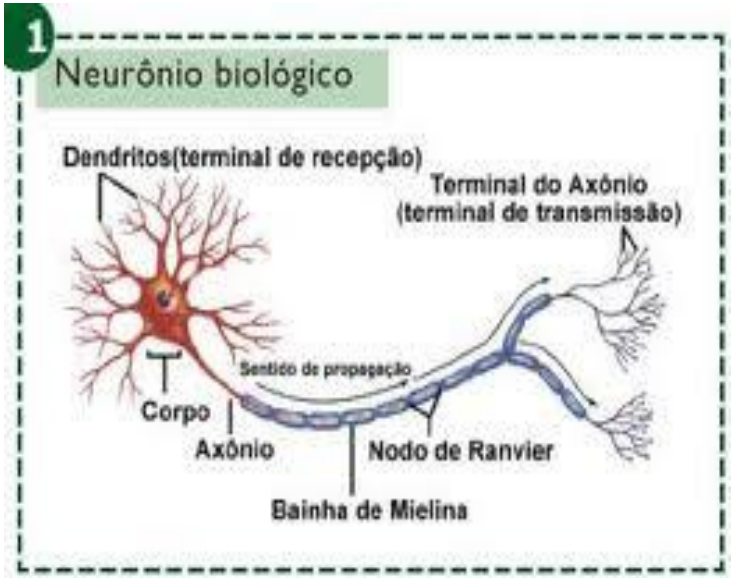
Redes Neurais Artificiais

profº Mauricio Conceição Mario

Redes Neurais Artificiais - RNAs

- As RNAs são modelos matemáticos que se assemelham às estruturas neurais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização (Braga et al., 2000; Haykin, 1994).



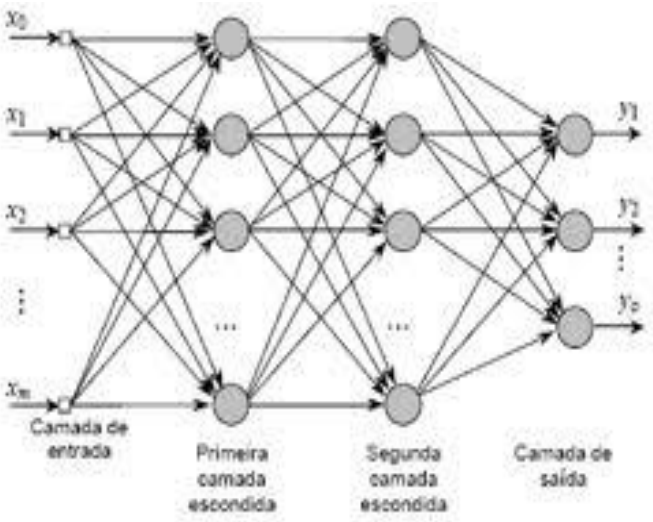


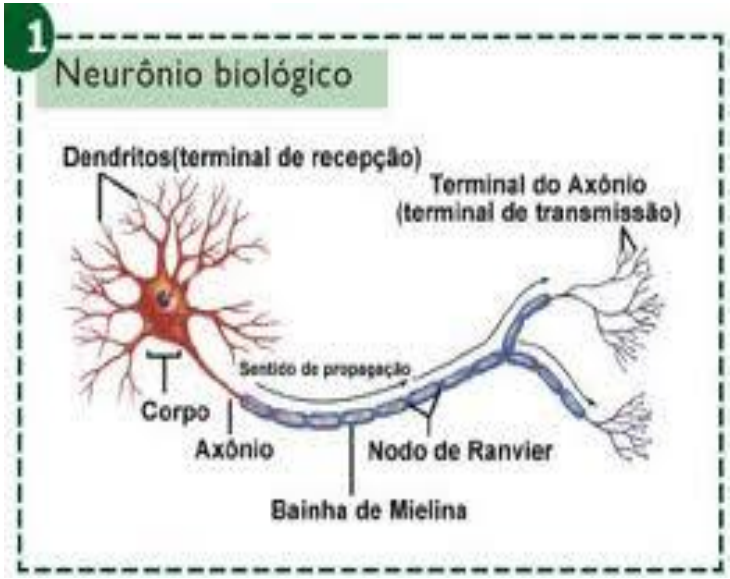
Figura 1 – Rede Neural Artificial Multicamadas

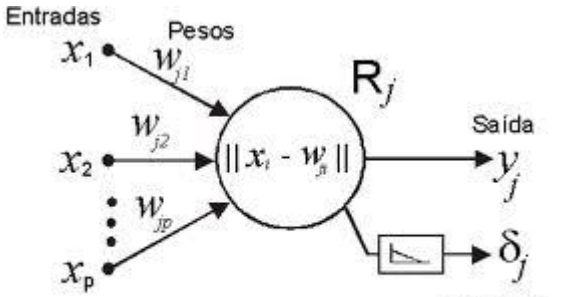
http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw=

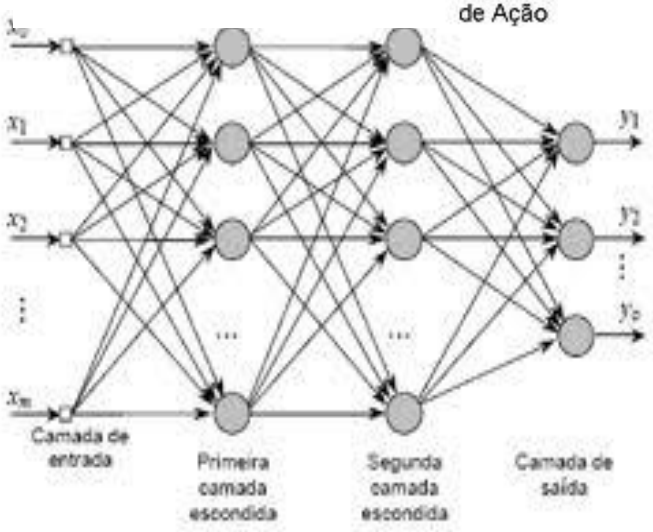
Redes Neurais Artificiais

- São sistemas inspirados nos neurônios biológicos e na estrutura massivamente paralela do cérebro, com capacidade de adquirir, armazenar e utilizar conhecimento experimental.

(notas de aula do profº Keiji Yamanaka, 2000)



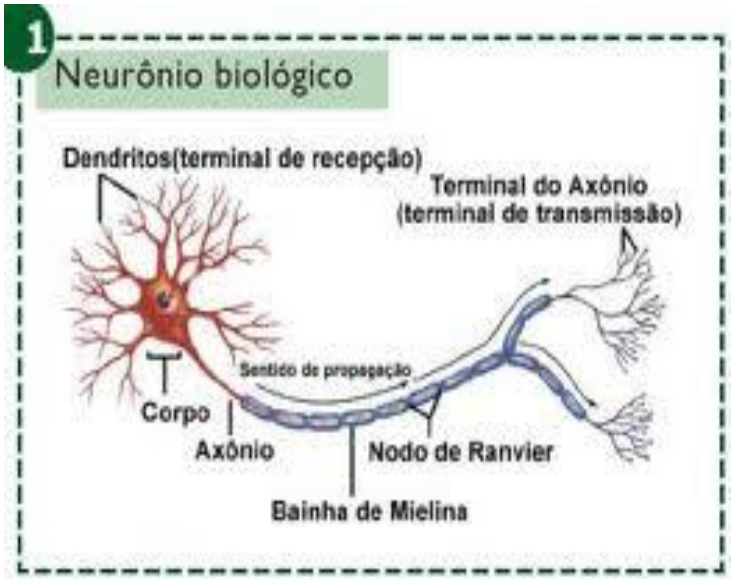




http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw= Figura 1 – Rede Neural Artificial Multicamadas

Redes Neurais Artificiais - Histórico

- O primeiro trabalho reconhecido como um processo de IA foi realizado por Warren McCulloch e Walter Pitts (McCulloch e Pitts, 1943) e está relacionado com as RNAs. McCulloch e Pitts basearam-se em três fontes: o conhecimento da fisiologia básica e da função dos neurônios do cérebro, uma análise formal da lógica proposicional, e a teoria da computação de Turing (Turing, 1950).
- A lógica proposicional juntamente com o estudo do cálculo de predicados formam a parte nuclear da Lógica Clássica.
- Alan Turing (1950) projetou o *"Teste de Turing"* para fornecer uma definição operacional satisfatória de "inteligência".



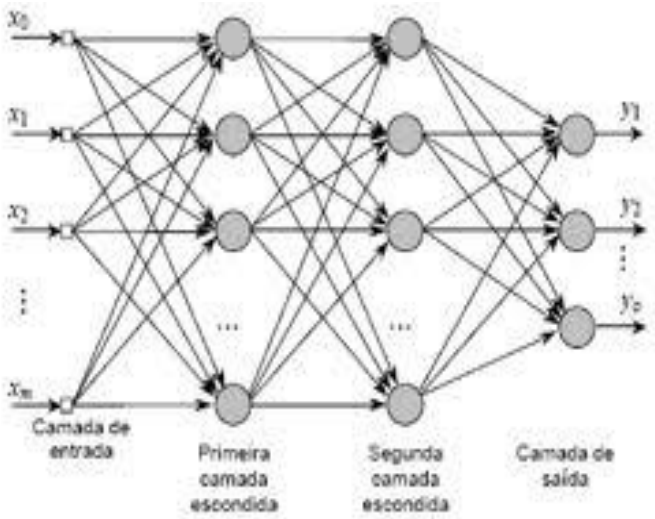


Figura 1 – Rede Neural Artificial Multicamadas

http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw=

Redes Neurais Artificiais

- **Aprendizado:** treinamento efetuado através da apresentação de exemplos.
  - Utilização de algoritmos para o ajuste dos parâmetros da rede neural;
  - Substitui, após o treinamento, uma programação necessária para a execução de determinada tarefa.

