

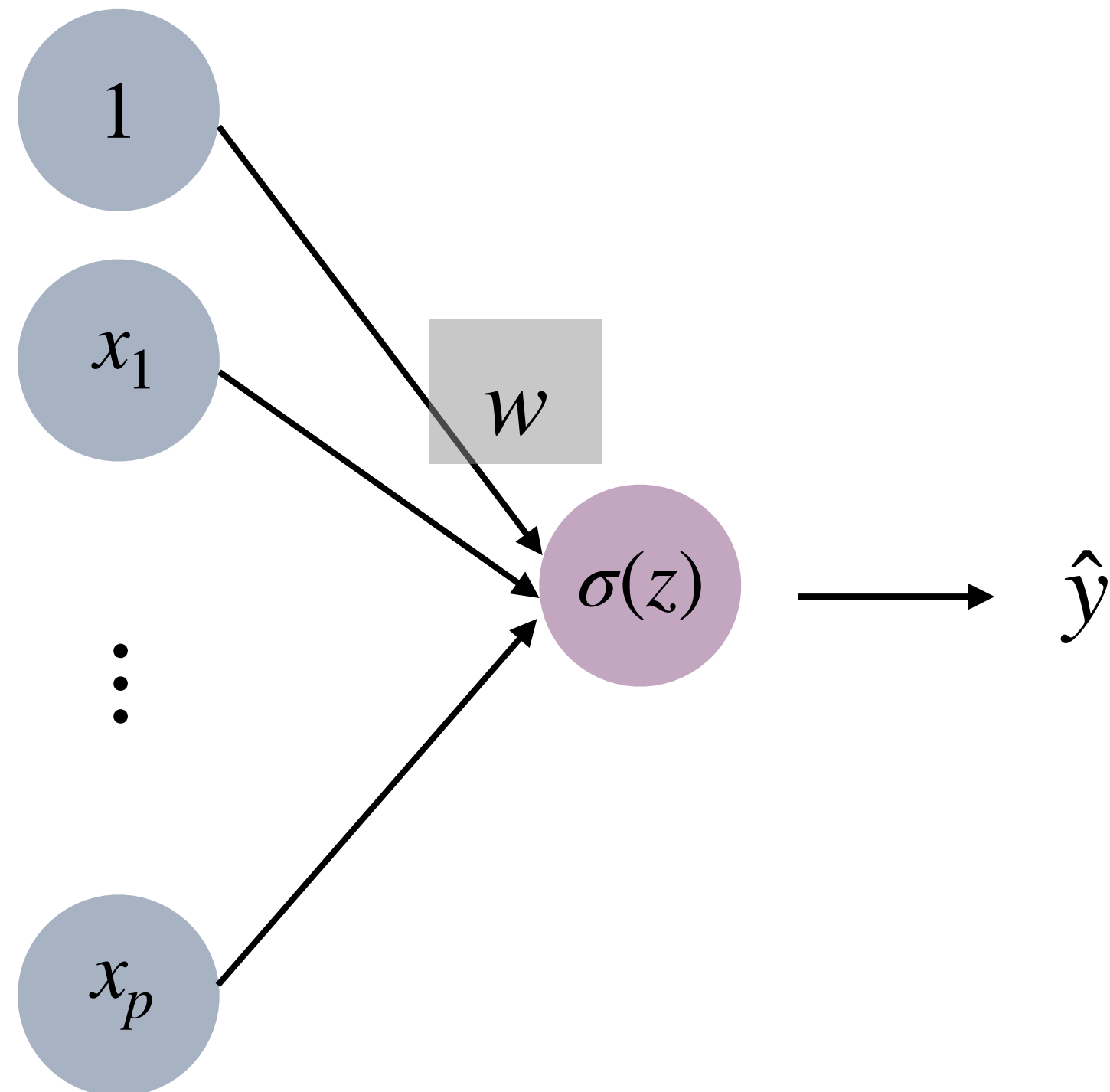
Aprendizagem estatística em altas dimensões

Florencia Leonardi

Conteúdo

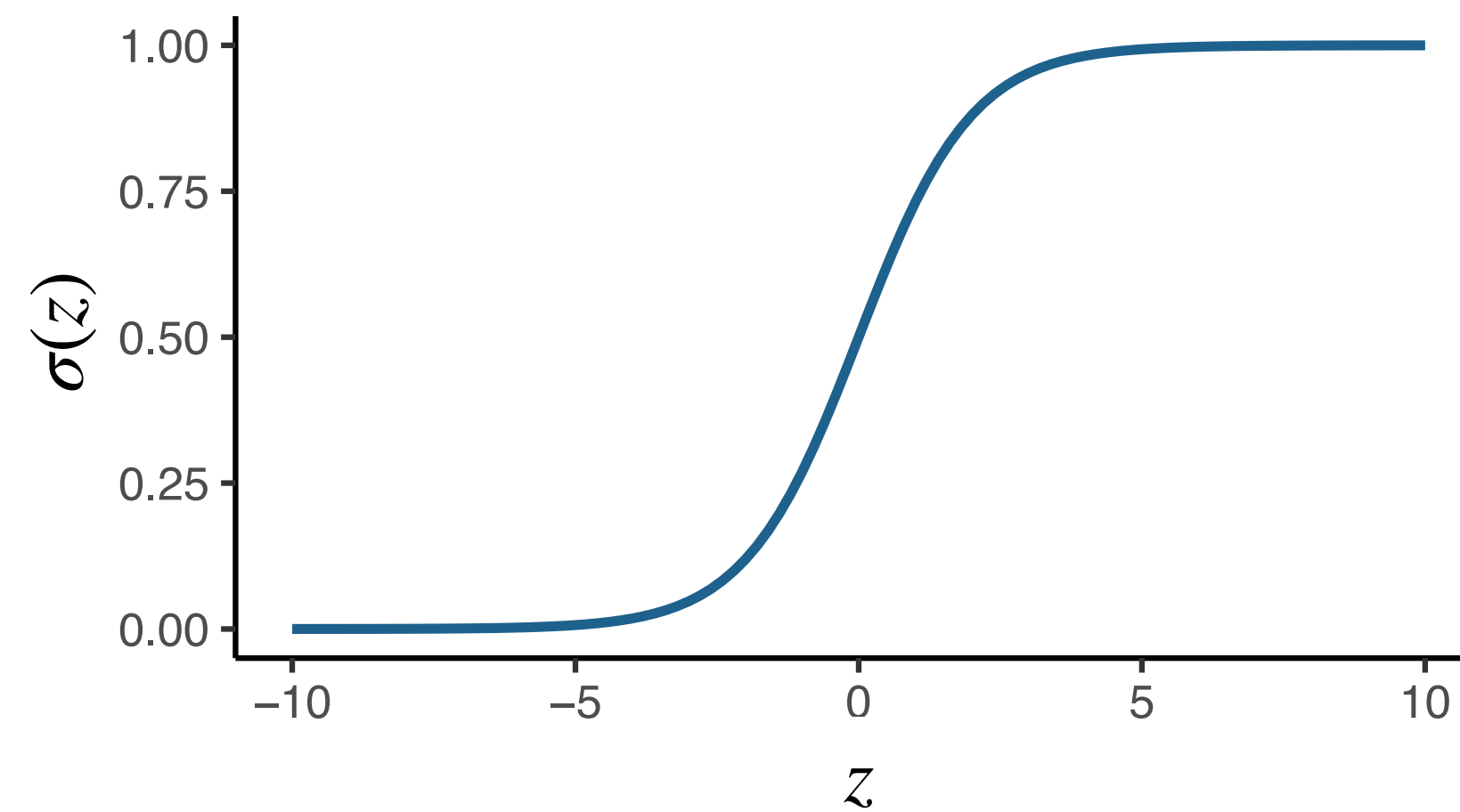
- * Introdução às redes neurais artificiais
- * Propagação para a frente e para atrás
- * Algoritmo descendente do gradiente e descendente do gradiente estocástico
- * Regularização para redes neurais

Regressão logística binária



$$z = x^T w = w_0 + \sum_{j=1}^p w_j x_j$$

$$g(x) = \sigma(z) = \sigma(x^T w)$$

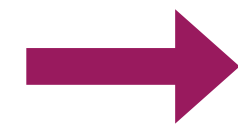


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\hat{y} = \mathbf{1}\{g(x) \geq 1 - g(x)\}$$

Regressão logística com mais de duas classes

$$f_{c_k}(x) = p(c_k | x) \quad k = 1, \dots, K$$



Funções objetivo

Modelamos $f_{c_k}(x)$ com $g_{c_k}(x) = \frac{e^{x^T w_{\cdot k}}}{\sum_{j=1}^K e^{x^T w_{\cdot j}}}$ para K vetores $w_{\cdot 1}, \dots, w_{\cdot K} \in \mathbb{R}^{p+1}$

Observemos que para todo x temos que $\sum_{k=1}^K g_{c_k}(x) = 1$, logo $g_{c_k}(x)$, $k = 1, \dots, K$ é uma distribuição de probabilidade sobre $\{c_1, \dots, c_K\}$

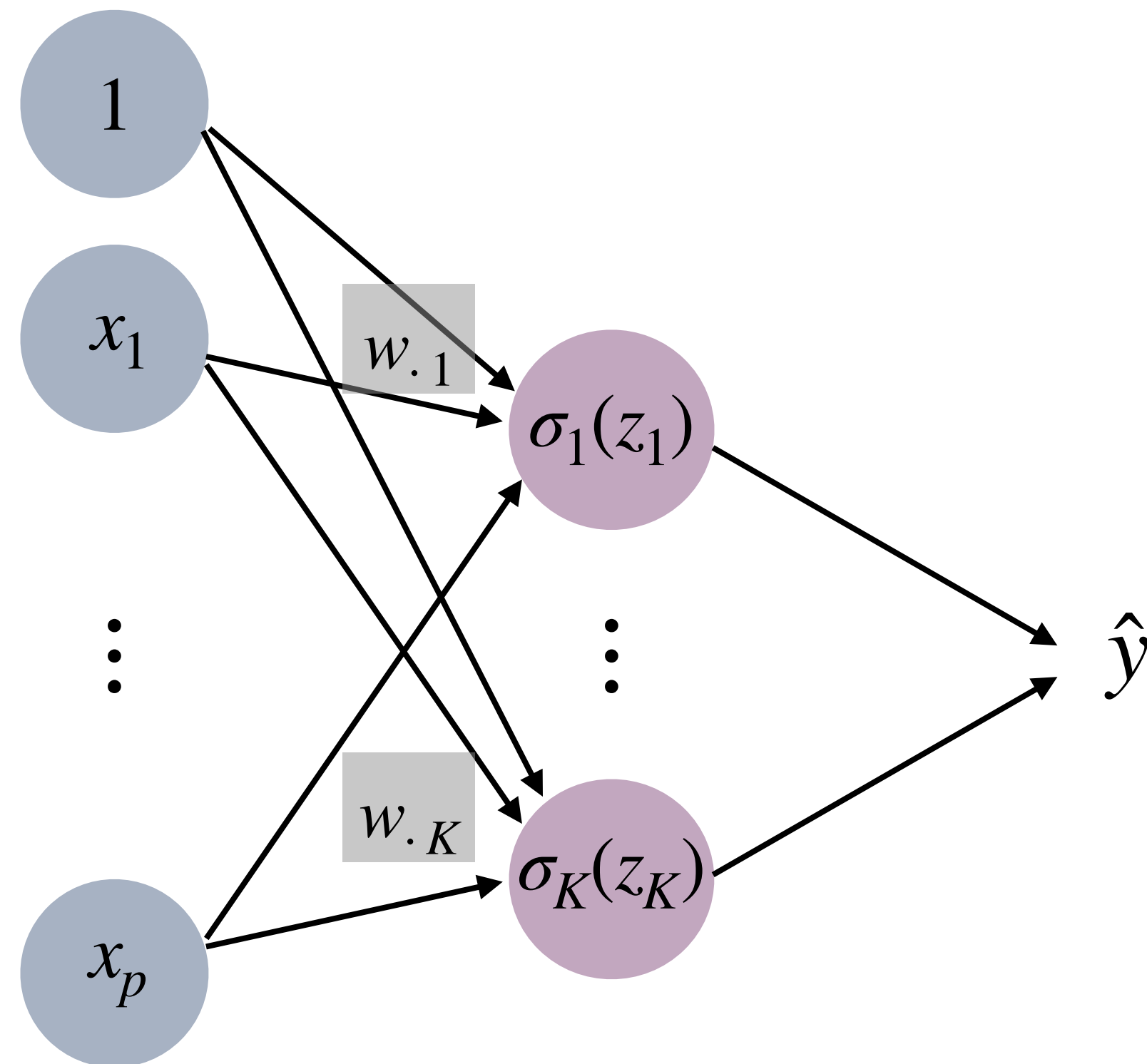
Regressão logística com mais de duas classes

$$w_{\cdot 1}, \dots, w_{\cdot K} \in \mathbb{R}^{p+1}$$

$$z_j = x^T w_{\cdot j} = w_{0j} + \sum_{i=1}^p w_{ij} x_i$$

$$g_{c_j}(x) = \sigma_j(z_j) = \sigma_j(x^T w_{\cdot j})$$

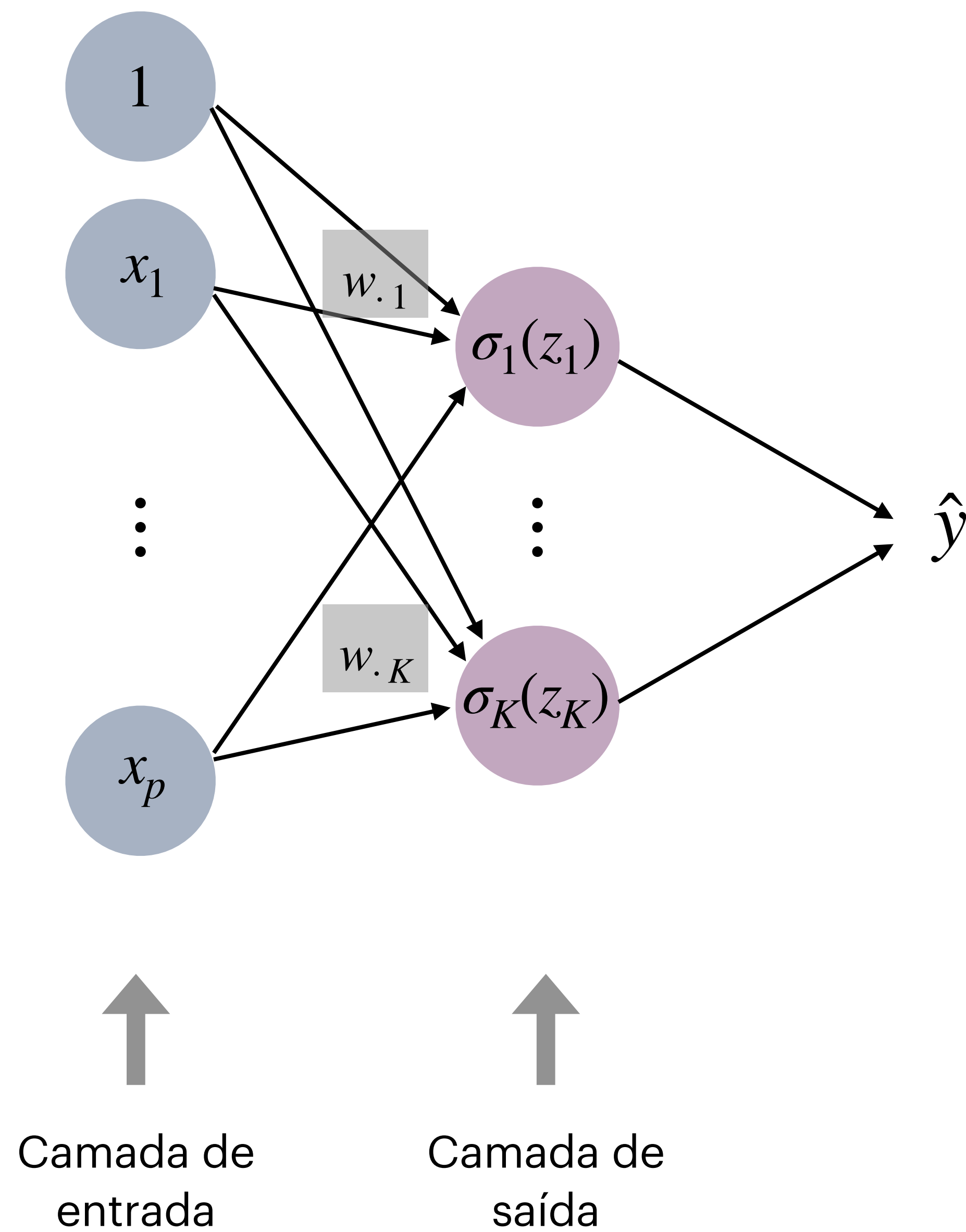
$$\sigma_j(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$



$$\sum_{k=1}^K g_{c_k}(x) = 1$$

$$\hat{y} = \arg \max_{c_k} \{ g_{c_k}(x) \}$$

$$\hat{E}_D(w) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbf{1}\{y_i = c_k\} \log g_{c_k}(x_i^T w_{\cdot k})$$



