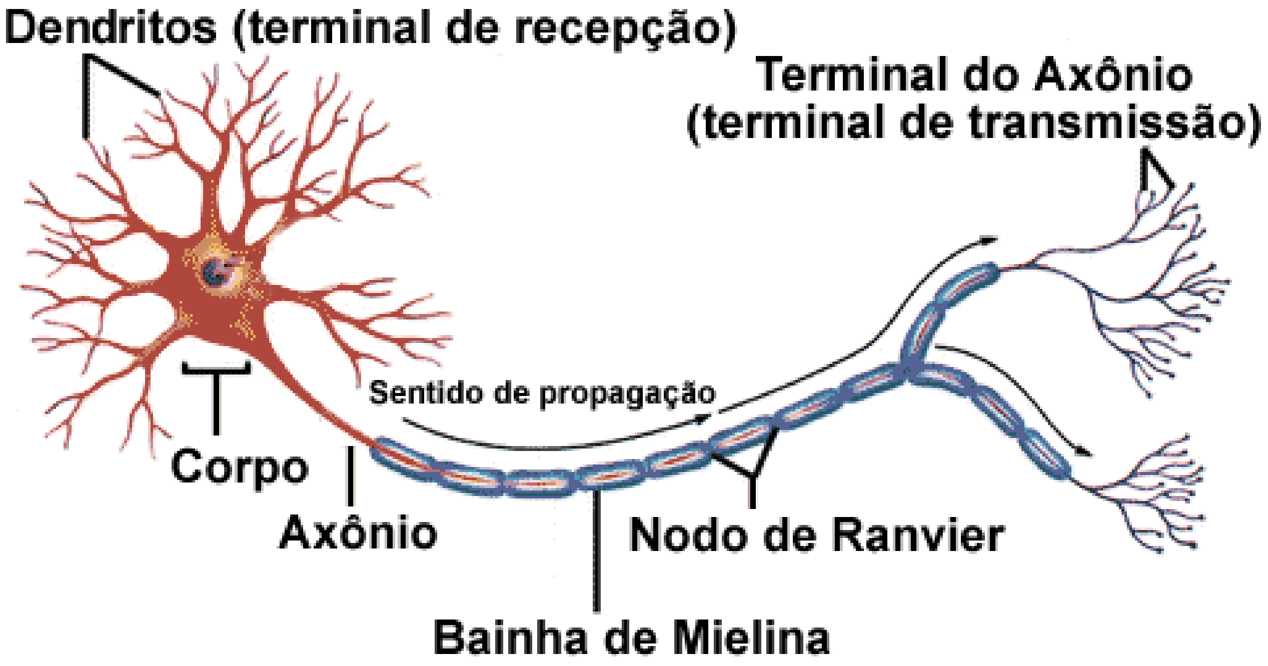
Redes Neurais Artificiais

- Redes neurais artificiais (assim como algoritmos genéticos) é mais uma abstração da biologia trazida para a computação.



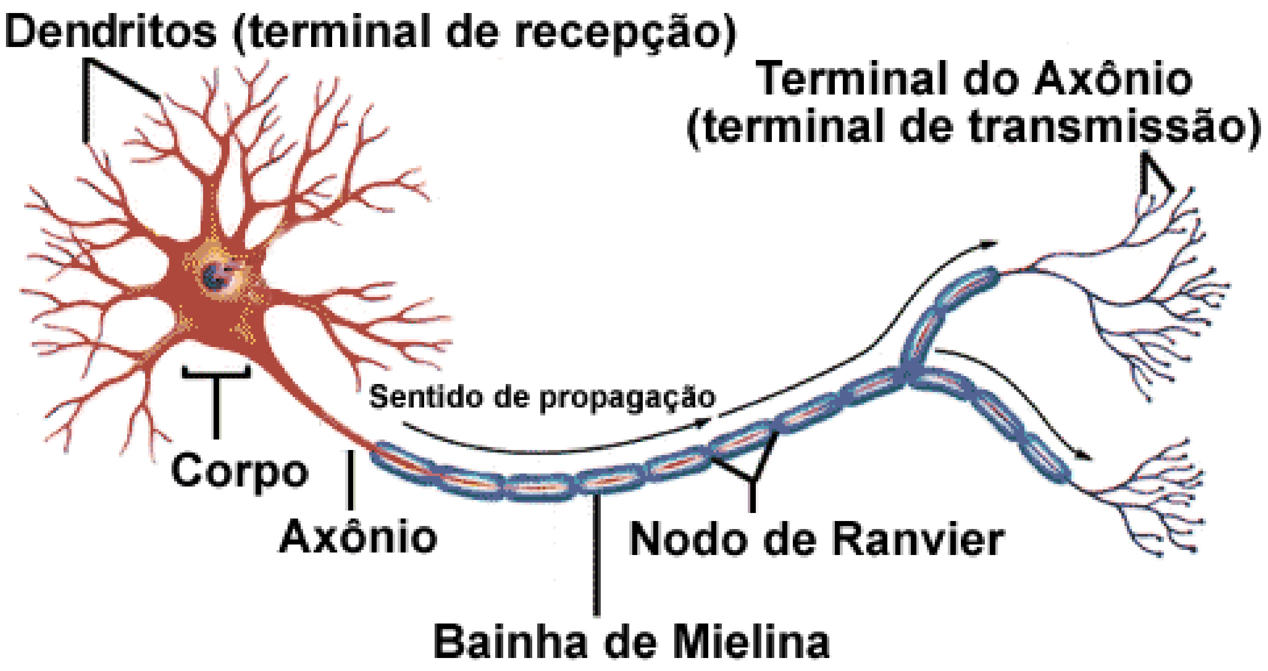
Redes Neurais Artificiais

- Os dendritos são responsáveis por receber os sinais.



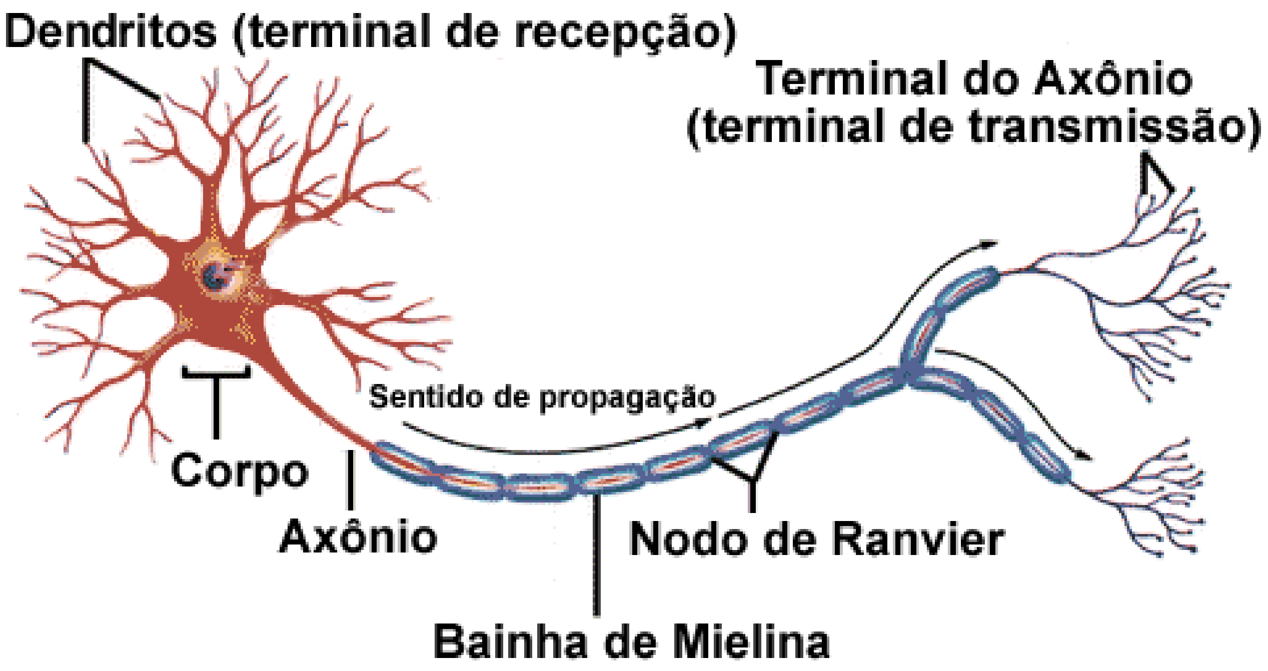
Redes Neurais Artificiais

- O axônio transmite os sinais para as terminações sinápticas e isso é repassado para outros neurônios.



