

Aprendizagem estatística em altas dimensões

Florencia Leonardi

Conteúdo

- * Modelo linear para classificação (regressão logística)
- * Algoritmo do gradiente descendente
- * Naive Bayes
- * K-vizinhos mais próximos

Modelo linear para regressão

No modelo de regressão linear, a família \mathcal{G} é uma família de funções lineares:

$$\mathcal{G} = \{g(x) = x^T \beta, \beta \in \mathbb{R}^{p+1}\}$$

$$x = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$$

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

$$x^T \beta = (1 \ x_1 \ \dots \ x_p) \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}}_{\mathbb{R}^{(p+1) \times 1}} = \beta_0 + \sum_{j=1}^p x_j \beta_j \underbrace{\quad}_{\mathbb{R}^{1 \times 1}}$$

Modelo linear para regressão

Para os modelos de regressão em geral usamos a função de custo quadrática

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

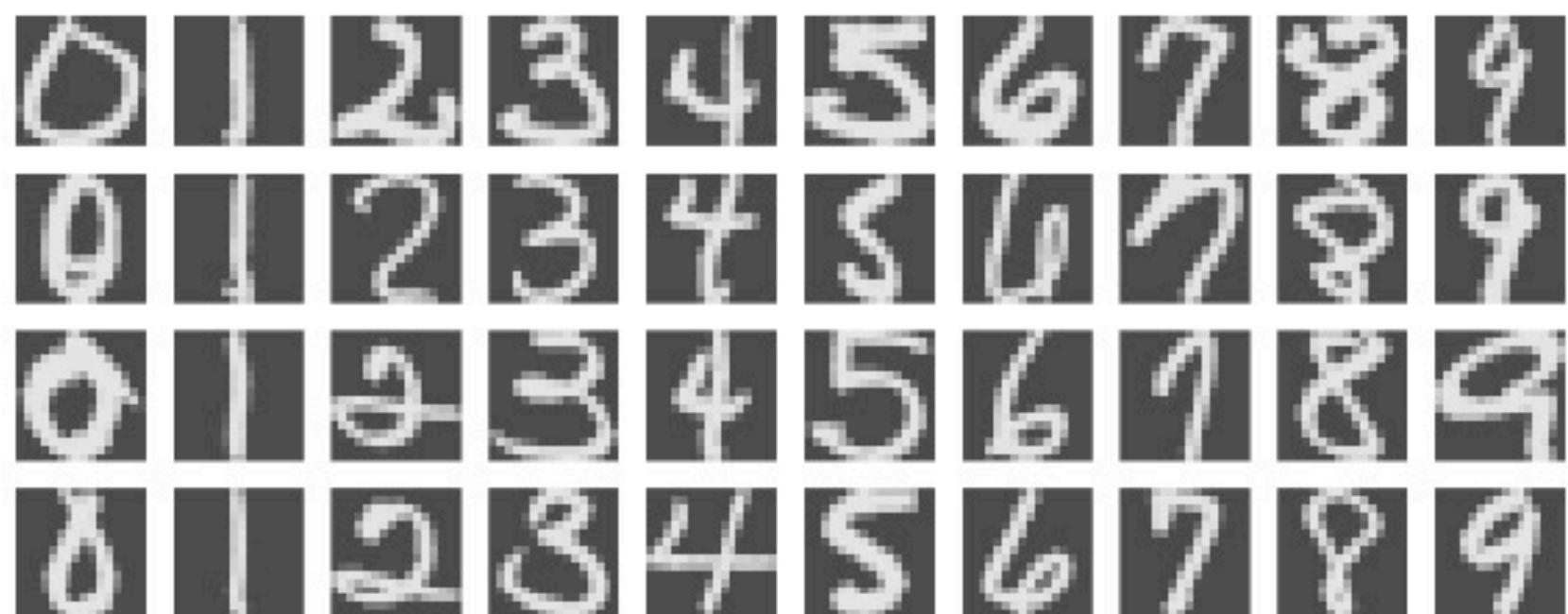
Escolhemos $g \in \mathcal{G}$ que minimize $\widehat{E}_D(g) = \frac{1}{n} \sum_{i=1}^n (y_i - g(x_i))^2$

Isto é equivalente a escolher $\beta \in \mathbb{R}^{p+1}$ que minimize $\widehat{E}_D(\beta) = \frac{1}{n} \sum_{i=1}^n (y_i - x_i^T \beta)^2$

O vetor β obtido pela minimização de $\widehat{E}_D(\beta)$, denotado por $\hat{\beta}$, é conhecido como estimador de mínimos quadrados

Modelo linear para classificação?

- * Num problema de classificação temos $y \in \mathcal{Y} = \{c_1, \dots, c_K\}$ então o modelo linear anterior não é adequado para modelar y diretamente ...
- * Neste caso podemos modelar como resposta o vetor de probabilidades condicionais $p(y = c_k | x)$, $k = 1, \dots, K$



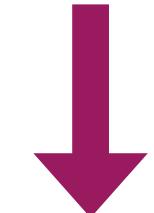
$$\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

$$x_i \in \mathcal{X} = \mathbb{R}^{16 \times 16}$$

$$y_i \in \mathcal{Y} = \{0, 1, \dots, 9\}$$

$$(x_i, y_i) \sim p(x, y)$$

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



Funções objetivo

Modelo linear para classificação?

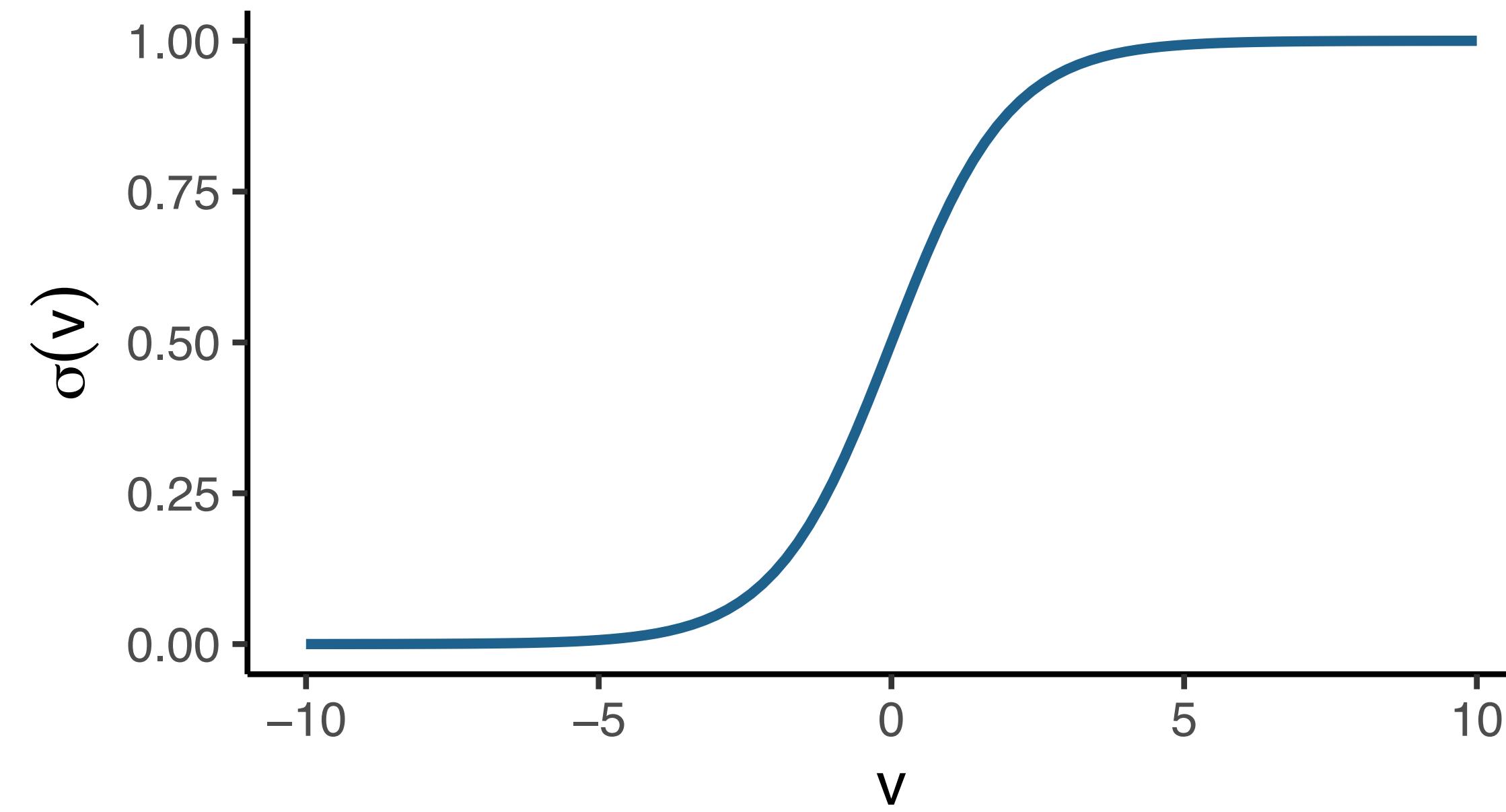
- * Se nós obtermos estimativas $g_{c_k}(x) = \hat{p}(c_k | x)$ para $k = 1, \dots, K$, então podemos predizer (classificar) uma nova observação x como $\hat{y} = \arg \max_{c_k} g_{c_k}(x)$
- * Como as probabilidades pertencem ao intervalo $[0,1]$, não podemos modelar diretamente $f_{c_k}(x)$ como funções lineares, já que pode acontecer de $x^T \beta$ não pertencer a este intervalo e a predição não fazer sentido
- * Então usamos uma função que “mapeia” as predições lineares $x^T \beta$ para o intervalo $[0,1]$

Regressão logística binária

Neste caso podemos considerar simplesmente $f_1(x) = p(1 | x)$, já que $f_0(x) = p(0 | x) = 1 - f_1(x)$

A função utilizada para levar a transformação linear $x^T \beta$ no intervalo $[0,1]$ é em geral a *função sigmoide*:

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



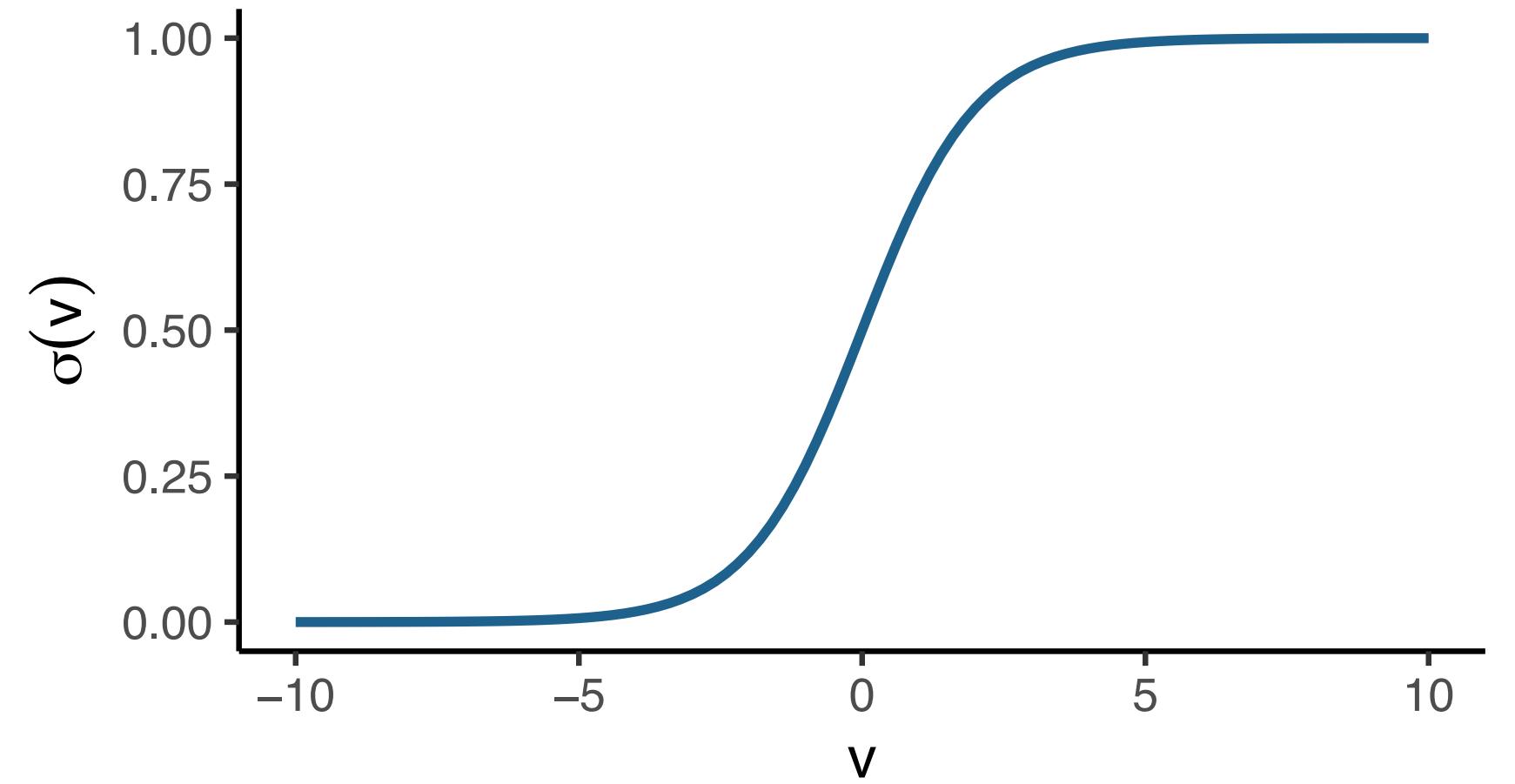
Regressão logística binária

A família \mathcal{G} de funções para aproximar $f_1(x)$ é então:

$$\mathcal{G} = \{g(x) = \sigma(x^T \beta), \beta \in \mathbb{R}^{p+1}\}$$

Uma vez obtida g a partir dos dados, podemos classificar novas observações x seguindo a regra:

