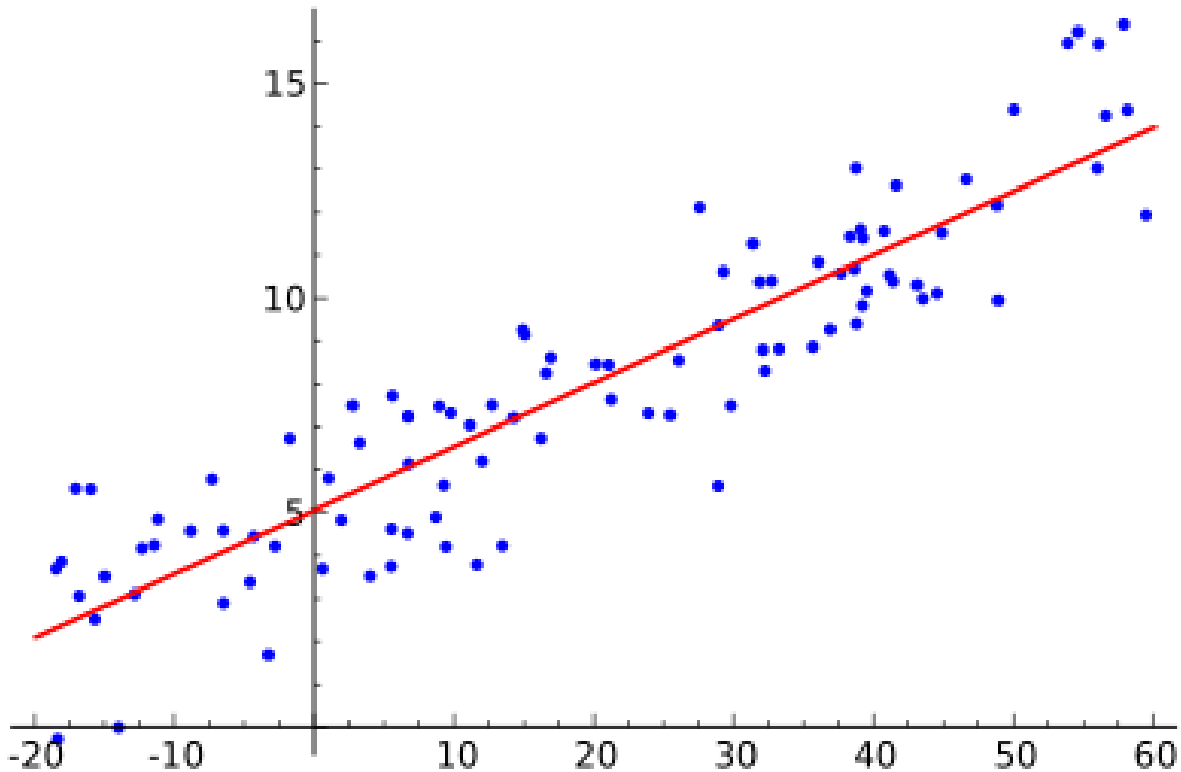
MAC5921 – Deep Learning

Aula 02 – 15/08/2023

Nina S. T. Hirata

Regressão linear

Exemplo com dados de entrada $\mathbf{x} \in \mathbb{R}$



Conjunto de exemplos

$$(x^{(n)}, y^{(n)}) \in \mathbb{R}^2 \qquad n = 1, 2, \ldots, N$$

Família de funções-hipótese $h : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{w}}(x) = w_0 + w_1 x, \qquad \mathbf{w} = (w_0, w_1)^T$$

Qualidade do ajuste – função de perda

$$\mathcal{L}(w_0, w_1) = \frac{1}{N} \sum_{n=1}^{N} \Big(\underbrace{\hat{y}^{(n)}}_{h(x^{(n)}) = w_0 + w_1 x^{(n)}} - y^{(n)} \Big)^2$$

$\mathbf{w} = (w_0, w_1)^T$ são os parâmetros

$(x^{(n)}, y^{(n)}) \in \mathbb{R}^2$, $n = 1, 2, \ldots, N$ são fixos

Notations: d-dimensional case ($d > 1$)

$$\tilde{\mathbf{x}} = (1, x_1, x_2, \ldots, x_d)^T \in \{1\} \times \mathbb{R}^d \longrightarrow \texttt{array (d+1,1)}$$

$$\mathbf{w} = (w_0, w_1, w_1, \ldots, w_d)^T \in \mathbb{R}^{d+1} \longrightarrow \texttt{array (d+1,1)}$$