Redes Neurais Artificiais

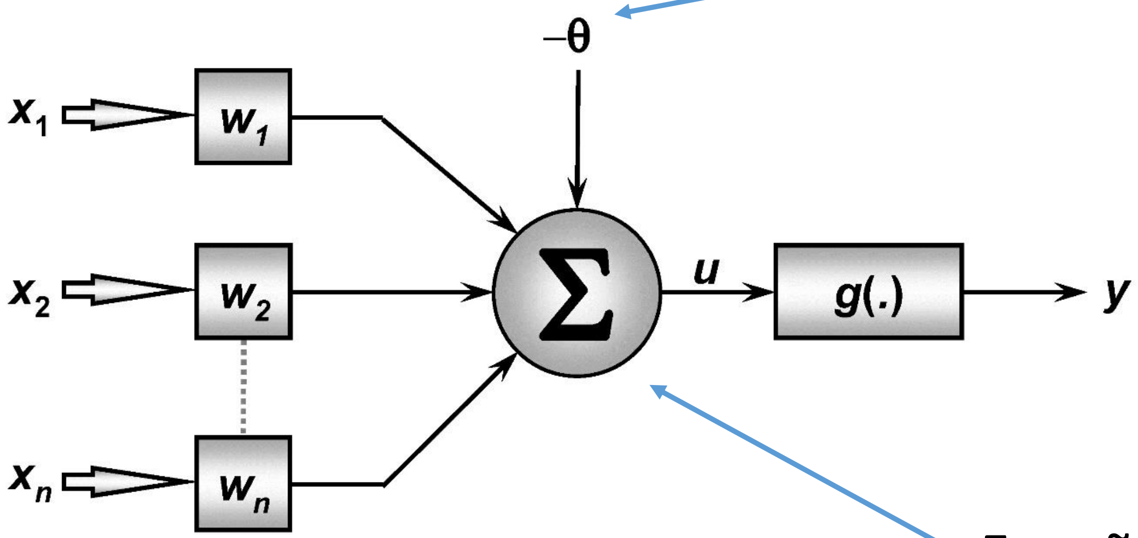
- Temos uma entrada e uma saída, pode-se ter várias entradas e várias saídas, isso varia de problema para problema.



Redes Neurais Artificiais

- Neurônio
  - Sinais de entrada {x1, x2, x3, ..., xn}
  - Pesos sinápticos {w1, w2, w3, ..., wn}

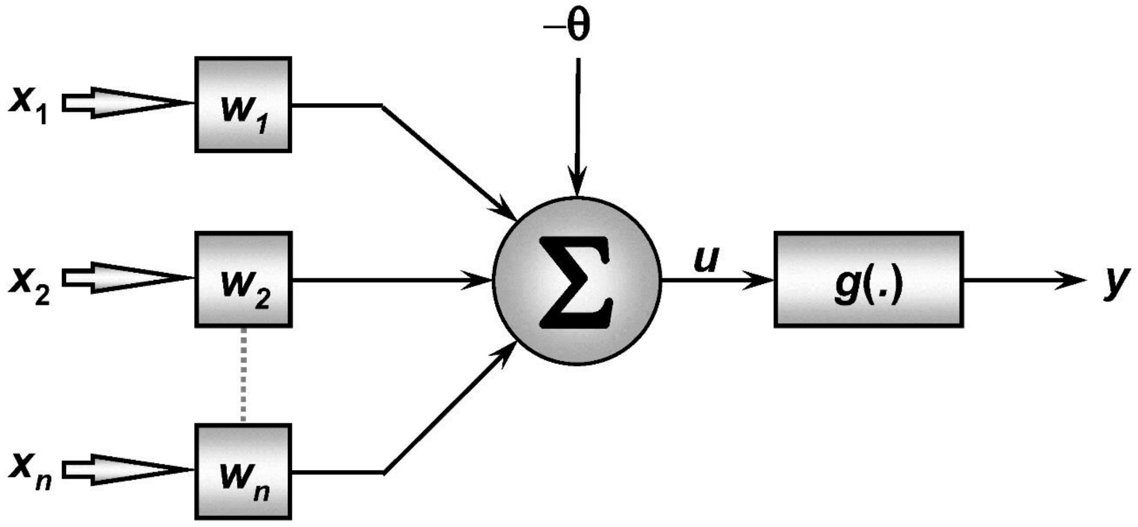
Limiar de ativação



Função agregadora

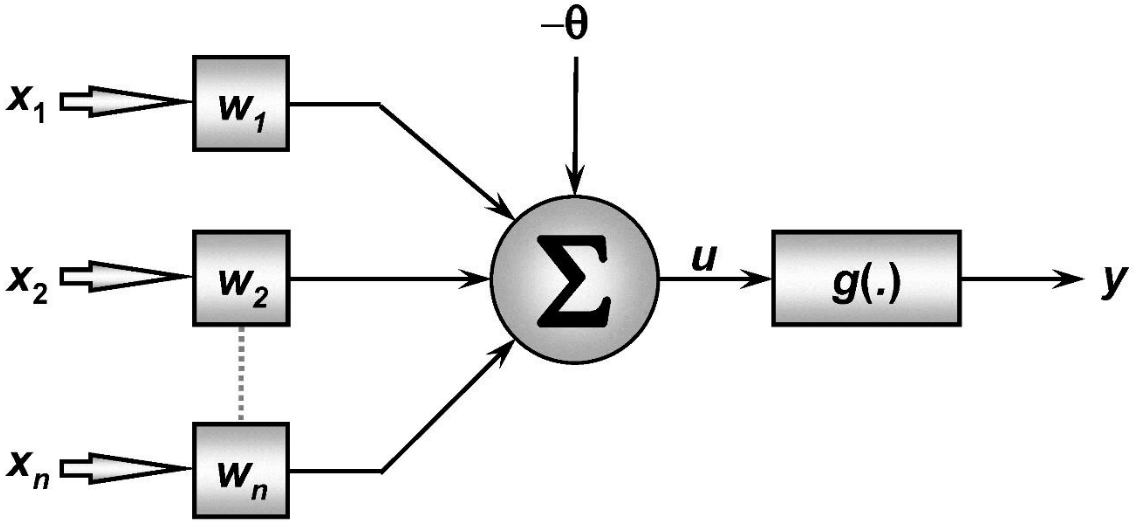
Redes Neurais Artificiais

- Os sinais de entrada {x1, x2, xn} são ponderados por {w1, w2, wn}.
- Existe um produto do sinal de entrada pelo peso que pondera.
- Isso é dado como entrada para o neurônio.