(notas de aula do profº Keiji Yamanaka, 2000)

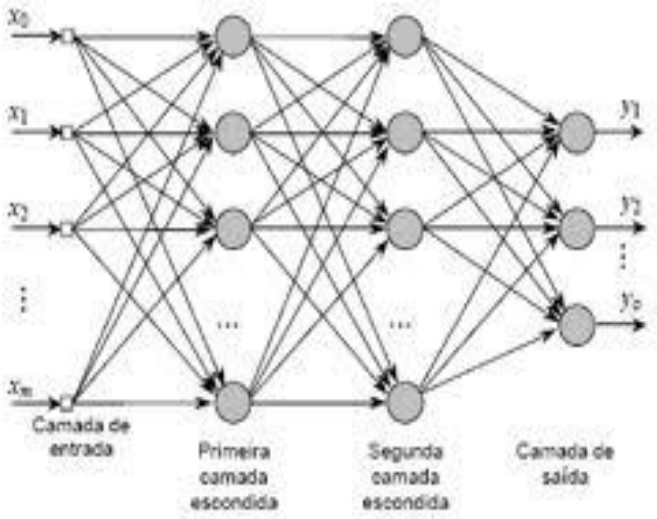


Figura 1 – Rede Neural Artificial Multicamadas

http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw=

Aprendizado

Generalização

Classificação

amostra que não participou do treinamento

Eletrocardiograma ou Eletroencefalograma

classificação

não pertence ao padrão

Saída da Rede Neural

1 1 1

1 1 1

1 1 0

pertence ao padrão

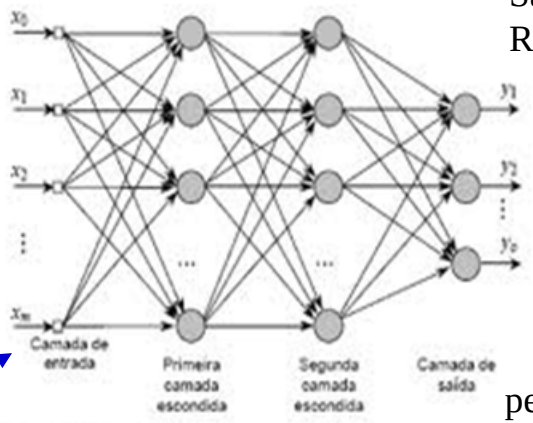


Figura 1 – Rede Neural Artificial Multicamadas

treinamento e aprendizado

conjunto de amostras de treinamento

generalização

amostra que não participou do treinamento

Redes Neurais Artificiais

- **Aplicações:**
  - reconhecimento de padrões
  - classificação;
  - aproximação funcional;
  - suporte à decisão;
  - extração de informação.

(notas de aula do profº Keiji Yamanaka, 2000)

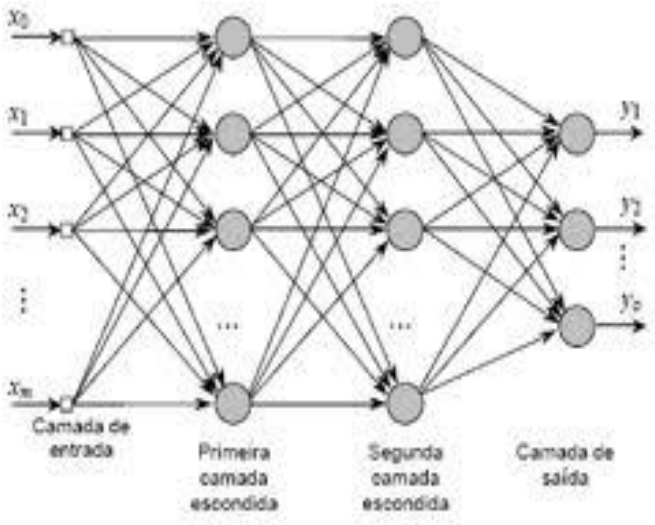


Figura 1 – Rede Neural Artificial Multicamadas

http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw=

Redes Neurais Artificiais - características

O **aprendizado** e a **generalização** constituem características importantes das RNAs.

- O aprendizado está associado à capacidade das RNAs adaptarem os seus parâmetros como conseqüência da sua interação com o meio externo. O processo de aprendizado é interativo e por meio dele a RNA deve melhorar o seu desempenho gradativamente à medida que interage com o meio externo.

- A generalização de uma RNA está associada à sua capacidade de dar respostas coerentes para dados não apresentados a ela previamente durante o treinamento.

(Rezende, 2003)

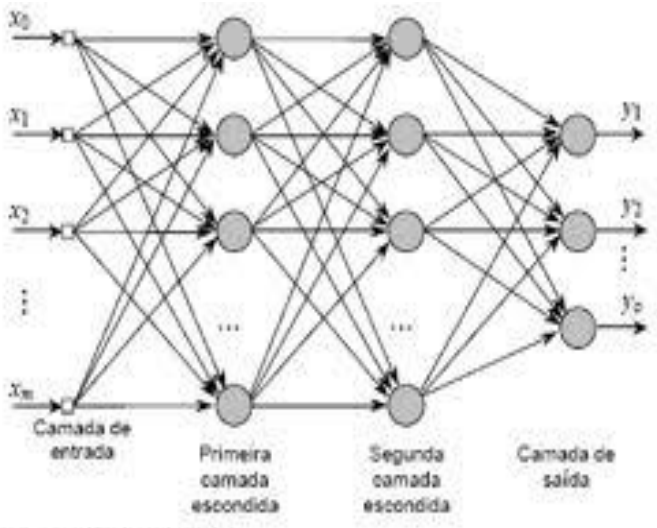


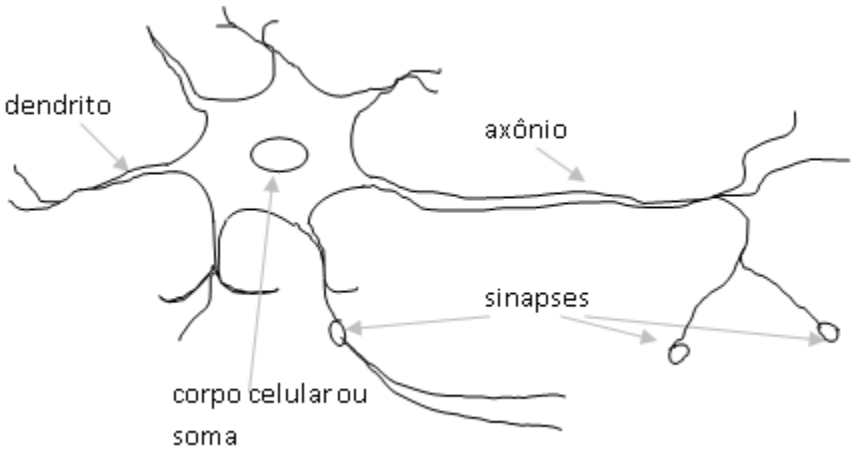
Figura 1 – Rede Neural Artificial Multicamadas

http://www.google.com.br/images?um=1&hl=pt-BR&rlz=1R2SUNC_pt-BRBR376&biw=

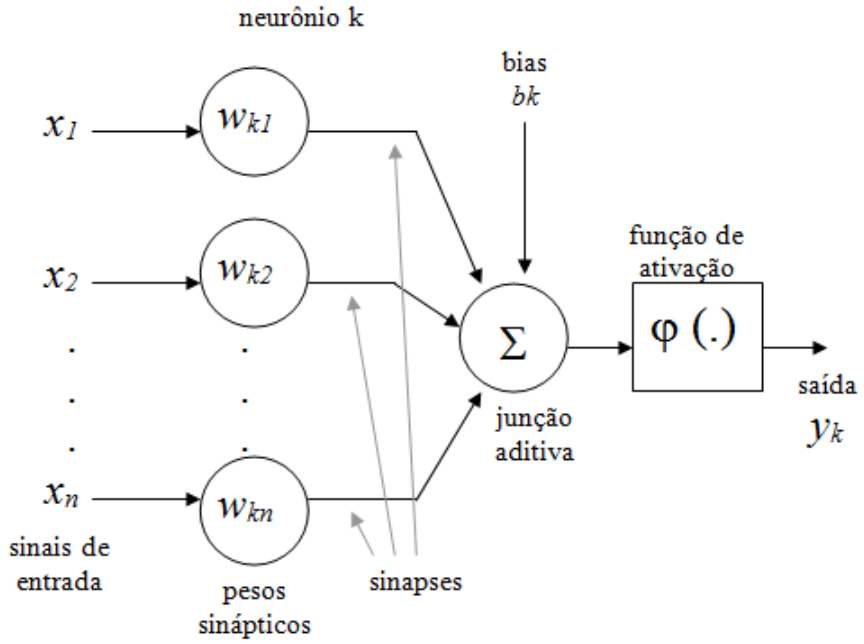
Redes Neurais Artificiais - características

Uma rede neural artificial se "assemelha" ao cérebro em alguns aspectos por essas características:

- O conhecimento é adquirido pela rede a partir de seu ambiente, através de um processo de aprendizagem;
- Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido (Haykin, 2001).
- O esboço de um neurônio biológico e, comparativamente, o modelo não-linear de um neurônio artificial são mostrados a seguir:

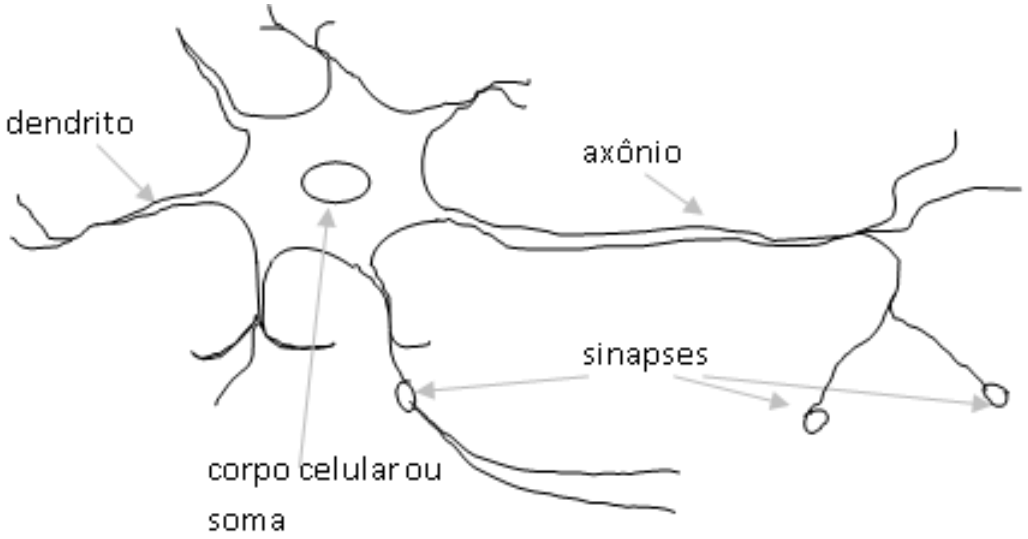


Esquemático simplificado de partes de uma célula nervosa ou neurônio.



Modelo de um neurônio artificial típico.