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \begin{bmatrix} w_0 & w_1 & \ldots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \ldots \\ x_d \end{bmatrix} = \mathbf{w}^T \tilde{\mathbf{x}}$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)} \right)^2$$

Regressão linear – solução analítica

Matriz de dados

$$\mathbf{X} = \begin{bmatrix} \tilde{\mathbf{x}}^{(1)\mathsf{T}} \\ \tilde{\mathbf{x}}^{(2)\mathsf{T}} \\ \vdots \\ \tilde{\mathbf{x}}^{(N)\mathsf{T}} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$$

Solução

$$\mathbf{w} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

Regressão linear – solução iterativa

Gradiente de $\mathcal{L}$

$$\nabla \mathcal{L}(\mathbf{w}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial \mathcal{L}}{\partial w_1}(\mathbf{w}) \\ \cdots \\ \frac{\partial \mathcal{L}}{\partial w_d}(\mathbf{w}) \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sum_n (\hat{y}^{(n)} - y^{(n)})\, x_i^{(n)}$$

Initial weight: $\mathbf{w}(0)$

Weight update rule (iteration r):

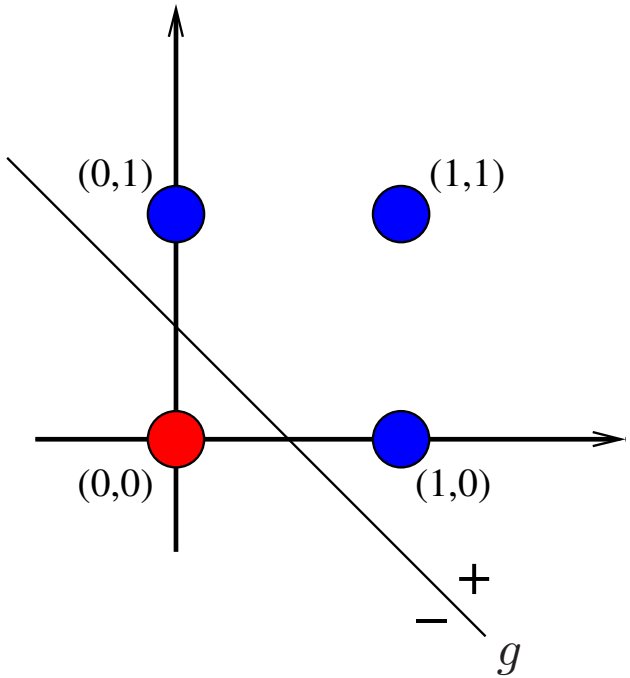
$$\Delta \mathbf{w}(r) = -\nabla \mathcal{L}(\mathbf{w})$$

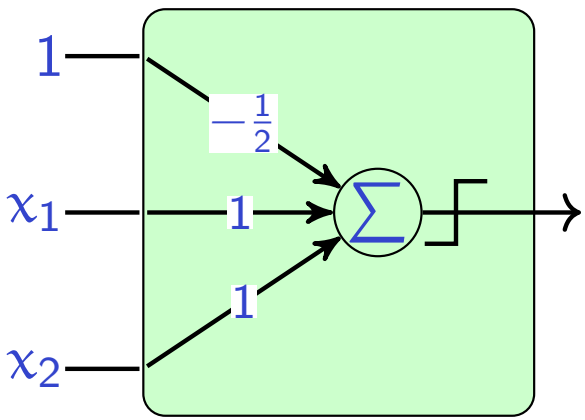
$$\mathbf{w}(r+1) = \mathbf{w}(r) + \eta \Delta \mathbf{w}(r)$$

η : learning rate (e.g, 0.001)

Problemas de classificação binária

x_1	x_2	$\phi(g(x_1, x_2))$
0	0	0
0	1	1
1	0	1
1	1	1





$$g(x_1, x_2) = x_1 + x_2 - \frac{1}{2}$$

Perceptron

Caso as classes sejam linearmente separáveis, o seguinte algoritmo pode ser usado

Algorithm 1 Perceptron

Input: $D = \{(\mathbf{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\} : n = 1, \ldots, N\}$

