Máquinas de Vetores Suporte

Prof. Clodoaldo A M Lima

Universidade de São Paulo

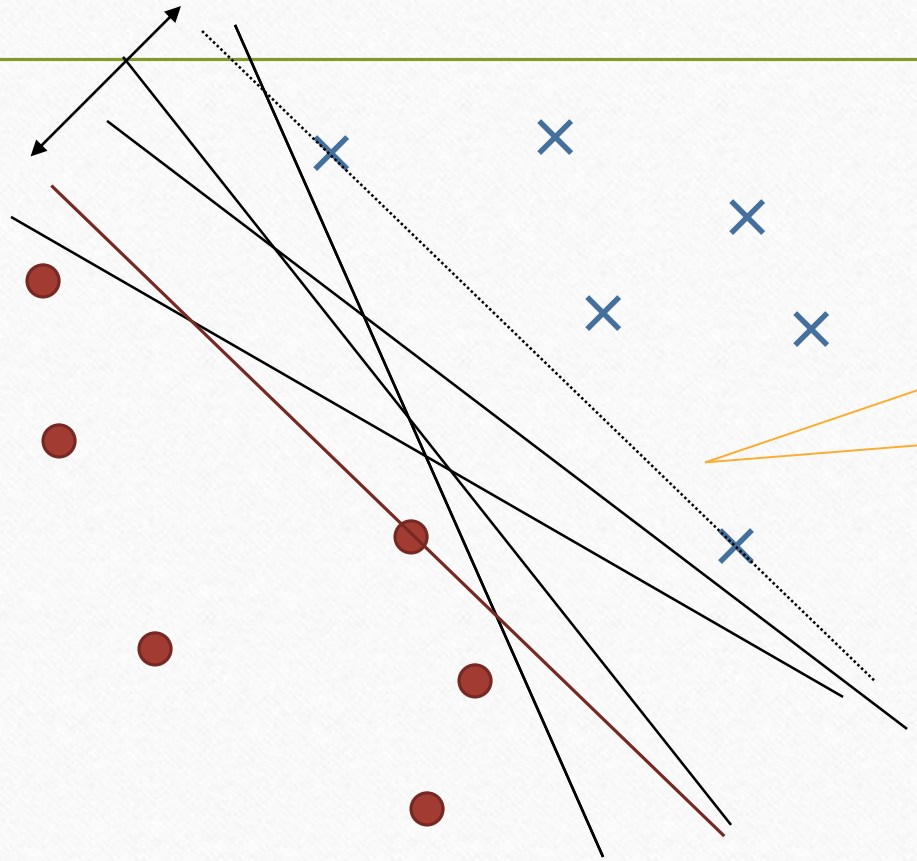
Introdução

- Maquinas de Vetores Suporte
  - Usa espaço de hipótese de funções lineares no espaço de característica de alta dimensionalidade, treinadas com um algoritmo baseado na teoria de otimização que implementa a teoria de aprendizado estatístico.
- Palavras chaves
  - Maquinas de aprendizado Linear
  - Funções kernel
    - Usado para definir o espaço de característica implícito, no qual a máquina de aprendizado linear opera.
    - Responsável pelo uso eficiente do espaço de característica de alta dimensionalidade.
  - Teoria de Otimização → Representação Compacta

Problemas Tratados

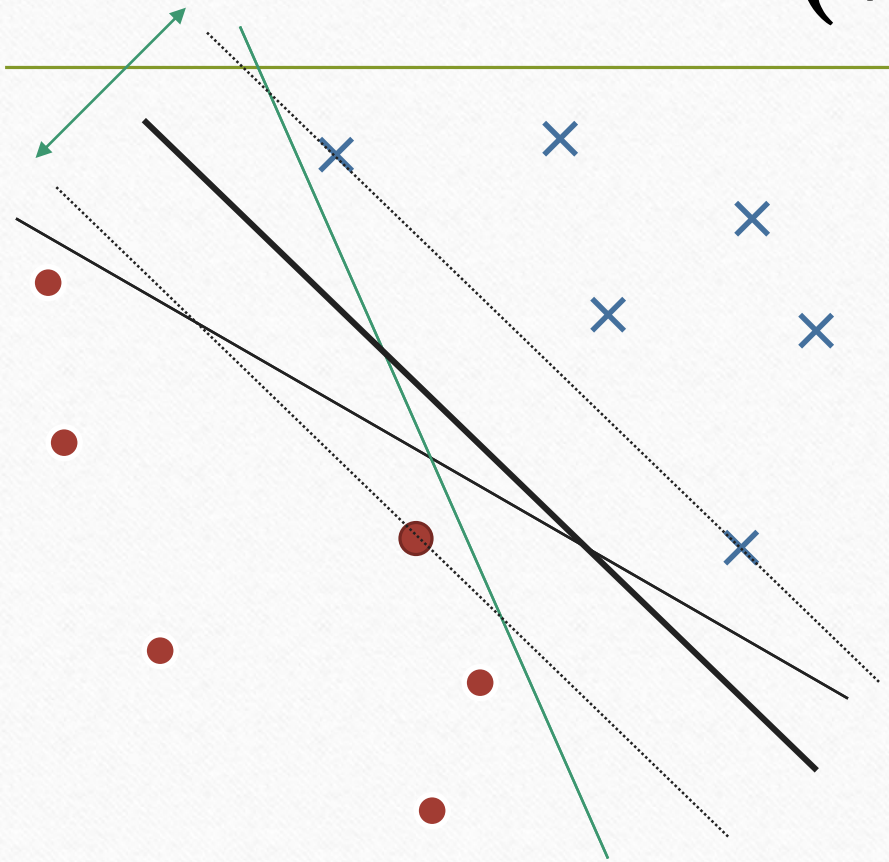
- **1) Problema conceitual**
  - Como controlar a complexidade do conjunto de aproximação.
    - Funções em alta dimensão a fim de proporcionar boa capacidade de generalização
    - Usar estimadores lineares penalizados com um grande número de funções-base
- **2) Problema Computacional**
  - Como realizar otimização numérica em espaço de alta dimensão
    - Usar uma representação kernel dual de funções lineares

Problema Linearmente Separável



- Há infinitas linhas que têm erro de treinamento zero
- Qual delas deveremos escolher?

Hiperplano de separação de margem ótima (Vapnik)



- vetores $\mathbf{x}_i$
- rótulos $y_i = \pm 1$

O Hiperplano de Separação Ótimo

- A distância $d(\mathbf{w},b;\mathbf{x})$ de um vetor $x \in \Re^n$ ao hiperplano pode ser expressa na forma:

$$d(\mathbf{w},b;\mathbf{x}) = \frac{|(\mathbf{w}\cdot\mathbf{x})+b|}{\|\mathbf{w}\|}$$

- Com isso, a margem de separação, sujeita às restrições $y_i[(\mathbf{w}\cdot\mathbf{x})+b] \geq 1$, com $i = 1,\ldots,N$., é dada por

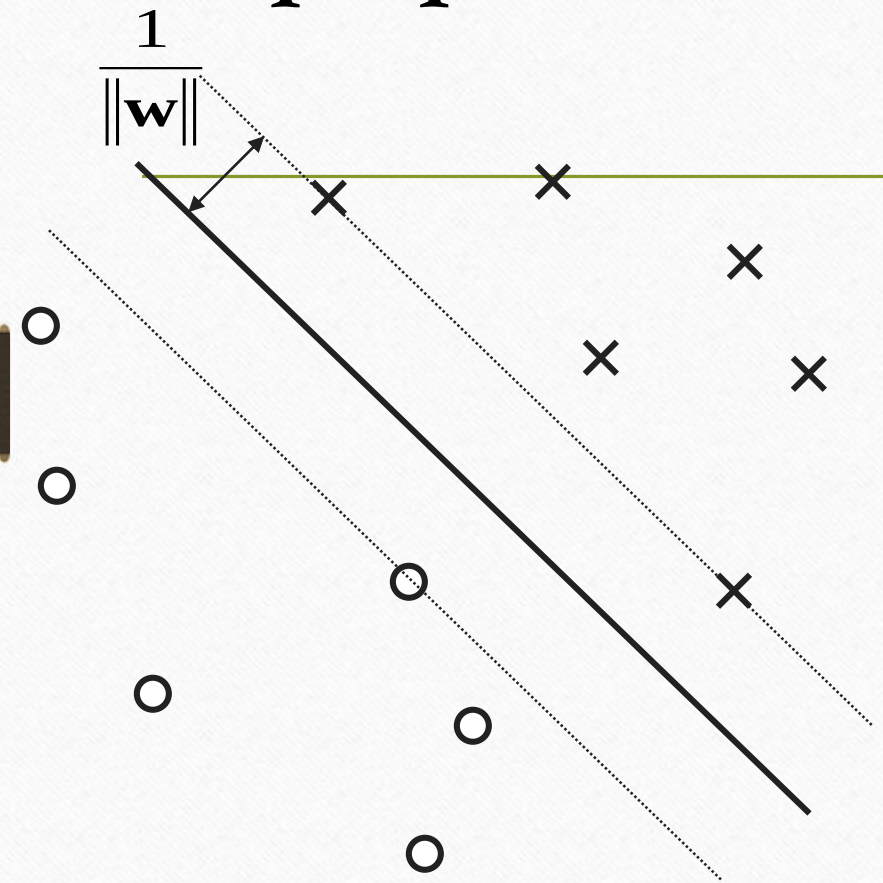
$$\rho(\mathbf{w},b) = \min_{\{x_i,y_i=1\}} d(\mathbf{w},b;\mathbf{x}_i) + \min_{\{x_j,y_j=-1\}} d(\mathbf{w},b;\mathbf{x}_j)$$

$$\rho(\mathbf{w},b) = \min_{\{x_i,y_i=1\}} \frac{|\mathbf{w}\cdot\mathbf{x}_i+b|}{\|\mathbf{w}\|} + \min_{\{x_i,y_i=-1\}} \frac{|\mathbf{w}\cdot\mathbf{x}_j+b|}{\|\mathbf{w}\|}$$

$$\rho(\mathbf{w},b) = \frac{1}{\|\mathbf{w}\|}\left(\min_{\{\mathbf{x}_i,y_i=1\}} |\mathbf{w}\cdot\mathbf{x}_i+b| + \min_{\{x_i,y_i=-1\}} |\mathbf{w}\cdot\mathbf{x}_j+b|\right)$$

$$\rho(\mathbf{w},b) = \frac{2}{\|\mathbf{w}\|}$$

Hiperplano de separação de margem ótima (Vapnik)



