Máquinas de Vetores de Suporte

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais
EEL410250 - Aprendizado de Máquina

EEL / CTC / UFSC

Tópicos

- SVM Linear
- Interpretação Geométrica e Formulação Dual
- Extensão com Kernels

SVM Linear

Regressão Logística

- Para todo classificador linear binário:
  - Vetor de atributos: $\mathbf{x} \in \mathbb{R}^n$
  - Variável de saída (rótulo): $y \in \{0, 1\}$ ou $y_s = 2y - 1 \in \{-1, +1\}$
  - Parâmetros do classificador: $\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$
  - Score de confiança (discriminante): $z = \mathbf{w}^T\mathbf{x} + b$
  - Regra de decisão (predição): $\hat{y} = 1[z > 0]$

- No caso da classificação por regressão logística, utilizamos como função perda a entropia cruzada:

$$L(y, \tilde{y}) = -y \log \tilde{y} - (1 - y) \log(1 - \tilde{y})$$

  onde $\tilde{y} = \sigma(z) = 1/(1 + e^{-z})$

- Como a variável auxiliar $\tilde{y}$ não é usada para predição, é mais conveniente reescrever $L(y, \tilde{y})$ em função de z (e y_s)

Regressão Logística

- Reescrevendo, temos

$$
\begin{aligned}
L(y_s, z) &= -y \log \tilde{y} - (1-y) \log(1-\tilde{y}) \\
&= y \log\left(\frac{1}{\sigma(z)}\right) + (1-y) \log\left(\frac{1}{1-\sigma(z)}\right) \\
&= y \log(1+e^{-z}) + (1-y) \log(1+e^{z}) \\
&= \begin{cases} \log(1+e^{-z}), & y = 1 \\ \log(1+e^{z}), & y = 0 \end{cases} \\
&= \log(1+e^{-y_s z})
\end{aligned}
$$

  a qual é conhecida como perda logística (*logistic loss*)

- Note que $L(-1, z) = L(1, -z)$

Classificação Linear Binária

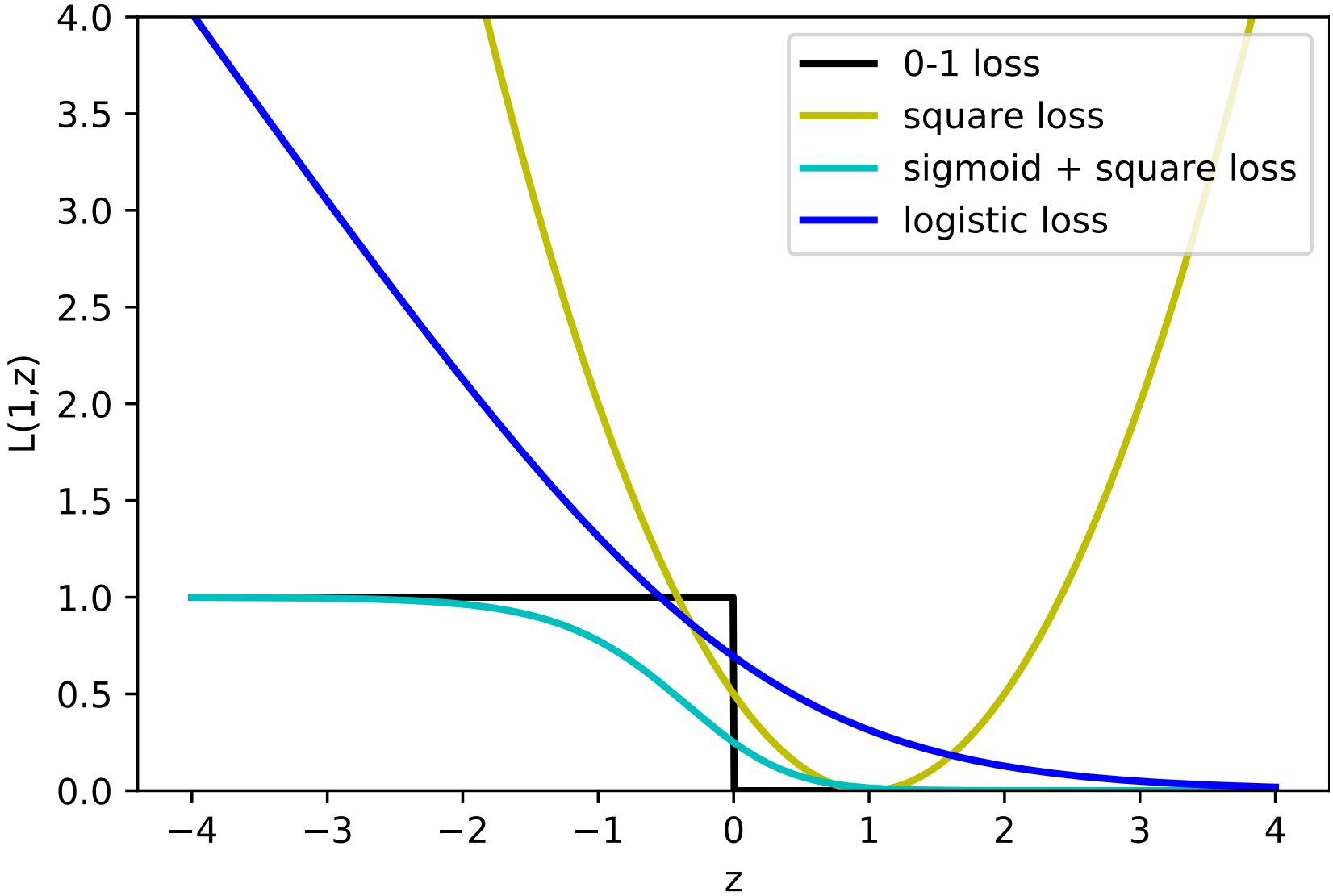
- A função custo do treinamento (com regularização ℓ_2) é dada por

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L(y_s^{(i)}, z^{(i)}) + \frac{\lambda}{2m} \|\mathbf{w}\|^2$$

- A diferença entre os classificadores está na função perda $L(y_s, z)$:
  - Perda 0-1:
$$L(y_s, z) = 1[y_s z < 0]$$
  - Perda quadrática:
$$L(y_s, z) = \tfrac{1}{2}(y_s - z)^2$$
  - Perda logística:
$$L(y_s, z) = \log(1 + e^{-y_s z})$$

- Note que, como $z = \mathbf{w}^T \mathbf{x} + b$ é linear, $J(\mathbf{w}, b)$ é convexa se e somente se $L(1, z)$ é convexa

Funções Perda



Máquina de Vetores de Suporte

- Um classificador SVM (*Support Vector Machine*) corresponde a utilizar a função perda conhecida como *hinge loss*

$$L(y_s, z) = \max\{0,\, 1 - y_s z\}$$

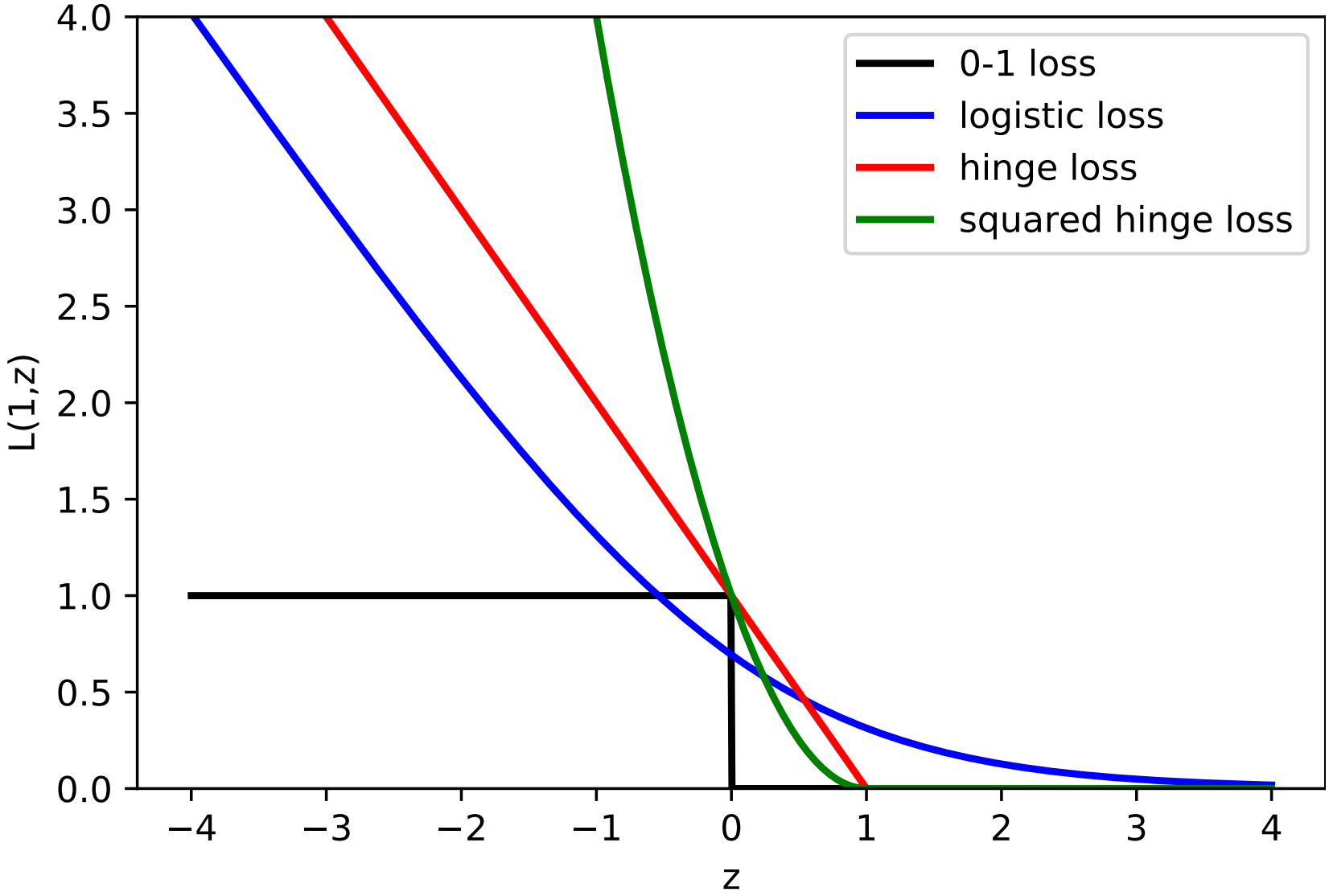
  em conjunto com regularização L2

- Em alguns casos também se utiliza a perda *hinge* quadrática:

$$L(y_s, z) = \max\{0,\, 1 - y_s z\}^2$$

  - Diferenciável em todos os pontos

Funções Perda



SVM

- Função custo:

