Support Vector Machines (SVMs)

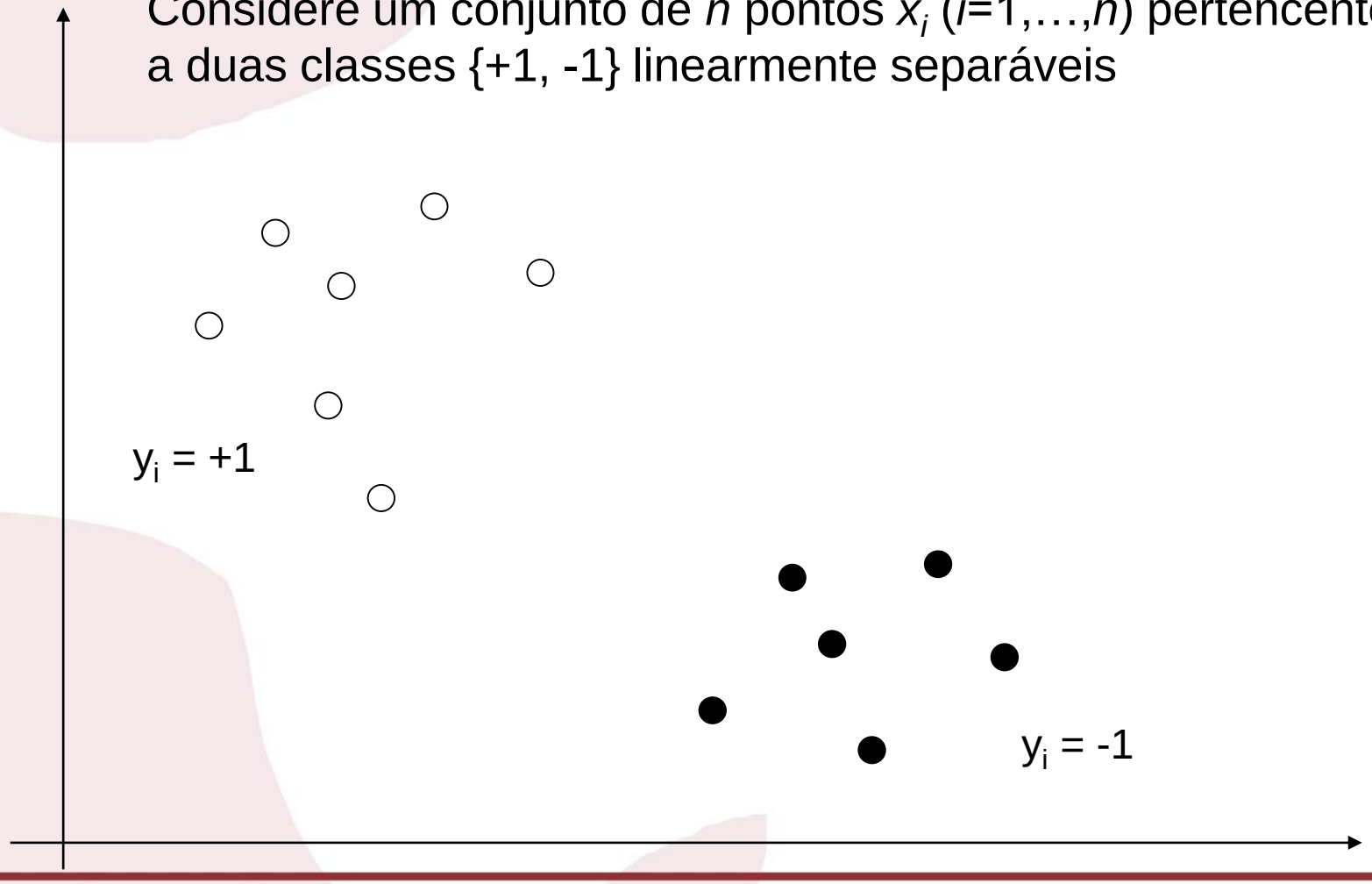
Cleber Zanchettin
UFPE - Universidade Federal de Pernambuco
CIn - Centro de Informática

Introdução

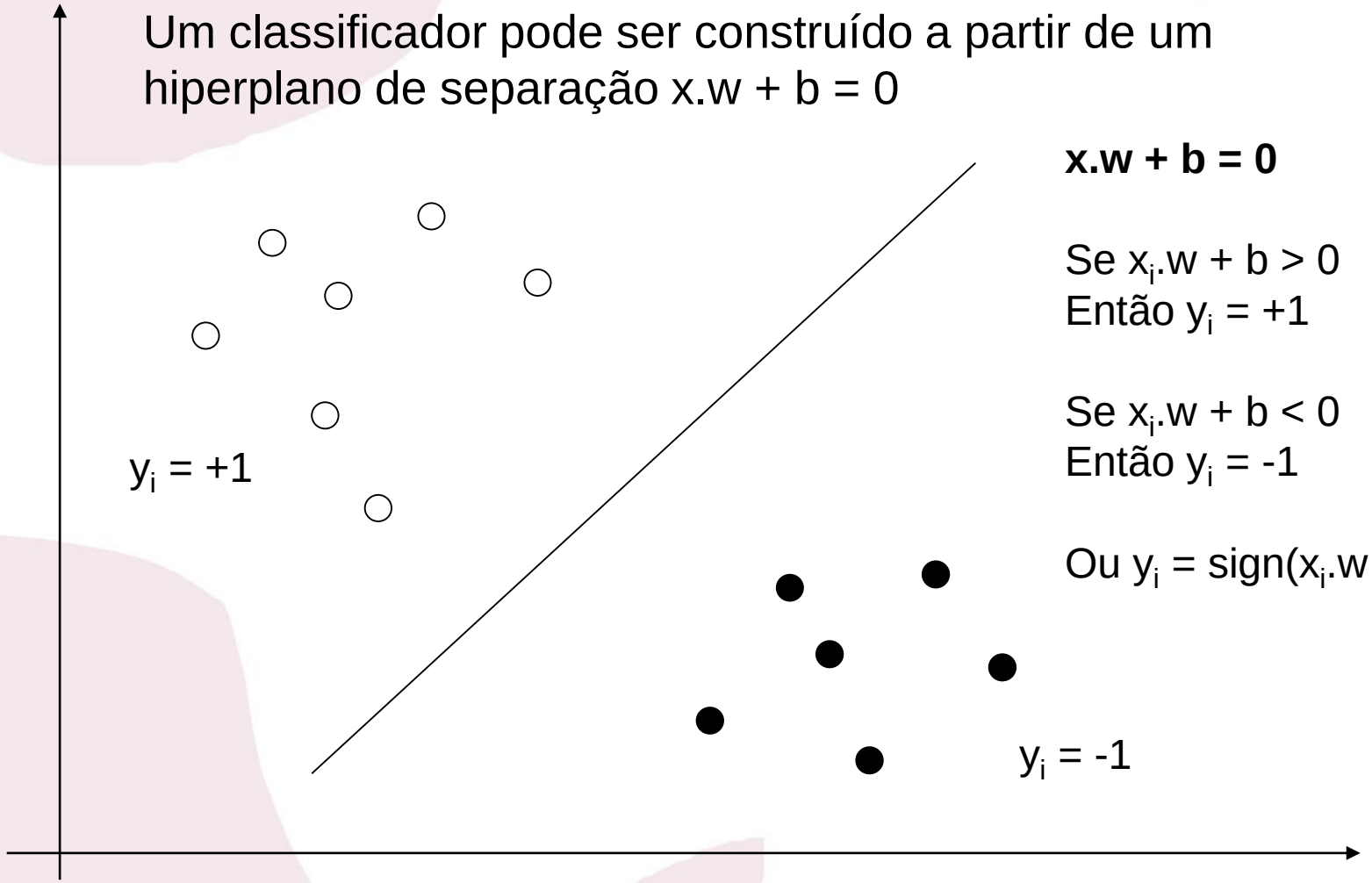
- 1968: base matemática
  - Teoria de Lagrange
- [Vapnik et al, 1992] Primeiro artigo
- [Vapnik et al, 1998] Definição detalhada
- Última década
  - Série de artigos com aplicações de SVM
  - Série de artigos com otimizações de SVM

Motivação

Considere um conjunto de *n* pontos x_i (*i*=1,…,*n*) pertencentes a duas classes {+1, -1} linearmente separáveis



Um classificador pode ser construído a partir de um hiperplano de separação x.w + b = 0



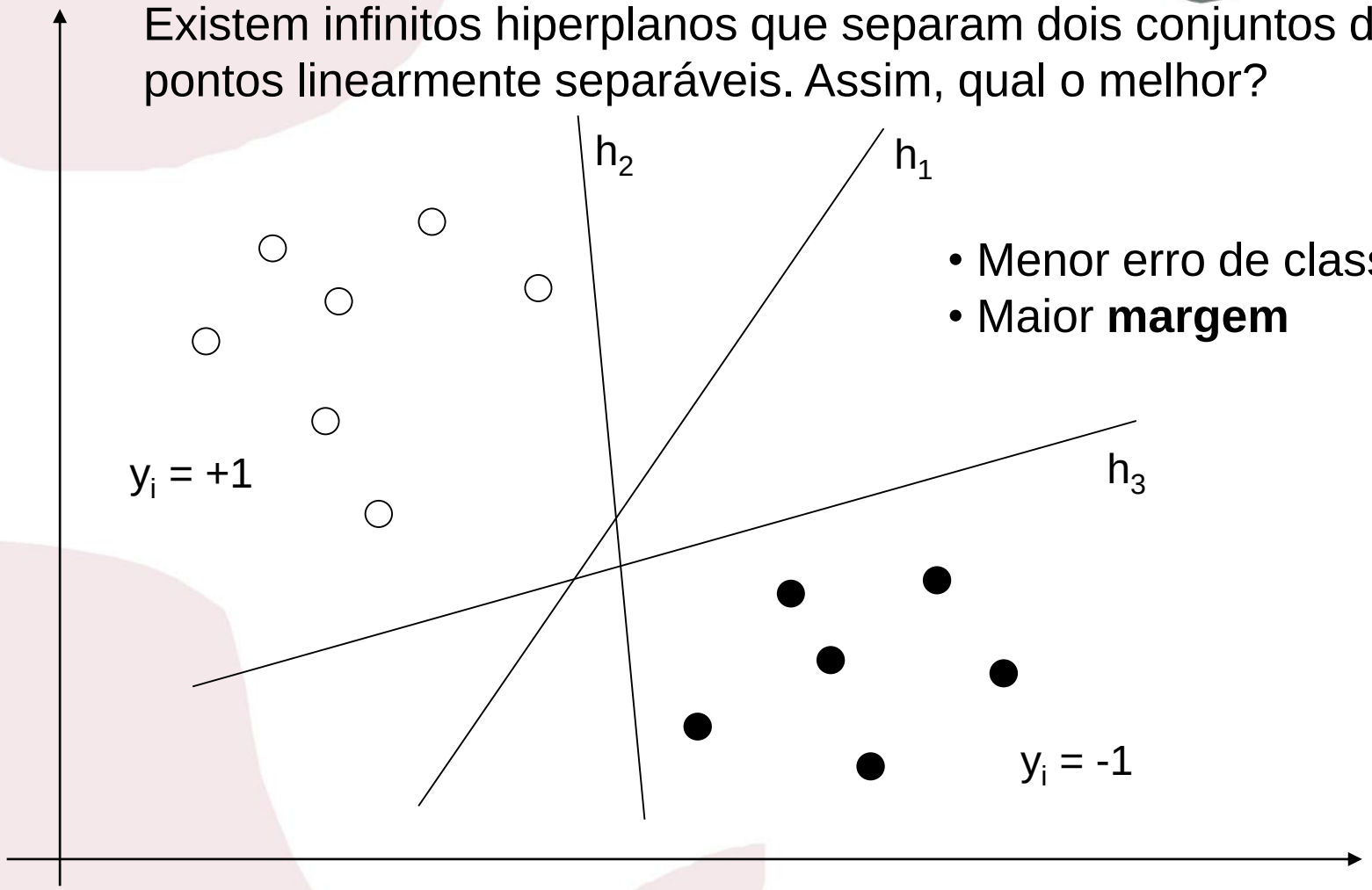
x.w + b = 0

Se $x_i.w + b > 0$
Então $y_i = +1$

Se $x_i.w + b < 0$
Então $y_i = -1$

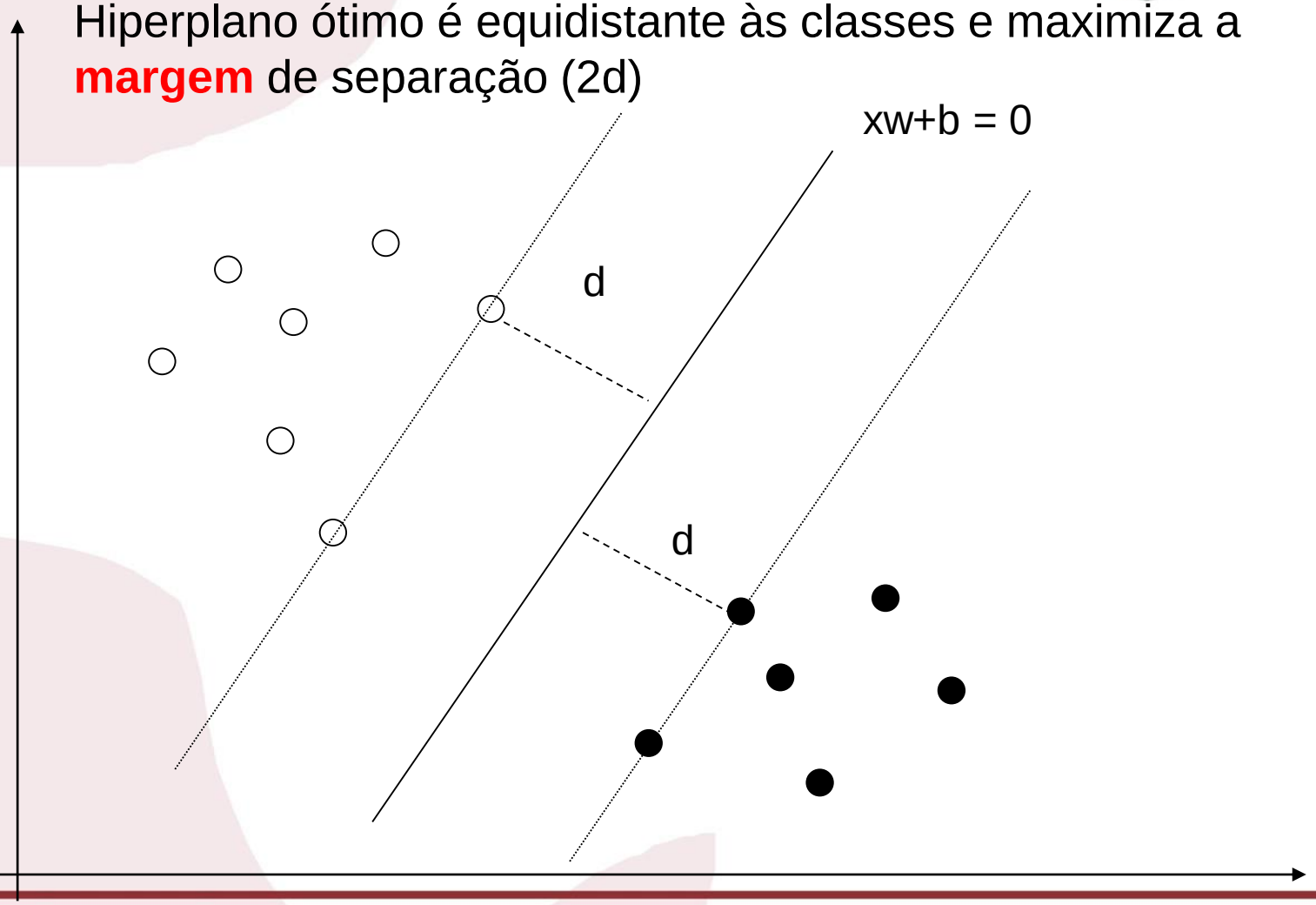
Ou $y_i = sign(x_i.w + b)$

Existem infinitos hiperplanos que separam dois conjuntos de pontos linearmente separáveis. Assim, qual o melhor?

