Aprendizagem Automática

João Paulo Pordeus Gomes

Redes Neurais

Redes Neurais

- Funcionamento inspirado no neurônio biológico.
- Tarefas de Aprendizado de Máquina
  - Classificação
  - Regressão

Neurônio Biológico x Neurônio Artificial

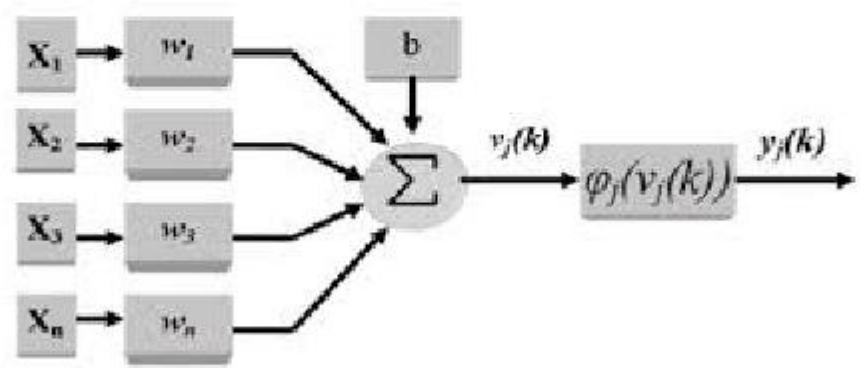


Figura 2: Representação do neurônio artificial.

Neurônio Biológico x Neurônio Artificial

McCulloch-Pitts

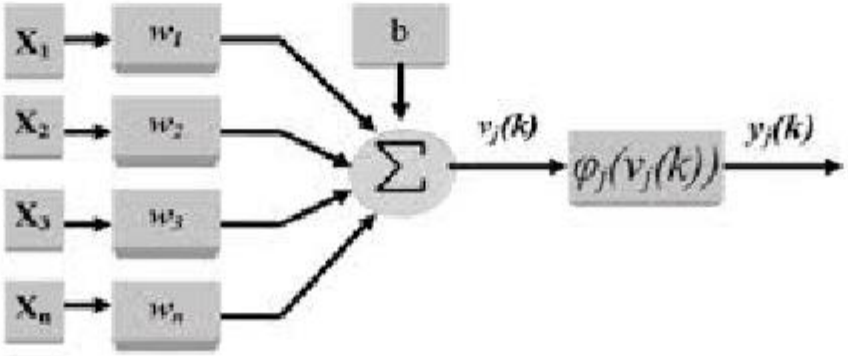


Figura 2: Representação do neurônio artificial.

Modelo de McCulloch-Pitts

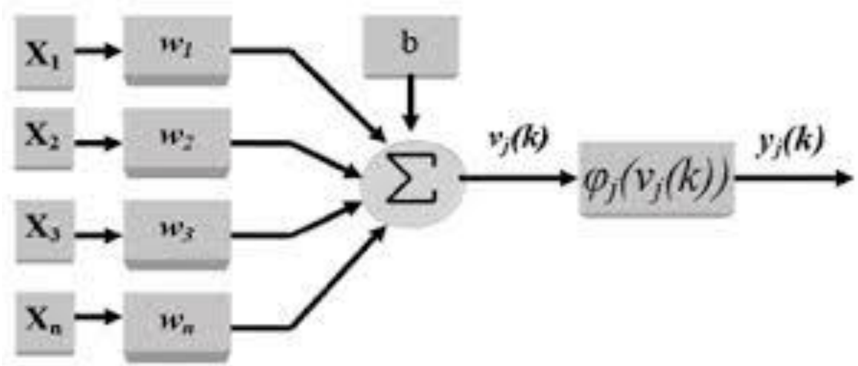


Figura 2: Representação do neurônio artificial.

$$v_j = w_1 x_1 + w_2 x_2 + \cdots - b$$

$$\varphi(v_j) = 1 \; se \; v_j > 0$$
$$\varphi(v_j) = 0 \; se \; v_j < 0$$

Modelo de McCulloch-Pitts

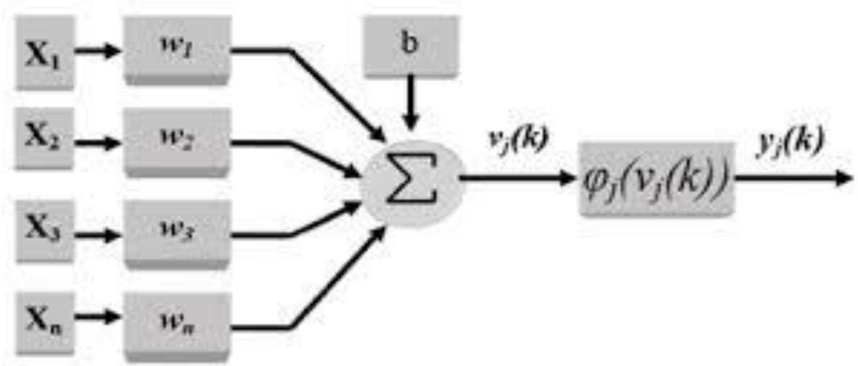


Figura 2: Representação do neurônio artificial.

$v_j = w_1x_1 + w_2x_2 + \cdots - w_0x_0 = \boldsymbol{w^T x}$ onde $w_0 = -1$ e $x_0 = b$

$\varphi(v_j) = 1\ se\ v_j > 0$

$\varphi(v_j) = 0\ se\ v_j < 0$

Perceptron

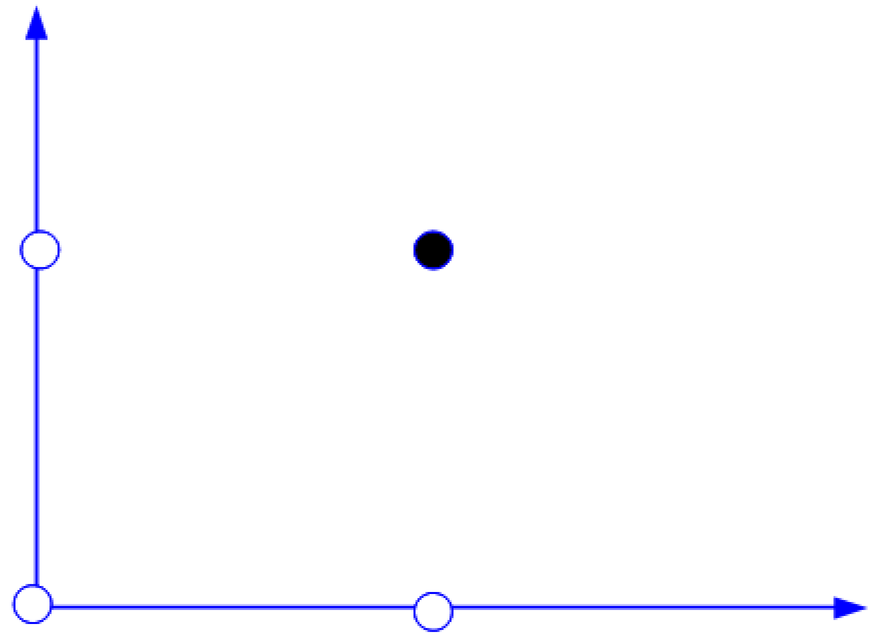
- Modelo
  - $\bar{y}_i = \varphi(\boldsymbol{w}^T \boldsymbol{x}_i)$

- Regra de Aprendizagem
  - $w = w - \alpha \frac{\partial J}{\partial w}$
  - $\boldsymbol{w} = \boldsymbol{w} + \alpha \frac{1}{n} \sum_{i=1}^{n} e_i \boldsymbol{x}_i$

Perceptron

- Problemas linearmente separáveis

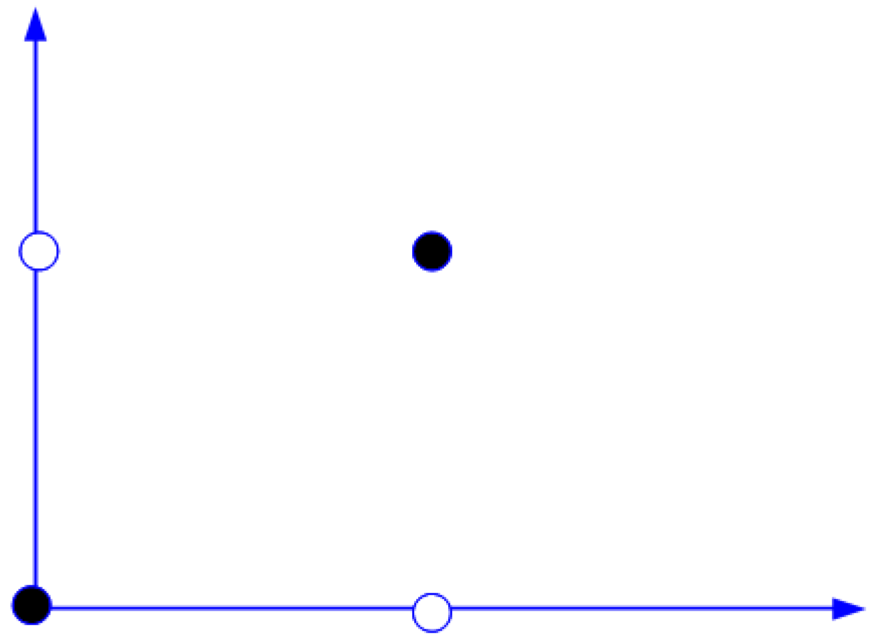
Entrada (x)	Saída (y)
0,0	0
0,1	0
1,0	0
1,1	1



Perceptron

- Minsky e Papert (1969)
  - Problema ou-exclusivo

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

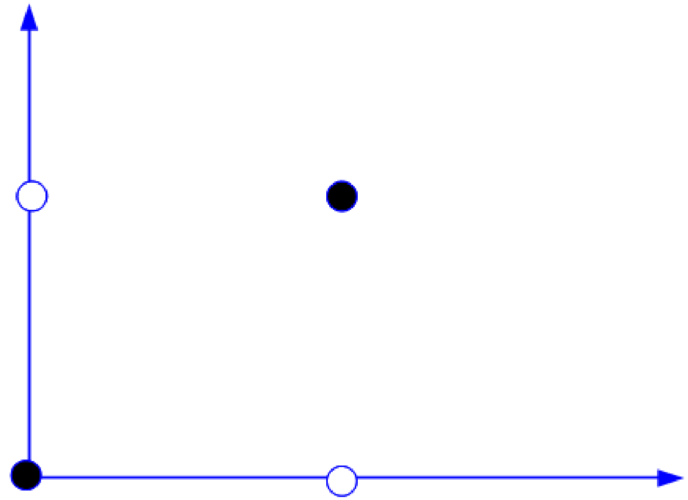


Perceptron

- Minsky e Papert (1969)
  - Problema ou-exclusivo
- Provaram que pode ser resolvido se for utilizada mais de uma camada de neurônios

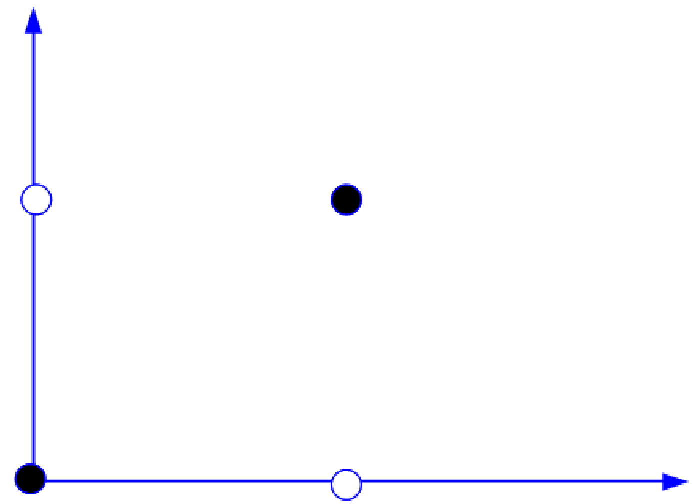
Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0