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \max\{0,\, 1 - y_s^{(i)} z^{(i)}\} + \frac{\lambda}{2m} \|\mathbf{w}\|^2$$

  onde $z^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + b$

- Por convenção, multiplica-se por mC, onde $C = 1/\lambda$:

$$J(\mathbf{w}, b) = C \sum_{i=1}^{m} \max\{0,\, 1 - y_s^{(i)} z^{(i)}\} + \frac{1}{2} \|\mathbf{w}\|^2$$

  - C é um parâmetro de regularização que expressa a preferência por uma classificação correta

- Conhecido como SVM linear com problema de otimização primal

SVM

- Na literatura, o SVM é mais conhecido como um classificador de máxima margem com margem suave (*soft margin*)
- Esta interpretação geométrica dá origem a um problema de otimização equivalente: otimização quadrática com restrições
- Admite formulação dual, que por sua vez permite:
  - Métodos computacionalmente mais eficientes em certas situações
  - Extensão com kernels

Interpretação Geométrica e Formulação Dual

Interpretação Geométrica

- Considere um classificador linear: $z = \mathbf{w}^T\mathbf{x} + b$
- Considere o hiperplano de separação:

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\} = \left\{\mathbf{x} : \mathbf{w}^T\left(\mathbf{x} + \frac{b}{\|\mathbf{w}\|}\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) = 0\right\}$$

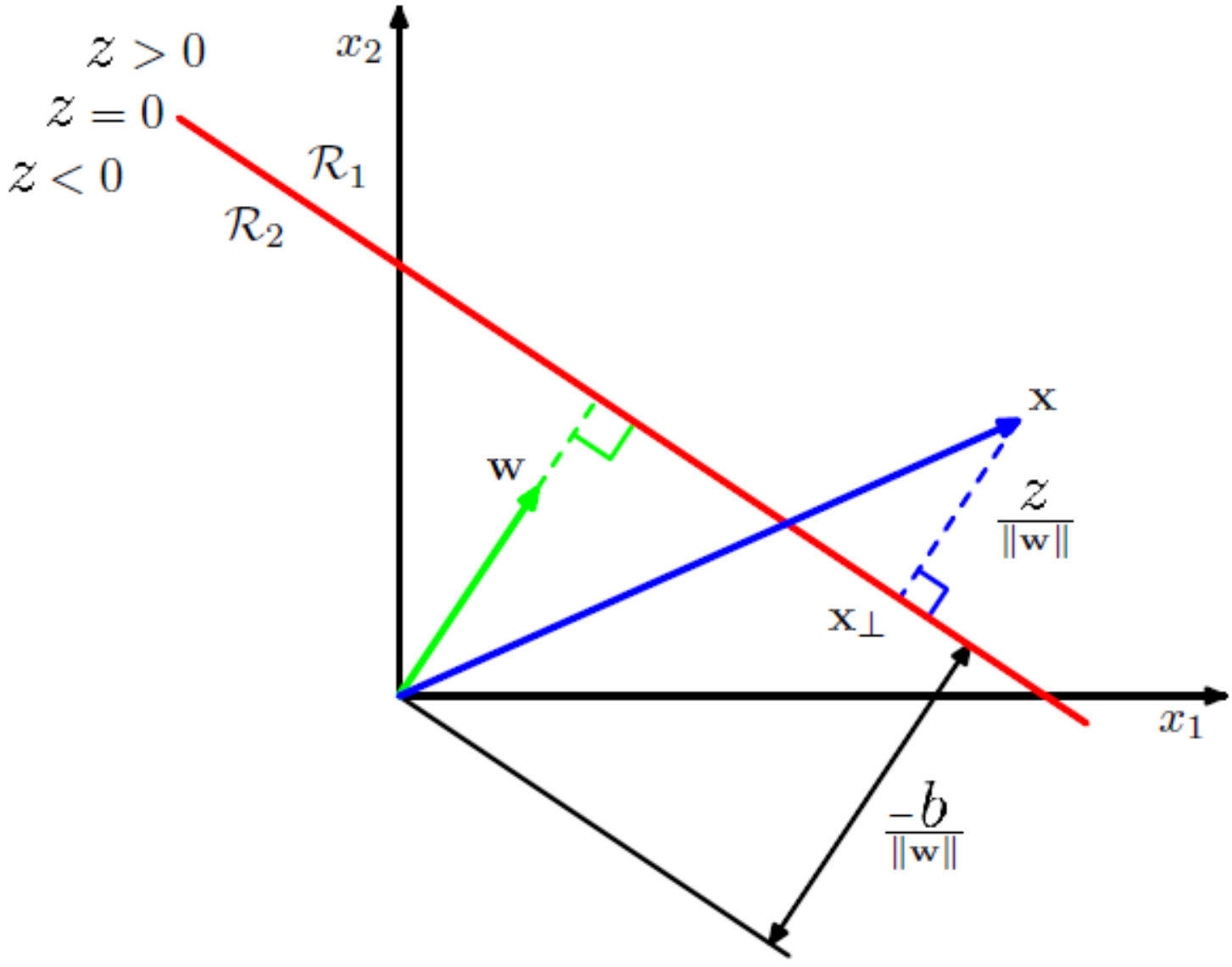
- A distância (com sinal) entre $\mathbf{x}$ e $\mathcal{H}$ é dada por

$$d(\mathbf{x}, \mathcal{H}) = \frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|} + \frac{b}{\|\mathbf{w}\|} = \frac{z}{\|\mathbf{w}\|}$$

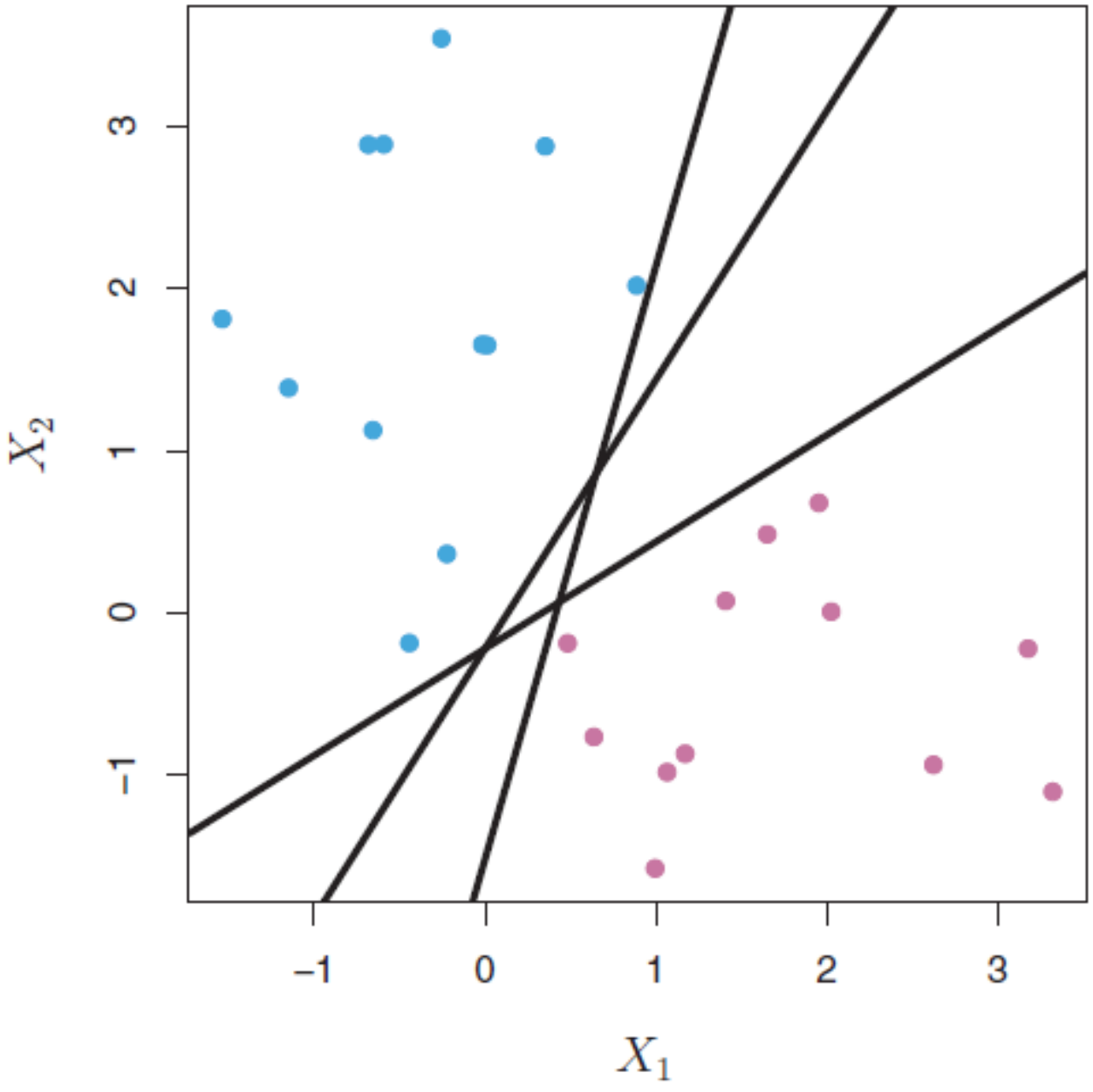
- Para um conjunto de treinamento $\{(\mathbf{x}^{(i)}, y_s^{(i)})\}$, a margem do classificador é definida como