σ_j

↑

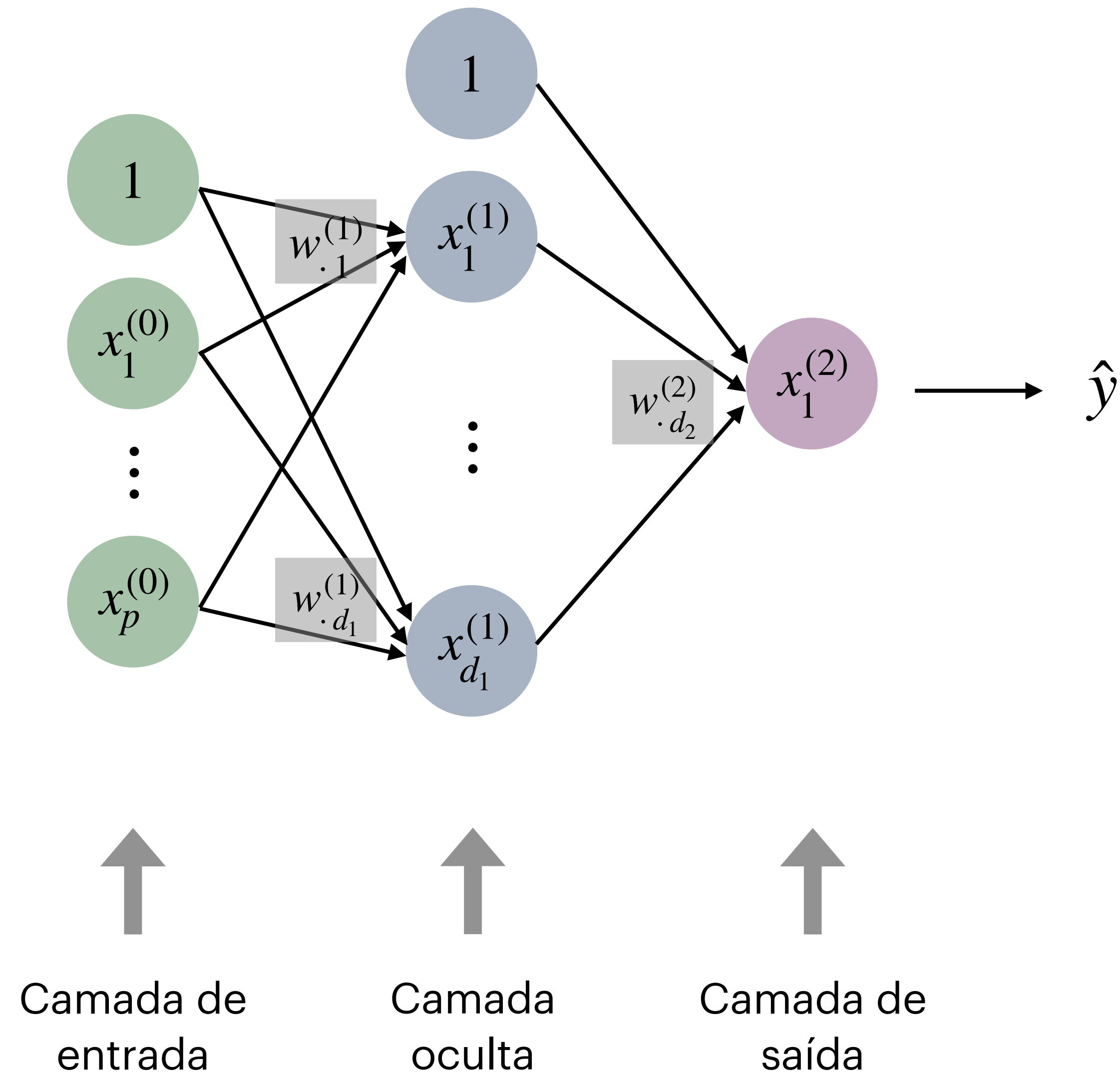
Funções de
ativação

$w_{\cdot j}$

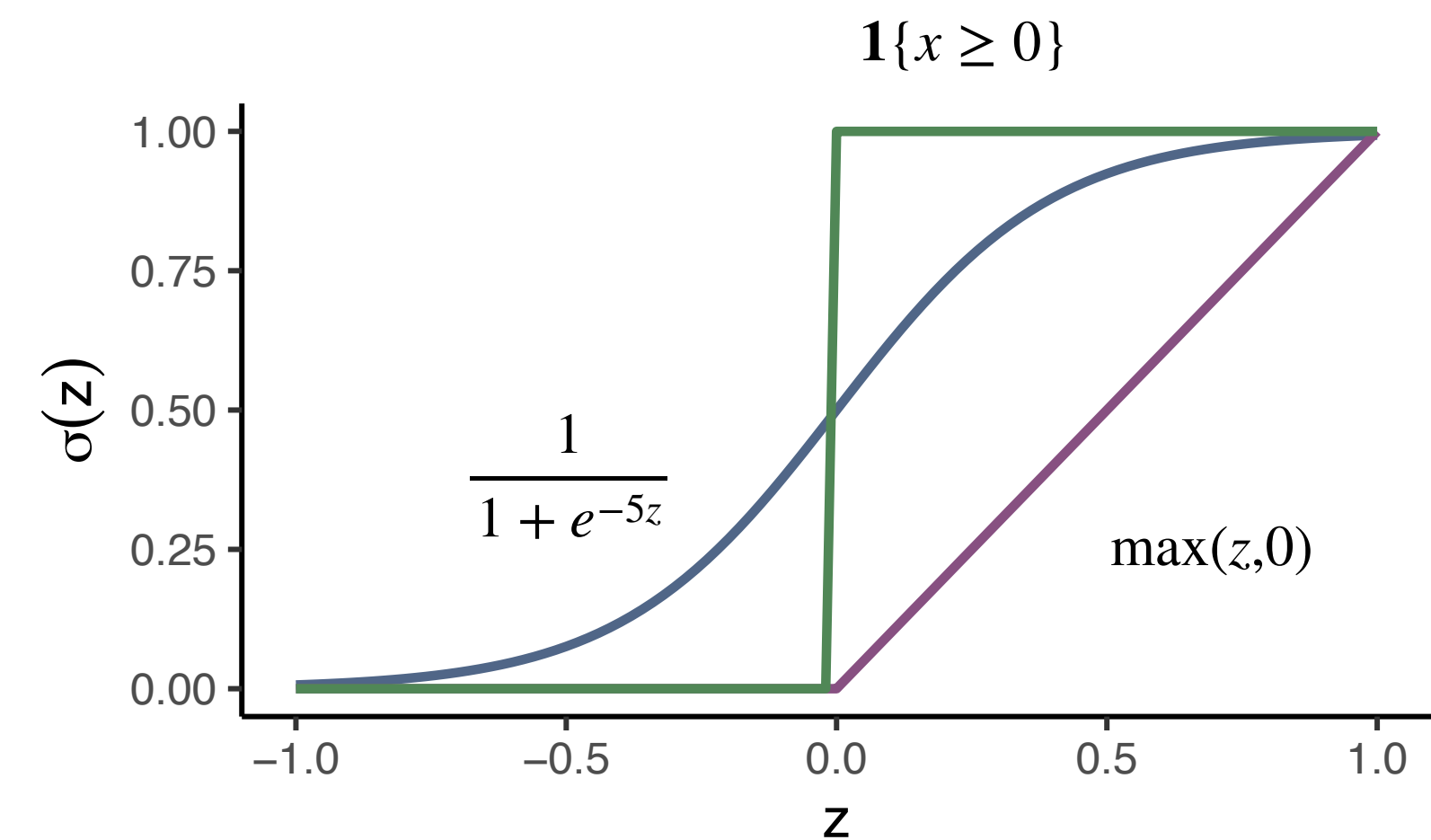
↑

Vetores de
pesos
(parâmetros)