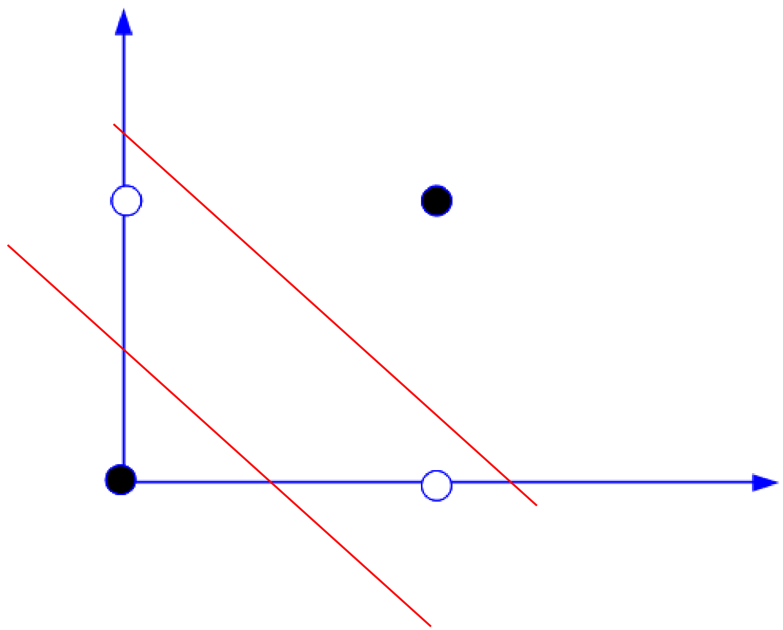
Perceptron

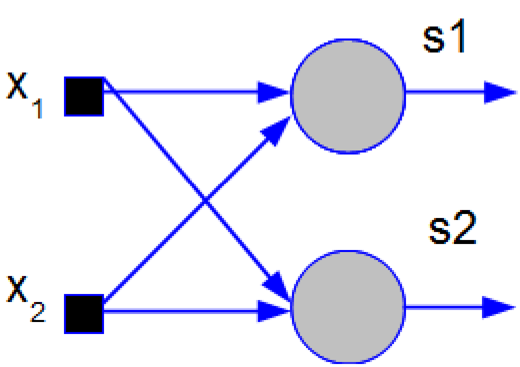
Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0



Perceptron

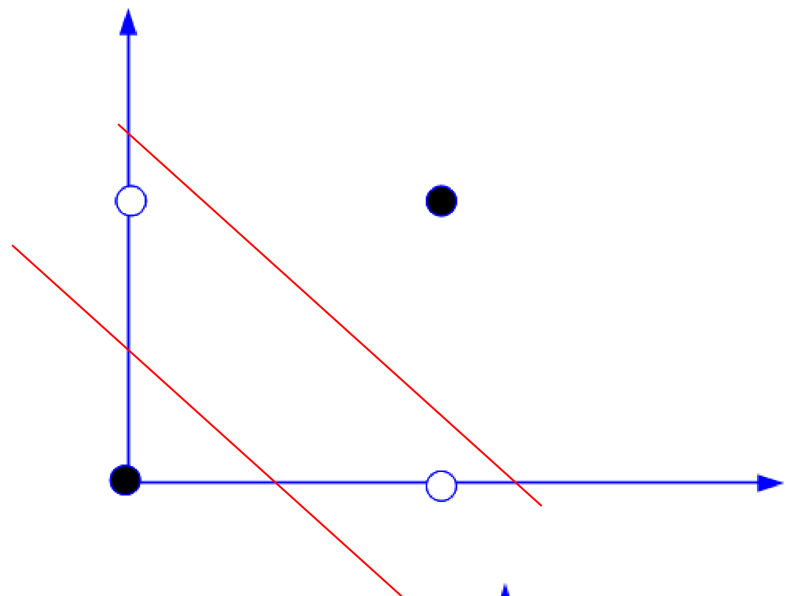
Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