Output: $\mathbf{w} \in \mathbb{R}^{1+d}$

$\mathbf{w} \leftarrow$ small random value

while there exists $(\mathbf{x}, y)$ in D such that $\text{sign}(\mathbf{w}^T \tilde{\mathbf{x}}) \neq y$ **do**

$\mathbf{w} \leftarrow \mathbf{w} + y\,\tilde{\mathbf{x}}$

end while

return $\mathbf{w}$

Regressão logística – classificação binária

Quando as classes não são linearmente separáveis, podemos usar regressão logística

Família de funções-hipótese $h : \mathbb{R}^d \to [0, 1]$

$h_{\mathbf{w}}(\mathbf{x}) = \theta(\mathbf{w}^T \tilde{\mathbf{x}})$ para aproximar $P(y = 1|\mathbf{x})$

$$\theta(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1}$$

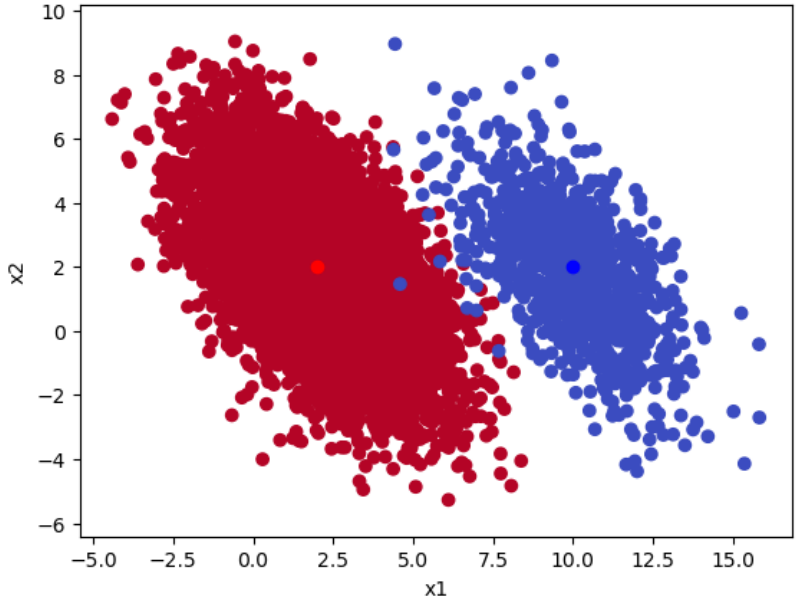
Regressão logística

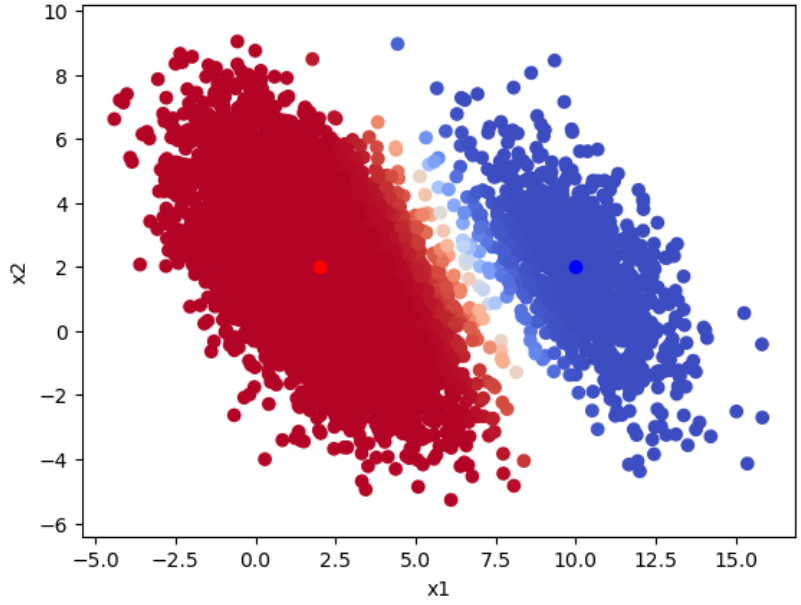
Considerando $y = 1$ a classe positiva e $y = 0$ a classe negativa