Rede neural de uma única camada oculta

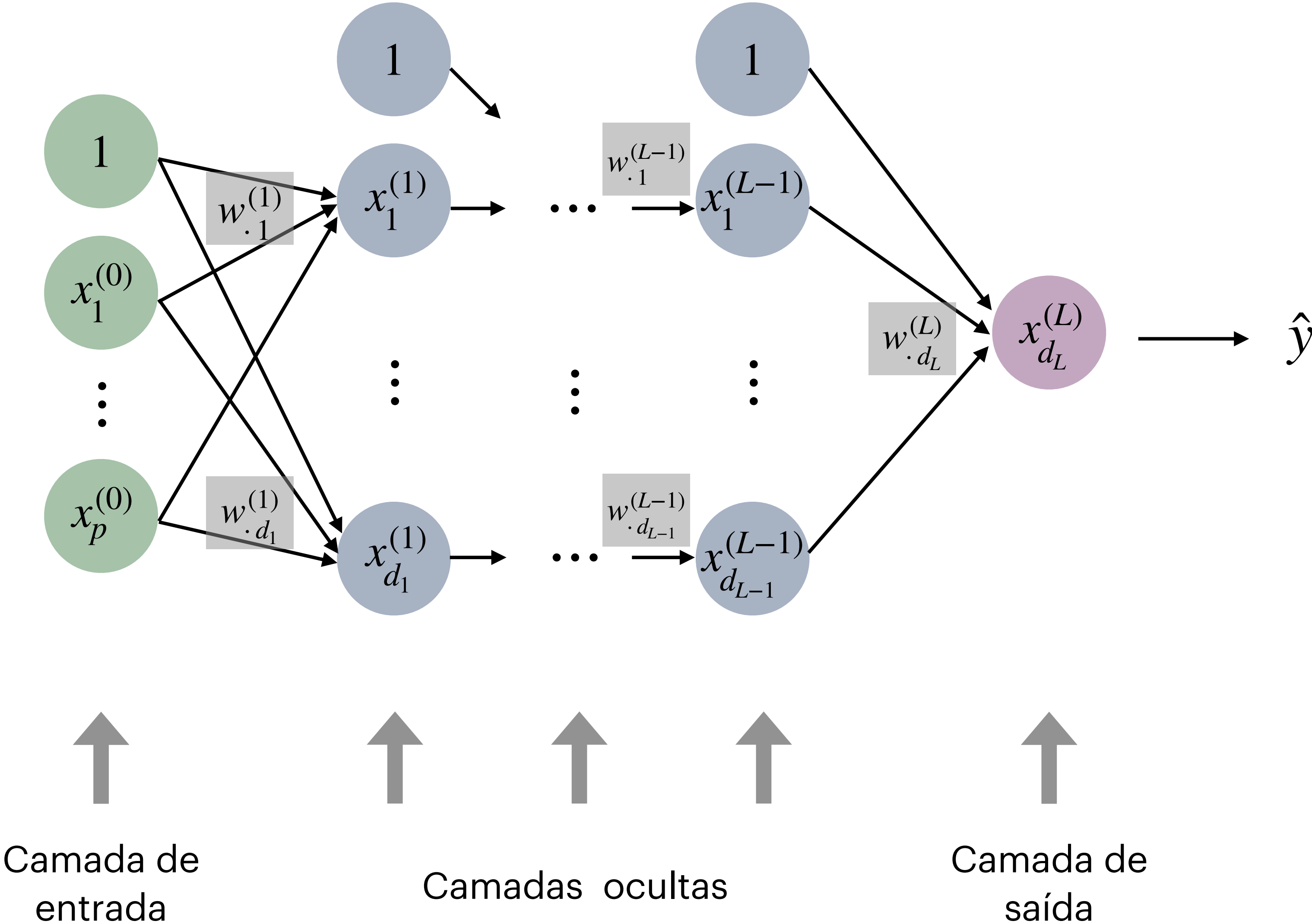


$$x_j^{(l)} = \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)})$$

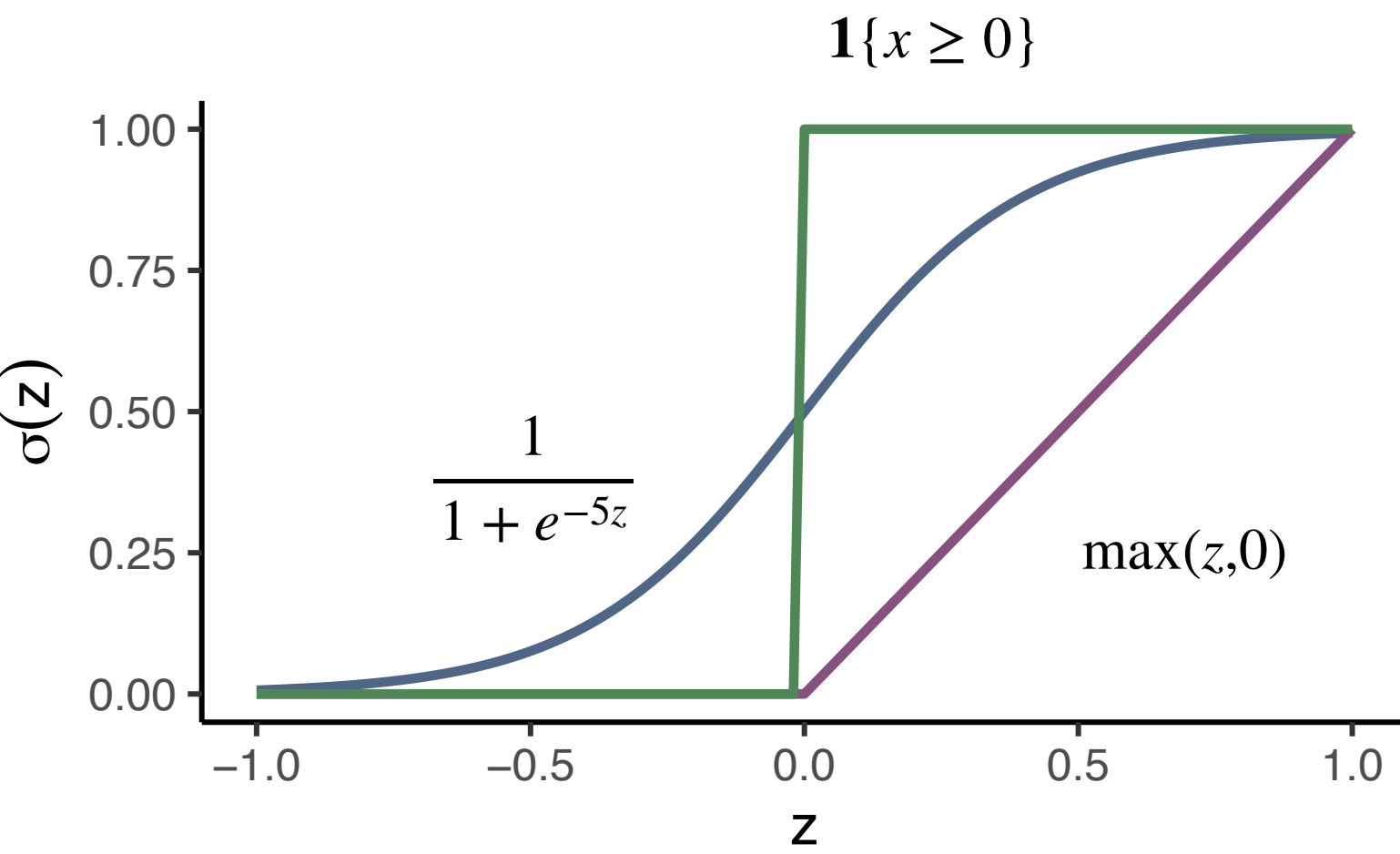


Funções de ativação

Rede neural de múltiplas camadas ocultas

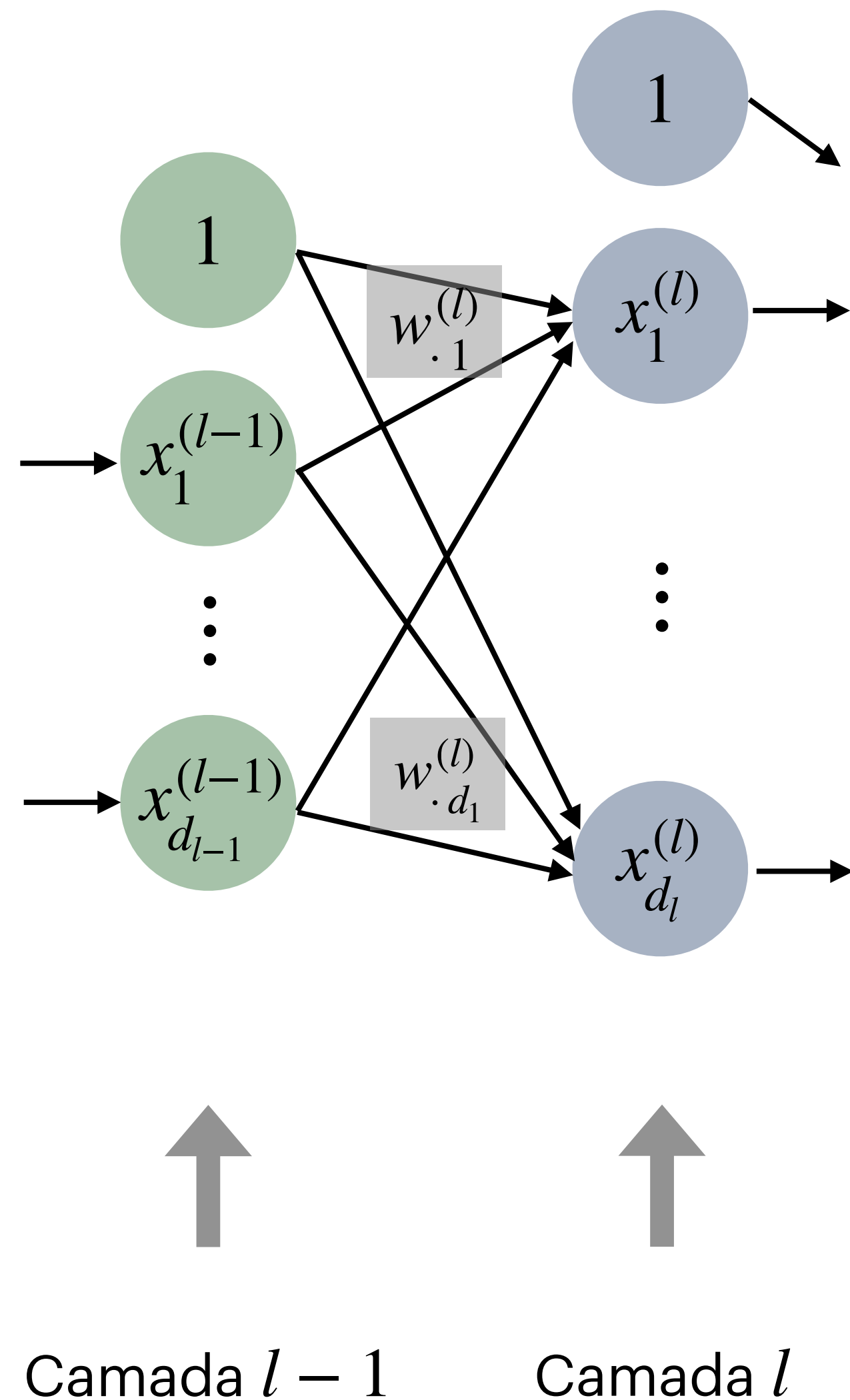


$$x_j^{(l)} = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)})$$



Funções de ativação

Rede neural de múltiplas camadas ocultas

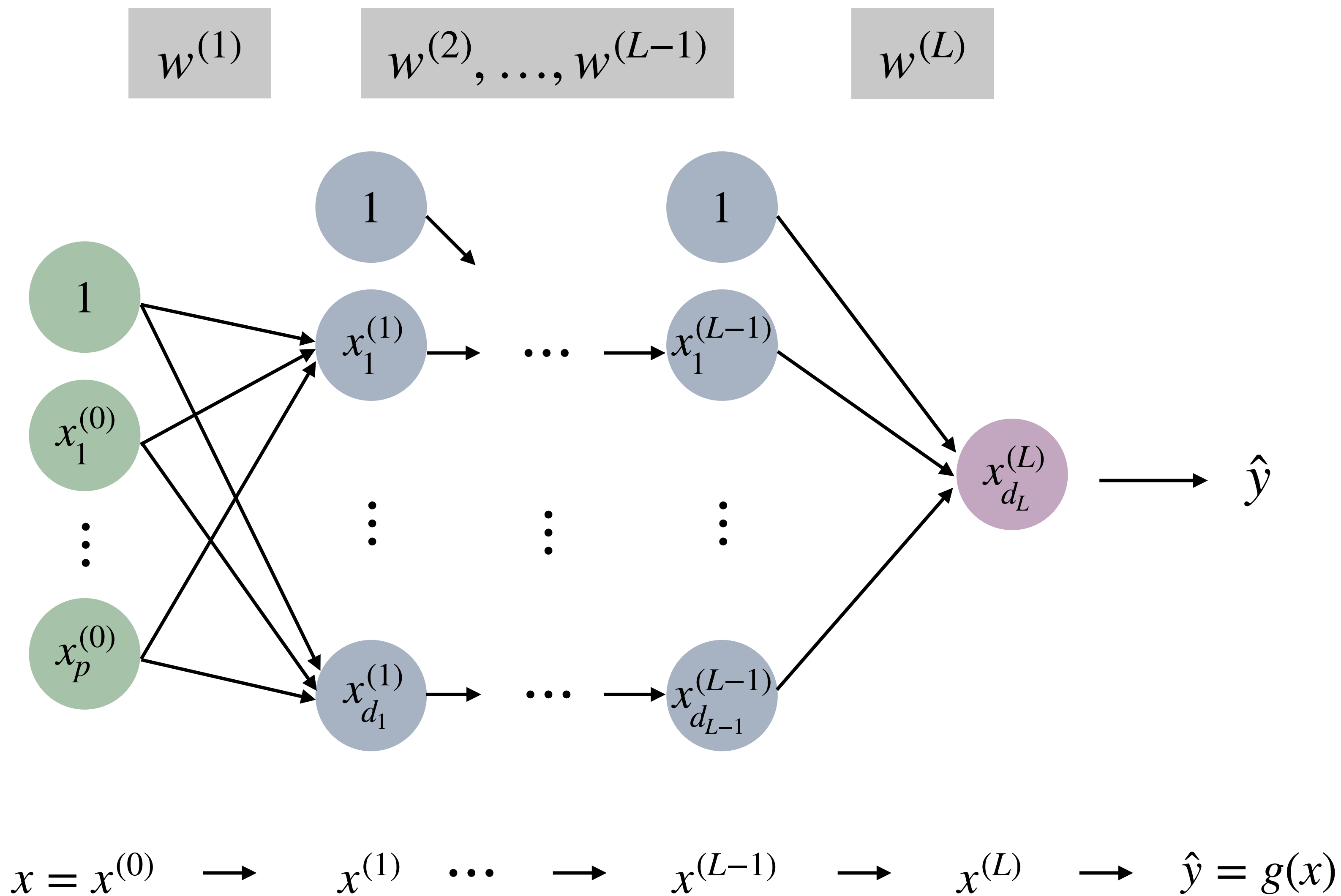


$$\left. \begin{aligned} z_j^{(l)} &= x^{(l-1),T} w_{\cdot j}^{(l)} \\ x_j^{(l)} &= \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)}) \end{aligned} \right\} \quad 1 \leq j \leq d_l$$

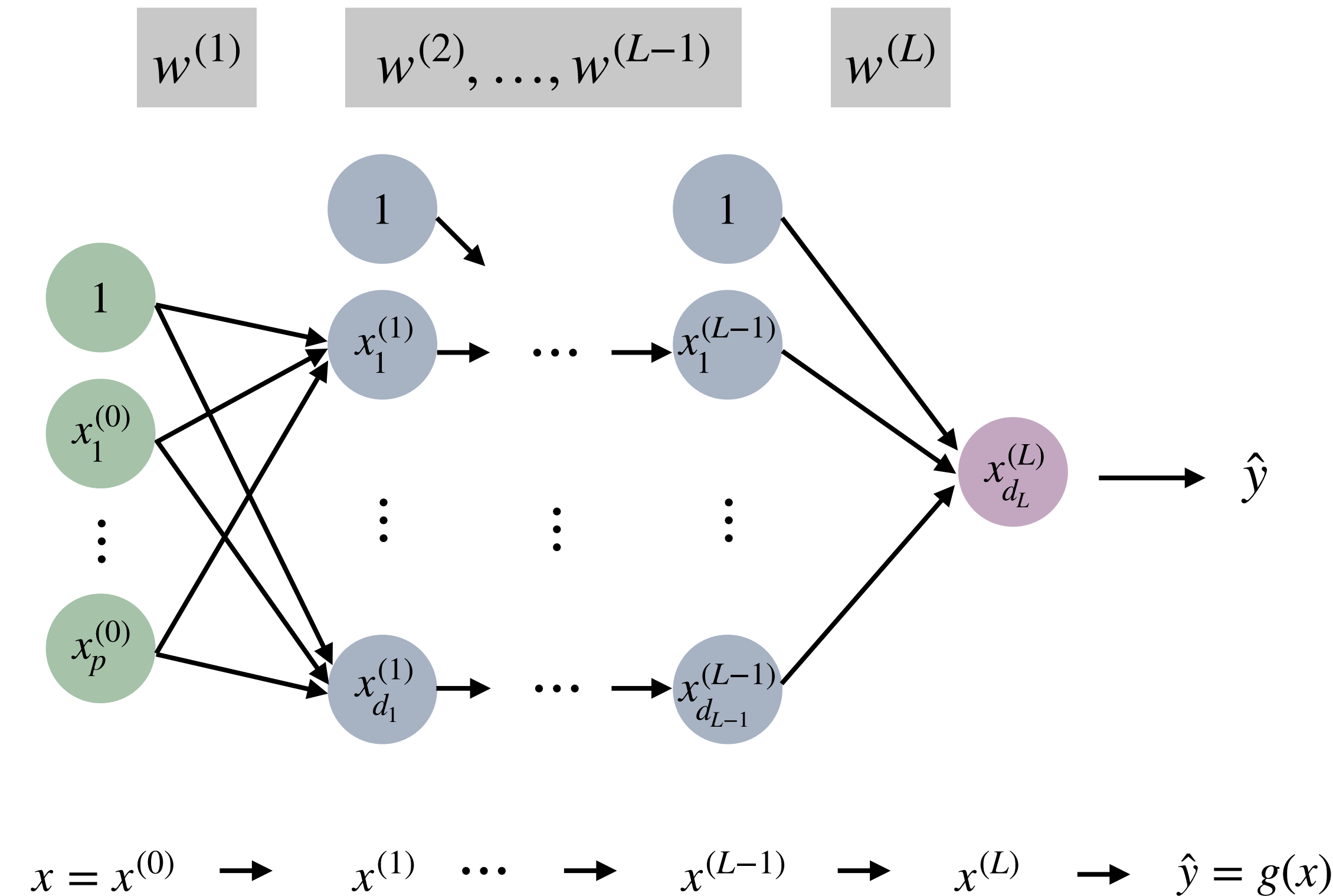
$$w_{ij}^{(l)} \quad \left\{ \begin{array}{ll} 1 \leq l \leq L & \text{camadas} \\ 0 \leq i \leq d_{l-1} & \text{entradas} \\ 1 \leq j \leq d_l & \text{saídas} \end{array} \right.$$

$$w^{(l)} \in \mathbb{R}^{(d_{l-1}+1) \times d_l}$$

Propagação para a frente



Rede neural de múltiplas camadas ocultas



Propagação para a frente

No modelo de redes neurais, a família \mathcal{G} é uma família de funções não lineares:

$$\mathcal{G} = \{g(x) = \sigma(z^{(L)}) : w^{(l)} \in \mathbb{R}^{(d_{l-1}+1) \times d_l}, 1 \leq l \leq L\}$$

$$\left. \begin{aligned} z_j^{(l)} &= x^{(l-1),T} w_{\cdot j}^{(l)} \\ x_j^{(l)} &= \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)}) \end{aligned} \right\} 1 \leq j \leq d_l$$