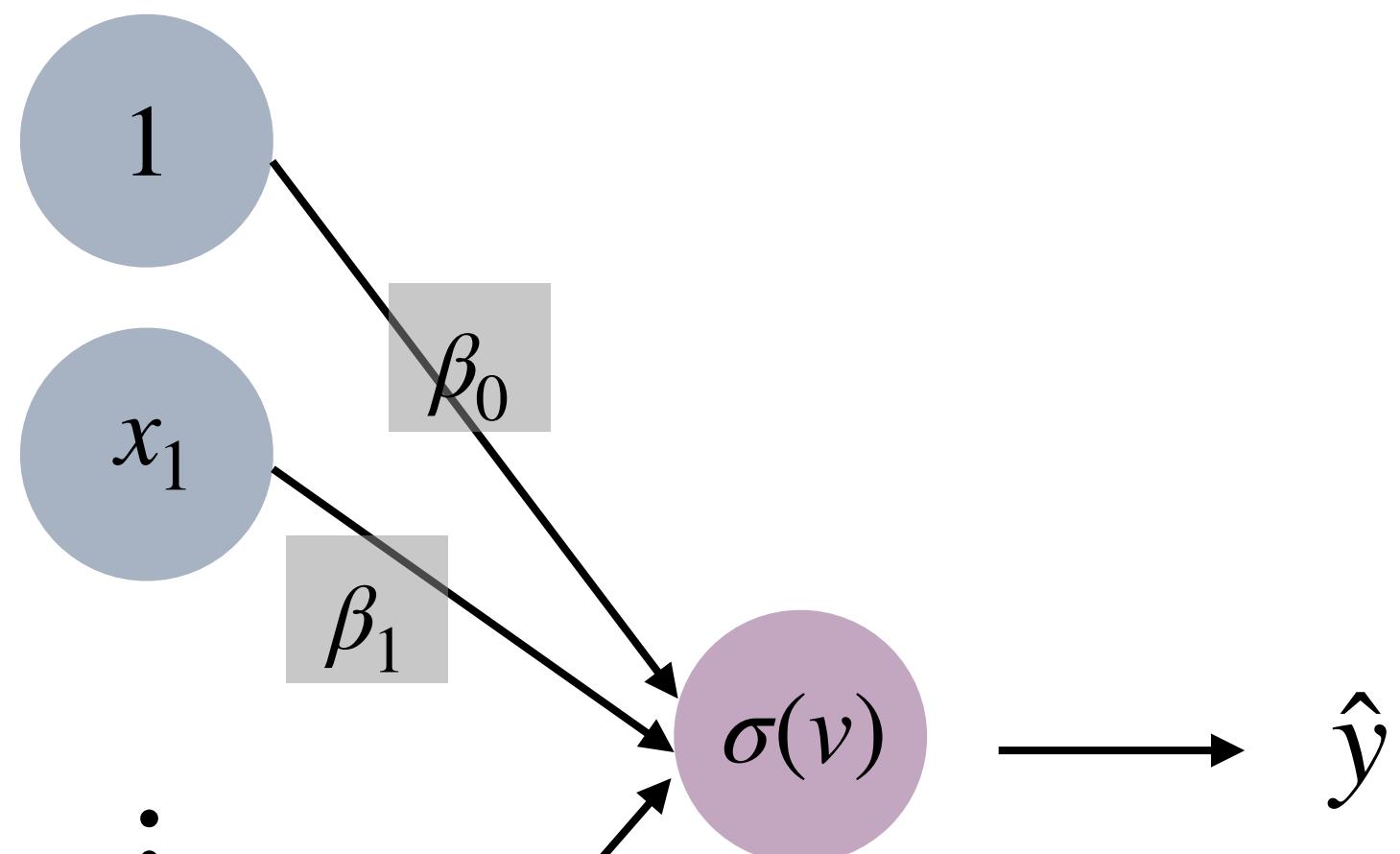
$$\hat{y} = \begin{cases} 1, & \text{se } g(x) \geq 1 - g(x) \\ 0, & \text{se } g(x) < 1 - g(x) \end{cases}$$



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

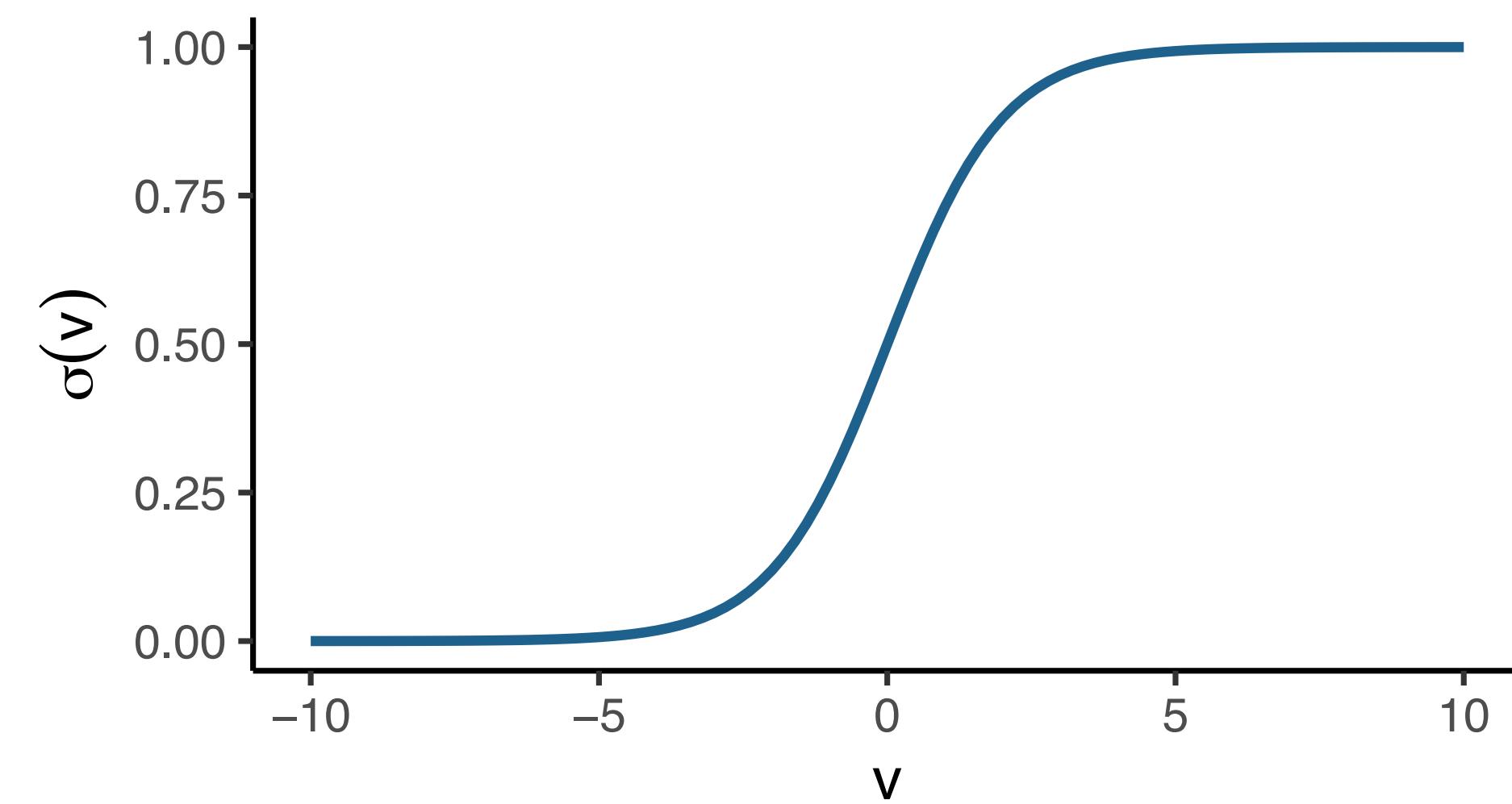
Regressão logística binária

Este modelo pode ser representado por uma “rede”



$$v = x^T \beta = \beta_0 + \sum_{j=1}^p \beta_j x_j$$

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



Função de custo

A função de custo utilizada em regressão logística é derivada do critério de máxima verossimilhança e está definida como:

$$L(y, g(x)) = - (1 - y)\log(1 - g(x)) - y \log g(x)$$

Observemos que se $g(x) = p(y = 1 | x)$ então

$$\exp[-L(y, g(x))] = [1 - g(x)]^{(1-y)} g(x)^y = \begin{cases} 1 - g(x), & \text{se } y = 0 \\ g(x), & \text{se } y = 1 \end{cases}$$

Então ao minimizar $L(y, g(x))$ estamos maximizando a probabilidade de y dado x

Estimação

Para $\mathcal{G} = \{g(x) = \sigma(x^T \beta), \beta \in \mathbb{R}^{p+1}\}$

e $L(y, g(x)) = - (1 - y)\log(1 - g(x)) - y \log g(x)$

procuramos $g \in \mathcal{G}$ que minimize

$$\widehat{E}_D(g) = \frac{1}{n} \sum_{i=1}^n L(y_i, g(x_i)) = -\frac{1}{n} \sum_{i=1}^n [(1 - y_i)\log(1 - g(x_i)) + y_i \log g(x_i)]$$

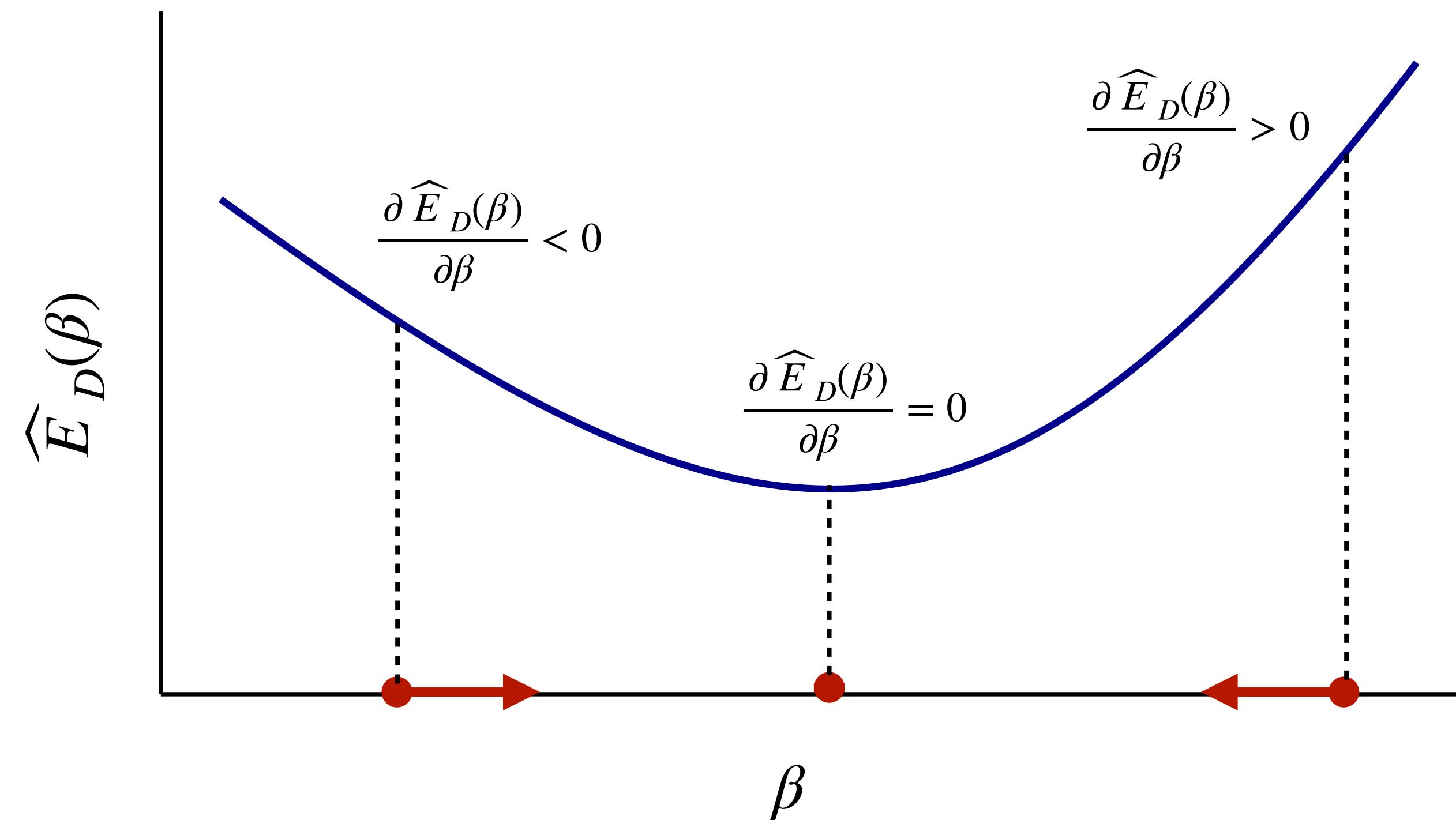
Isto é equivalente a escolher $\beta \in \mathbb{R}^{p+1}$ que minimize

$$\widehat{E}_D(\beta) = \frac{1}{n} \sum_{i=1}^n L(y_i, \sigma(x_i^T \beta)) = -\frac{1}{n} \sum_{i=1}^n [(1 - y_i)\log(1 - \sigma(x^T \beta)) + y_i \log \sigma(x^T \beta)]$$

Algoritmo “gradiente descendente”

Como no caso de regressão linear, $\widehat{E}_D(\beta)$ é uma função convexa

Mas neste caso não há uma solução analítica para $\hat{\beta}$



$$\frac{\partial \widehat{E}_D(\beta)}{\partial \beta} = \frac{1}{n} \sum_{i=1}^n (\sigma(x_i\beta) - y_i)x_i$$