$$\min_{\mathbf{w}} : \|\mathbf{w}\|$$

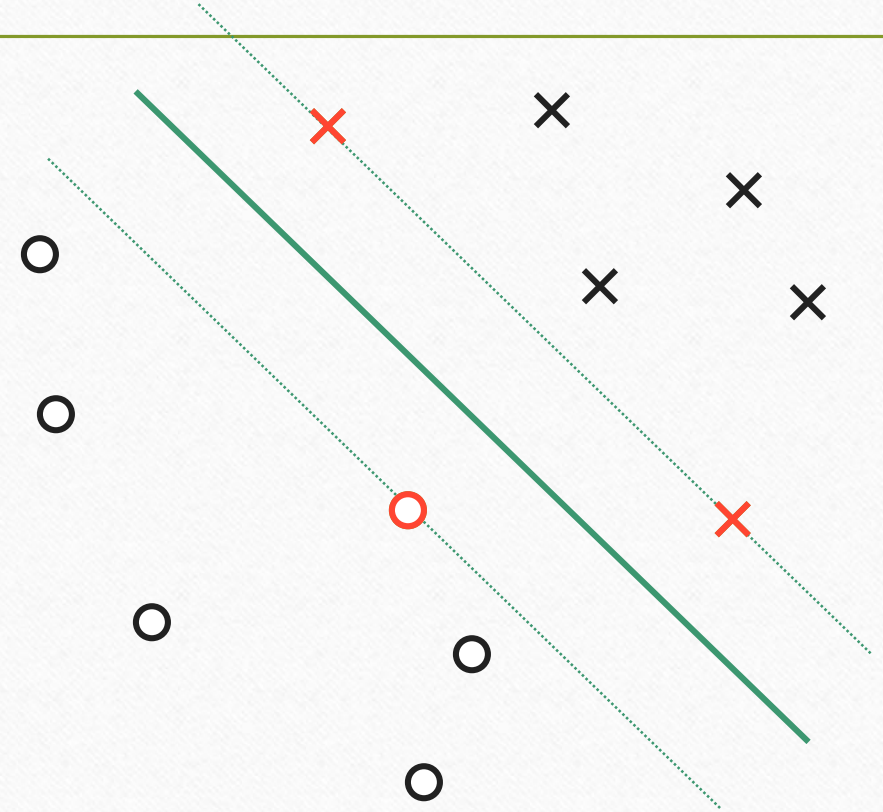
$$\frac{1}{2}\|w\|^2$$

$$y_i(\mathbf{w}.\mathbf{X}_i + b) \geq 1$$

$$f(\mathbf{X}) = sign(\mathbf{w} \cdot \mathbf{X} + b)$$

Hiperplano de separação de margem ótima

Vetores Suporte



– vetores $\mathbf{X}_i$

– rótulos $y_i = \pm 1$

$$\frac{1}{2}\|w\|^2$$

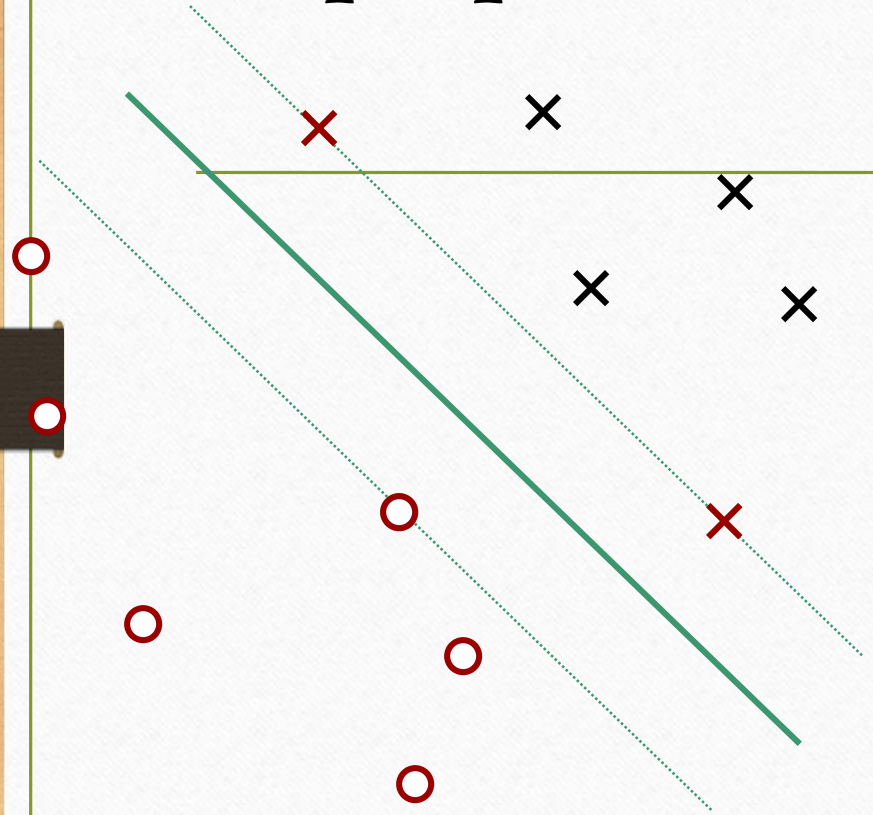
$$y_i(\mathbf{w}\cdot\mathbf{X}_i + b) \geq 1$$

$$f(\mathbf{X}) = sign(\mathbf{w}\cdot\mathbf{X} + b)$$

– Vetores suporte:

$$y_i(\mathbf{w}\cdot\mathbf{X}_i + b) = 1, \quad i \in S$$

Hiperplano de separação de margem ótima



- vetores $\mathbf{X}_i$
- rótulos $y_i = \pm 1$

$$\frac{1}{2}\|w\|^2$$

$$y_i(\mathbf{w}\cdot\mathbf{X}_i + b) \geq 1$$

$$f(\mathbf{X}) = sign(\mathbf{w}\cdot\mathbf{X} + b)$$

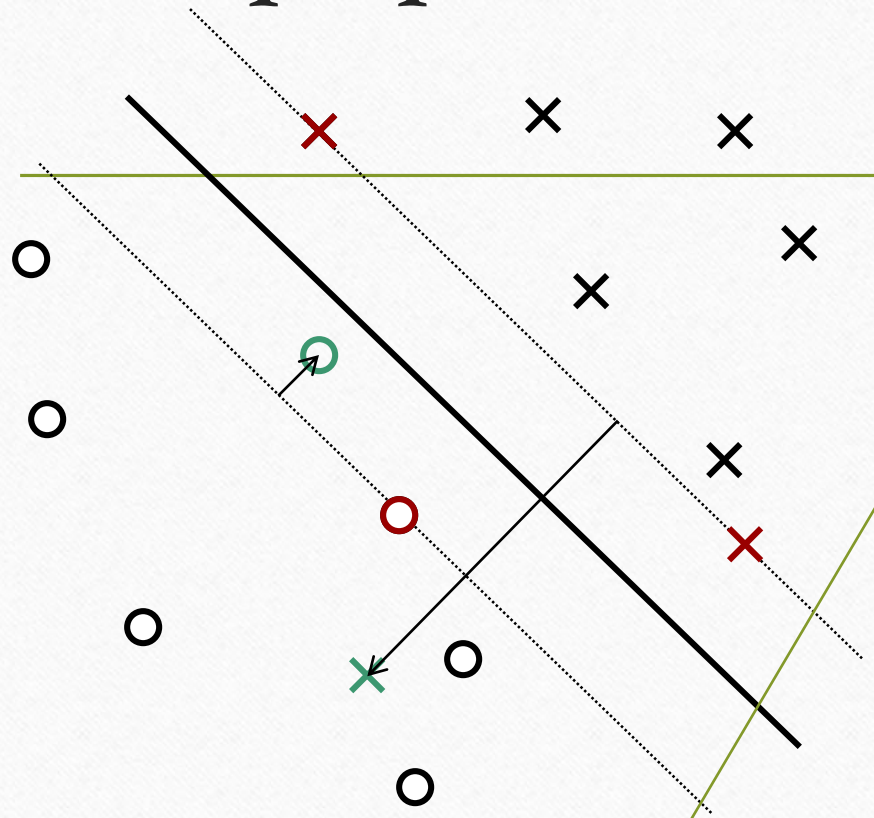
- Vetores suporte:

$$y_i(\mathbf{w}\cdot\mathbf{X}_i + b) = 1, \quad i \in S$$

$$\mathbf{w} = \sum_{i\in S} \alpha_i y_i \mathbf{X}_i$$

$$f(\mathbf{X}) = sign(\sum_{i\in S} \alpha_i y_i \mathbf{X}_i \cdot \mathbf{X} + b)$$

Hiperplano de separação com margem suave



- vetores $\mathbf{X}_i$
- rótulos $y_i = \pm 1$

$$y = \text{sign}(\mathbf{w} \cdot \mathbf{X} + b)$$

- Vetores suporte:

  (vetores da margem e vetores de erro)

$$y_i(\mathbf{w} \cdot \mathbf{X}_i + b) \leq 1, \quad i \in S$$

$$f(\mathbf{X}) = sign(\sum_{i \in S} \alpha_i y_i \mathbf{X}_i \cdot \mathbf{X} + b) \leftarrow \mathbf{w} = \sum_{i \in S} \alpha_i y_i \mathbf{X}_i$$

Formulação do SVM para classificação

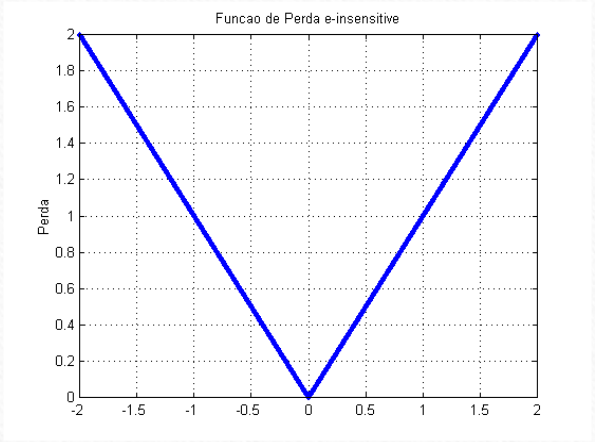
- Problema Primal

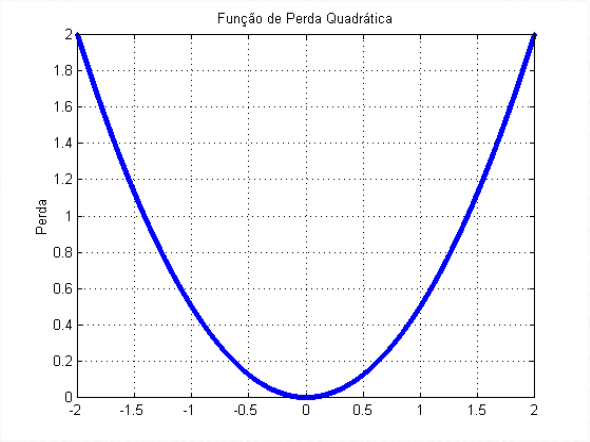
$$\Phi(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} F(\xi_i)$$

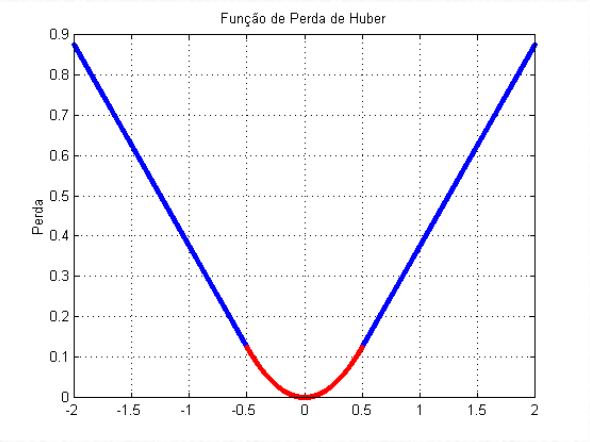
- Sujeito a

$$y_i\left[\left(\mathbf{w}\cdot\mathbf{x}_i\right)+b\right] \geq 1-\xi_i, \qquad i=1,\ldots,N$$

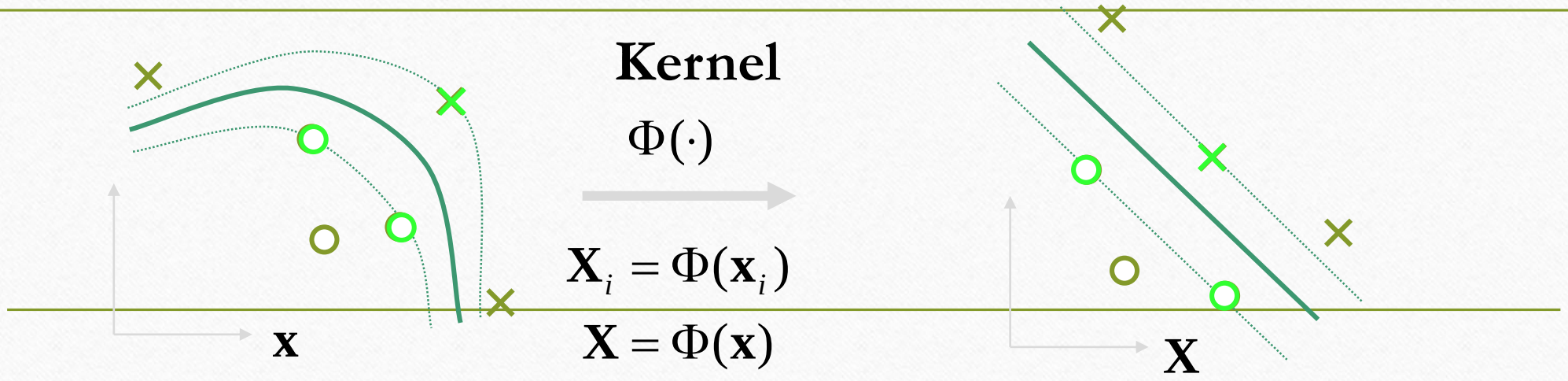
Tipo de Perda







Kernel



$$\Phi(\cdot)$$

$$\mathbf{X}_i = \Phi(\mathbf{x}_i)$$

$$\mathbf{X} = \Phi(\mathbf{x})$$

$$\mathbf{X}_i \cdot \mathbf{X} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})$$

$$f(\mathbf{X}) = sign(\sum_{i \in S} \alpha_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b)$$

$$f(\mathbf{X}) = sign(\sum_{i \in S} \alpha_i y_i \mathbf{X}_i \cdot \mathbf{X} + b)$$

$$K(\cdot,\cdot)$$

$$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) = K(\mathbf{x}_i, \mathbf{x})$$

Condição de Mercer

$$f(\mathbf{x}) = sign(\sum_{i \in S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b)$$

Tipo de Kernel

i. Linear

$$K(x, y) = x \cdot y$$

ii. Polinomial

$$K(x, y) = (x \cdot y + 1)^d$$

iii. Função Gaussiana de Base Radial

$$K(x, y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$$

iv. Função Exponencial Base Radial

$$K(x, y) = \exp\left(-\frac{|x-y|}{2\sigma^2}\right)$$

v. Tangente

$$K(x, y) = \tanh(b(x \cdot y) + c)$$

vi. Séries de Fourier

$$K(x, y) = \frac{sin(N + \frac{1}{2})(x-y)}{sin\left(\frac{1}{2}(x-y)\right)}$$

vii. Linear Splines

$$K(x, y) = 1 + xy + xy\min(x, y) - \frac{(x+y)}{2}\left(\min(x, y)\right)^2 + \frac{1}{3}\left(\max(x, y)\right)^3$$

viii. Bn-splines

$$K(x, y) = B_{2n+1}(x-y)$$

Formulação do SVM para classificação

- Problema Primal

$$\Phi(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} F(\xi_i)$$

- Sujeito a

$$y_i\left[\left(\mathbf{w}\cdot\mathbf{x}_i\right)+b\right] \geq 1-\xi_i, \qquad i=1,\ldots,N$$

Formulação do SVM para classificação

- Problema dual

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$

- Sujeito a

$$0 \le \alpha_i \le C, \qquad i = 1,\ldots,N \qquad\qquad \sum_{i=1}^{N}\alpha_i y_i = 0$$