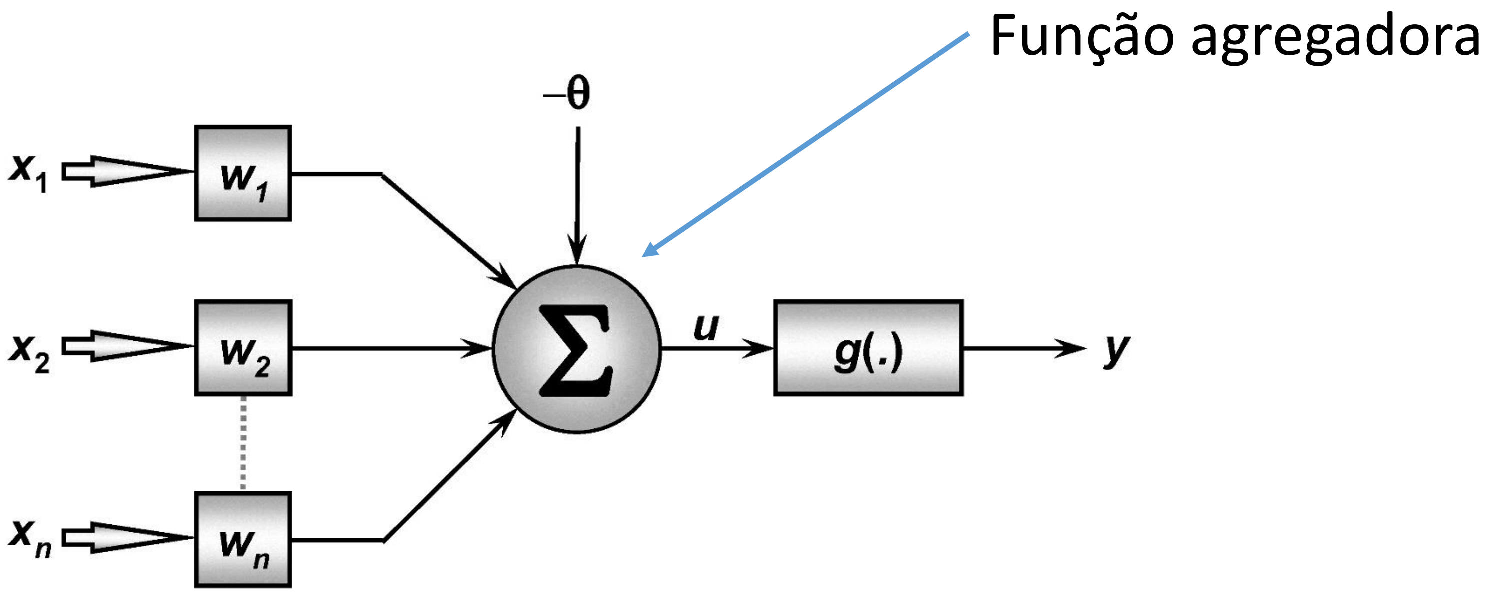
Redes Neurais Artificiais

- Os sinais são informações do problema.
- Os pesos sinápticos {w1, w2, wn} ponderam a entrada.
- Para cada uma das entradas, tem o peso para ponderar.



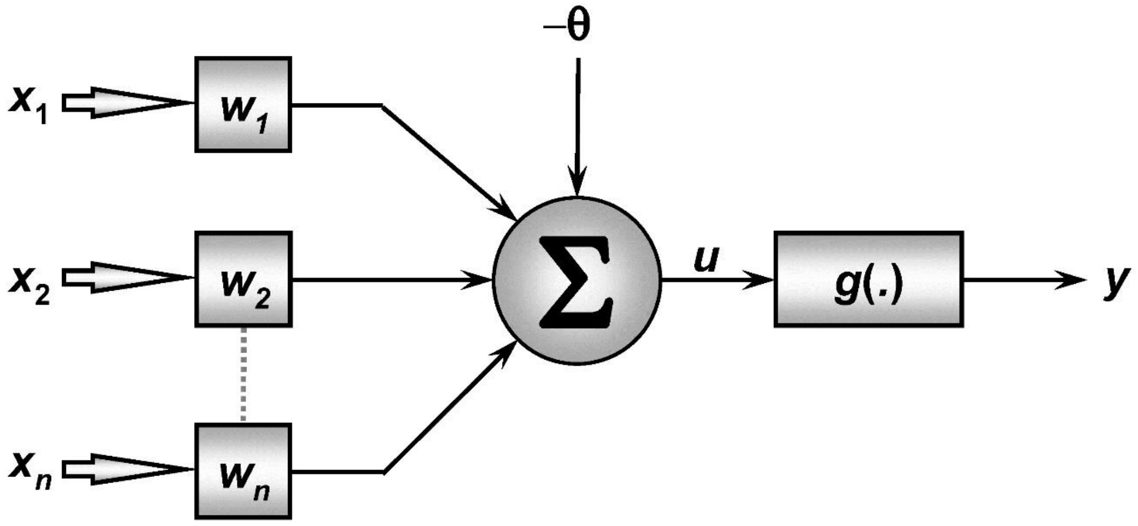
Redes Neurais Artificiais

- A função agregadora recebe todos os sinais e realiza a soma dos produtos dos sinais.



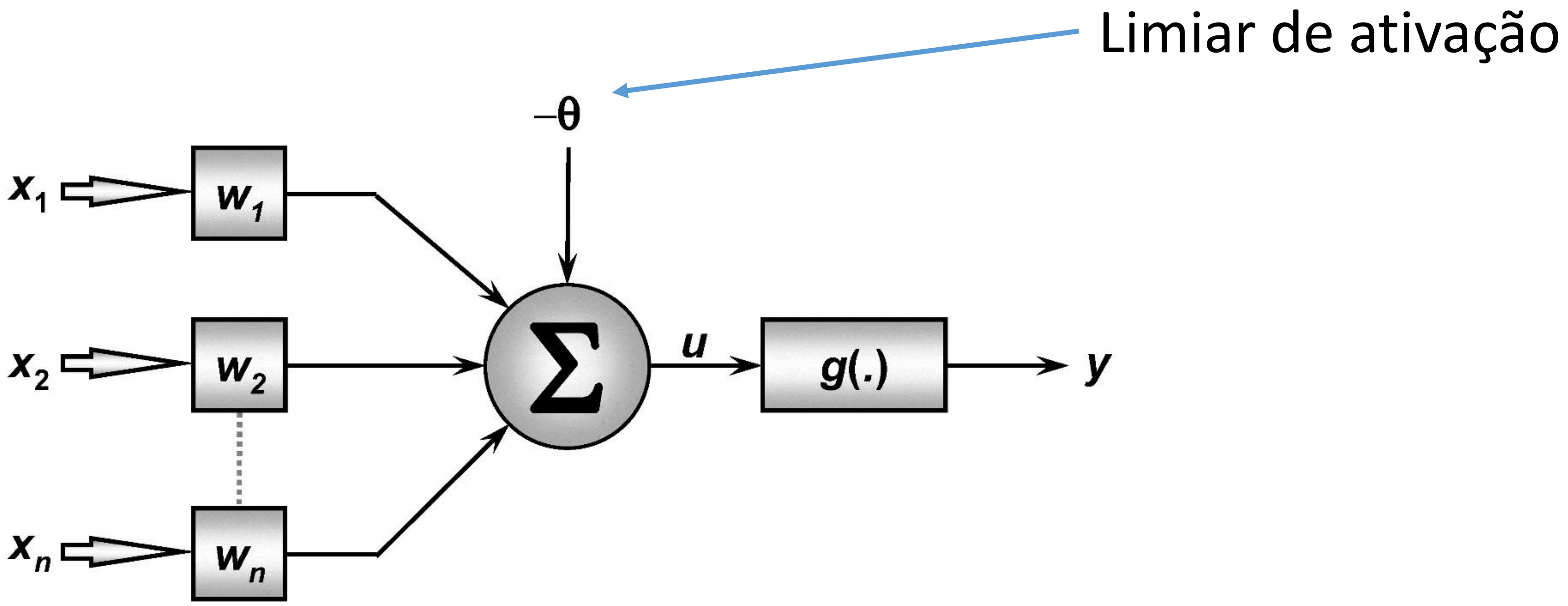
Redes Neurais Artificiais

- O neurônio deixa passar ou inibe um determinado sinal ou até mesmo altera a saída de acordo com a entrada.



Redes Neurais Artificiais

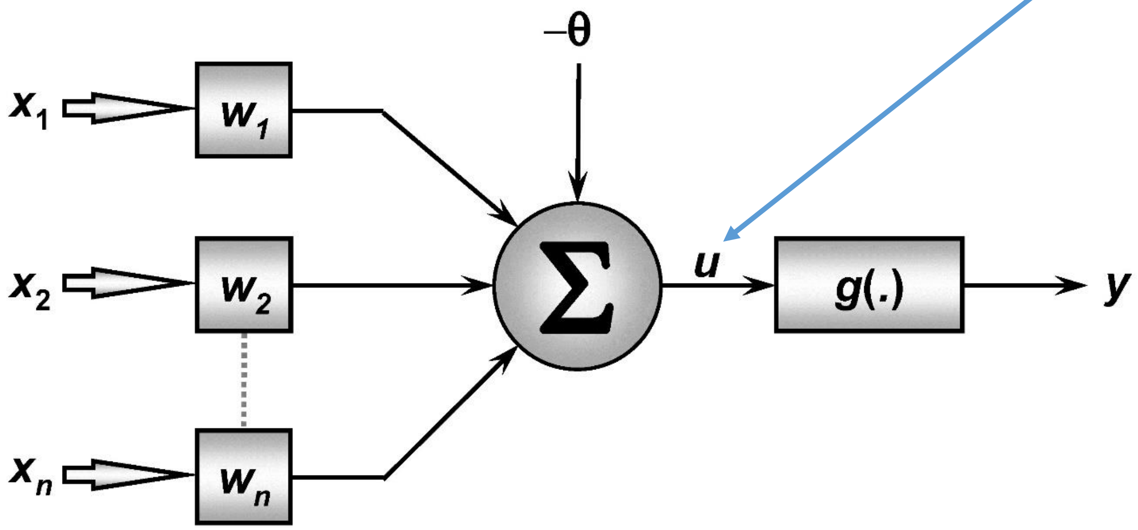
- O limiar é uma constante (geralmente é ponderada) que vai indicar um limiar para o sinal passar ou não.