Ajuste do modelo

$$\mathcal{G} = \{g(x) = \sigma(z^{(L)}) : w^{(l)} \in \mathbb{R}^{(d_{l-1}+1) \times d_l}, 1 \leq l \leq L\}$$

O ajuste do modelo é feito como em qualquer problema de aprendizagem, por meio da minimização de uma função de custo L

✱ Regressão: $L(g(x), y) = (g(x) - y)^2$

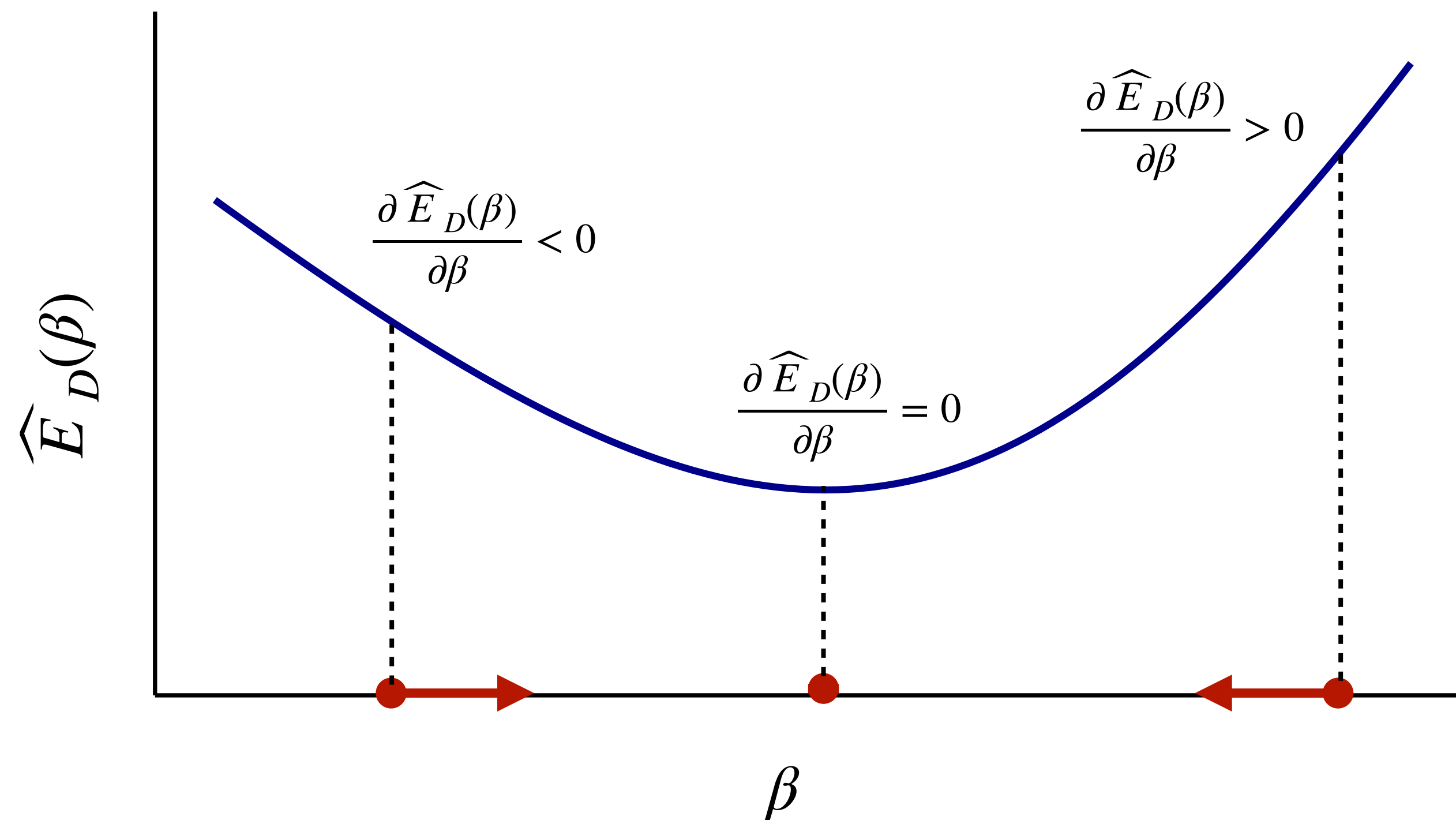
✱ Classificação: $L(g(x), y) = - \sum_{k=1}^K \mathbf{1}\{y = c_k\} \log g_{c_k}(x)$

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\} \quad \longrightarrow \quad \text{Encontrar } w \text{ que minimize } \widehat{E}_D(w) = \frac{1}{n} \sum_{i=1}^n L(g(x_i), y_i)$$

Algoritmo “gradiente descendente”

Regressão logística

Objetivo: achar β que minimize $\widehat{E}_D(\beta)$



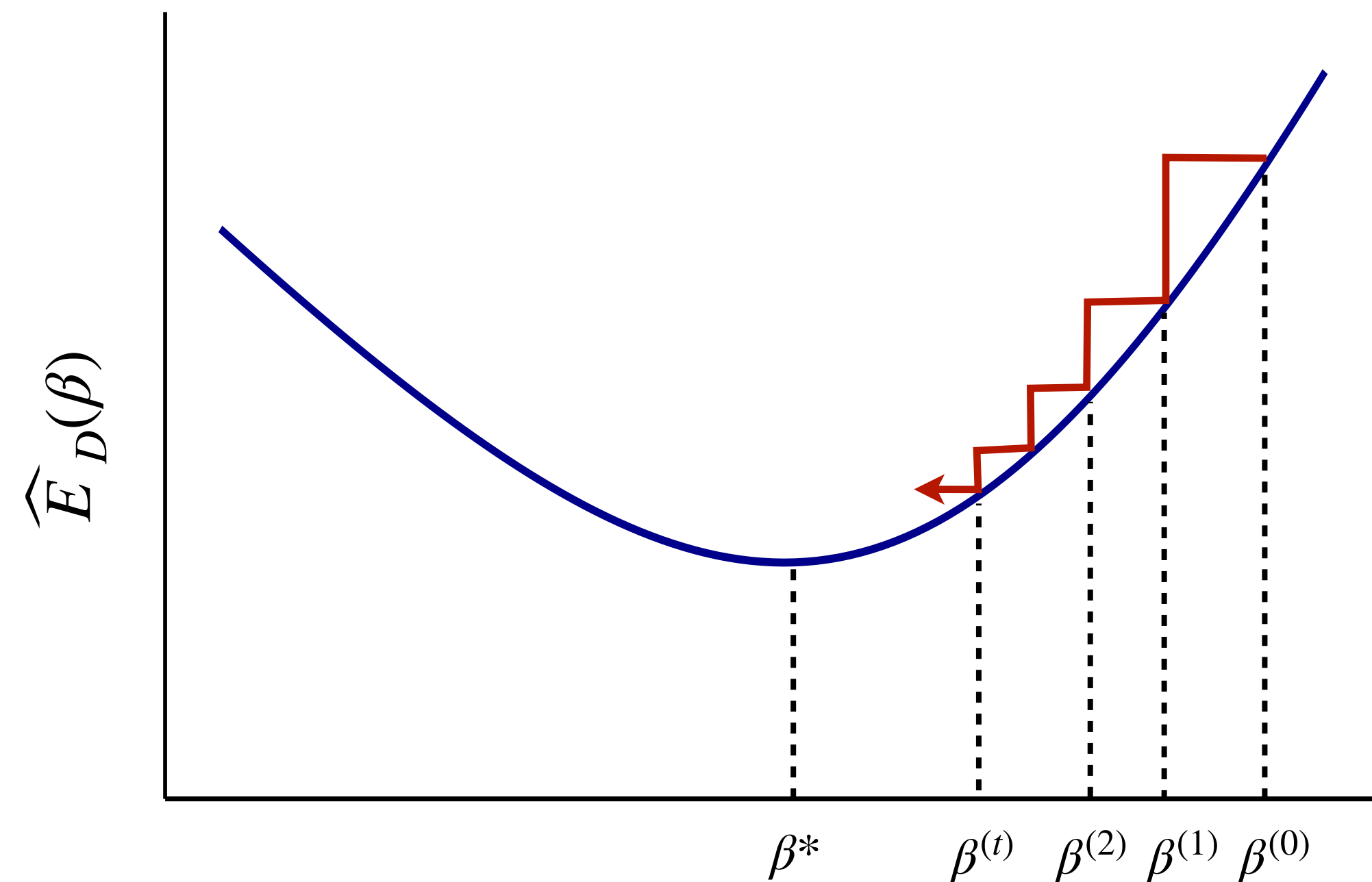
$$\nabla \widehat{E}_D(\beta) = \begin{pmatrix} \frac{\partial \widehat{E}_D(\beta)}{\partial \beta_0} \\ \vdots \\ \frac{\partial \widehat{E}_D(\beta)}{\partial \beta_p} \end{pmatrix}$$

$$\nabla \widehat{E}_D(\beta) = \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i) x_i \quad x_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}$$

Algoritmo “gradiente descendente”

Regressão logística

Objetivo: achar β que minimize $\widehat{E}_D(\beta)$



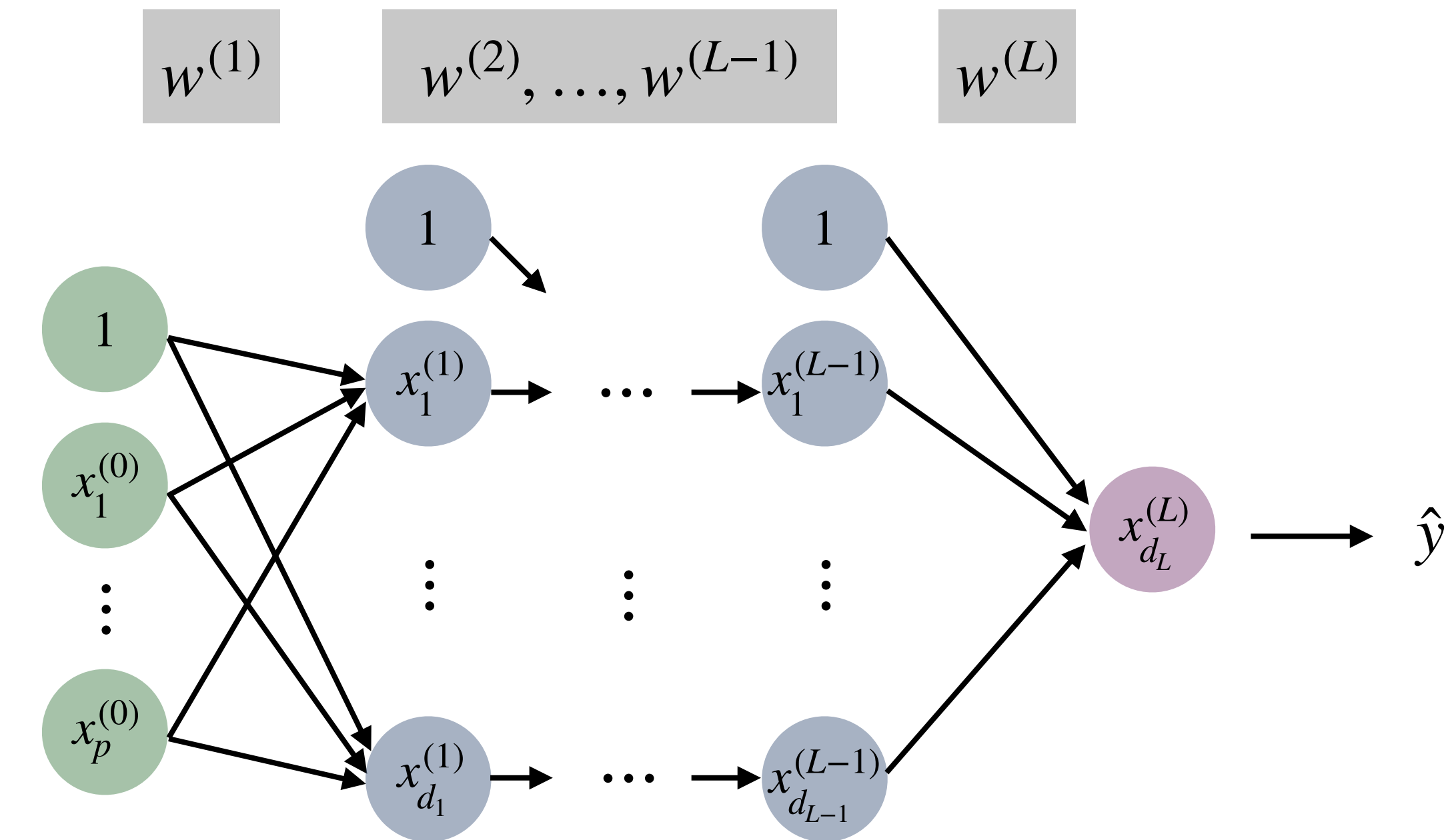
regressão logística

$$\beta^{(0)} = \vec{0} \quad (\text{inicialização})$$

$$\beta^{(t)} = \beta^{(t-1)} - \eta \nabla \widehat{E}_D(\beta^{(t-1)}) \quad (\text{iteração})$$

$$\eta \quad (\text{taxa de aprendizagem})$$

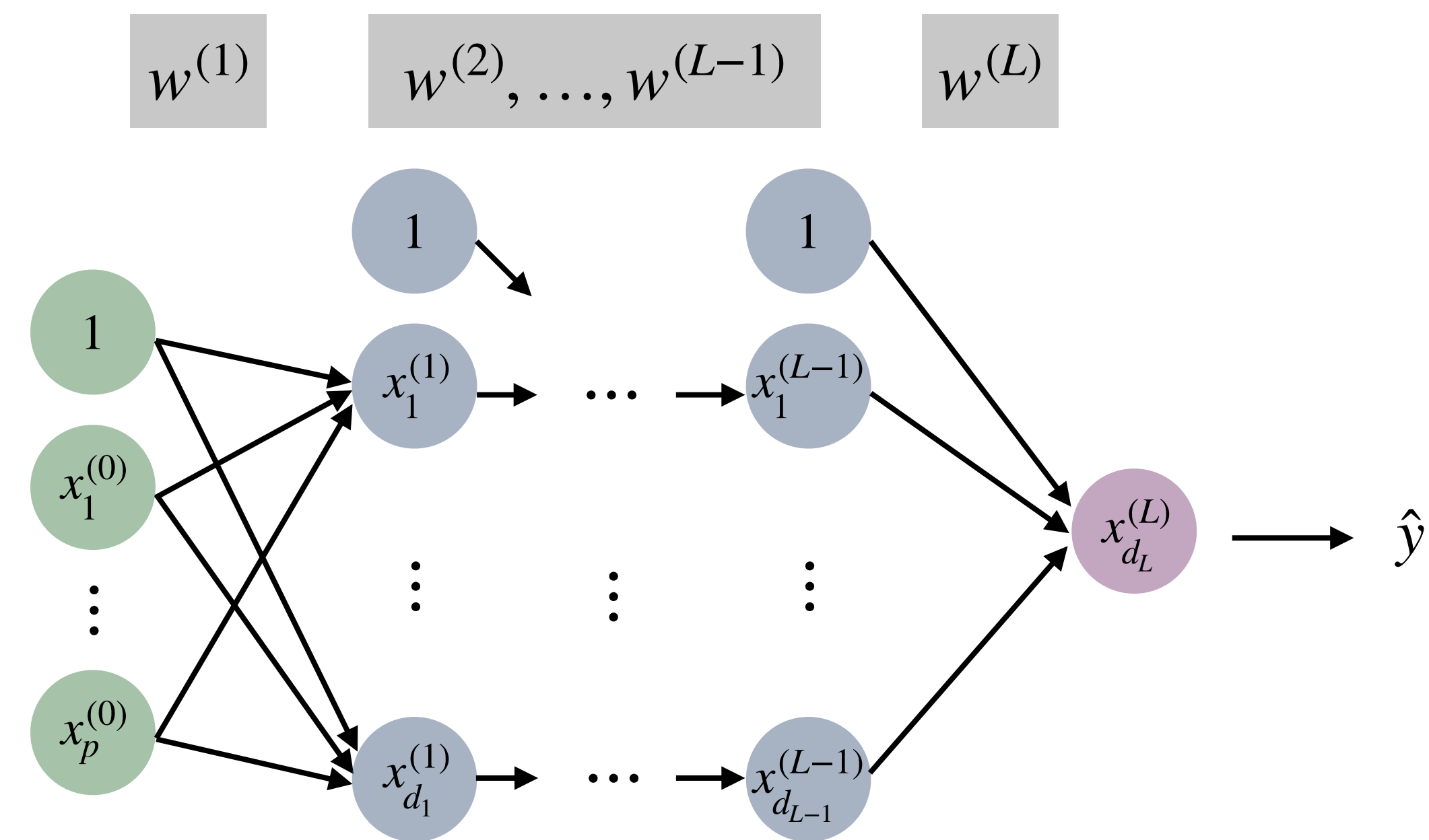
Encontrar w que minimize $\widehat{E}_D(w) = \frac{1}{n} \sum_{i=1}^n L(g(x_i), y_i)$



$$x = x^{(0)} \rightarrow x^{(1)} \rightarrow \dots \rightarrow x^{(L-1)} \rightarrow x^{(L)} \rightarrow \hat{y} = g(x) \rightarrow L(g(x), y)$$

Para usar o algoritmo descendente do gradiente precisamos calcular o gradiente de L em relação a w .