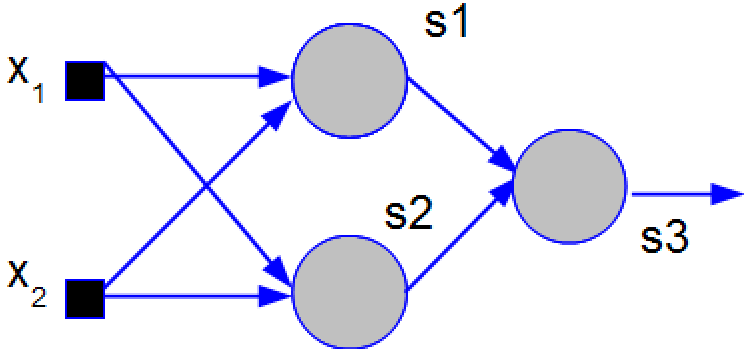


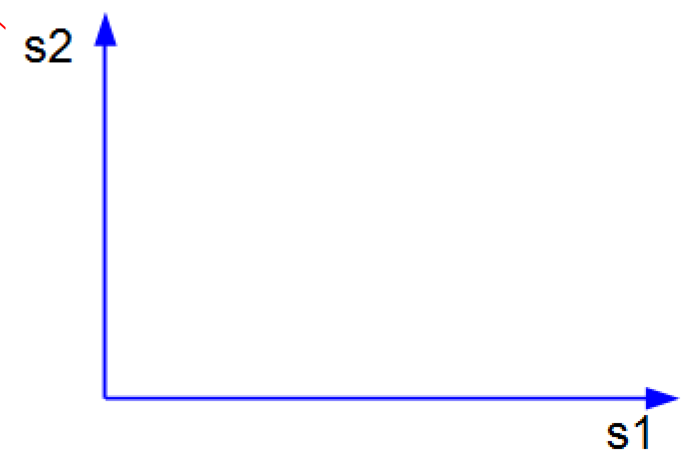


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

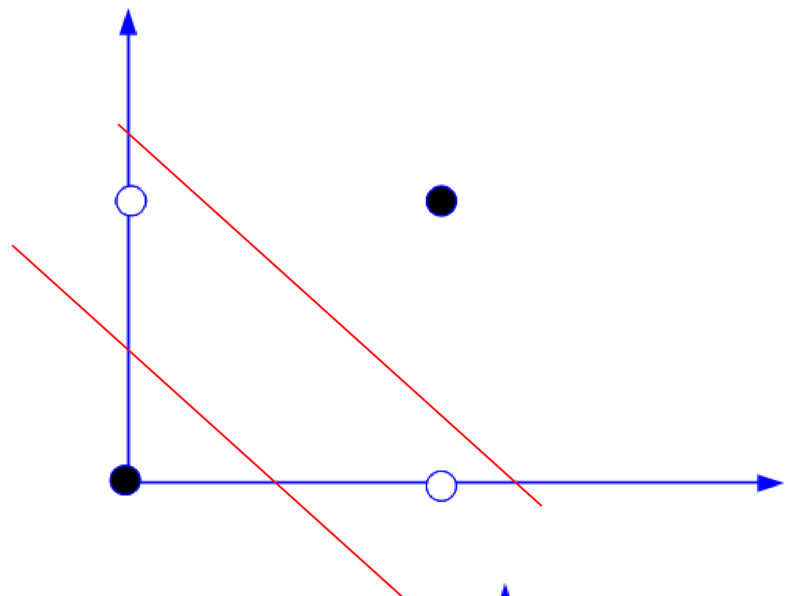


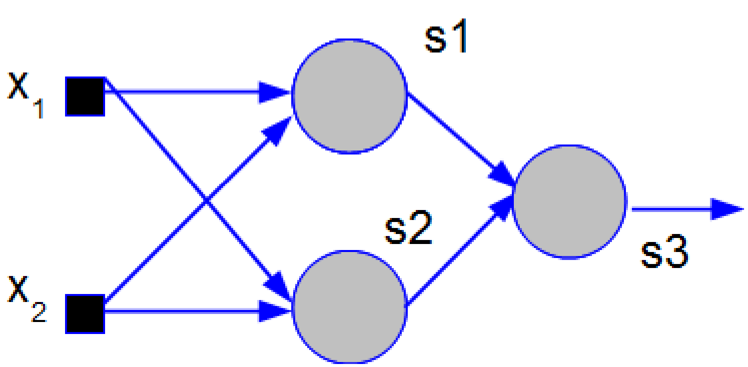


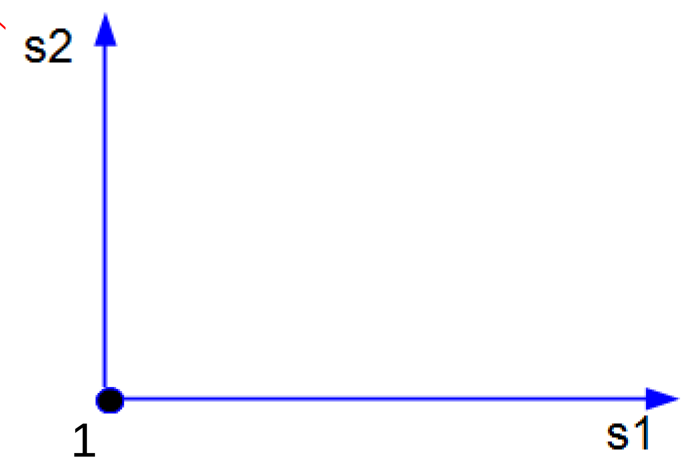


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

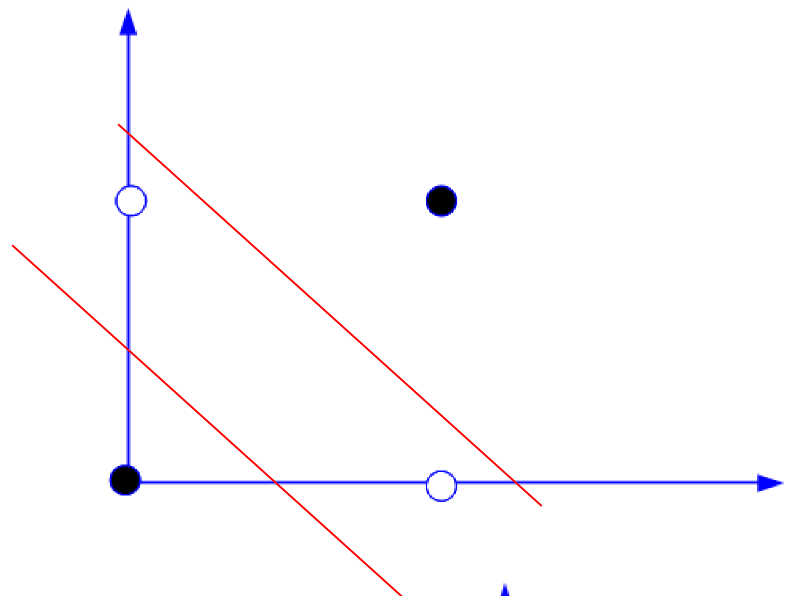


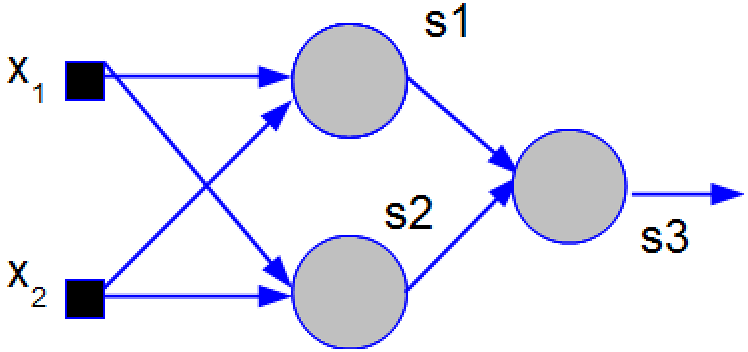


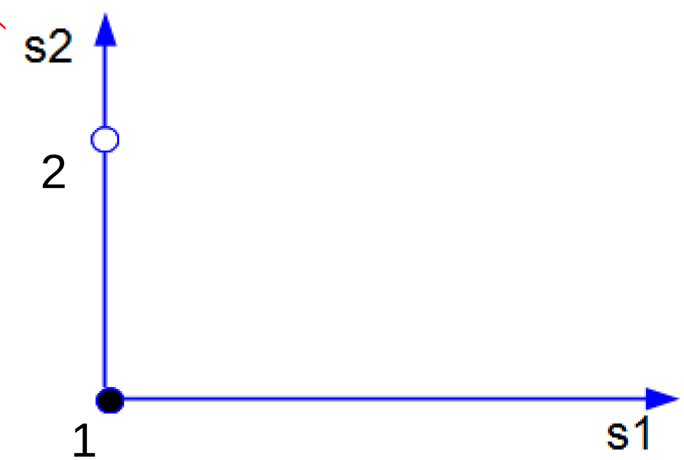


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

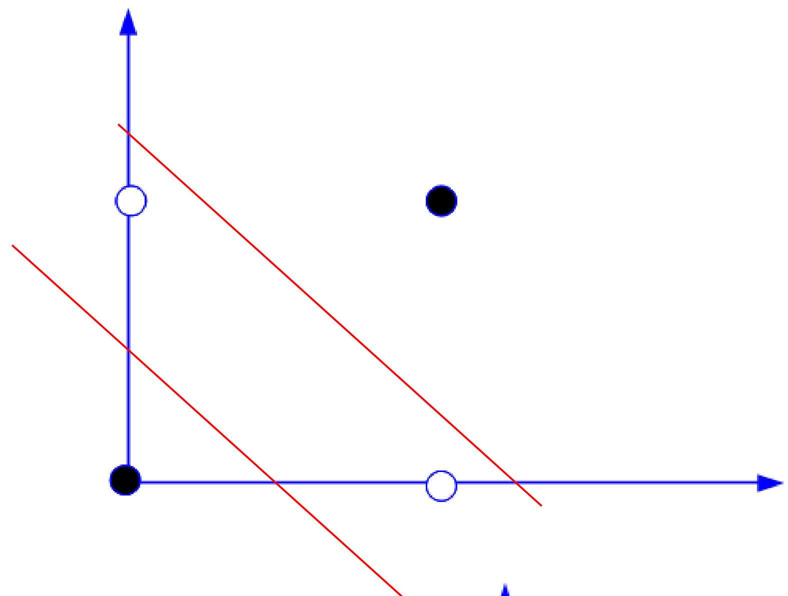


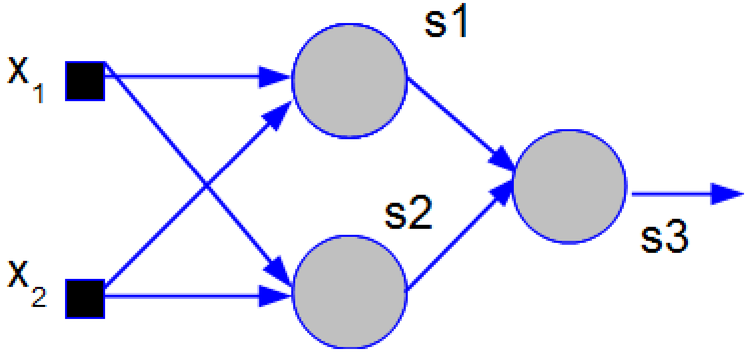


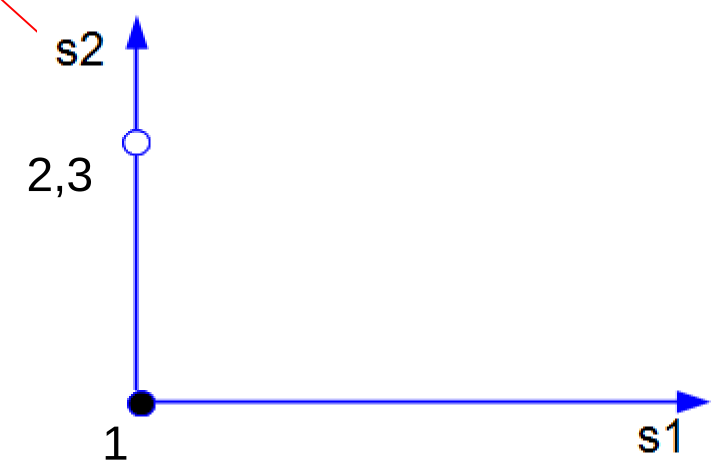


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