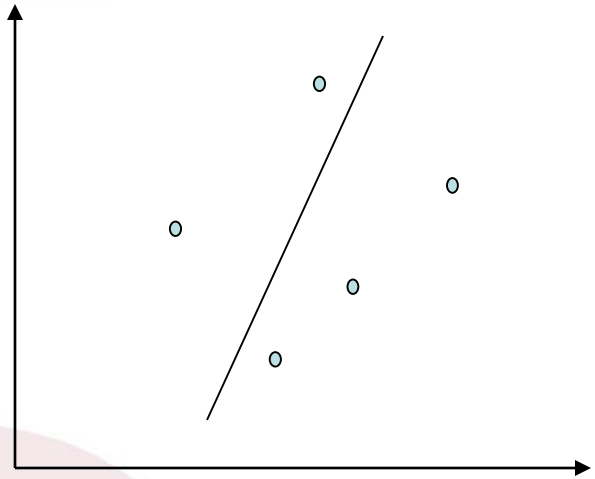


- Menor erro de classificação
- Maior **margem**

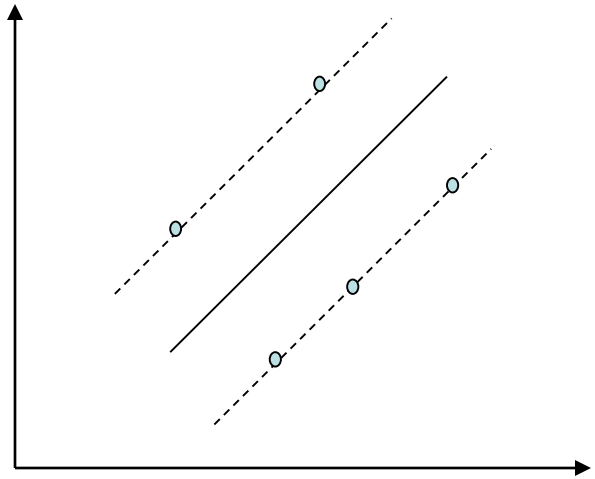
Hiperplano ótimo é equidistante às classes e maximiza a **margem** de separação (2d)



Comparação

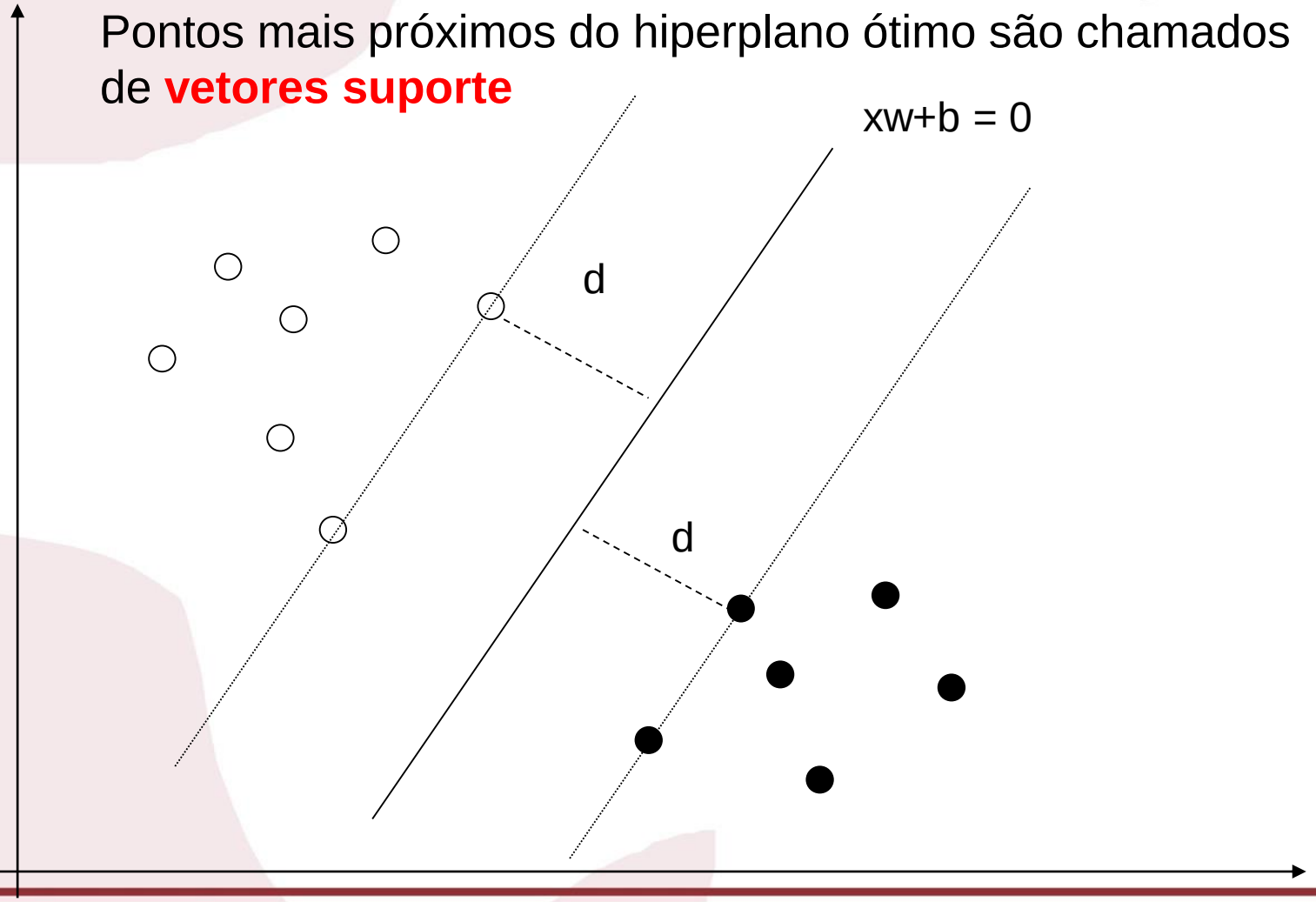


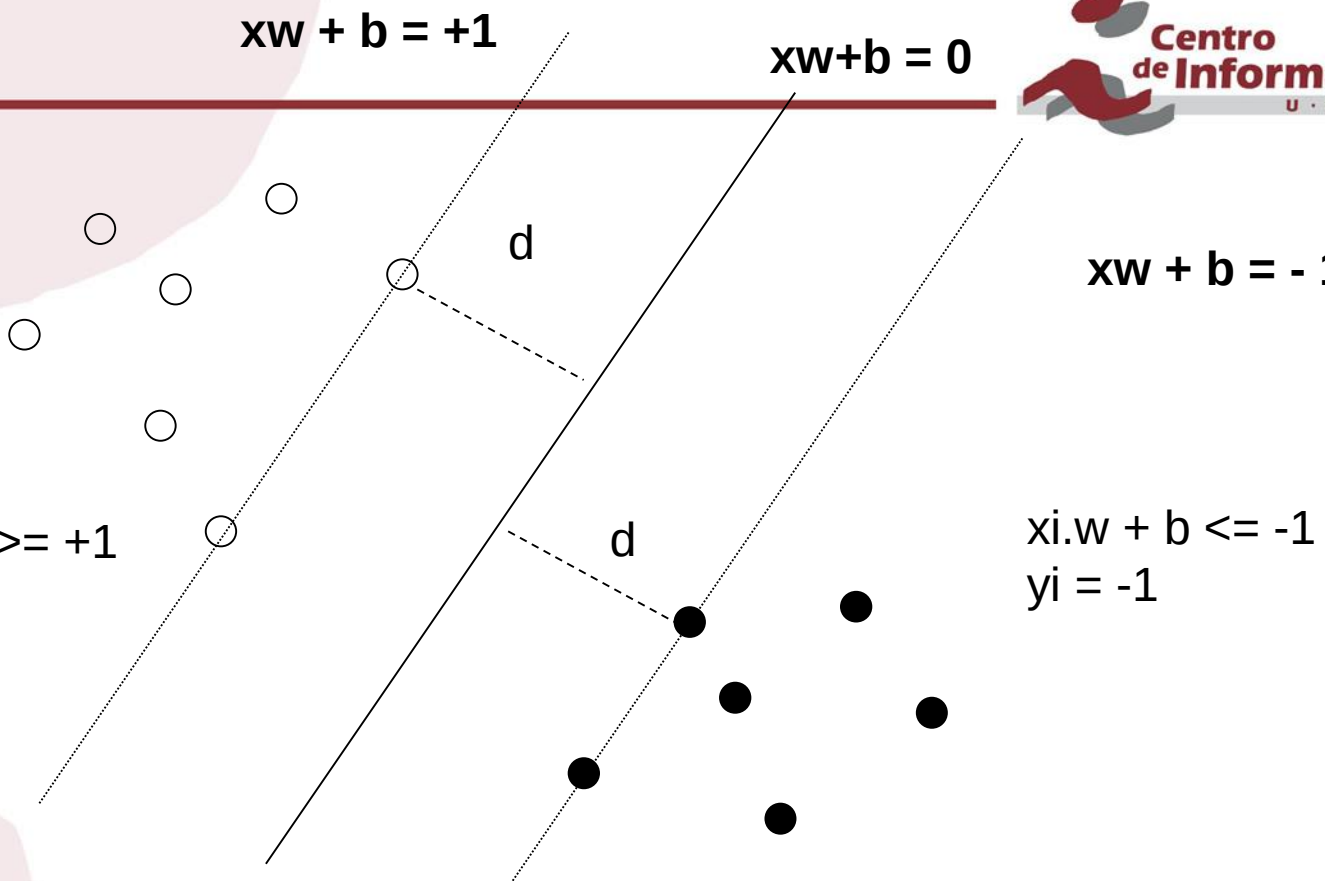
Rede Neural Linear



SVM Linear

Pontos mais próximos do hiperplano ótimo são chamados de **vetores suporte**





Hiperplanos superior e inferior podem ser reescalonados para:
xw + b = +1 e **xw + b = -1**

Margem 2d é calculada como: $\frac{2}{||w||}$

Considerando:

- Dados de treinamento
  - Tuplas no formato (x_1, x_2, ..., x_n, y)
    - Atributos X_i
    - Classe Y (+1, -1)

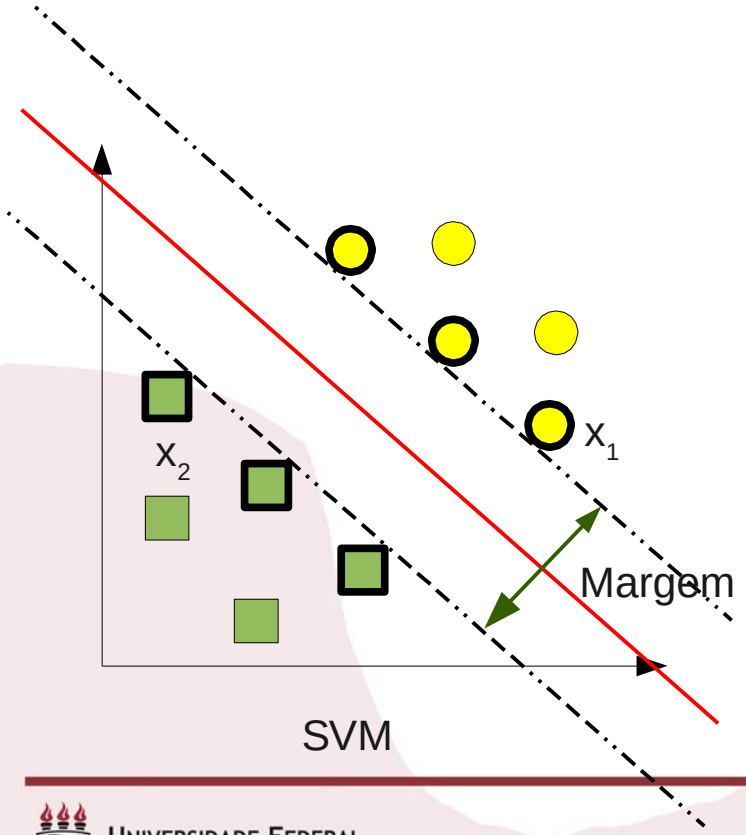
- Conjunto dito linearmente separável, se existir um hiperplano H (no espaço de entrada) que separe as tuplas de classes diferentes

- Determinar os **vetores de suporte**

- Encontrar o **hiperplano** ótimo
  - Com maior **margem**

- Considerando os vetores de suporte x_1 e x_2 temos que a distância entre um hiperplano que toca todos os vetores de suporte do lado de x_1 respeitando:

$$\mathbf{w} \cdot \mathbf{x}_1 + b = +1$$



Enquanto todos os vetores de suporte do lado de x_2 respeitam:

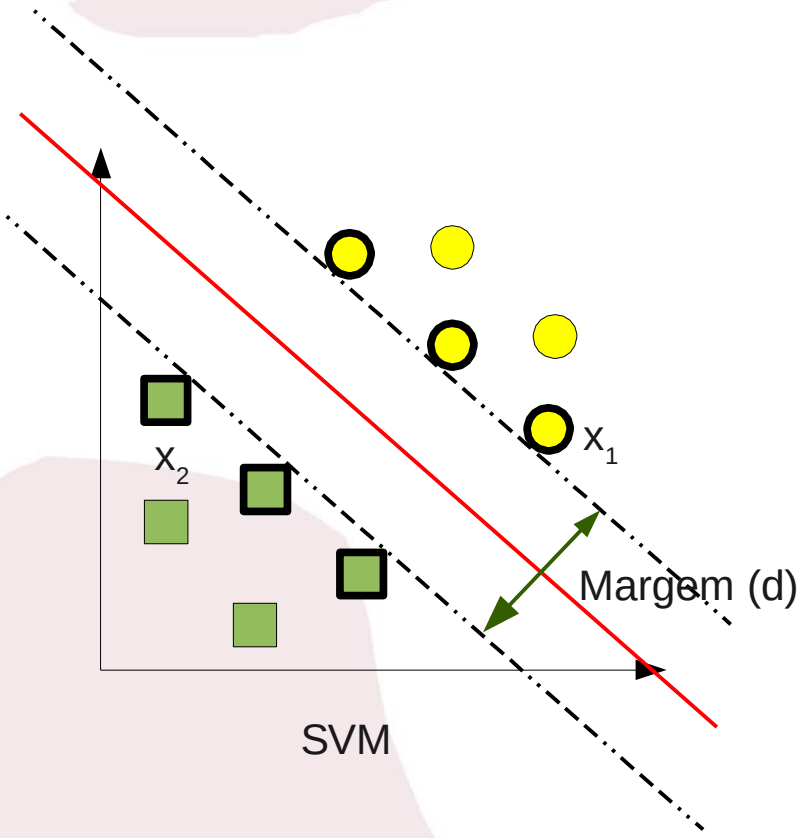
$$\mathbf{w} \cdot \mathbf{x}_2 + b = -1$$

Assim, a diferença entre ambas as funções define a ***margem***:

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

Logo, a diferença entre x_1 e x_2 é dada por:

$$\mathbf{x}_1 - \mathbf{x}_2 = \frac{2}{\mathbf{w}}$$



Como buscamos pela máxima margem ou distância projetada entre x_1 e x_2, buscamos maximizar:

$$d = \mathbf{x_1} - \mathbf{x_2} = \frac{\mathbf{2}}{|| \mathbf{w} ||}$$

Sendo a ***distância mínima*** entre o hiperplano separador e os dados de treinamento dada por:

$$\frac{1}{|| \mathbf{w} ||}$$

Logo podemos maximizar esse termo acima ou minimizar o termo abaixo:

$$|| \mathbf{w} ||$$

A 1D Example

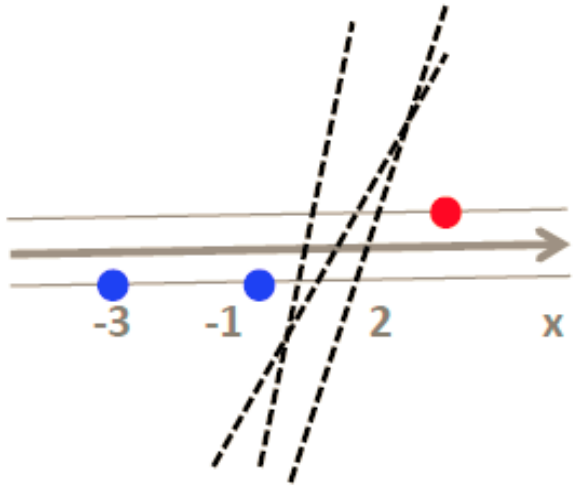
- Suppose we have three data points

  x = -3, y = -1

  x = -1, y = -1

  x = 2, y = 1

- Many separating perceptrons, T[ax+b]
  - Anything with ax+b = 0 between -1 and 2