$$M = \min_{i=1,\ldots,m} \frac{y_s^{(i)} z^{(i)}}{\|\mathbf{w}\|}$$

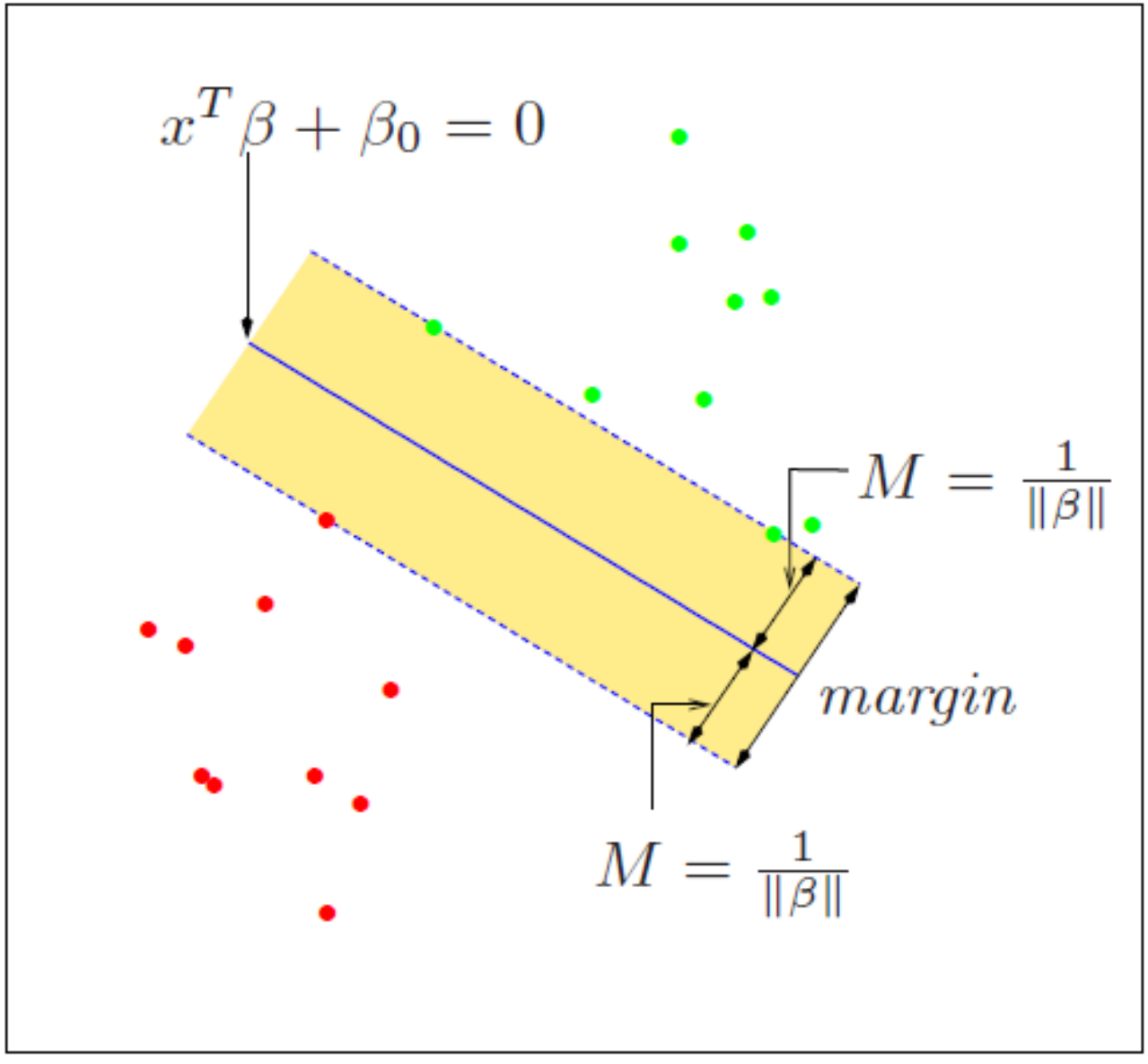
Exemplo


$z > 0$
$z = 0$
$z < 0$
x_2
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathbf{x}$
$\mathbf{w}$
$\frac{z}{\|\mathbf{w}\|}$
$\mathbf{x}_\perp$
x_1
$\frac{-b}{\|\mathbf{w}\|}$

Exemplo



Exemplo



Classificador de Máxima Margem

- Para um conjunto de treinamento linearmente separável, faz sentido escolher o classificador que maximiza a margem:

$$\max_{\mathbf{w},b} \quad \min_{i} \quad \frac{y_s^{(i)} z^{(i)}}{\|\mathbf{w}\|}$$

ou, equivalentemente,

$$\begin{aligned} \max_{\mathbf{w},b,M} \quad & M \\ \text{s.t.} \quad & \frac{y_s^{(i)} z^{(i)}}{\|\mathbf{w}\|} \geq M, \ \forall i \end{aligned}$$

- Note que multiplicar $(\mathbf{w}, b)$ por um escalar a não altera a solução:

$$\frac{z^{(i)}}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T \mathbf{x}^{(i)} + b}{\|\mathbf{w}\|} = \frac{a\mathbf{w}^T \mathbf{x}^{(i)} + ab}{\|a\mathbf{w}\|}$$

- Isso implica que podemos estipular arbitrariamente o valor de $\|\mathbf{w}\|$

Classificador de Máxima Margem

- Escolhendo $\|\mathbf{w}\| = 1/M$, obtemos

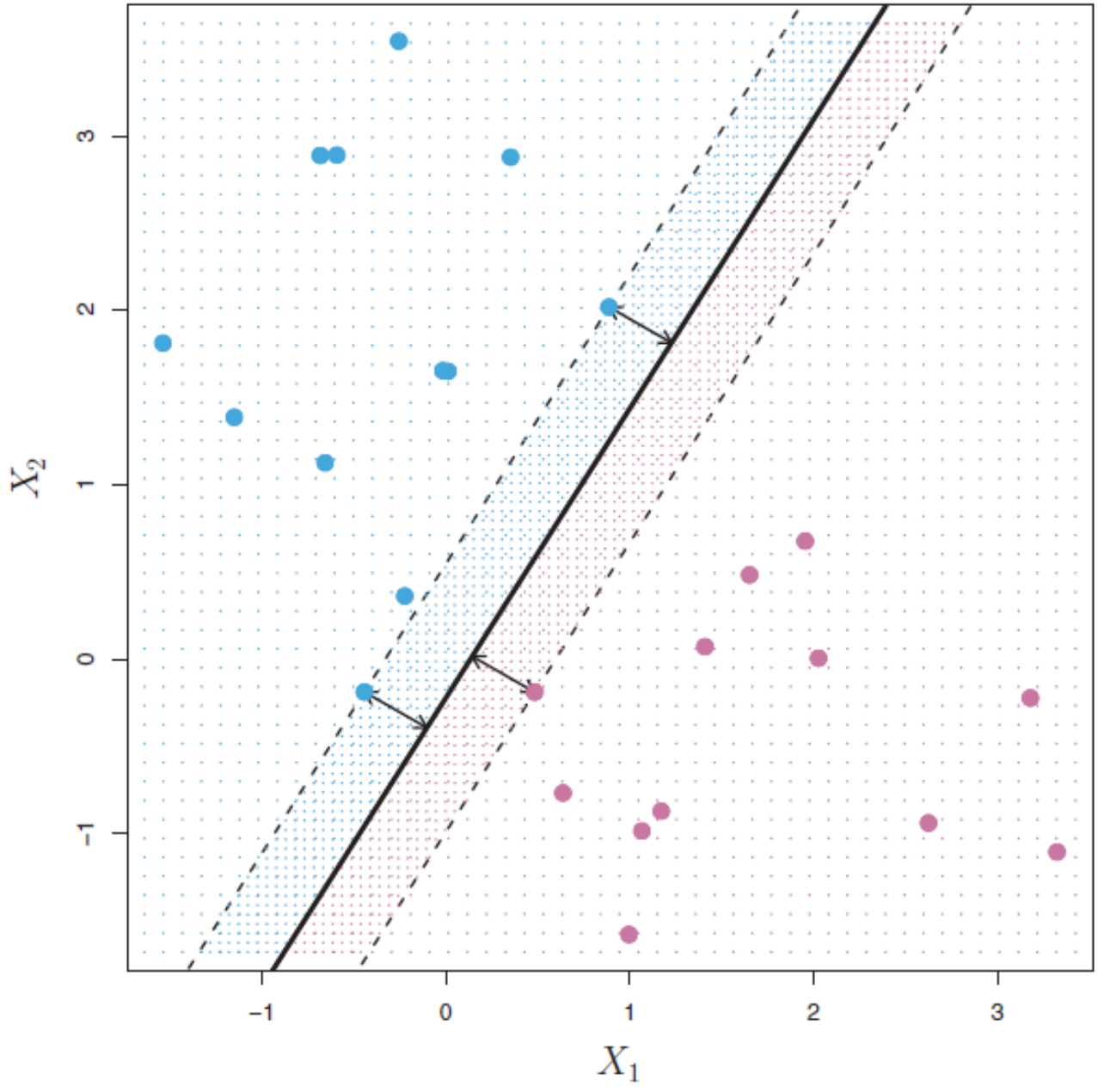
$$\begin{aligned} \max_{\mathbf{w},b} \quad & 1/\|\mathbf{w}\| \\ \text{s.t.} \quad & y_s^{(i)} z^{(i)} \geq 1, \ \forall i \end{aligned}$$

  ou, equivalentemente,

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_s^{(i)} z^{(i)} \geq 1, \ \forall i \end{aligned}$$

- Vetores $\mathbf{x}^{(i)}$ situados em cima da margem ($y_s^{(i)} z^{(i)} = 1$) são chamados de vetores de suporte