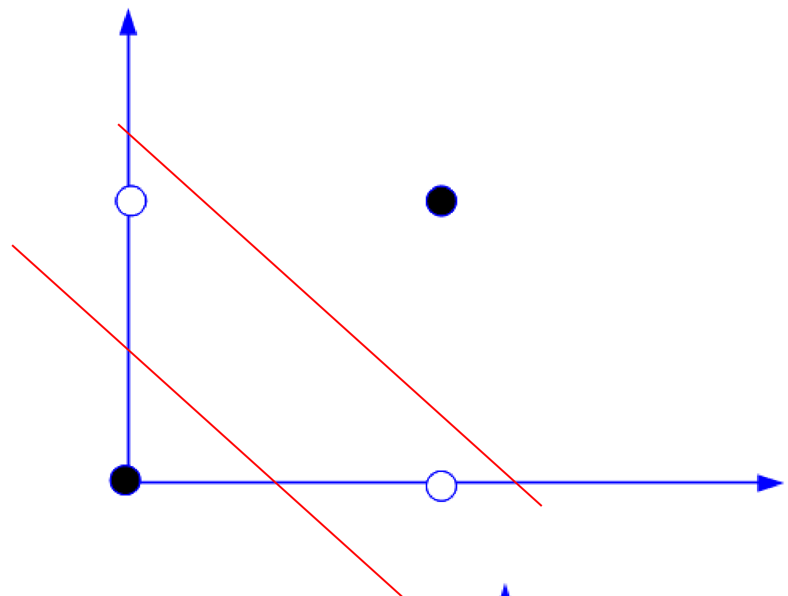


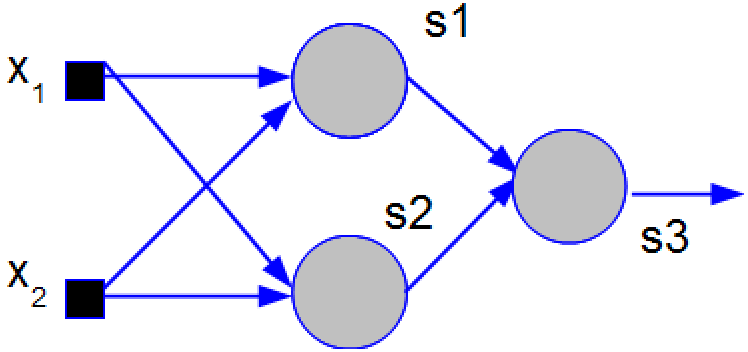


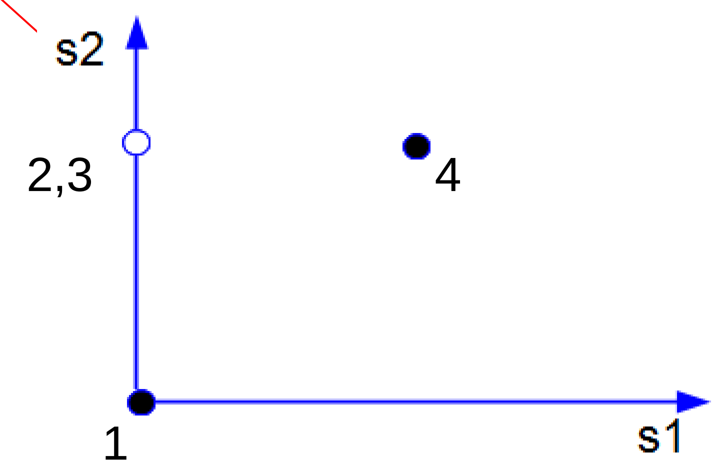


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0

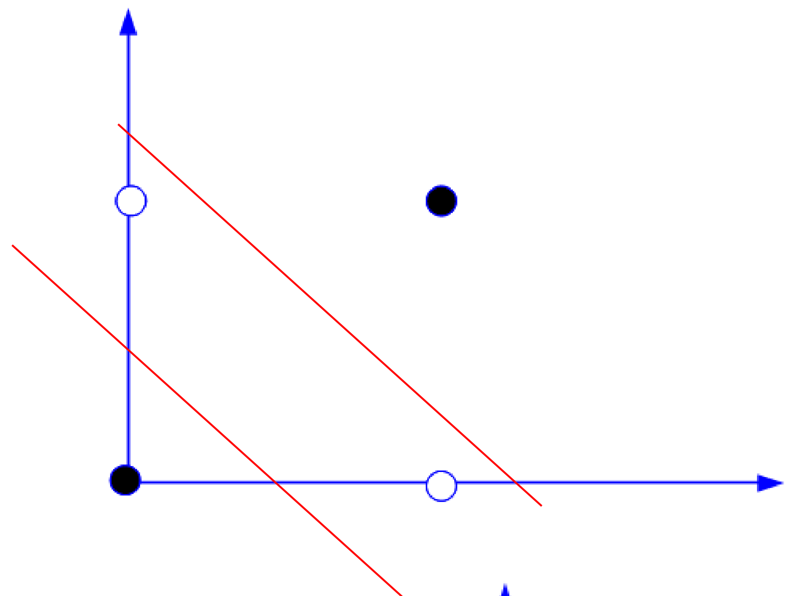


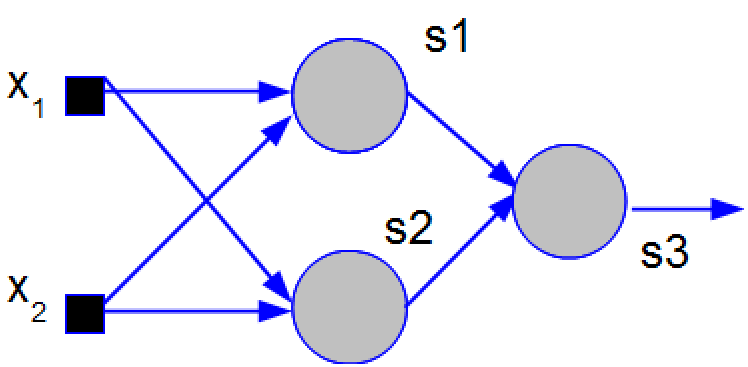


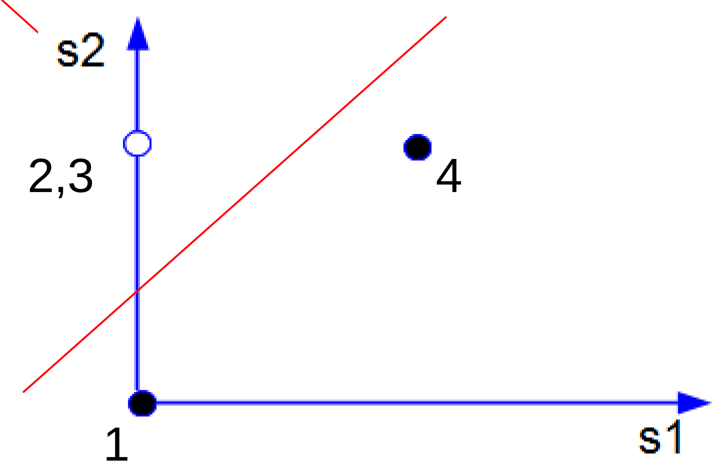


Perceptron

Entrada (x)	Saída (y)
0,0	0
0,1	1
1,0	1
1,1	0







Perceptron de Múltiplas Camadas

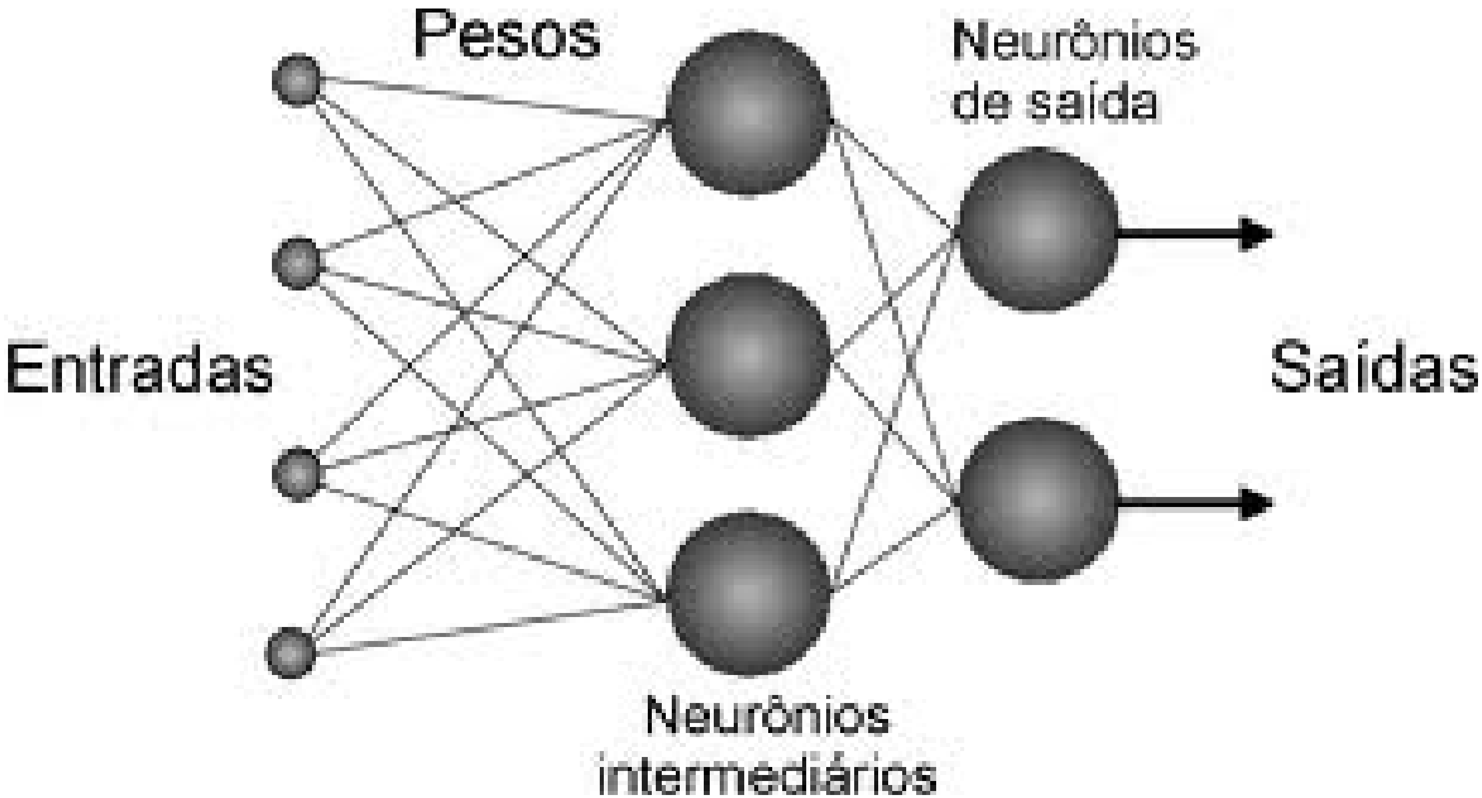
- Rede MLP (MultiLayer Perceptron)
  - Problemas não linearmente separáveis

Redes MLP

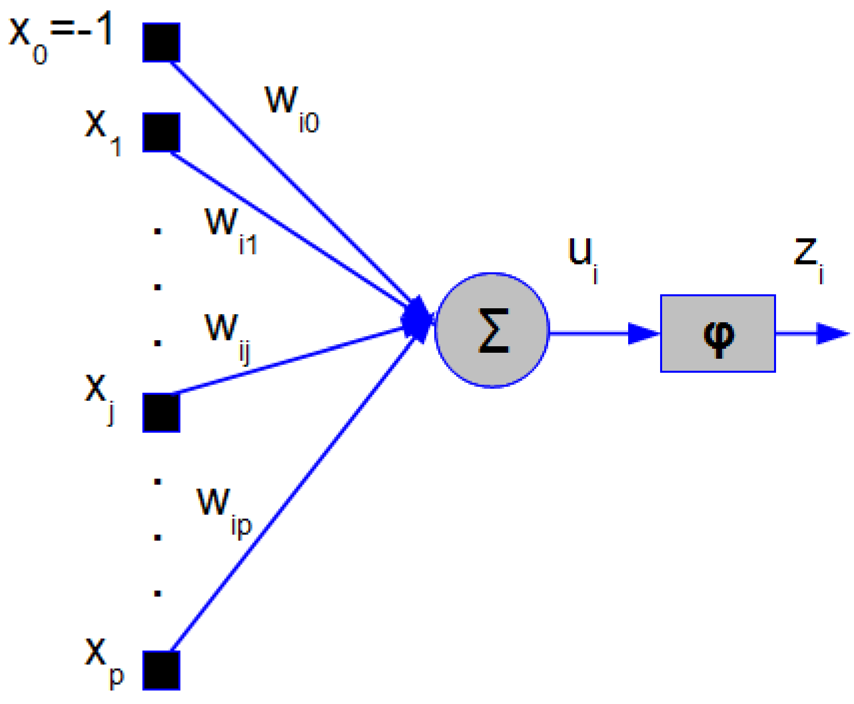
Redes Neurais

Redes MLP

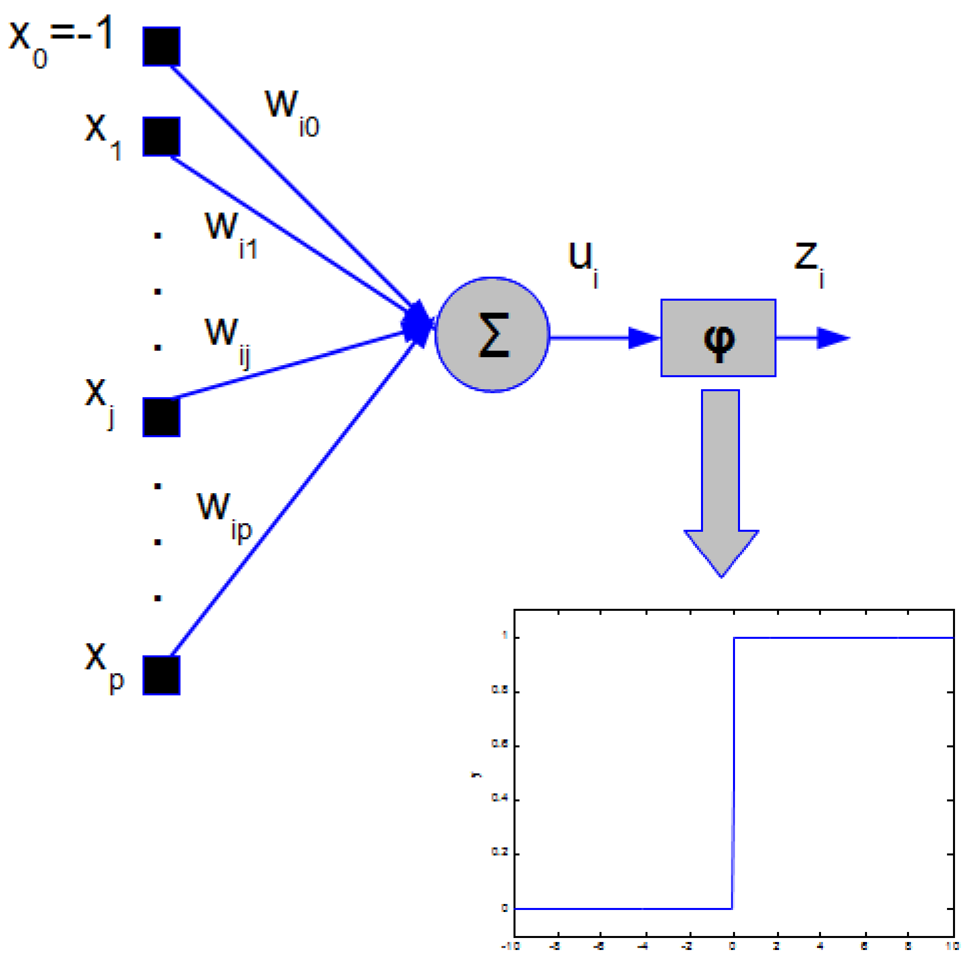
- Redes com múltiplas camadas de neurônios artificiais