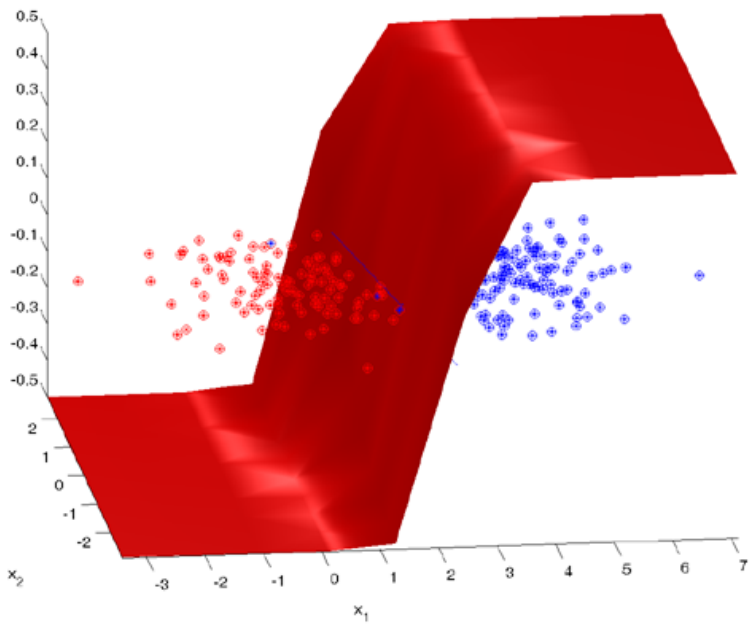
Função de perda

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N} \sum_{n=1}^{N} y^{(n)} \ln \hat{y}^{(n)} + (1 - y^{(n)}) \ln(1 - \hat{y}^{(n)})$$

com $\hat{y}^{(n)} = \theta(\mathbf{w}^T \tilde{\mathbf{x}}^{(n)})$







Source: http://strijov.com/sources/demoDataGen.php

Problema de classificação multiclasses

Conjunto de exemplos

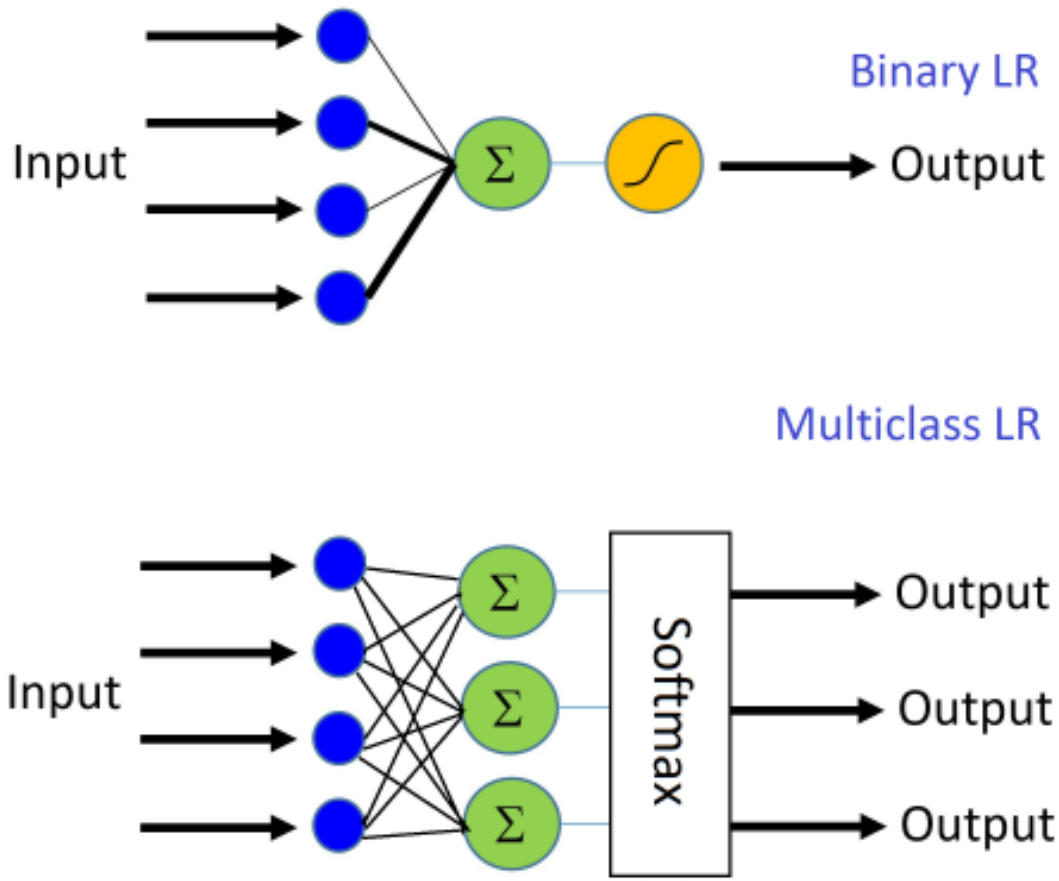
$$(\mathbf{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times C \qquad n = 1, 2, \ldots, N$$