A 1D Example

- Suppose we have three data points

  x = -3, y = -1

  x = -1, y = -1

  x = 2, y = 1

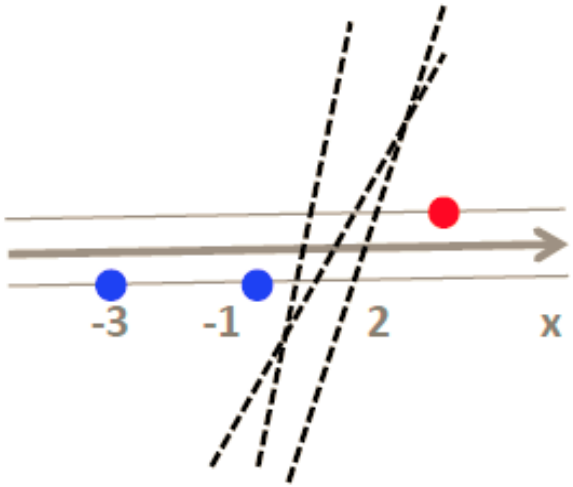
- Many separating perceptrons, T[ax+b]
  - Anything with ax+b = 0 between -1 and 2

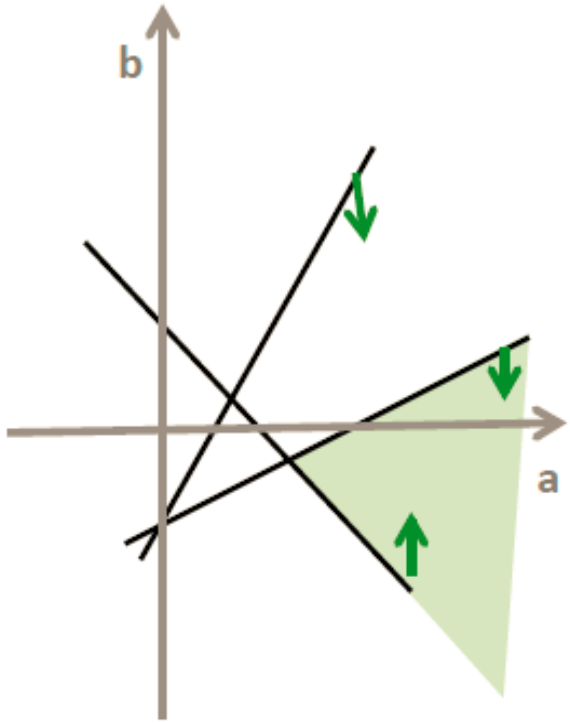
- We can write the margin constraints

  a (-3) + b < -1 => b < 3a - 1

  a (-1) + b < -1 => b < a - 1

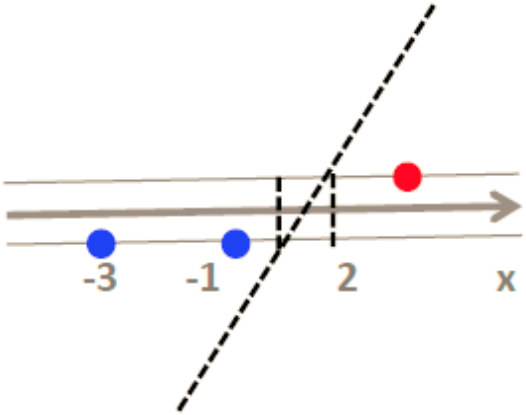
  a (2) + b > +1 => b > -2a + 1

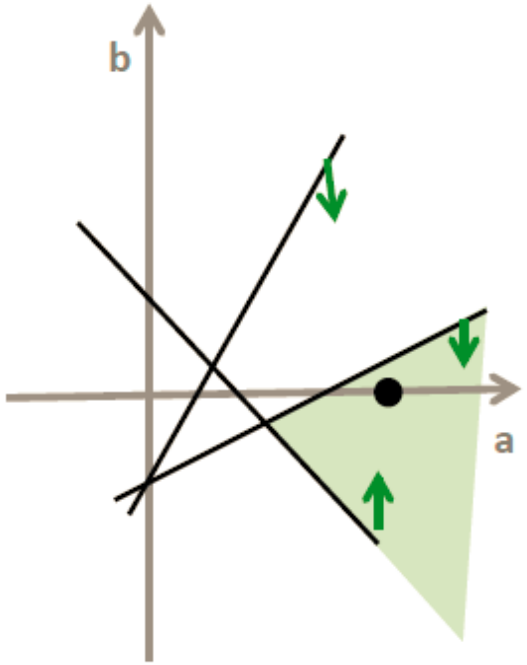




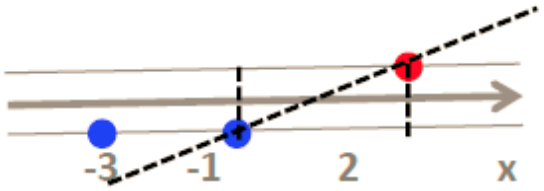
A 1D Example

- Suppose we have three data points
  - x = -3, y = -1
  - x = -1, y = -1
  - x = 1, y = 1
- Many separating perceptrons, T[ax+b]
  - Anything with ax+b = 0 between -1 and 2
- We can write the margin constraints
  - a (-3) + b < -1 => b < 3a - 1
  - a (-1) + b < -1 => b < a - 1
  - a (2) + b > +1 => b > -2a + 1
- Ex: a = 1, b = 0

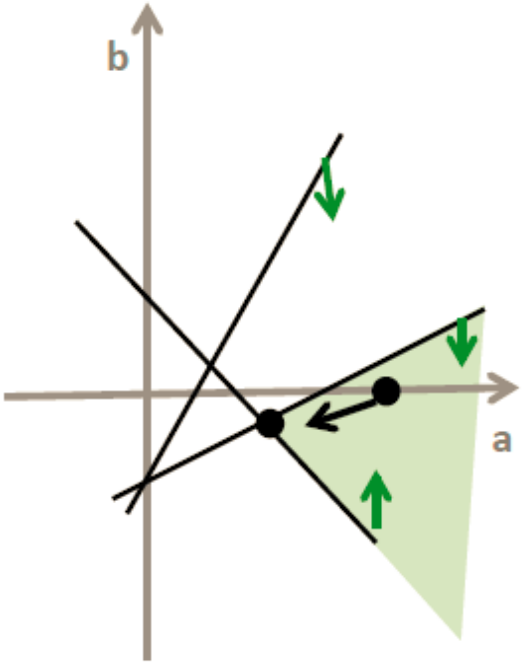




A 1D Example



- Suppose we have three data points
  - x = -3, y = -1
  - x = -1, y = -1
  - x = 1, y = 1
- Many separating perceptrons, T[ax+b]
  - Anything with ax+b = 0 between -1 and 2
- We can write the margin constraints
  - a (-3) + b < -1 => b < 3a - 1
  - a (-1) + b < -1 => b < a - 1
  - a (2) + b > +1 => b > -2a + 1
- Ex: a = 1, b = 0
- Minimize ||a|| => a = .66, b = -.33
  - Two data on the margin; constraints "tight"



Hiperplano ótimo

Margem = 2 / ||w||

- É aquele que possui maior margem
- É aquele que possui menor ||w||
- Determinação do hiperplano
  - Problema de otimização restrita
  - Minimizar uma função de custo (produto interno) sujeito a restrições
  - Multiplicadores de Lagrange

Problema

Maximizar: $\frac{2}{||w||}$

Sujeito a:

$x_i.w + b >= +1$ se $y_i = +1$

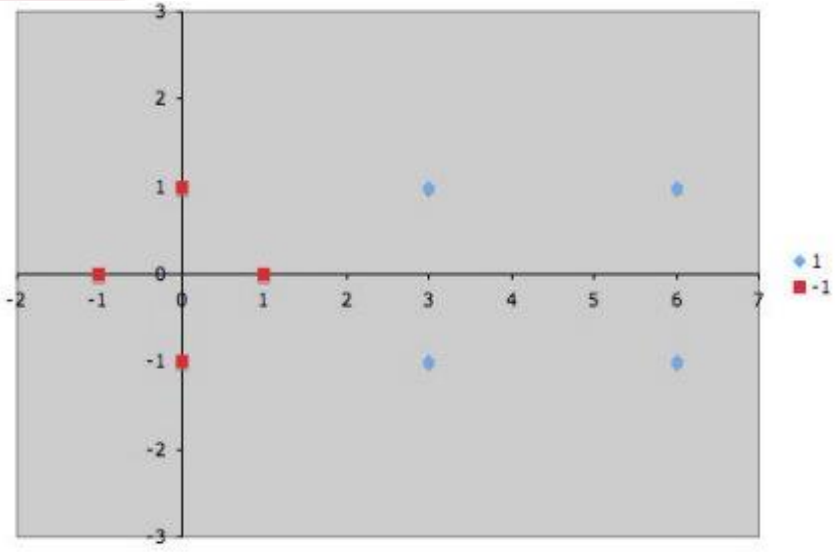
$x_i.w + b <= -1$ se $y_i = -1$

Ou de forma mais conveniente:

Minimizar: $\frac{||w||^2}{2}$

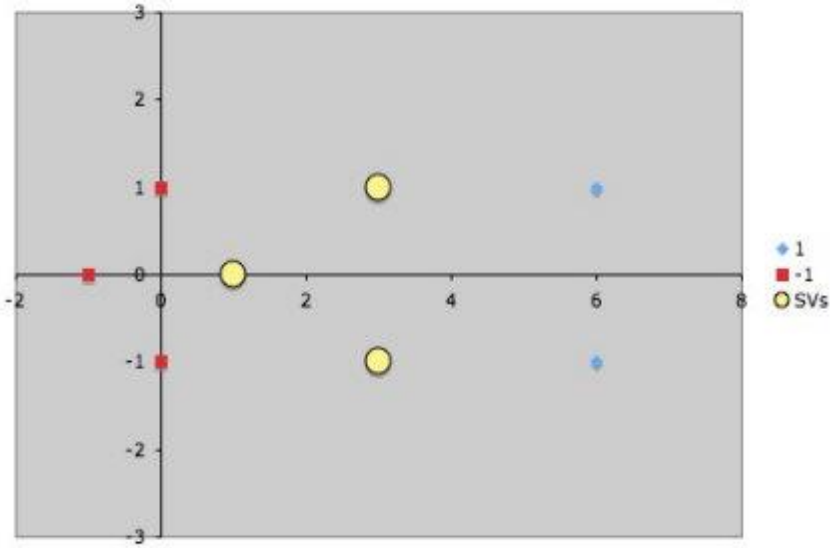
Sujeito a: $y_i (x_i.w + b) – 1 >= 0$

Exemplo



- -1, 0, -1
- 0, -1, -1
- 0, 1, -1
- 1, 0, -1
- 3, -1, +1
- 3, 1, +1
- 6, -1, +1
- 6, 1, +1

Exemplo



- -1, 0, -1
- 0, -1, -1
- 0, 1, -1
- **1, 0, -1**
- **3, -1, +1**
- **3, 1, +1**
- 6, -1, +1
- 6, 1, +1

Exemplo

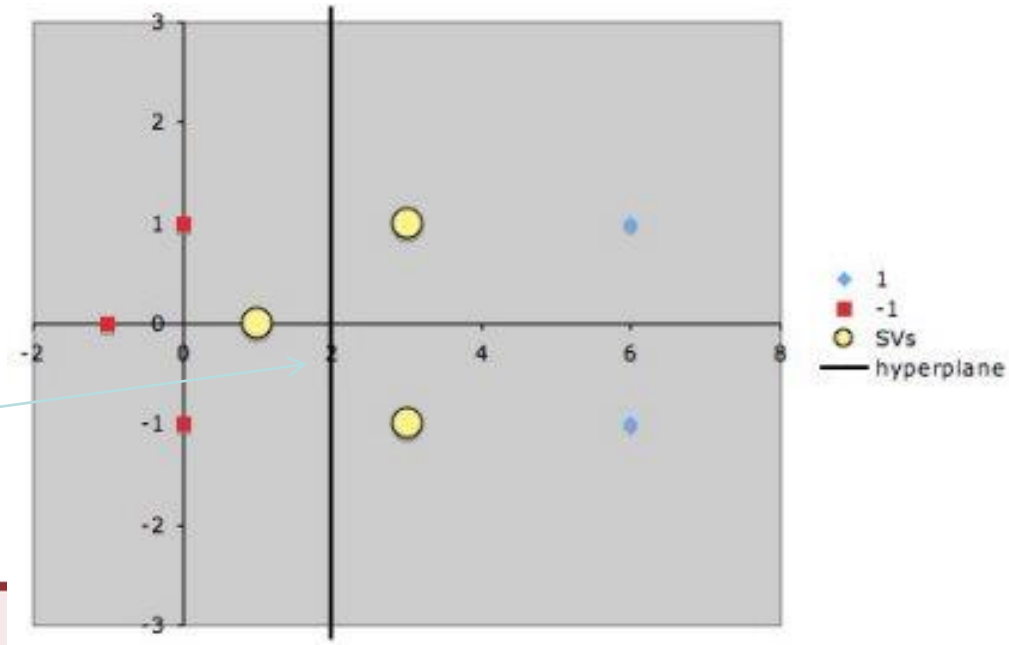
H_1: $w \cdot x + b = 1$
H_2: $w \cdot x + b = -1$

$w_1x_1 + w_2x_2 + b = -1$
$1w_1 + 0w_2 + b = -1$
→ $b = -1 - w_1$

(1, 0) → -1
(3, -1) → +1
(3, 1) → +1

$w_1x_1 + w_2x_2 + b = 1$
$3w_1 - 1w_2 + b = 1$
→ $w_2 = 3w_1 - 1 - w_1 - 1$
→ $w_2 = 2w_1 - 2$
$3w_1 + 1w_2 + b = 1$
→ $3w_1 + 2w_1 - 2 - 1 - w_1 = 1$
→ $w_1 = 1$
→ $b = -2$
→ $w_2 = 0$

$(1, 0) \cdot x - 2 = 0$
$X_1 = 2$



Exemplo

H: (1, 0) · x - 2 = 0

H: x_1 - 2 = 0

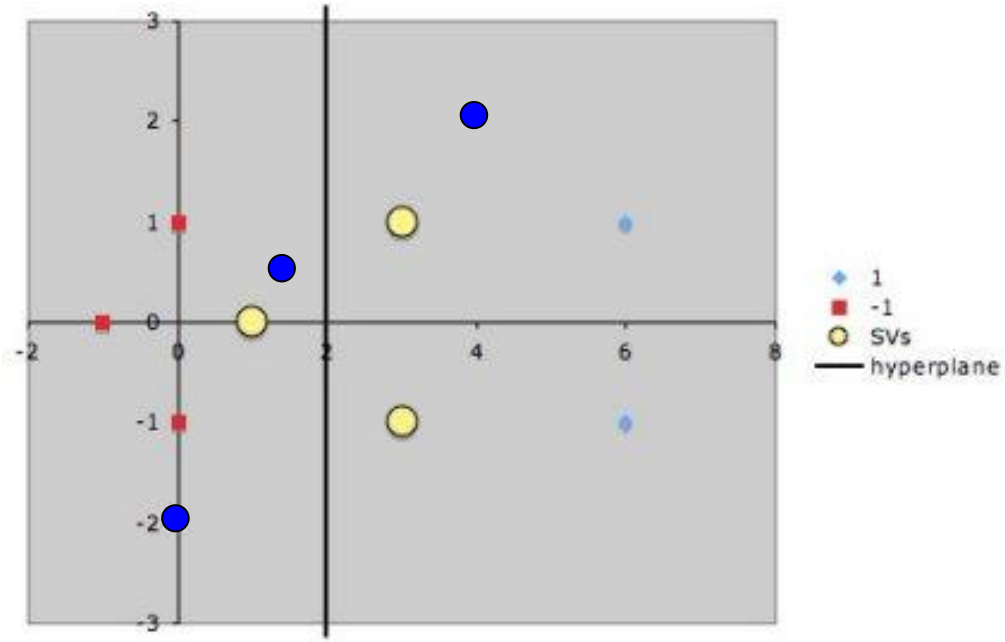
Dados de Teste

(4, 2), (1.5, 0.5), (0, -2)

4 - 2 = 2 [+1]

1.5 - 2 = -0.5 [-1]

0 - 2 = -2 [-1]



Como encontrar os vetores de suporte?