- Fórmula Usual em Otimização

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\alpha K \alpha^T + c^T\alpha$$

$$0 \le \alpha_i \le C, \qquad i = 1,\ldots,N \qquad\qquad A\alpha = b$$

Exemplo

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$

$$\begin{cases} \alpha_i \geq 0, i = 1,\ldots, N \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases}$$

$$f(\mathbf{x}) = sign\left(\sum_{SVs}\alpha^*_i y_i K(\mathbf{x}_i,\mathbf{x}) + b^*\right)$$

$$K(x_i, x_j) = (< x_i, x_j > +1)^2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

Exemplo

$$K(x_i, x_j) = (< x_i, x_j > +1)^2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

x_1	x_2	y
-1	-1	-1
1	-1	1
-1	1	1
1	1	-1

Matriz Kernel

9	1	1	1
1	9	1	1
1	1	9	1
1	1	1	9

9	1	1	1
1	9	1	1
1	1	9	1
1	1	1	9

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$

$$\min_{\alpha} W(\alpha) = \frac{1}{2}(\alpha_1\alpha_1(-1)(-1)9 + \alpha_1\alpha_2(-1)(1)1 + \alpha_1\alpha_3(-1)(1)1 + \alpha_1\alpha_4(-1)(-1)1$$
$$\alpha_2\alpha_1(1)(-1)1 + \alpha_2\alpha_2(1)(1)9 + \alpha_2\alpha_3(1)(1)1 + \alpha_2\alpha_4(1)(-1)1$$
$$\alpha_3\alpha_1(1)(-1)1 + \alpha_3\alpha_2(1)(1)1 + \alpha_3\alpha_3(1)(1)9 + \alpha_3\alpha_4(1)(-1)1$$
$$\alpha_4\alpha_1(-1)(-1)1 + \alpha_4\alpha_2(-1)(1)1 + \alpha_4\alpha_3(-1)(1)1 + \alpha_4\alpha_4(-1)(-1)9)$$
$$-\alpha_1 - \alpha_2 - \alpha_3 - \alpha_4$$

$$\begin{cases} \alpha_i \geq 0, i = 1,\ldots, N \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases}$$

$$-\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0$$

$$-\alpha_1+\alpha_2+\alpha_3-\alpha_4=0$$

$$\min_{\alpha} W(\alpha)=\frac{1}{2}(9\alpha_1\alpha_1-\alpha_1\alpha_2-\alpha_1\alpha_3+\alpha_1\alpha_4$$
$$-\alpha_2\alpha_1+9\alpha_2\alpha_2+\alpha_2\alpha_3-4\alpha_2\alpha_4$$
$$-\alpha_3\alpha_1+\alpha_3\alpha_2+9\alpha_3\alpha_3-4\alpha_3\alpha_4$$
$$\alpha_4\alpha_1-4\alpha_4\alpha_2-4\alpha_4\alpha_3+9\alpha_4\alpha_4)$$

$$-\alpha_1-\alpha_2-\alpha_3-\alpha_4$$

$$\min_{\alpha} W(\alpha)=\frac{1}{2}(9\alpha_1\alpha_1-2\alpha_1\alpha_2-2\alpha_1\alpha_3+2\alpha_1\alpha_4+9\alpha_2\alpha_2+2\alpha_2\alpha_3-2\alpha_2\alpha_4$$
$$+9\alpha_3\alpha_3-2\alpha_3\alpha_4+9\alpha_4\alpha_4)-\alpha_1-\alpha_2-\alpha_3-\alpha_4$$

$$\frac{\partial W(\alpha)}{\partial \alpha_1}=\frac{1}{2}(18\alpha_1-2\alpha_2-2\alpha_3+2\alpha_4)-1=0\rightarrow 8\alpha_1-1=0\rightarrow \alpha_1=\frac{1}{8}$$

$$\frac{\partial W(\alpha)}{\partial \alpha_2}=\frac{1}{2}(-2\alpha_1+18\alpha_2+2\alpha_3-2\alpha_4)-1=0\rightarrow 8\alpha_2-1=0\rightarrow \alpha_2=\frac{1}{8}$$

$$\min_{\alpha} W(\alpha) = \frac{1}{2}(9\alpha_1\alpha_1 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2\alpha_2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3\alpha_3 - 2\alpha_3\alpha_4 + 9\alpha_4\alpha_4) - \alpha_1 - \alpha_2 - \alpha_3 - \alpha_4$$

$$\frac{\partial W(\alpha)}{\partial \alpha_3} = \frac{1}{2}(-2\alpha_1 + 2\alpha_2 + 18\alpha_3 - 2\alpha_4) - 1 = 0 \rightarrow 8\alpha_3 - 1 = 0 \rightarrow \alpha_3 = \frac{1}{8}$$

$$\frac{\partial W(\alpha)}{\partial \alpha_4} = \frac{1}{2}(2\alpha_1 - 2\alpha_2 - 2\alpha_3 + 18\alpha_4) - 1 = 0 \rightarrow 8\alpha_4 - 1 = 0 \rightarrow \alpha_4 = \frac{1}{8}$$

Cálculo da saída

$$f(\mathbf{x}) = sign\left(\sum_{SVs} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x})\right) = sign\left(\sum_{SVs} \alpha_i^* y_i (< x_i, x > +1)^2\right)$$

$$f(\mathbf{x}) = sign\left(\frac{1}{8}(-1)\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2 + \frac{1}{8}(1)\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2 + \frac{1}{8}(1)\left(\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2 + \frac{1}{8}(-1)\left(\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2\right)$$

$$f(\mathbf{x}) = sign\left(-\frac{1}{8}(-x_1 - x_2 + 1)^2 + \frac{1}{8}(x_1 - x_2 + 1)^2 + \frac{1}{8}(-x_1 + x_2 + 1)^2 - \frac{1}{8}(x_1 + x_2 + 1)^2\right)$$

$$f(\mathbf{x}) = sign\left(-\frac{1}{8}\left((x_1 + x_2)^2 - 2(x_1 + x_1) + 1\right) + \frac{1}{8}\left((x_1 - x_2)^2 + 2(x_1 - x_2) + 1\right) + \frac{1}{8}\left((-x_1 + x_2)^2 + 2(-x_1 + x_2) + 1\right) - \frac{1}{8}\left((x_1 + x_2)^2 + 2(x_1 + x_2) + 1\right)\right)$$

Cálculo da saída

$$f(\mathbf{x}) = sign\left(-\frac{1}{8}\left((x_1+x_2)^2 - 2(x_1+x_1)+1\right)+\frac{1}{8}\left((x_1-x_2)^2+2(x_1-x_2)+1\right)+\frac{1}{8}\left((-x_1+x_2)^2+2(-x_1+x_2)+1\right)-\frac{1}{8}\left((x_1+x_2)^2+2(x_1+x_2)+1\right)\right)$$

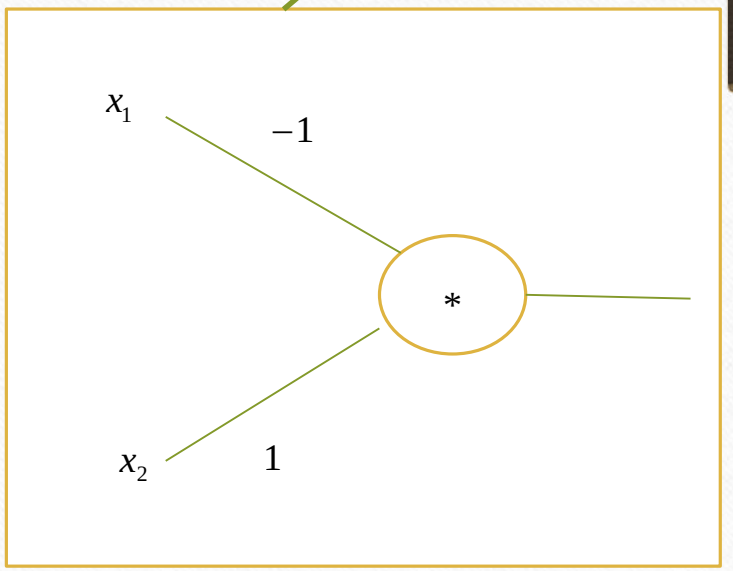
$$f(\mathbf{x}) = sign\left(\frac{1}{8}\left(-(x_1+x_2)^2+2(x_1+x_1)-1+(x_1-x_2)^2+2(x_1-x_2)+1+(-x_1+x_2)^2+2(-x_1+x_2)+1-(x_1+x_2)^2-2(x_1+x_2)-1\right)\right)$$

$$f(\mathbf{x}) = sign\left(\frac{1}{8}\left(-2(x_1+x_2)^2+2(x_1-x_2)^2\right)\right)$$

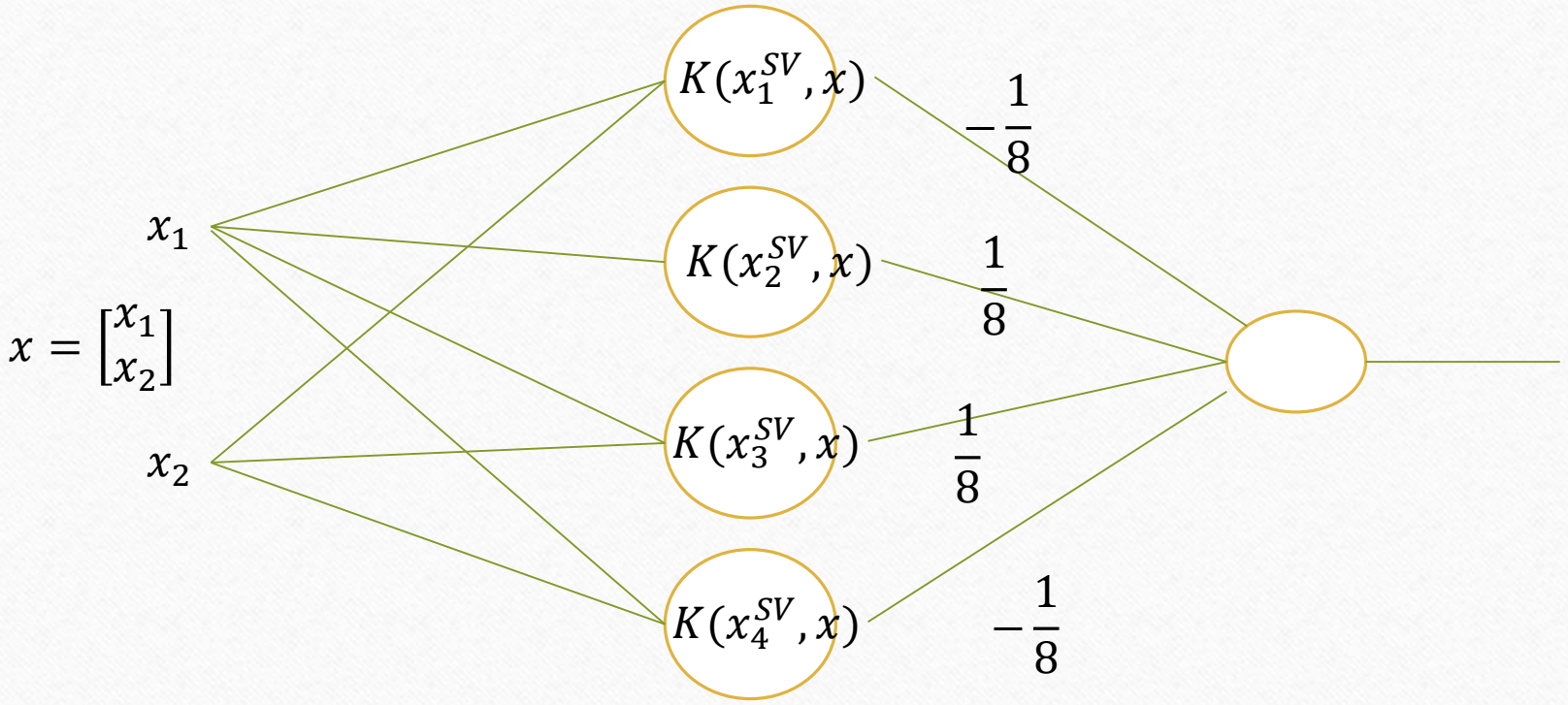
$$f(\mathbf{x}) = sign\left(\frac{2}{8}\left(-x_1^2-2x_1x_2-x_2^2+x_1^2-2x_1x_2+x_2^2\right)\right)$$