Redes Neurais Artificiais – RNAs: neurônio biológico

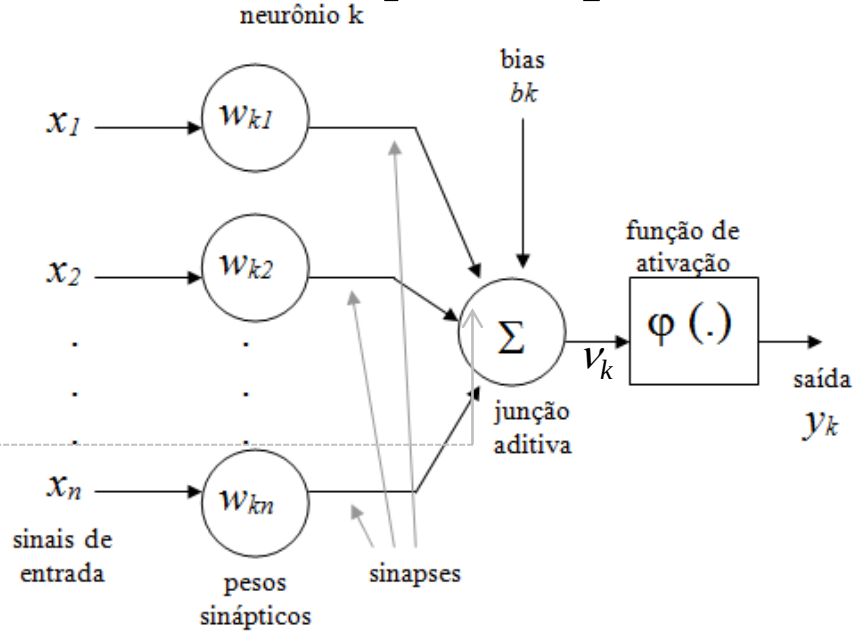


Um neurônio faz conexões com 10 a 100.000 outros neurônios, em junções chamadas sinapses. Os sinais se propagam de um neurônio para outro por meio de uma reação eletroquímica. Os sinais controlam a atividade cerebral em curto prazo, e também permitem mudanças a longo prazo na posição e na conectividade dos neurônios. Acredita-se que esses mecanismos formem a base para o aprendizado no cérebro (Russell e Norvig, 2004).

Redes Neurais Artificiais – modelo de um neurônio

O modelo típico de **neurônio artificial** forma a base para o desenvolvimento de Redes Neurais Artificiais. O modelo é formado por um conjunto de **sinapses**, caracterizadas por um **peso**.

$$u_k = \sum_{j=1}^{n} w_{kj}\, x_j$$



Ou neurônio de *McCulloch-Pitts* - MCP

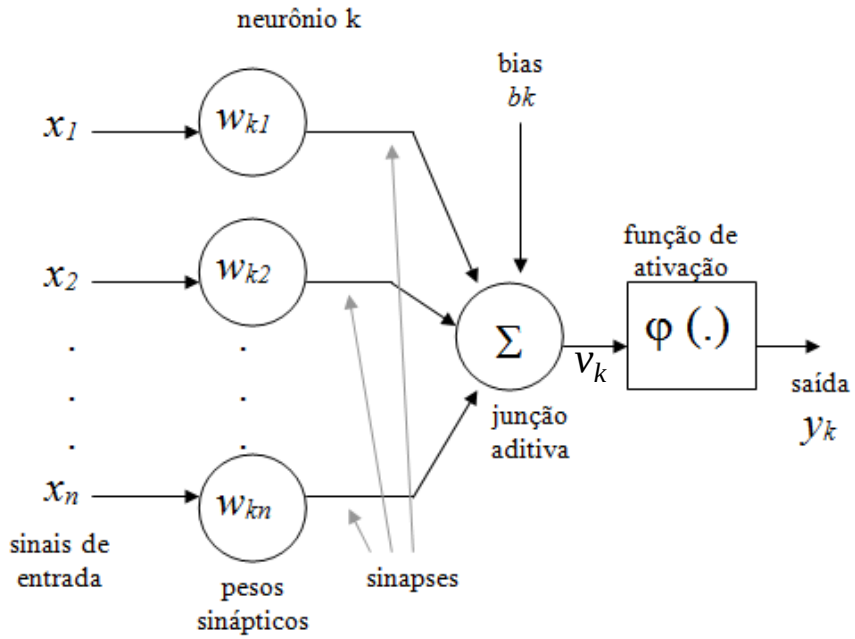
Um sinal x_j aplicado na entrada da sinapse ***j*** conectada ao **neurônio *k*** é multiplicada pelo peso sináptico w_{kj}, onde o primeiro índice *k* se refere ao neurônio em questão e o segundo *j* se refere ao terminal de entrada da sinapse à qual o peso se refere. O **peso sináptico** de um neurônio artificial pode estar em um intervalo que inclui valores positivos e negativos (Haykin, 2001).

Redes Neurais Artificiais – modelo de um neurônio

A junção aditiva soma os sinais de entrada, ponderados pelas respectivas sinapses do neurônio artificial. O ***bias*** ou ajuste aplicado externamente produz o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se o mesmo é respectivamente positivo ou negativo.

$$v_k = uk + b_k$$

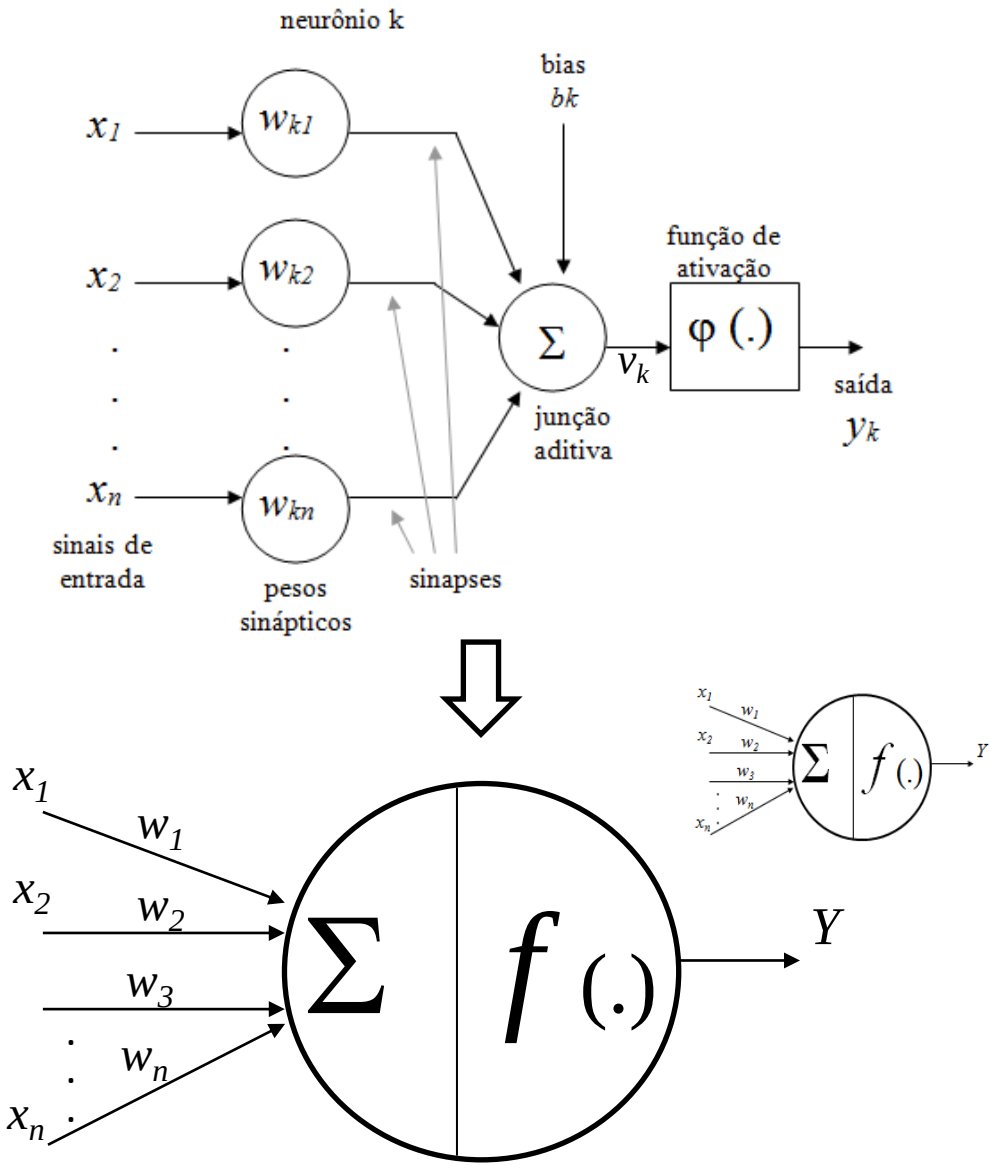
Ou neurônio de *McCulloch-Pitts* - MCP



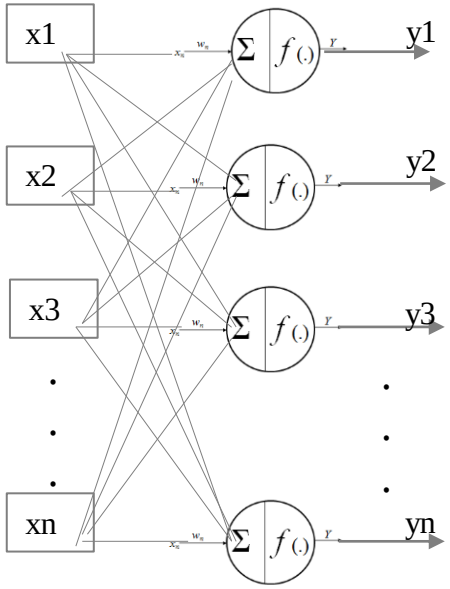
A função de ativação limita a amplitude da saída de um neurônio. O intervalo normalizado da amplitude de saída de um neurônio está entre [0, 1] ou [-1, 1] (Haykin, 2001).