K rótulos de classe (categoria): $C = \{c_1, c_2, \ldots, c_K\}$

Para cada $\mathbf{x}$, quero saber $P(y = c_j|\mathbf{x})$, $j = 1, \ldots, K$

Função `softmax`

$$\hat{p}_j = \hat{P}(y = c_j|\mathbf{x}) = \frac{e^{\mathbf{w}_j^T \tilde{\mathbf{x}}}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T \tilde{\mathbf{x}}}}, \quad j = 1, 2, \ldots, K$$



Fonte: https://www.cntk.ai/pythondocs/CNTK_103B_MNIST_LogisticRegression.html

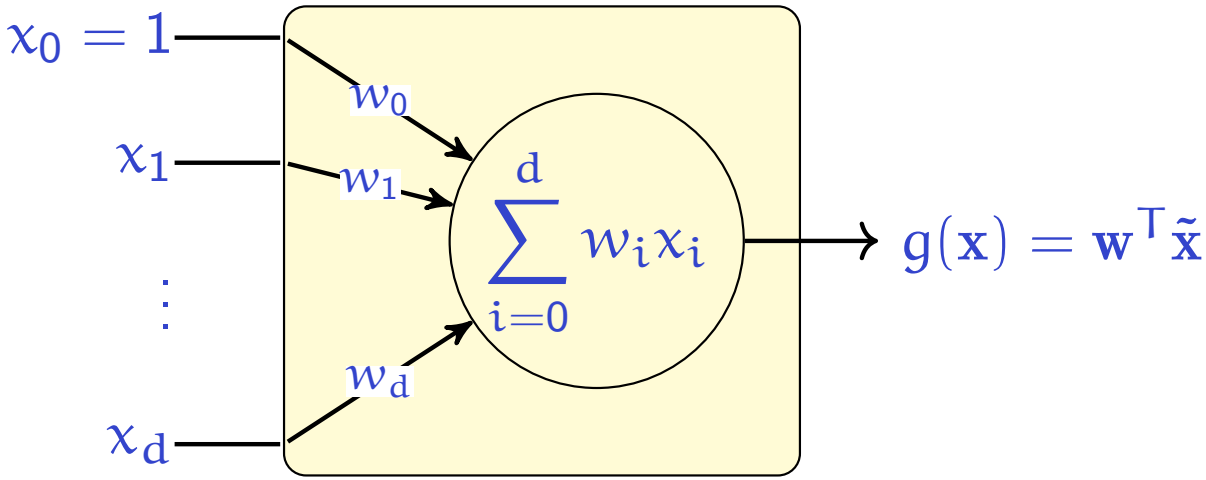
Cross-entropy loss

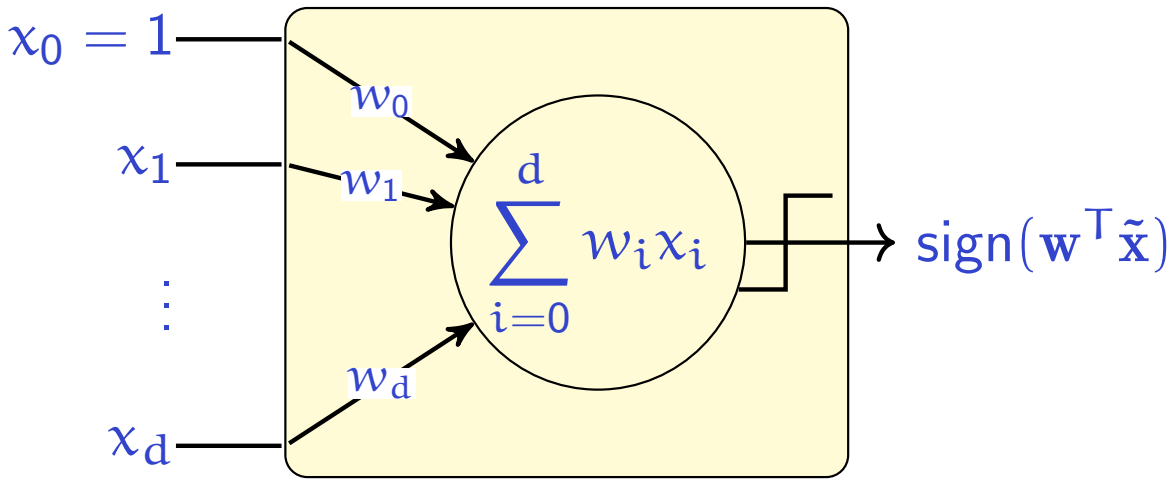
$$\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{j=1}^{K} y_j^{(n)} \log \hat{p}_j^{(n)}$$