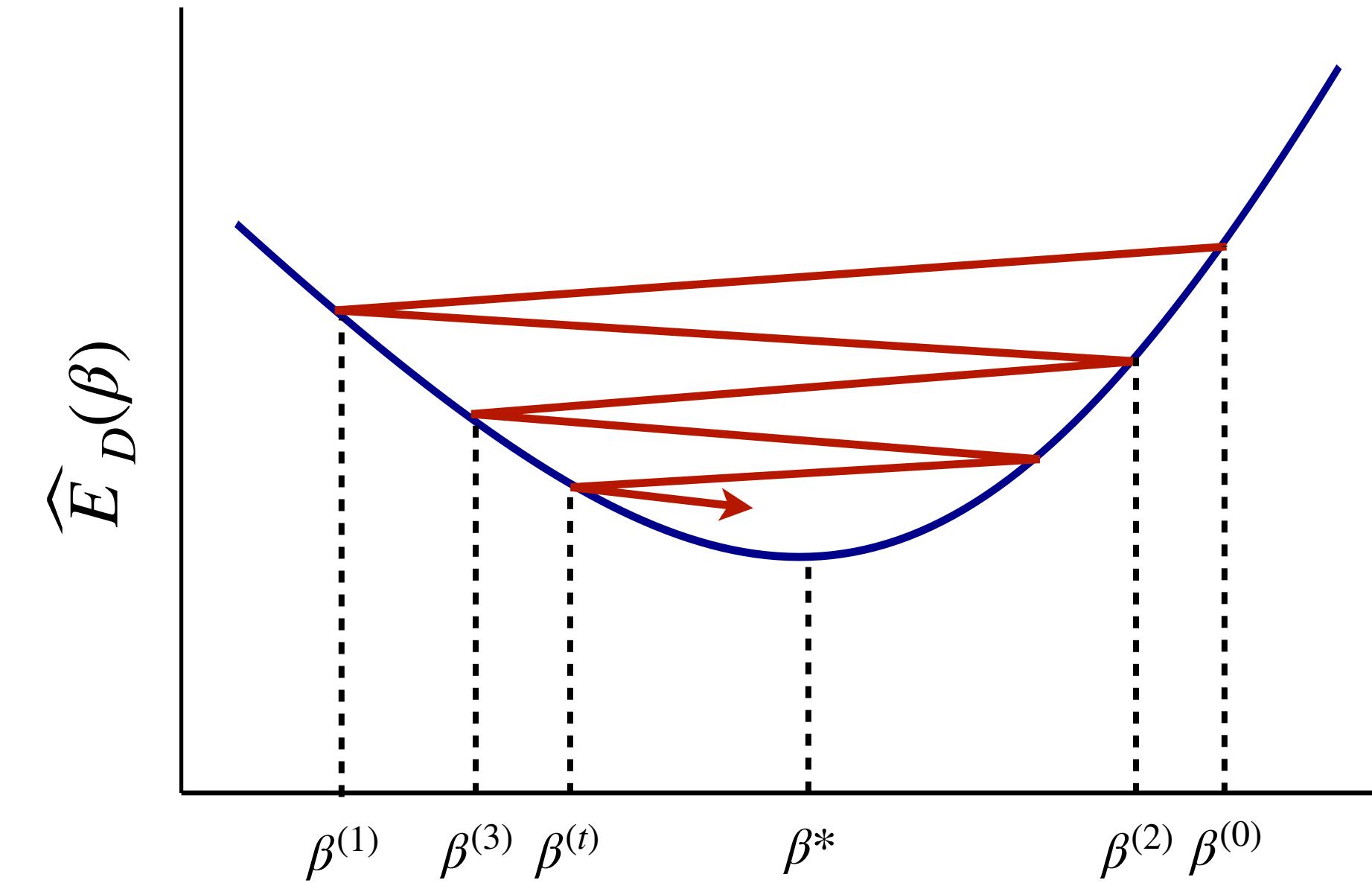
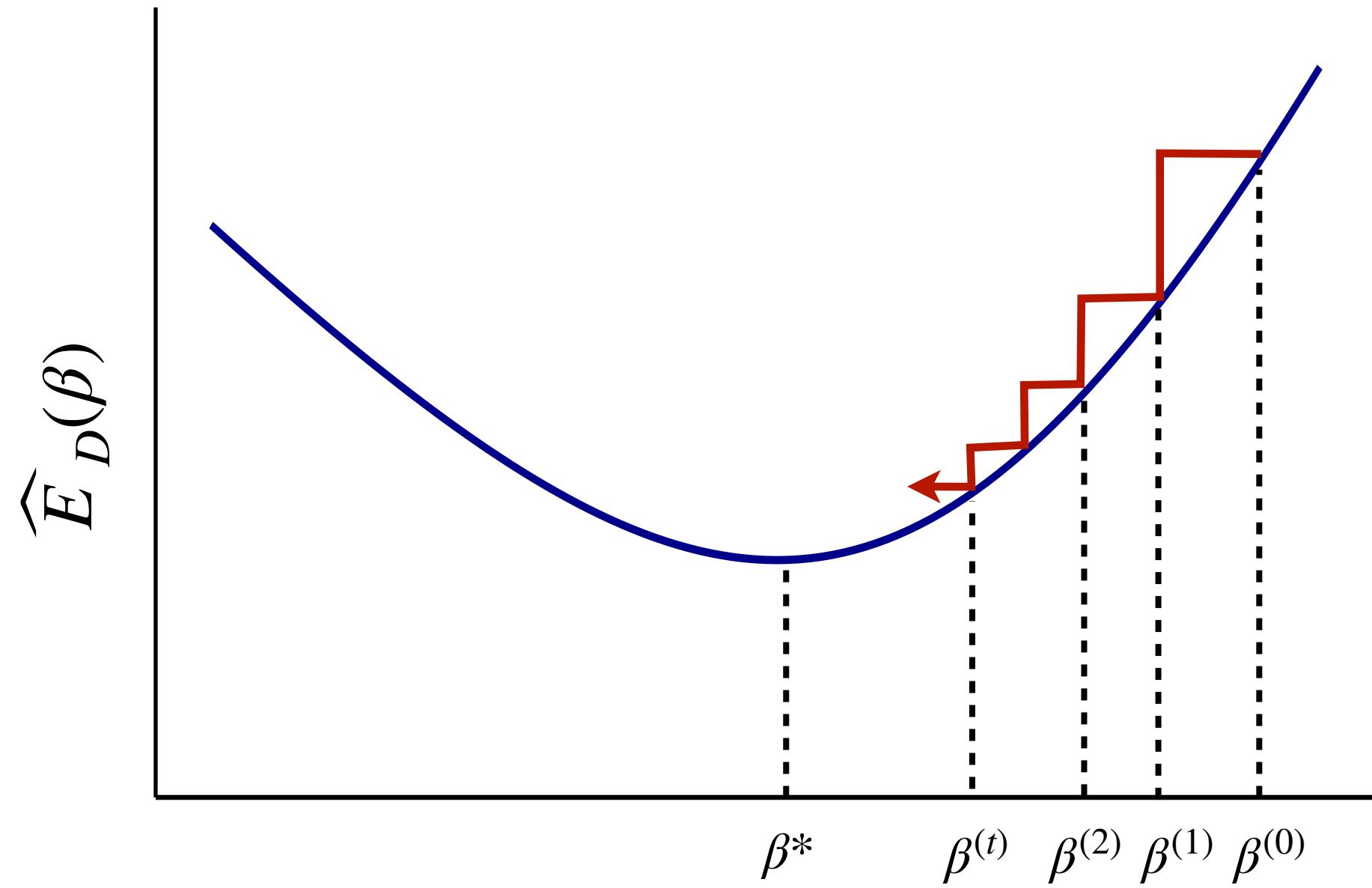
$$\beta^{(t)} = \beta^{(t-1)} - \eta \frac{\partial \widehat{E}_D(\beta^{(t-1)})}{\partial \beta}$$

Algoritmo “gradiente descendente”

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

$$\beta^{(t)} = \beta^{(t-1)} - \eta \frac{\partial \widehat{E}_D(\beta)}{\partial \beta}$$

η (taxa de aprendizagem)



Algoritmo “gradiente descendente”

$$\nabla \widehat{E}_D(\beta) = \begin{pmatrix} \frac{\partial \widehat{E}_D}{\partial \beta_0}(\beta) \\ \vdots \\ \frac{\partial \widehat{E}_D}{\partial \beta_p}(\beta) \end{pmatrix} = \begin{pmatrix} \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i) \\ \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i)x_{i1} \\ \vdots \\ \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i)x_{ip} \end{pmatrix}$$
$$= \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i)x_i$$

$$x_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

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

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

η (taxa de aprendizagem)

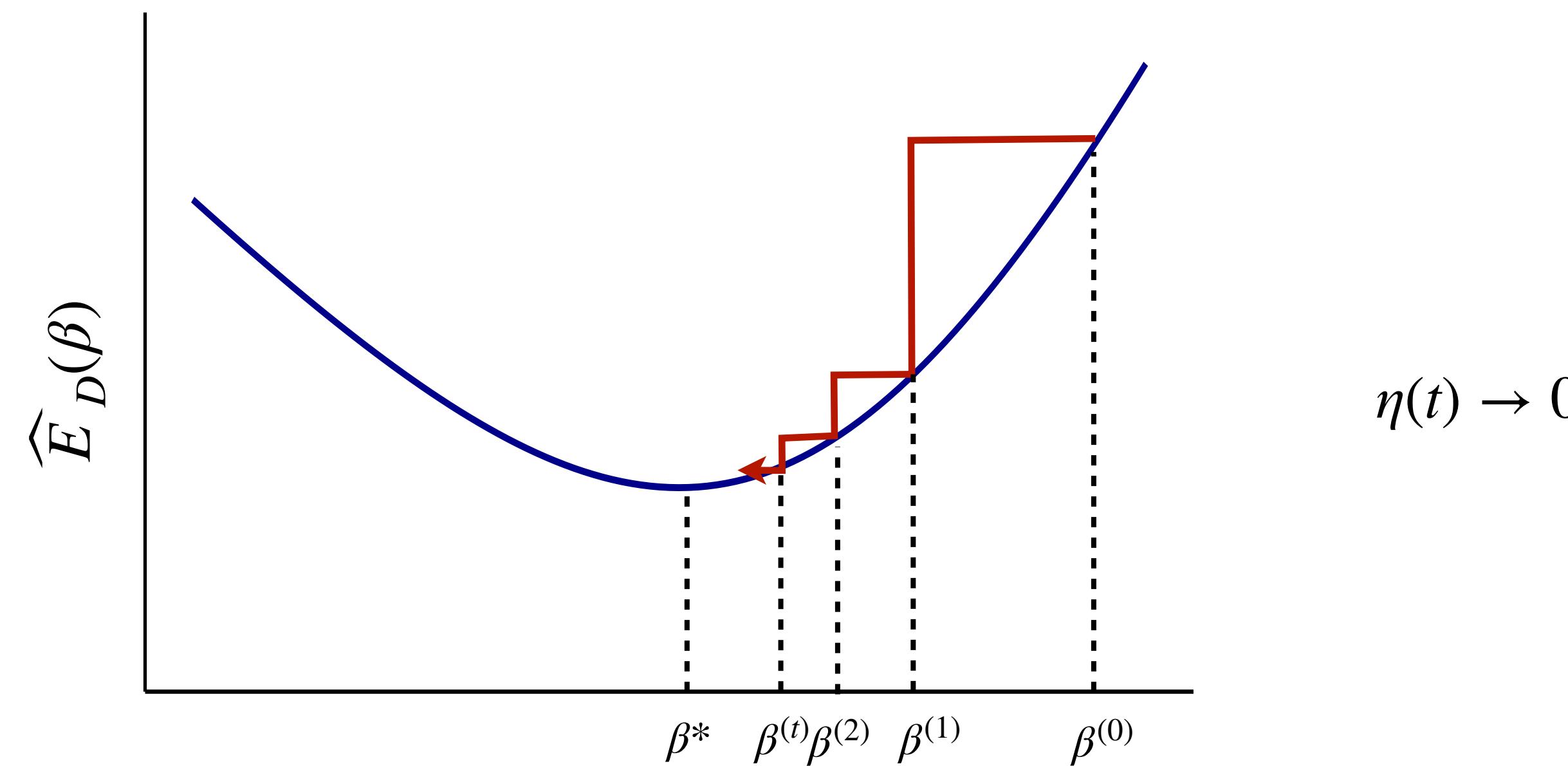
Algoritmo “gradiente descendente”

$$\nabla \widehat{E}_D(\beta) = \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i) x_i$$

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

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

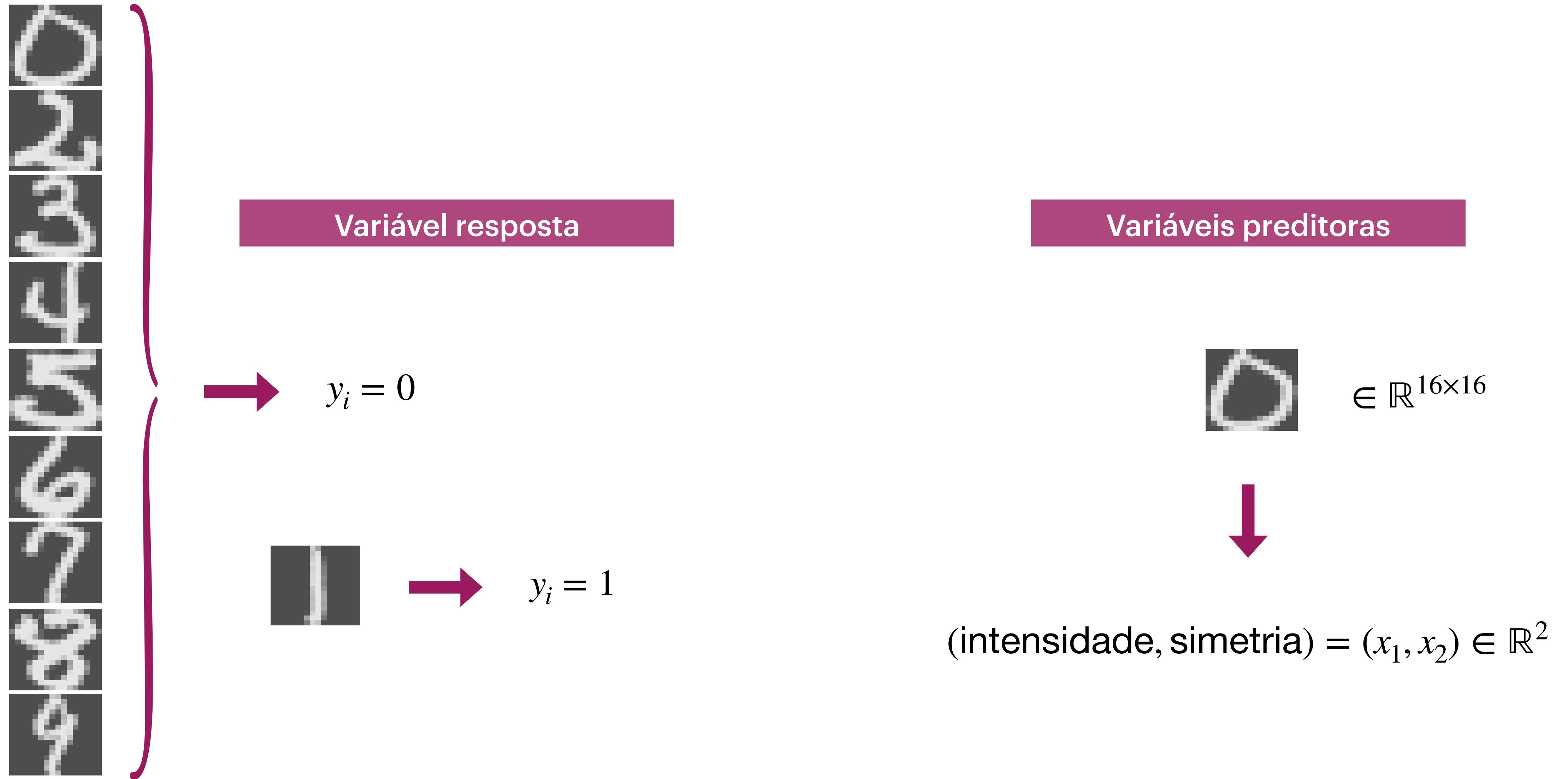
$\eta(t)$ (taxa de aprendizagem)