$$y_k = \varphi(u_k + b_k)$$

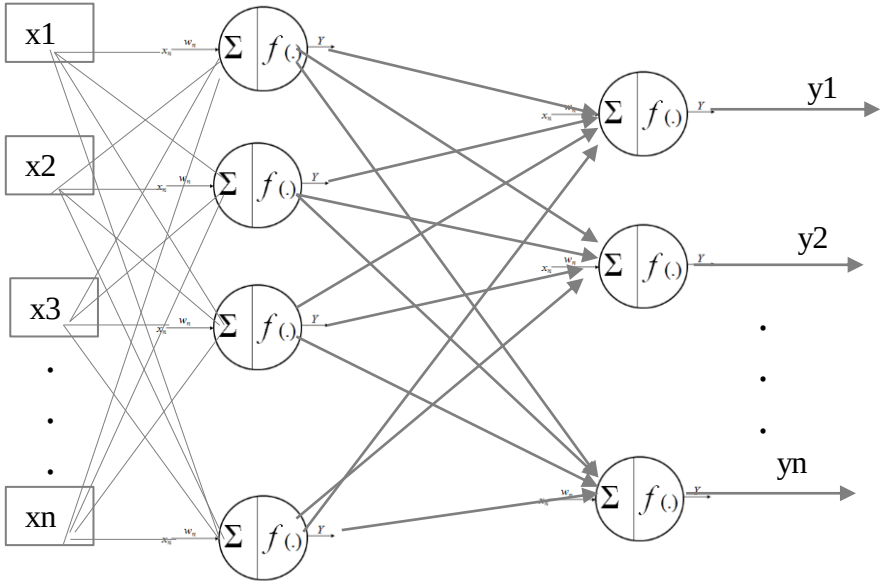
Representação simplificada de um neurônio



Arquiteturas de RNAs: redes *feedforward*



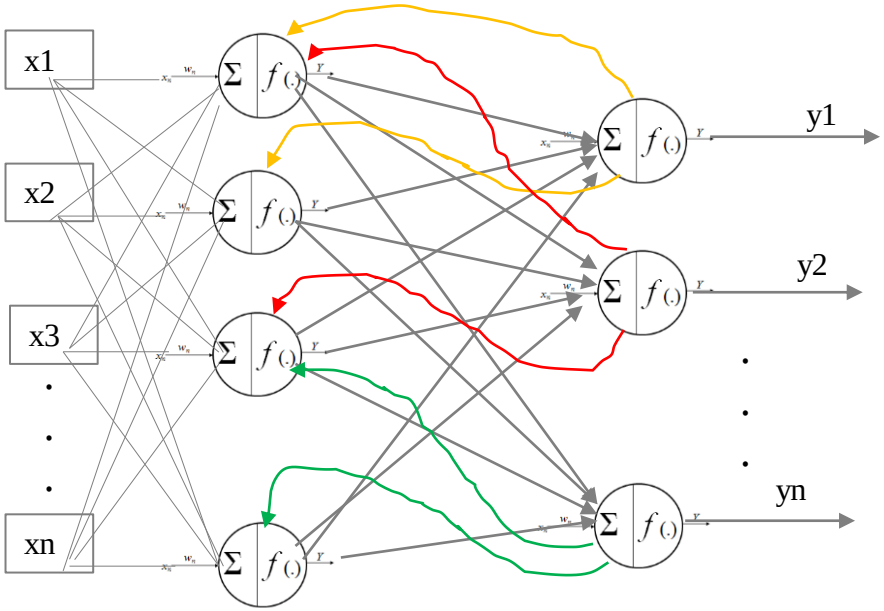
Rede *feedforward* (alimentada para frente) de 1 camada



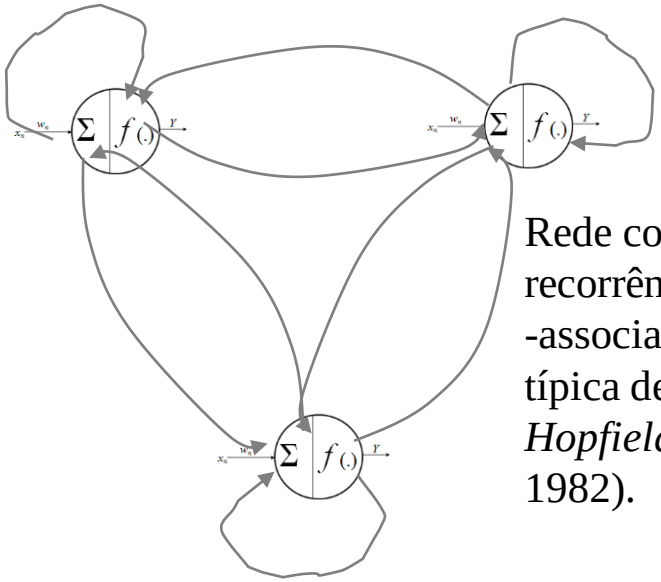
Rede *feedforward* de 2 camadas

As redes *feedforward* são redes alimentadas para frente. A rede *feedforward* de 1 camada é capaz de resolver problemas com várias variáveis e funções acopladas, mas tem restrições com relação a problemas complexos. A rede *feedforward* de 2 camadas tem maior capacidade computacional e universalidade na aproximação de funções contínuas devido a camada intermediária. São consideradas estáticas por não possuírem recorrência: as suas saídas em um determinado instante dependem apenas das entradas atuais (Braga, 2007).

Arquiteturas de RNAs: redes com recorrência



Rede com recorrência entre saídas e camadas intermediárias



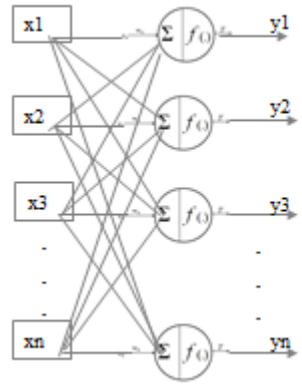
Rede competitiva ou recorrência auto--associativa: estrutura típica de uma rede de *Hopfield* (Hopfield, 1982).

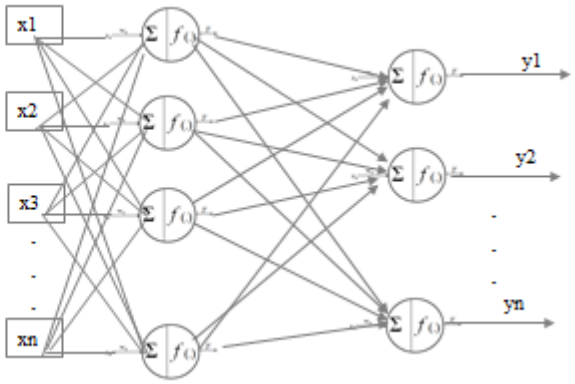
Estas estruturas de rede possuem conexões recorrentes entre neurônios de um mesmo nível ou entre neurônios de saída e de camadas anteriores.

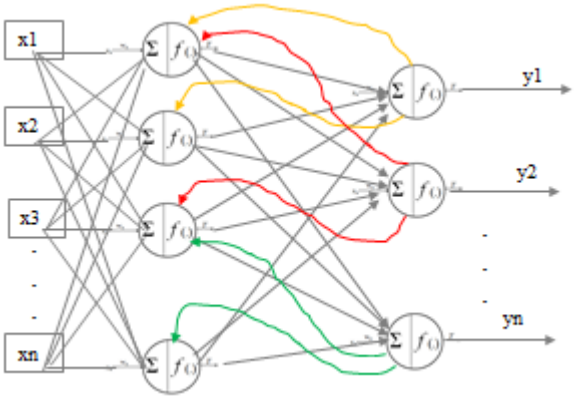
Na rede com recorrência entre saídas e camadas intermediárias, as saídas não dependem apenas das entradas, mas também do seu valor atual. Esta estrutura de RNA é utilizada na resolução de problemas que envolvam processamento temporal, como em previsão de eventos futuros.

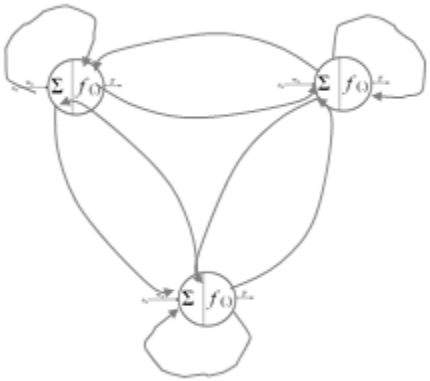
A rede com recorrência auto-associativa possui um único nível de neurônios, em que a saída de cada um deles está conectada às entradas de todos os outros. A rede não possui entradas externas, e sua operação se dá em função da dinâmica de mudança de estados dos neurônios, que operam de forma auto--associativa (Braga, 2007).

Arquiteturas de RNAs: critérios de escolha







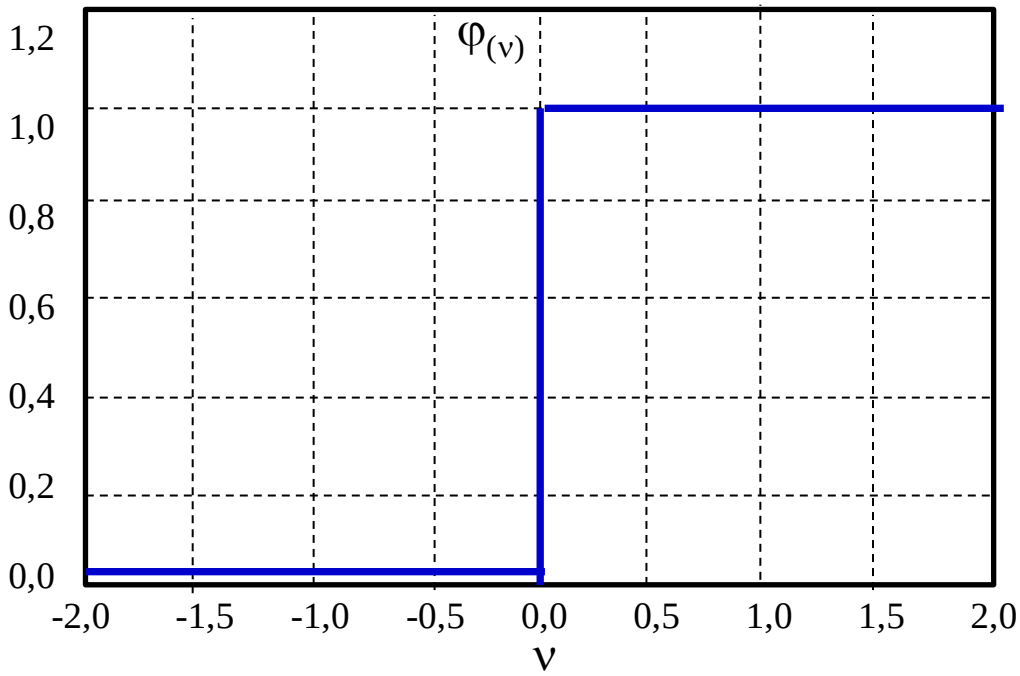


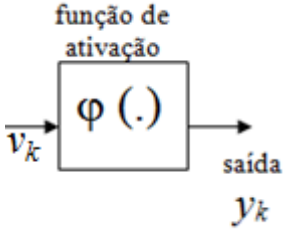
A definição da estrutura de uma RNA para a resolução de um determinado problema depende de vários fatores:

- Complexidade do problema;
- Dimensionalidade do espaço de entrada;
- Características dinâmicas ou estáticas;
- Conhecimento a *priori* sobre o problema;
- Representatividade dos dados.