- Multiplicadores de Lagrange

Minimizar: $\frac{\|w\|^2}{2}$

Sujeito a: $y_i (x_i.w + b) - 1 >= 0$

Minimizar: $L_P = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i * [y_i (x_i.w + b) - 1]$

Multiplicador de Langrange $\alpha i >= 0$
(pode ser visto como a “força” da i-ésima restrição)

Multiplicadores de Lagrange

• Método dos Multiplicadores de Lagrange: Empregado para resolver problemas de extremos sujeitos a restrições de igualdade.

• Seja o problema a seguir:

$$\max \text{ (min) } f(\mathbf{x})$$

$$\text{s.a.} \quad g_i(\mathbf{x}) = 0, \qquad i = 1, \ldots, N$$

onde f e g_i (i=1,..,N) são funções reais de n ($n > N$) variáveis que se assumem duas vezes diferenciáveis num determinado conjunto D.

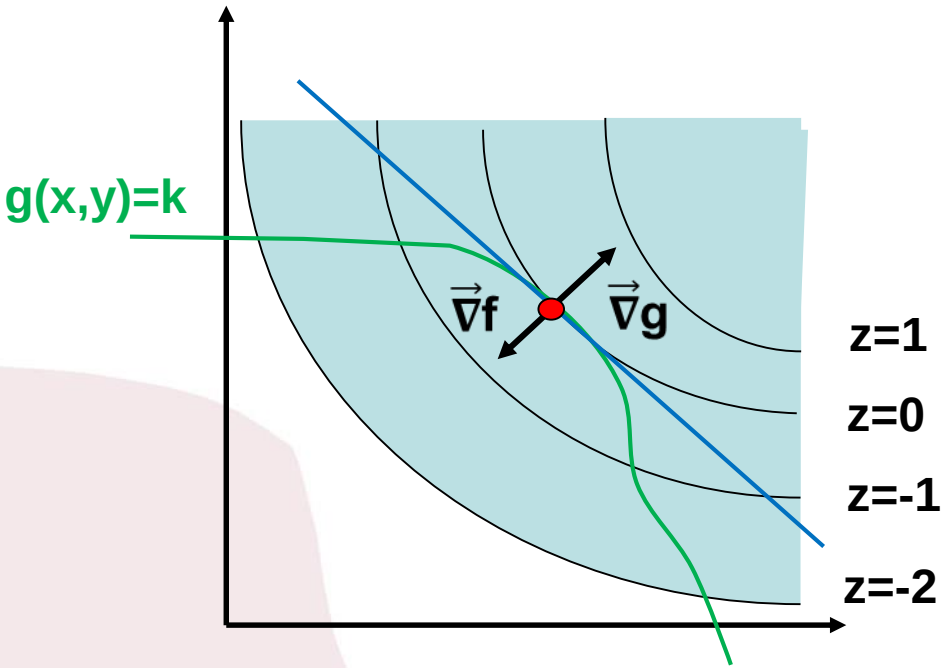
• Chama-se função de Lagrange ou lagrangiano à função:

$$L(\mathbf{x}, ?) = f(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i g_i(\mathbf{x})$$

Multiplicadores de lagrange

Max/Min **f(x,y)=z**
s.a. g(x,y)=k

$$\vec{\nabla} f = \lambda \vec{\nabla} g$$



Multiplicadores de Lagrange

Minimizar: $L_P = \frac{1}{2}||w||^2 - \sum_i \alpha_i * [\, y_i (x_i.w + b) - 1 \,]$

L_P deve ser minimizada com respeito a w e b, e maximizada com respeito a α_i, ou seja:

$$\max_{\alpha} \left\{ \min_{w,b} \left\{ L_P(w,b,\alpha) \right\} \right\}$$

Multiplicadores de Lagrange

$$\max_{\alpha} \left\{ \min_{w,b} \left\{ L_P(w,b,\alpha) \right\} \right\}$$

$$\frac{\partial L_P}{\partial w} = 0 \longrightarrow w = \sum_i \alpha_i x_i y_i \quad \textbf{(1)}$$

$$\frac{\partial L_P}{\partial b} = 0 \longrightarrow \sum_i \alpha_i y_i = 0 \quad \textbf{(2)}$$

Substituindo (1) e (2) em L_P, teremos um novo problema de otimização (ver próximo slide)

Multiplicadores de Lagrange

Substituindo (1) e (2) no problema: $\max_{\alpha} \{ \min_{w,b} \{ L_P(w,b,\alpha) \} \}$

$\Downarrow$

Maximizar: $\max_{\alpha} L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . x_i x_j . y_i y_j$

- Os vetores x_i e x_j são o vetor de entrada e o padrão de entrada pertencente ao *j*-ésimo exemplo.
- Problema resolvido comumente por métodos de otimização quadrática
- Sequential Minimal Optimization (Algoritmo SMO)
- Solução única e ótima!!!

Multiplicadores de Lagrange

Centro de Informática U·F·P·E

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . x_i x_j . y_i y_j$$

Observações importantes:

- Existe um α_i para cada exemplo de treinamento.

- Na solução ótima de L_D, $\alpha_i > 0$ para os vetores suporte e $\alpha_i = 0$ para os outros exemplos !!!

- Intuição: O hiperplano ótimo depende apenas dos vetores suporte.

Multiplicadores de Lagrange

Maximizar: $\max_{\alpha} L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . x_i x_j . y_i y_j$

Observações importantes:

- Maximizando L_D, o hiperplano ótimo é obtido diretamente:

$$w = \sum_i \alpha_i x_i y_i$$

$b = 1 – w . x_{(s)}$, onde $x_{(s)}$ é um vetor suporte no hiperplano superior

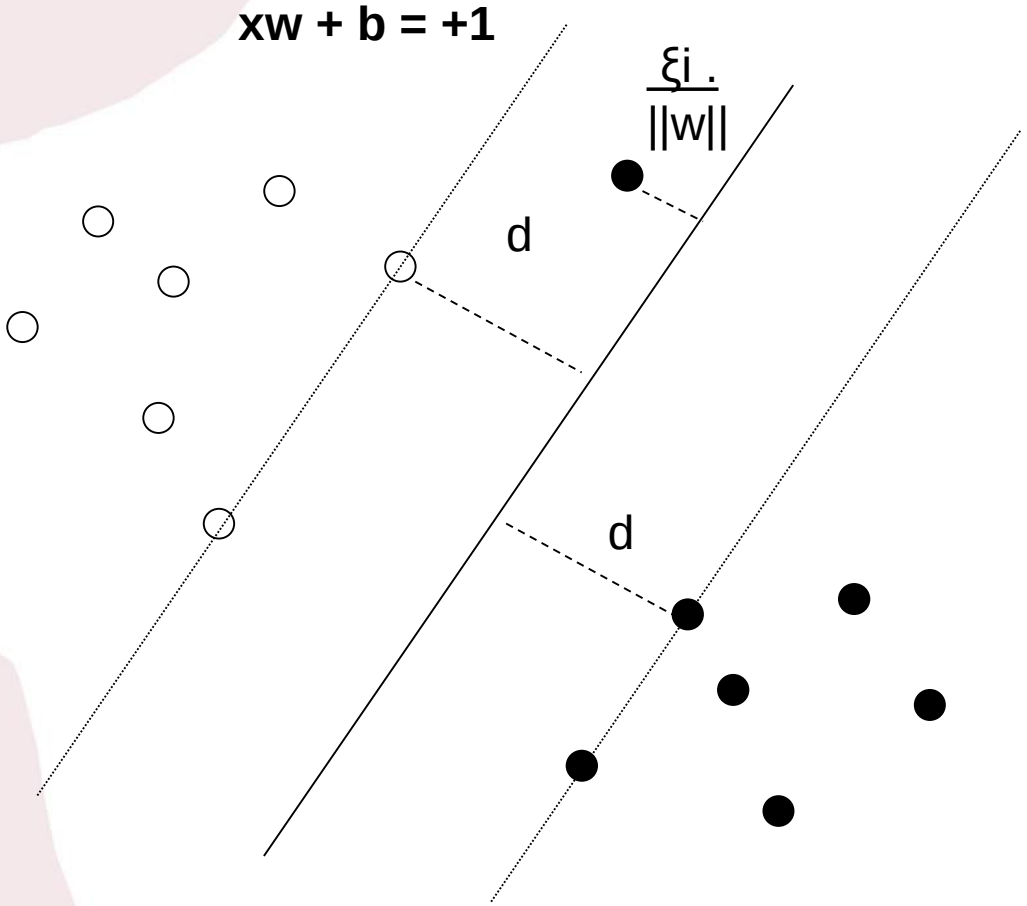
SVM – Soft Margin

- Formulação anterior definida para conjuntos linearmente separáveis
  - Hard Margin SVM

- Para conjuntos não-linearmente separáveis pequenos erros pode ser tolerados

Minimizar: $\frac{\|w\|^2}{2} + C \sum_i \xi_i$

Sujeito a: $y_i (x_i.w + b) - 1 + \xi_i >= 0$


xw + b = +1
ξi .
||w||
d
xw + b = - 1
d
xw+b = 0

SVM – Soft Margin

- A derivação do Lagrangiano introduz apenas uma restrição para α_i

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . x_i x_j . y_i y_j$$

$$0 \leq \alpha i \leq C$$

Valores de α_i limitados pelo parâmetro de complexidade C

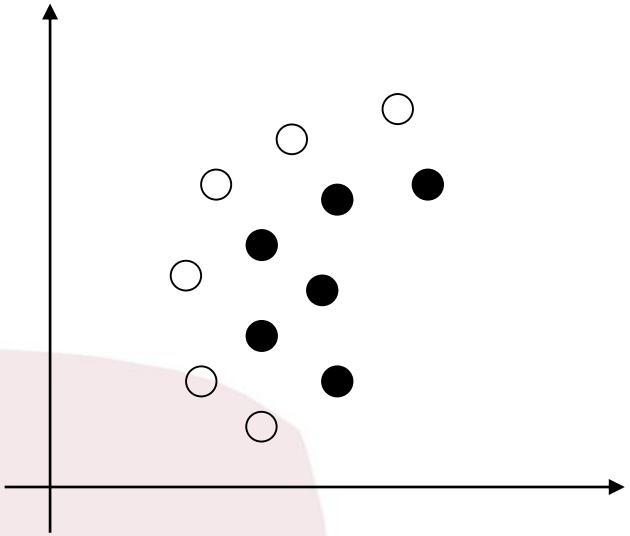
SVM Não-Linear

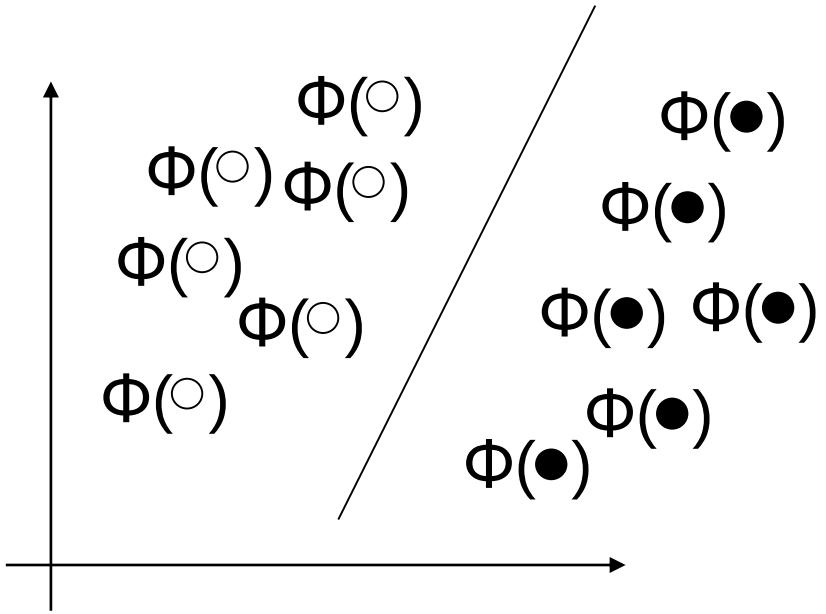
- SVM linear ainda é muito limitado mesmo com margens flexíveis

- Generalização não-linear de SVM
  - Mapear espaço original para espaço não-linear de maior dimensão onde exemplos sejam linearmente separáveis;

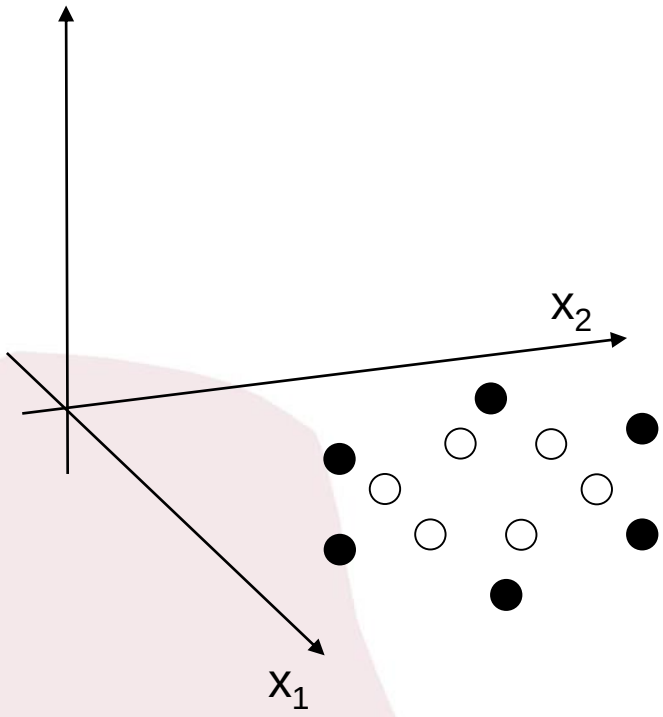
  - Construir hiperplano ótimo no novo espaço.

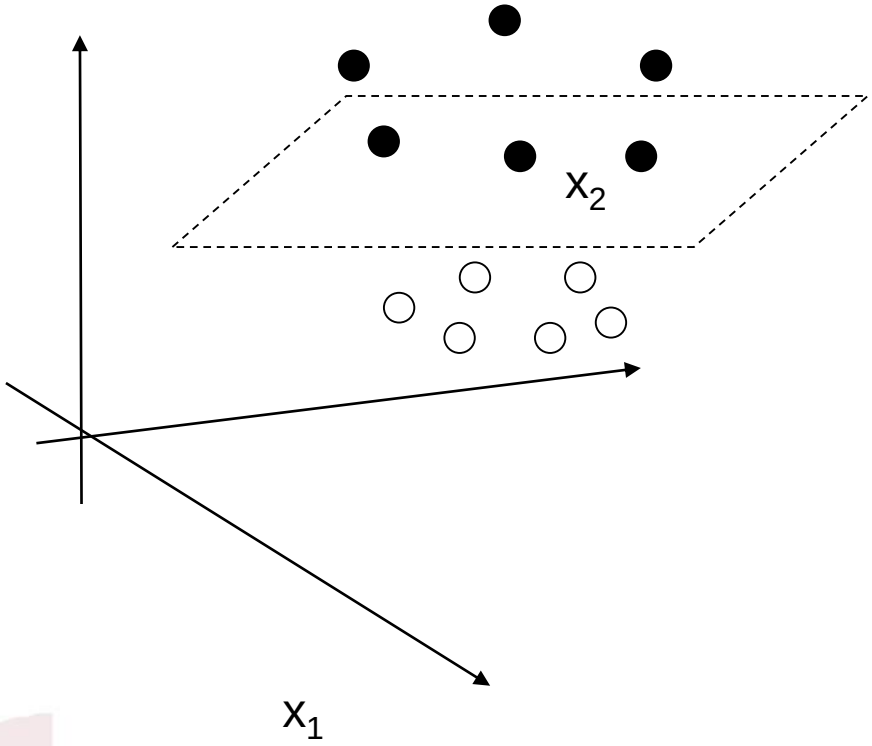
SVM Não-Linear





SVM Não-Linear

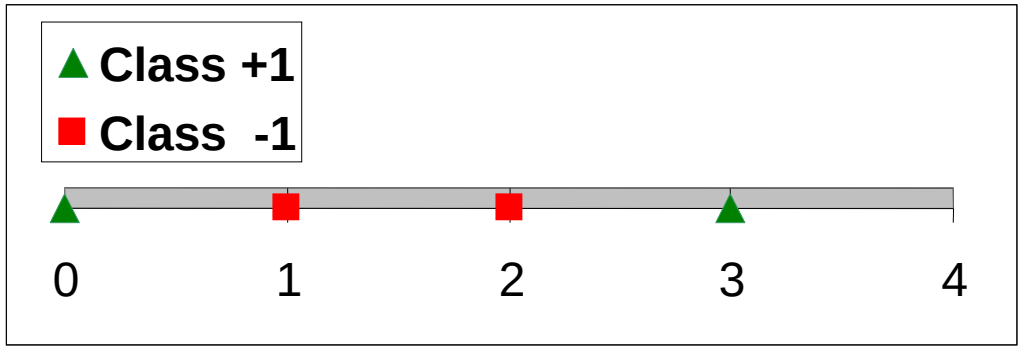




Exemplo

Como separar as duas classes com apenas um ponto?