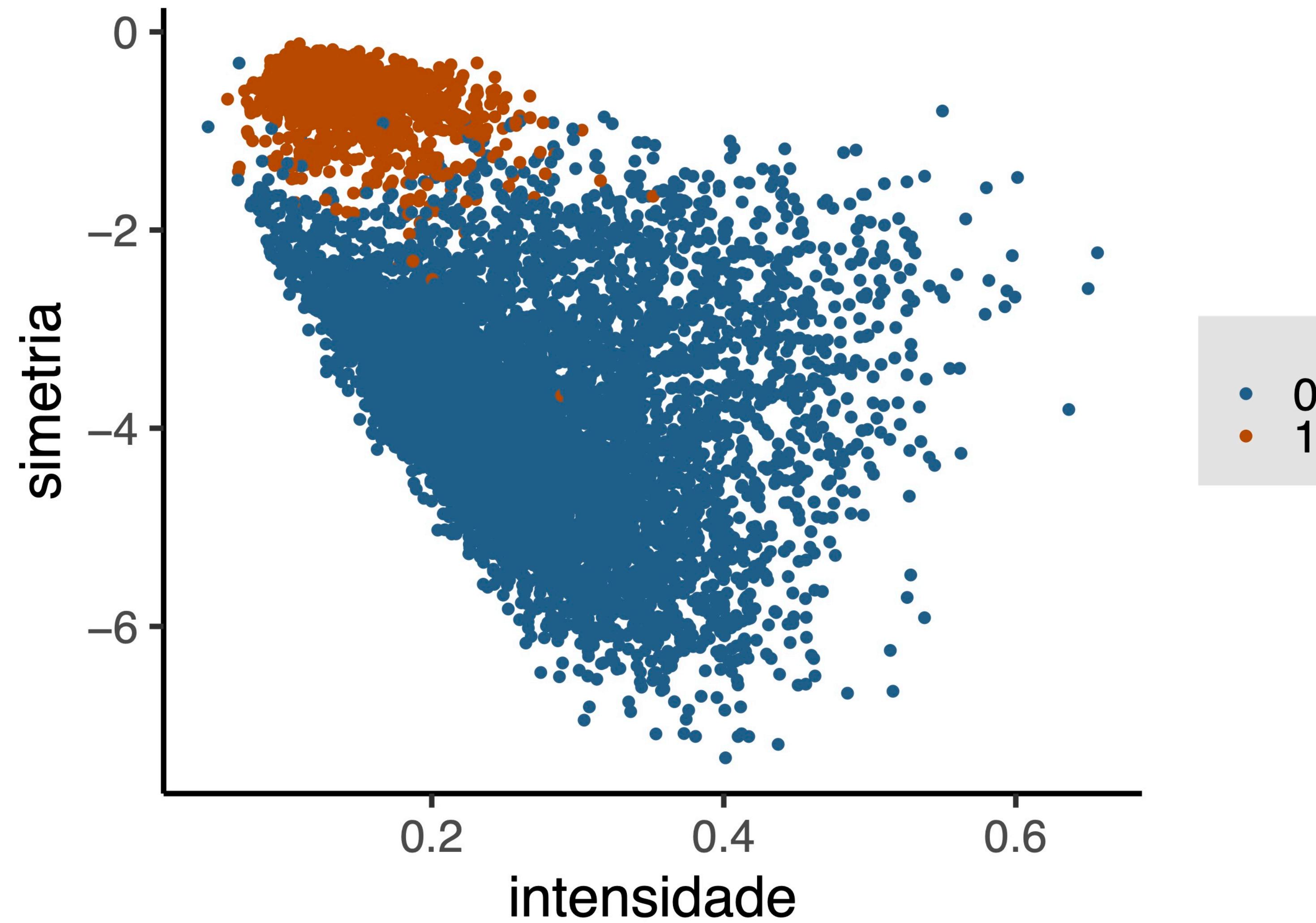
Algoritmo: gradiente descendente (regressão logística)

1. Inicialize os pesos em $t = 0$ para $\beta^{(0)} = \vec{0}$
2. Para $t = 0, 1, 2, \dots$
 - a. Calcule $\nabla \widehat{E}_D(\beta^{(t)}) = \frac{1}{n} \sum_{i=1}^n (\sigma(x_i^T \beta) - y_i)x_i$
 - b. Atualize os pesos $\beta^{(t+1)} = \beta^{(t)} - \eta \nabla \widehat{E}_D(\beta^{(t)})$
 - c. Volte ao passo 2 até que seja tempo de parar
3. Retorne a estimativa $\beta^{(t+1)}$

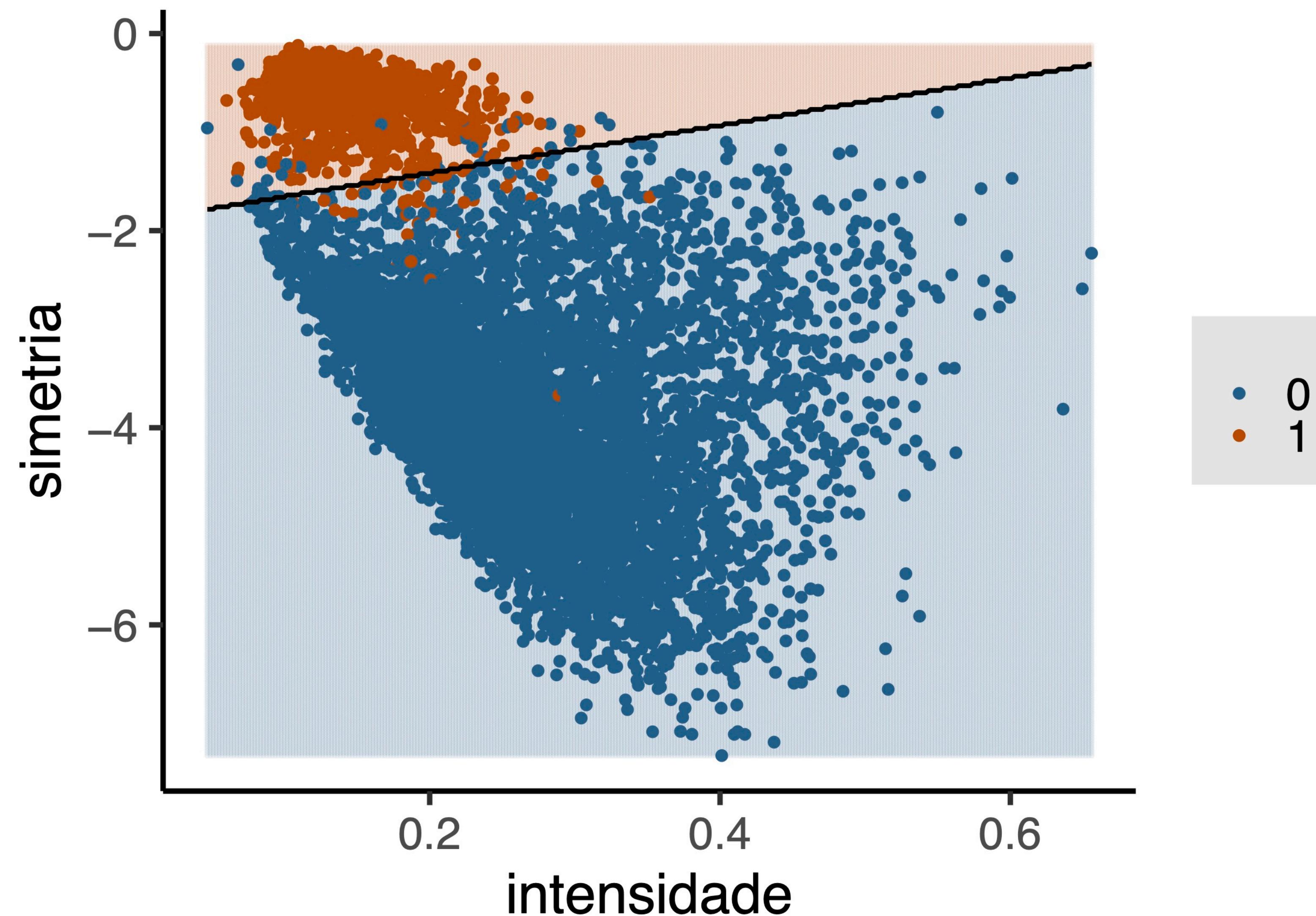
Classificação binária



Classificação binária



Regressão logística binária



Regressão logística binária

Desempenho na amostra de teste \mathcal{D}_{Te}

$$\hat{\beta} = (9.39, -11.84, 4.92)$$

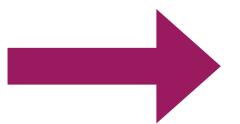
$$\hat{y} = \begin{cases} 1, & \text{se } \sigma(x^T \hat{\beta}) \geq 1 - \sigma(x^T \hat{\beta}) \\ 0, & \text{se } \sigma(x^T \hat{\beta}) < 1 - \sigma(x^T \hat{\beta}) \end{cases}$$

		Classe verdadeira	
		0	1
Classe predita	0	1732	29
	1	11	235

$$\text{Precisão} = \frac{1732 + 235}{2007} \times 100 = 98\%$$

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

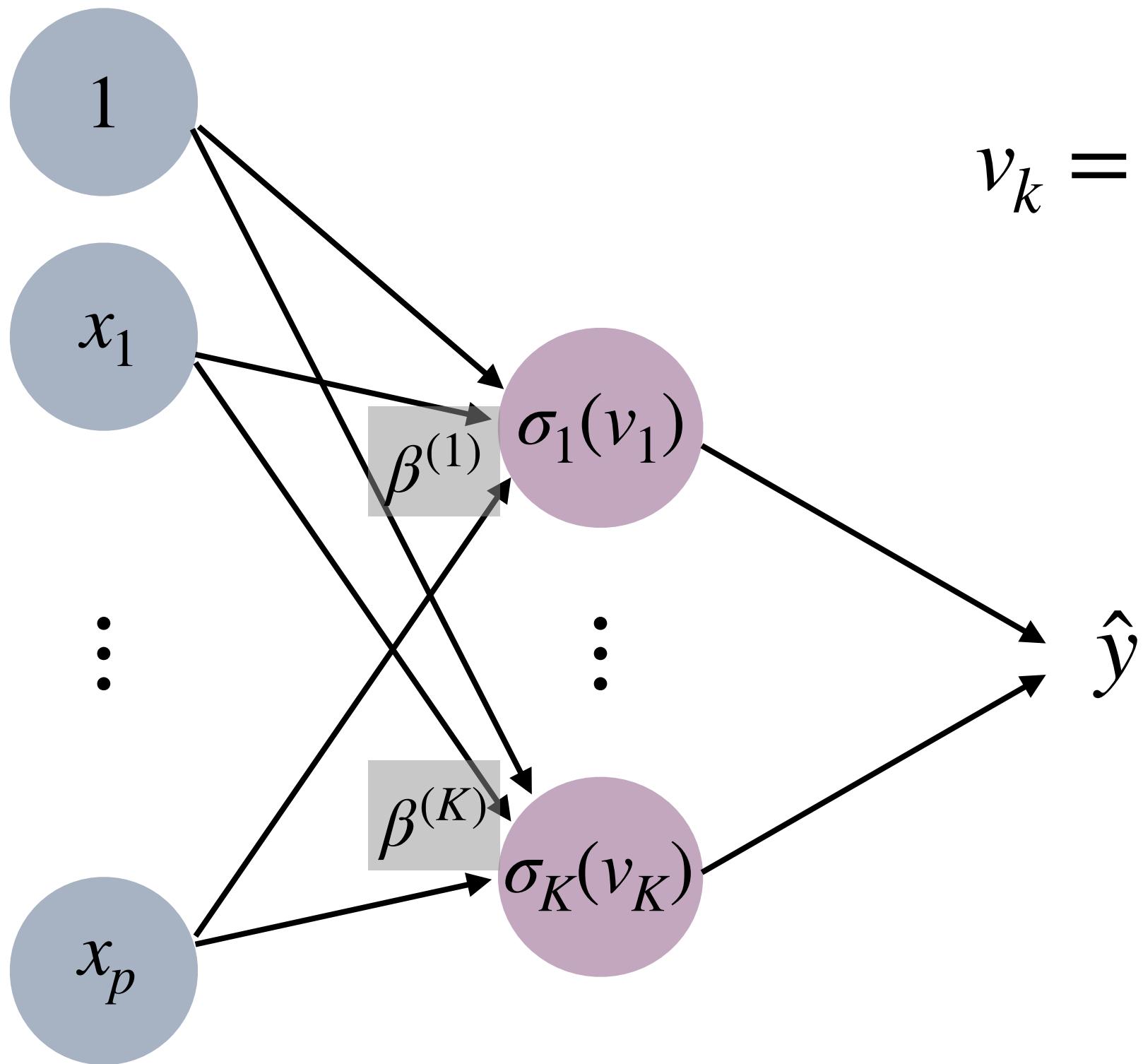
Observemos que para todo x temos que $\sum_{k=1}^K g_{c_k}(x) = 1$, logo $g_{c_k}(x), k = 1, \dots, K$ é uma distribuição de probabilidade sobre $\{c_1, \dots, c_K\}$

Regressão logística com mais de duas classes

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



Funções objetivo



$$v_k = x^T \beta^{(k)}$$

$$\sigma_k(v_k) = \frac{e^{v_k}}{\sum_{j=1}^K e^{v_j}}$$

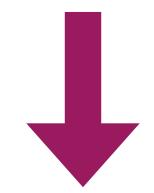
$$g_{c_k}(x) = \sigma_k(x^T \beta^{(k)})$$