Para calculá-lo usaremos um conjunto de equações que se denominam “retro-propagação” (*backward propagation*)



$$\left. \begin{aligned} z_j^{(l)} &= x^{(l-1)} w_{\cdot j}^{(l)} \\ x_j^{(l)} &= \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)}) \end{aligned} \right\} 1 \leq j \leq d_l$$

$$x = x^{(0)} \rightarrow x^{(1)} \rightarrow \dots \rightarrow x^{(L-1)} \rightarrow x^{(L)} \rightarrow \hat{y} = g(x) \rightarrow L(g(x), y)$$

$$\frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} = x_i^{(l-1)}$$

Precisamos calcular $\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \times \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$

$$\frac{\partial L}{\partial z_j^{(l)}} = \delta_j^{(l)} = ??$$

Propagação para atrás

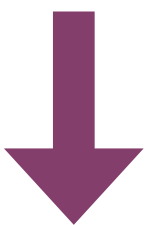
$$\frac{\partial L}{\partial z_j^{(l)}} = \delta_j^{(l)}$$

$$z_j^{(l)} = x^{(l-1)} w_{\cdot j}^{(l)}$$

$$x_j^{(l)} = \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)})$$

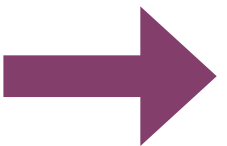
$$L(g(x), y) = (g(x) - y)^2$$

$$g(x) = \sigma(z^{(L)})$$



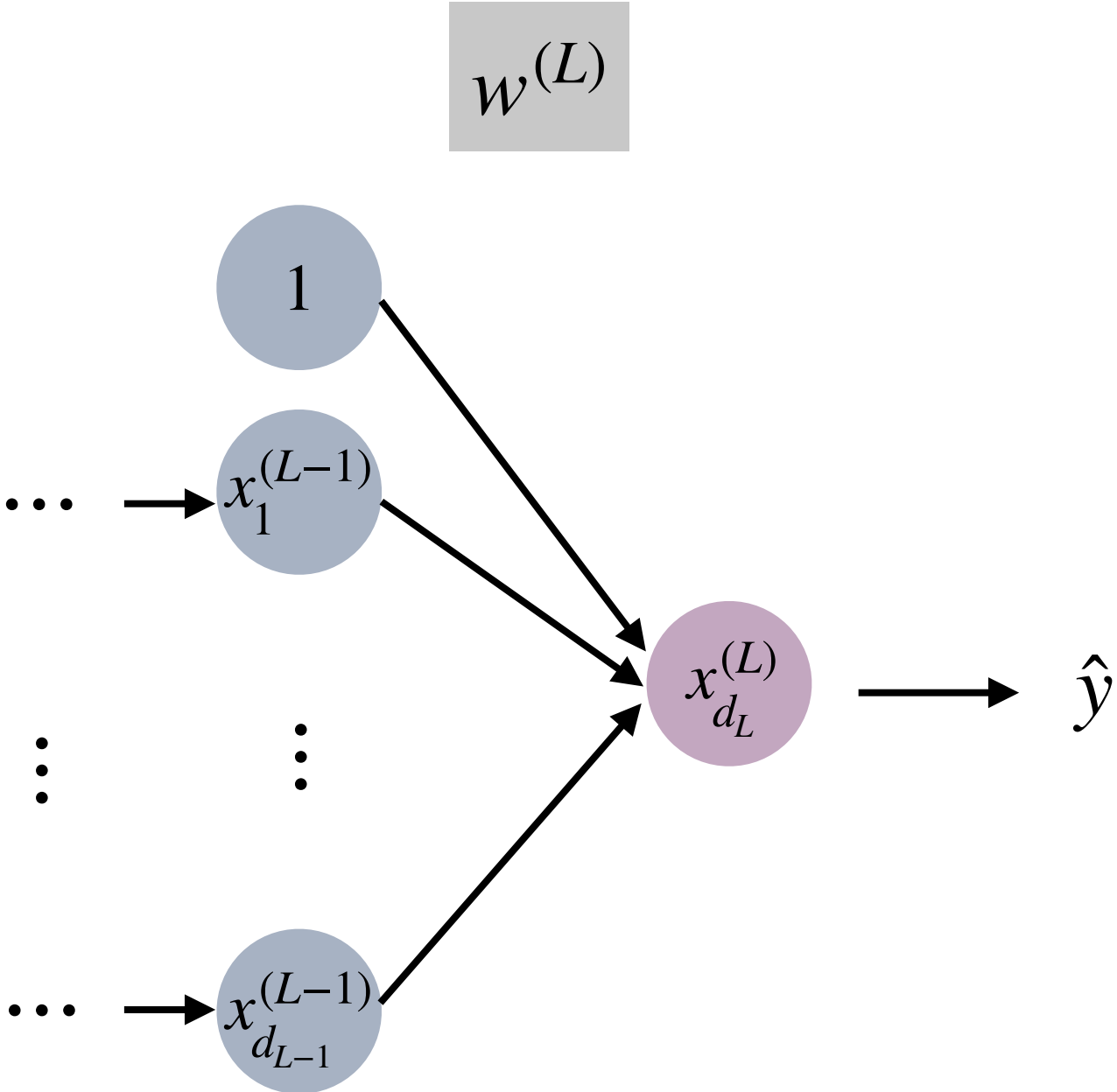
$$\delta^{(L)} = 2(\sigma(z^{(L)}) - y) \times \sigma'(z^{(L)})$$

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



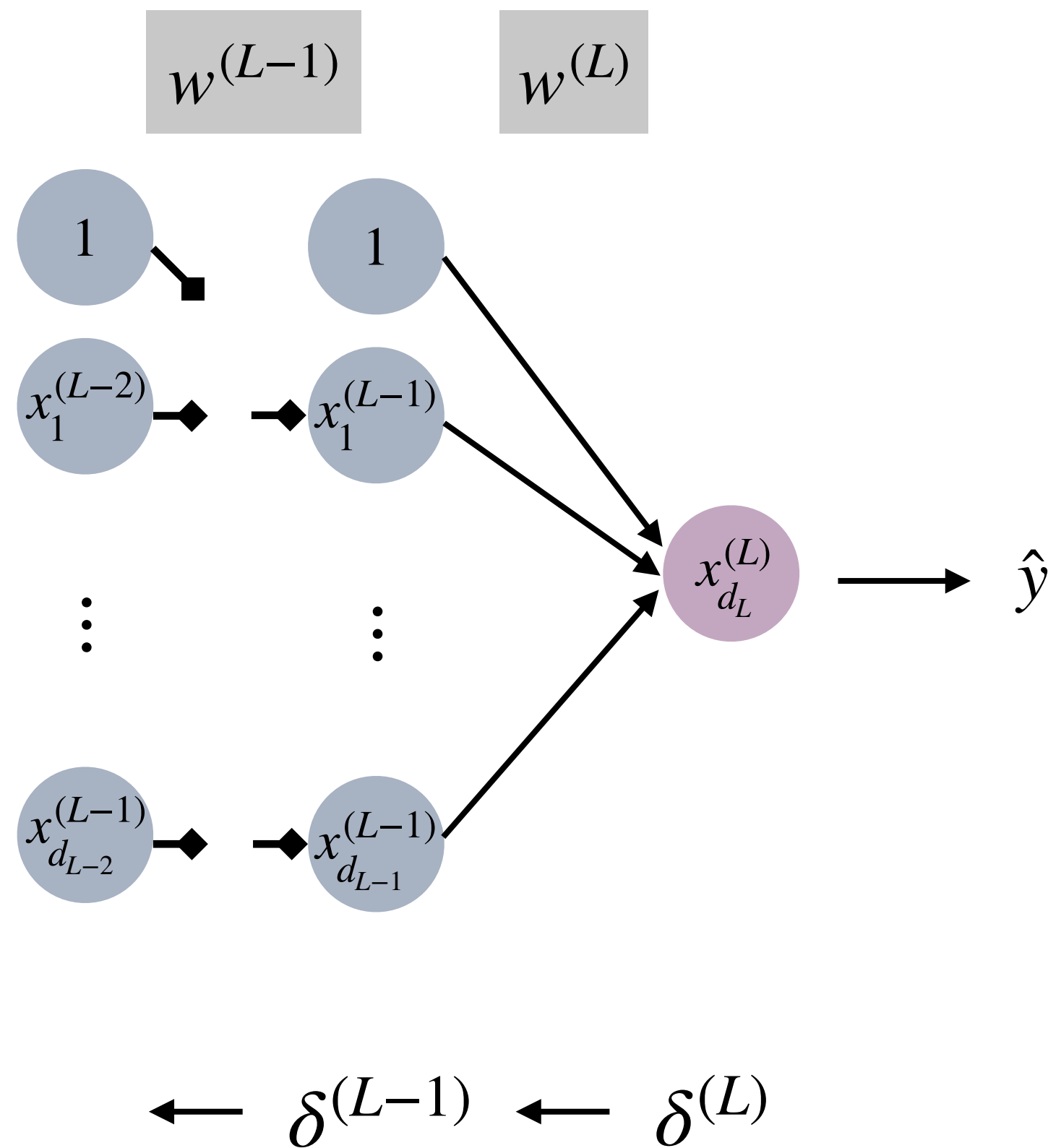
$$\sigma'(z) = \frac{e^{-z}}{1 + e^{-z}}$$

$$= \sigma(z)(1 - \sigma(z))$$



$$x^{(L-1)} \rightarrow x^{(L)} \rightarrow \hat{y} = g(x) \rightarrow L(g(x), y)$$

Propagação para atrás



$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \times \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$$

$$\delta_j^{(l)}$$

$$z_j^{(l)} = x^{(l-1)} w_{\cdot j}^{(l)}$$

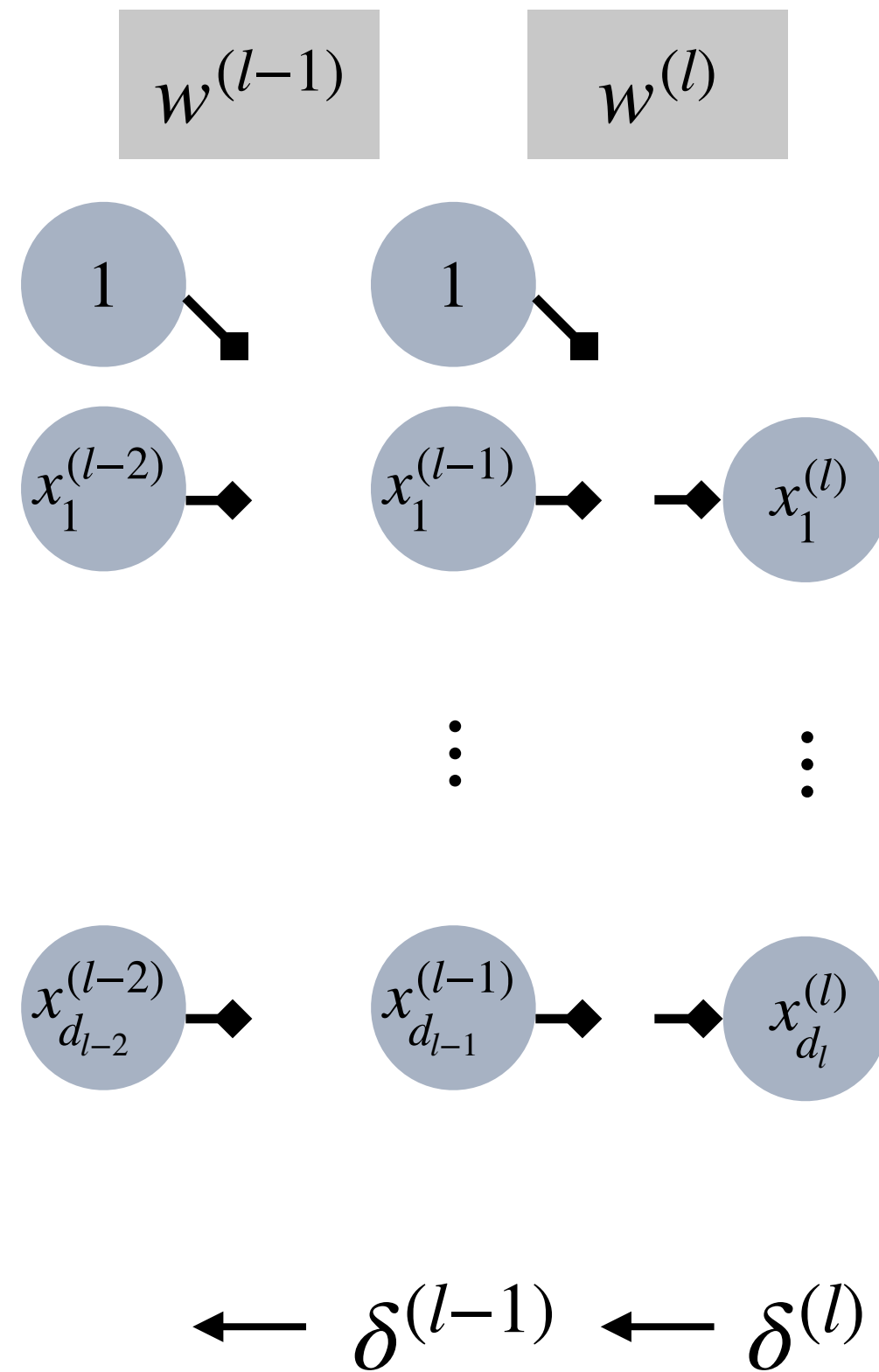
$$x_j^{(l)} = \sigma(z_j^{(l)}) = \sigma(x^{(l-1),T} w_{\cdot j}^{(l)})$$

$$L(\sigma(z^{(L)}), y) = (\sigma(z^{(L)}) - y)^2$$

$$\sigma(z^{(L)}) = \sigma(x^{(L-1)} w^{(L)}) = \sigma\left(\sum_{j=0}^{d_{L-1}} \sigma(z_j^{(L-1)}) w_j^{(L)}\right)$$

$$\delta_j^{(L-1)} = \frac{\partial L}{\partial z_j^{(L-1)}} = \frac{\partial L}{\partial z_1^{(L)}} \times \frac{\partial z_1^{(L)}}{\partial x_j^{(L-1)}} \times \frac{\partial x_j^{(L-1)}}{\partial z_j^{(L-1)}} = \delta_1^{(L)} \times w_{j1}^{(L)} \times \sigma'(z_j^{(L-1)})$$