$$f(\mathbf{x}) = sign(-x_1x_2)$$

$$b = \frac{1}{N_s}\sum_{s=1}^{N_{SV}}(y_s - f(x_s))$$



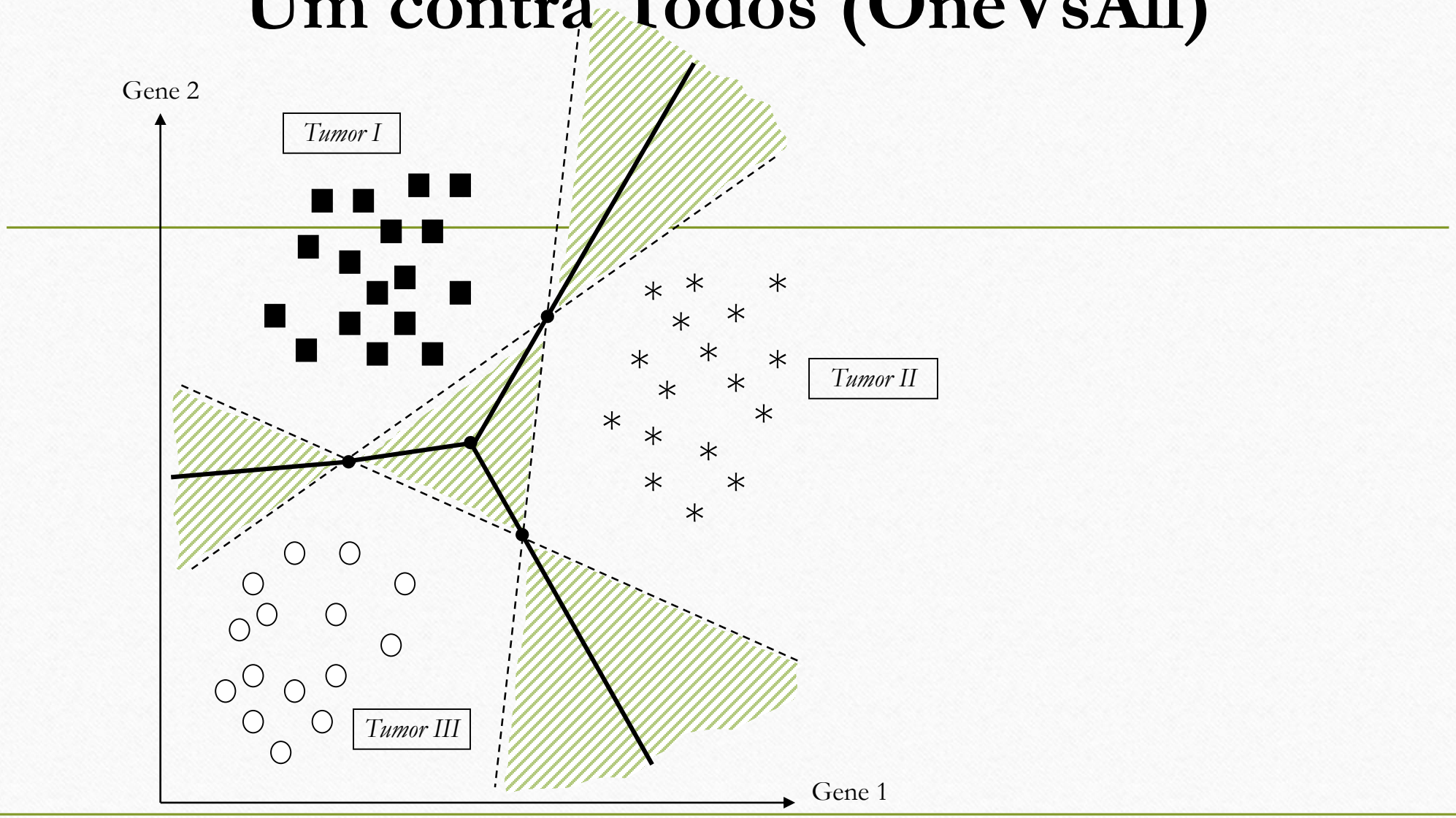
$$f(\mathbf{x}) = sign\left(\frac{1}{8}(-1)\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2+\frac{1}{8}(1)\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2+\frac{1}{8}(1)\left(\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2+\frac{1}{8}(-1)\left(\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+1\right)^2\right)$$



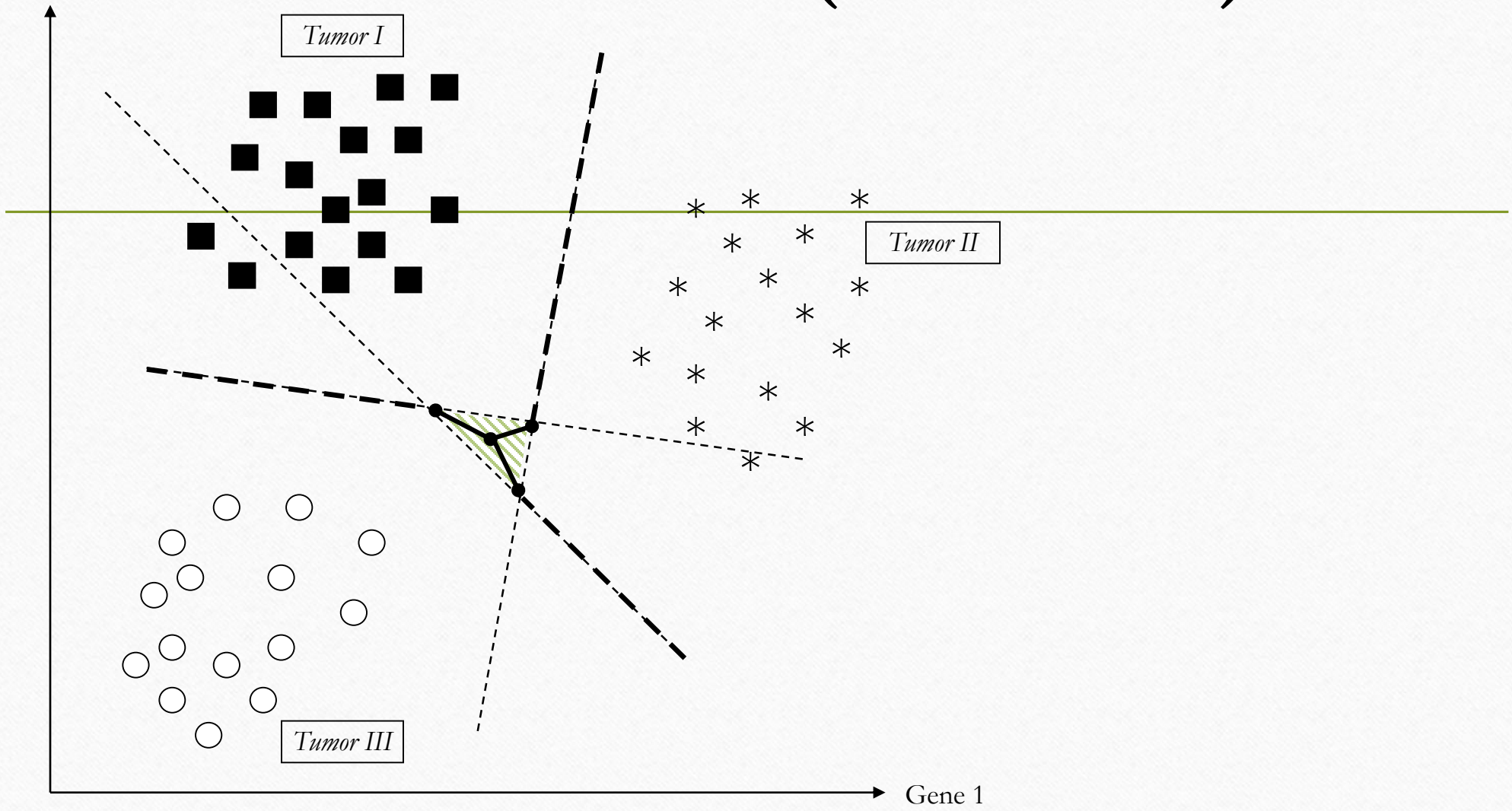
Problema com Múltiplas Classes

- Um contra Todos
- Um contra um
- DAGSVM
- Método proposto por Weston and Watkins
- Método proposto por Crammer and Singer

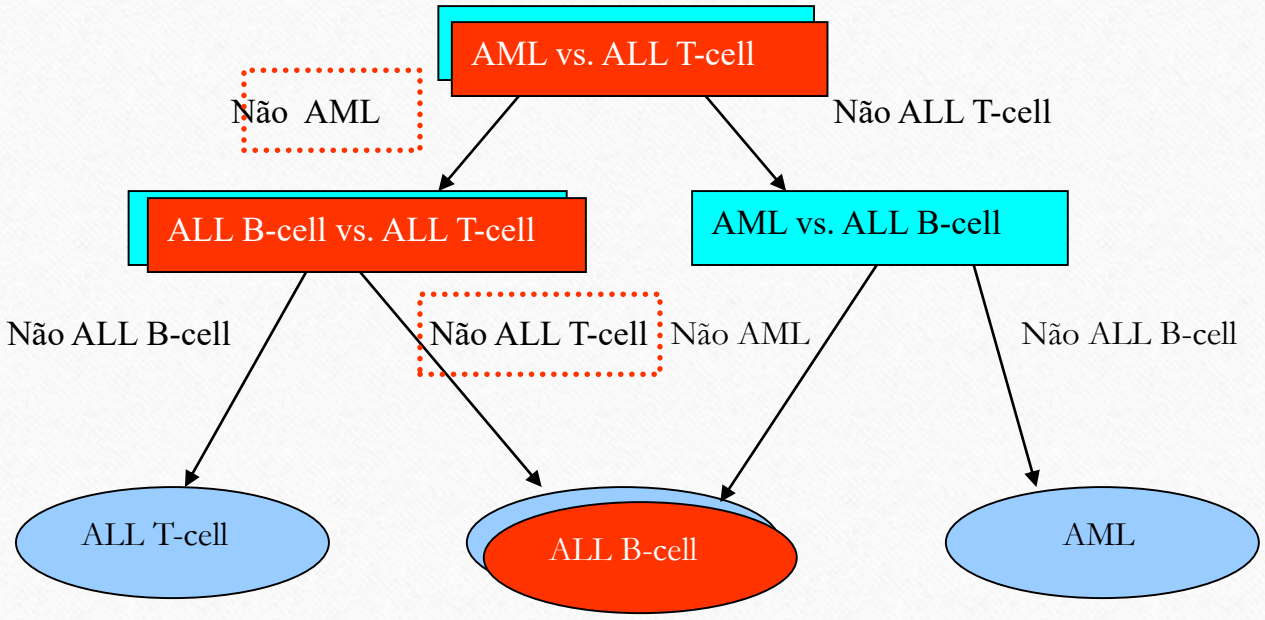
Um contra Todos (OneVsAll)



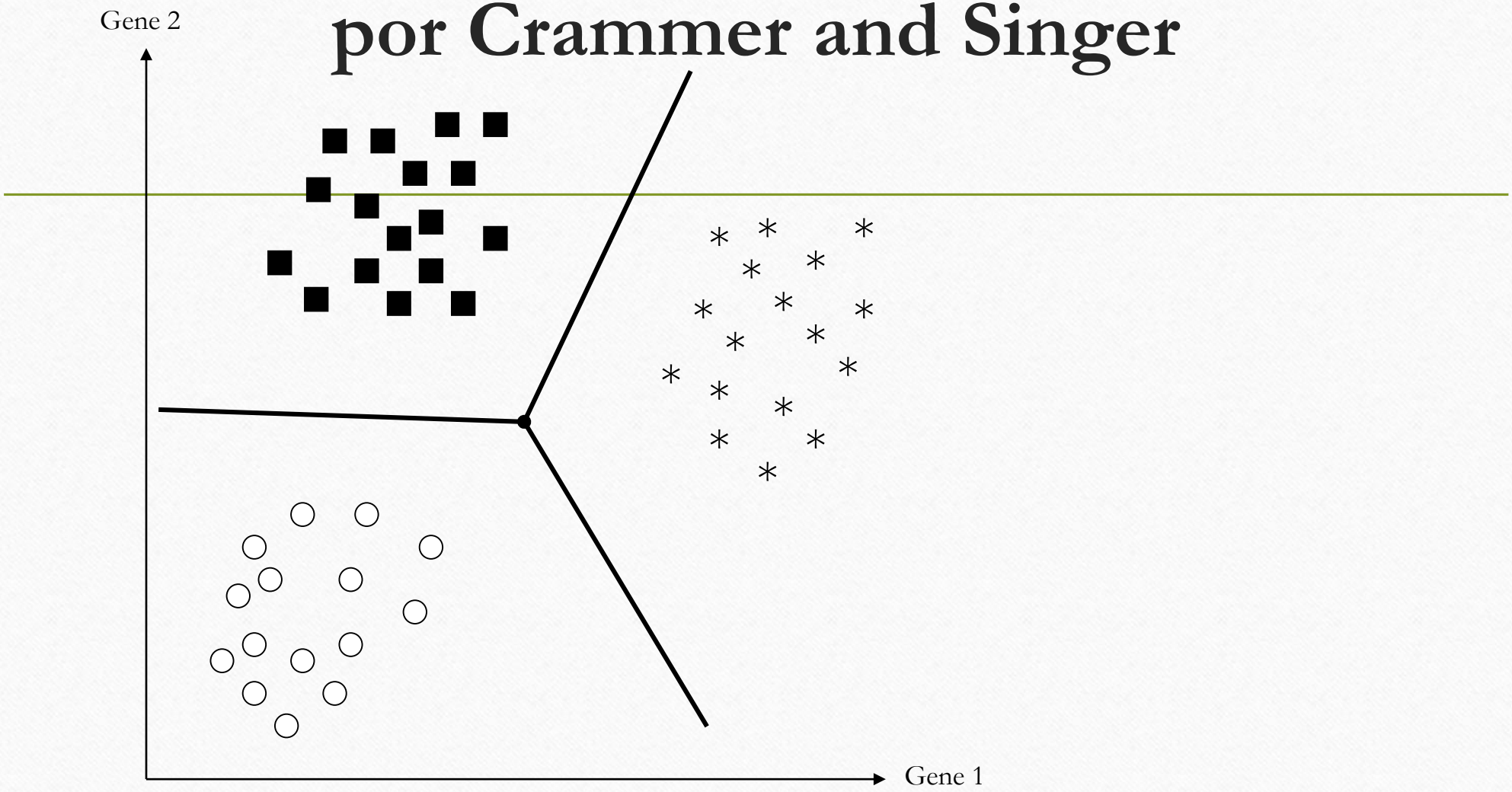
Um contra UM (OneVsOne)