(Braga, 2007)

Redes Neurais Artificiais – funções de ativação





O neurônio com esta função de ativação é referido como o modelo de *McCulloch-Pitts* (McCulloch-Pitts, 1943).

1. Função de *Limiar* ou função de *Heavside ou* função *degrau*:

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

ou como no neurônio

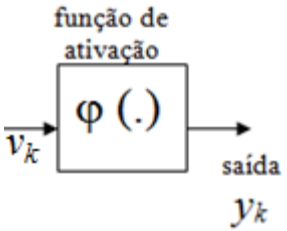
$$\varphi_k = \begin{cases} 1 \text{ se } \nu_k > 0 \\ 0 \text{ se } \nu_k \leq 0 \end{cases}$$

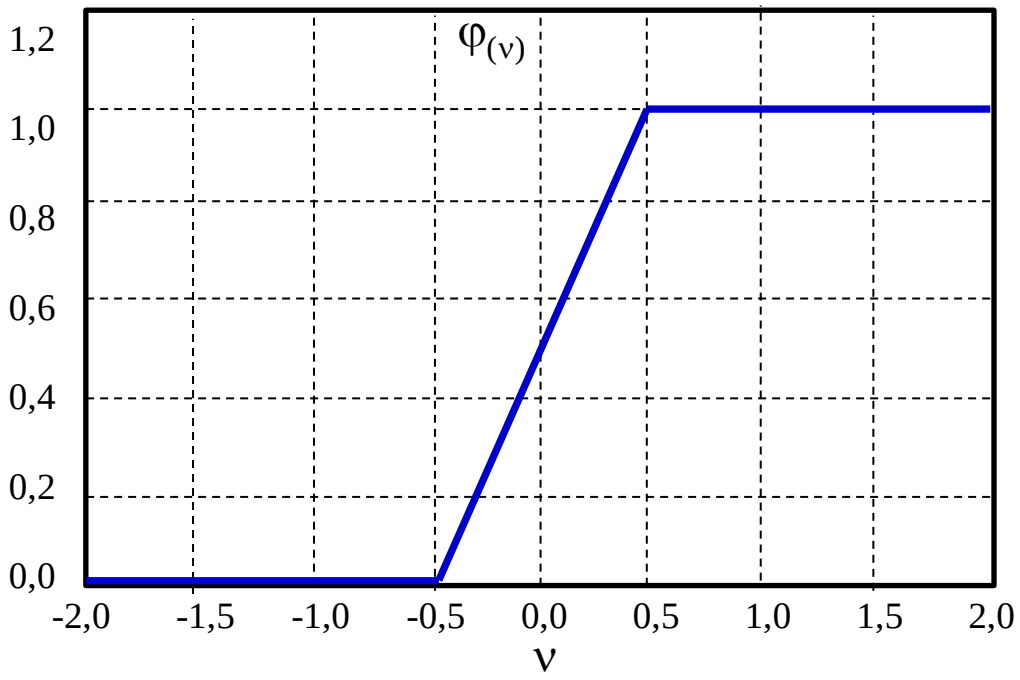
onde ν_k é:

$$\nu_k = \sum_{j=1}^{m} w_{kj}\, xj + bk$$

(Haykin, 2001).

Redes Neurais Artificiais – funções de ativação



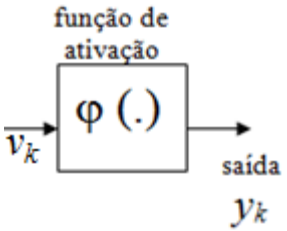


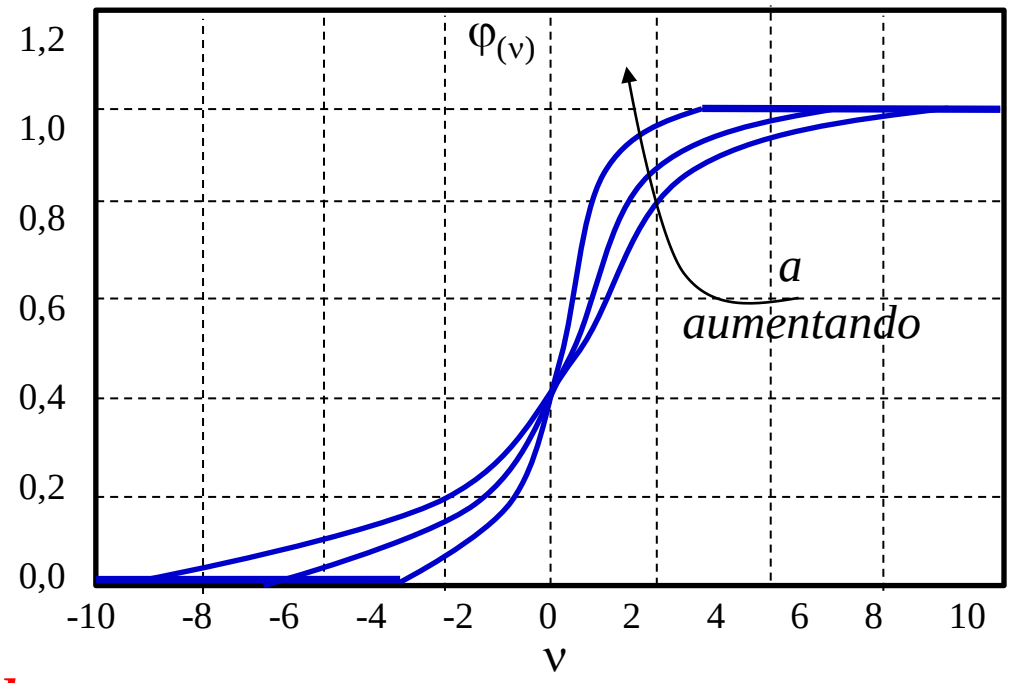
2. Função *Limiar por Partes*:

$$\varphi(\nu) = \begin{cases} 1, & \nu \geq +\frac{1}{2} \\ \nu, & +\frac{1}{2} > \nu > -\frac{1}{2} \\ 0, & \nu \leq -\frac{1}{2} \end{cases}$$

(Haykin, 2001).

Redes Neurais Artificiais – funções de ativação





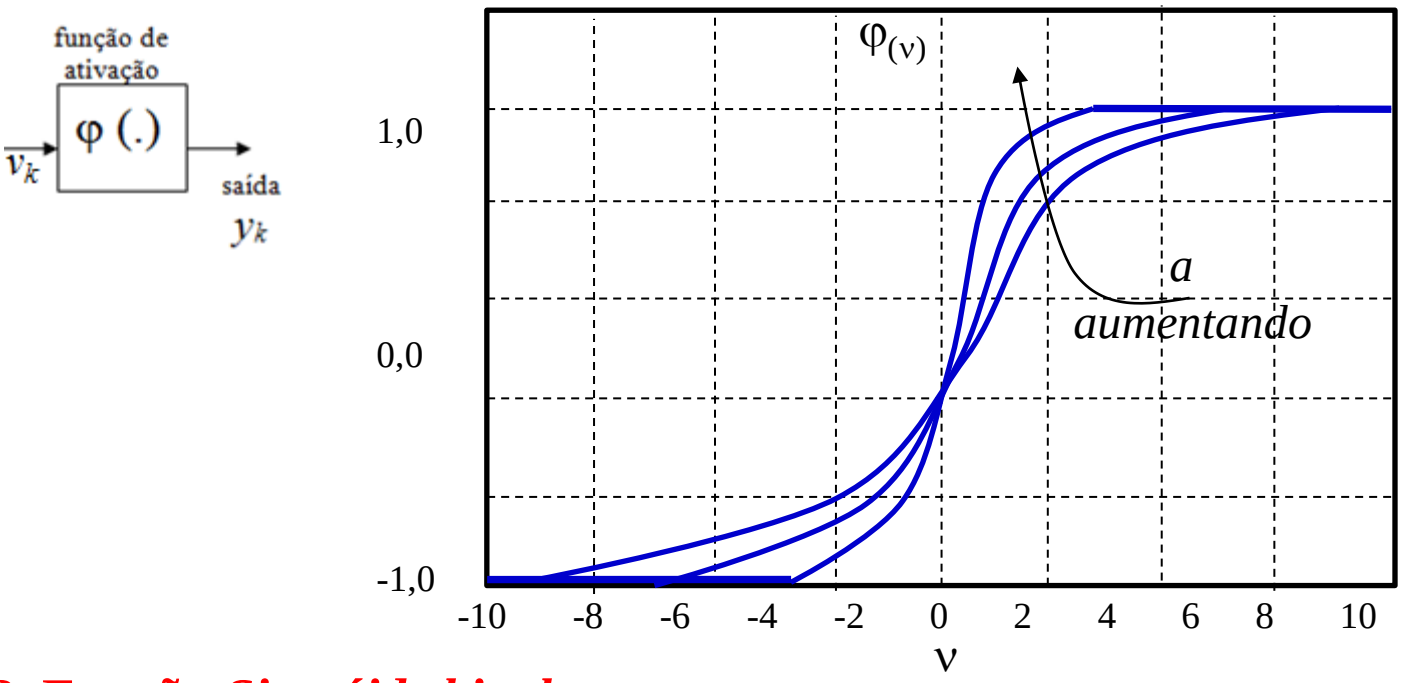
3. Função *Sigmóide*:

$$\varphi_{(\nu)} = \frac{1}{1+ e^{-av}}$$

Onde *a* é o parâmetro de inclinação da função sigmóide.

(Haykin, 2001).