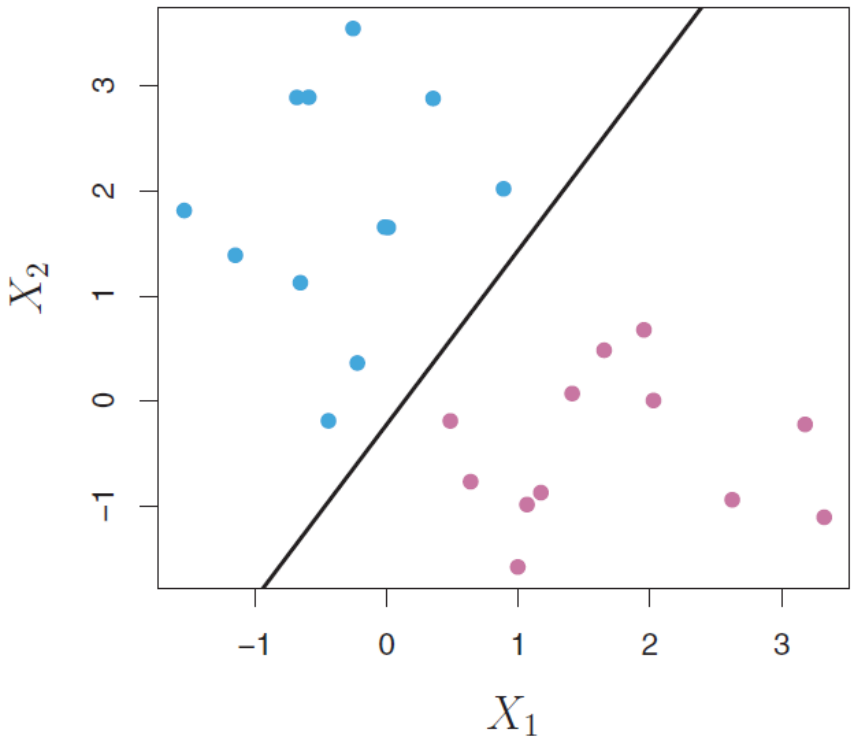
Exemplo

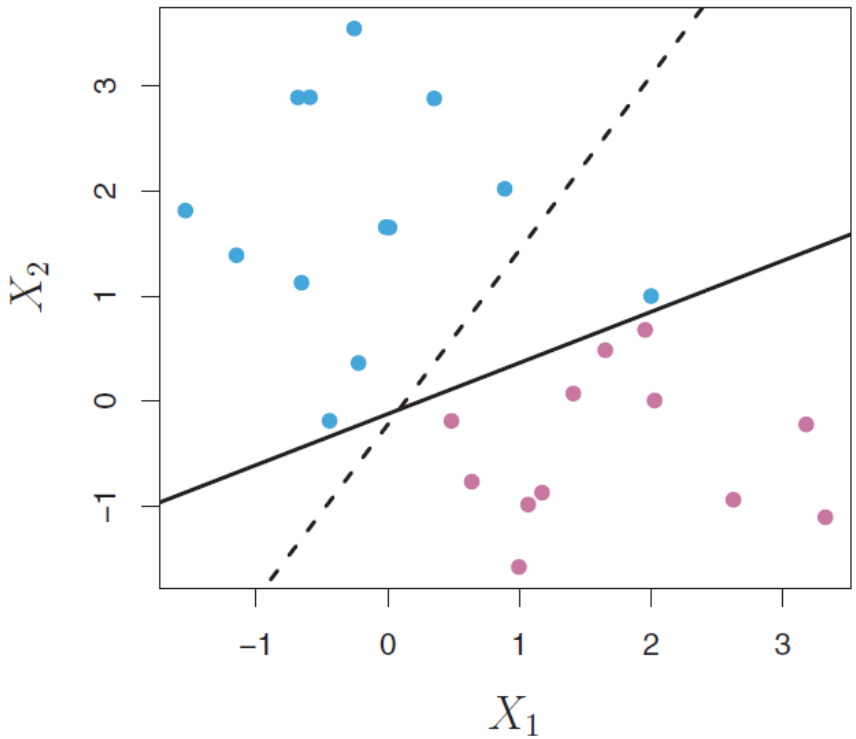


Problemas com o classificador de máxima margem

- Nem sempre o conjunto de treinamento é linearmente separável (geralmente não é)
- Mesmo que fosse, ainda assim o classificador seria muito sensível a amostras próximas da margem
  - Sugere a ocorrência de overfitting

Exemplo





Margem suave *(soft margin)*

- Podemos suavizar a restrição de margem fazendo uso de variáveis de folga (*slack variables*) $\zeta_1, \ldots, \zeta_m$:

$$\begin{aligned} \min_{\mathbf{w}, b, \zeta_1, \ldots, \zeta_m} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_s^{(i)} z^{(i)} \geq 1 \qquad , \ \forall i \end{aligned}$$

Margem suave *(soft margin)*

- Podemos suavizar a restrição de margem fazendo uso de variáveis de folga (*slack variables*) $\zeta_1, \ldots, \zeta_m$:

$$\begin{aligned} \min_{\mathbf{w}, b, \zeta_1, \ldots, \zeta_m} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \zeta_i \\ \text{s.t.} \quad & y_s^{(i)} z^{(i)} \geq 1 - \zeta_i, \quad \zeta_i \geq 0, \ \forall i \end{aligned}$$

  - Cada $\zeta_i = 0$ representa uma margem satisfeita ($y_s^{(i)} z^{(i)} \geq 1$)
  - Cada $\zeta_i > 0$ representa uma violação de margem ($y_s^{(i)} z^{(i)} < 1$)
  - Cada $\zeta_i > 1$ representa uma classificação errada ($y_s^{(i)} z^{(i)} < 0$)
  - C representa o peso dado às violações de margem no custo total

- $\uparrow C \implies \downarrow$ violações de margem $\implies \downarrow$ classificações erradas
- $\downarrow C \implies \downarrow \|\mathbf{w}\|^2 \implies \uparrow$ margem