DDAGSVM


AML vs. ALL T-cell
Não AML
Não ALL T-cell
ALL B-cell vs. ALL T-cell
AML vs. ALL B-cell
Não ALL B-cell
Não ALL T-cell
Não AML
Não ALL B-cell
ALL T-cell
ALL B-cell
AML

Método proposto por Weston and Watkins e por Crammer and Singer



Problema de Regressão

- Desenvolvido por Vapnik (1995)

$$(x_1, y_1), \ldots, (x_l, y_l), x \in R^n, y \in R$$

- Modelo: Dado um conjunto de amostras estimar a função:

$$f(x) = (w \cdot \phi(x)) + b$$

- Minimizando

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} F(\xi_i)$$

Formulação do SVM para Regressão

- Problema Primal

$$\Phi(w, \xi^*, \xi) = \frac{1}{2}(w \cdot w) + C\left(\sum_{i=1}^{N} \xi + \sum_{i=1}^{N} \xi_i^*\right)$$

- Sujeito a

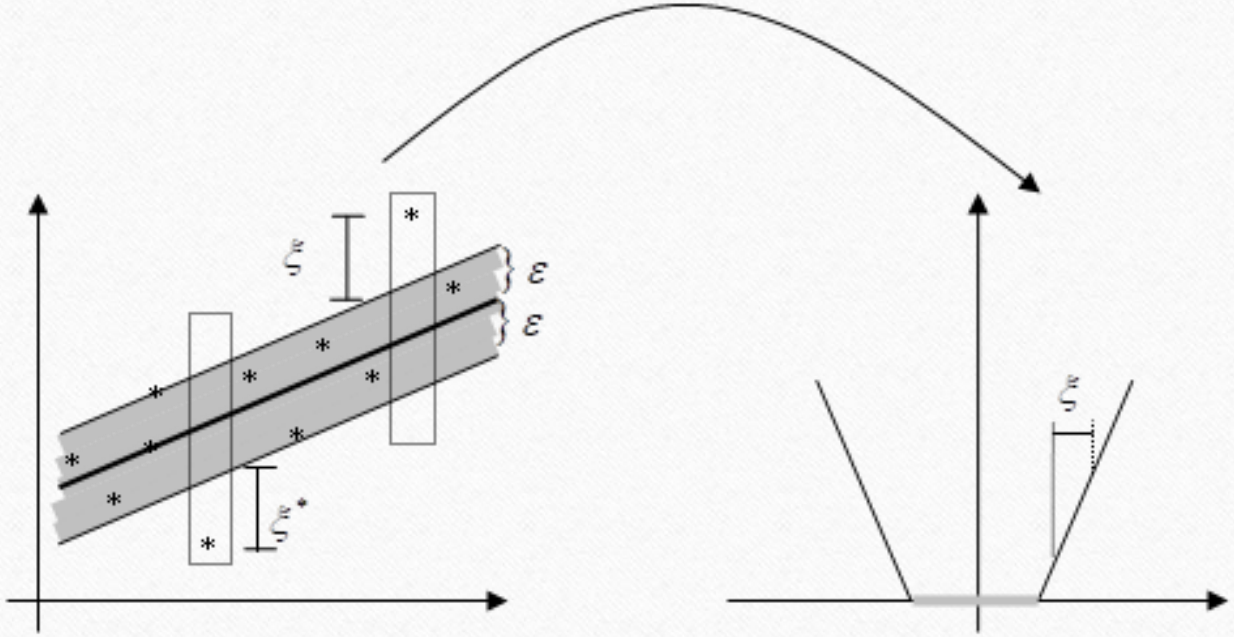
$$y_i - (w \cdot x_i) - b \leq \varepsilon + \xi_i^*, \qquad i = 1, \ldots, N$$

$$(w \cdot x_i) + b - y_i \leq \varepsilon + \xi_i, \qquad i = 1, \ldots, N$$

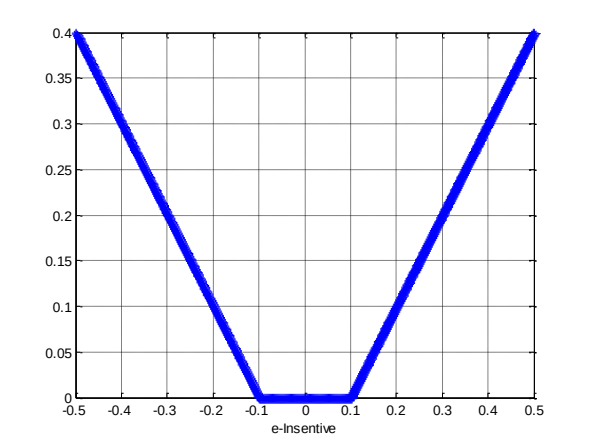
$$\xi_i^* \geq 0$$

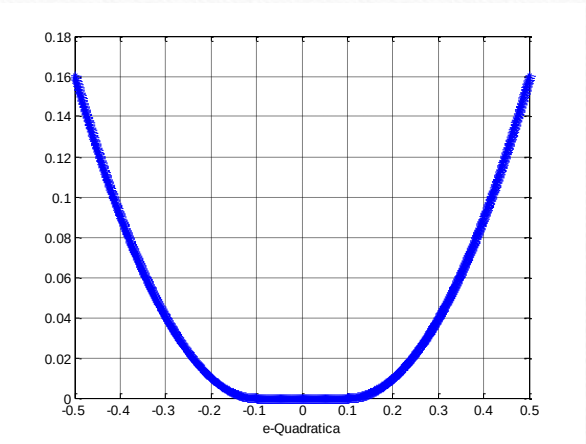
$$\xi_i \geq 0$$

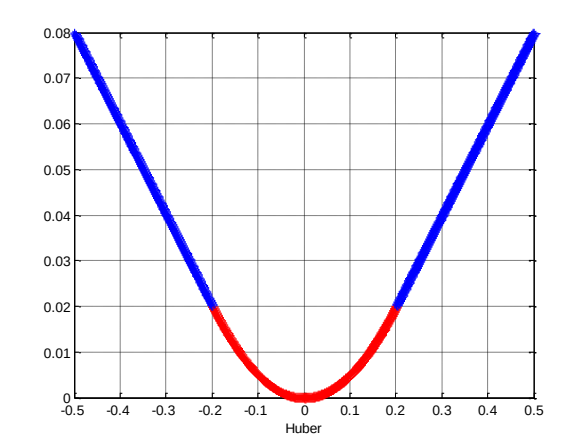
Funções de penalidade para regressão



Funções de penalidade para regressão







- ε-insensível
- ε-quadrática
- huber

Formulação do SVM para Regressão (ε-insensível)

- Problema dual

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_j)K(\mathbf{x}_i,\mathbf{x}_j)(\alpha_i^* - \alpha_j) + \sum_{i=1}^{N}\varepsilon_i(\alpha_i^* + \alpha_i) - \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i)$$

- Sujeito a

$$0 \le \alpha_i^* \le C, \qquad i = 1,\ldots,N$$

$$0 \le \alpha_i \le C, \qquad i = 1,\ldots,N$$

$$\sum_{i=1}^{N}\alpha_i^* = \sum_{i=1}^{N}\alpha_i, \qquad i = 1,\ldots,N$$

- Fórmula Usual em Otimização

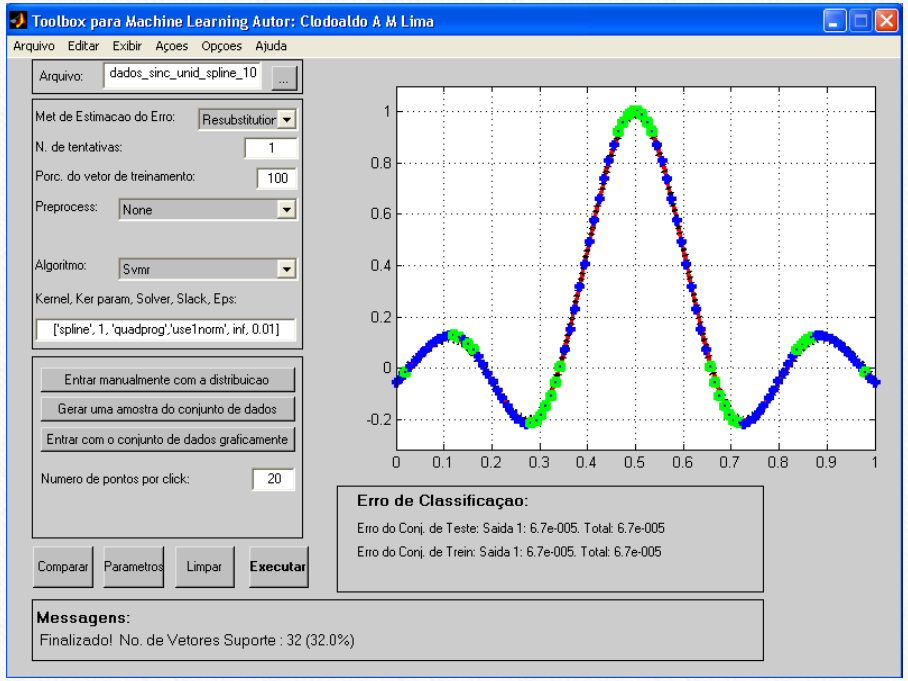
$$\min_{\alpha} W(\alpha) = \frac{1}{2}\beta H\beta^T + c^T\beta$$

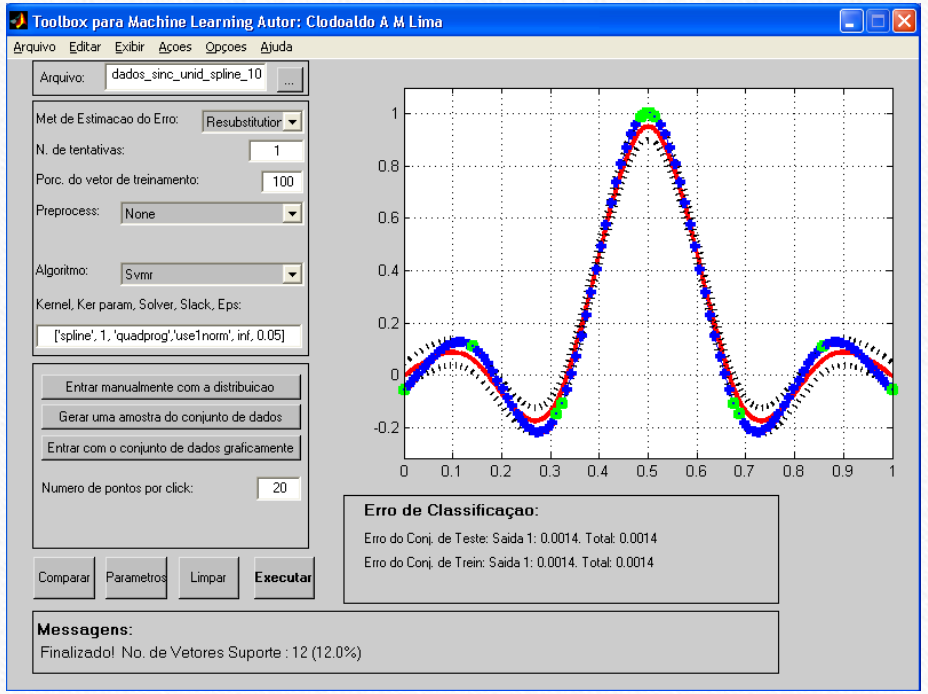
$$0 \le \beta_i \le C, \qquad i = 1,\ldots,N \qquad A\beta = b$$

$$\beta = \begin{bmatrix}\alpha^* \\ \alpha\end{bmatrix} \qquad H = \begin{bmatrix}K & -K \\ -K & K\end{bmatrix}$$

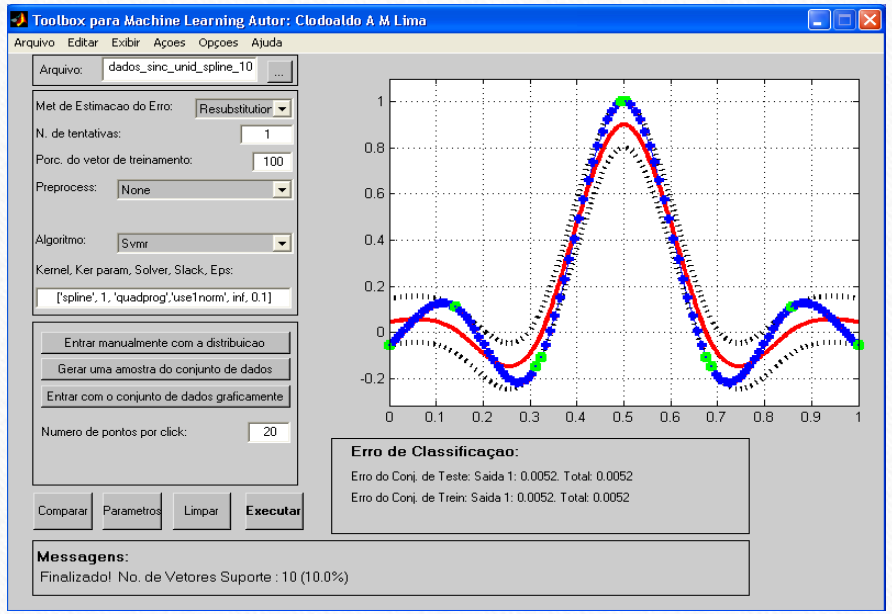
$$A = \begin{bmatrix}1_N \\ -1_N\end{bmatrix} \qquad c = \begin{bmatrix}-y+\varepsilon \\ y+\varepsilon\end{bmatrix}$$