Redes Neurais Artificiais

- “g” é a função de ativação. O valor “u” é dado como entrada para a função de ativação.

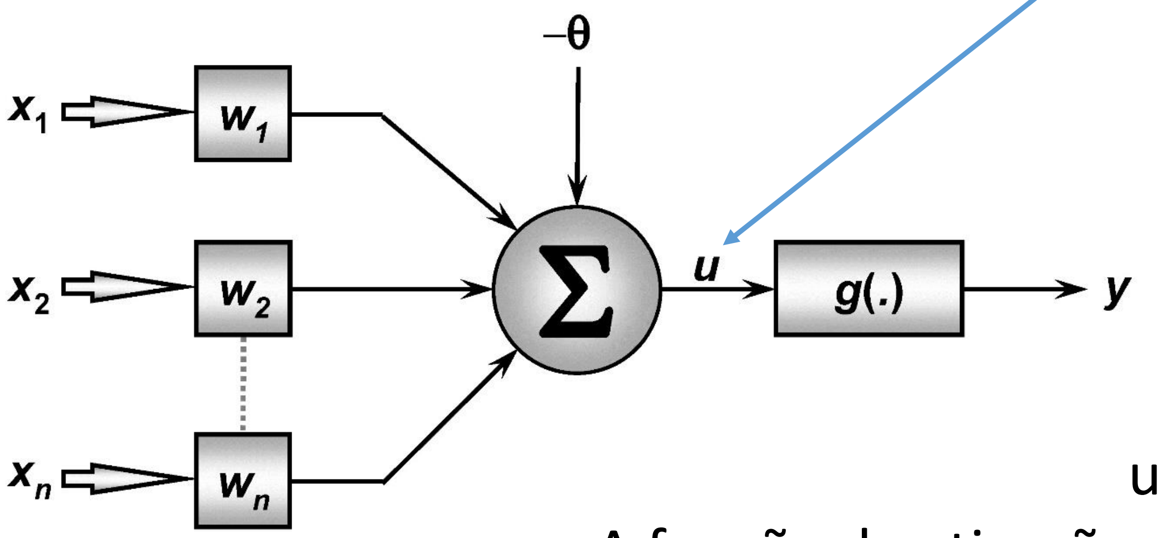
Potencial de ativação



Redes Neurais Artificiais

- Potencial de ativação: $u = \sum_{i=1}^{n} w_i * x_i - \theta$ Somatório do produto das entradas

Potencial de ativação



u = g(u)

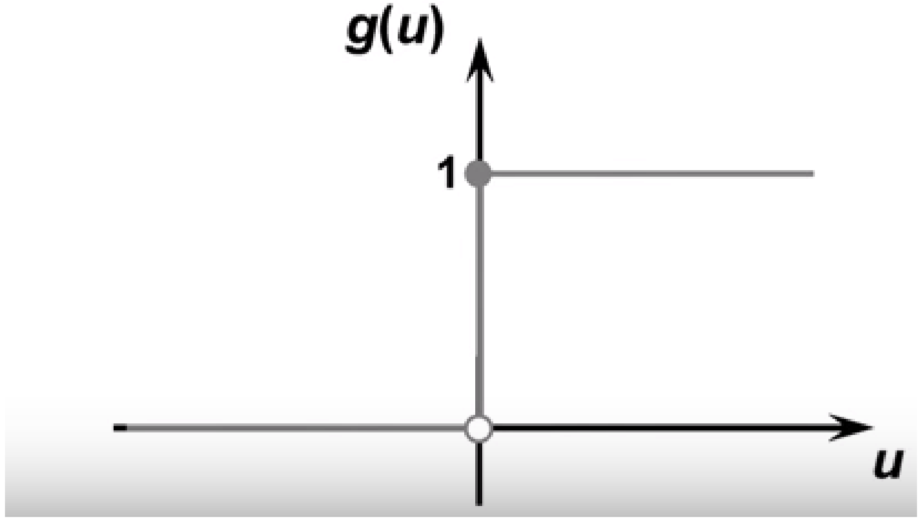
A função de ativação deixa o sinal passar ou não

Redes Neurais Artificiais

- Existem algumas funções de ativação:
  - Função degrau

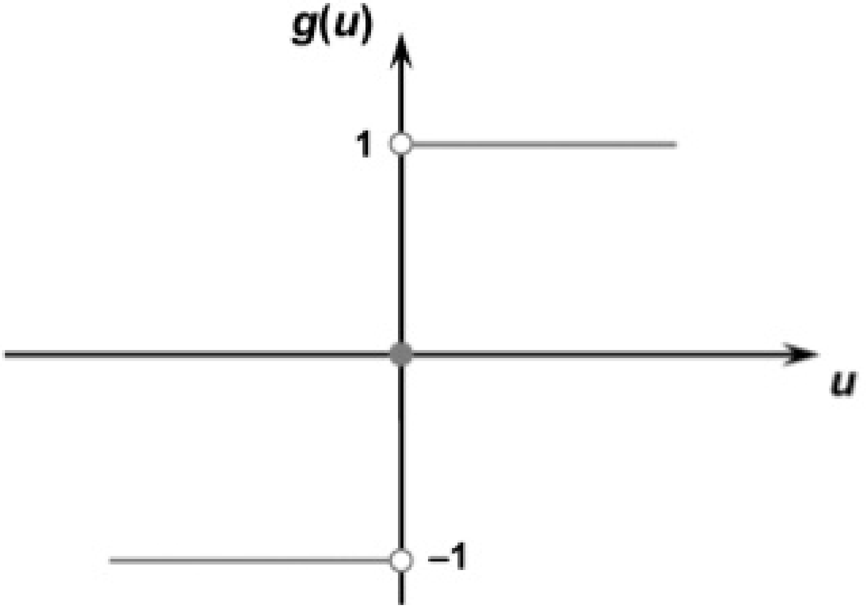
$$g(u) = \begin{cases} 1 & se\ (u \geq 0) \\ 0 & se\ (u < 0) \end{cases}$$

1 significa que houve ativação
0 significa que não houve ativação



Redes Neurais Artificiais

- Função degrau bipolar



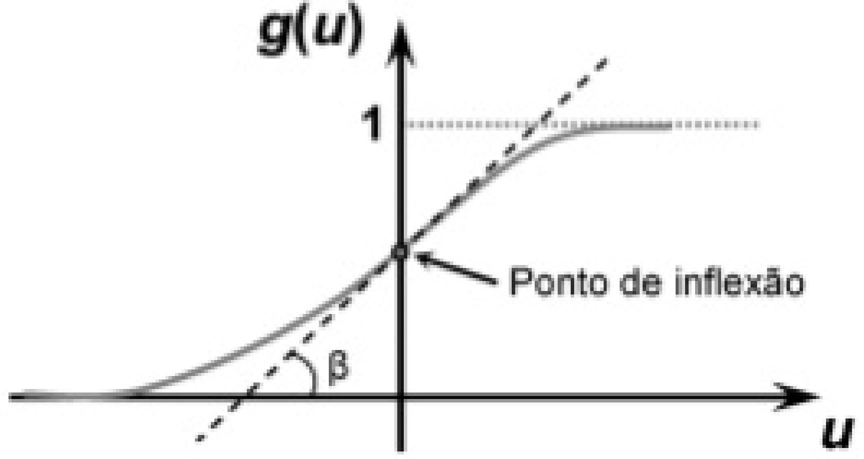
$$g(u) = \begin{cases} 1 & se\ (u > 0) \\ 0 & se\ (u = 0) \\ -1 & se\ (u < 0) \end{cases}$$