Redes Neurais Artificiais – funções de ativação



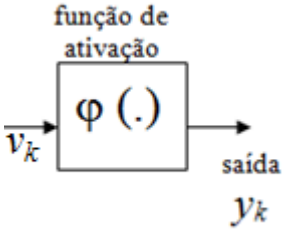
3. Função *Sigmóide bipolar*:

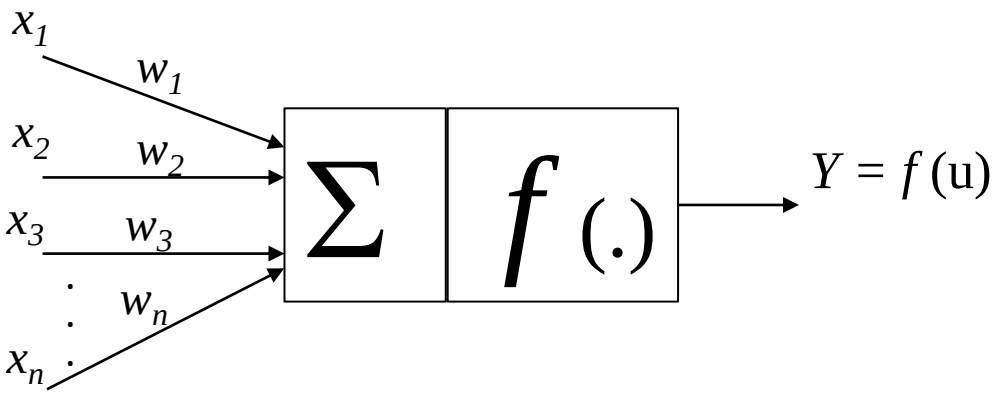
$$\varphi_{(\nu)} = \frac{2}{1+ e^{-a\nu}} - 1$$

Onde *a* é o parâmetro de inclinação da função sigmóide.

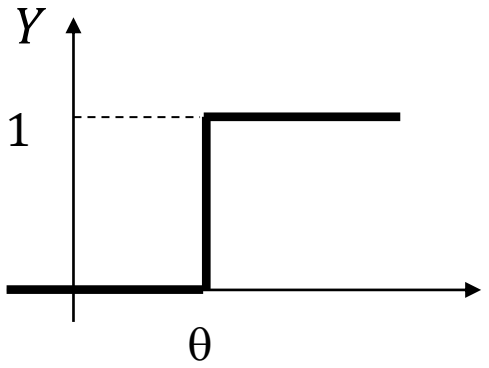
(Haykin, 2001).

Redes Neurais Artificiais – função de ativação ou porta de limiar em degrau





$$Y = f(\mathrm{u}) = \begin{cases} 1 & u = \sum w_i\, xi > \theta \\ 0 & u = \sum w_i\, xi \leq \theta \end{cases}$$



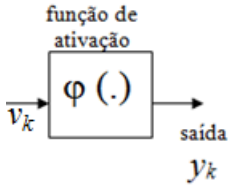
As entradas de uma porta de limiar linear são restritas a valores booleanos “0” ou “1”.

A saída Y é obtida através da aplicação da função de ativação f (.) sobre a soma ponderada das entradas u, ou seja, $Y = f$ (u).

$$u = \sum_{i=1}^{n} x_i\, w_i$$

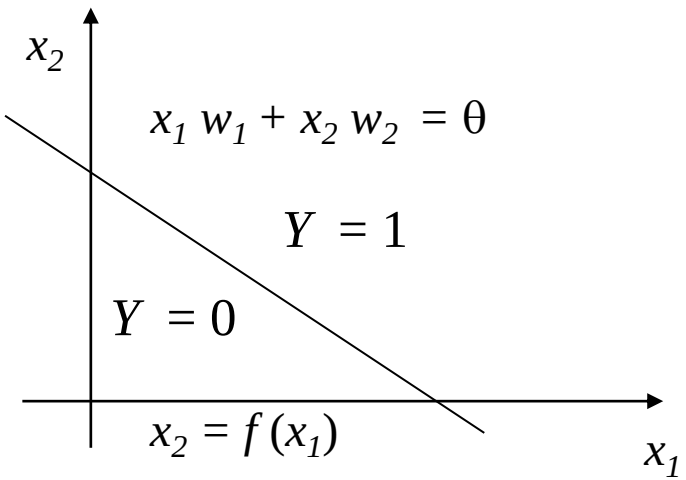
Redes Neurais Artificiais – porta de limiar linear

$$Y = f(u) = \begin{cases} 1 & u = \sum w_i\, xi > \theta \\ 0 & u = \sum w_i\, xi \leq \theta \end{cases}$$



De acordo com a equação, as portas de limiar lineares estão restritas à solução de problemas que sejam ***linearmente separáveis***, ou seja, cuja solução possa ser obtida pela separação de duas regiões por meio de uma reta. O espaço de entrada da porta de limiar é dividido em vetores ***x*** que levam a saída ***Y*** para "1" ou "0".

Os vetores ***x*** ficam em lados opostos de uma superfície de separação linear, sendo que, para $\boldsymbol{\theta > 0}$, os pontos que correspondem a $\boldsymbol{Y = 1}$ ficam acima e os pontos que correspondem a $\boldsymbol{Y = 0}$ ficam abaixo da superfície. Abaixo, representação para duas entradas (Braga et. al, 2007).



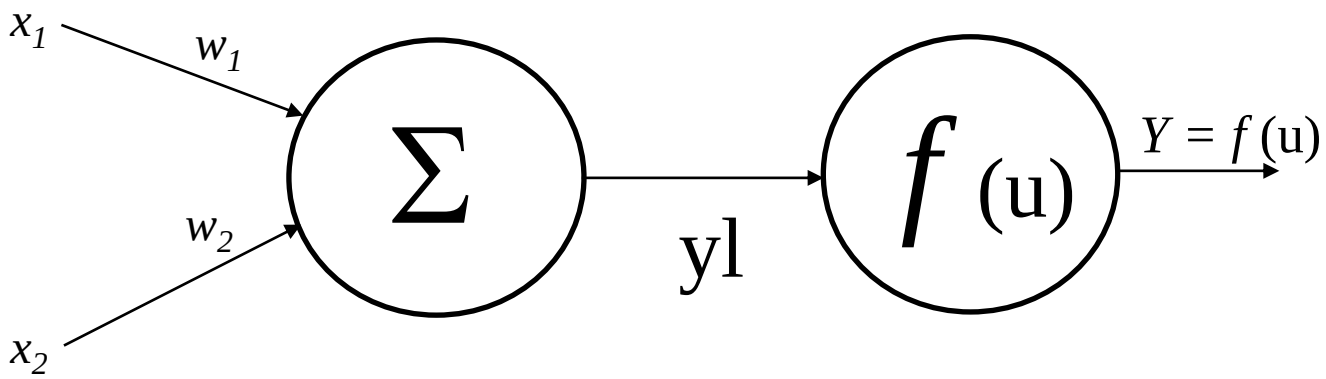
Porta de limiar quadrática

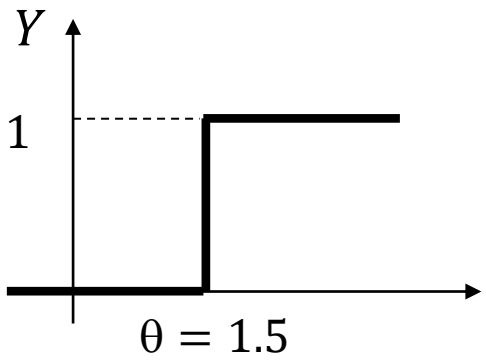
Recomendada quando houver muitas entradas, tem maior capacidade computacional por possuir mais parâmetros ajustáveis

$$Y = f(u) = \begin{cases} 1 & u = \sum w_i\, x_i + \sum w_{ij} w_i\, x_i > \theta \\ 0 & u = \sum w_i\, x_i + \sum w_{ij} w_i\, xi \leq \theta \end{cases}$$

Algoritmo de porta de limiar degrau

modelo:





Exemplo: construir neurônio artificial utilizando função de ativação degrau para implementar a função booleana E (AND).

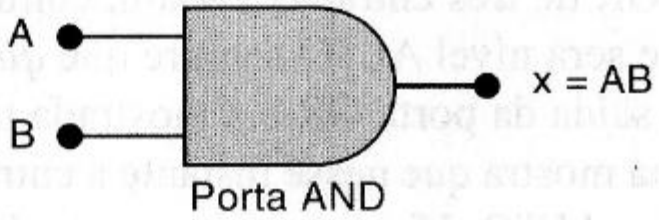
Porta e função booleana "E"

$$x = A \cdot B$$

AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

(a)

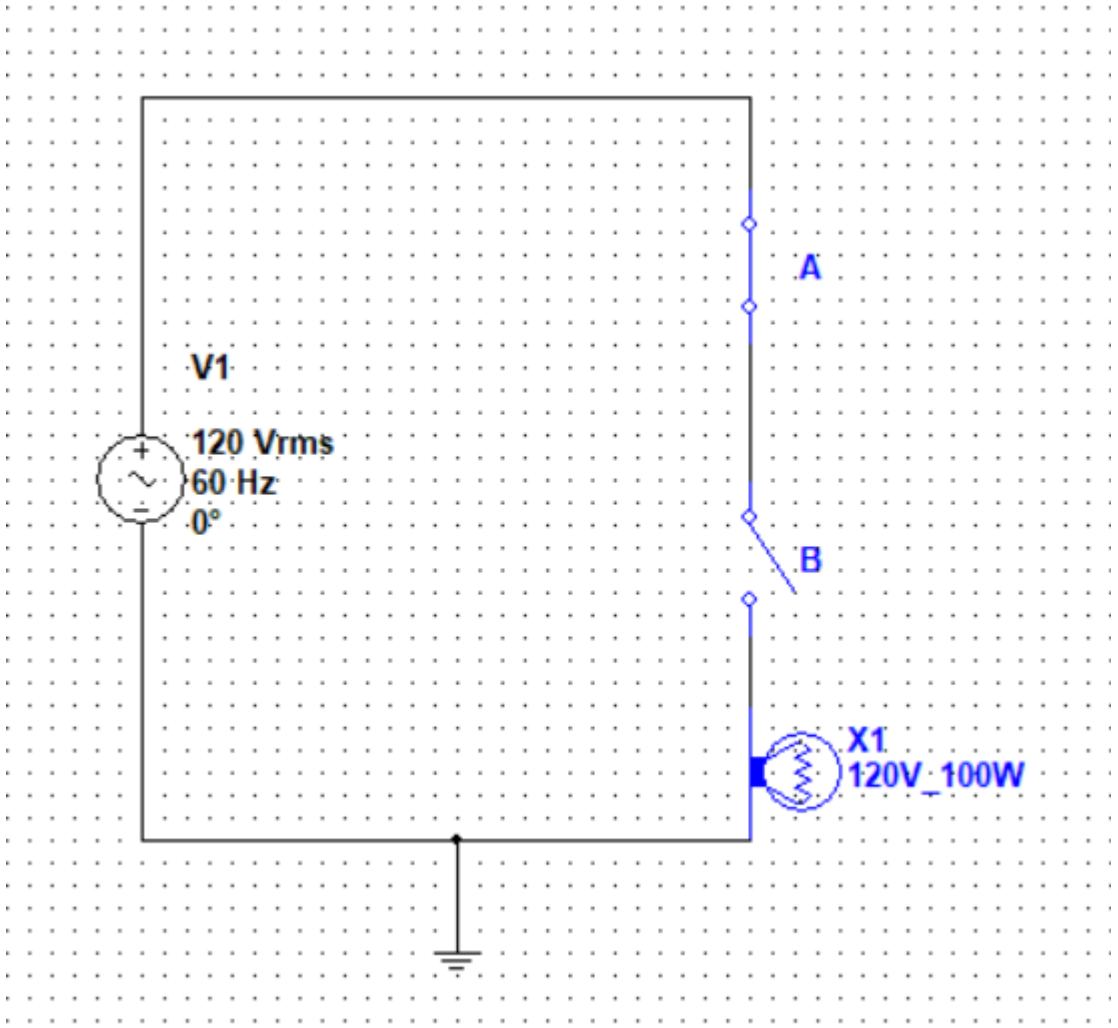


(b)

(a) Tabela-verdade para a operação AND; (b) símbolo da porta AND.