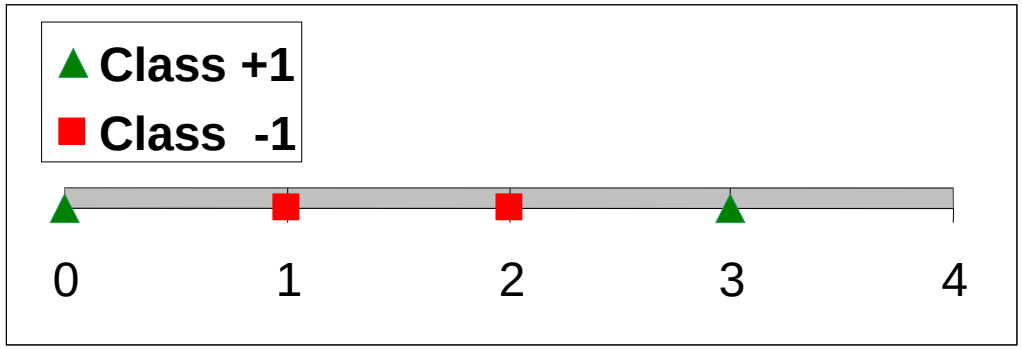
X_1	Class
0	+1
1	-1
2	-1
3	+1



Exemplo

SVM usa uma função não linear sobre os atributos do espaço de características inicial

X_1	Class
0	+1
1	-1
2	-1
3	+1



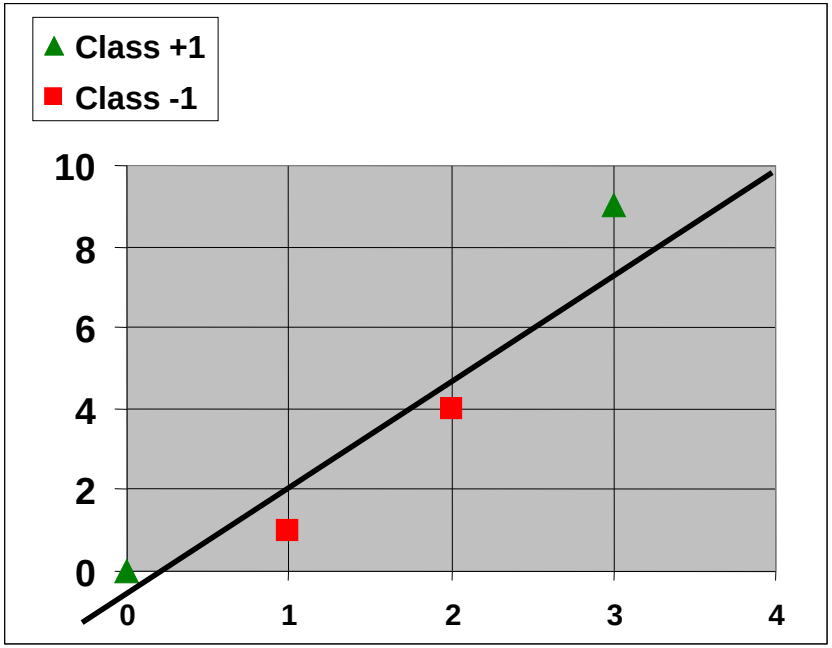
$\Phi(X_1) = (X_1, X_1^2)$

Esta função torna o problema bidimensional

Exemplo

SVM usa uma função não linear sobre os atributos do espaço de características inicial

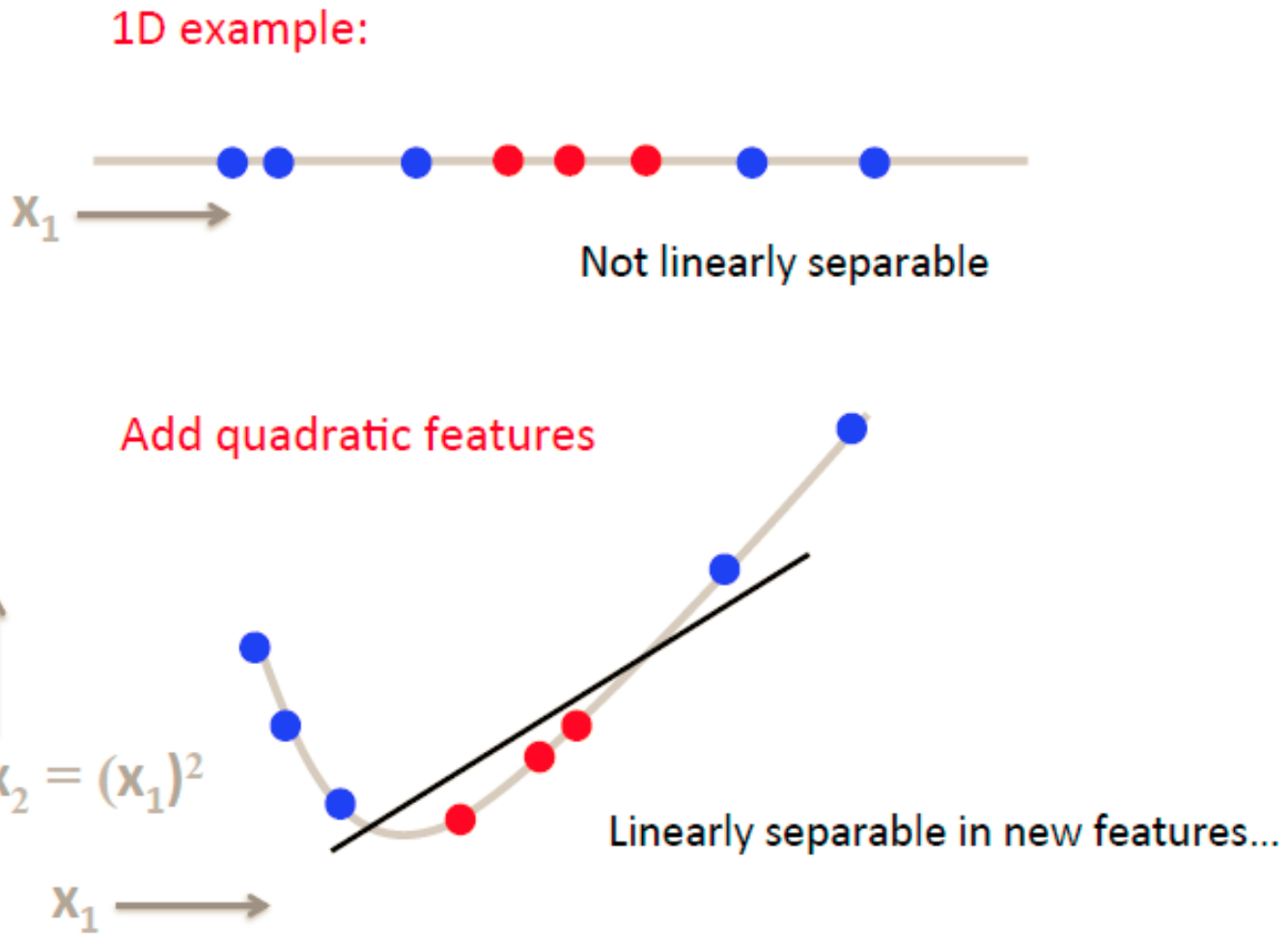
X_1	X_1^2	Class
0	0	+1
1	1	-1
2	4	-1
3	9	+1



$\Phi(X_1) = (X_1, X_1^2)$

Esta função torna o problema bidimensional e os dados linearmente separáveis

Exemplo


1D example:
x_1
Not linearly separable
Add quadratic features
$x_2 = (x_1)^2$
x_1
Linearly separable in new features...

Exemplo

X_1	X_1^2	Class
0	0	+1
1	1	-1
2	4	-1
3	9	+1

- $w \cdot x + b = +1$

  $w_1x_1 + w_2x_2 + b = +1$

  $0w_1 + 0w_2 + b = +1$ → $b = 1$

  $3w_1 + 9w_2 + b = +1$

- $w \cdot x + b = -1$

  $w_1x_1 + w_2x_2 + b = -1$ substituindo b e após w_1

  $1w_1 + 1w_2 + b = -1$ → $w_1 = -2 - w_2$

  $2w_1 + 4w_2 + b = -1$ → $-4 -2w_2 + 4w2 + 1 = -1$

- $w \cdot x + b = 0$ $w_2 = 1$ e $w_1 = -3$

  $w_1x_1 + w_2x_2 + b = 0$ → **$-3x_1 + x_2 + 1 = 0$**

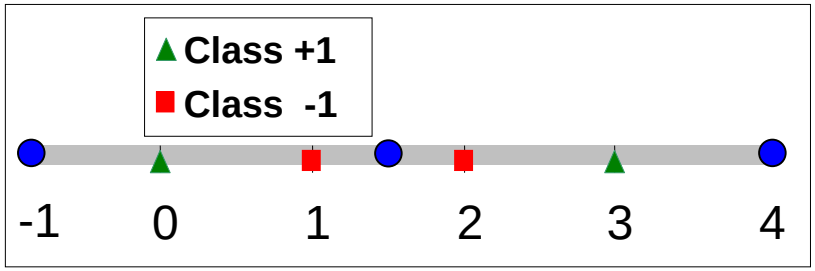
Exemplo

H: $-3x_1 + x_2 + 1 = 0$
Dados de Teste (1.5), (-1), (4)

(1.5) mapear para (1.5, 2.25)
-3 . 1.5 + 2.25 + 1 = - 1.15 [-1]

(-1) mapear para (-1,1)
-3 . -1 + 1 + 1 = 5 [+1]

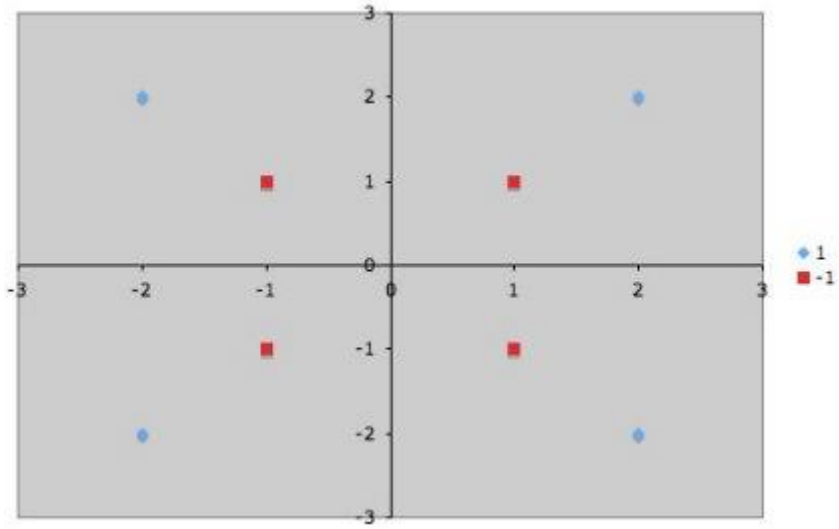
(4) mapear para (4,16)
-3 . 4 + 16 + 1 = 5 [+1]



Outro Exemplo

Como separar as duas classes com apenas uma reta?

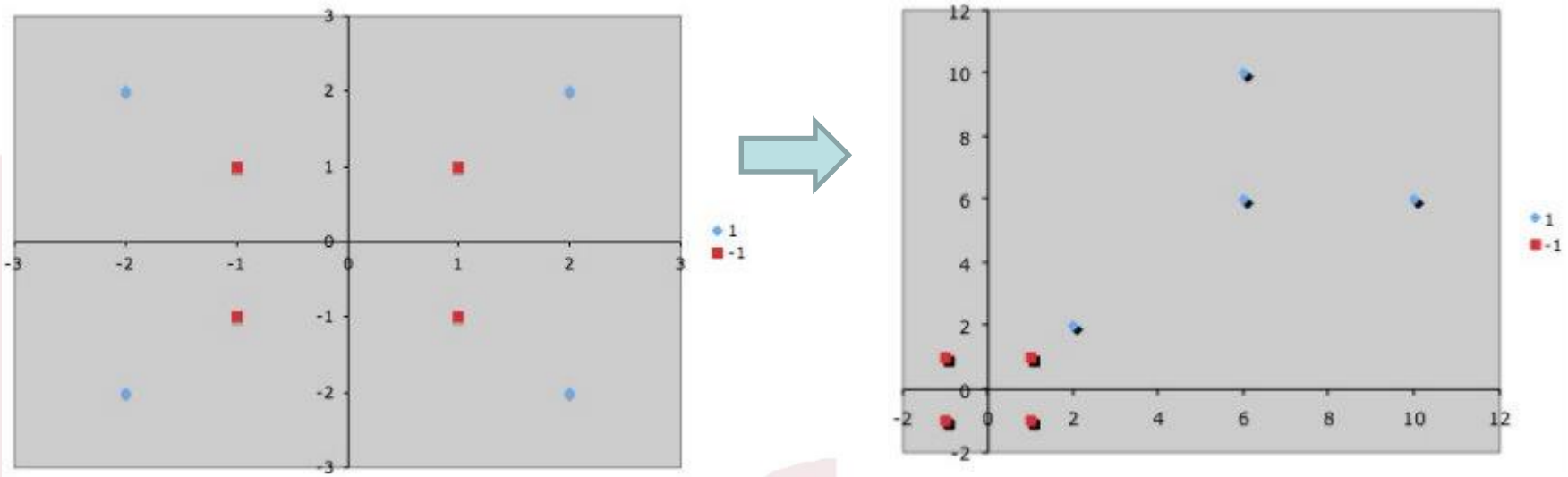
X_1	X_2	Class
1	1	-1
-1	1	-1
1	-1	-1
-1	-1	-1
2	2	+1
-2	2	+1
2	-2	+1
-2	-2	+1



Outro Exemplo

$$\Phi(x_1, x_2) = \begin{cases} (4-x_2+|x_1-x_2|,\ 4-x_1+|x_1-x_2|), & \sqrt{(x_1^2 + x_2^2)} > 2 \\ (x_1, x_2) \end{cases}$$

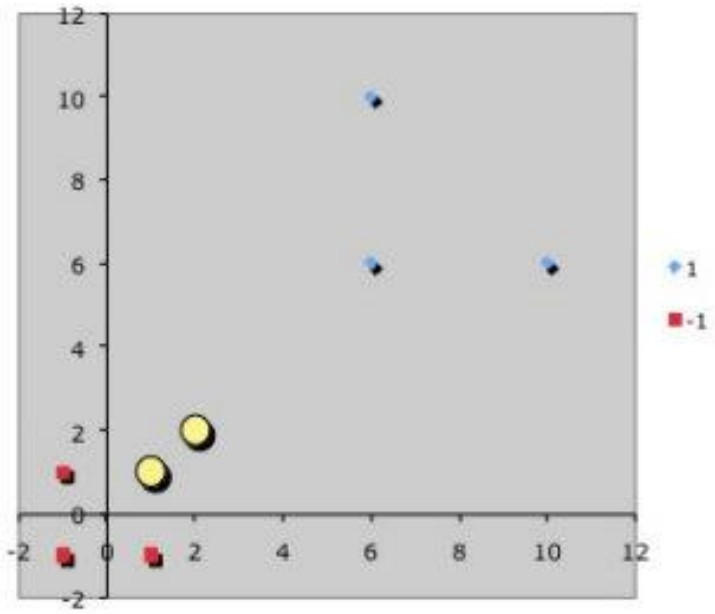
Esta função mantém o problema bidimensional



Outro Exemplo

Vetores de Suporte

x_1	x_2	Class
1	**1**	**-1**
-1	1	-1
1	-1	-1
-1	-1	-1
2	**2**	**+1**
6	6	+1
10	6	+1
6	10	+1