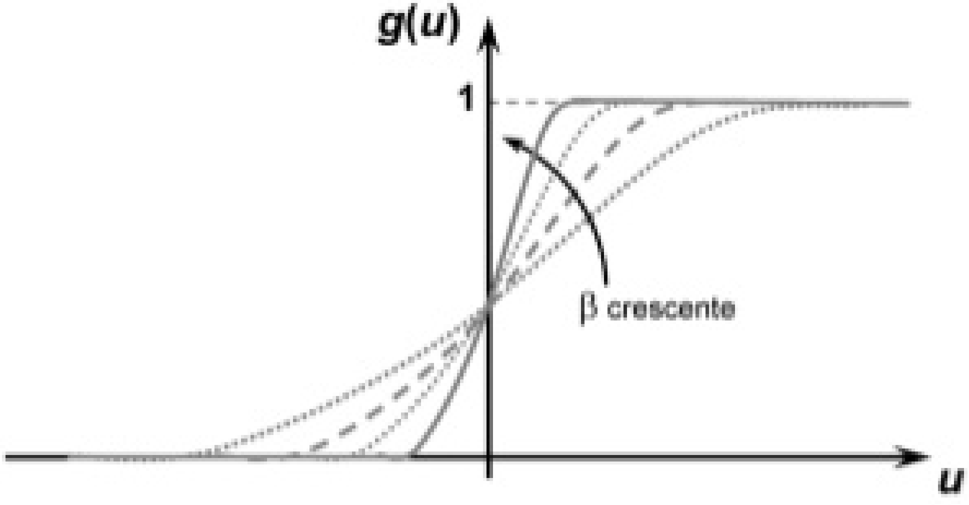
$$g(u) = \begin{cases} 1 & se\ (u \geq 0) \\ -1 & se\ (u < 0) \end{cases}$$

Redes Neurais Artificiais

- Função logística $g(u) = \dfrac{1}{1 + e^{-\beta u}}$

$e = 2{,}718281 \Rightarrow (Número\ de\ Euler)$
$\beta = constante\ de\ inclinação$

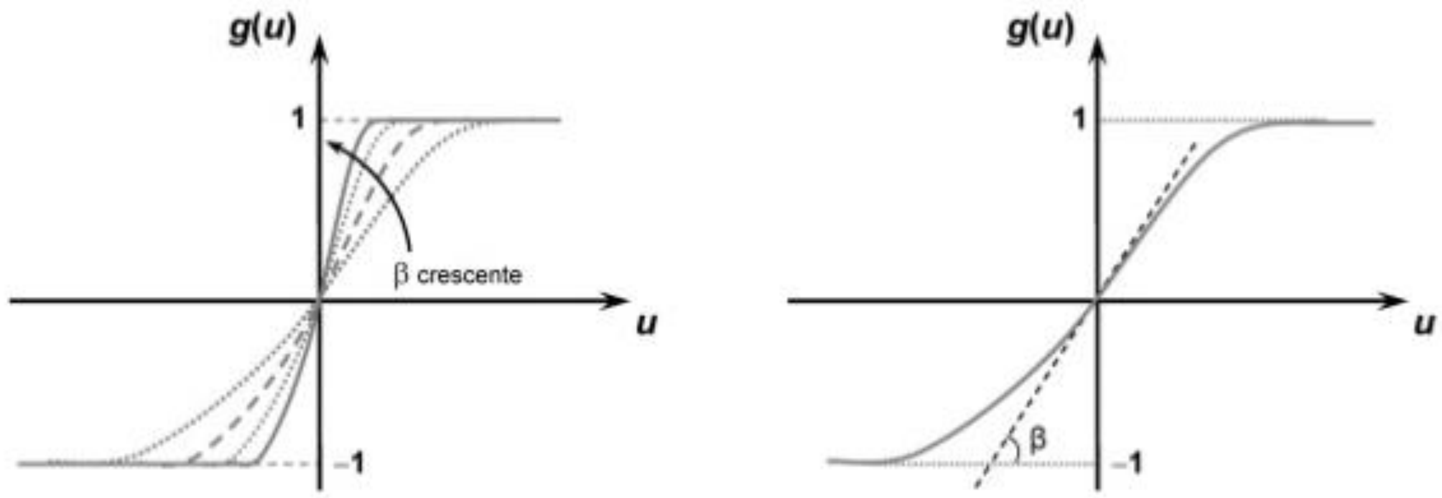




Os limites de saída são de 0 a 1.

Redes Neurais Artificiais

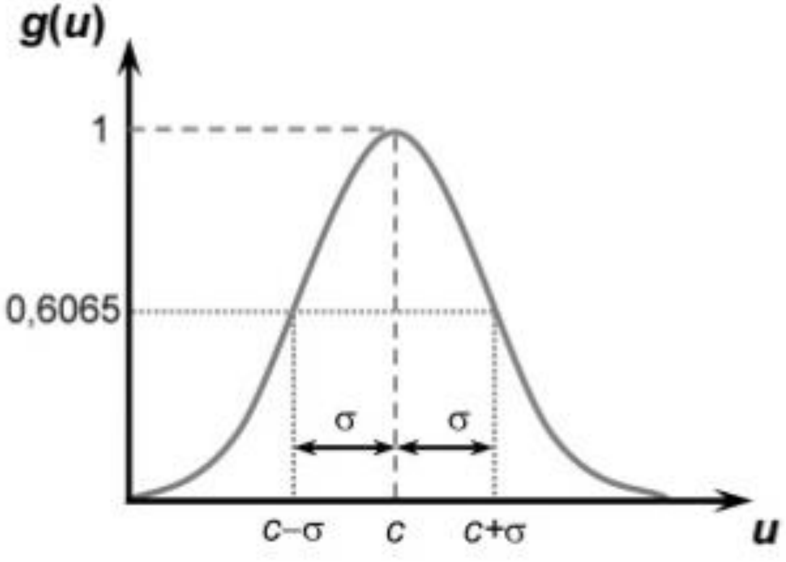
- Função tangente hiperbólica $$g(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}}$$



Os limites de saída são de -1 a 1.

Redes Neurais Artificiais

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



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

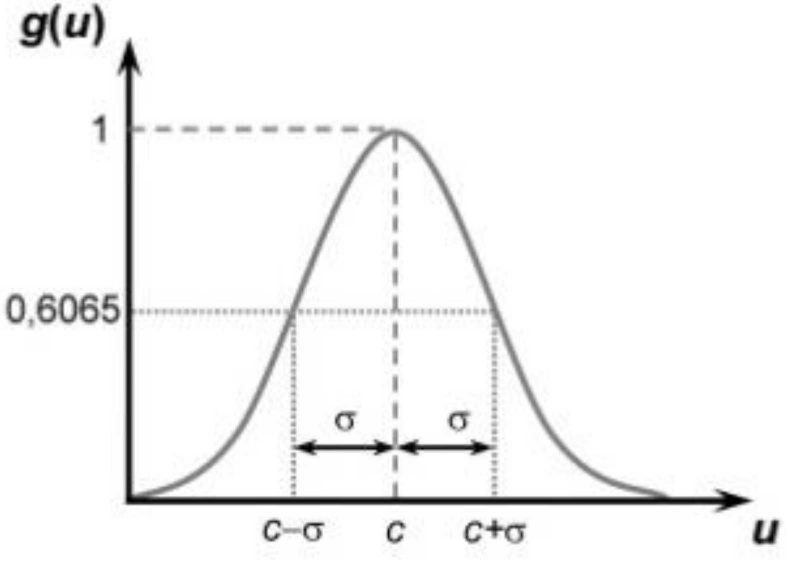
$\sigma \Rightarrow Desvio\ Padrão$

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

Muito utilizada em estatística e reconhecimento de padrões.

Redes Neurais Artificiais

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

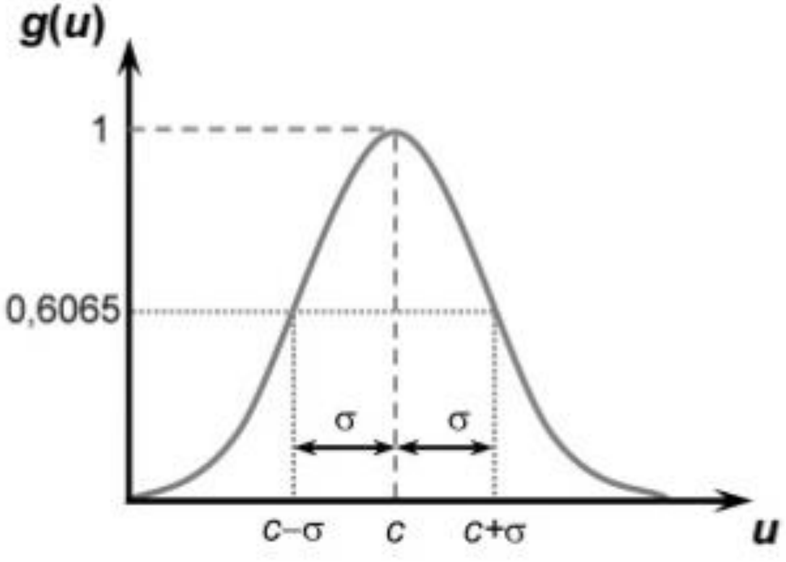


$e = 2{,}718281 \Rightarrow Número\ de\ Euler$
$\sigma \Rightarrow Desvio\ Padrão$
$c \Rightarrow Centro\ da\ Função\ Gaussiana$

A ideia é concentrar as entradas no intervalo do meio.