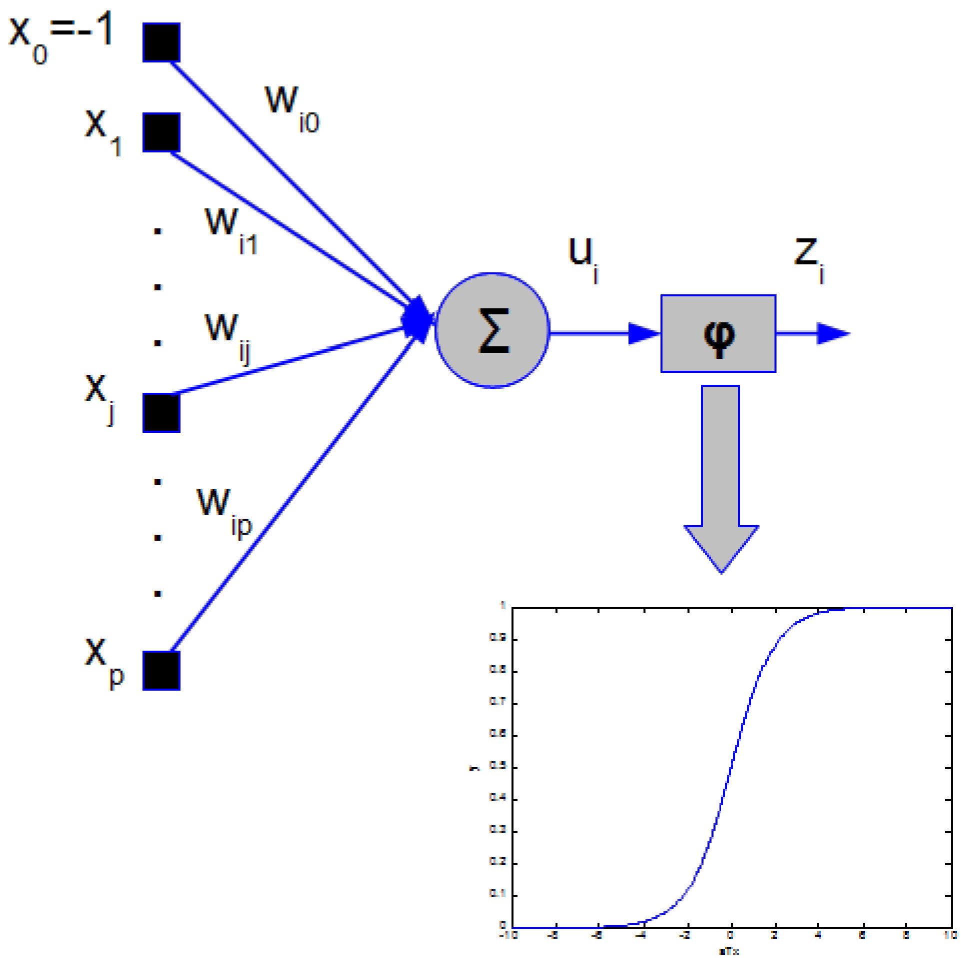
Neurônio Artificial



Neurônio Artificial

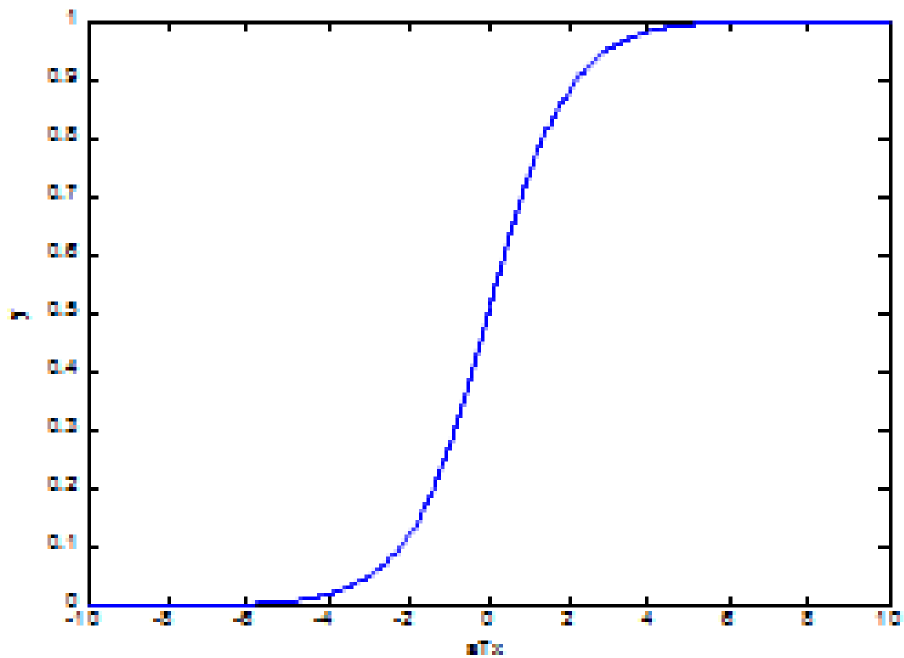


Neurônio Artificial

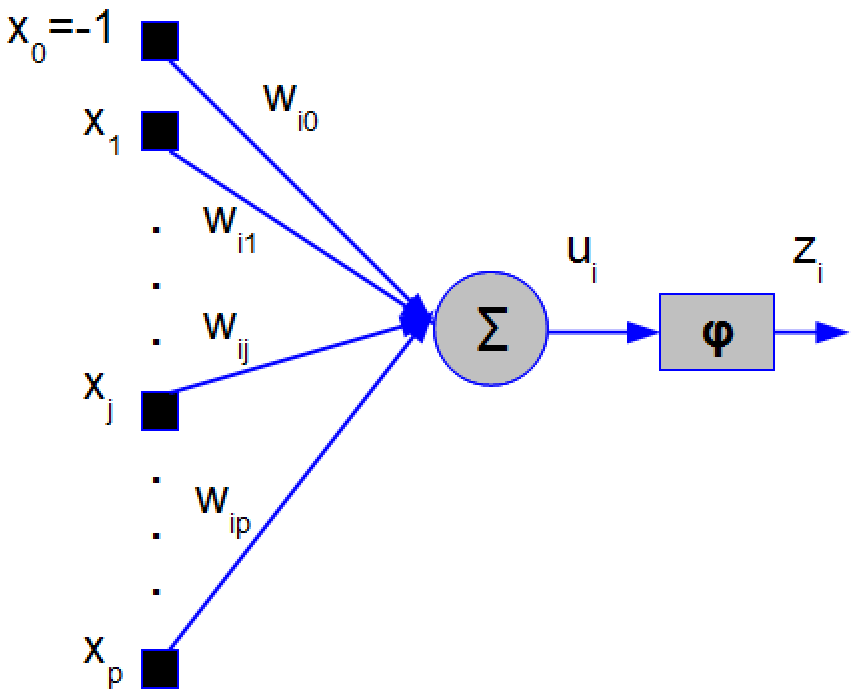


Neurônio Artificial

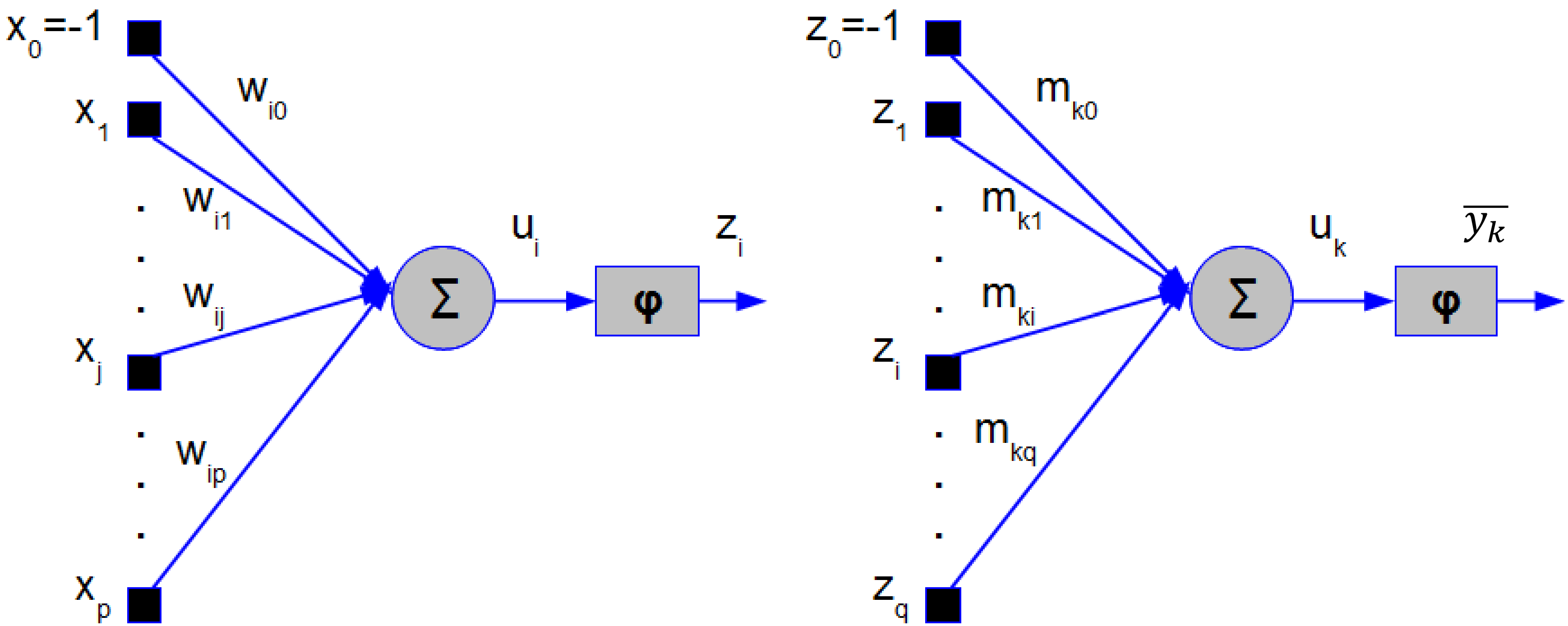
- Função Logística
  - $f(x) = \frac{1}{1+e^{-x}}$
  - $f'(x) = f(x)[1 - f(x)]$



Rede MLP

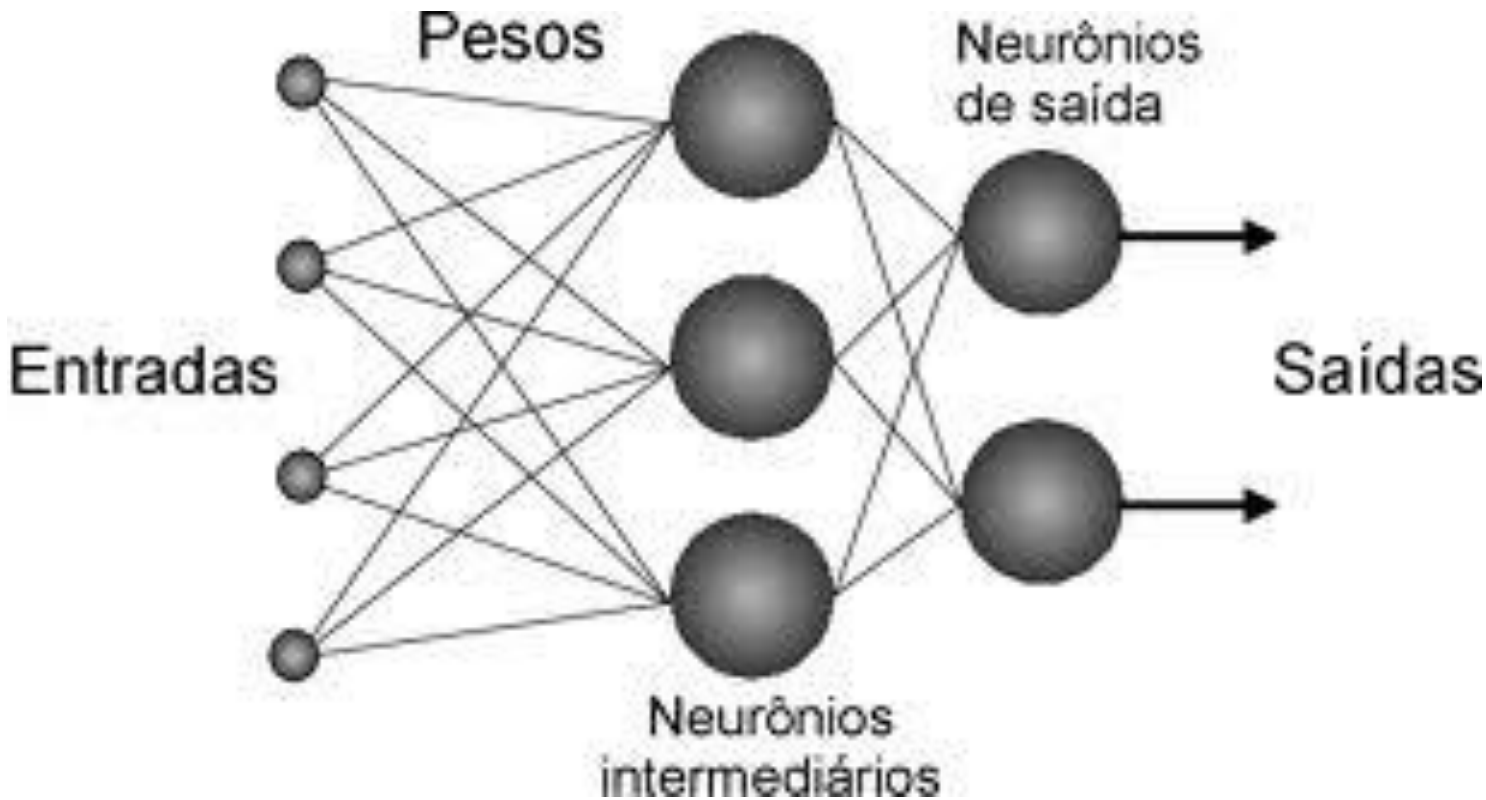


Rede MLP



x_0=-1
x_1
x_j
x_p
w_{i0}
w_{i1}
w_{ij}
w_{ip}
Σ
u_i
φ
z_i
z_0=-1
z_1
z_i
z_q
m_{k0}
m_{k1}
m_{ki}
m_{kq}
Σ
u_k
φ
$\overline{y_k}$