Propagação para atrás



$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \times \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$$

$$\downarrow$$

$$\delta_j^{(l)}$$

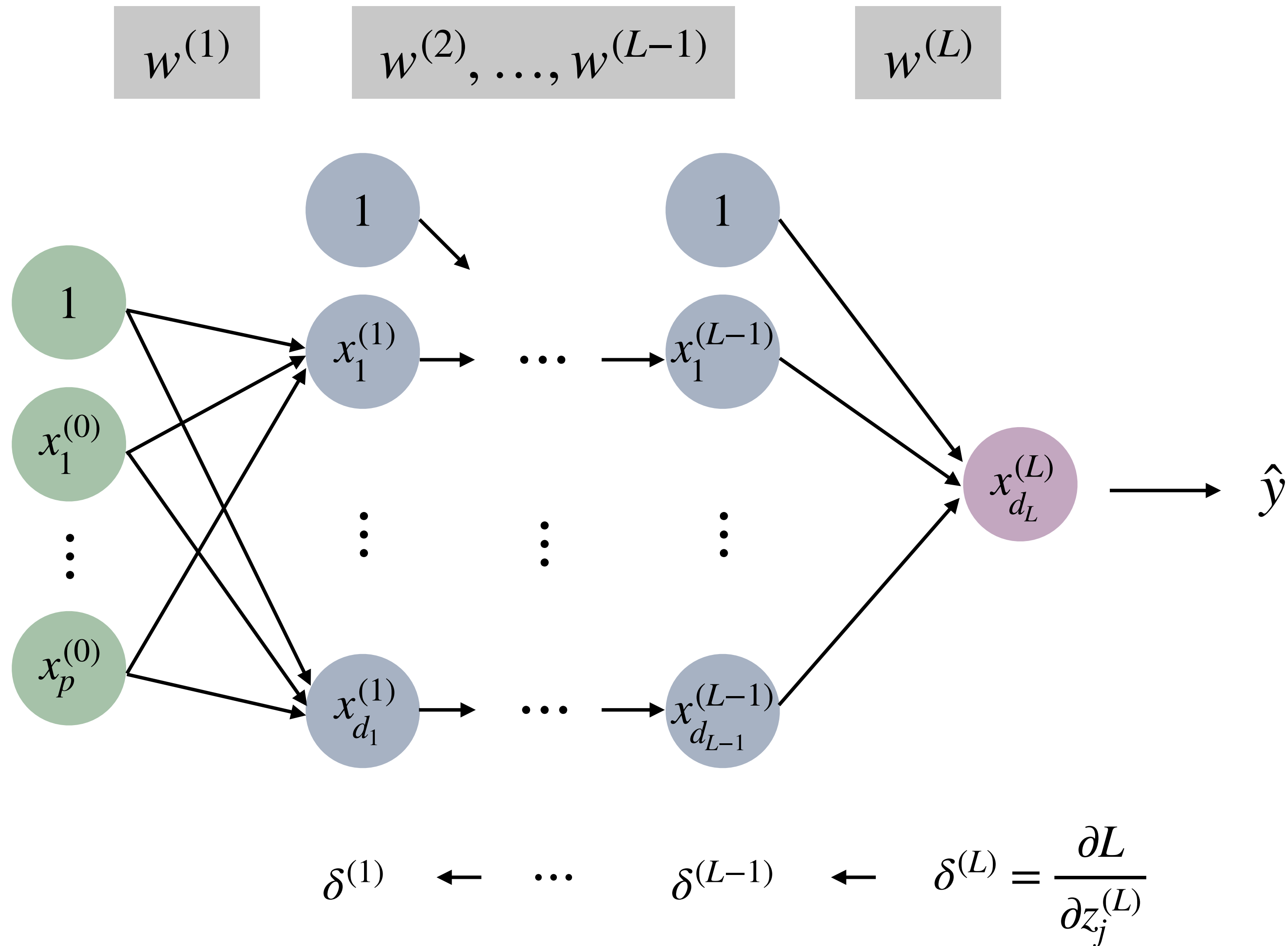
$$z_j^{(l)} = x^{(l-1)} w_{\cdot j}^{(l)}$$

$$x_j^{(l)} = \sigma(z_j^{(l)}) = \sigma(x^{(l-1), T} w_{\cdot j}^{(l)})$$

$$\delta_j^{(l-1)} = \frac{\partial L}{\partial z_j^{(l-1)}} = \sum_{r=1}^{d_l} \frac{\partial L}{\partial z_r^{(l)}} \times \frac{\partial z_r^{(l)}}{\partial x_j^{(l-1)}} \times \frac{\partial x_j^{(l-1)}}{\partial z_j^{(l-1)}}$$

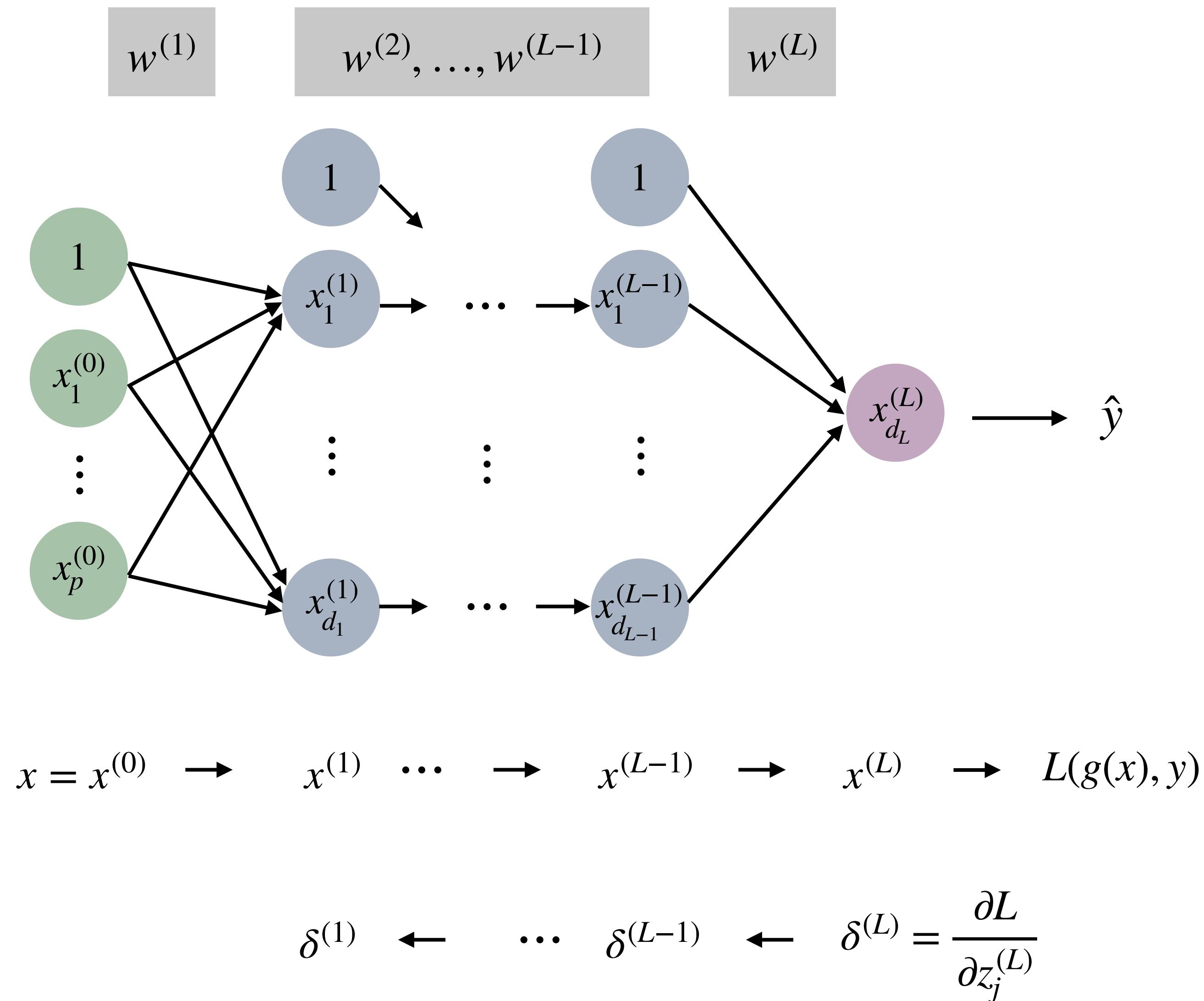
$$= \sum_{r=1}^{d_l} \delta_r^{(l)} \times w_{jr}^{(l)} \times \sigma'(z_j^{[l-1]})$$

Propagação para atrás



$$\frac{\partial L}{\partial z_j^{(l)}} = \delta_j^{(l)}$$

Propagação para atrás



$$\frac{\partial L}{\partial w^{(l)}_{ij}} = \frac{\partial L}{\partial z^{(l)}_j} \times \frac{\partial z^{(l)}_j}{\partial w^{(l)}_{ij}}$$

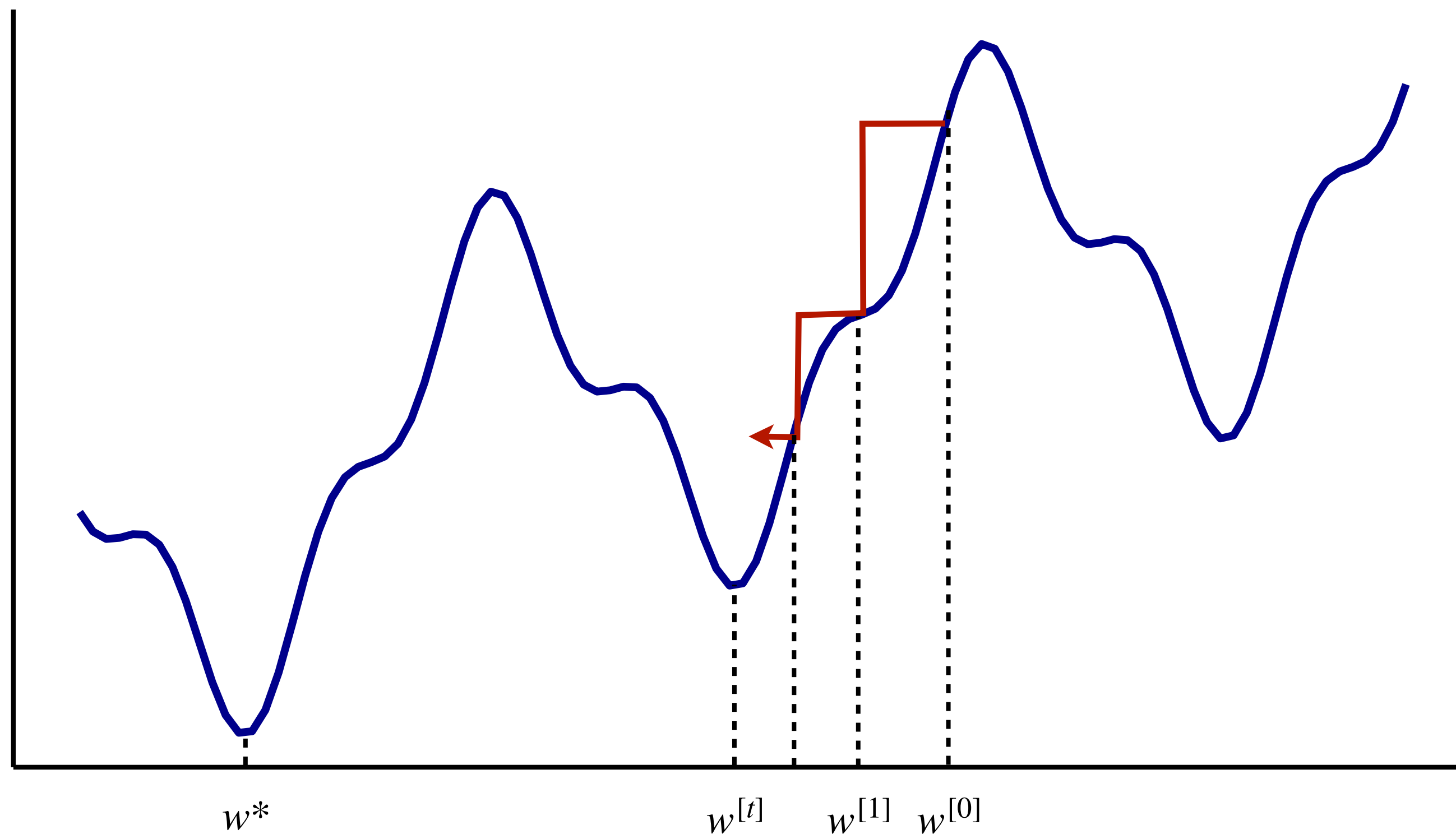
\downarrow $\delta^{(l)}_j$ \downarrow $x^{(l-1)}_i$

$$\nabla \hat{L}(w) = \begin{pmatrix} \frac{\partial L}{\partial w^{(1)}_{01}}(w) \\ \vdots \\ \frac{\partial \hat{L}}{\partial w^{(L)}_{d_{L-1} d_L}}(w) \end{pmatrix}$$

Algoritmo “gradiente descendente”

Redes neurais

Objetivo: achar w que minimize $\widehat{E}_D(w) = \frac{1}{n} \sum_{i=1}^n L(g(x_i), y_i)$



redes neurais

$w^{[0]} \sim \mathcal{N}(0,1)$ (inicialização)

$w^{[t]} = w^{[t-1]} - \eta \nabla \widehat{E}_D(w^{[t-1]})$ (iteração)

η (taxa de aprendizagem)

Algoritmo: gradiente descendente (redes neurais)