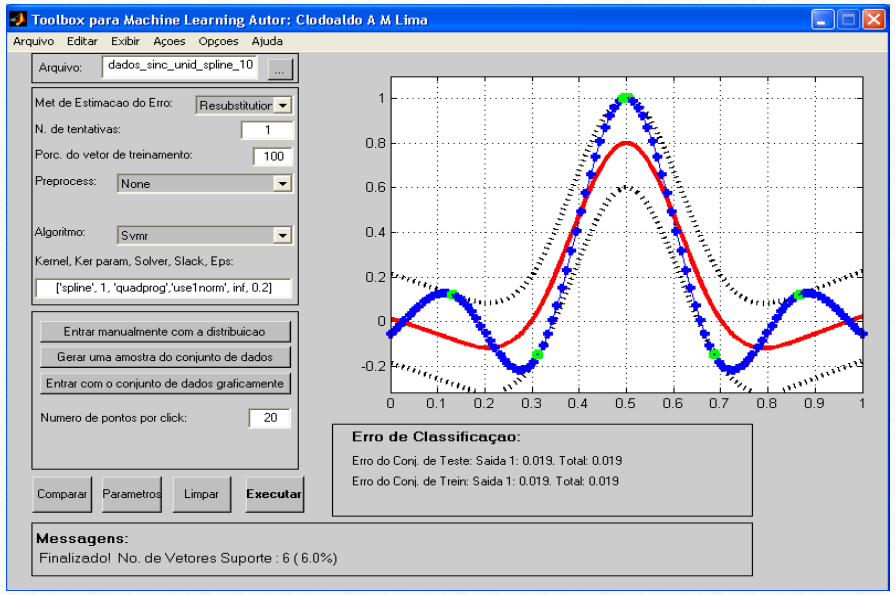
Exemplo


Toolbox para Machine Learning Autor: Clodoaldo A M Lima
Arquivo Editar Exibir Acoes Opcoes Ajuda
Arquivo: dados_sinc_unid_spline_10
Met de Estimacao do Erro: Resubstitutior
N. de tentativas: 1
Porc. do vetor de treinamento: 100
Preprocess: None
Algoritmo: Svmr
Kernel, Ker param, Solver, Slack, Eps:
['spline', 1, 'quadprog','use1norm', inf, 0.01]
Entrar manualmente com a distribuicao
Gerar uma amostra do conjunto de dados
Entrar com o conjunto de dados graficamente
Numero de pontos por click: 20
Comparar Parametros Limpar Executar
Erro de Classificaçao:
Erro do Conj. de Teste: Saida 1: 6.7e-005. Total: 6.7e-005
Erro do Conj. de Trein: Saida 1: 6.7e-005. Total: 6.7e-005
Messagens:
Finalizado! No. de Vetores Suporte : 32 (32.0%)


Toolbox para Machine Learning Autor: Clodoaldo A M Lima
Arquivo Editar Exibir Acoes Opcoes Ajuda
Arquivo: dados_sinc_unid_spline_10
Met de Estimacao do Erro: Resubstitutior
N. de tentativas: 1
Porc. do vetor de treinamento: 100
Preprocess: None
Algoritmo: Svmr
Kernel, Ker param, Solver, Slack, Eps:
['spline', 1, 'quadprog','use1norm', inf, 0.05]
Entrar manualmente com a distribuicao
Gerar uma amostra do conjunto de dados
Entrar com o conjunto de dados graficamente
Numero de pontos por click: 20
Comparar Parametros Limpar Executar
Erro de Classificaçao:
Erro do Conj. de Teste: Saida 1: 0.0014. Total: 0.0014
Erro do Conj. de Trein: Saida 1: 0.0014. Total: 0.0014
Messagens:
Finalizado! No. de Vetores Suporte : 12 (12.0%)

Exemplo


Toolbox para Machine Learning Autor: Clodoaldo A M Lima
Arquivo Editar Exibir Açoes Opçoes Ajuda
Arquivo: dados_sinc_unid_spline_10
Met de Estimacao do Erro: Resubstitution
N. de tentativas: 1
Porc. do vetor de treinamento: 100
Preprocess: None
Algoritmo: Svmr
Kernel, Ker param, Solver, Slack, Eps:
['spline', 1, 'quadprog','use1norm', inf, 0.1]
Entrar manualmente com a distribuicao
Gerar uma amostra do conjunto de dados
Entrar com o conjunto de dados graficamente
Numero de pontos por click: 20
Comparar
Parametros
Limpar
Executar
Erro de Classificaçao:
Erro do Conj. de Teste: Saida 1: 0.0052. Total: 0.0052
Erro do Conj. de Trein: Saida 1: 0.0052. Total: 0.0052
Messagens:
Finalizado! No. de Vetores Suporte : 10 (10.0%)


Toolbox para Machine Learning Autor: Clodoaldo A M Lima
Arquivo Editar Exibir Açoes Opçoes Ajuda
Arquivo: dados_sinc_unid_spline_10
Met de Estimacao do Erro: Resubstitution
N. de tentativas: 1
Porc. do vetor de treinamento: 100
Preprocess: None
Algoritmo: Svmr
Kernel, Ker param, Solver, Slack, Eps:
['spline', 1, 'quadprog','use1norm', inf, 0.2]
Entrar manualmente com a distribuicao
Gerar uma amostra do conjunto de dados
Entrar com o conjunto de dados graficamente
Numero de pontos por click: 20
Comparar
Parametros
Limpar
Executar
Erro de Classificaçao:
Erro do Conj. de Teste: Saida 1: 0.019. Total: 0.019
Erro do Conj. de Trein: Saida 1: 0.019. Total: 0.019
Messagens:
Finalizado! No. de Vetores Suporte : 6 (6.0%)

Condições de KKT

- A solução que satisfaz as condições de KKT é uma solução ótima

- Condições para o SVM com norm1

$$\alpha_j = 0 \longmapsto y_j f(x_j) > 1$$

$$0 < \alpha_j < C \longmapsto y_j f(x_j) = 1$$

$$\alpha_j = C \longmapsto y_j f(x_j) < 1$$

Algumas Variações das SVM

- LS-SVM – Least Square Suporte Vector Machine
- RVM – Relevance Vector Machine
- SSVM – Smooth Support Vector Machine
- LSVM – Lagrangian Support Vector Machine
- PSVM – Proximal Support Vector Machine
- TSVM – Twin Support Vector Machine
- USVM – Universum Support Vector Machine

Máquina de Vetores-Suporte baseada em Quadrados Mínimos – (LS-SVM)

- Abordagem proposta por JOACHIMS (1998),
- Utiliza-se restrições de igualdade ao invés de restrições de desigualdade e a função-custo é a soma do erro quadrático (SSE – *Squared Sum of Error*) como é freqüentemente usada no treinamento de redes neurais.
- Limitação
  - Perda da natureza esparsa do problema, pois todas as amostras do conjunto de treinamento estão contribuindo para o modelo e a importância relativa delas é dada pelo seu valor suporte. Em outras palavras, todas as amostras passam a desempenhar o papel de vetores-suporte.
  - A função custo SSE sem regularização pode conduzir a estimativas que são menos robustas, isto é, quando há *outliers* nos dados de treinamento ou quando não se pode supor que o erro cometido pelo modelo tenha uma distribuição gaussiana.

LS-SVM para problemas de classificação

- Problema Primal

$$J(\mathbf{w}, b, e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{k=1}^{N} e_k^2$$

- Sujeito a

$$y_k[\mathbf{w}^T\varphi(\mathbf{x}_k) + b] = 1 - e_k, \qquad k = 1, \ldots, N$$

LS-SVM para problemas de classificação

- Função Lagrangeana

$$L(w,b,e,\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{k=1}^{N} e_k^2 - \sum_{k=1}^{N}\alpha_k\{y_k[\mathbf{w}^T\varphi(\mathbf{x}_k)+b]-1+e_k\}$$

- As condições de otimalidade

$$\begin{cases} \dfrac{\partial l}{\partial w} = 0 \rightarrow \mathbf{w} = \sum_{k=1}^{N}\alpha_k y_k \varphi(\mathbf{x}_k) \\ \dfrac{\partial l}{\partial b} = 0 \rightarrow \sum_{k=1}^{N}\alpha_k y_k = 0 \\ \dfrac{\partial l}{\partial e_k} = 0 \rightarrow \alpha_k = Ce_k \\ \dfrac{\partial l}{\partial \alpha_k} = 0 \rightarrow y_k[\mathbf{w}^T\varphi(\mathbf{x})+b]-1+e_k = 0,\ k = 1,\ldots,N \end{cases}$$

LS-SVM para problemas de classificação

- A solução é dada por

$$\begin{bmatrix} 0 & -Y^T \\ Y & ZZ^T + C^{-1}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix}$$

- A condição de Mercer pode ser aplicada novamente para a matriz $\Omega = ZZ^T$, onde:

$$\Omega_{kl} = y_k y_l K(\mathbf{x}_k, \mathbf{x}_l)$$

LS-SVM para problemas de regressão

- Problema Primal

$$\min_{w,b,e} J(\mathbf{w},b,e) = \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + \frac{C}{2}\sum_{k=1}^{N} e_k^2$$

- Sujeito

$$y_k = \mathbf{w}^T \varphi(\mathbf{x}_k) + b + e_k \qquad k = 1,\ldots, N$$

- Na formulação primal, tem-se o modelo

$$f(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b$$

LS-SVM para problemas de regressão

- Define-se o Lagrangeano, como:

$$L(\mathbf{w},b,e,\alpha)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+\frac{C}{2}\sum_{k=1}^{N}e_k^2-\sum_{k=1}^{N}\alpha_k\{\mathbf{w}^T\varphi(\mathbf{x}_k)+b+e_k-y_k\}$$

- As condições de otimalidade são dadas por:

$$\begin{cases}\dfrac{\partial l}{\partial w}=0\rightarrow\mathbf{w}=\sum_{k=1}^{N}\alpha_k\varphi(\mathbf{x}_k)\\ \dfrac{\partial l}{\partial b}=0\rightarrow\sum_{k=1}^{N}\alpha_k=0\\ \dfrac{\partial l}{\partial e_k}=0\rightarrow\alpha_k=Ce_k,k=1,\ldots,N\\ \dfrac{\partial l}{\partial \alpha_k}=0\rightarrow\mathbf{w}^T\varphi(\mathbf{x})+b+e_k-y_k=0,k=1,\ldots,N\end{cases}$$

LS-SVM para problemas de regressão

- Depois da eliminação de **w**, obtém-se a solução:

$$\begin{bmatrix} 0 & 1_v^T \\ 1_v^T & \Omega + C^{-1}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{Y} \end{bmatrix}$$

- Utilizando a condição de Mercer, o modelo LS-SVM resultante para regressão torna-se:

$$f(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k K(\mathbf{x}, \mathbf{x}_k) + b$$

ALGORITMOS PARA PROGRAMAÇÃO QUADRÁTICA

- Observe que a função Lagrangeano L é uma função quadrática em temos de α_i e as restrições define uma região convexa, portanto qualquer ótimo local será também um ótimo global

- Abordagens práticas para problemas de Programação Quadrática (QP)
  - Chunking: uma vez que o valor de L é inalterado se remover as linhas e colunas da matriz de kernel correspondente para multiplicadores com valores iguais a zero, o problema original pode dividido em uma seria de pequenos problemas. Essa idéia pode ser implementada utilizando gradiente conjugado.

ALGORITMOS PARA PROGRAMAÇÃO QUADRÁTICA

- Métodos de Decomposição
  - Quebra o problema original em uma série de problemas menores, entretanto, todos estes problemas de programação quadrática são de tamanho fixo, de modo que o método pode ser aplicado a conjunto de dados de qualquer tamanho

- Sequencial minimal optimization(SMO)
  - Leva a ideia de chunking ao extremo, considera apenas dois multiplicadores de Lagrange de cada vez, portanto os subproblemas podem ser solucionados analiticamente.
  - A cada passo, a escolha de um par de multiplicadores é dada por uma heurística (a heurística original é baseada nas condições de KKT).

CHUNKING

- O algoritmo chunking começa com um subconjunto arbitrário (working set) o qual pode ajustar a memória e soluciona o problema de otimização por um otimizador geral.
- Vetores suporte (SVs) permanece no chunk, enquanto os outros pontos são descartados e substituídos por um novo working set com violações graves das condições de KKT.
- A razão desta operação é que somente os vetores suporte contribui para a forma da função de decisão final. Além disso, o algoritmo de chunking é baseado na esparsidade da solução do SVM. Isto é, os vetores suporte correspondem a uma pequena fração de todo conjunto.