$$\hat{p}_j^{(n)} = \hat{P}(y^{(n)} = j \,|\, \tilde{\mathbf{x}}^{(n)}) = \frac{e^{\mathbf{w}_j^T \tilde{\mathbf{x}}^{(n)}}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T \tilde{\mathbf{x}}^{(n)}}}$$

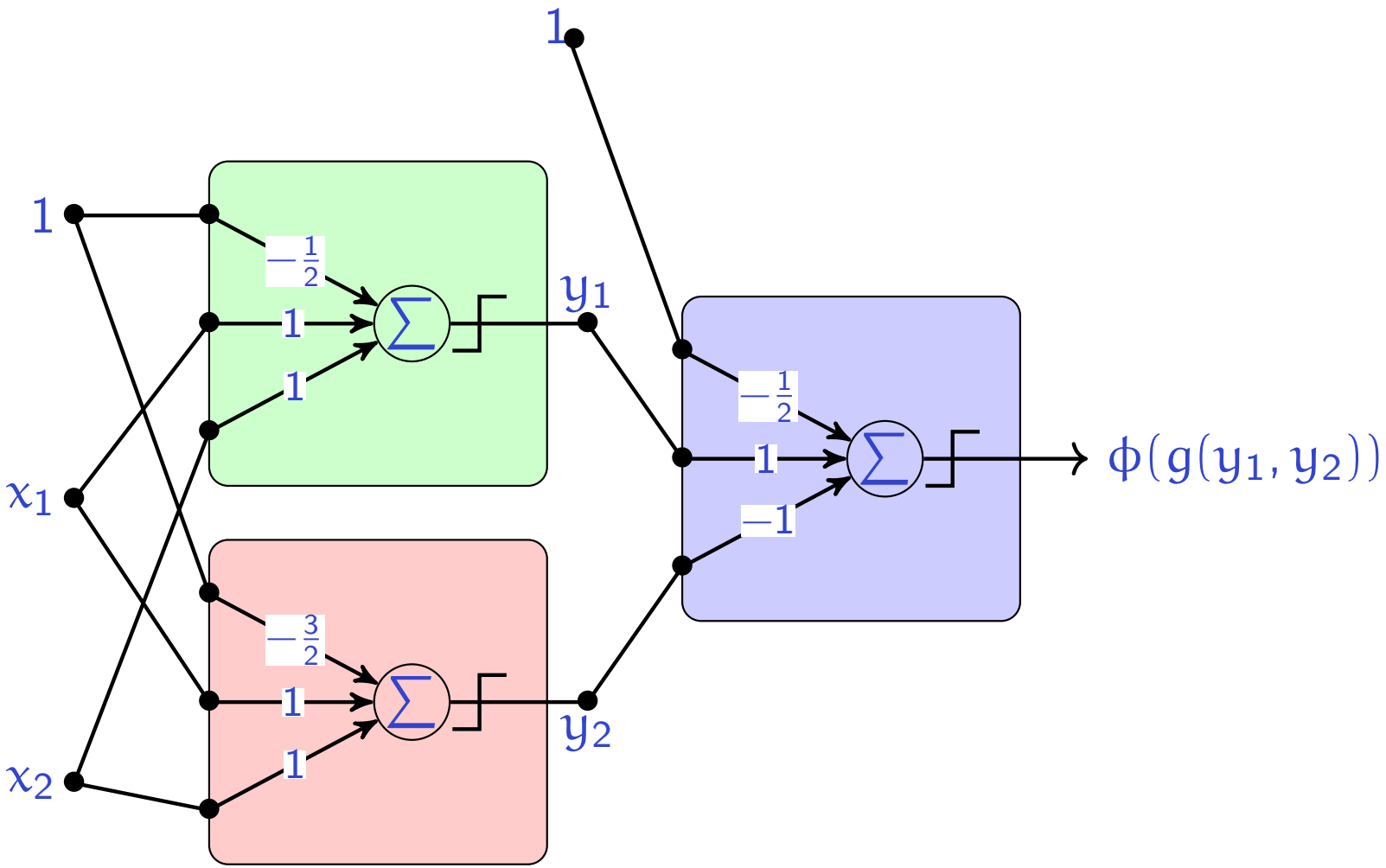
$$\sum_{j=1}^{K} \hat{p}_j^{(i)} = 1$$

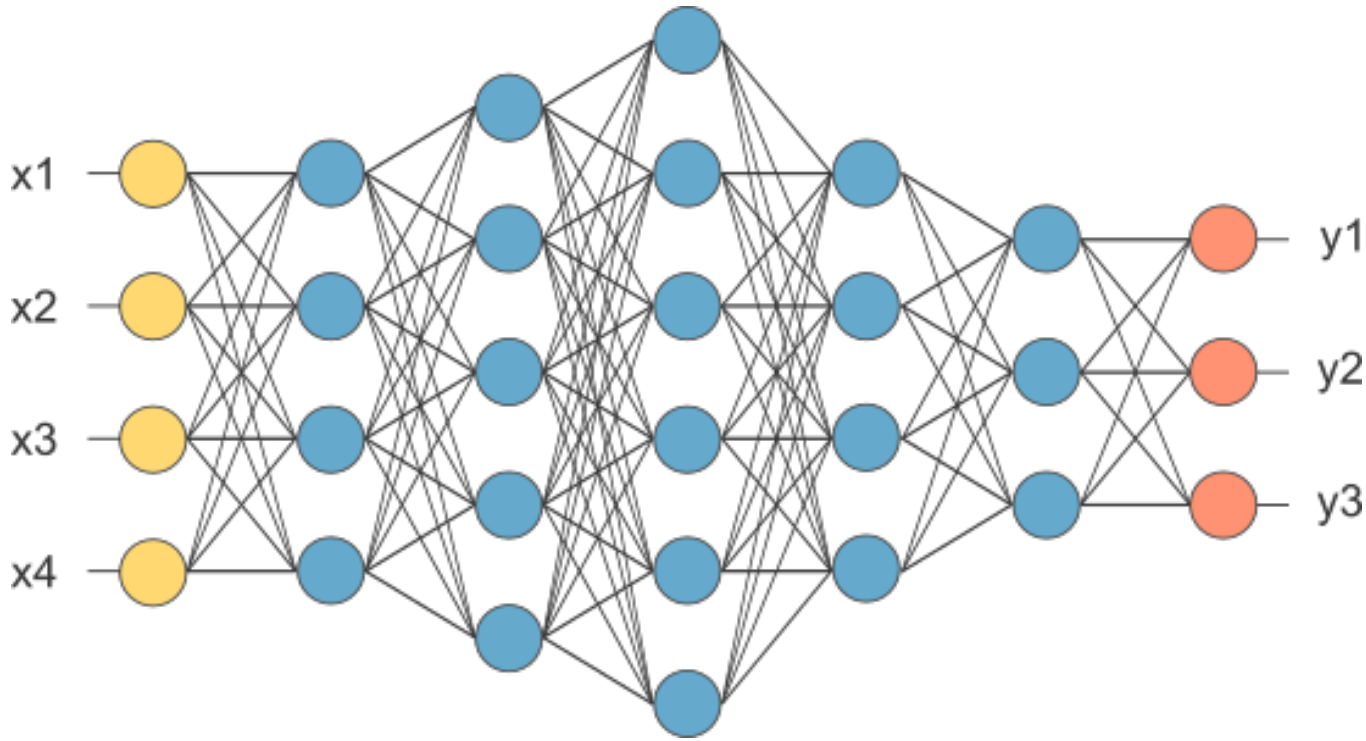
Modelos lineares: neurônios ??





Conectando neurônios




x1
x2
x3
x4
y1
y2
y3