Margem suave *(soft margin)*

- Obs: As restrições

$$\zeta_i \geq 0, \qquad y_s^{(i)} z^{(i)} \geq 1 - \zeta_i, \quad \forall i$$

podem ser reescritas como

$$\zeta_i \geq \max\{0,\, 1 - y_s^{(i)} z^{(i)}\}, \quad \forall i$$

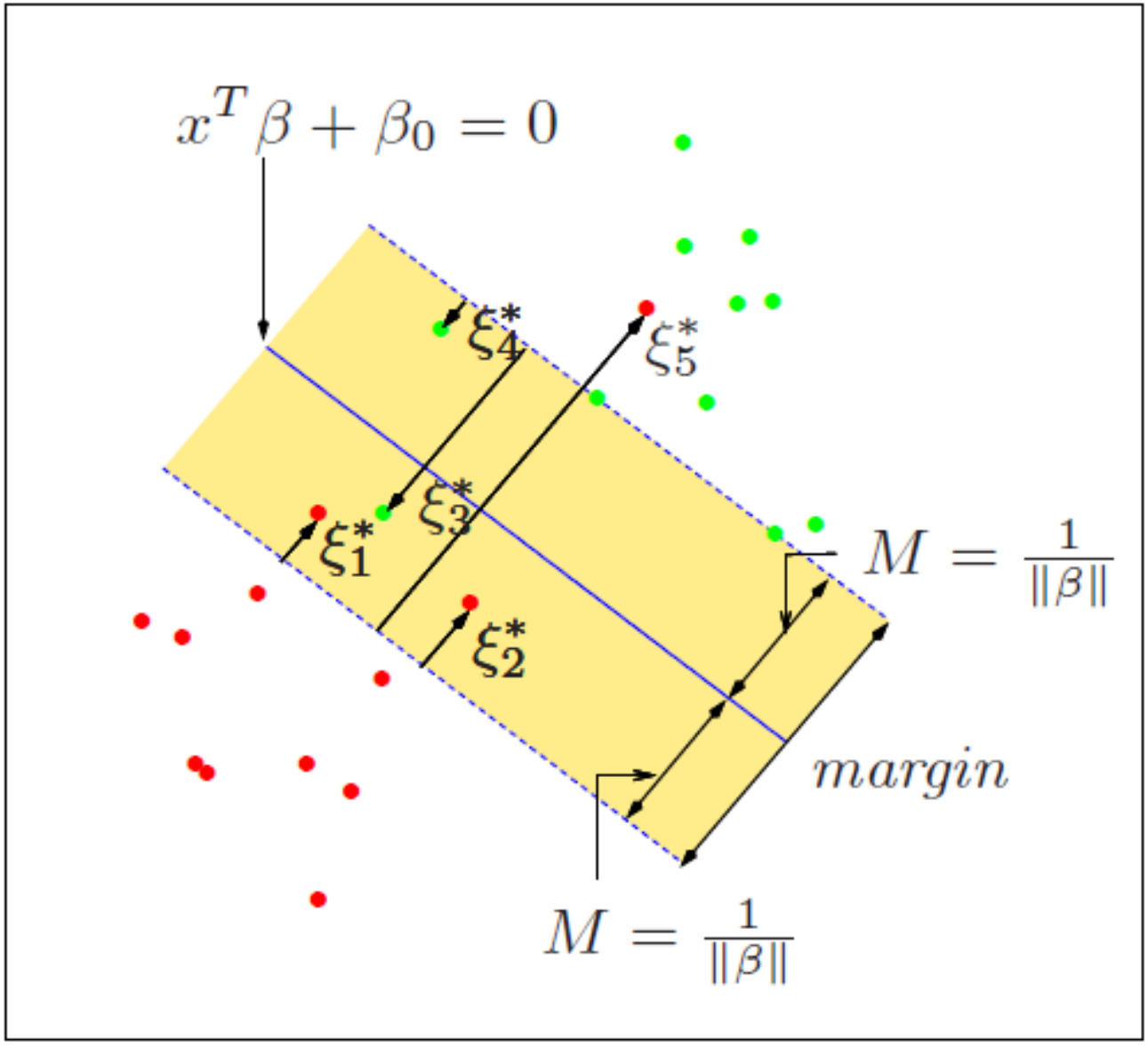
e eliminadas do problema

- Resulta no problema de otimização sem restrições

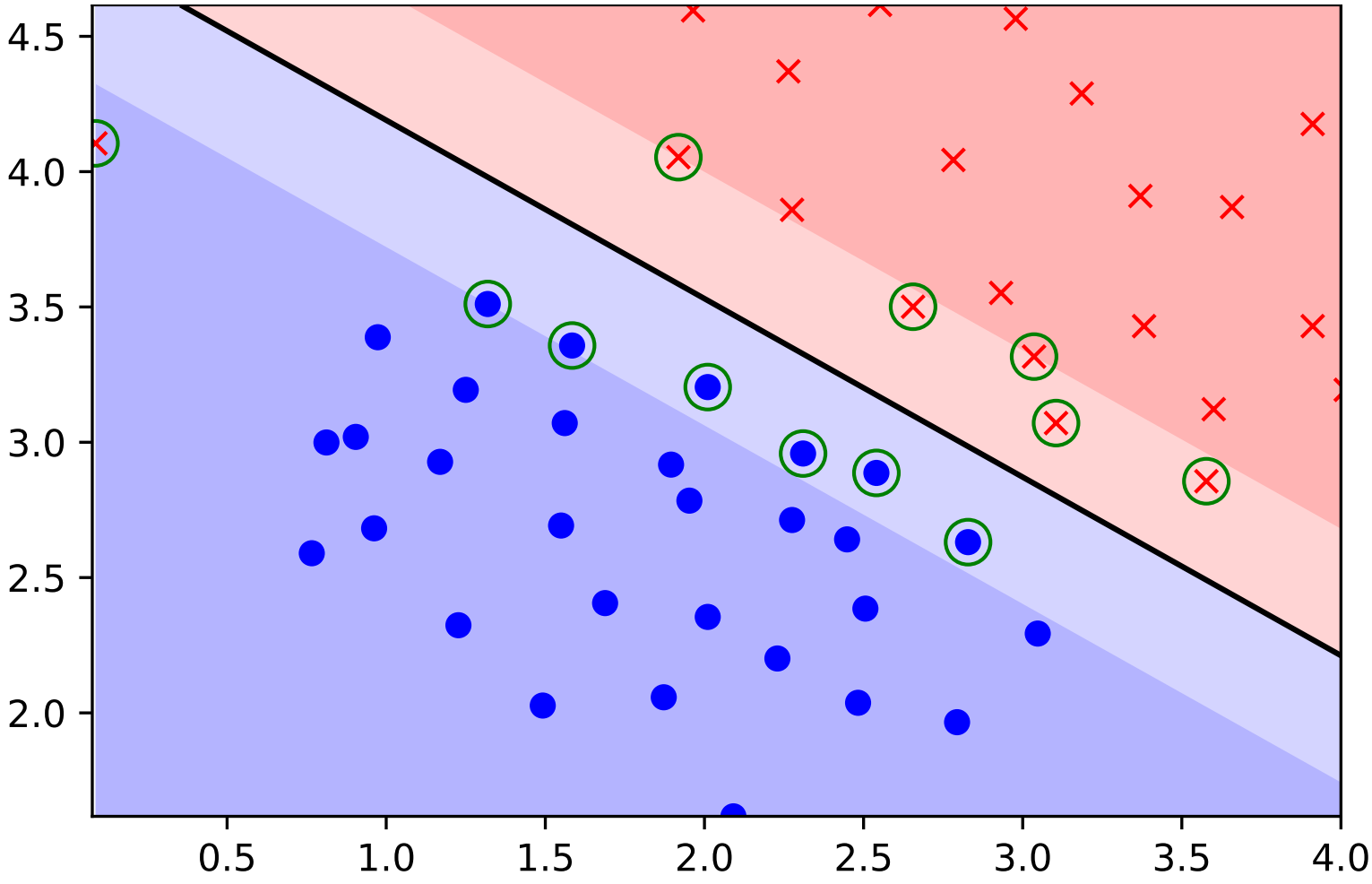
$$\min_{\mathbf{w},b} \; \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \max\{0,\, 1 - y_s^{(i)} z^{(i)}\}$$

correspondente à formulação inicial

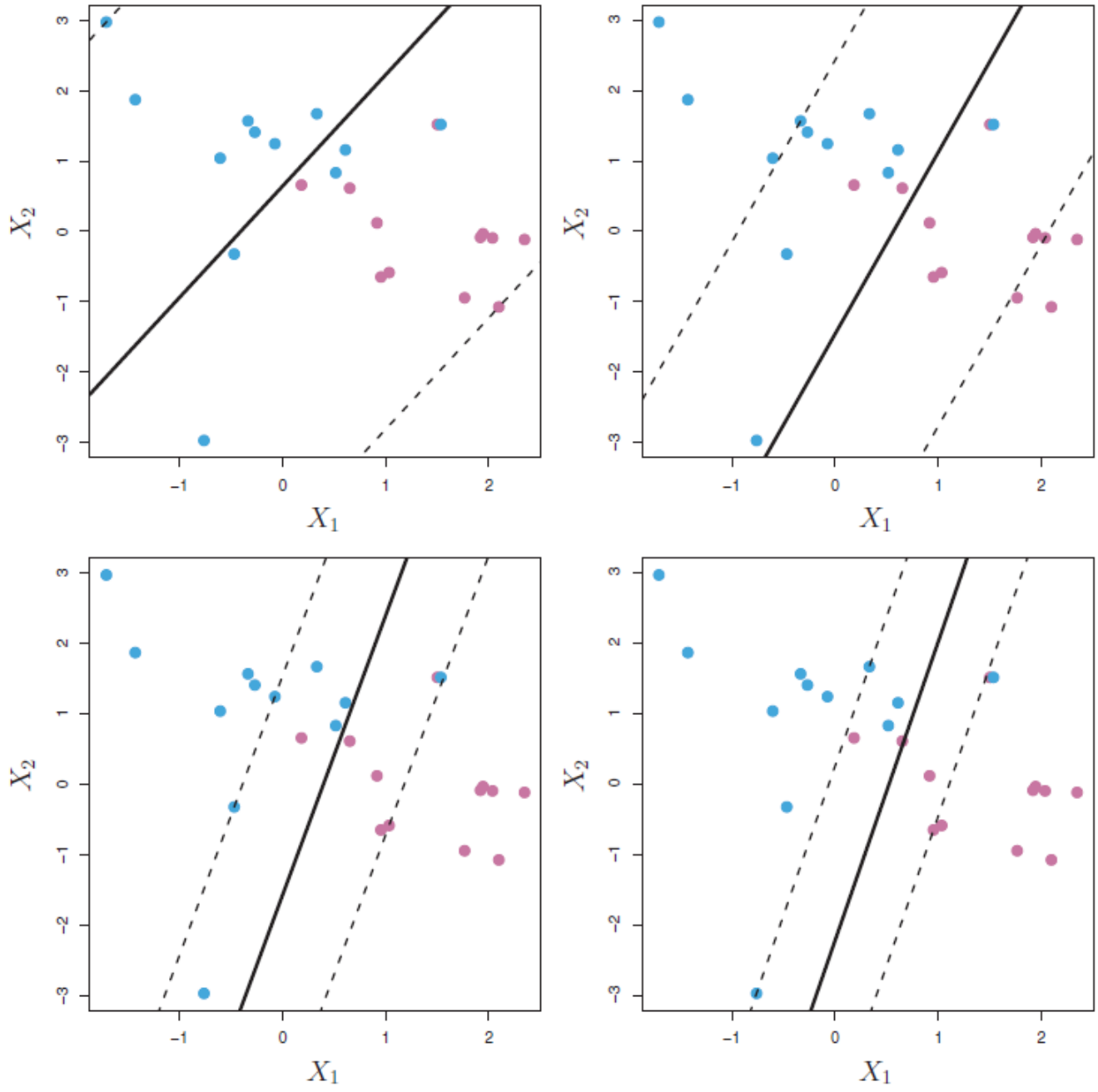
Exemplo


$x^T\beta + \beta_0 = 0$
ξ_4^*
ξ_5^*
ξ_3^*
ξ_1^*
ξ_2^*
$M = \frac{1}{\|\beta\|}$
$margin$
$M = \frac{1}{\|\beta\|}$

Exemplo



Exemplo (aumentando C)



Formulação Dual

- O problema primal

$$\begin{aligned} \min_{\mathbf{w},b,\zeta_1,\ldots,\zeta_m} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i \\ \text{s.t.} \quad & y_s^{(i)} z^{(i)} \geq 1 - \zeta_i, \quad \zeta_i \geq 0, \ \forall i \end{aligned} \tag{1}$$

(onde $z^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + b$) possui $n + 1 + m$ variáveis e $2m$ restrições

- Usando a teoria de otimização convexa (dualidade lagrangiana), é possível converter (1) em um problema dual mais simples

$$\begin{aligned} \min_{\alpha_1,\ldots,\alpha_m} \quad & \tfrac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} + \mathbf{1}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \ \forall i, \quad \sum_i \alpha_i y_s^{(i)} = 0 \end{aligned} \tag{2}$$

onde $Q_{ij} = y_s^{(i)} y_s^{(j)} \mathbf{x}^{(i)T}\mathbf{x}^{(j)}$

  - Apenas m variáveis e $2m + 1$ restrições

Formulação Dual

- Uma propriedade importante da solução ótima é que $\alpha_i \neq 0$ se e somente se $\mathbf{x}^{(i)}$ é um vetor de suporte $\implies$ solução esparsa

- Uma vez resolvido (2), a solução ótima de (1) é dada por

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_s^{(i)} \mathbf{x}^{(i)} = \sum_{i \in \mathcal{S}} \alpha_i y_s^{(i)} \mathbf{x}^{(i)}$$

  onde $\mathcal{S}$ denota o conjunto de índices dos vetores de suporte, e

$$b = y_s^{(i^*)} - \mathbf{w}^T \mathbf{x}^{(i^*)}$$

  onde $i^* \in \mathcal{S}$ denota qualquer índice tal que $0 < \alpha_{i^*} < C$.