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

Regularização e seleção de variáveis

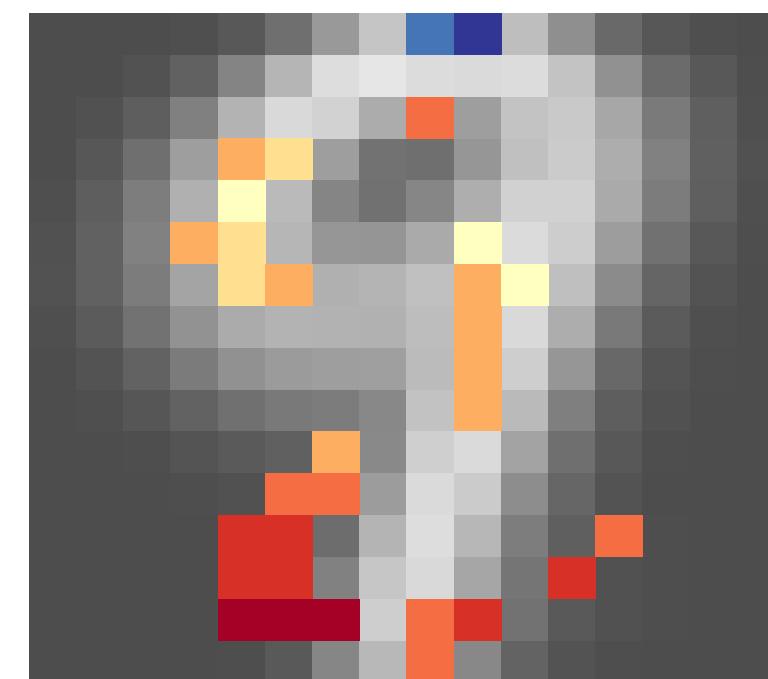
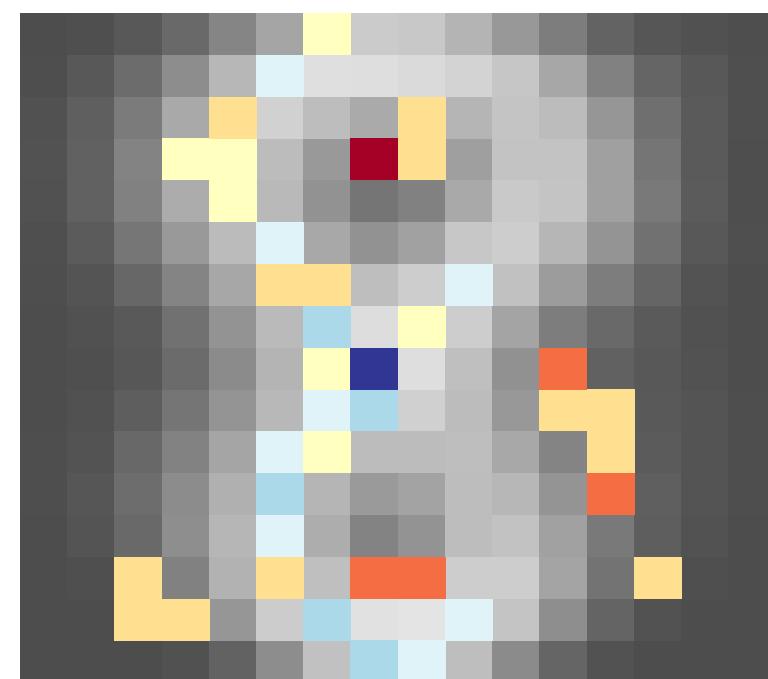
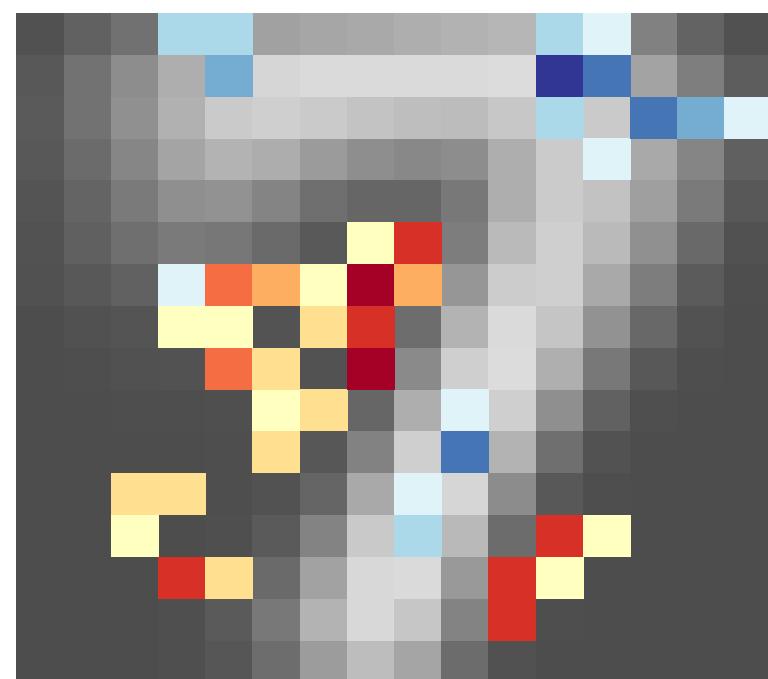
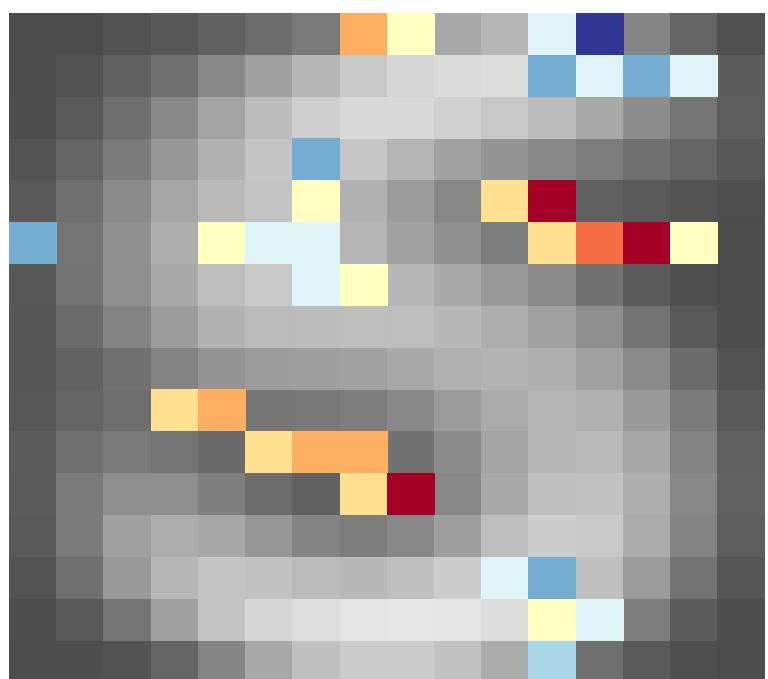
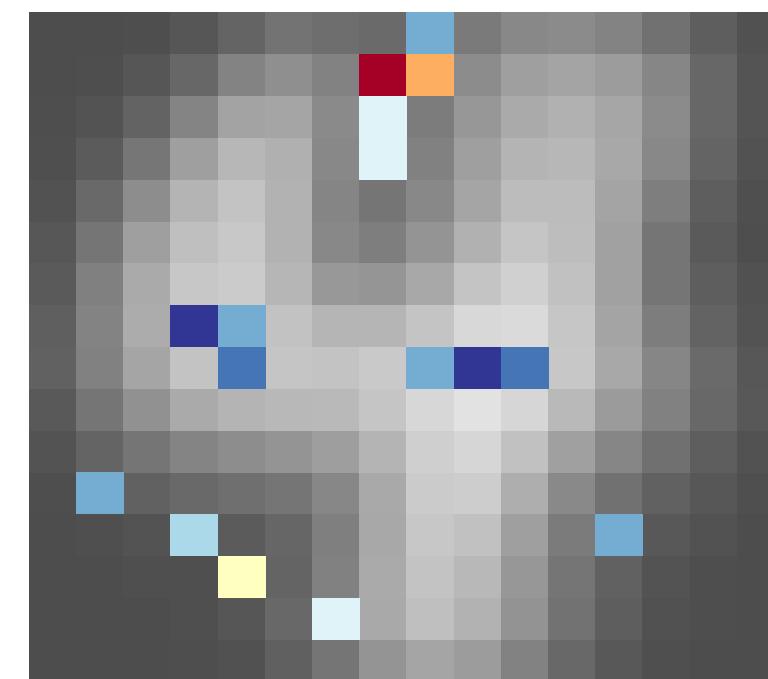
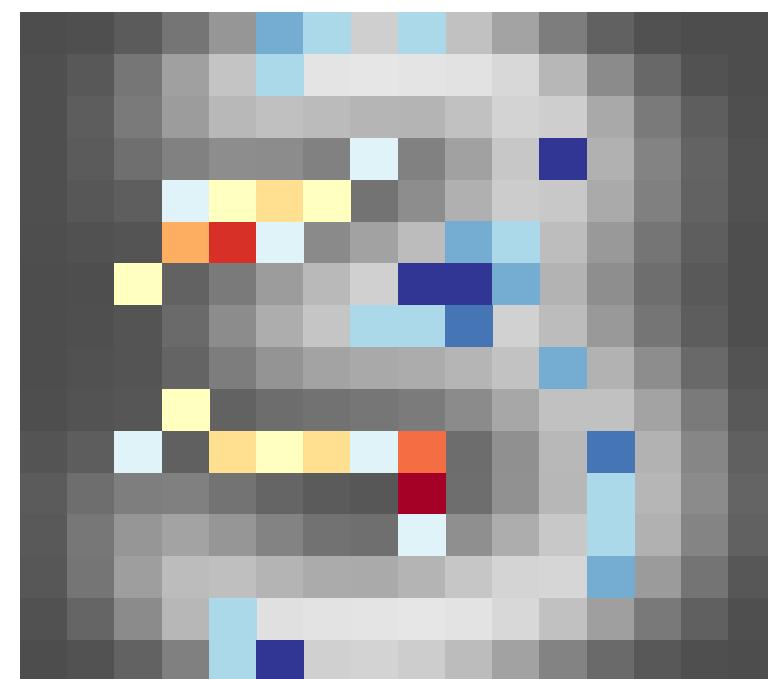
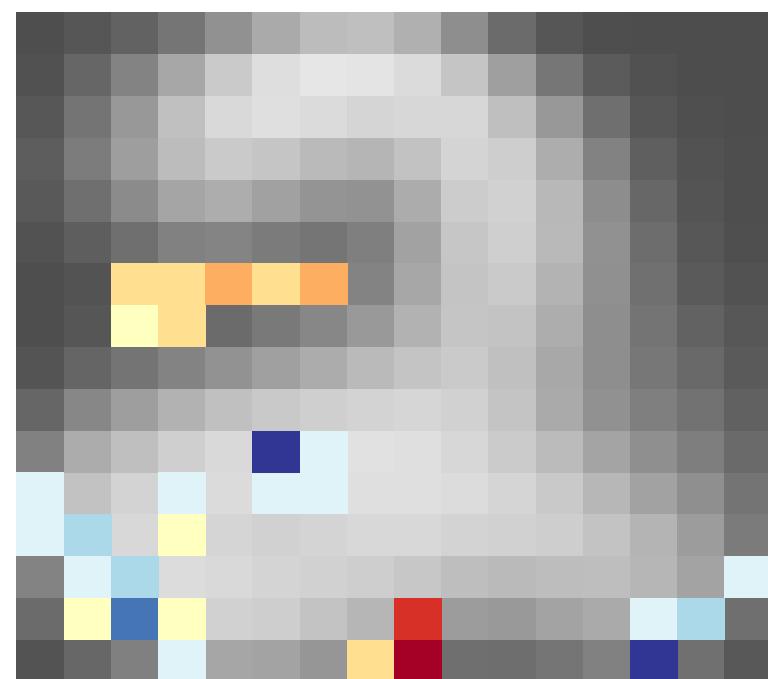
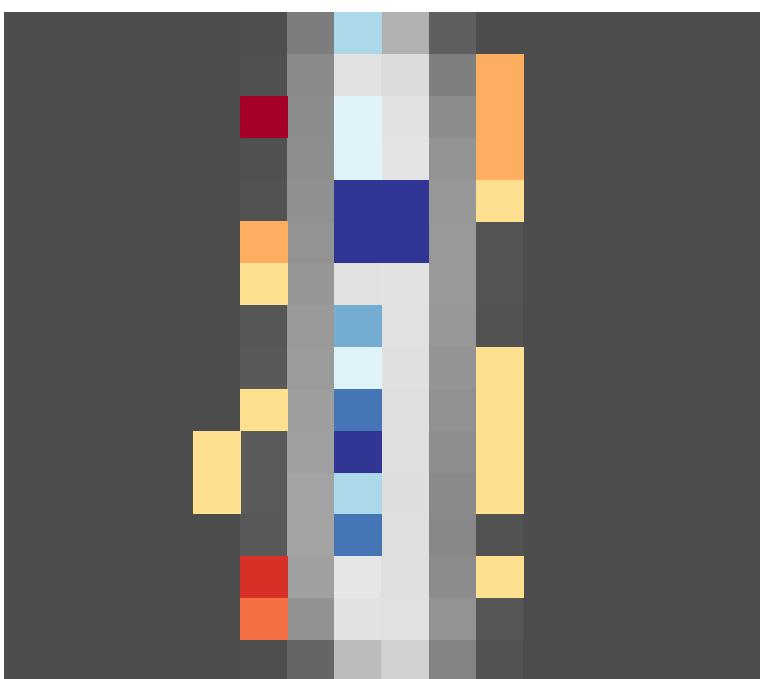
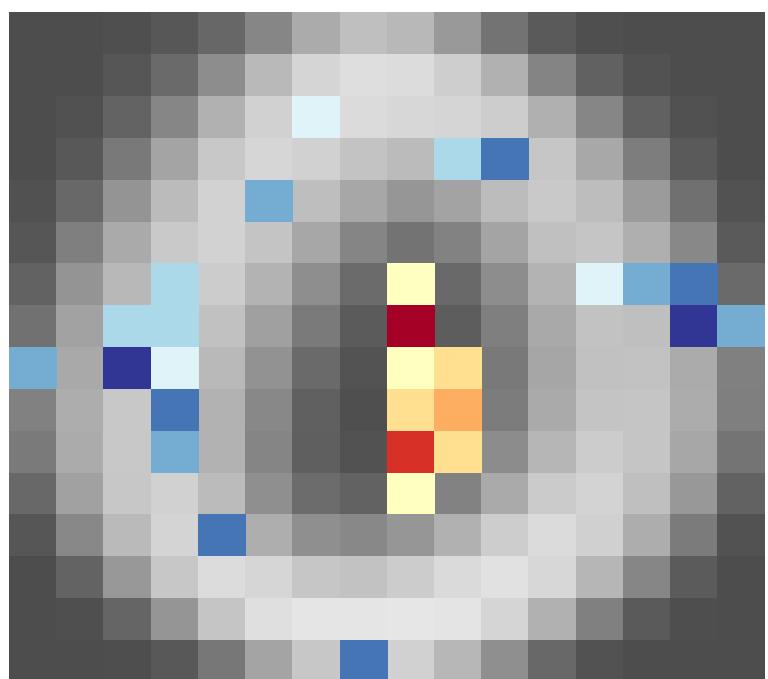
$$\begin{aligned}\widehat{E}_D(\beta) &= -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbf{1}\{y_i = c_k\} \log g_{c_k}(x) \\ &= -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbf{1}\{y_i = c_k\} x_i^T \beta^{(k)} - \log \left(\sum_{k=1}^K \exp(x_i^T \beta^{(k)}) \right)\end{aligned}$$
$$x_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}$$
$$\beta^{(k)} = \begin{pmatrix} \beta_0^{(k)} \\ \beta_1^{(k)} \\ \vdots \\ \beta_p^{(k)} \end{pmatrix}$$

Em alta dimensão, podemos escolher $\beta \in \mathbb{R}^{p+1}$ que minimize $\widehat{E}_D(\beta) + \lambda \|\beta\|_1$



Lasso para regressão logística

Modelo para cada classe de dígito



Naive Bayes

Como na regressão logística, nossa função objetivo é $f_{c_k}(x) = p(c_k | x), k = 1, \dots, K$

Pela definição de probabilidade condicional e o teorema de Bayes:

$$p(c_k | x) = \frac{p(x, c_k)}{p(x)} = \frac{p(x | c_k)p(c_k)}{p(x)} = \frac{p(x | c_k)p(c_k)}{\sum_{j=1}^K p(x | c_j)p(c_j)}$$

