Supervised learning

- Multi-class
- Support vector machines
- Neural networks
- Feature selection

Muti-class classification

Classification task with K classes $\{C_1, C_2, \ldots C_K\}$

Currently, there is no clear solution to the problem of multi-class classification

3 approaches:

- One vs All
- All vs All
- Hierarchical classification

One vs All

K classifiers: a class VS all other classes

$$\begin{cases} C_1 \text{ VS } \{C_2, C_3, \ldots C_K\} \text{ -> score}_1 \\ C_2 \text{ VS } \{C_1, C_3, \ldots C_K\} \text{ -> score}_2 \\ \ldots \\ C_i \text{ VS } \{C_1, \ldots, C_{i-1}, C_{i+1}, \ldots C_K\} \text{ -> score}_i \\ \ldots \\ C_K \text{ VS } \{C_1, C_2, \ldots C_{K-1}\} \text{ -> score}_K \end{cases}$$

$C^* \text{ <- } \operatorname{argmax}_i\{\text{score}_1, \text{score}_2, \ldots \text{score}_i, \ldots \text{score}_K\}$

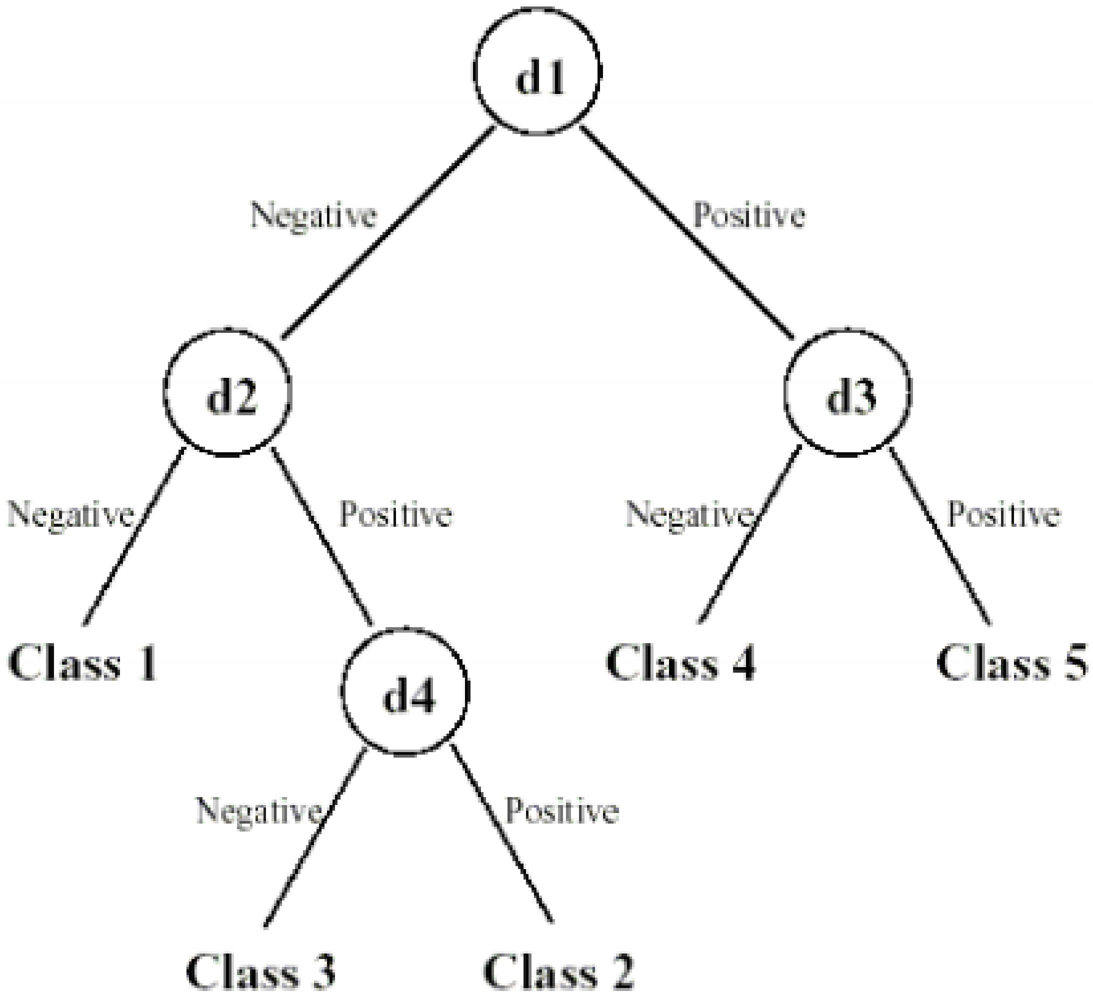
All VS All

$K(K-1)/2$ binary classifiers: C_i VS C_j for all $i>j$

	C_1	C_2	…	C_j	…	C_K	Total
C_1		0		1		0	6
C_2	1			1		0	5