CHUNKING

- Um problema com algoritmo chunking é que pode haver mais vetores suporte ativos durante o processo de otimização que aqueles finais tal que seu tamanho vai além do espaço de chunking.

- O método de seleção de um novo working set por avaliando as condições KKT sem eficiente caching do kernel conduz para alto custo computacional.

METODOS DE DECOMPOSIÇÃO

- Osuna sugeriu manter o tamanho constante da matriz para todo subproblema QP, o qual implica em adicionar e deletar o mesmo número de exemplos a todo passo.
- Usando uma matriz de tamanho constante permite o treinamento de conjunto de dados de tamanho arbitrário.
- O algoritmo proposto por Osuna sugere adicionar e deletar um exemplo a cada passo. Tal algoritmo converge, embora possa ser não muito rápido na prática
- Na prática, pesquisadores adicionam múltiplos e deletam múltiplos exemplos de acordo com várias heurísticas.

METODOS DE DECOMPOSIÇÃO

- Tipicamente, estas heurísticas adicionam violadores de KKT a cada passo e deletam aqueles α_i que são zero ou C.
- Joachims publicou um algoritmo para adicionar e deletar exemplos a partir dos passos do QP, o qual rapidamente decrementa a função objetiva.
- Todos os métodos de decomposição requerem um pacote numérico para solucionar um QP

COMPARATIVO

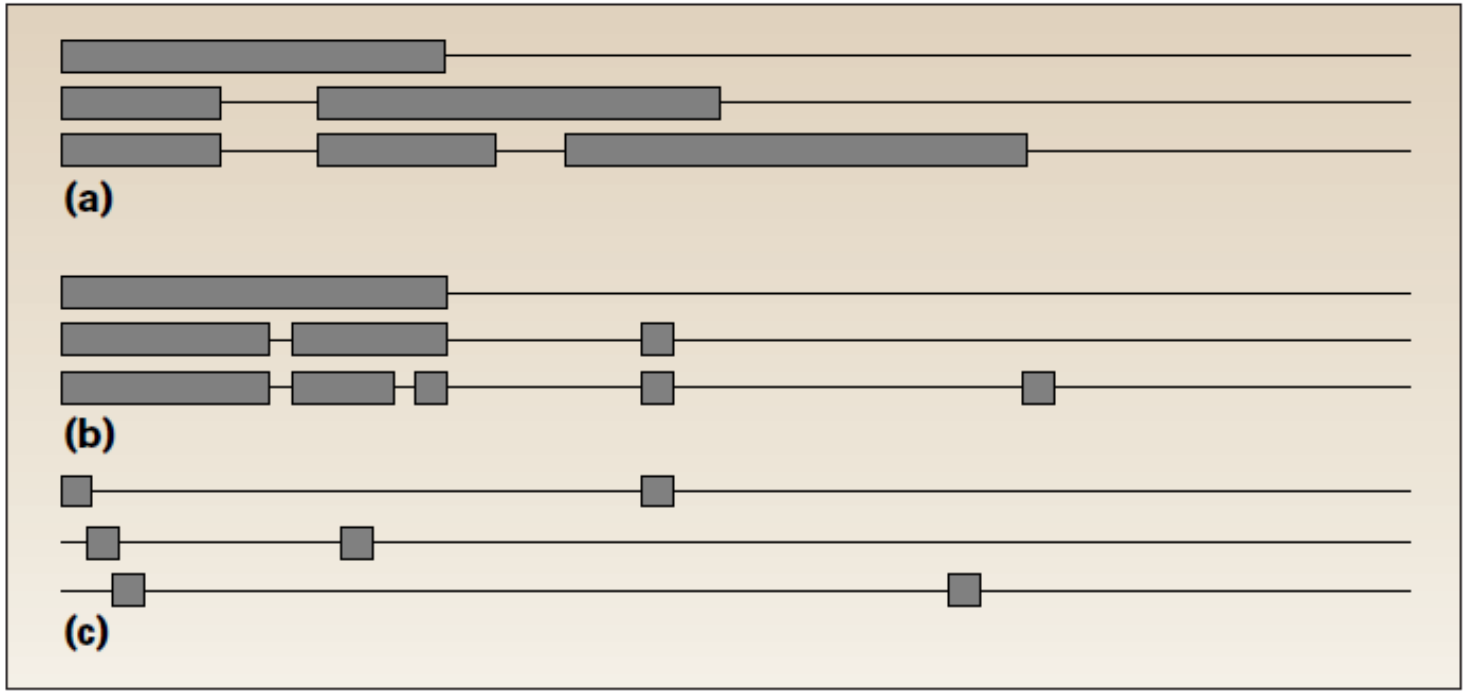


Figure 12. Three alternative methods for training SVMs: (a) Chunking, (b) Osuna's algorithm, and (c) SMO. There are three steps for each method. The horizontal thin line at every step represents the training set, while the thick boxes represent the α_i being optimized at that step. A given group of three lines corresponds to three training iterations, with the first iteration at the top.

Algoritmo Decomposição

- Problema dual

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$

- Sujeito a

$$0 \le \alpha_i \le C, \qquad i = 1,\ldots,N \qquad\qquad \sum_{i=1}^{N}\alpha_i y_i = 0$$

- Fórmula Usual em Otimização

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\alpha K \alpha^T + c^T\alpha$$

$$0 \le \alpha_i \le C, \qquad i = 1,\ldots,N \qquad \boldsymbol{\alpha}^T y = 0$$

$$K_{ij} = y_i y_j k(x_i, x_j)$$

QP1

Algoritmo Decomposição

- O tamanho do problema de otimização depende do número de exemplos de treinamento N.
- Como o tamanho da matriz K é N^2, para tarefas de aprendizado com 10000 treinamentos exemplos e mais, torna-se impossível manter K na memória.
- Muitos implementações padrões de solucionadores de QP requerem armazenamento explícito de K, o que proíbe sua aplicação.
- Uma alternativa seria recalcular K sempre que necessário. Mas isso se torna proibitivamente caro, se K for necessário com frequência.

Algoritmo Decomposição

- Uma abordagem para fazer o treinamento de SVMs tratável em problemas com muitos treinamentos exemplos é decompor o problema em uma série de tarefas menores.
- SVMlight segue a ideia de decomposição de Osuna et al. (1997). Essa decomposição divide OP1 em uma parte inativa e ativa - o chamada conjunto de trabalho.
- A principal vantagem dessa decomposição é que ela sugere algoritmos com requisitos de memória lineares com o número de exemplos de treinamento e lineares com o número de SVs.

Algoritmo Decomposição - Joachims (1999)

- Uma desvantagem potencial é que esses algoritmos podem precisar de um longo tempo de treinamento. Para resolver esse problema, este Joachims(1999) propõe um algoritmo que incorpora as seguintes idéias:
  - Um método eficiente e eficaz para selecionar o conjunto de trabalho.
  - Sucessivos encolhimento (shrinking) do problema de otimização. Explora a propriedade que muitos problemas de aprendizagem SVM possuem
    - muito menos vetores de suporte (SVs) que exemplos de treinamento.
    - muitos SVs têm um α_i no limite superior C.
  - Melhorias computacionais, como armazenamento em cache e atualizações incrementais do gradiente e os critérios de termino

Algoritmo Decomposição Geral

- Em cada iteração do OP1 são divididos em duas categorias
  - O conjunto B de variáveis livres
    - São aquelas que podem ser atualizadas na iteração atual
    - São referidas como conjunto de trabalho
    - O conjunto de trabalho tem um tamanho constante q muito menor que N
  - O conjunto NB de variáveis fixadas
    - São aquelas que são temporariamente fixadas em um valor particular

Algoritmo Decomposição Geral

- O algoritmo trabalha como segue

- Enquanto a condição de otimalidade são violadas
  - Seleciona q variáveis para o conjunto de trabalho. As N-q variáveis são fixadas em seu valor atual
  - Decompor o problema e solucionar o subproblema QP: Otimize W(α) sobre B
- Termina e retorne α

Algoritmo Decomposição Geral

- Como o algoritmo pode detectar que encontrou o valor ideal para α?
- Desde que é garantido que as condições do OP1 tenham uma Hessiana K semi-definida positiva e todas as restrições sejam lineares, OP1 é um problema de otimização convexa.
- Para esta classe de problemas as condições de Kuhn-Tucker são condições necessárias e suficientes para otimalidade.

Algoritmo Decomposição Geral

- Se as condições de otimalidade não acontecem, o algoritmo decompõe o problema OP1 e resolve problema menor de QP resultante disso.
- A decomposição garante que isso levará a progresso na função objetivo W (α), se o conjunto de trabalho B preenche alguns requisitos mínimos (ver Osuna et al. (1997b)).
- Em particular, OP1 é decomposto separando as variáveis no conjunto de trabalho B daquelas que são fixadas (NB).
- Vamos supor que y e K estejam adequadamente arranjados em relação a B e NB

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\alpha K \alpha^T + c^T \alpha$$

$$0 \le \alpha_i \le C, \qquad i = 1, \ldots, N \qquad y^T \alpha = 0$$

$$\begin{bmatrix} y_B^T & y_{NB}^T \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_{NB} \end{bmatrix} = 0$$

$$y_B^T \alpha_B + y_{NB}^T \alpha_{NB} = 0$$

$$y_B^T \alpha_B = -y_{NB}^T \alpha_{NB}$$

$$\alpha = \begin{bmatrix} \alpha_B \\ \alpha_{NB} \end{bmatrix} \qquad y = \begin{bmatrix} y_B \\ y_{NB} \end{bmatrix} \qquad \mathrm{K} = \begin{bmatrix} K_{BB} & K_{BNB} \\ K_{NBB} & K_{NBNB} \end{bmatrix}$$

$$min_{\alpha}\, W(\alpha) = \frac{1}{2}\begin{bmatrix} \alpha_B & \alpha_{NB} \end{bmatrix} \begin{bmatrix} K_{BB} & K_{BNB} \\ K_{NBB} & K_{NBNB} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_{NB} \end{bmatrix} - \begin{bmatrix} 1_B & 1_{NB} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_{NB} \end{bmatrix}$$

$$min_{\alpha}\, W(\alpha) = \frac{1}{2}\alpha_B K_{BB} \alpha_{BB+} \frac{1}{2}\alpha_{NB} K_{NBB} \alpha_{B+} \frac{1}{2}\alpha_B K_{BNB} \alpha_{NB+} \frac{1}{2}\alpha_{NBNB} K_{NBNB} \alpha_B - 1_B\, \alpha_B - 1_{NB} \alpha_{NB}$$

$$min_{\alpha}\, W(\alpha) = \frac{1}{2}\alpha_B K_{BB} \alpha_B + (\alpha_{NB} K_{NBB} - 1_B)\alpha_B + \frac{1}{2}\alpha_{NBNB} K_{NBNB} \alpha_{NB} - 1_{NB} \alpha_{NB}$$

Algoritmo Decomposição Geral

$$min_{\alpha}\, W(\alpha) = \frac{1}{2}\alpha_B K_{BB}\alpha_B + (\alpha_{NB}K_{NBB} - 1_B)\alpha_B + \frac{1}{2}\alpha_{NBNB}K_{NBNB}\alpha_{NB} - 1_{NB}\alpha_{NB}$$

$$y_B^T\alpha_B = -y_{NB}^T\alpha_{NB}$$

$$0 \leq \alpha_B \leq C$$

OP2

Algoritmo Decomposição Geral

- Como as variáveis em N são fixas, os termos constante $\frac{1}{2}\alpha_{NBNB}K_{NBNB}\alpha_{NB} - 1_{NB}\alpha_{NB}$. Eles podem ser omitidos sem alterar a solução do OP2.
- OP2 é um problema de programação quadrática semi-definido positivo que é pequeno o suficiente para ser resolvido pela maioria dos métodos de tradicionais.
- É fácil ver que mudar o α_i no conjunto de trabalho para a solução do OP2 é um passo ideal sobre B. Progresso rápido depende muito se o algoritmo pode selecionar bons conjuntos de trabalho.

Selecionando de um conjunto de trabalho

- Ao selecionar o conjunto de trabalho, é desejável selecionar um conjunto de variáveis na qual a iteração atual fará muito progresso em direção ao mínimo de $W(\alpha)$.
- A ideia encontrar uma direção viável mais íngreme possível d, que tenha apenas q elementos diferentes de zero.
- As variáveis correspondentes a esses elementos comporão o trabalho atual conjunto.
- Essa abordagem leva ao seguinte problema de otimização:

Selecionando de um conjunto de trabalho

QP3

$$min_d \quad V(d) = g(\alpha^{(t)})^T d \qquad (1)$$

$$y^T d = 0 \qquad (2)$$

$$d_i \geq 0 \qquad para\ i: \alpha_i = 0 \qquad (3)$$

$$d_i \leq 0 \qquad para\ i: \alpha_i = C \qquad (4)$$

$$-1 \leq d \leq 1 \qquad (5)$$

$$\{d_i, d_i \neq 0\} = q \qquad (6)$$

- Eq 1 afirma que é necessária uma direção de descida.
- As restrições 2, 3 e 4 garantem que a direção da descida é projetada ao longo da restrição de igualdade e obedece àrestrições vinculadas ativas.
- A restrição 5 normaliza o vetor descendente para tornar o problema de otimização bem definido.
- Finalmente, a última restrição 6 afirma que a direção da descida deve envolver apenas q variáveis.
- As variáveis diferente de zero d_i estão incluídos no conjunto de trabalho B.
- Dessa forma, selecionamos o conjunto de trabalho com o direção de descida mais íngreme e viável.