Aqui, $p(x | c_j)$ denota a densidade condicional à classe c_j e $p(c_j)$ é a probabilidade *a priori* da classe c_j

Naive Bayes assume que $p(x | c_j) = \prod_{i=1}^p p(x_i | c_j)$

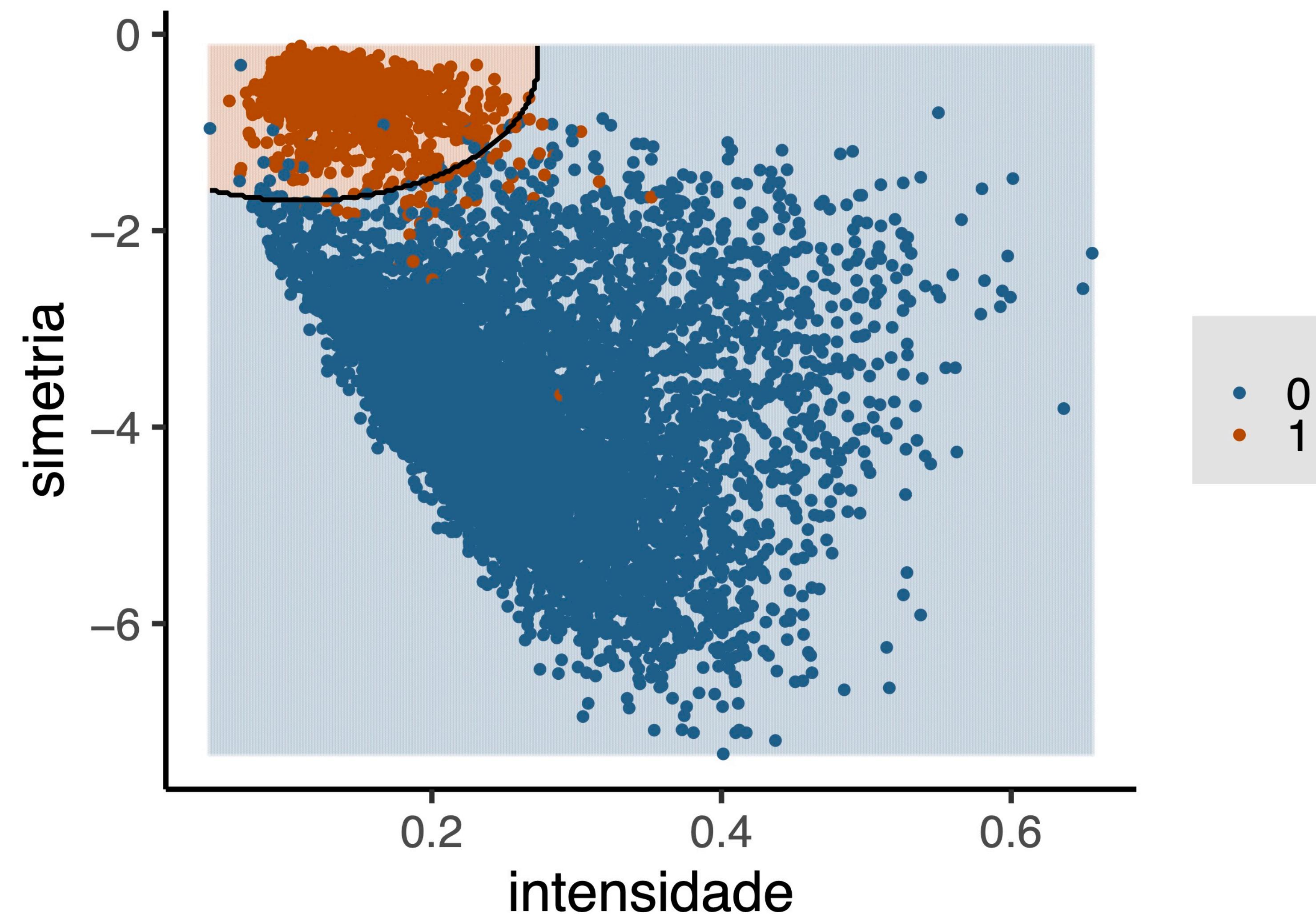
$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

Naive Bayes

$$p(c_k | x) = \frac{p(x_1 | c_k) \dots p(x_p | c_k) p(c_k)}{\sum_{j=1}^K p(x_1 | c_j) \dots p(x_p | c_j) p(c_j)}$$

- * As probabilidades à priori $p(c_j)$ podem ser estimadas facilmente pelas frequências observadas na amostra por $\hat{p}(c_j) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}\{y_i = c_j\}$
- * As densidades condicionais $p(x_i | c_j)$ podem ser estimadas por métodos paramétricos (exemplo: estimar μ_j e σ_j de uma densidade Gaussiana) ou não paramétricos (exemplo: *kernels*)

Naive Bayes



Naive Bayes

Desempenho na amostra de teste \mathcal{D}_{Te}

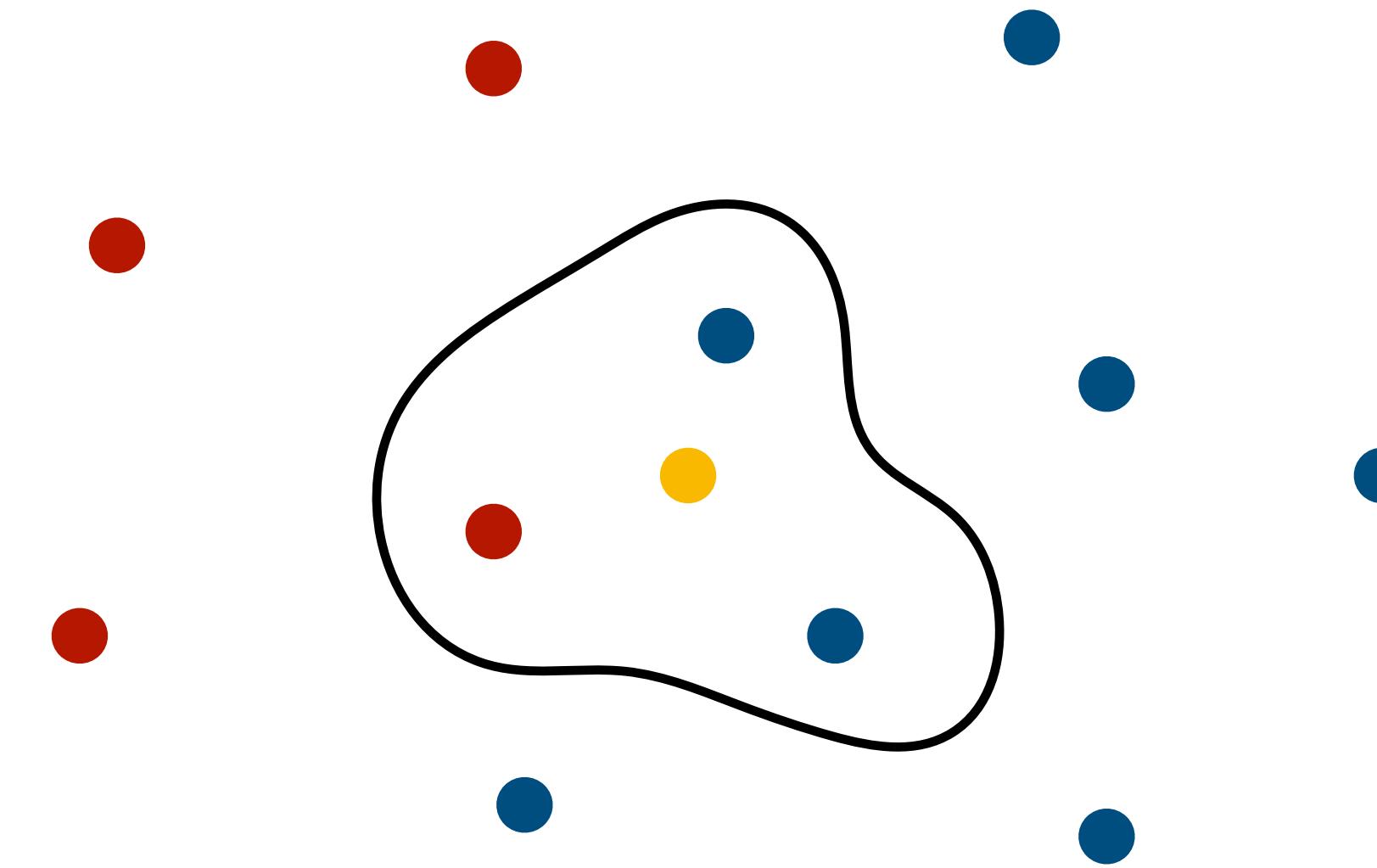
$$\hat{y} = \begin{cases} 1, & \text{se } \hat{p}(1|x) \geq \hat{p}(0|x) \\ 0, & \text{se } \hat{p}(1|x) < \hat{p}(0|x) \end{cases}$$

		Classe verdadeira	
		0	1
Classe predita	0	1733	29
	1	10	235

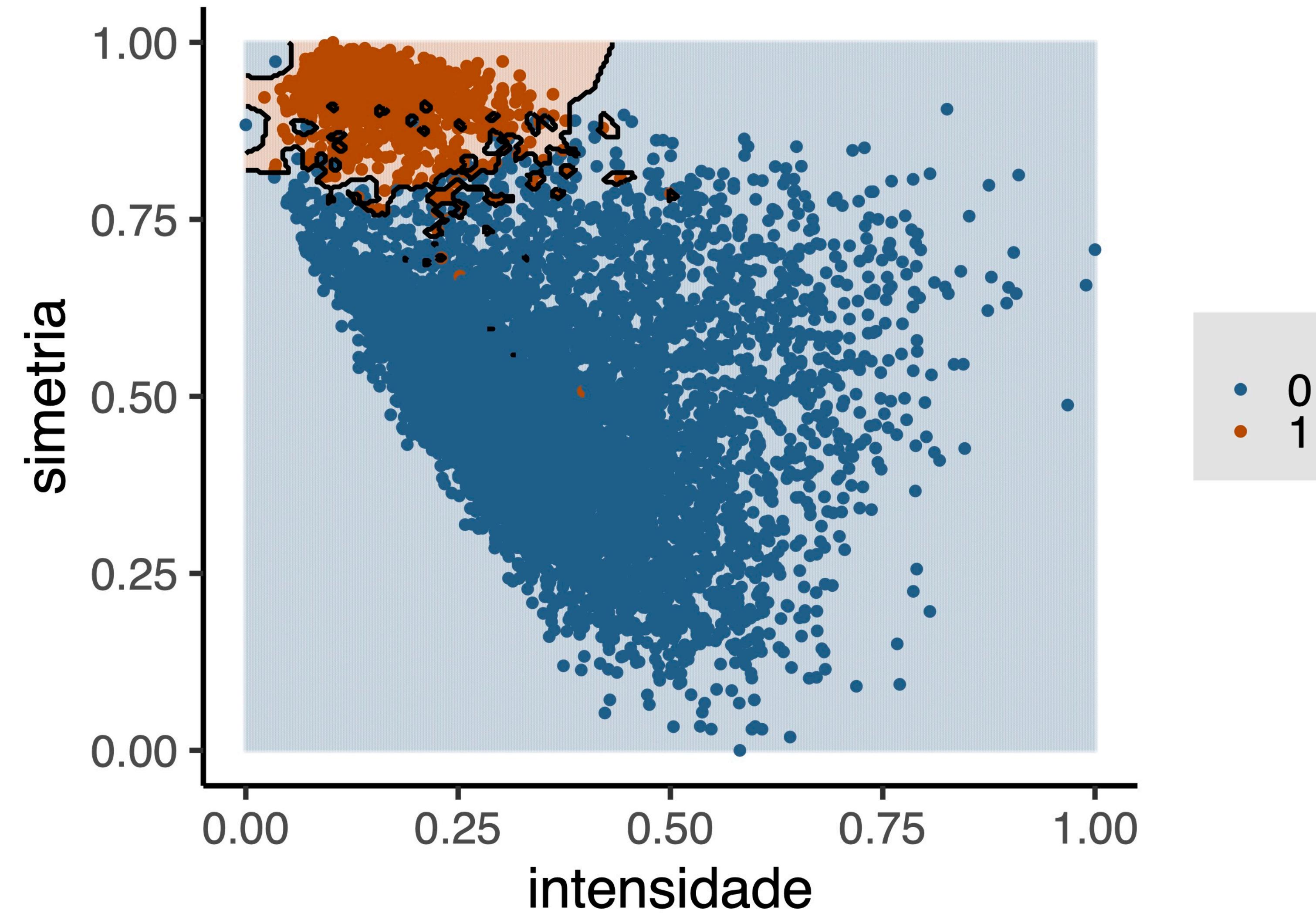
$$\text{Precisão} = \frac{1733 + 235}{2007} \times 100 = 98,1\%$$

K-vizinhos mais próximos

- * Este é um exemplo de um método não paramétrico (não assume nenhuma hipótese sobre a forma específica da distribuição condicional $p(x | c_k)$)
- * Para classificar um novo ponto x , primeiro encontramos os K -vizinhos mais próximos de x e atribuímos a x a classe mais frequente entre estes vizinhos