(Tocci et al, 2011)

Porta e função booleana “E”



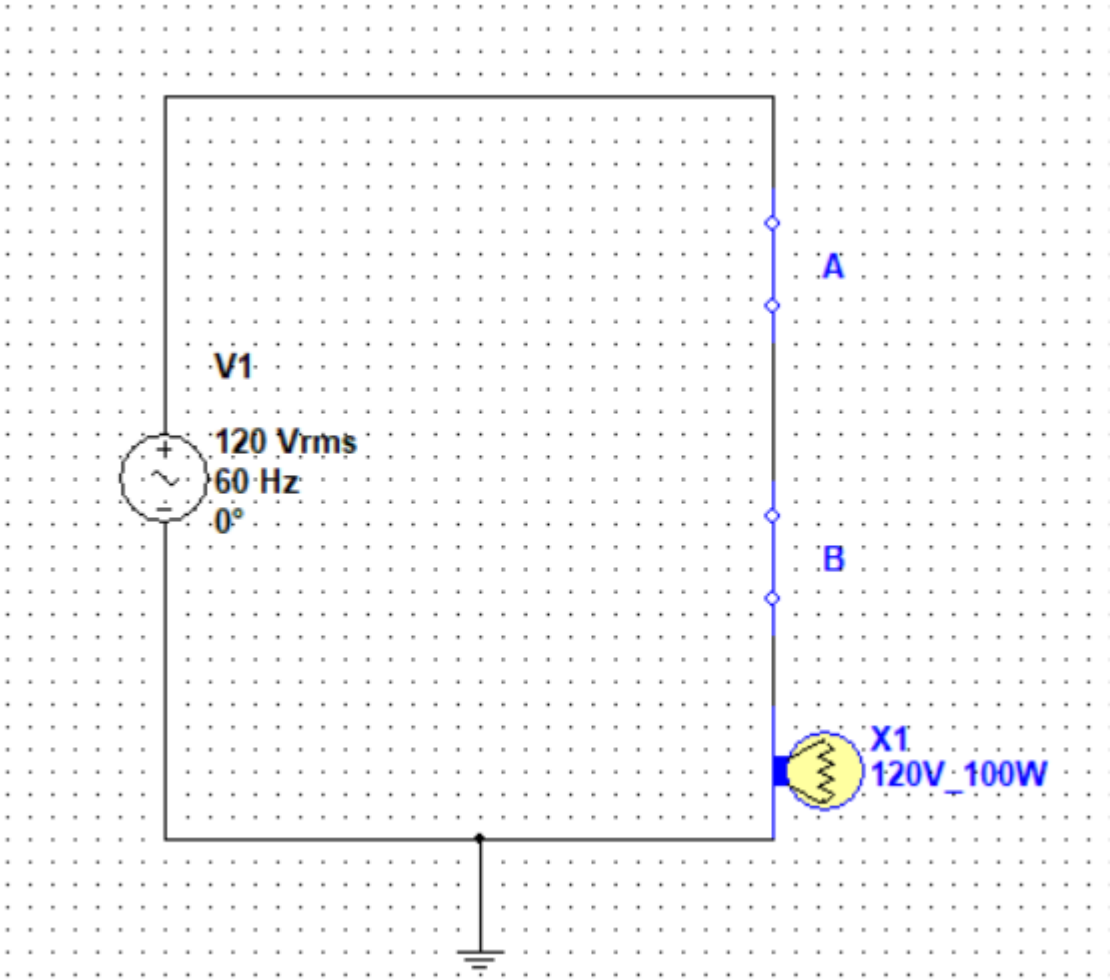
AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

chave fechada = 1
chave aberta = 0
lâmpada acesa = 1
lâmpada apagada = 0

(Tocci et al, 2011)

Porta e função booleana “E”



AND

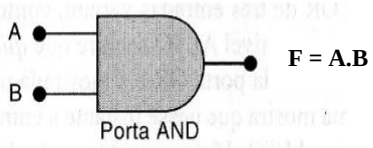
A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

chave fechada = 1
chave aberta = 0
lâmpada acesa = 1
lâmpada apagada = 0

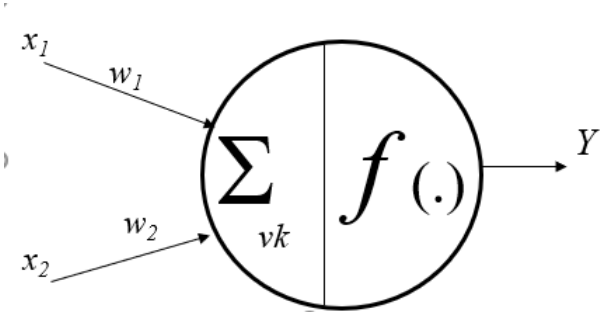
(Tocci et al, 2011)

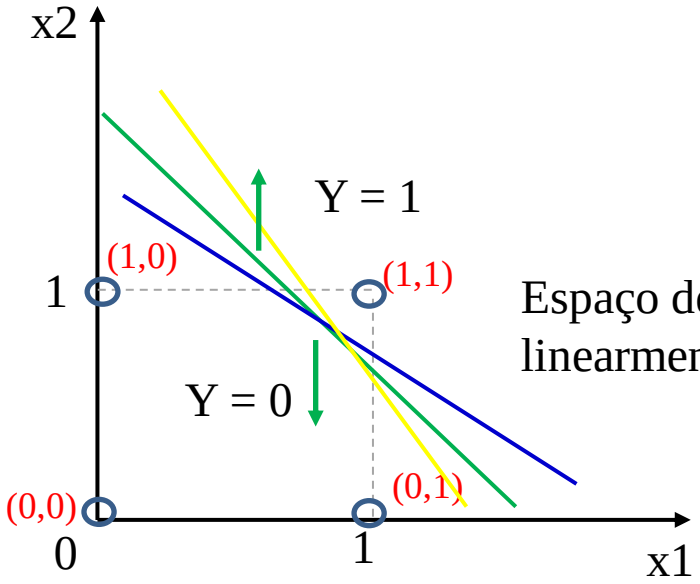
Neurônio MCP implementando função booleana. Condição: espaço de saída linearmente separável.

A = x2	B = x1	F = Y	*vk* = x1*w1 + x2*w2	θ = 1.5 (limiar) *vk* > θ → Y = 1 *vk* ≤ θ → Y = 0
0	0	0	*vk* = 0*1 + 0*1 = 0	0
0	1	0	*vk* = 0*1 + 1*1 = 1	0
1	0	0	*vk* = 1*1 + 0*1 = 1	0
1	1	1	*vk* = 1*1 + 1*1 = 2	1



Neurônio MCP





Espaço de saída é linearmente separável

Definição das variáveis:
pesos: w1 e w2
entradas: x1 e x2
limiar da função de ativação: limiar
retorno do cálculo da função de ativação: func_ativacao
recebe o valor da função de ativação: y1
valor da saída: y

classe degrau

função de ativação( ):
func_ativacao =
x1[n]*w1 + x2[n]*w2

função degrau( ):
contador: n = 0
para cada par de entradas x1 e x2
y1 = função de ativação( )
se (y1 > limiar) → y = 1
senão → y = 0
n = n + 1

não

n = 1?

sim

classe testa_degrau

chama função verifica_degrau( ) →
para todas as combinações das entradas x1 e x2

função verifica_degrau( ):
saida = x1[n]*w1 + x2[n]*w2

Classe degrau

degrau.java

```
11   */
12  public class degrau {
13      double w1 = 1.0; double w2 = 1.0;
14      double x1[] = {0.0, 0.0, 1.0, 1.0};
15      double x2[] = {0.0, 1.0, 0.0, 1.0};
16      double func_ativacao;
17      double limiar = 1.5;
18      int y;
19      double y1;
20
21      public void degrau() {
22          int n = 0;
23          while  (n < 2){
24   for (int i = 0; i <= 3 ; i++ ){
25       y1 = calcula_ativação(x1[i], x2[i], w1, w2);
26
27   if (y1 >  limiar)
28      y = 1;
29   else y = 0;
30   }
31   n = n + 1;
32       }
33  }
34      public double calcula_ativação(double x1, double x2, double peso1, double peso2 ){
35
36          func_ativacao =  x1*peso1 + x2*peso2; //bias +
37          return func_ativacao;
38      }
```

Classe degrau

```
    public void verifica_degrau (double x1, double x2, double peso1, double peso2){
        double saida =  x1*peso1 + x2*peso2;
        if (saida >  limiar)
        y = 1;
        else y = 0;
        System.out.println(" porta E: F = (x1 =" + x1 + ") E (x2 =" + x2 + " )   = " + y);
    }
}
```

```
double w1 = 1.0; double w2 = 1.0;
double x1[] = {0.0, 0.0, 1.0, 1.0};
double x2[] = {0.0, 1.0, 0.0, 1.0};
double func_ativacao;
double limiar = 1.5;
```