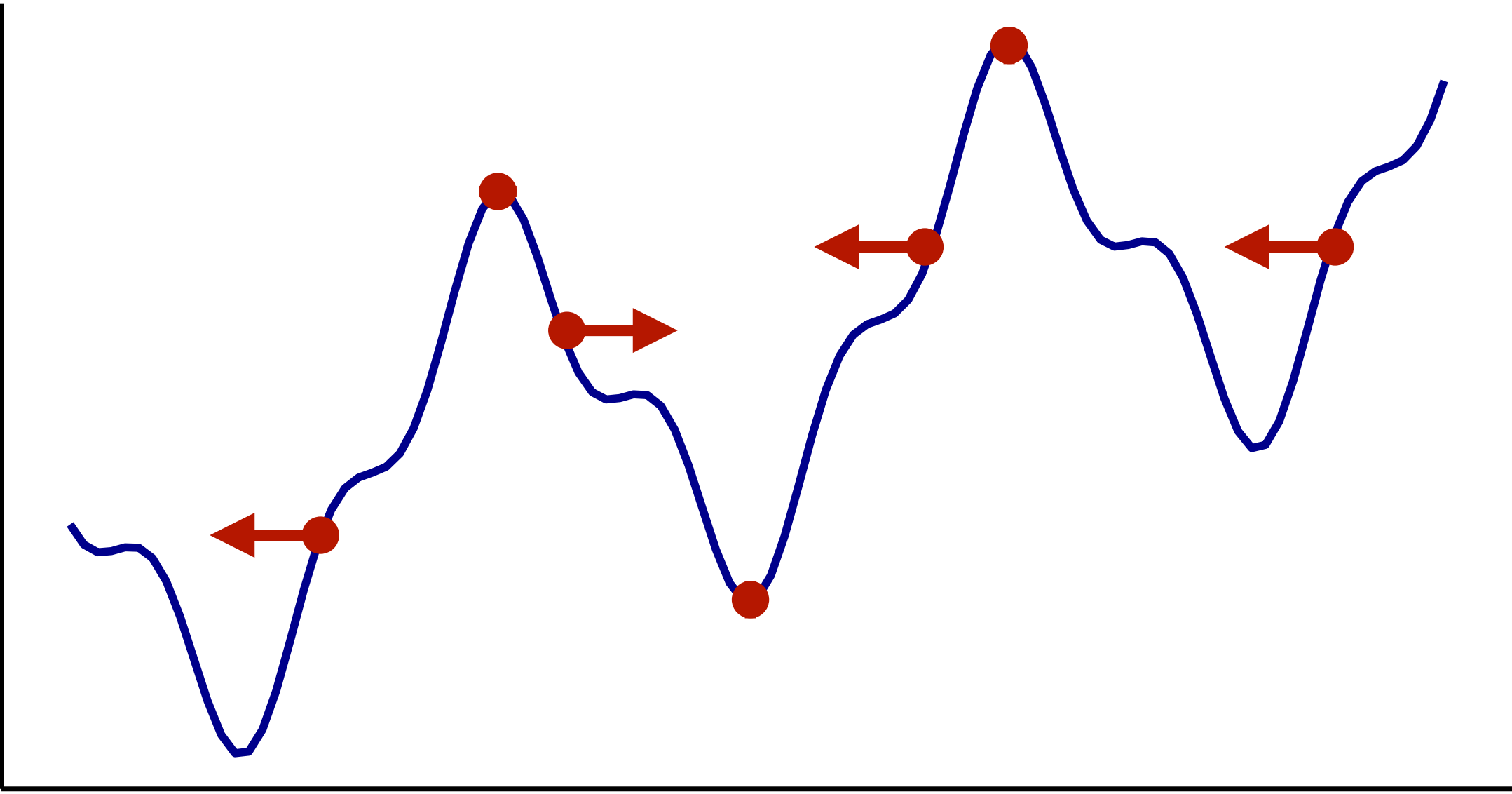
1. Inicialize todos os pesos $w^{(l)[0]}, 1 \leq l \leq L$ de forma aleatória em $t = 0$
2. Para $t = 1, 2, \dots$
 - (a) *Propagação para a frente*: calcule todos os vetores $x^{(l)[t]}, 1 \leq l \leq L$, usando os pesos $w^{(l)[t-1]}, 1 \leq l \leq L$
 - (b) *Propagação para atrás*: calcule todos os vetores $\delta_j^{(l)}, 1 \leq l \leq L$, usando os valores de $x^{(l)[t]}$ e $w^{(l)[t-1]}, 1 \leq l \leq L$
 - (c) Atualize os pesos $w_{ij}^{(l)[t]} = w_{ij}^{(l)[t-1]} - \eta x_i^{(l-1)[t]} \delta_j^{(l)[t]}$
 - (d) Itere os passos (a)-(d) até a parada
3. Retorne os pesos finais $w^{(l)}, 1 \leq l \leq L$

Descendente do gradiente estocástico

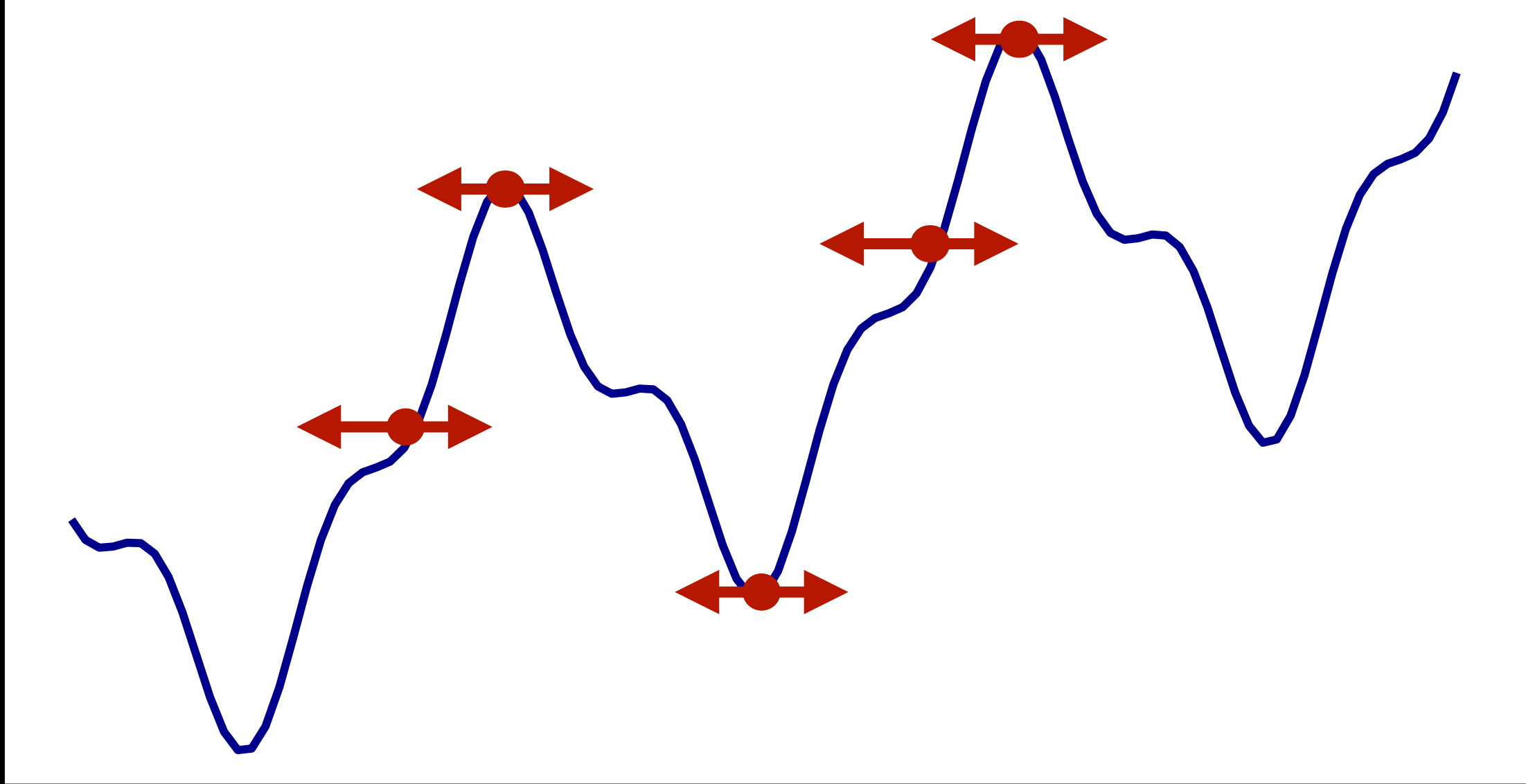
Objetivo: achar w que minimize $\widehat{E}_D(w) = \frac{1}{n} \sum_{i=1}^n L(g(x_i), y_i)$

- * No cálculo do gradiente de $\widehat{E}_D(w)$, devemos fazer a soma sobre os n pontos do conjunto de treinamento \mathcal{D}_{Tr}
- * Se n for muito grande, isto pode ser muito demorado ...
- * Para acelerar este cálculo, no método do descendente do gradiente estocástico, escolhemos um subconjunto de observações menor ("*batch*") e calculamos o gradiente de L somente sobre estes pontos

Descendente do gradiente estocástico



descendente do gradiente



descendente do gradiente estocástico

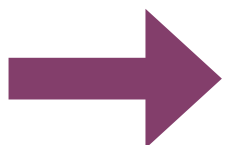
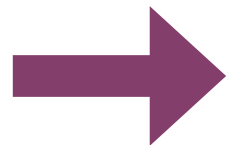
Regularização

No modelo de redes neurais existem várias formas diferentes de regularização (diminuição da variância do modelo). Algumas são

- ✱ Penalidade do tipo RIDGE ou LASSO
- ✱ Aprendizagem por “supressão de nós” (*dropout*)
- ✱ Monitoramento do erro de validação e parada antecipada (*early stopping*)

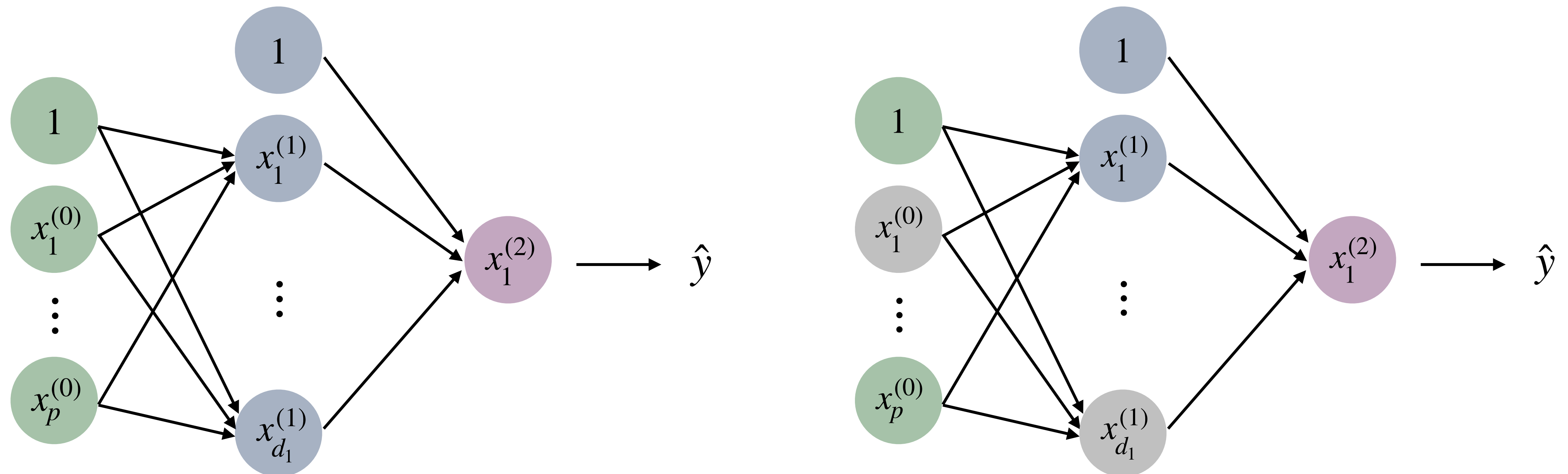
Penalidade LASSO ou RIDGE

Assim como em outros métodos, podemos adicionar uma penalidade na norma dos parâmetros da rede, e utilizar o método do descendente do gradiente para

- * achar w que minimize $\widehat{E}_D(w) + \lambda \|w\|_2^2$  regularização RIDGE
- * achar w que minimize $\widehat{E}_D(w) + \lambda \|w\|_1$  regularização LASSO

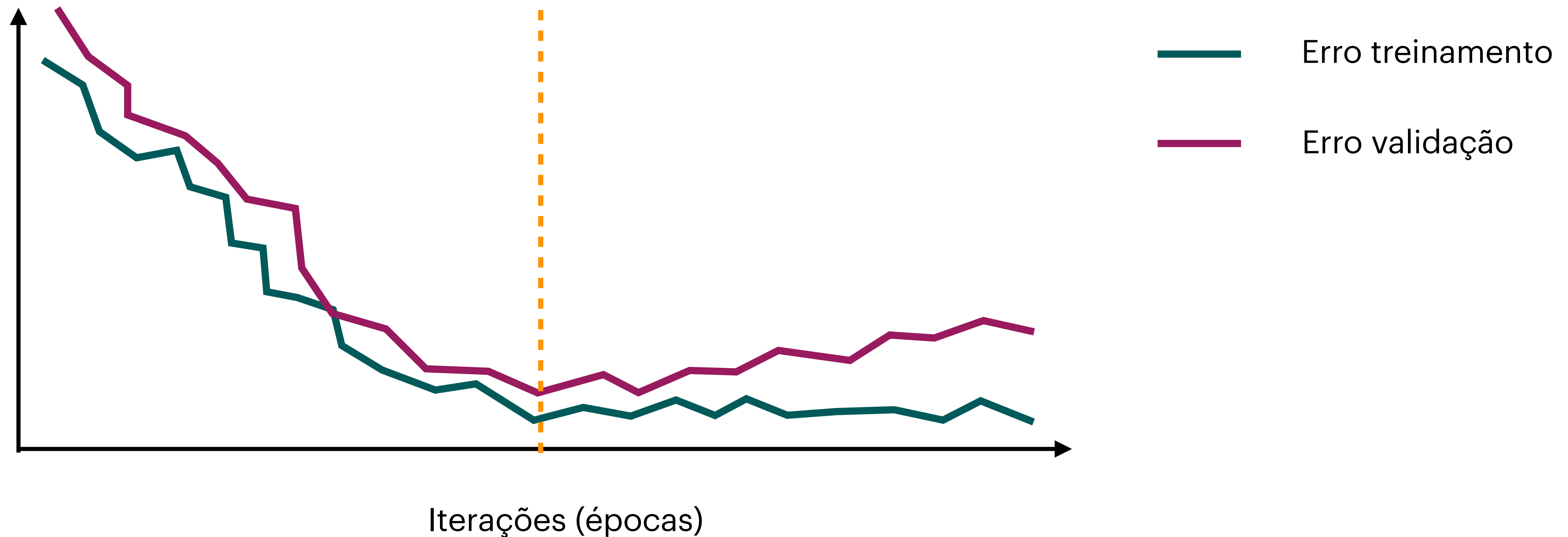
Aprendizagem por “supressão de nós”

- * Esta forma de regularização consiste em suprimir uma proporção ϕ de “nós” da rede cada vez que uma observação do conjunto de treinamento é processada.
- * Na prática, isso significa que a função de ativação nesses nós é substituída pela função nula.