Redes Neurais Artificiais

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

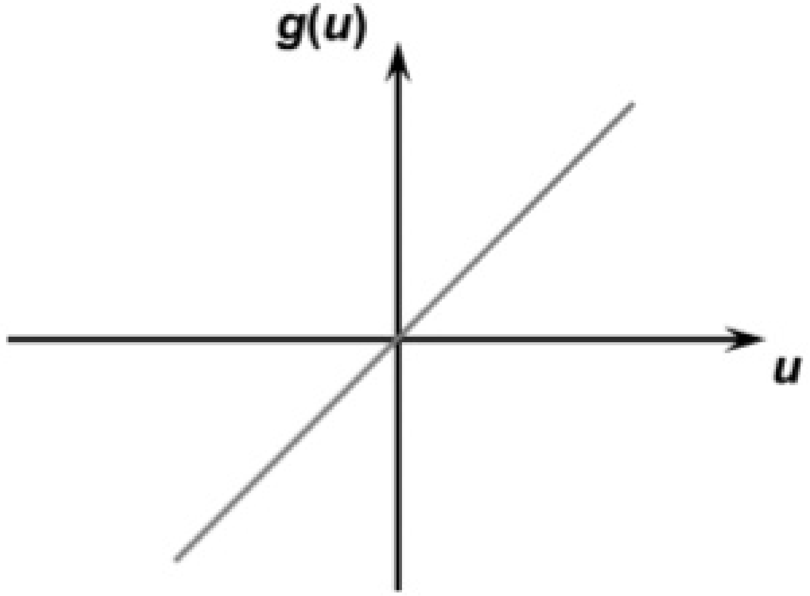


$e = 2{,}718281 \Rightarrow \textit{Número de Euler}$
$\sigma \Rightarrow \textit{Desvio Padrão}$
$c \Rightarrow \textit{Centro da Função Gaussiana}$

Os sinais fora do desvio padrão vão tender a 0.

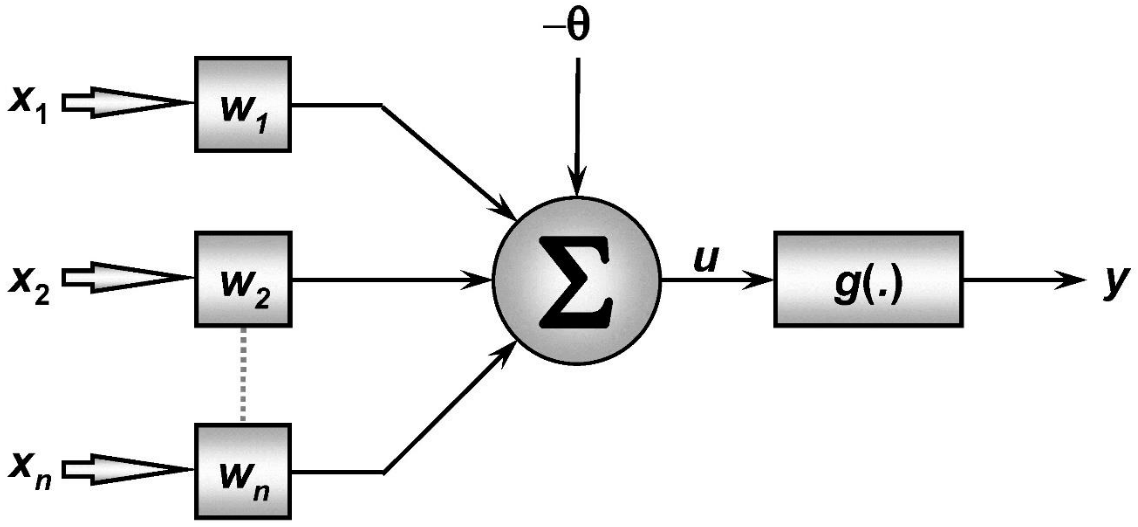
Redes Neurais Artificiais

- Função linear g(u) = u



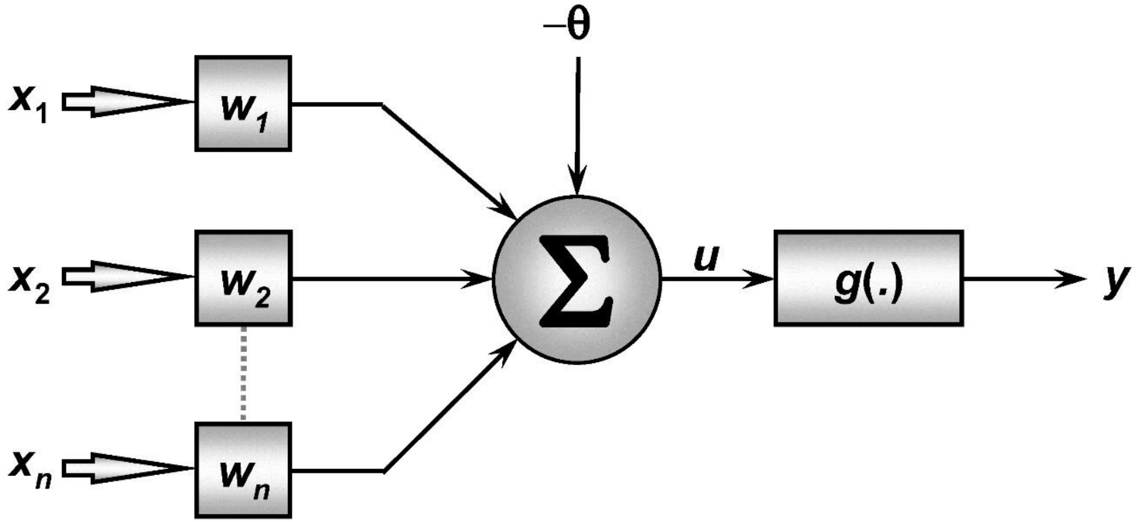
Rede Perceptron

- Rede muito simples.
- Possui apenas um neurônio.



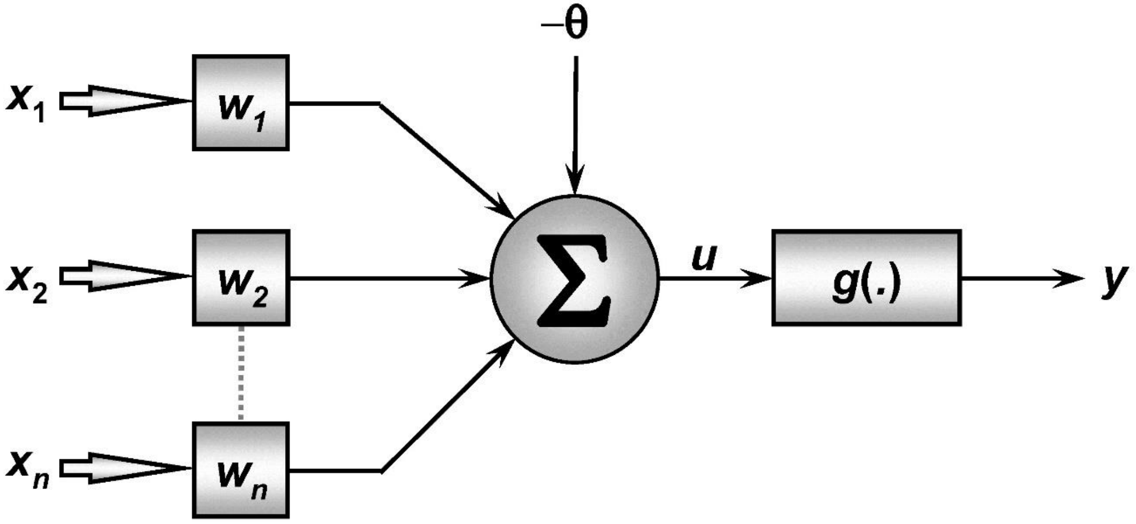
Rede Perceptron

- Se o nosso “g” é a função degrau bipolar, então a saída será 1 ou -1.
- Será 1 se o somatório dos produtos for >= 0.
- Será -1 se o somatório dos produtos for < 0.