Convergência

- A estratégia de seleção, as condições de otimização e a decomposição juntas especifica o algoritmo de otimização.
- Um requisito mínimo que esse algoritmo deve satisfazer é que
  - termina somente quando a solução ideal é encontrada
  - se não estiver na solução idela, dá um passo em direção à melhor
- O primeiro requisito pode ser facilmente atendido verificando-se (condições necessárias e suficiente de otimalidade) em cada iteração.
- Para responde a segunda, vamos supor que o $\alpha(t)$ não não é ótima. Logo, a estratégia de seleção para o conjunto de trabalho retorna um problema de otimização do tipo OP2. A construção desse problema de otimização mostra que existe um d que é uma direção viável para a descida. De acordo com os resultados de Zoutendijk (1970), o problema OP2 atual não é o ideal. Portanto, otimizar o OP2 levará a um valor mais baixo para a função objetivo do OP2. Uma vez que a solução de OP2 também é viável para OP1 devido à decomposição, também obtemos um valor mais baixo para OP1. Isso significa que temos uma redução na função objetiva de OP1 a cada iteração.

Como solucionar o OP3

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \quad \Rightarrow \quad \frac{\partial W(\alpha)}{\partial \alpha_i} = \sum_{j=1}^{N}\alpha_j y_i y_j K(x_i,x_j) - 1$$

$$f(x) = \sum_{i=1}^{N}\alpha_i y_i K(x_i,x) + b \quad \Rightarrow \quad f(x) = \sum_{i=1}^{N}\alpha_i y_i K(x_i,x)$$

$$\frac{\partial W(\alpha)}{\partial \alpha_i} = y_i(f(x_i) - y_i)$$

Como solucionar o OP3

- A solução para o OP3 é fácil de calcular usando uma estratégia simples. Considere direção igual

$$dir_i = y_i \frac{\partial W(\alpha)}{\partial \alpha_i} = f(x_i) - y_i$$

- Classifique todos os α_i de acordo com dir_i em ordem decrescente. Além disso, vamos exigir que q é um número par. Escolha sucessivamente os elementos q /2 no topo da lista para os quais $0<\alpha_i(t)<C$, ou di =-yi que obedece a equação 3 e 4.
- Resumidamente se $y_i=1$, escolha $0<\alpha_i(t) \leq C$, se $y_i=-1$, escolha $0 \leq \alpha_i(t) <C$
- Da mesma forma, escolha o q /2 elementos do final da lista para os quais $0<\alpha_i(t)<C$, ou di =yi que obedece a equação 3 e 4
- Resumidamente se $y_i=1$, escolha $0 \leq \alpha_i(t) <C$, se $y_i=-1$, escolha $0 <\alpha_i(t) \leq C$

Código (Matlab)

- Entrada
  - X – matriz de entrada (Nxm)
  - Y – matriz de saída (Nx1)
  - N – número de amostras
  - qpsize– tamanho do problema QP
  - chsize – tamanho do conjunto de chunking

```
% Encontra os indices dos exemplos da classe +1 and -1
class1 = logical(uint8(Y>=0));
class0 = logical(uint8(Y<0));
% Usa o mesmo limite superior para todos os exemplo
C = repmat(C, [N 1]);
% Verifica se o parâmetro qpsize é menor que N
QPsize = min(N, qpsize);
% Cria um vetor SVMout, que contém a saída da função de decisão do SVM para cada exemplo
SVMout = zeros(N, 1);
% Saída desejada Y deve ser +1 e -1
Y(class1) = 1;
Y(class0) = -1;
```

Código (Matlab)

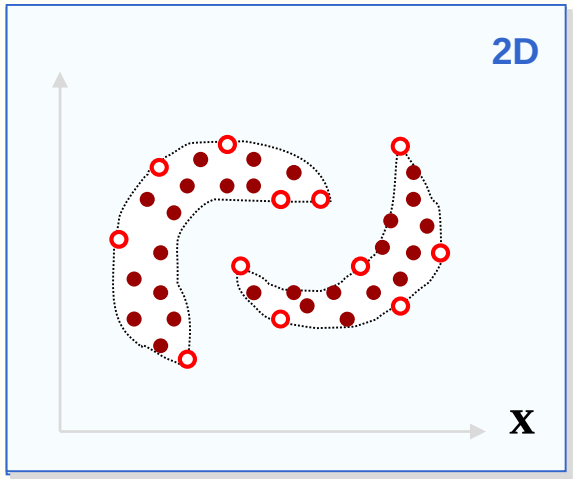
```
if ~any(alpha0),
  alpha = zeros([N 1]);
   randomWS = 1;  % Se alpha inicial é zero, o working set será escolhido randomicamente
else
  randomWS = 0;
end
if length(find(Y>0))==N % Exemplos somente da classe positiva
            bias = 1;
            alpha = zeros([N 1]);
            return;
elseif length(find(Y<0))==N, %Exemplos somente da  classe negativa
            bias = -1;
            alpha = zeros([N 1]);
             return;
end
```

Código (Matlab)

- while 1,
  - % Passo 1: Determine os vetores suportes
  - % Passo 2: Encontra a saída do SVM para todos os exempos
  - % Passo 3: Cálcula o bias da função de decisão
  - % Passo 4: Cálcula os valores das condições de Karush-Kuhn-Tucker do QP. Se nenhuma violação destas condições, solução ótima tem sido encontrada, podemos finalizar
  - % Passo 5: Determine um novo conjunto de trabalho
  - % Passo 6: Seleciona o conjunto de trabalho. Objetivo é selecionar numero igual de exemplos do conjuntos (QPsize/2, QPsize)
  - % Passo 7: Determina a parte linear do QP
  - % Passo 8: Soluciona o QP
- end

Support Vector Clustering

Prof. Dr. Clodoaldo A M Lima

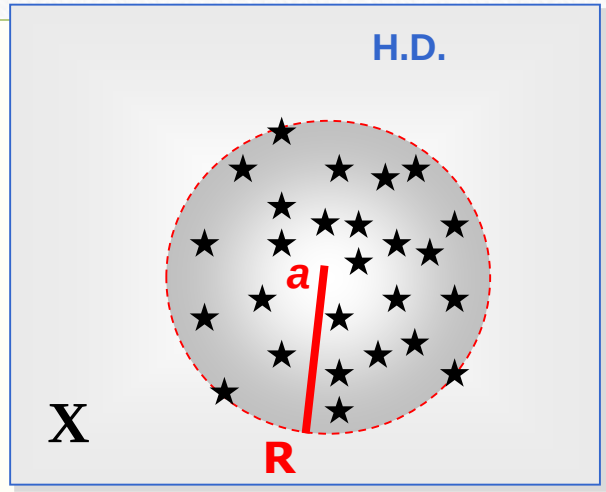


$$\Phi(\cdot)$$

$$\mathbf{X}_i = \Phi(\mathbf{x}_i)$$

$$\mathbf{X} = \Phi(\mathbf{x})$$

$$\mathbf{X}_i \cdot \mathbf{X} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})$$



$$R^2(x) = \Phi(x)\cdot\Phi(x) - 2\sum_{i\in S}\alpha_i\Phi(x)\cdot\Phi(x_i) + \sum_{i,j\in S}\alpha_i\alpha_j\Phi(x_i)\cdot\Phi(x_j)$$

$$R^2(x) = X\cdot X - 2\sum_{i\in S}\alpha_i X\cdot X_i + \sum_{i,j\in S}\alpha_i\alpha_j X_i\cdot X_j$$

$K(\cdot,\cdot)$

$$\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}) = K(\mathbf{x}_i,\mathbf{x})$$

Mercer's Condition

$$R^2(x) = K(x,x) - 2\sum_{i\in S}\alpha_i K(x,x_i) + \sum_{i,j\in S}\alpha_i\alpha_j K(x_i,x_j)$$

Treinamento

Há dois passos principais no Treinamento do SVC

Step 1 –Treinamento SVC

Este é responsável para o treinamendo do modelo

Step 2 – Rotulação Cluster

Este checa a conectividade para cada par de pontos baseado no critério corte obtido do SVC treinado

Complexidade é O(n^2m), onde *n* é o número de pontos, e m<<n é o número de pontos amostrados sobre vertice que é usualmente constante entre 10 e 20

Passo 1 – Treinamento SVM

- Problema Primal

$$Min \quad R^2 + C\sum_{i=1}^{N}\xi_i$$

s.a

$$\left\|\Phi(x_i) - a\right\|^2 \leq R^2 + \xi_i$$

- Problema Dual

$$\underset{\alpha}{Min} \quad \sum_{i=1}^{N}\alpha_i K(x_i, x_i) - \sum_{i,j=1}^{N}\alpha_i\alpha_j K(x_i, x_j)$$

s.a

$$\sum_{i=1}^{N}\alpha_i = 1$$

$$0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, N$$

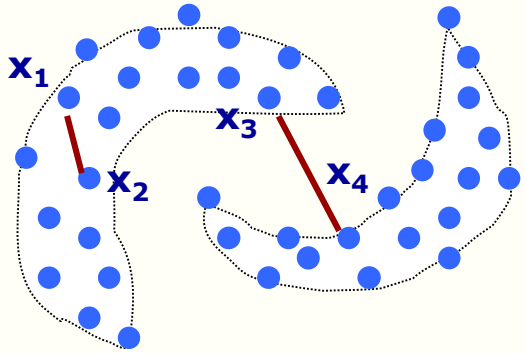
Passo 2 – Rotulação dos Cluster

- Duas Abordagens
  - a)Checar o link entre todos os pairs pontos
  - b)Metodo Heuristico
    - Ben–Hur et al (2001)
      - Checar o link entre todos pairs e support vector
    - Jianhua et al (2002)
      - Proximity Graph
        - ✓ Delaunay Diagrams
        - ✓ Minimum Spanning Trees
        - ✓ Nearest Neighboors

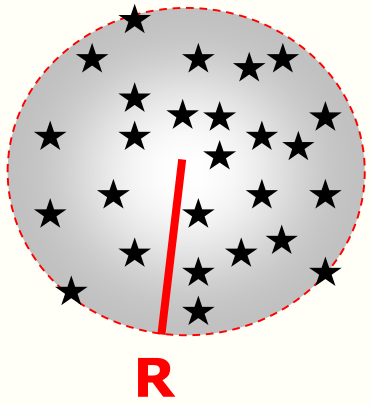
(a) All pairs

Case 1

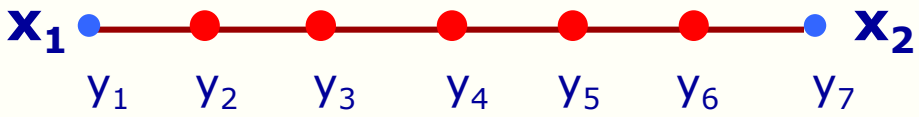
sample vs sample



2D



H.D.

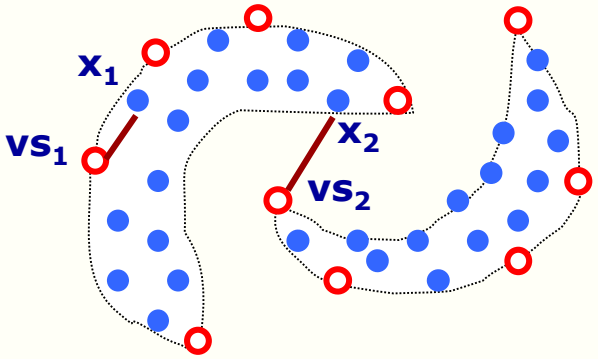


A_{ij} { 1 se $D(y_j) <= R$; 0 caso contrario

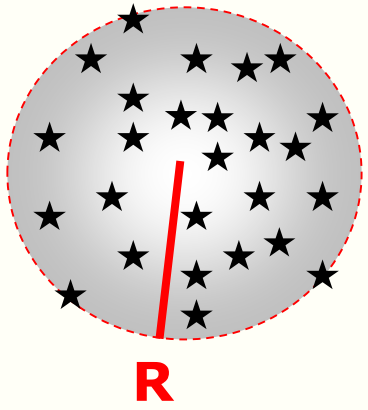
A_{ij}

	x_1	x_2	x_3	x_4	. . .
x_1	-	1	1	0	…
x_2	1	-	0	0	…
x_3	1	0	-	0	…
x_4	0	0	0	-	…
. . .	…	…	…	…	-

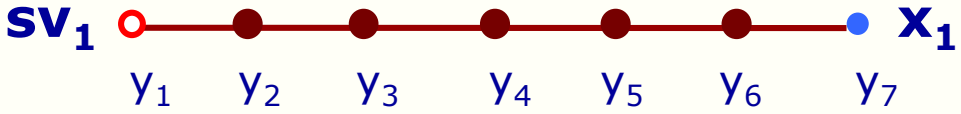
(b) Ben-Hur (2001)



2D



H.D.



$$A_{ij} \begin{cases} 1 & \text{se } D(y_j) <= R \\ 0 & \text{caso contrario} \end{cases}$$

Case 2

Support vector vs sample

A_{ij}

	x_1	x_2	x_3	x_4	. . .
vs_1	1	0	1	…	…
vs_2	0	0	…	…	…
vs_3	1	…	-	…	…
vs_4	0	…	…	-	…
. . .	…	…	…	…	-