- Treinamento via formulação dual é mais eficiente do que primal se e somente se $m \leq n$

Observação

- Se desejássemos, poderíamos realizar a predição diretamente no domínio dual, sem calcular explicitamente o vetor $\mathbf{w}$
- Predição (score de confiança):

$$z = b + \mathbf{w}^T \mathbf{x} = b + \sum_{i \in \mathcal{S}} \alpha_i y_s^{(i)} {\mathbf{x}^{(i)}}^T \mathbf{x}$$

- Nesse caso, após o treinamento, guardaríamos os parâmetros b, $\{\alpha_i y_s^{(i)}, i \in \mathcal{S}\}$ e o conjunto de vetores de suporte $\{\mathbf{x}^{(i)}, i \in \mathcal{S}\}$
- Porém, essa abordagem não traz qualquer vantagem (para SVM linear), uma vez que a complexidade da predição se torna $O(|\mathcal{S}|n)$

Extensão com Kernels

Extensão com Funções Não-Lineares

- Assim como outros modelos, um classificador SVM pode ser estendido com funções não-lineares $\varphi_j(\mathbf{x})$, $j = 1, \ldots, n'$, resultando em um vetor de atributos

$$\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \ldots, \varphi_{n'}(\mathbf{x}))^T$$

- Treinamento: idêntico ao problema (2), exceto que a matriz $\mathbf{Q}$ é dada por

$$Q_{ij} = y_s^{(i)} y_s^{(j)} \boldsymbol{\varphi}(\mathbf{x}^{(i)})^T \boldsymbol{\varphi}(\mathbf{x}^{(j)})$$

- Predição (score de confiança):

$$z = b + \sum_{i \in \mathcal{S}} \alpha_i y_s^{(i)} \boldsymbol{\varphi}(\mathbf{x}^{(i)})^T \boldsymbol{\varphi}(\mathbf{x})$$

Kernel Trick

- Observe que nenhuma etapa depende diretamente de $\boldsymbol{\varphi}(\mathbf{x})$, mas apenas do produto interno

$$K(\mathbf{x}^{(i)}, \mathbf{x}) \triangleq \boldsymbol{\varphi}(\mathbf{x}^{(i)})^T \boldsymbol{\varphi}(\mathbf{x})$$

  chamado de função de kernel

- Por esse motivo, não precisamos sequer implementar o cálculo de $\boldsymbol{\varphi}(\mathbf{x})$ se dispusermos de uma função que calcula $K(\mathbf{x}^{(i)}, \mathbf{x})$

- Em muitos casos, isso nos permite implementar eficientemente extensões para espaços de dimensão $n' \gg m$ (até mesmo de dimensão infinita) definidos implicitamente através de um kernel

- Resulta em um classificador não-paramétrico (i.e., número de parâmetros depende do tamanho do conjunto de treinamento)

Exemplos de Funções de Kernel

- Kernel linear:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Kernel polinomial:

$$K(\mathbf{x}, \mathbf{x}') = (r + \gamma \mathbf{x}^T \mathbf{x}')^d$$

  - Corresponde à adição de todos os monômios de grau $\leq d$ (com coeficientes específicos que dependem de r e γ)
  - Nesse caso, r, γ e d (assim como C) são hiperparâmetros
  - Exemplo ($n = 2 \longrightarrow n' = 6$):

$$\begin{aligned}
(1 + \mathbf{x}^T \mathbf{x}')^2 &= (1 + (x_1, x_2)^T (x_1', x_2'))^2 \\
&= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + 2x_1 x_1' + 2x_2 x_2' + x_1^2 x_1'^2 + 2x_1 x_1' x_2 x_2' + x_2^2 x_2'^2 \\
&= \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')
\end{aligned}$$

onde $\boldsymbol{\varphi}(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$

Exemplos de Funções de Kernel

- Kernel RBF (*Radial Basis Function*) ou gaussiano:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

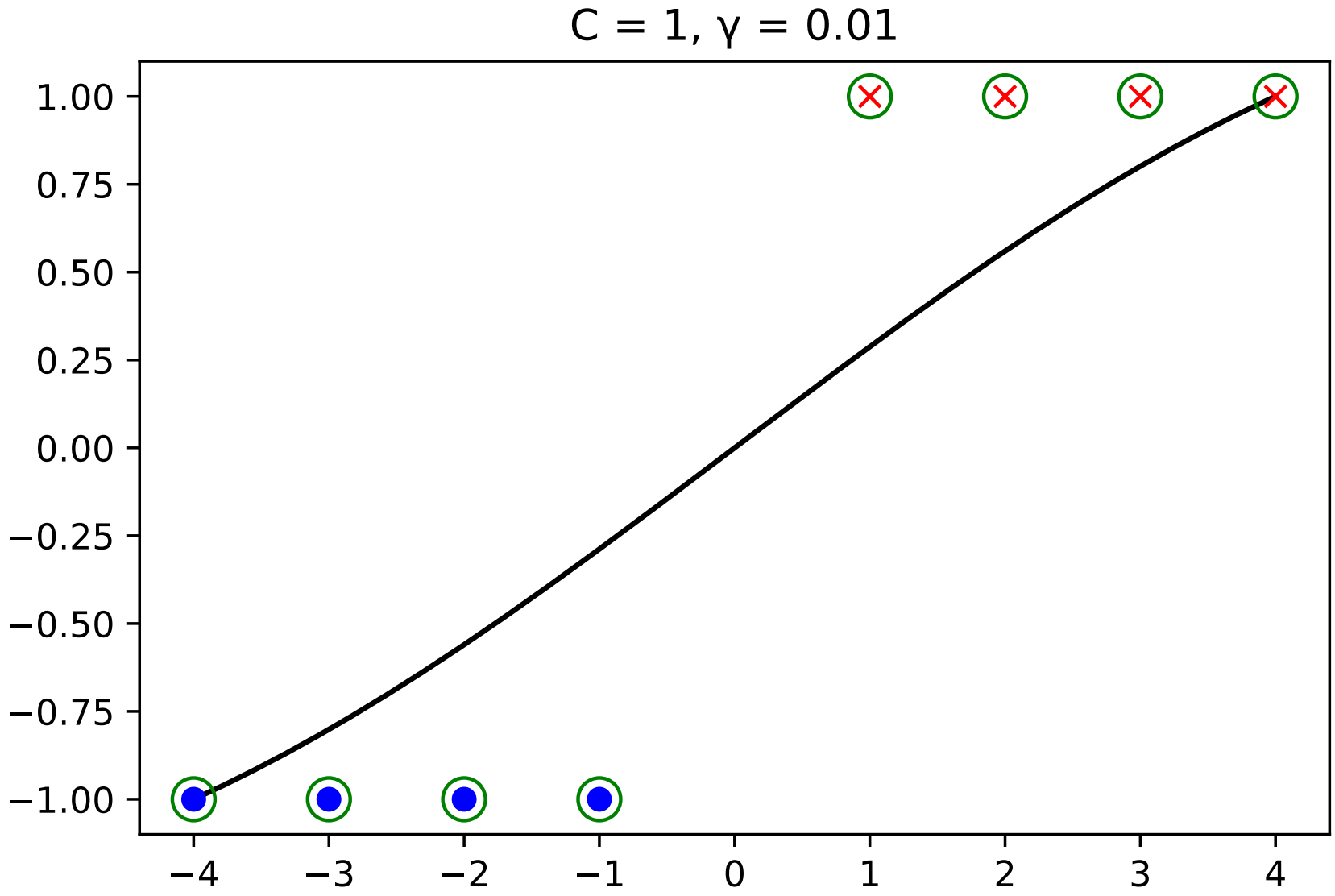
  - Quanto maior γ, mais "concentrada" é a gaussiana
  - Corresponde a um vetor $\boldsymbol{\varphi}(\mathbf{x})$ de comprimento infinito (!)
  - Desempenho nunca inferior ao linear se (C, γ) puderem ser escolhidos

- Kernel sigmoidal:

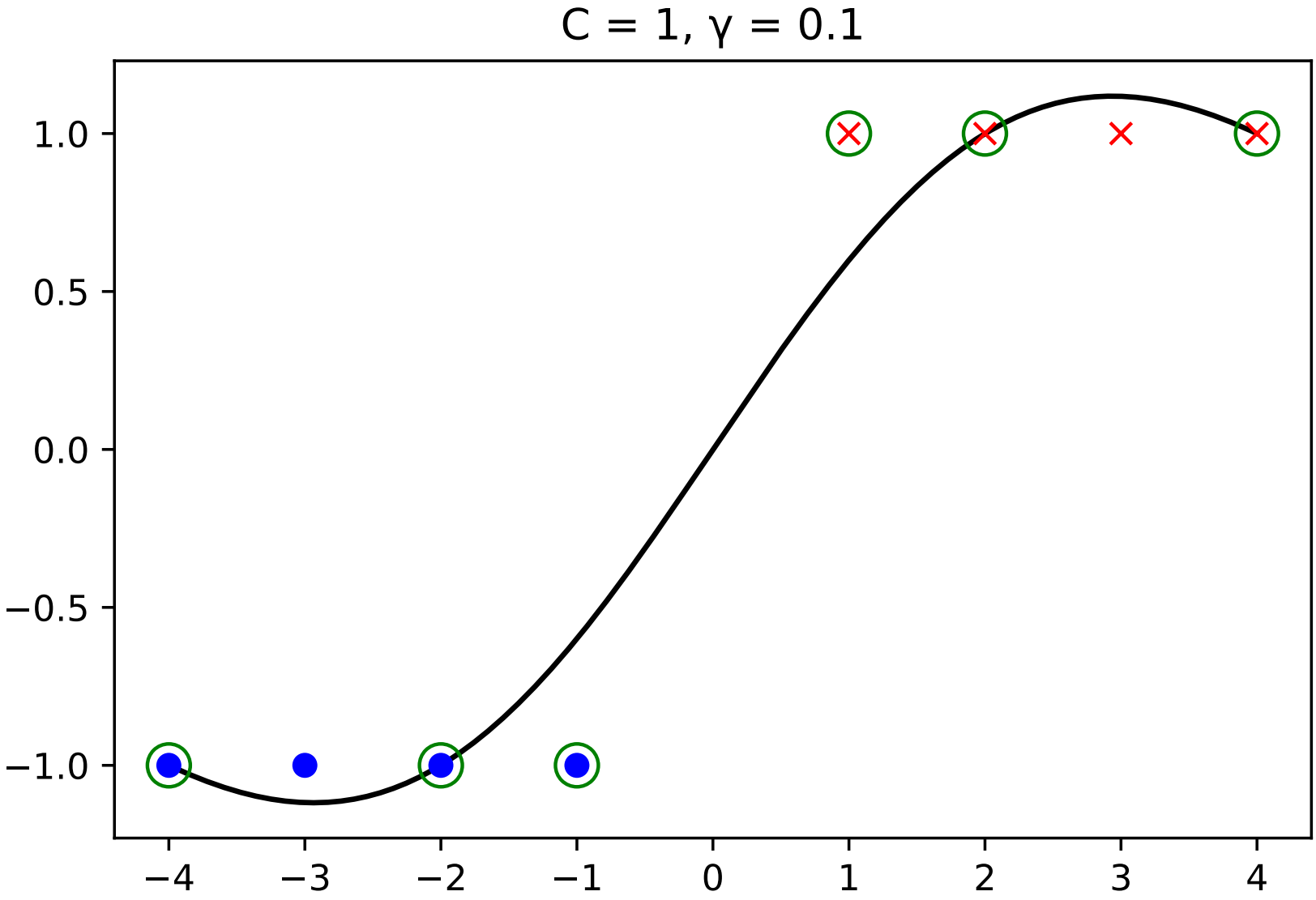
$$K(\mathbf{x}, \mathbf{x}') = \tanh(r + \gamma \mathbf{x}^T \mathbf{x}')$$