Rede Perceptron

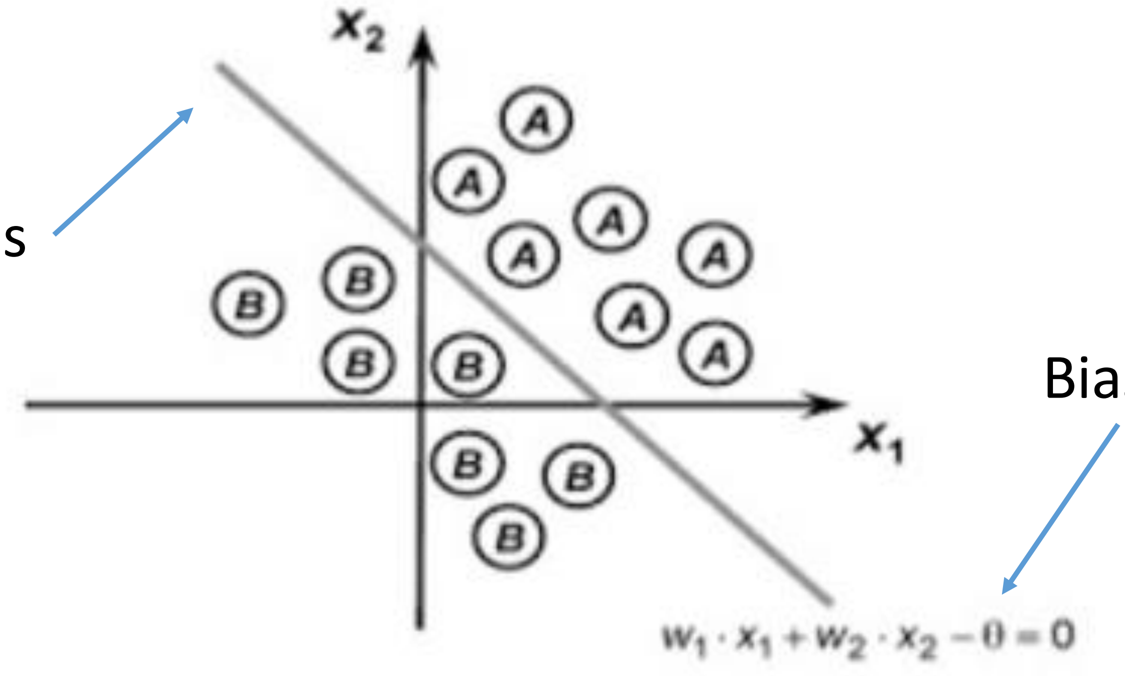
$$y = \begin{cases} 1, se \left(\sum_{i=1}^{n} w_i * xi - \theta \right) \geq 0 \leftrightarrow w_1 * x_1 + w_2 * x_2 - \theta \geq 0 \\ -1, se \left(\sum_{i=1}^{n} w_i * xi - \theta \right) < 0 \leftrightarrow w_1 * x_1 + w_2 * x_2 - \theta < 0 \end{cases}$$



Rede Perceptron

- Exemplo de classificação: linearmente separável.

Reta que separa as classes

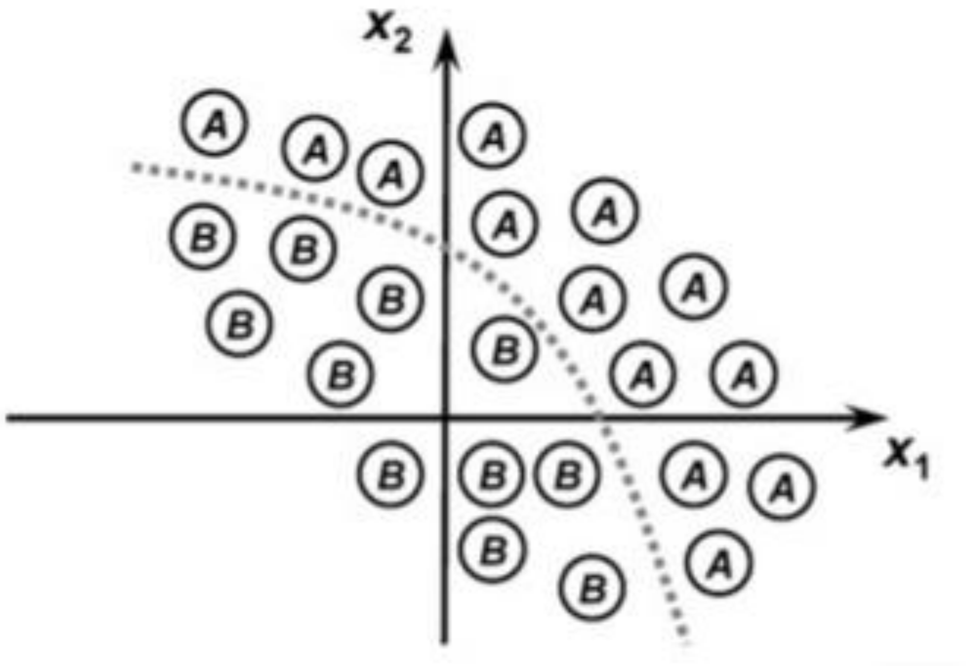


Bias = -1

Se não houvesse o bias, a reta sempre iria passar pela origem.

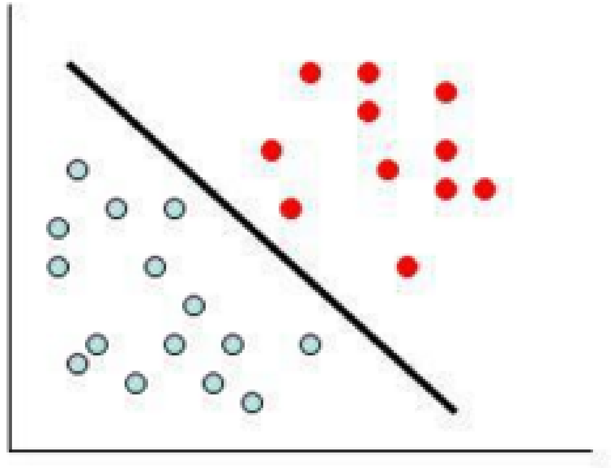
Rede Perceptron

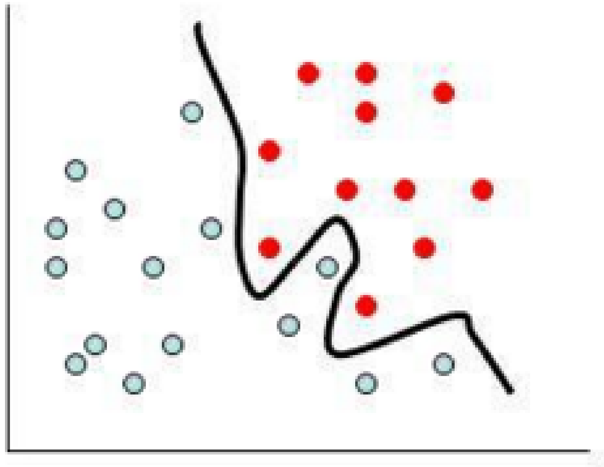
- A rede Perceptron não classifica amostras que não são linearmente separáveis.



Rede Perceptron

- Problemas reais, na maioria das vezes, são mais complexos.