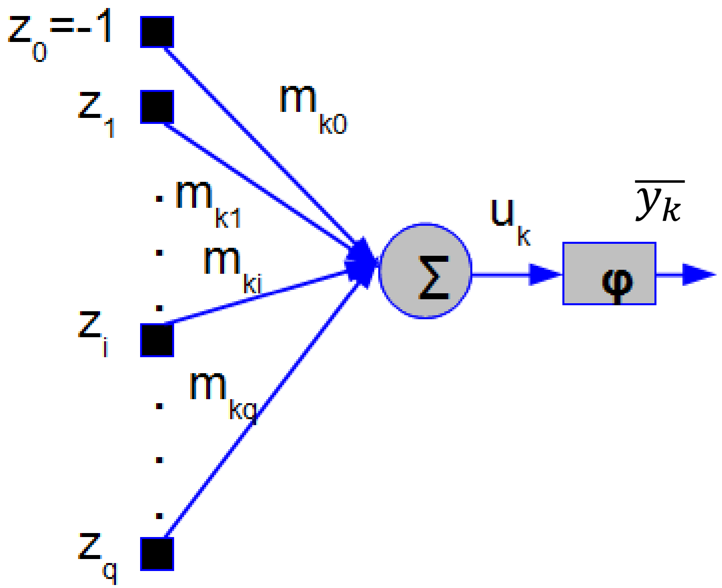
Rede MLP



Pesos
Neurônios
de saída
Entradas
Saídas
Neurônios
intermediários

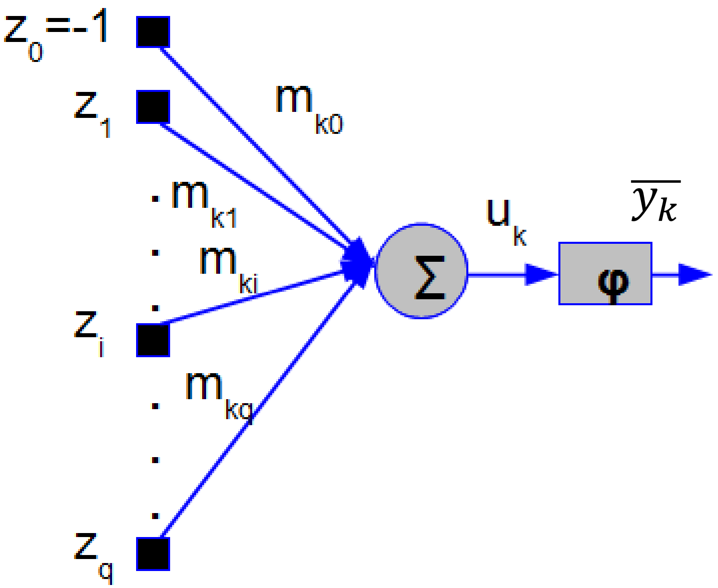
Rede MLP

- Atualização dos pesos da camada de saída
  - $m = m - \alpha \frac{\partial J}{\partial m}$
  - $J(m_{ki}) = \frac{1}{2}\{[y_k - \overline{y_k}]^2\}$



Rede MLP

- Atualização dos pesos da camada de saída
  - $m = m - \alpha \dfrac{\partial J}{\partial m}$
  - $J(m_{ki}) = \frac{1}{2}\{[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)]^2\}$



Rede MLP

- Atualização dos pesos da camada de saída
  - $m = m - \alpha \frac{\partial J}{\partial m}$
  - $J(m_{ki}) = \frac{1}{2}\{[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)]^2\}$
  - $\frac{\partial J}{\partial m_{ki}} = \frac{1}{2} 2[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)][(-1)\varphi'(\sum_{i=0}^{q} m_{ki}\, z_i) z_i]$

Rede MLP

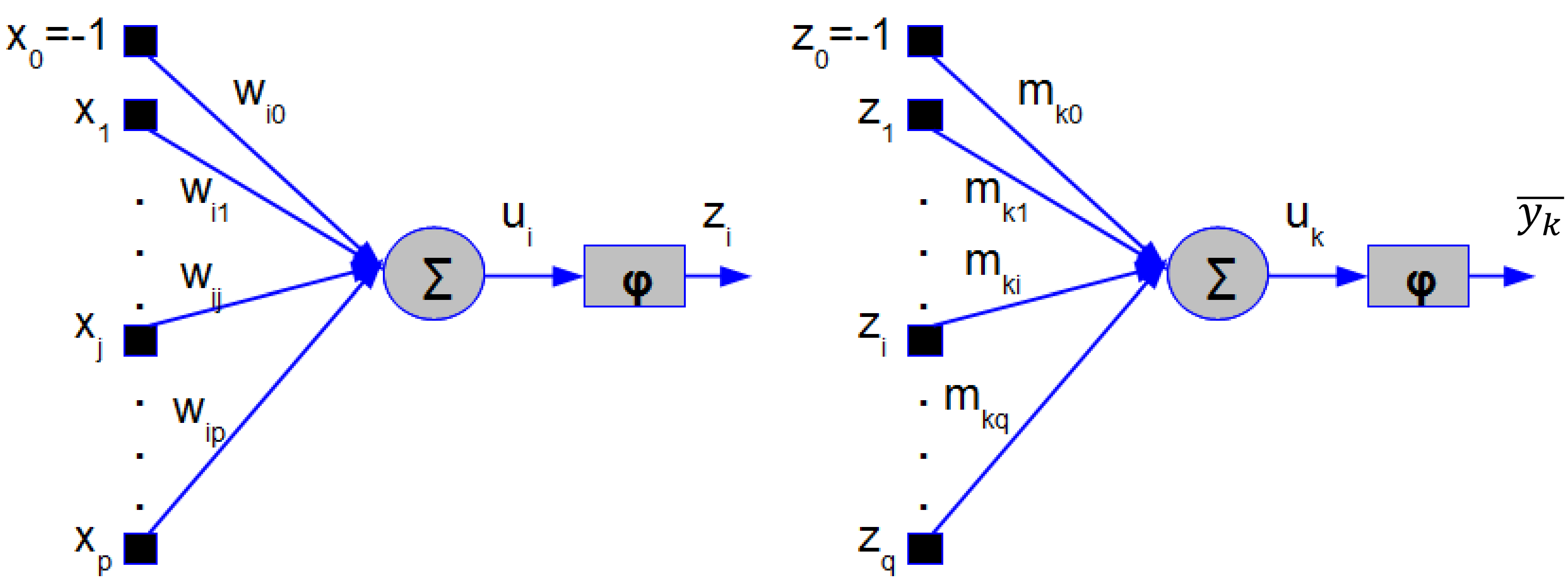
- Atualização dos pesos da camada de saída
  - $m = m - \alpha \frac{\partial J}{\partial m}$
  - $J(m_{ki}) = \frac{1}{2}\{[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)]^2\}$
  - $\frac{\partial J}{\partial m_{ki}} = \frac{1}{2} 2[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)][(-1)\varphi'(\sum_{i=0}^{q} m_{ki}\, z_i) z_i]$
  - $\frac{\partial J}{\partial m_{ki}} = -e_k \varphi'(u_k) z_i$

Rede MLP

- Atualização dos pesos da camada de saída
  - $m = m - \alpha \frac{\partial J}{\partial m}$
  - $J(m_{ki}) = \frac{1}{2}\{[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)]^2\}$
  - $\frac{\partial J}{\partial m_{ki}} = \frac{1}{2} 2[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)][(-1)\varphi'(\sum_{i=0}^{q} m_{ki}\, z_i) z_i]$
  - $\frac{\partial J}{\partial m_{ki}} = -e_k \varphi'(u_k) z_i$
  - $m_{ki} = m_{ki} + \alpha e_k \varphi'(u_k) z_i$

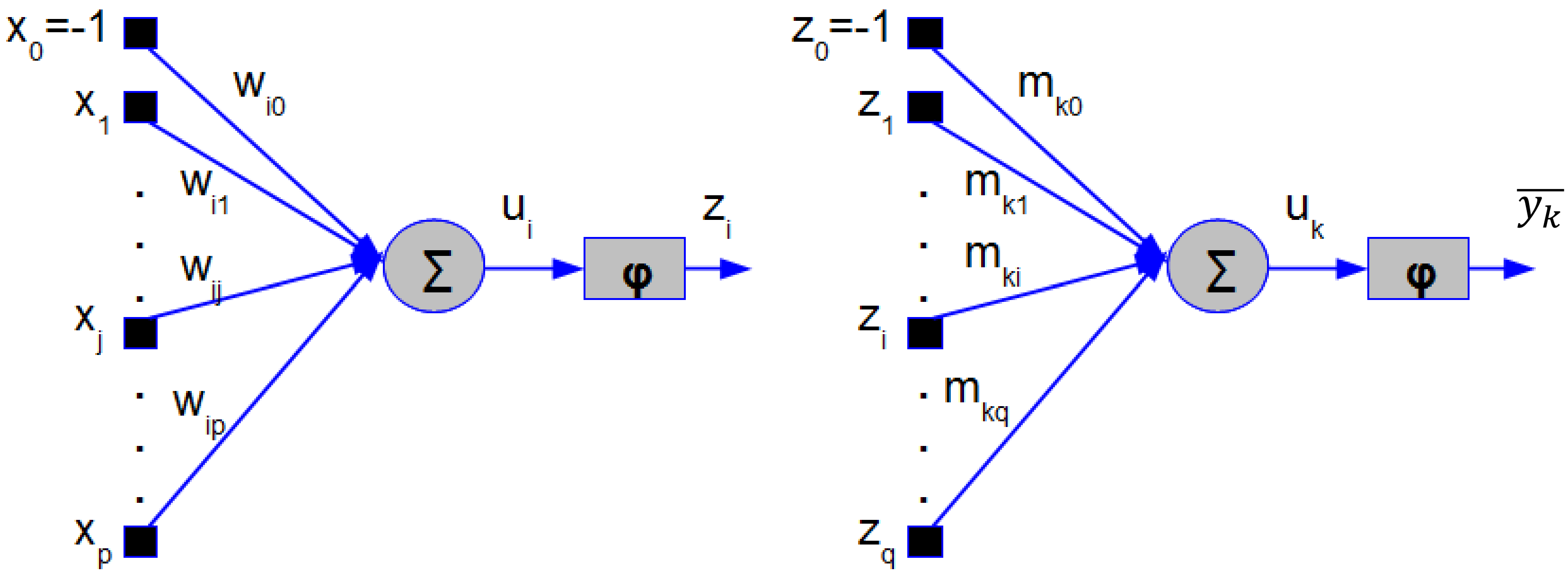
Rede MLP

- Atualização dos pesos da camada oculta
  - $w = w - \alpha \frac{\partial J}{\partial w}$
  - $J(w_{ij}) = \frac{1}{2}\left[\sum_{k=1}^{r}(y_k - \overline{y_k})^2\right]$