Outro Exemplo

H_1: $w \cdot x + b = 1$
H_2: $w \cdot x + b = -1$

Vetores de Suporte

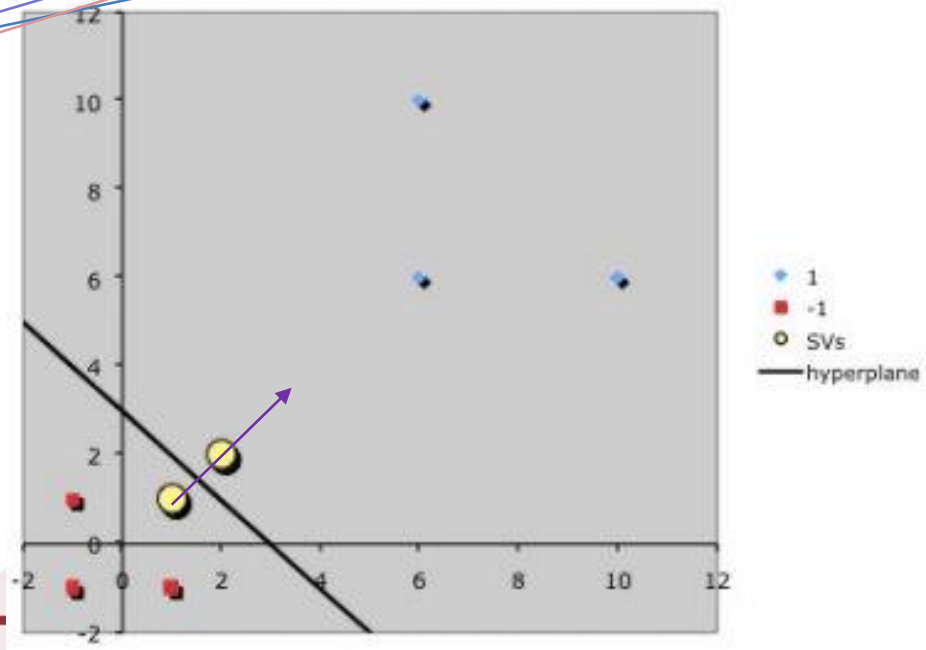
x_1	x_2	Class
1	**1**	**-1**
2	**2**	**+1**

$w_1x_1 + w_2x_2 + b = -1$
$1w_1 + 1w_2 + b = -1$
→ $w_1 = -1 -b -w_2$

$w_1x_1 + w_2x_2 + b = 1$
$2(-1 -b -w_2) + 2w2 + b = 1$
$-2 -2b -2w2 +2w2 + b = 1$
→ $b = -3$

$(2-1)\ x_2 = (2-1)\ x_1$ Equação da reta
$2x_2 - x_2 = 2x_1 - x_1$
$x_2 = x_1$
$w_1 = w_2 = 1$

H_0: $(1,1) \cdot x -3 = 0$
$x1 + x2 -3 = 0$



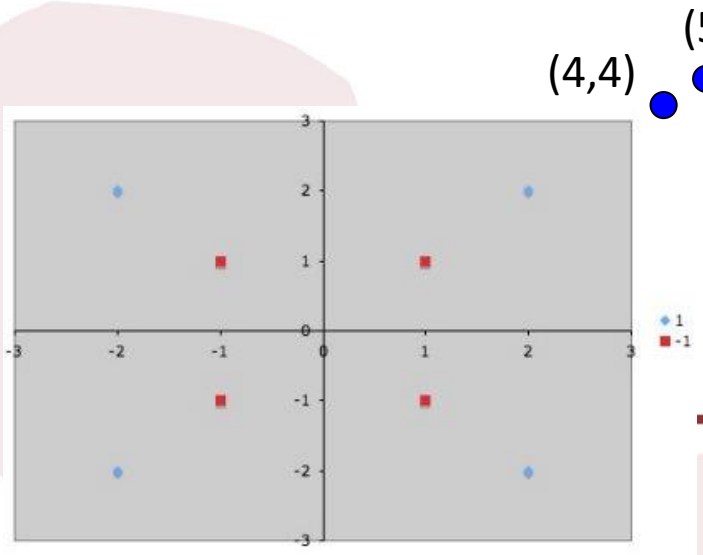
Outro Exemplo

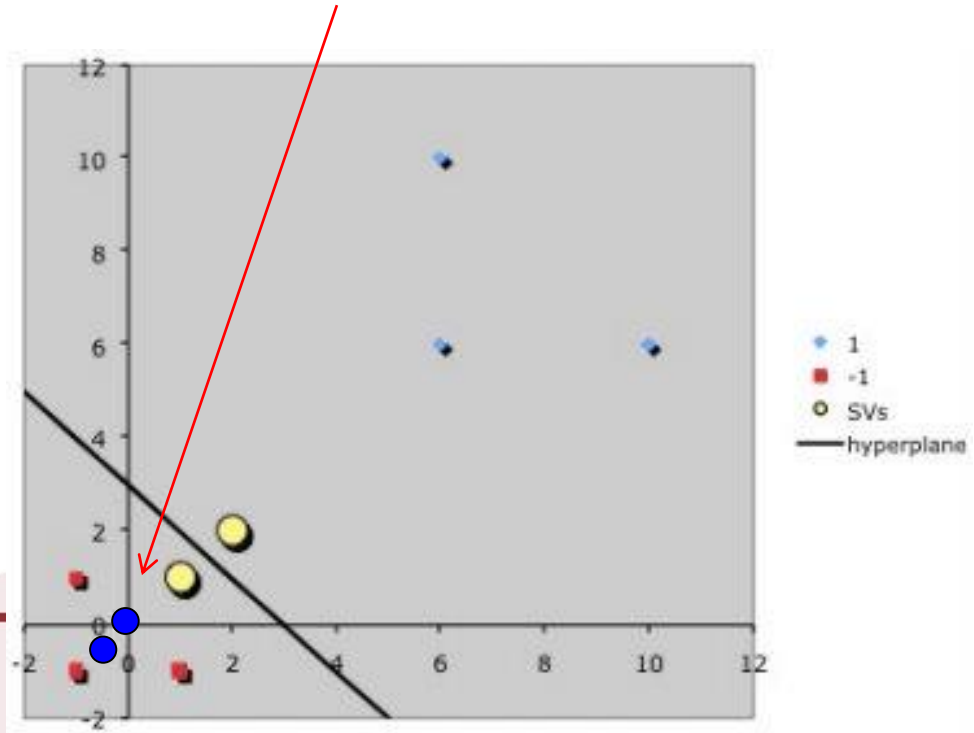
$$\Phi(x_1, x_2) = \begin{cases} (4-x_2+|x_1-x_2|,\ 4-x_1+|x_1-x_2|), & \sqrt{(x_1^2 + x_2^2)} > 2 \\ (x_1, x_2) \end{cases}$$

Esta função realmente separa o espaço original de forma linear?

Dados de teste (5,5) $\Phi(5,5) = (-1, -1)$

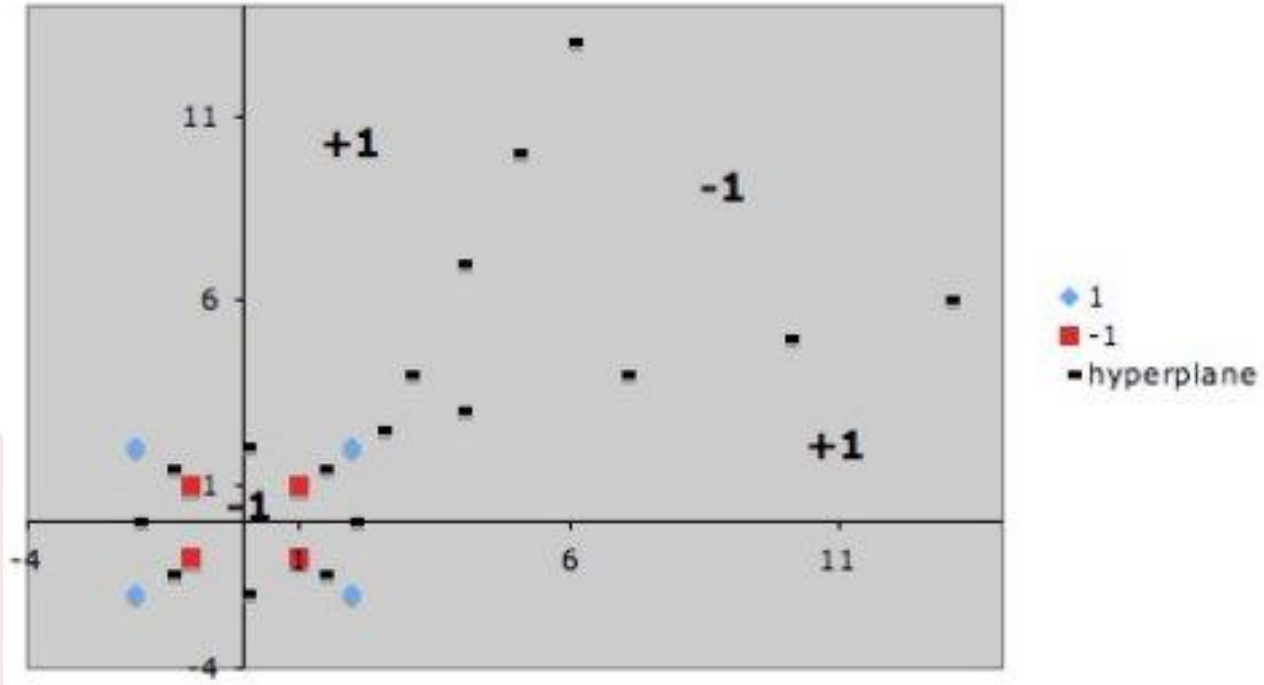
Dados de teste (4,4) $\Phi(4,4) = (0, 0)$ **Erros de classificação!**





Outro Exemplo

- Função de mapeamento não é ideal



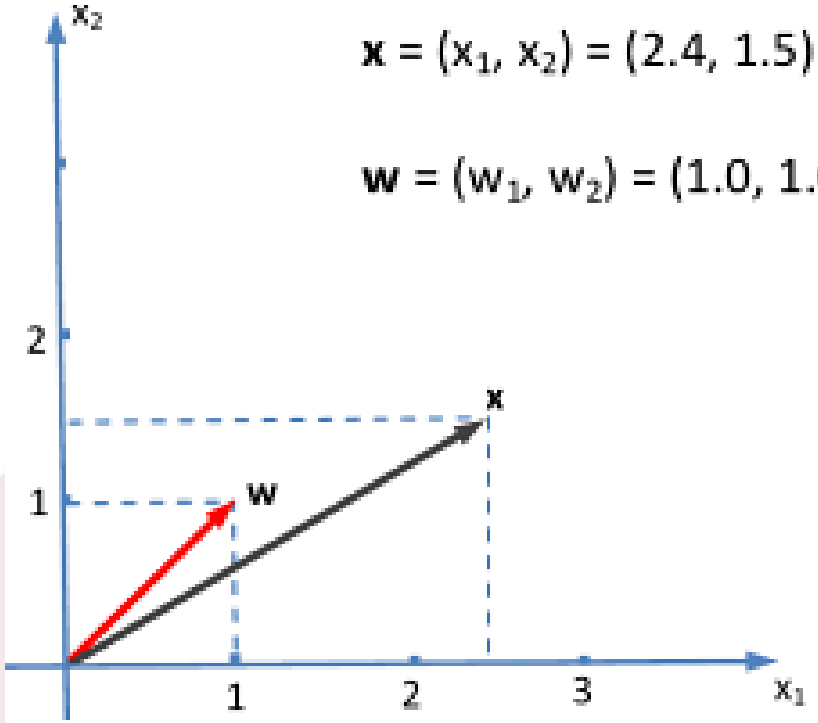
Problema

- Como escolher a função $\Phi(x)$ tal que o espaço de características transformado seja eficiente para classificação e não possua custo computacional alto demais?
  - Com uma função especial, chamada **função kernel** é possível calcular o produto escalar $\Phi(x_i).\ \Phi(x_j)$ sem mesmo conhecer o mapeamento Φ!

SVM (Kernels)

- Em SVMs não-lineares, pontos são mapeados implicitamente através da função de Kernel

- Propriedade básica:
  - Kernel é um produto de vetores em algum espaço
  - $K(x_i, x_j) = \Phi(x_i).\ \Phi(x_j)$



$\mathbf{x} = (x_1, x_2) = (2.4, 1.5)$

$\mathbf{w} = (w_1, w_2) = (1.0, 1.0)$

Produto Interno

$$\mathbf{w} \cdot \mathbf{x} = (w_1 \cdot x_1 + w_2 \cdot x_2)$$
$$= (1.0 \cdot 2.4 + 1.0 \cdot 1.5)$$
$$= (2.4 + 1.5)$$
$$\mathbf{w} \cdot \mathbf{x} = 3.9$$

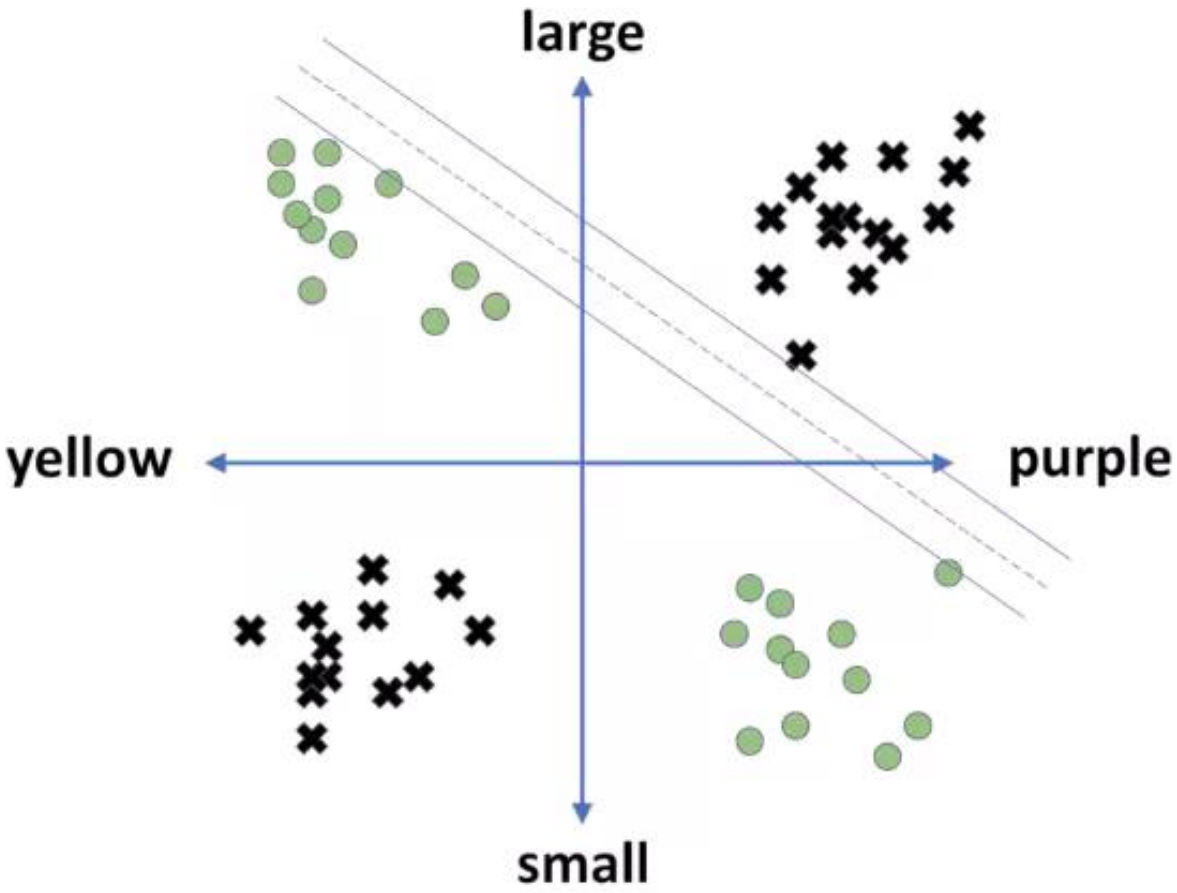
Note que o resultado de um **Produto Interno** é um **valor escalar** e não um vetor!