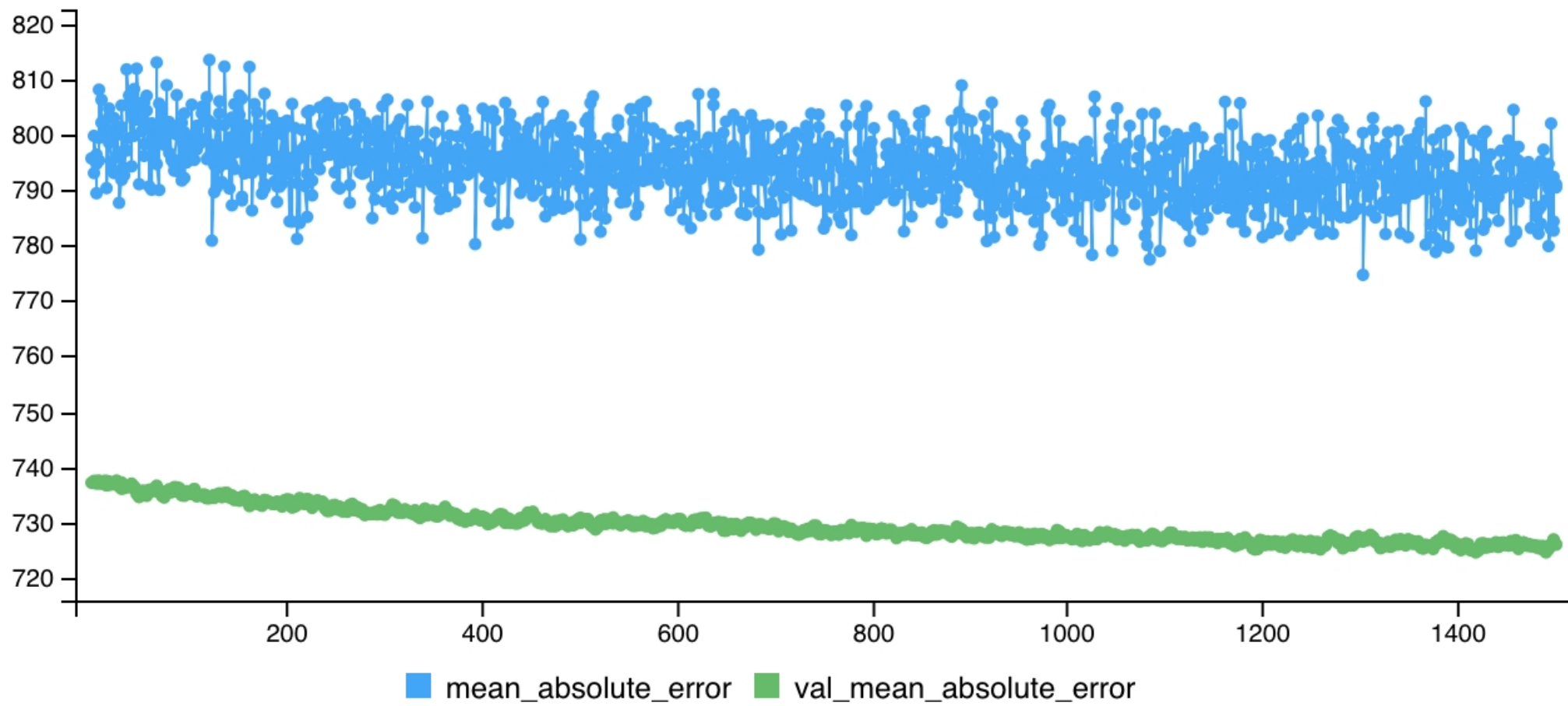
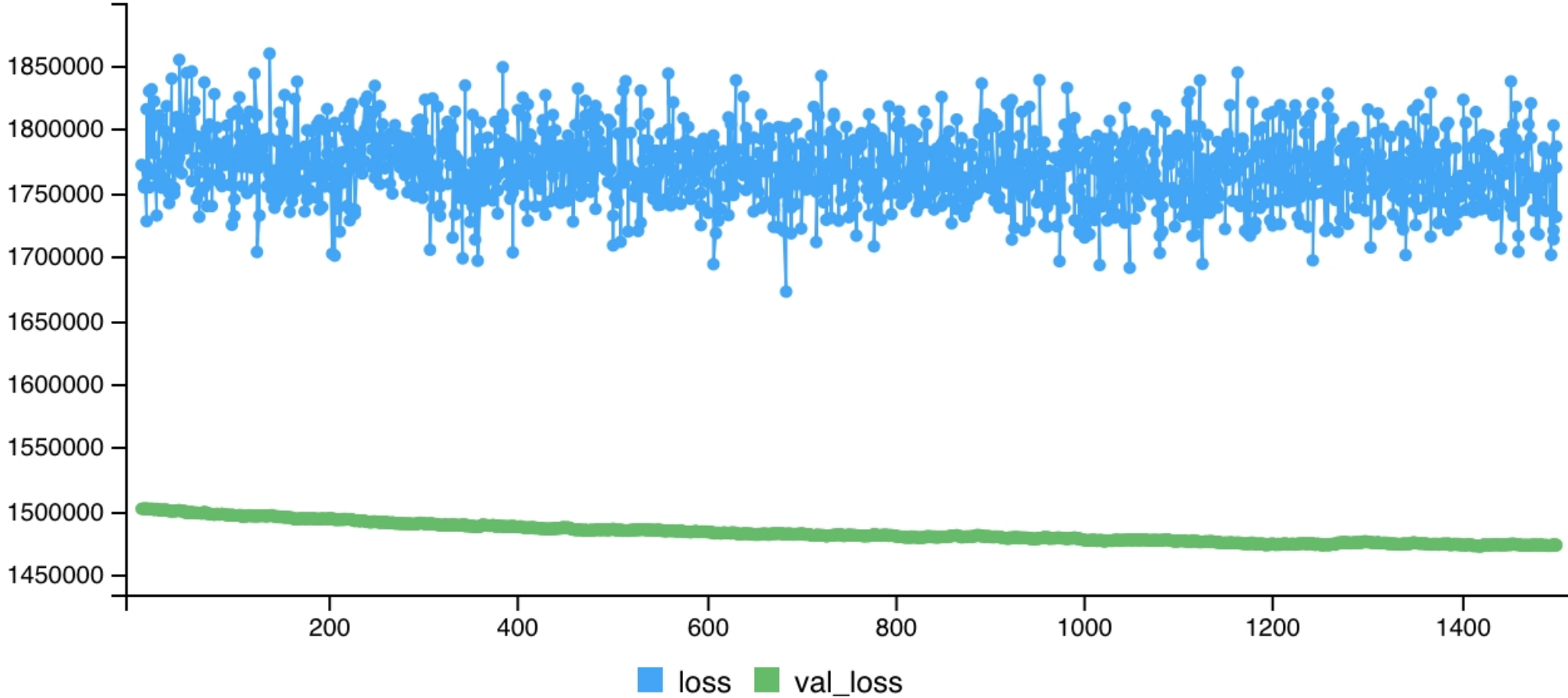
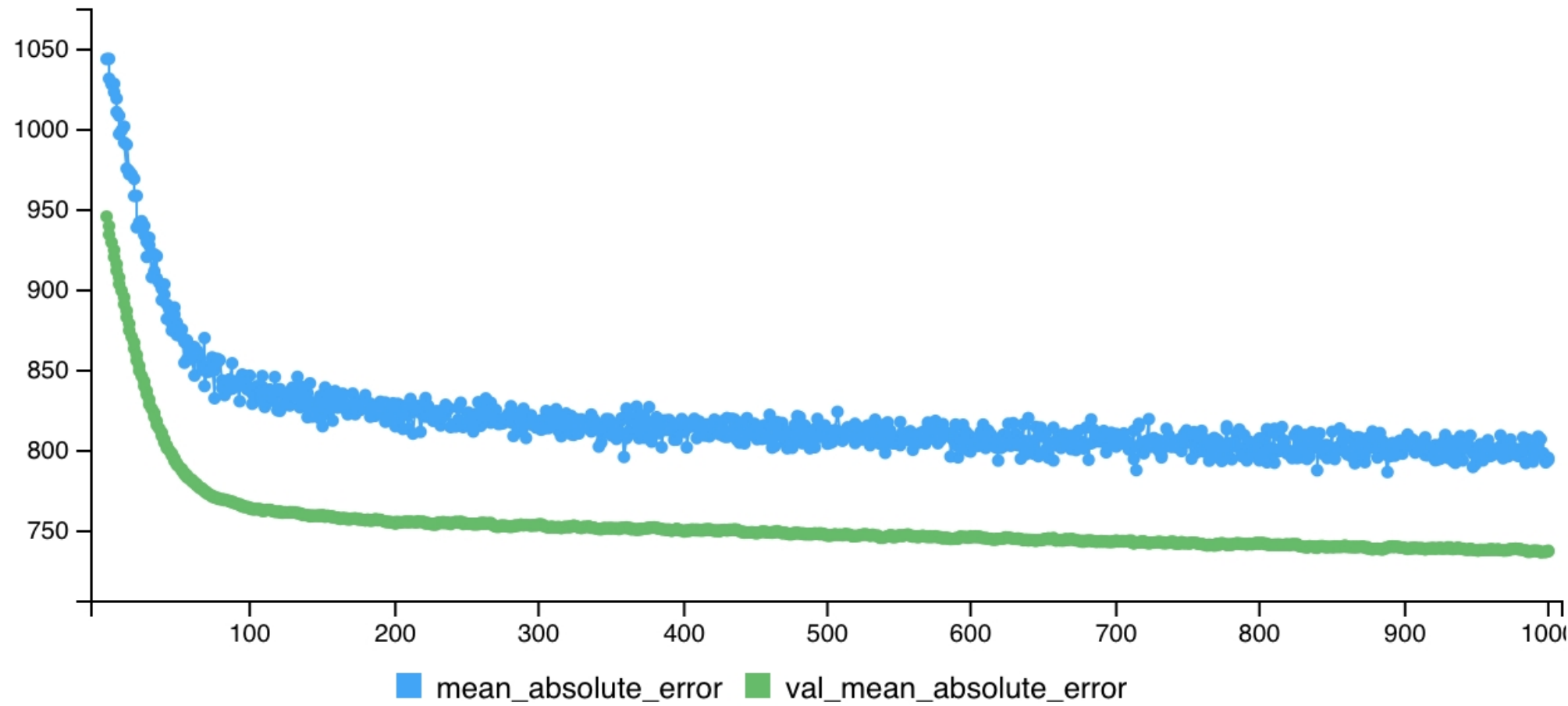
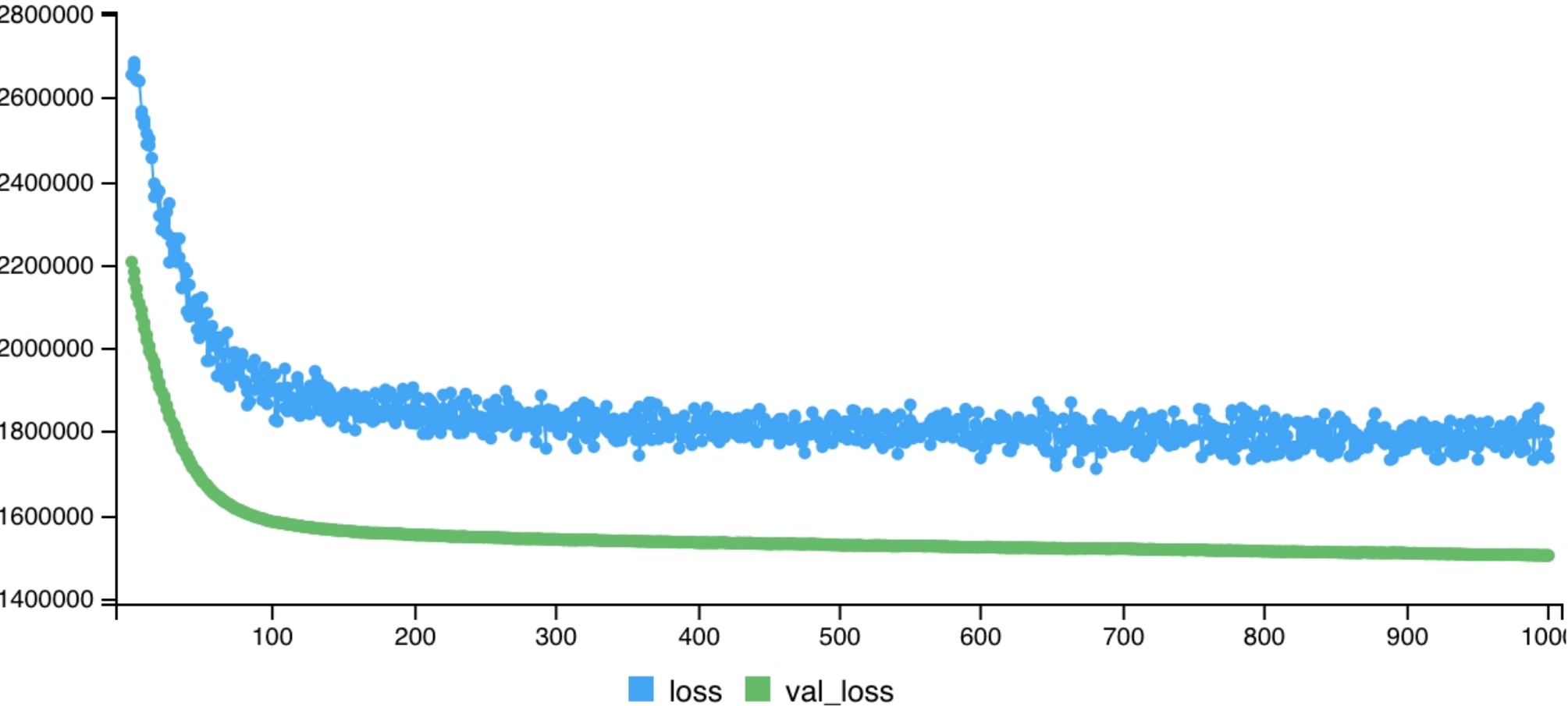


Parada antecipada

Consiste em monitorar o erro numa amostra de validação como função das iterações do algoritmo de otimização e interromper o algoritmo quando o erro na amostra de validação começar a aumentar



Parada anticipada



Preço do aluguel em São Paulo

	Erro validação ⁽¹⁾	Erro teste ⁽¹⁾
Modelo linear	775,98	842,40
LASSO	769,57	833.63
Redes neurias ⁽²⁾	720,76	793,05

Dados de apartamentos de até 200 m2 disponíveis para aluguel.

(1) Erro absoluto médio

(2) Rede neural com 50 unidades, função de ativação ReLU e supressão de 40%