- Note que γ, r e d (assim como C) são hiperparâmetros

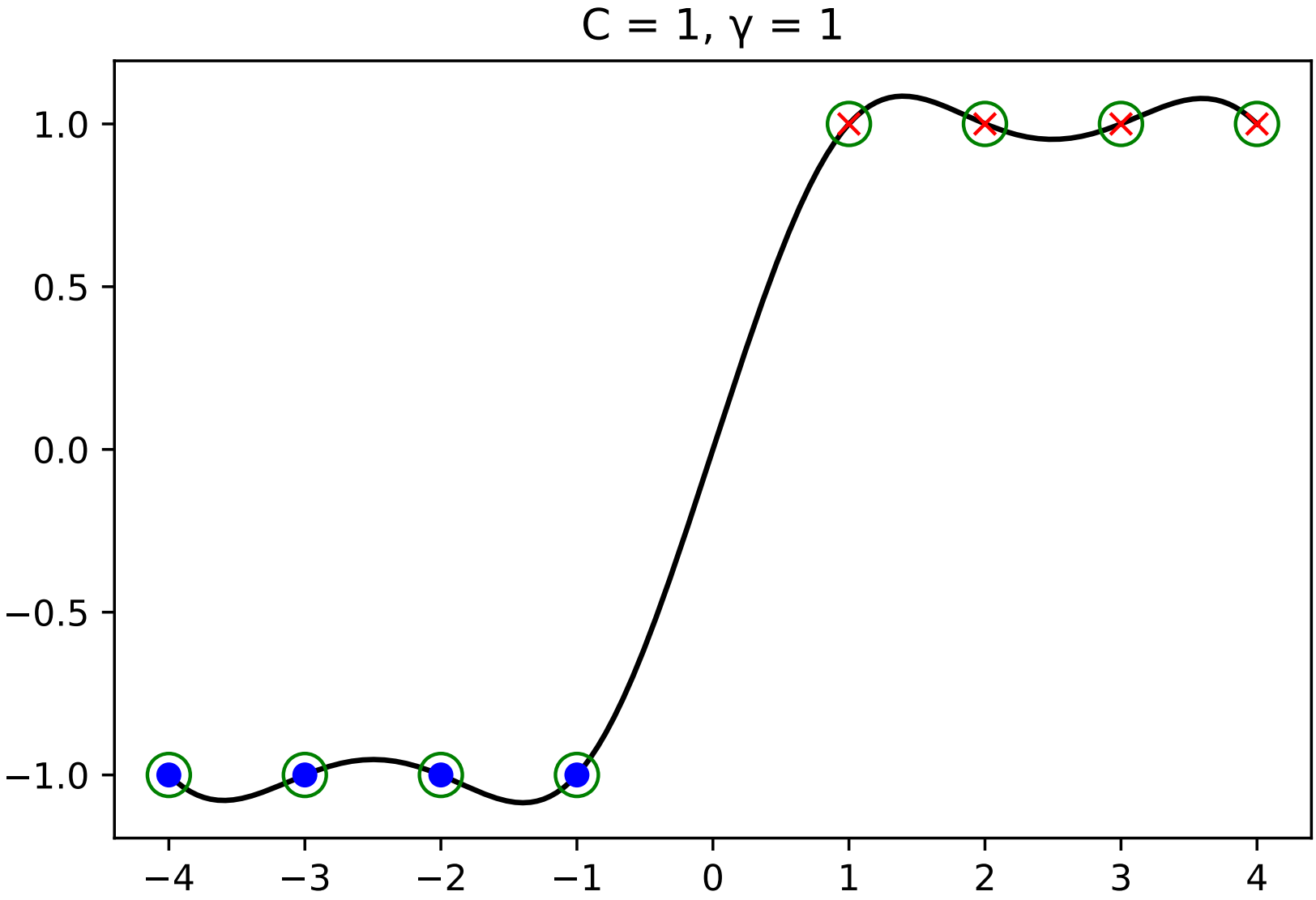
Exemplos (Kernel RBF)



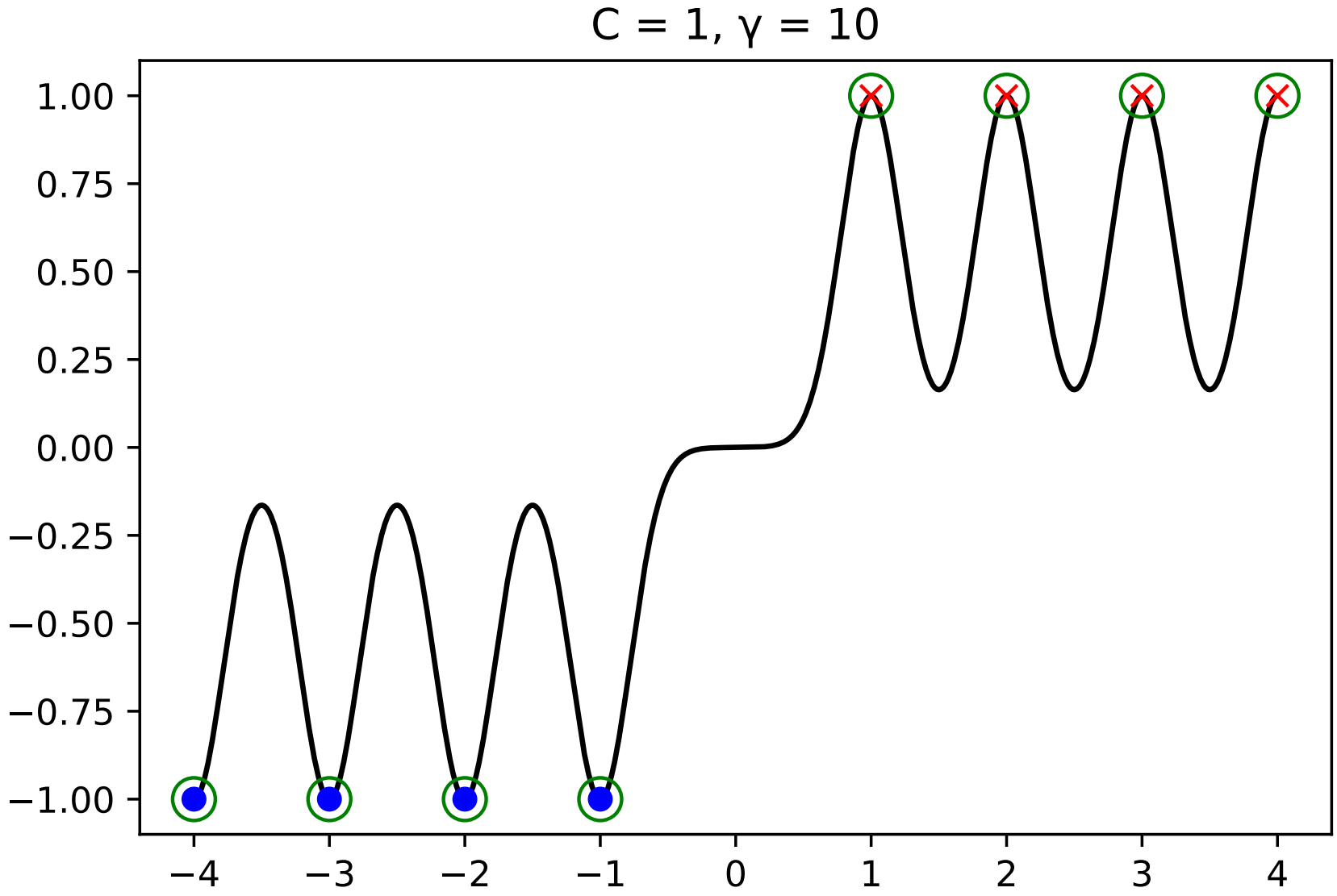
Exemplos (Kernel RBF)