The Kernel Trick

- Parece ingênuo mas pode transformar quaisquer algoritmos lineares que possam ser expressos em termos de produtos internos em algoritmos não-lineares.
- Incrementar o número de dimensões do espaço
- E incrementar muito! mover seu problema para um espaço em que exista uma dimensão independente para cada uma das possíveis entradas de sua função!

https://www.youtube.com/watch?v=3liCbRZPrZA

The Kernel Trick



Vídeo: Brandon Rohrer

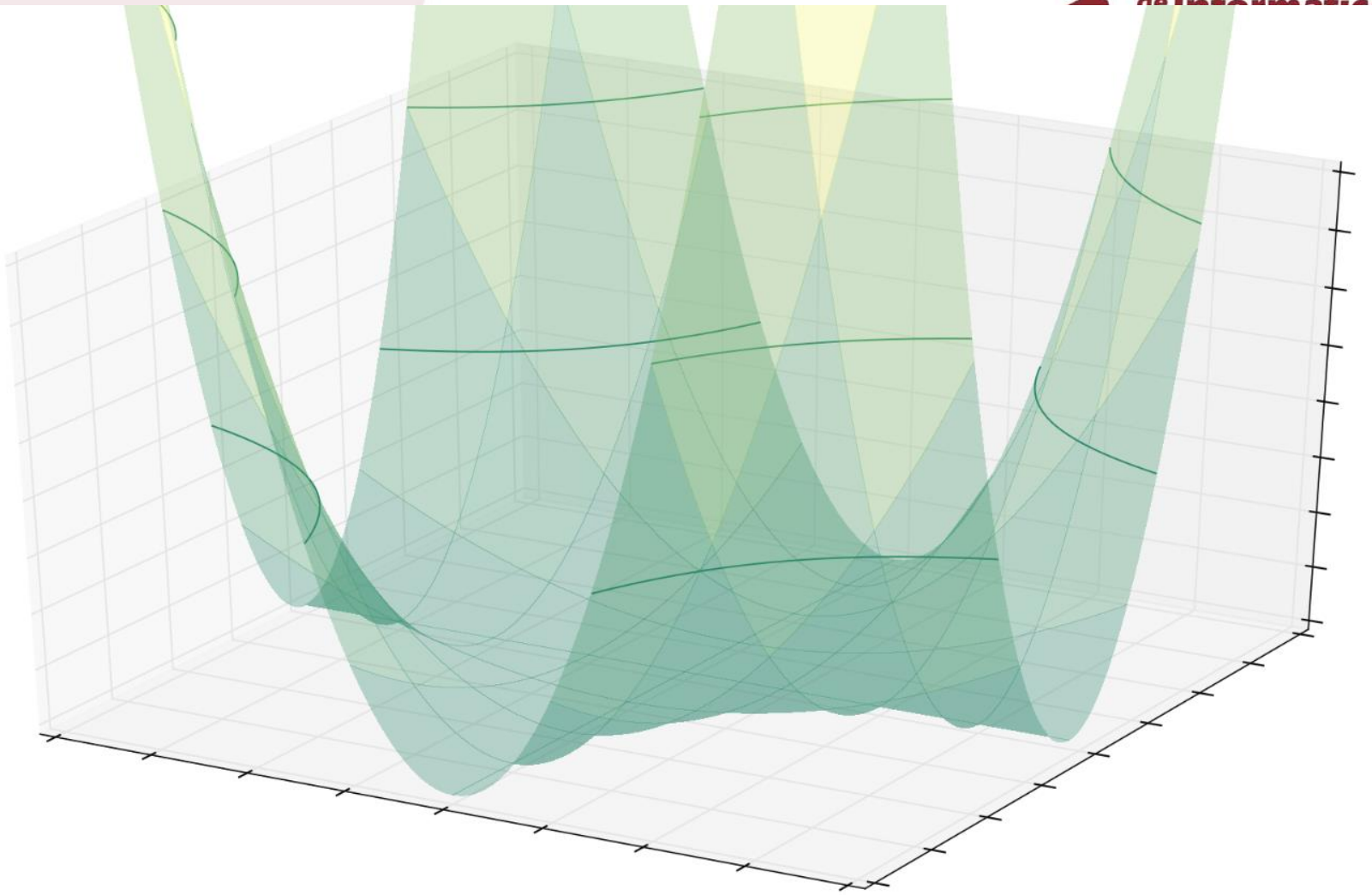


Imagem: Brandon Rohrer

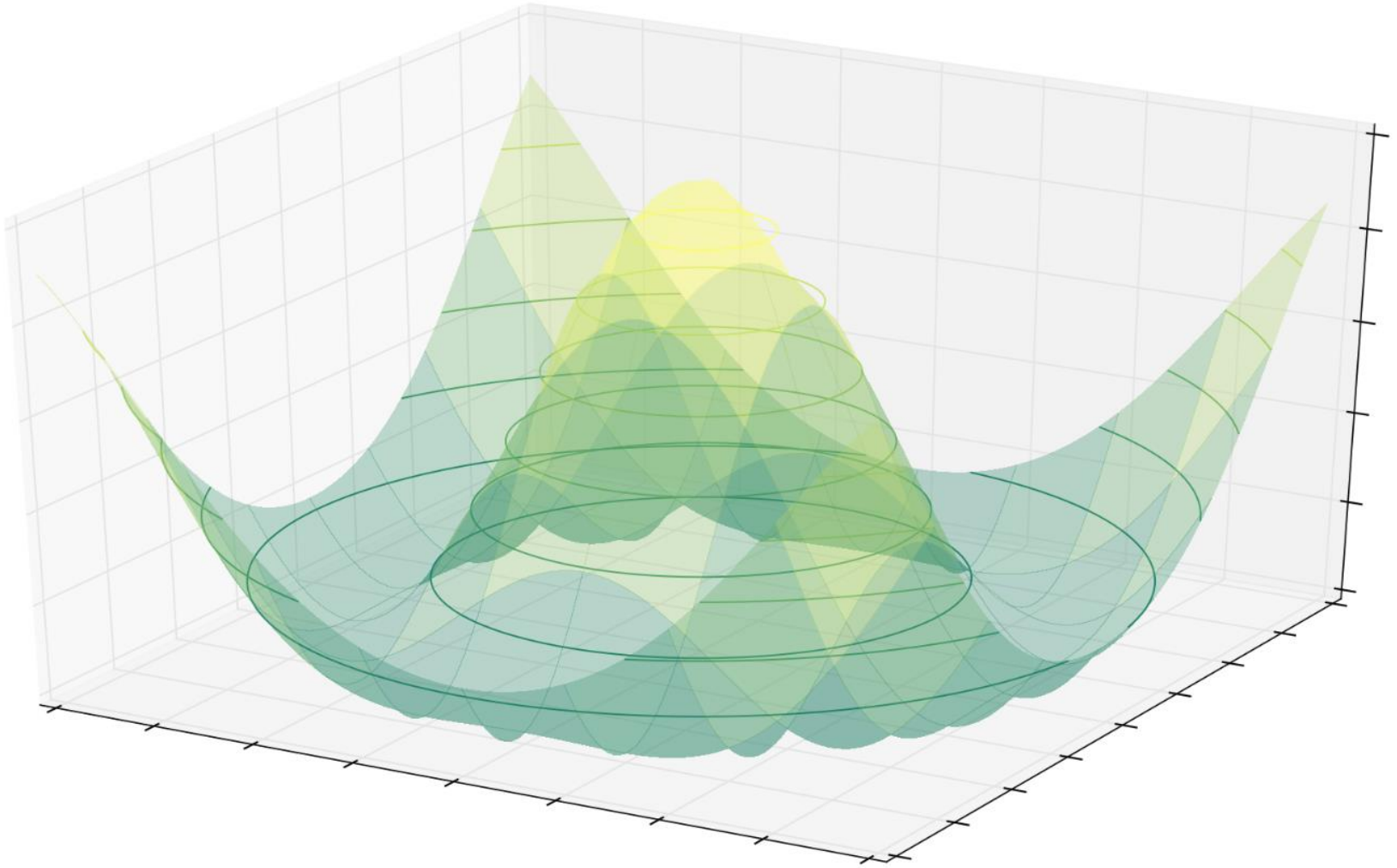


Imagem: Brandon Rohrer

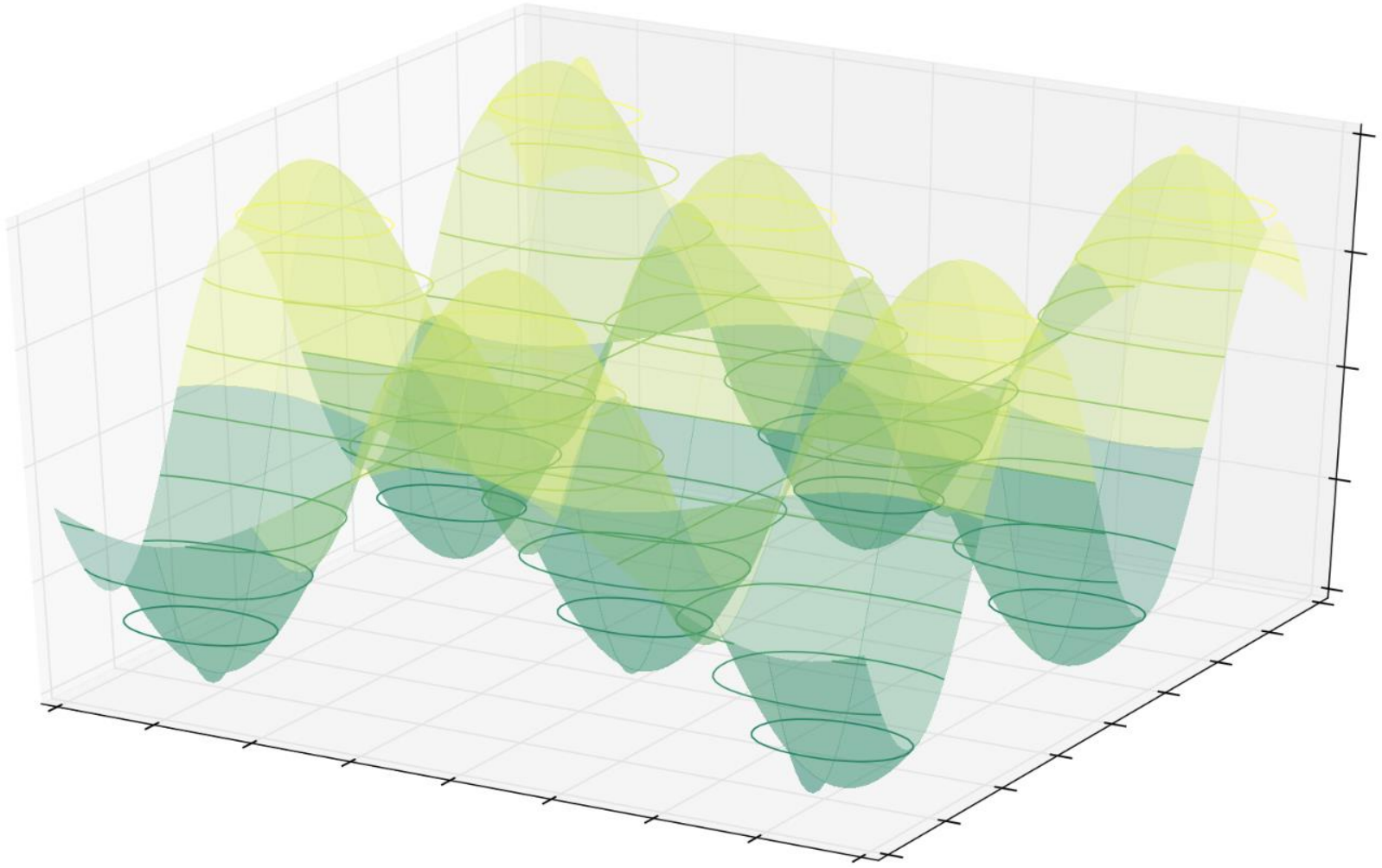
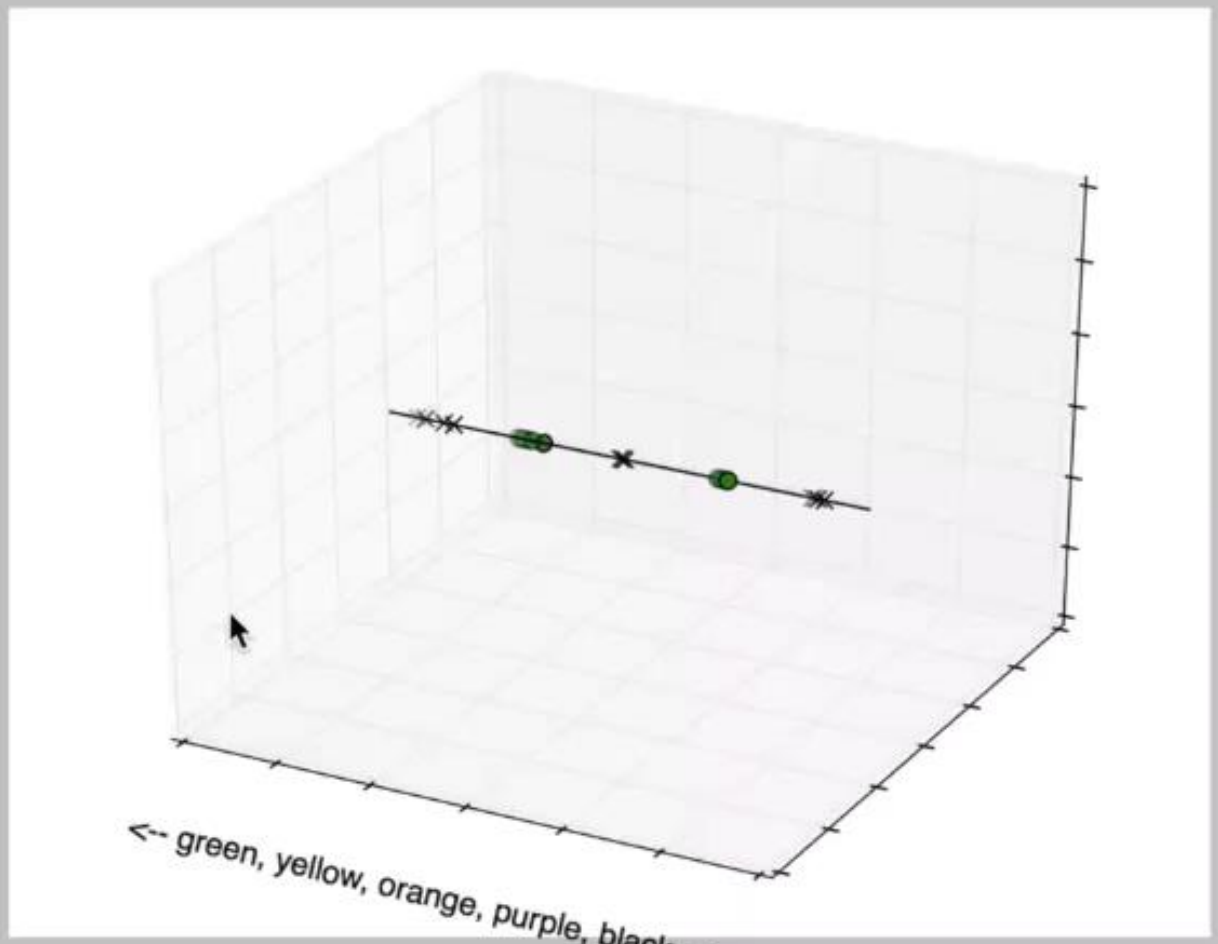


Imagem: Brandon Rohrer

The Kernel Trick



Vídeo: Brandon Rohrer

The Kernel Trick

- O classificador linear depende do produto interno entre exemplos

$$K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j$$

- Se cada ponto for mapeado para a um espaço de alta dimensão através de uma transformação Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, o produto interno fica:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$$

- Uma função de kernel é uma função que é equivalente a um produto interno em um espaço de maior dimensionalidade

The Kernel Trick

- Exemplo:

  Vetores de 2 dimensões $\mathbf{x}=[x_1 \;\; x_2]$; seja $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j)^2$,

  Precisamos mostrar que $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$:

  $$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

  $$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^{\mathsf{T}}\, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

  $$= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_j), \quad \text{onde } \boldsymbol{\varphi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

- Não precisamos calcular $\boldsymbol{\varphi}(\mathbf{x})$ explicitamente

The Kernel Trick

- Para algumas funções $K(\mathbf{x}_i,\mathbf{x}_j)$ checar que $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathsf{T}}\varphi(\mathbf{x}_j)$ pode ser difícil
- Como provar que o espaço de transformação existe?
- Por construção
- Propriedades matemáticas (Teorema de Mercer)
  - *Toda função simétrica definida semi-positiva é um kernel*
- *Quem se importa? ☺*

Porque usar Kernels?