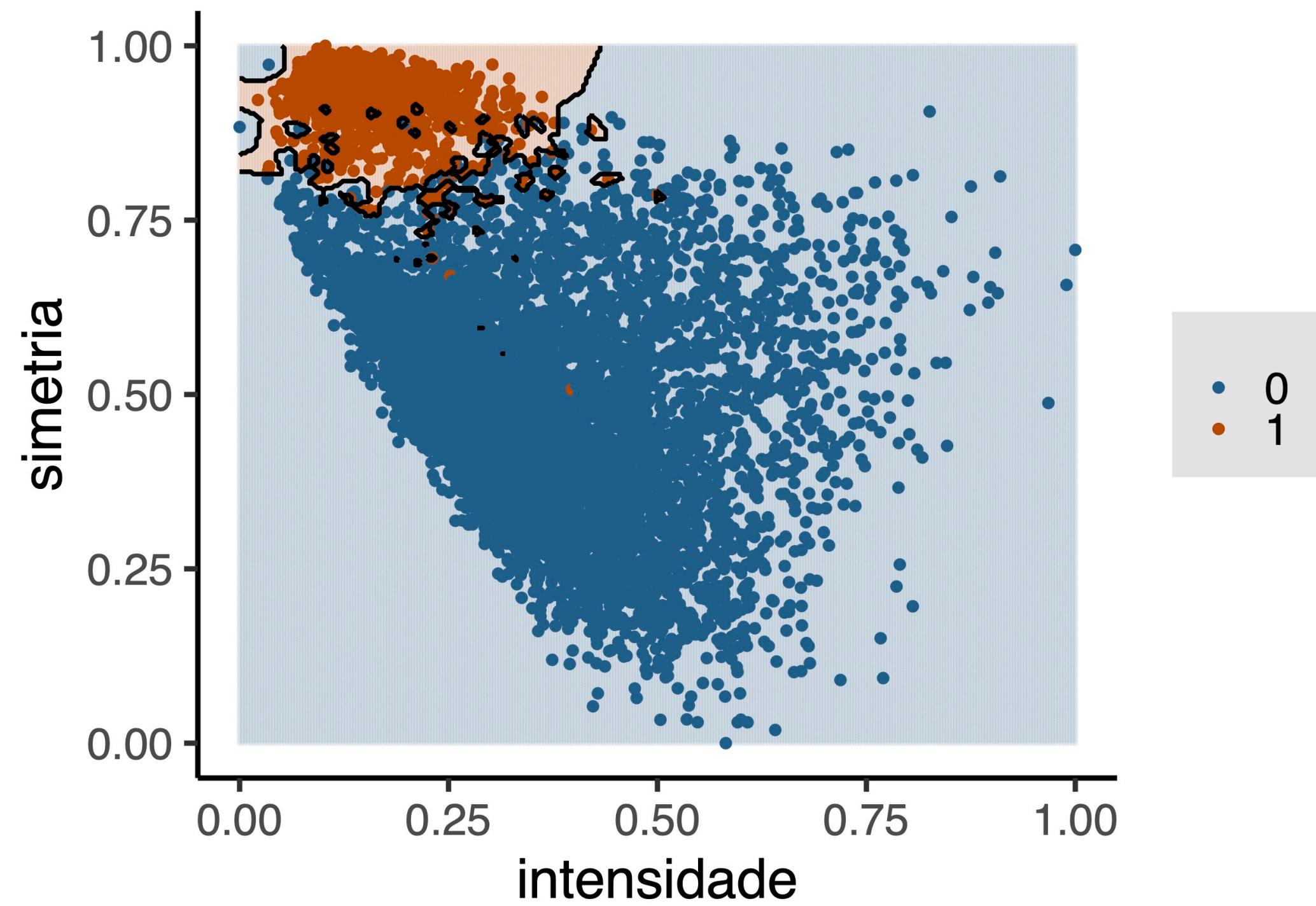


1 vizinho mais próximo



1 vizinho mais próximo

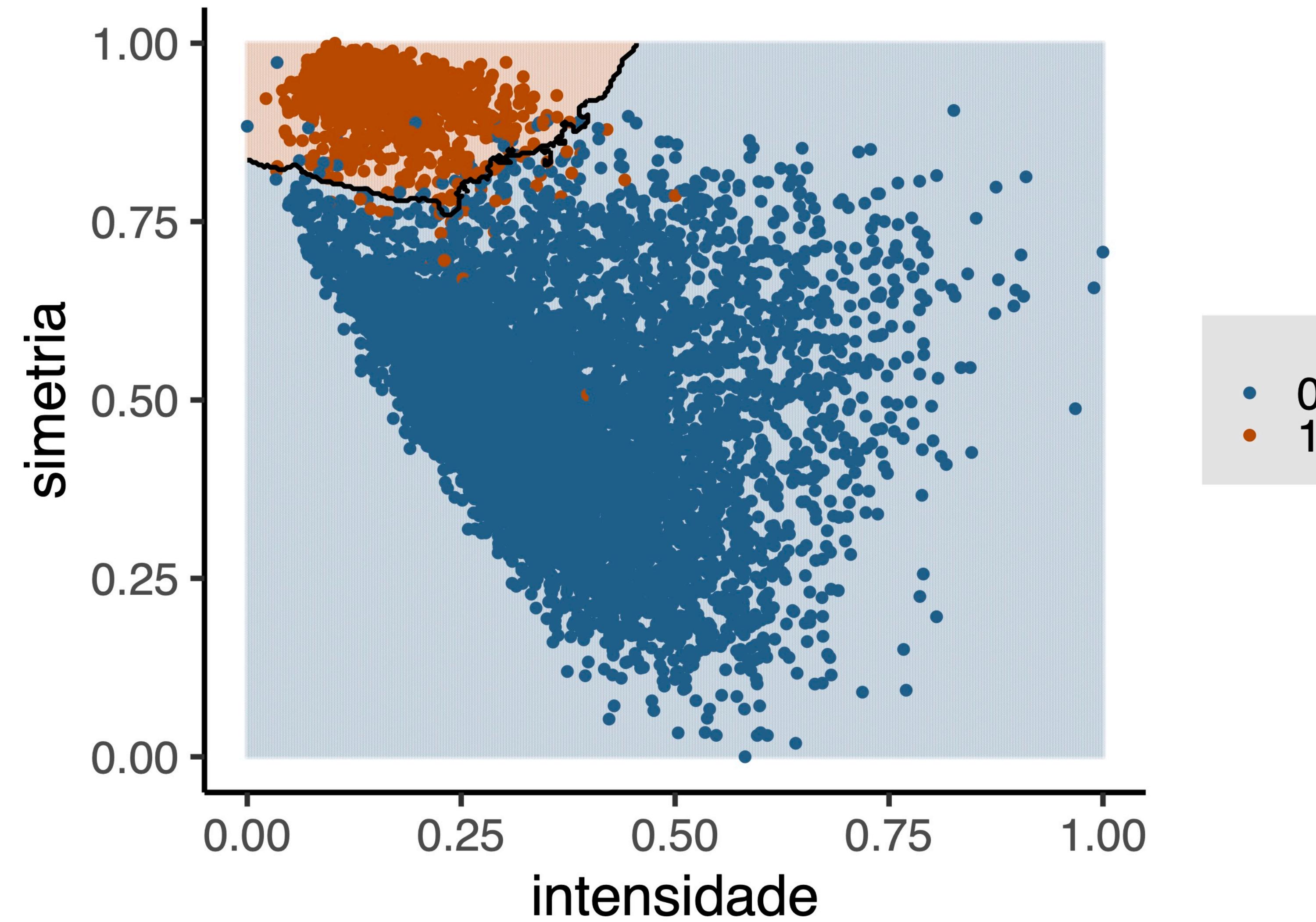
Desempenho na amostra de teste \mathcal{D}_{Te}



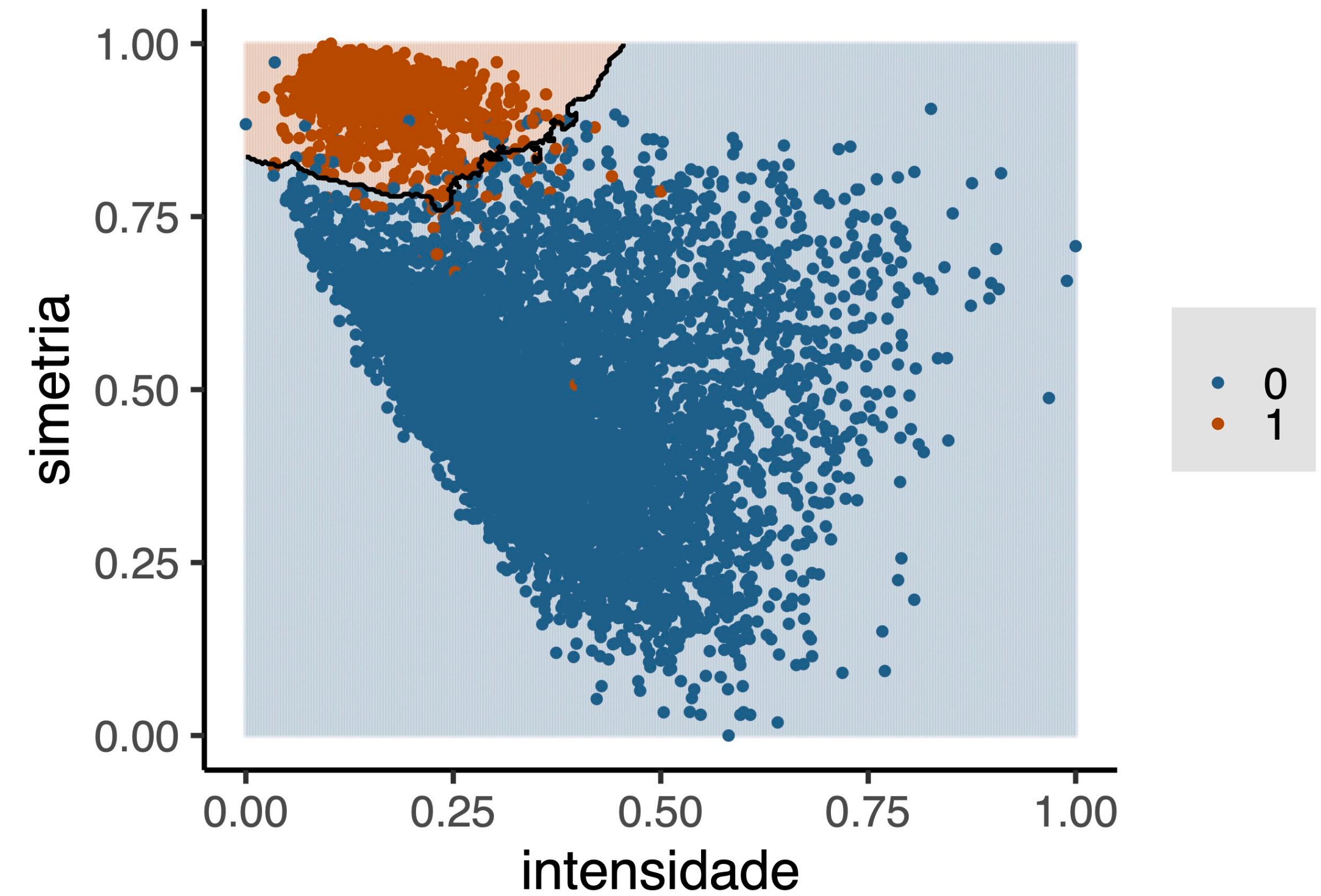
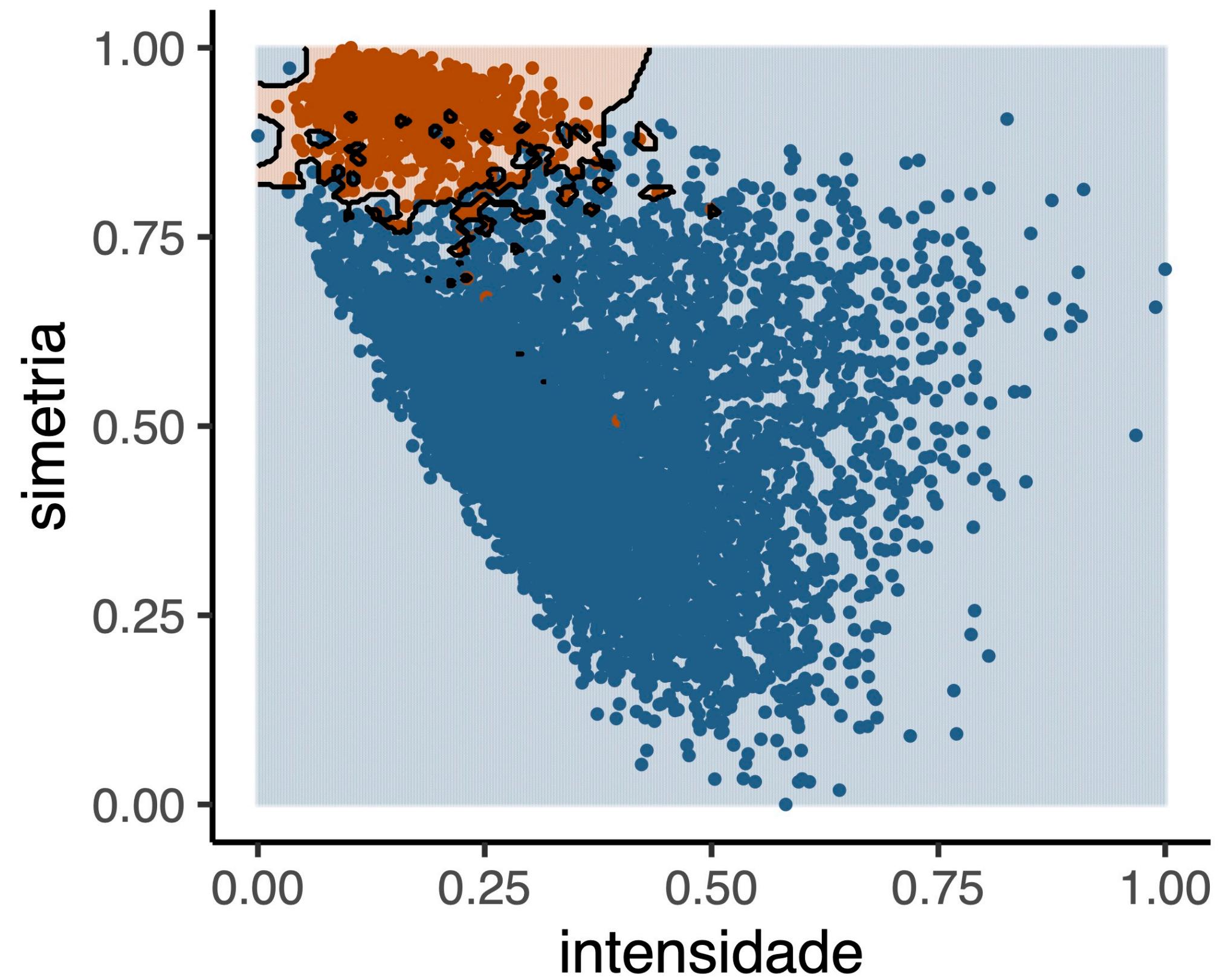
		Classe verdadeira	
		0	1
Classe predita	0	1725	31
	1	18	233