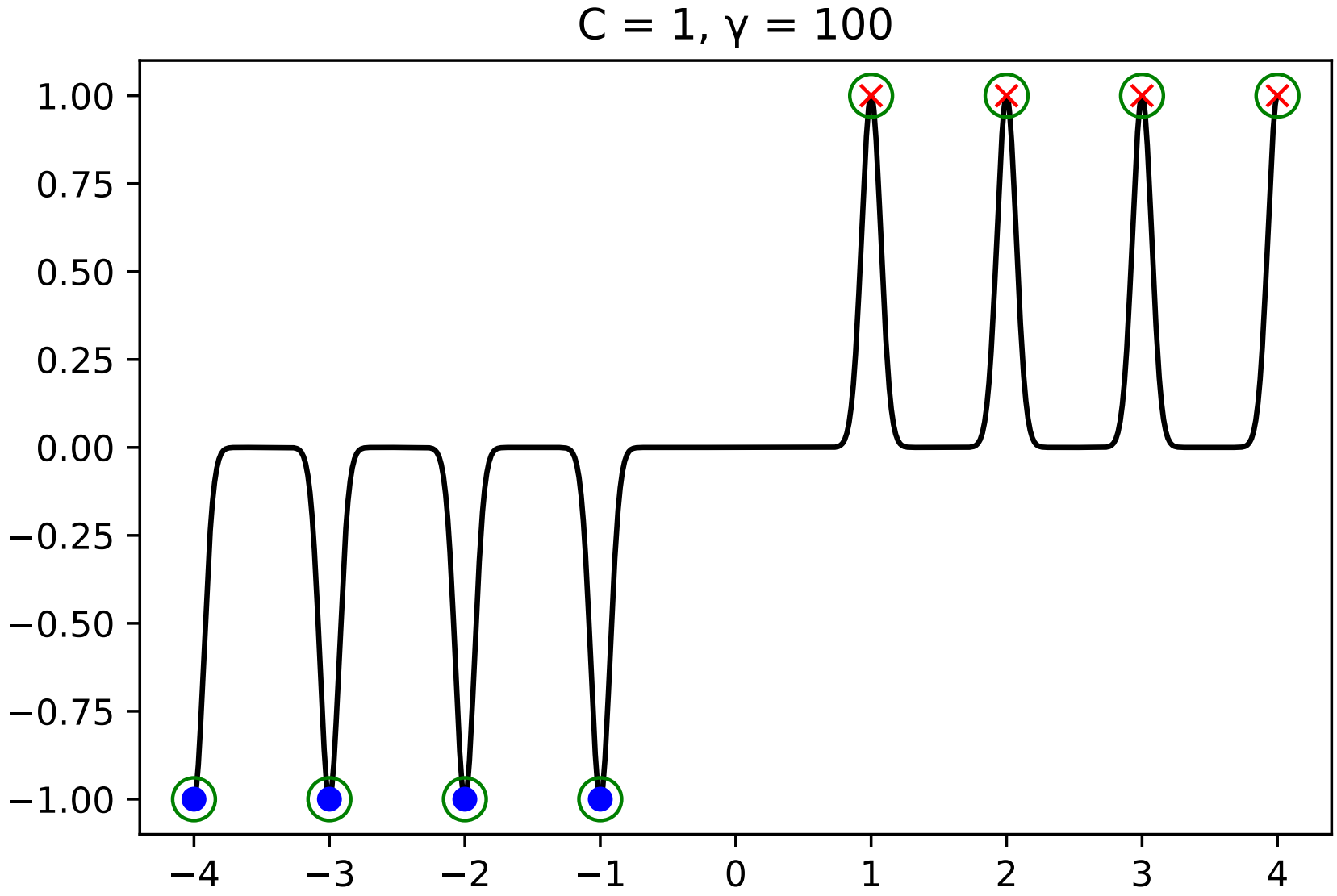
Exemplos (Kernel RBF)



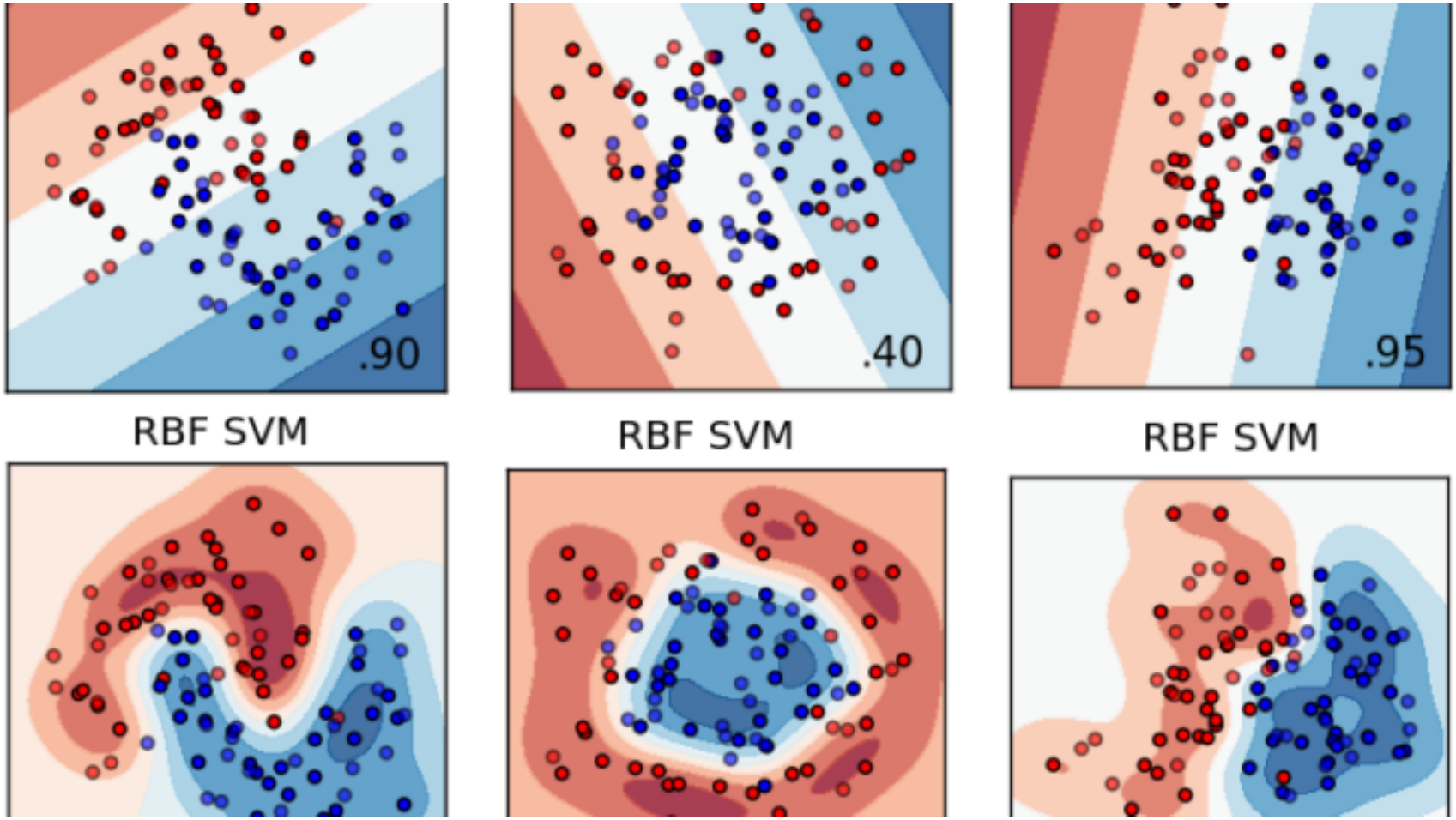
Exemplos (Kernel RBF)



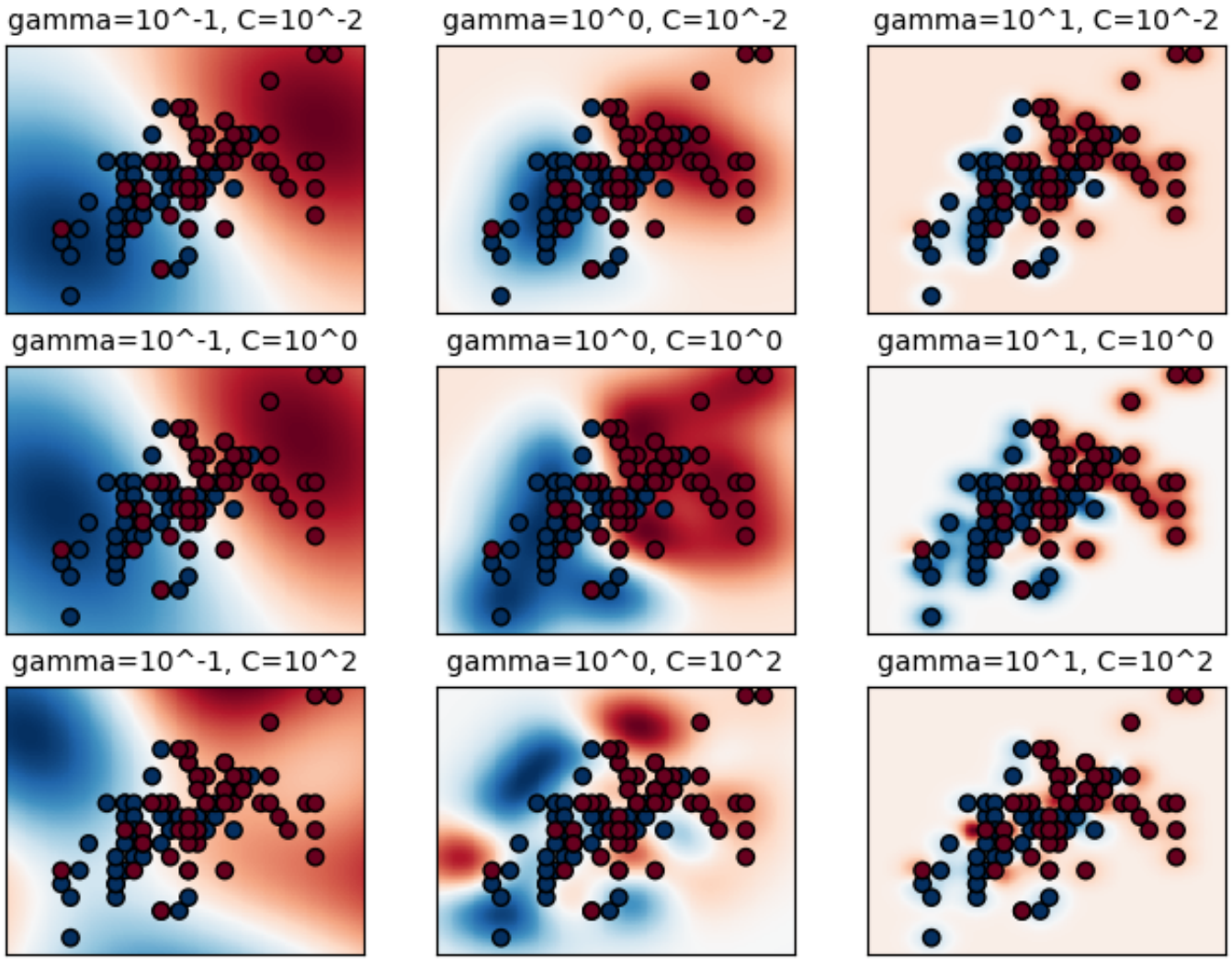
Exemplos (Kernel RBF)



Exemplos (Kernel RBF)



Exemplos (Kernel RBF)



Na Prática

- Escalonamento é importante
- Kernel RBF é uma boa escolha “padrão”
- Se n é muito grande, o kernel linear pode ser uma solução mais eficiente
- Hiperparâmetros (como C e γ) são tipicamente encontrados através de validação cruzada no conjunto de treinamento