- Lembrando:

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . x_i x_j . y_i y_j$$

- Fazendo mapeamento Φ:

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . \Phi(x_i)\, \Phi(x_j) . y_i y_j$$

Porque usar Kernels?

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . \Phi(x_i)\, \Phi(x_j) . y_i\, y_j$$

$$K(x_i, x_j) = \Phi(x_i) . \Phi(x_j)$$

$$\underset{\alpha}{\text{Maximizar:}}\ L_D = \sum_i \alpha_i - \sum_{i,j} \alpha_i . \alpha_j . K(x_i, x_j) . y_i\, y_j$$

Porque usar Kernels?

- Hiperplano ótimo é definido por:

$$xw + b = \phi(x).\sum_i \alpha_i \phi(x_i) y_i + b =$$

$$= \sum_i \alpha_i \phi(x_i)\phi(x) y_i + b =$$

$$= \sum_i \alpha_i y_i K(x_i, x) + b$$

- O parâmetro $b = 1 - wx_{(s)}$ é definido por:

$$b = 1 - \phi(x_{(s)})\sum_i \alpha_i \phi(x_i) y_i = 1 - \sum_i \alpha_i \phi(x_i)\phi(x_{(s)}) y_i$$

$$= 1 - \sum_i \alpha_i y_i K(x_i, x_{(s)})$$

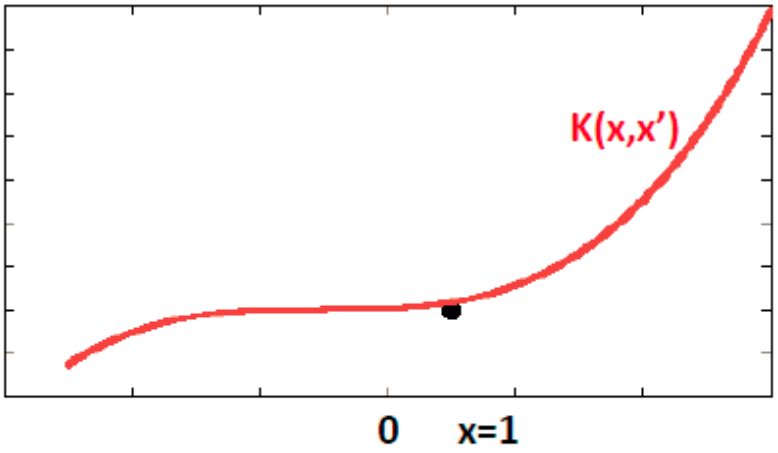
Porque usar Kernels

- Não é necessário definir explicitamente o mapeamento Φ
  - Em alguns casos, é impossível definir Φ

- Todo o treinamento e uso do modelo são realizados apenas usando o Kernel

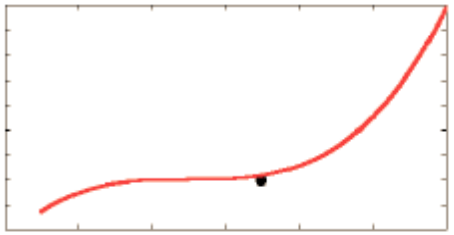
Common kernel functions

- Some commonly used kernel functions & their shape:
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$



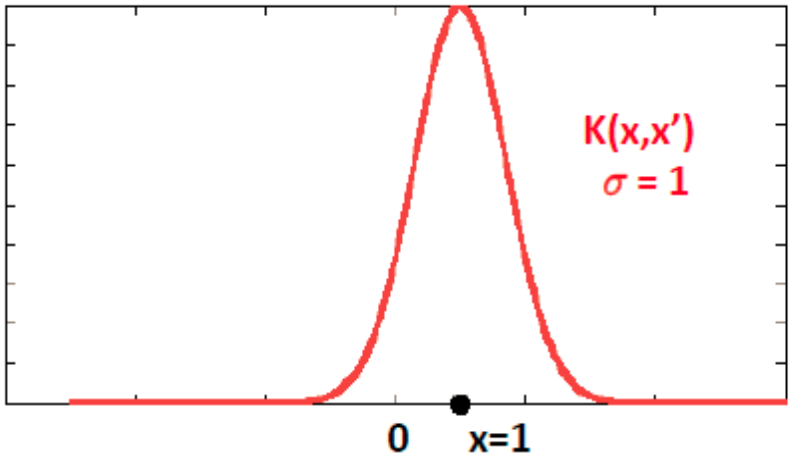
Common kernel functions

- Some commonly used kernel functions & their shape:



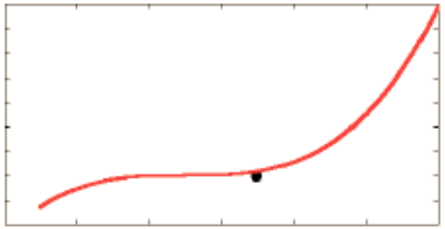
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$
- Radial Basis Functions

$$K(a,b) = \exp(-(a-b)^2/2\sigma^2)$$

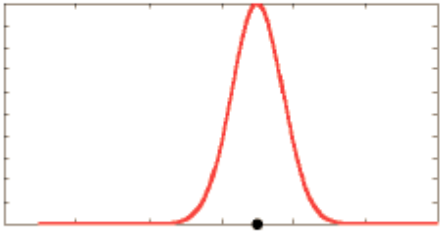


Common kernel functions

- Some commonly used kernel functions & their shape:



- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$

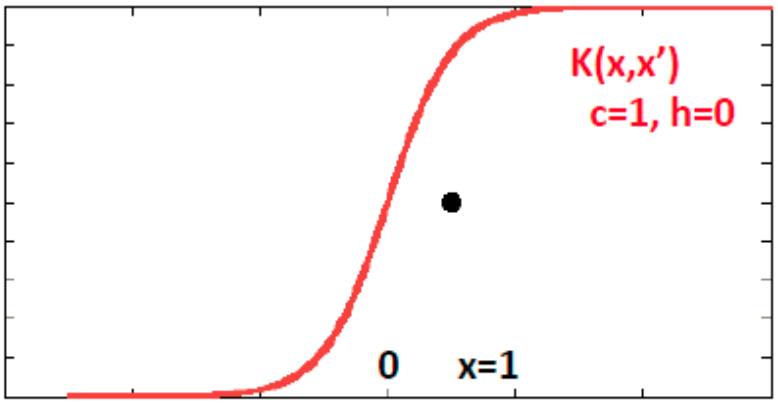


- Radial Basis Functions

$$K(a,b) = \exp(-(a-b)^2/2\sigma^2)$$

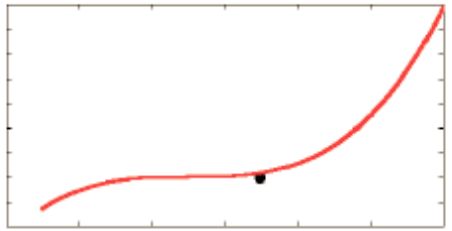
- Saturating, sigmoid-like:

$$K(a,b) = \tanh(ca^T b + h)$$



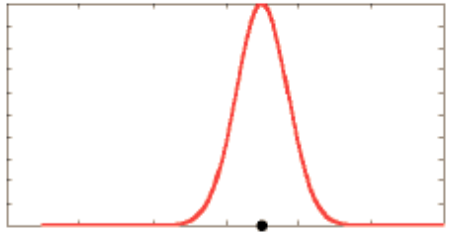
Common kernel functions

- Some commonly used kernel functions & their shape:
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$



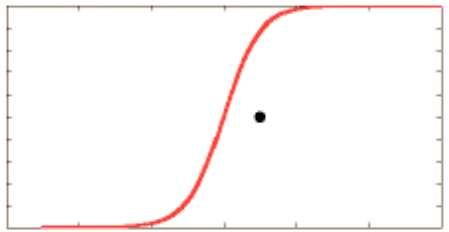
- Radial Basis Functions

$$K(a,b) = \exp(-(a-b)^2/2\sigma^2)$$



- Saturating, sigmoid-like:

$$K(a,b) = \tanh(ca^T b + h)$$



- Many for special data types:
  - String similarity for text, genetics
- In practice, may not even be Mercer kernels…

SVMs para múltiplas classes

- **One-versus-all**
  - Train *n* binary classifiers, one for each class against all other classes.
  - Predicted class is the class of the most confident classifier

- **One-versus-one**
  - Train *n(n-1)/2* classifiers, each discriminating between a pair of classes
  - Several strategies for selecting the final classification based on the output of the binary SVMs

- **Truly MultiClass SVMs**
  - Generalize the SVM formulation to multiple categories

SVM – Seleção de Modelos

- A escolha do Kernel é importante para o desempenho das SVMs
- Dependendo do Kernel utilizado alguns parâmetros devem ser definidos
- Parâmetro de complexidade C é outro aspecto importante

SVM – Seleção de Modelos

- Kernel RBF é mais flexível que o polinomial

- Kernel RBF depende de parâmetro gamma (γ)
  - Valores altos dão maior flexibilidade ao modelo mas também aumentam risco de overfitting

SVM – Seleção de Modelos

- Sobre parâmetro C:
  - Valores muito altos propiciam a geração de modelos mais complexos (risco de overfitting)
  - Valores muito baixos podem aumentar risco de underfitting

SVM – Seleção de Modelos

- Grid-Search (Hsu et al. 2007)
  - Separe conjunto de treinamento e teste
  - Com o conjunto de treinamento, realize validação cruzada para encontrar melhores parâmetros C e γ
    - C = 2^-5, ..., 2^+15
    - γ = 2^-15, ..., 2^+3
  - Use o melhor par de C e γ e treine a SVM com o conjunto todo de treinamento
  - Teste SVM treinada

SVM - Comentários

- SVM se situa dentre as técnicas de aprendizado mais poderosas
- Baseada em uma teoria matemática forte
- Ou seja, justificável teoricamente e com bom desempenho empírico

SVM - Comentários

- Apesar de ter poucos parâmetros para selecionar (e.g., função de kernel), escolha adequada é importante

- Maior desvantagem é o tempo de treinamento e uso (dependendo da quantidade de classes)

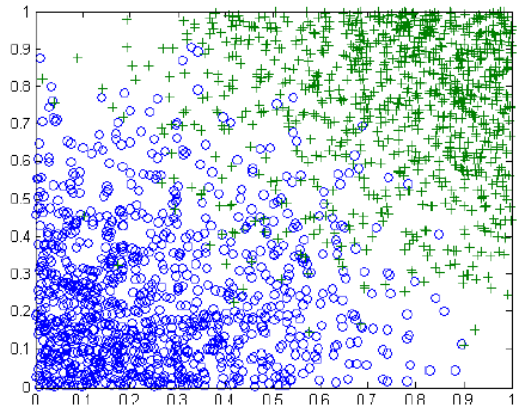
Demo

MLP
Applet: http://lcn.epfl.ch/tutorial/english/mlp/html/index.html
http://freeisms.com/MLPAppletItself.html

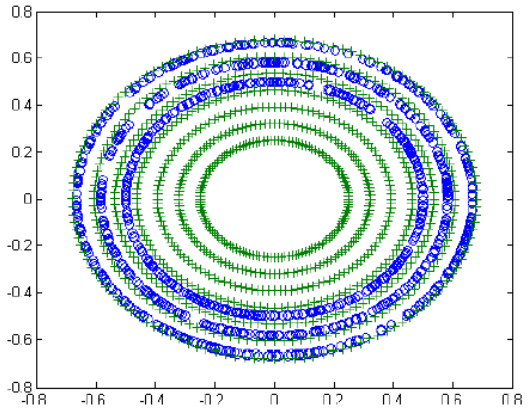
RBF
Applet: http://lcn.epfl.ch/tutorial/english/rbf/html/index.html
http://www.cvlibs.net/projects/gausspro.html

SVM
Applet: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
http://www.cs.jhu.edu/~jason/tutorials/SVMApplet/

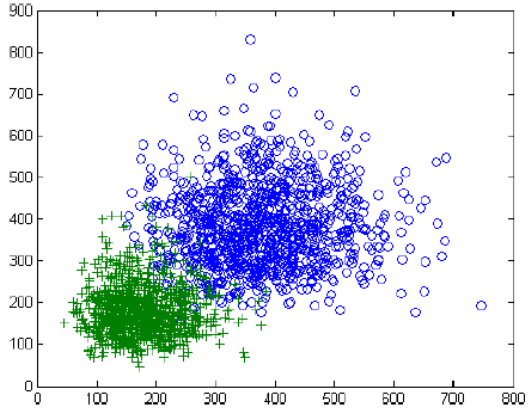
Vários classificadores
Applet: http://www.cs.technion.ac.il/~rani/LocBoost/



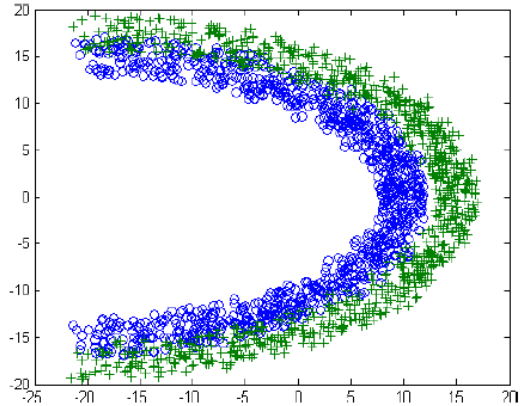
(a) Beta Distribution



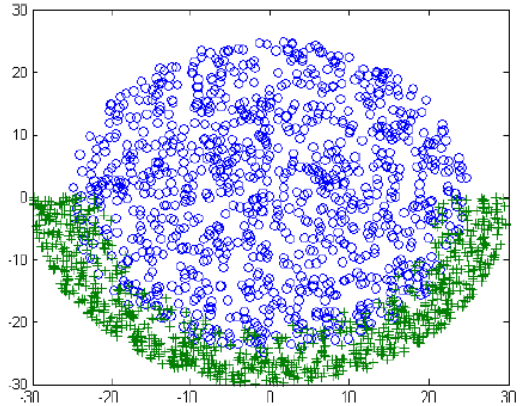
(b) Cluster in Cluster



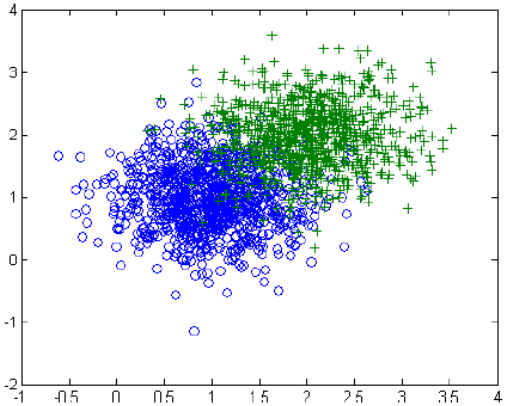
(c) Gamma Distribution



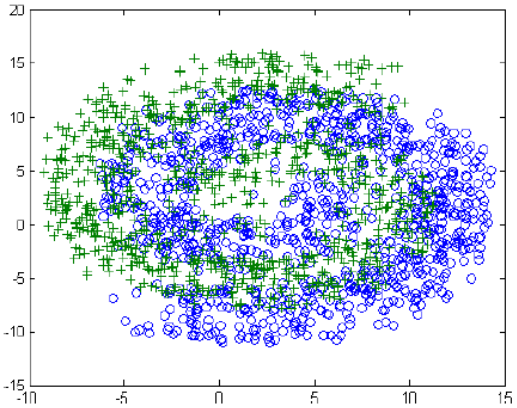
(d) Half Kernel



(e) Moon



(f) Normal Distribution



(g) Spirals

Toy problems

Bibiografia básica

C. BURGES, A Tutorial on Support Vector Machines for Pattern Recognition.

S. GUNN, Support Vector Machines for Classification and Regression.

Fonte

Site do Prof. **Ricardo Prudêncio** / CIN - UFPE:
http://www.cin.ufpe.br/~rbcp

Site do Prof. **Luis Alvares** / UFRGS:
http://http://www.inf.ufrgs.br/~alvares/

Applet de simulação
http://cs.stanford.edu/people/karpathy/svmjs/demo/
http://www.csie.ntu.edu.tw/~cjlin/libsvm/
http://www.cs.jhu.edu/~jason/tutorials/SVMApplet/