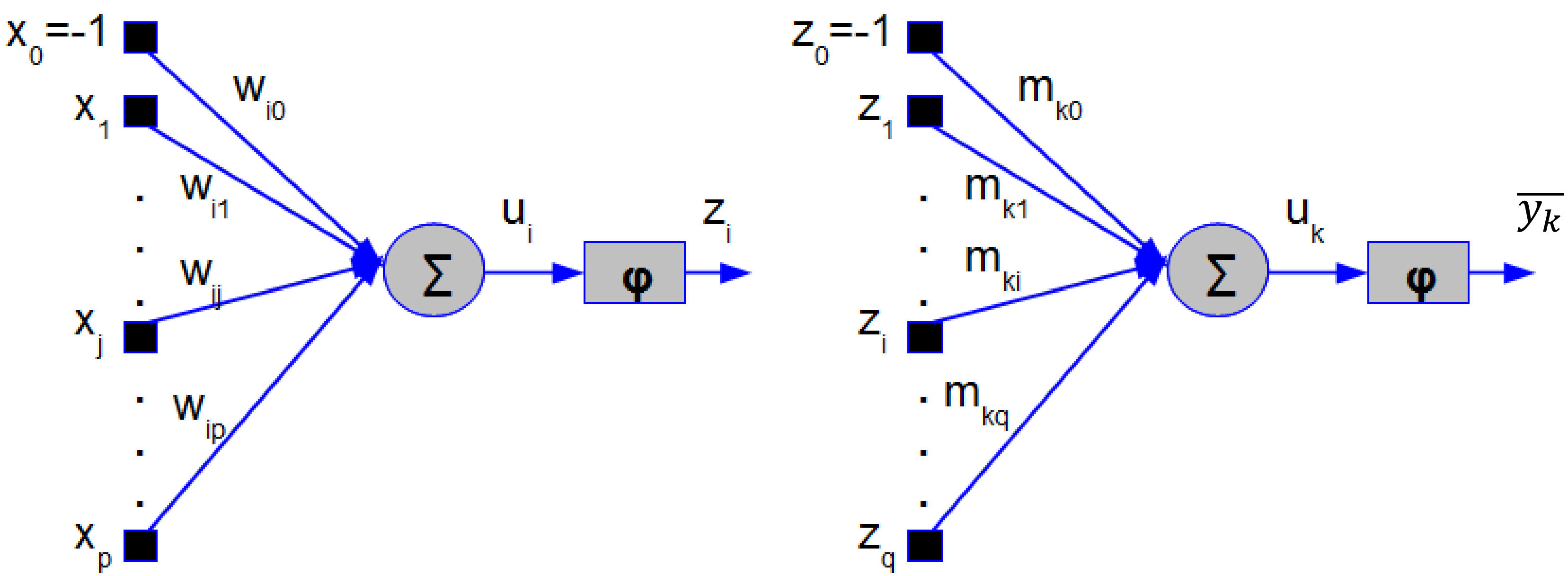
Rede MLP

- Atualização dos pesos da camada oculta
  - $\text{w} = w - \alpha \frac{\partial J}{\partial w}$
  - $J(w_{ij}) = \frac{1}{2}\{\sum_{k=1}^{r}[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, z_i)]^2\}$



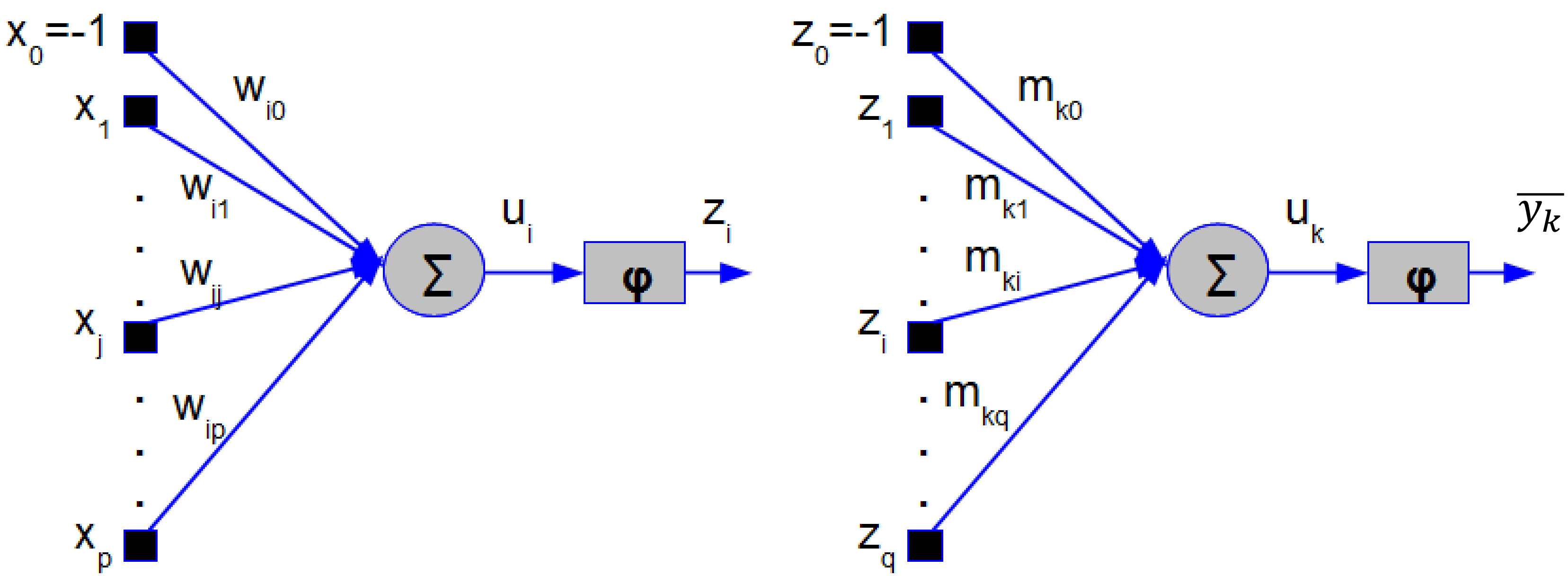
Rede MLP

- Atualização dos pesos da camada oculta
  - $w = w - \alpha \frac{\partial J}{\partial w}$
  - $J(w_{ij}) = \frac{1}{2}\{\sum_{k=1}^{r}[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, \varphi(u_i))]^2\}$



Rede MLP

- Atualização dos pesos da camada oculta
  - $w = w - \alpha \frac{\partial J}{\partial w}$
  - $J(w_{ij}) = \frac{1}{2}\{\sum_{k=1}^{r}[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, \varphi(\sum_{j=0}^{p} w_{ij} x_j))]^2\}$



Rede MLP

- Atualização dos pesos da camada oculta
  - $\mathrm{w} = w - \alpha \frac{\partial J}{\partial w}$
  - $J\left(w_{ij}\right) = \frac{1}{2}\{\sum_{k=1}^{r}[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, \varphi(\sum_{j=0}^{p} w_{ij}x_j))]^2\}$
  - $\frac{\partial J}{\partial w_{ij}} = -\varphi'(u_i)x_j \sum_{k=1}^{r} e_k\varphi'(u_k)\, m_{ki}$

Rede MLP

- Atualização dos pesos da camada oculta
  - $\mathrm{w} = w - \alpha \frac{\partial J}{\partial w}$
  - $J\left(w_{ij}\right) = \frac{1}{2}\{\sum_{k=1}^{r}[y_k - \varphi(\sum_{i=0}^{q} m_{ki}\, \varphi(\sum_{j=0}^{p} w_{ij} x_j))]^2\}$
  - $\frac{\partial J}{\partial w_{ij}} = -\varphi'(u_i) x_j \sum_{k=1}^{r} e_k \varphi'(u_k)\, m_{ki}$
  - $w_{ij} = w_{ij} + \alpha\varphi'(u_i) x_j \sum_{k=1}^{r} e_k \varphi'(u_k)\, m_{ki}$

Backpropagation

- $w_{ij} = w_{ij} + \alpha\varphi'(u_i)x_j \sum_{k=1}^{r} e_k\varphi'(u_k)\, m_{ki}$
- $m_{ki} = m_{ki} + \alpha e_k\varphi'(u_k)z_i$

Backpropagation

- $w_{ij} = w_{ij} + \alpha\varphi'(u_i)x_j \sum_{k=1}^{r} e_k\varphi'(u_k)\, m_{ki}$
- $m_{ki} = m_{ki} + \alpha e_k\varphi'(u_k)z_i$
- Gradiente local
  - $\delta_k = e_k\varphi'(u_k)$

Backpropagation

- $w_{ij} = w_{ij} + \alpha\varphi'(u_i)x_j \sum_{k=1}^{r} \delta_k \, m_{ki}$
- $m_{ki} = m_{ki} + \alpha\delta_k z_i$
- Gradiente local
  - $\delta_k = e_k\varphi'(u_k)$

Backpropagation

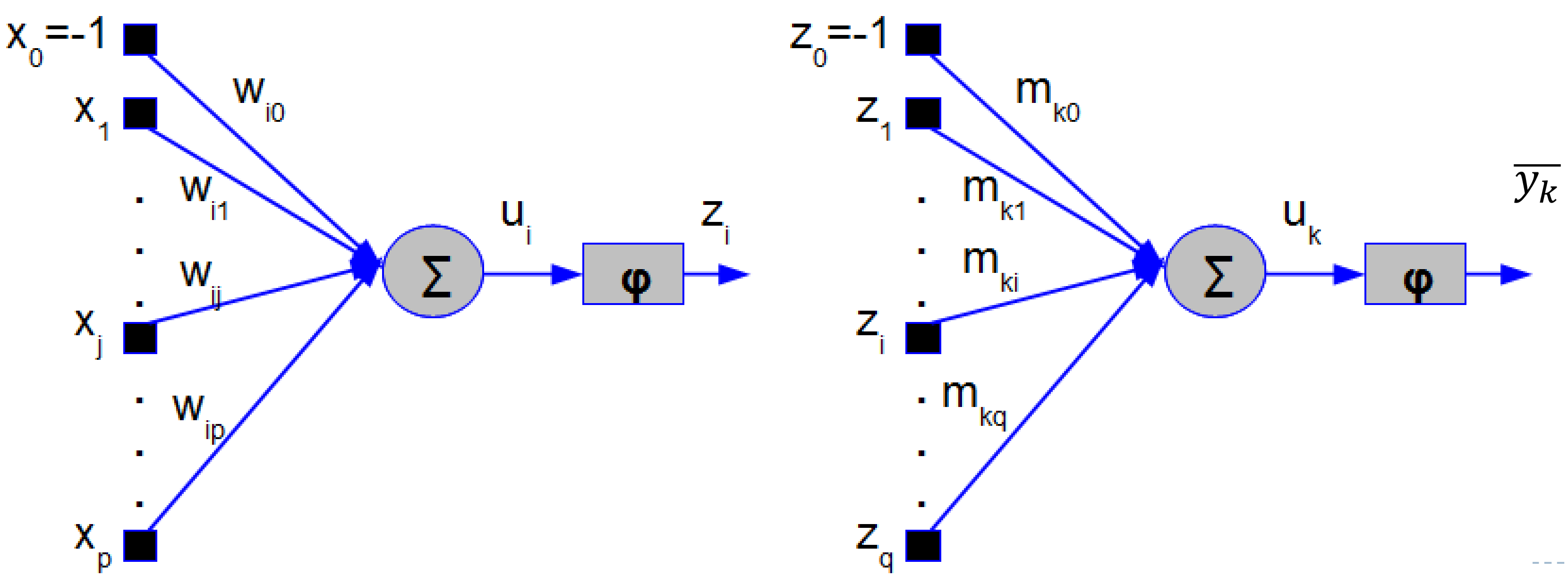
- $w_{ij} = w_{ij} + \alpha\varphi'(u_i)x_j \sum_{k=1}^{r} \delta_k \, m_{ki}$
- $m_{ki} = m_{ki} + \alpha\delta_k z_i$
- Gradiente local
  - $\delta_k = e_k\varphi'(u_k)$
  - $\delta_i = \varphi'(u_i) \sum_{k=1}^{r} \delta_k \, m_{ki}$