$$\text{Precisão} = \frac{1725 + 233}{2007} \times 100 = 97,6\%$$

15 vizinhos mais próximos

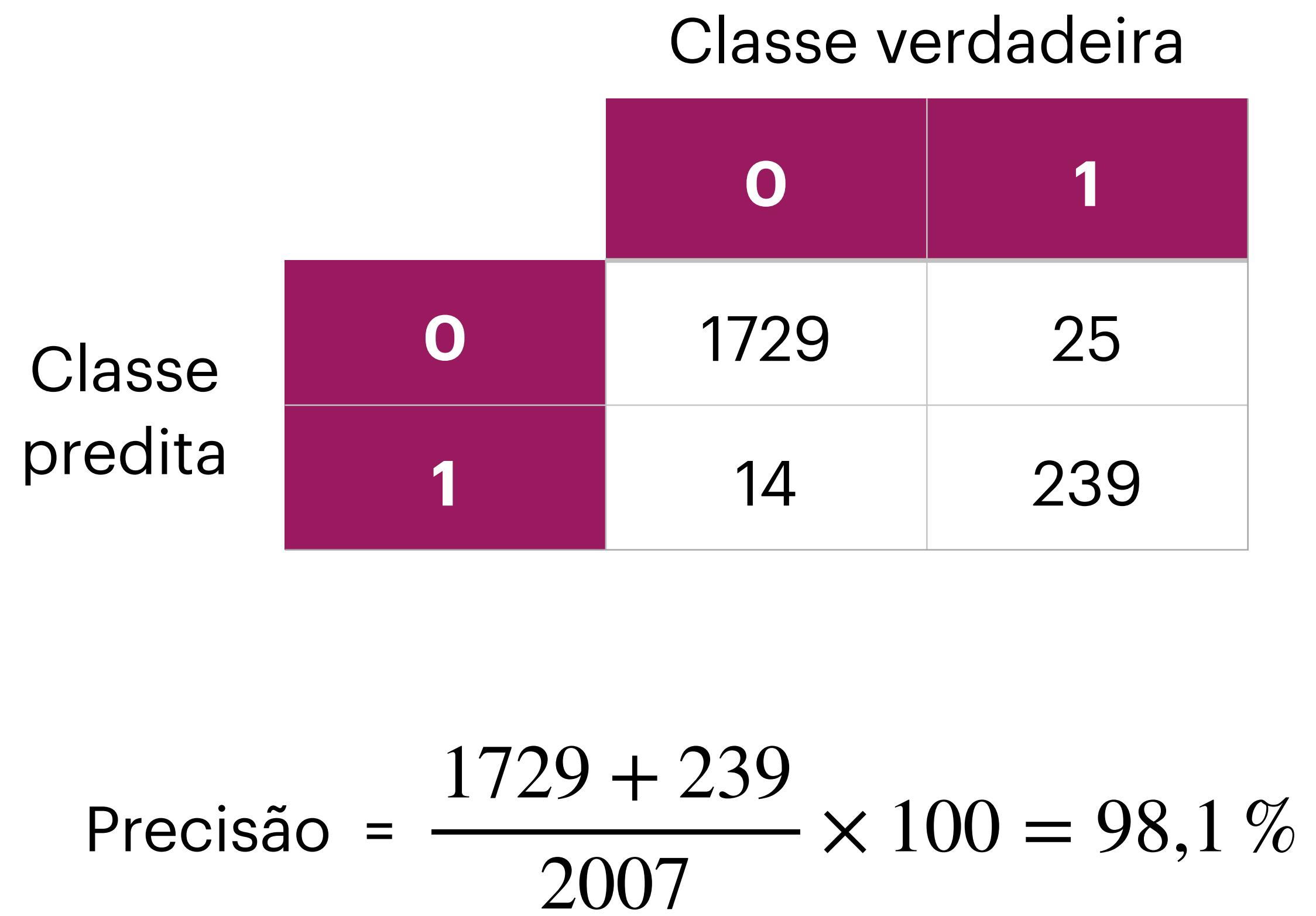
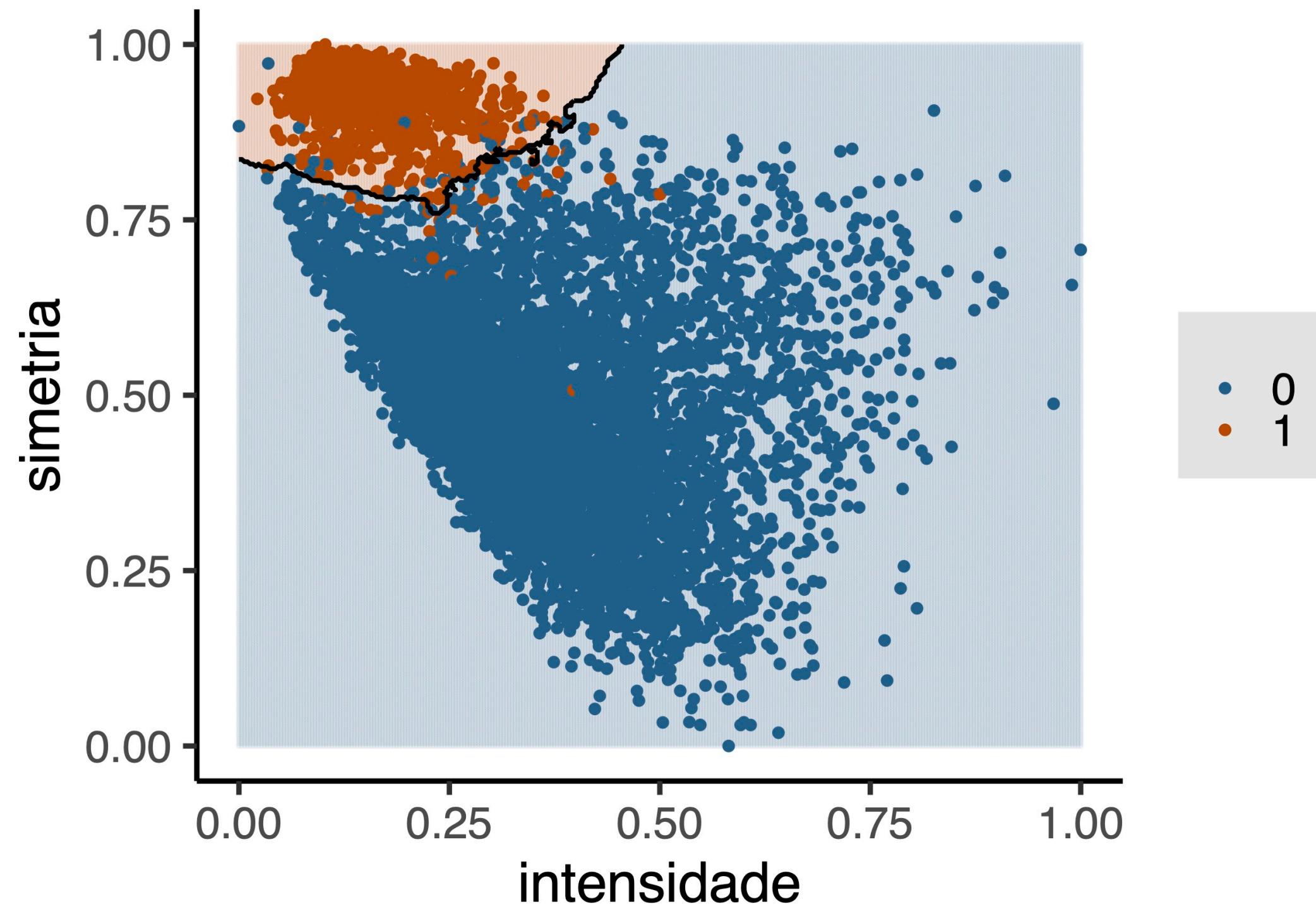


1 vs 15 vizinhos mais próximos



15 vizinhos mais próximos

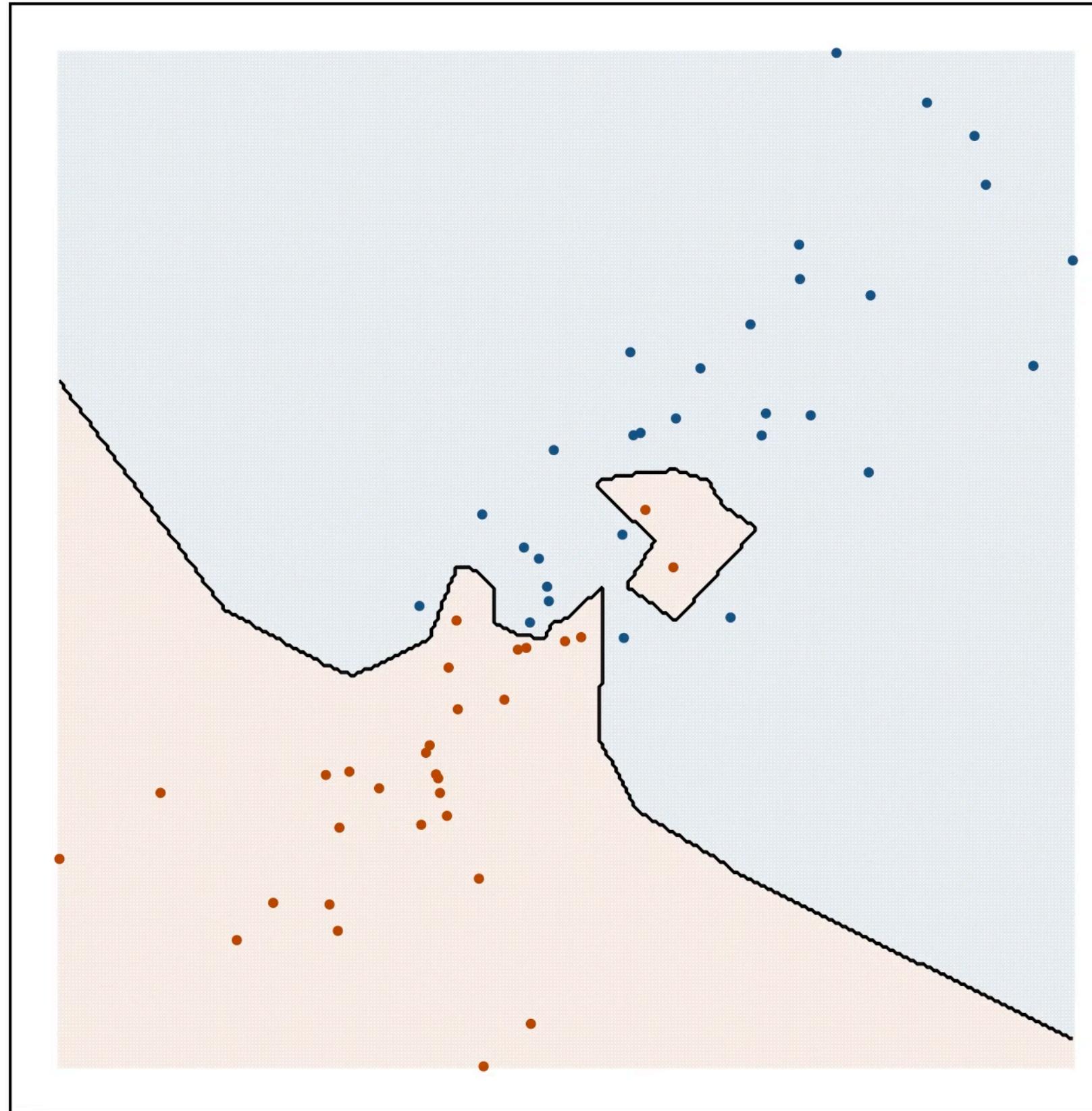
Desempenho na amostra de teste \mathcal{D}_{Te}



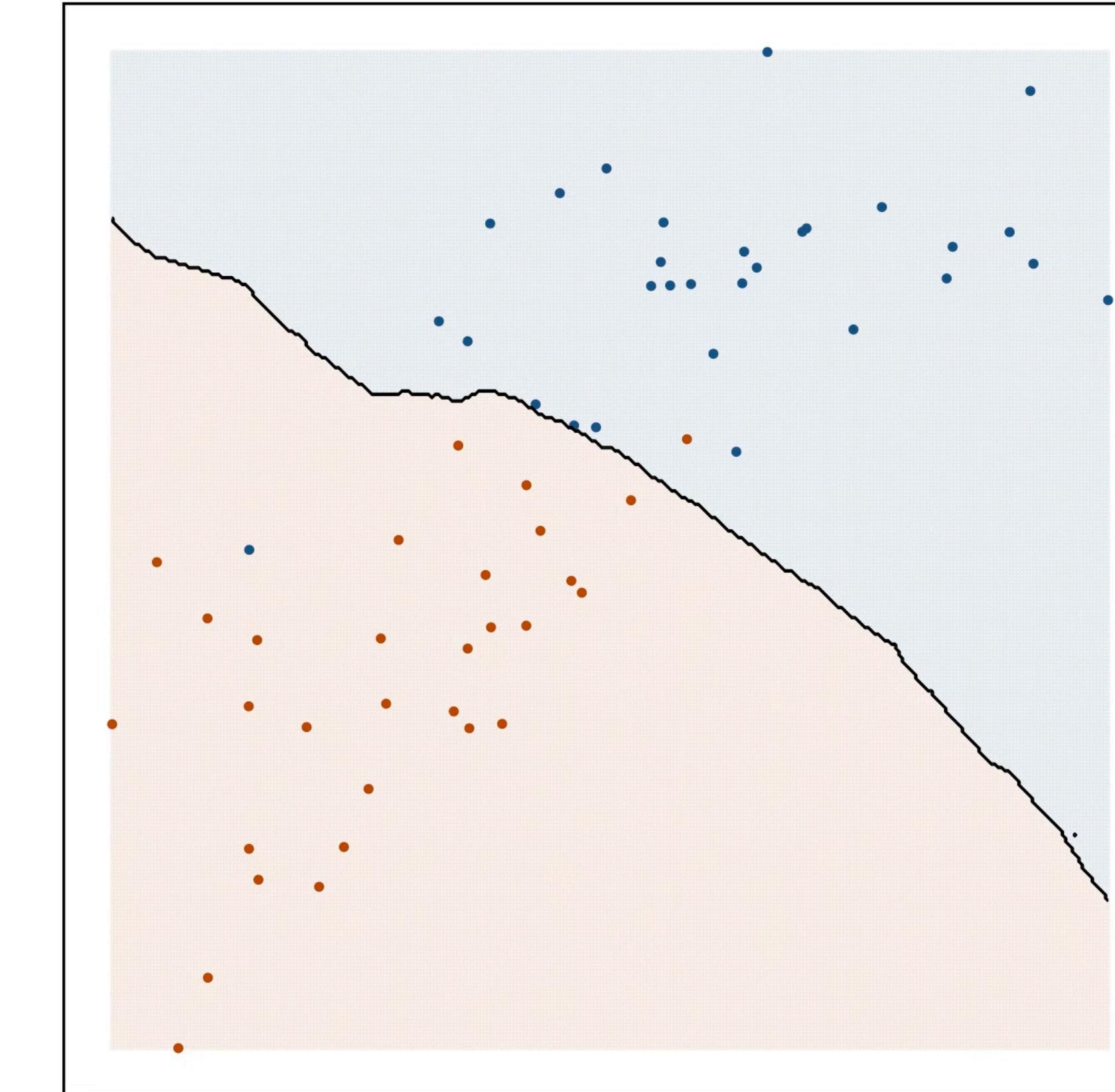
Decomposição do erro

- * Como em classificação em geral não utilizamos a função de custo quadrática, o erro de predição não se compõe exatamente como uma soma do viés ao quadrado e a variância
- * Mas o erro é afetado por termos similares ao viés e à variância e pode-se fazer uma analogia com o caso de regressão
- * Métodos mais flexíveis possuem um “viés” menor e uma “variância” maior, enquanto que métodos menos flexíveis possuem características contrárias
- * A escolha do modelo deve levar em consideração estas características em relação ao tamanho da amostra disponível

Custo benefício de modelos mais/menos flexíveis



1 vizinho mais próximo



15 vizinhos mais próximos