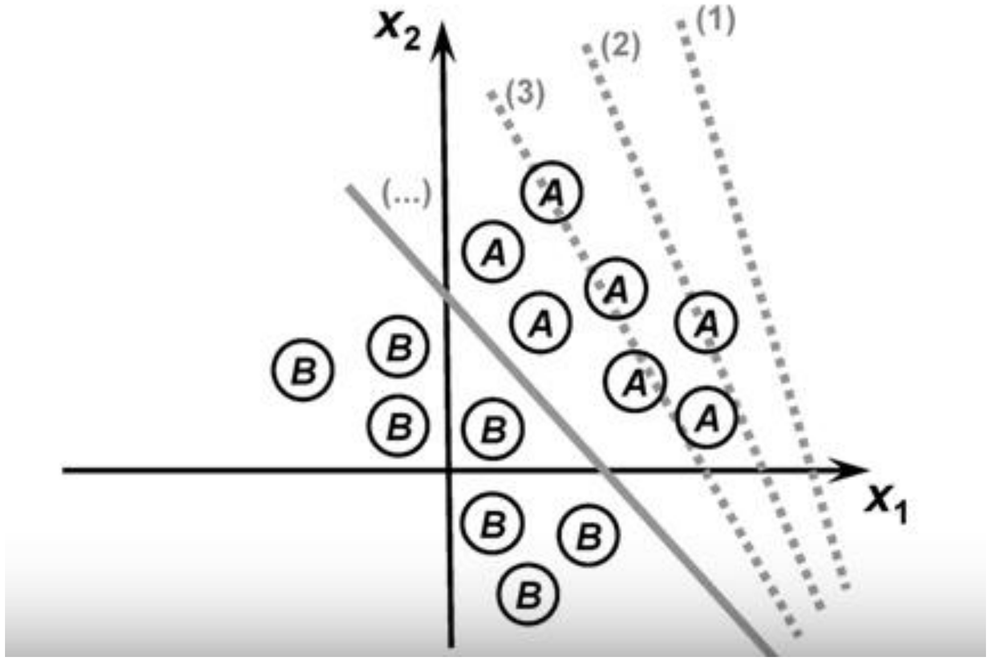


Rede Perceptron

- A operação OR por exemplo é linearmente separável.
  - (0, 0) = 0
  - (1, 0) = 1
  - (0, 1) = 1
  - (1, 1) = 1
- Já a operação XOR não é linearmente serpável.
  - (0, 0) = 0
  - (1, 0) = 1
  - (0, 1) = 1
  - (1, 1) = 0

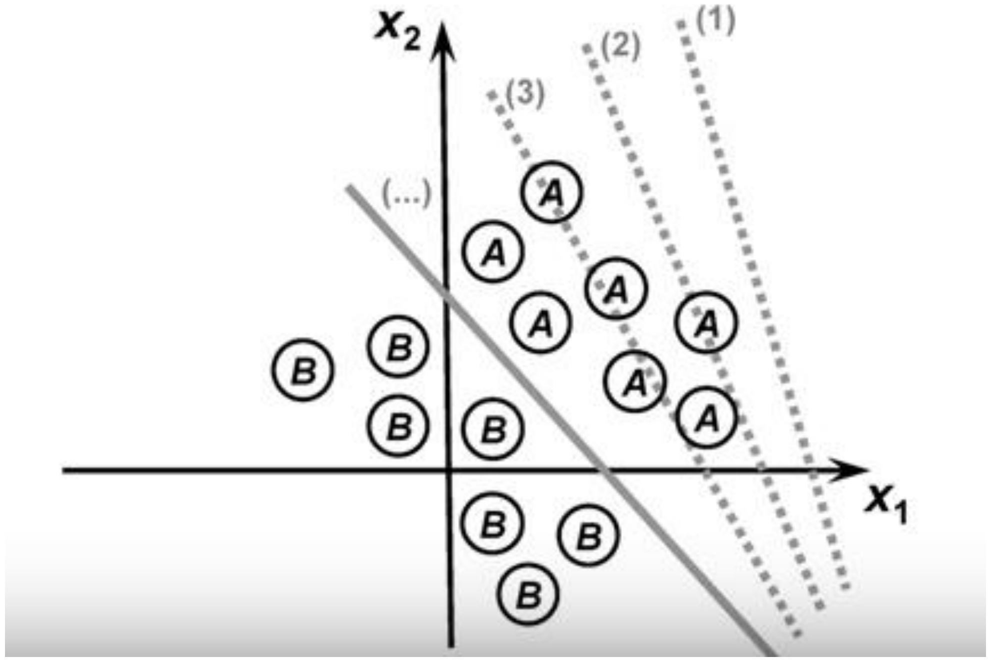
Rede Perceptron

- Treinamento: a cada iteração vai tentando achar a reta que faz a separação das classes.



Rede Perceptron

- Aprender em uma rede neural é ajustar os pesos de forma que tenhamos a saída desejada.



Rede Perceptron

- Processo de treinamento:
  - O objetivo é o ajuste dos pesos!
  - O algoritmo vai fazendo atualizações iterativamente até chegar aos pesos corretos.
  - Na primeira iteração os pesos são gerados aleatoriamente.
  - Normalmente os pesos que são gerados estão no intervalo (0, 1).

Rede Perceptron

- Processo de treinamento:

η ➔ Constante da taxa de aprendizado ($0 < \eta < 1$)

y ➔ Valor de saída produzida pelo *Perceptron*

$d^{(k)}$ ➔ Valor desejado para k-ésima amostra de treinamento

$x^{(k)}$ ➔ K-ésima amostra de treinamento

w ➔ Vetor contendo os pesos (inicialmente gerados aleatoriamente)

Notação matemática: $w^{Atual} = w^{Anterior} + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Notação algorítmica: $w = w + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Rede Perceptron

- Processo de treinamento:
  - A taxa de aprendizado diz o quão rápido a rede chega ao seu processo de classificação. Se colocar um valor muito pequeno, demora a convergir. Se colocar um valor muito alto, pode sair fora do ajuste e nunca convergir.

η ➔ Constante da taxa de aprendizado ($0 < \eta < 1$)

y ➔ Valor de saída produzida pelo *Perceptron*

$d^{(k)}$ ➔ Valor desejado para k-ésima amostra de treinamento

$x^{(k)}$ ➔ K-ésima amostra de treinamento

w ➔ Vetor contendo os pesos (inicialmente gerados aleatoriamente)

Notação matemática: $w^{Atual} = w^{Anterior} + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Notação algorítmica: $w = w + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Rede Perceptron

- Processo de treinamento:
  - O "y" é o valor produzido pelo neurônio.

η ➔ Constante da taxa de aprendizado ($0 < \eta < 1$)

y ➔ Valor de saída produzida pelo *Perceptron*

$d^{(k)}$ ➔ Valor desejado para k-ésima amostra de treinamento

$x^{(k)}$ ➔ K-ésima amostra de treinamento

w ➔ Vetor contendo os pesos (inicialmente gerados aleatoriamente)

Notação matemática: $w^{Atual} = w^{Anterior} + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Notação algorítmica: $w = w + \eta * \left(d^{(k)} - y\right) * x^{(k)}$

Rede Perceptron

- Algoritmo:

Início {Algoritmo *Perceptron* – Fase de Treinamento}

<1> Obter o conjunto de amostras de treinamento $\{\boldsymbol{x}^{(k)}\}$;

<2> Associar a saída desejada $\{d^{(k)}\}$ para cada amostra obtida;

<3> Iniciar o vetor $\boldsymbol{w}$ com valores aleatórios pequenos;

<4> Especificar a taxa de aprendizagem $\{\eta\}$;

<5> Iniciar o contador de número de épocas $\{época \leftarrow 0\}$;

<6> Repetir as instruções:

- <6.1> *erro* ← "inexiste";
- <6.2> Para todas as amostras de treinamento $\{\boldsymbol{x}^{(k)}, d^{(k)}\}$, fazer:
  - <6.2.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)}$;
  - <6.2.2> $y \leftarrow \text{sinal}(u)$;
  - <6.2.3> Se $y \neq d^{(k)}$
    - <6.2.3.1> Então $\begin{cases} \boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ erro \leftarrow \text{"existe"} \end{cases}$
- <6.3> $época \leftarrow época + 1$;

Até que: *erro* ← "inexiste"

Fim {Algoritmo *Perceptron* – Fase de Treinamento}

Rede Perceptron

- Após o treinamento, temos a fase de operação:

Início {Algoritmo *Perceptron* – Fase de Operação}

<1> Obter uma amostra a ser classificada { $\boldsymbol{x}$ };

<2> Utilizar o vetor $\boldsymbol{w}$ ajustado durante o treinamento;

<3> Executar as seguintes instruções:

<3.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}$;

<3.2> $y \leftarrow \text{sinal}(u)$;

<3.3> Se $y = -1$

<3.3.1> Então: amostra $\boldsymbol{x} \in$ *{Classe A}*

<3.4> Se $y = 1$

<3.4.1> Então: amostra $\boldsymbol{x} \in$ *{Classe B}*

Fim {Algoritmo *Perceptron* – Fase de Operação}

Dica de livro

Contato

mcastrosouza@live.com

www.geeksbr.com

https://twitter.com/mcastrosouza