Backpropagation

- $w_{ij} = w_{ij} + \alpha\delta_i x_j$
- $m_{ki} = m_{ki} + \alpha\delta_k z_i$
- Gradiente local
  - $\delta_k = e_k \varphi'(u_k)$
  - $\delta_i = \varphi'(u_i) \sum_{k=1}^{r} \delta_k \, m_{ki}$

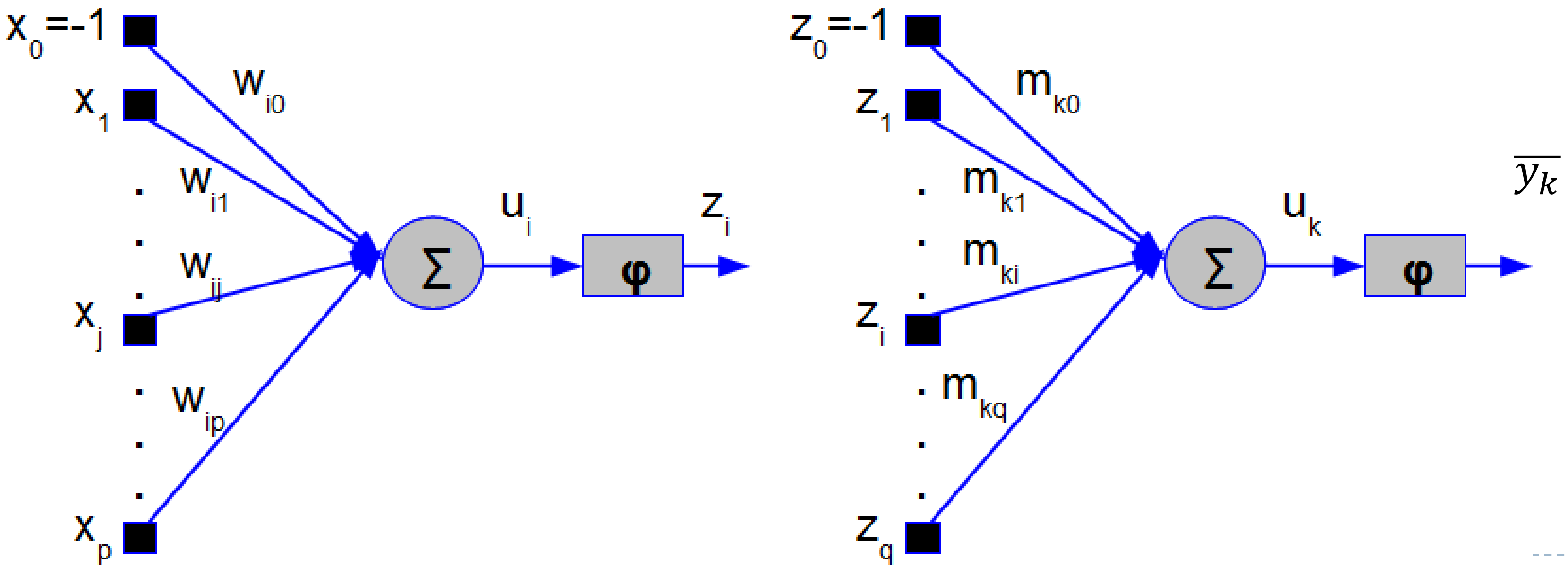
MLP - Treinamento

- Inicializa os pesos com valores ente 0 e 1
- Duas fases
  - Sentido direto
  - Sentido inverso



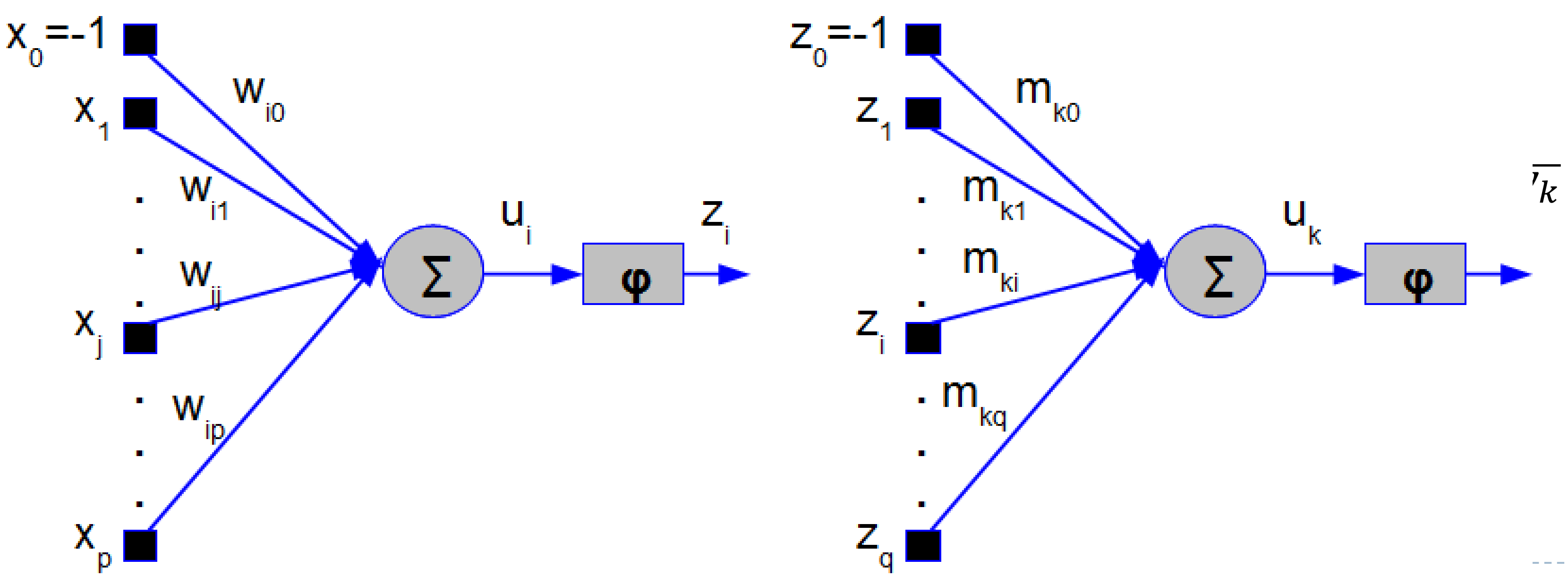
MLP - Treinamento

- Para cada amostra de treinamento
  - Sentido direto
  - Calcula $\overline{y_k}$
  - Calcula $e_k = y_k - \overline{y_k}$



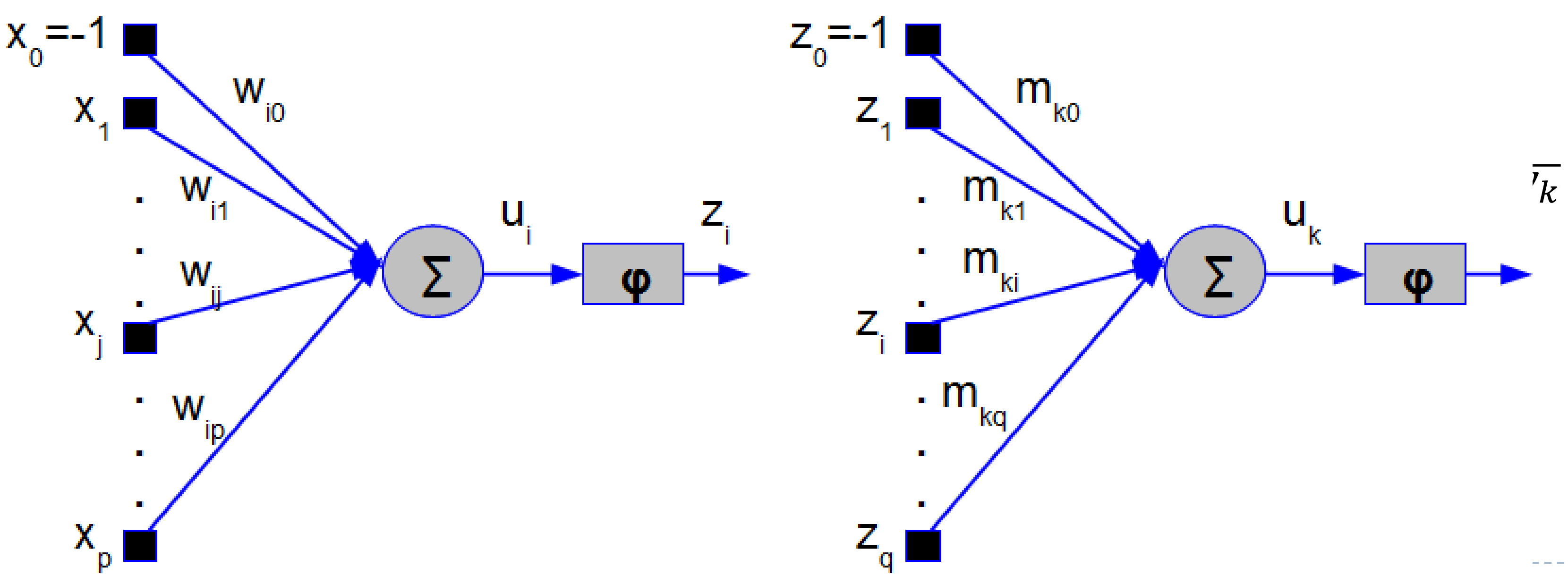
MLP - Treinamento

- Para cada amostra de treinamento
  - Sentido inverso
  - Calcula os gradientes locais
    - $\delta_k = e_k \varphi'(u_k)$
    - $\delta_i = \varphi'(u_i) \sum_{k=1}^{r} \delta_k \, m_{ki}$



MLP - Treinamento

- Para cada amostra de treinamento
  - Sentido inverso
  - Atualiza os pesos
    - $w_{ij} = w_{ij} + \alpha\delta_i x_j$
    - $m_{ki} = m_{ki} + \alpha\delta_k z_i$



Dúvidas ?