Classe testa_degrau

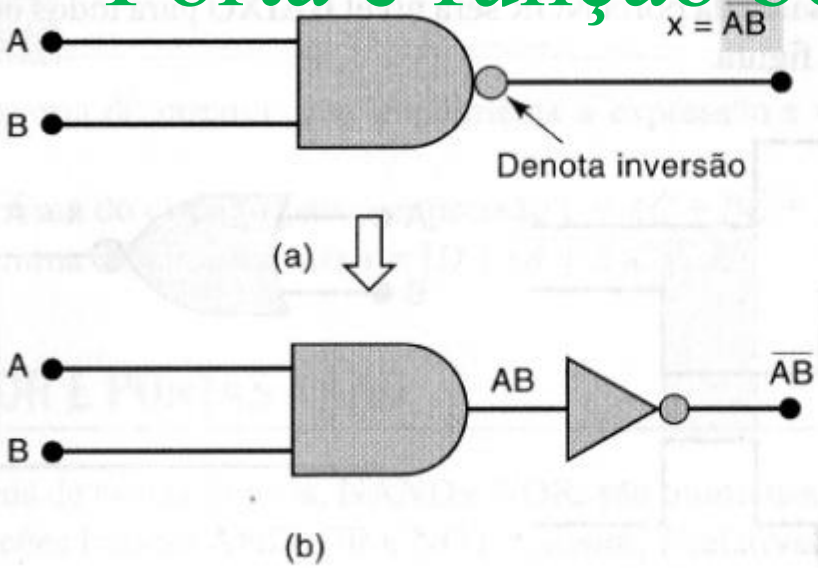
```
public class testa_degrau {

    public static void main(String args[]){

    degrau ld = new degrau();

    ld.verifica_degrau(ld.x1[0], ld.x2[0], ld.w1, ld.w2);
    ld.verifica_degrau(ld.x1[1], ld.x2[1], ld.w1, ld.w2);
    ld.verifica_degrau(ld.x1[2], ld.x2[2], ld.w1, ld.w2);
    ld.verifica_degrau(ld.x1[3], ld.x2[3], ld.w1, ld.w2);

    }
}
```

console

```
run:
 porta E: F = (x1 =0.0) E (x2 =0.0 )  = 0
 porta E: F = (x1 =0.0) E (x2 =1.0 )  = 0
 porta E: F = (x1 =1.0) E (x2 =0.0 )  = 0
 porta E: F = (x1 =1.0) E (x2 =1.0 )  = 1
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

2. Construir algoritmo para implementar uma rede neural que tenha funcionalidade de uma função booleana NE (NAND):

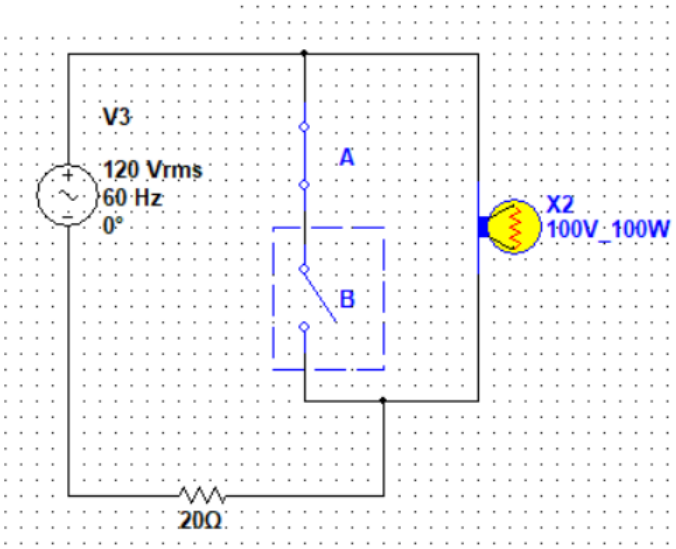
Porta e função booleana "NE"



		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(c)

(a) Símbolo da porta NAND; (b) circuito equivalente; (c) tabela-verdade.



(Tocci et al, 2011)

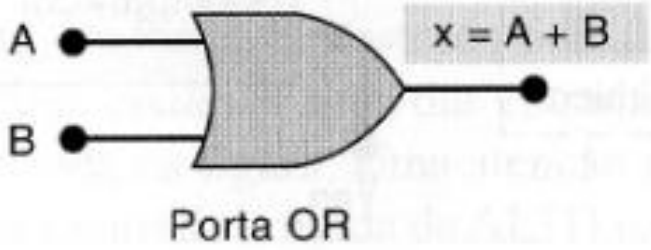
1. Construir algoritmo para implementar uma rede neural que tenha funcionalidade de uma função booleana OU (OR):

Porta e função booleana "OU"

OR

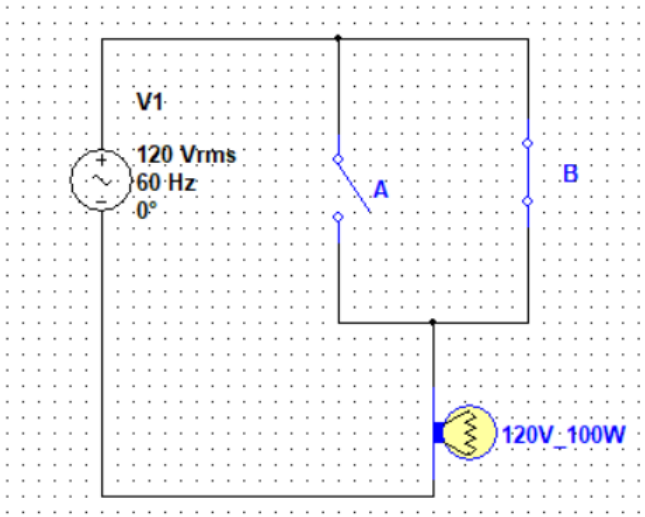
A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

(a)



(b)

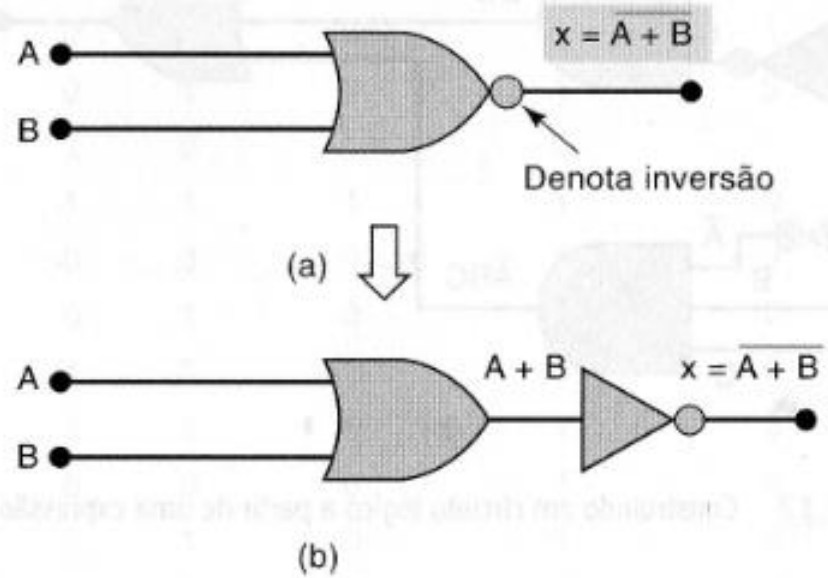
(a) Tabela-verdade que define a operação OR; (b) símbolo de uma porta OR de duas entradas.



(Tocci et al, 2011)

3. Construir algoritmo para implementar uma rede neural que tenha funcionalidade de uma função booleana NOR (NOU):

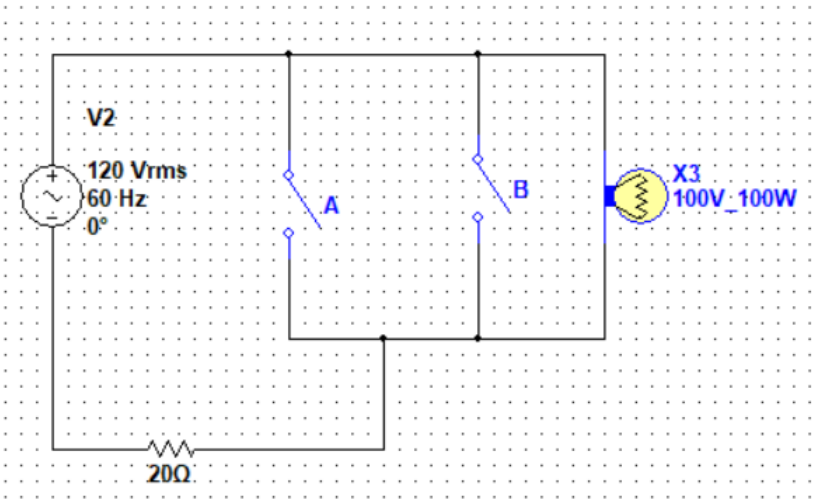
Porta e função booleana "NOU"



		OR	NOR
A	B	A + B	A + B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(c)

(a) Símbolo da porta NOR; (b) circuito equivalente; (c) tabela-verdade.



(Tocci et al, 2011)

Referências Bibliográficas

- Braga AP, Carvalho APLF, Ludermir TB. *Redes Neurais Artificiais: teoria e aplicações*. Livros Técnicos e Científicos, Rio de Janeiro – RJ; 2007.

- Haykin S. *Neural Networks – A Comprehensive Foundation.* Prentice-Hall; 1994.

- Haykin S. *Redes Neurais – Princípios e prática.* 2a ed.. Porto Alegre: Bookman; 2001.

- Hebb DO. *The Organization of Behavior.* John Wiley & Sons; 1949.

- Heckerman D. *Probabilistic Similarity Networks.* MIT Press, Cambridge, Massachussets; 1991.

-Hopfield JJ. *Neurons with graded response have collective computational properties like those of two-state neurons.* Proceedings of the National Academy of Sciences of the United States of America, 79, 2554-2558; 1982.

- *Sistemas Digitais – princípios e aplicações - 11a. Edição*
Ronald J. Tocci / Neal S. Wildmer / Gregory L. Moss
Pearson / Prentice Hall - 2011

Referências Bibliográficas

- Mario MC. *Proposta de Aplicação das Redes Neurais Artificiais Paraconsistentes como Classificador de Sinais Utilizando Aproximação Funcional*. Univ. Federal de Uberlândia, Dissertação de Mestrado, Uberlândia; 2003.

- McCarthy J. *Programs with commom sense.* In Proceedings of the Symposium on Mechanisation of Thought Processes, Vol. 1, pp. 77-84, London. Her Majesty's Stationery Office; 1958.

- McCulloch W, Pitts W. *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics 5, 115-133; 1943.

- Rosenblatt F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan, Chicago; 1962.

- Rumelhart DE, McClelland JL. *Parallel Distributed Processing.* MIT Press, Cambridge, Massachussets; 1986.

- Turing A. *Computing machinery and intelligence.* Mind, 59, 433-460; 1950.