…						0	
C_i	1	1				0	10
…							
C_K	0	0		0			2

$C^* <- \text{argmax}_i\{Total_1, Total_2, \ldots Total_i, \ldots Total_K\}$

Hierarchical classification



Performance of multi-class classifiers

Confusion matrix

	C_1	C_2	…	C_j	…	C_K
C_1	M_{11}	M_{12}		M_{1j}		M_{1K}
C_2	M_{21}	M_{22}				
…						
C_i	M_{j1}	M_{j2}		M_{jj}		M_{jK}
…						
C_K	M_{K1}	M_{K2}		M_{Kj}		M_{KK}

Cost Matrix

	C_1	C_2	…	C_j	…	C_K
C_1	α_{11}	α_{12}		α_{1j}		α_{1K}
C_2	α_{21}	α_{22}				
…						
C_i	α_{j1}	α_{j2}		α_{jj}		α_{jK}
…						
C_K	α_{K1}	α_{K2}		α_{Kj}		α_{KK}

No ROC space can be defined

Evaluation done by classification cost

Support Vector Machine (SVM)

Supervised Learning

Introduction

- Binary classification method
- Introduced by Vladimir Vapnik in 1995
- Competitive in many cases
- Relatively easy to use
- Many extention: Regression, density estimation, kernel PCA,…

Classification problem

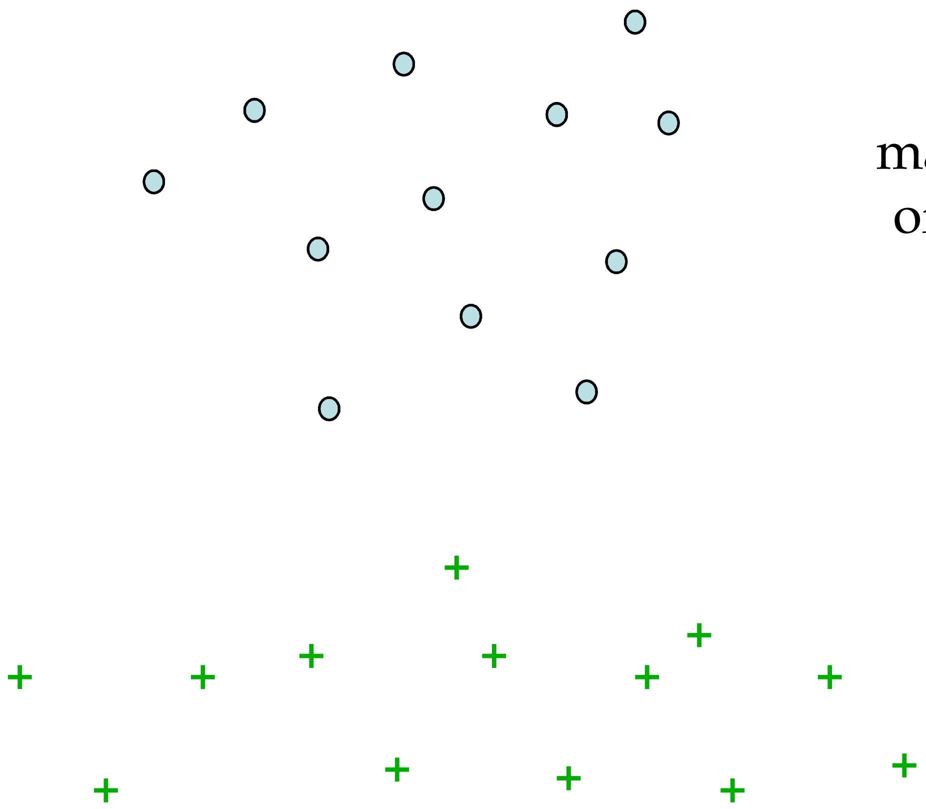
- Training examples: x_i , i=1,…,l
- Consider a simple case with two classes:

  Define a class vector y

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ in class P} \\ -1 & \text{if } \mathbf{x}_i \text{ in class N} \end{cases}$$

- Classifier: hyperplane that separates all examples

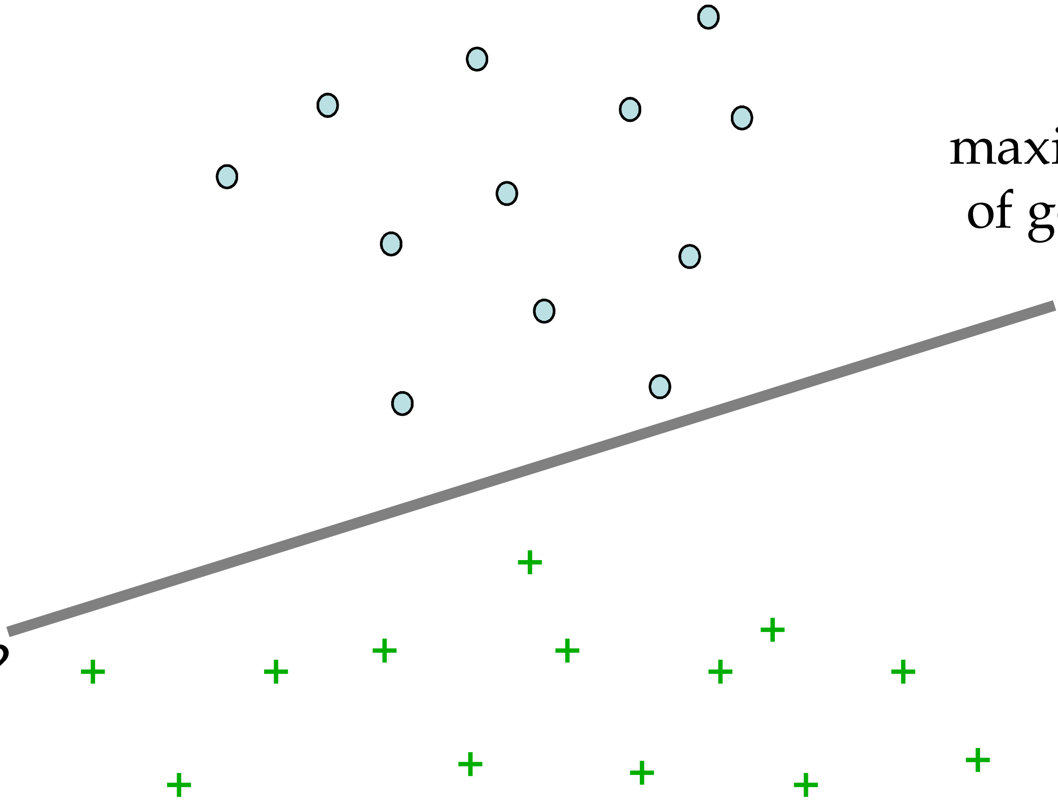
2-class problem



Objective:
maximize the number
of good classification

2-class problem

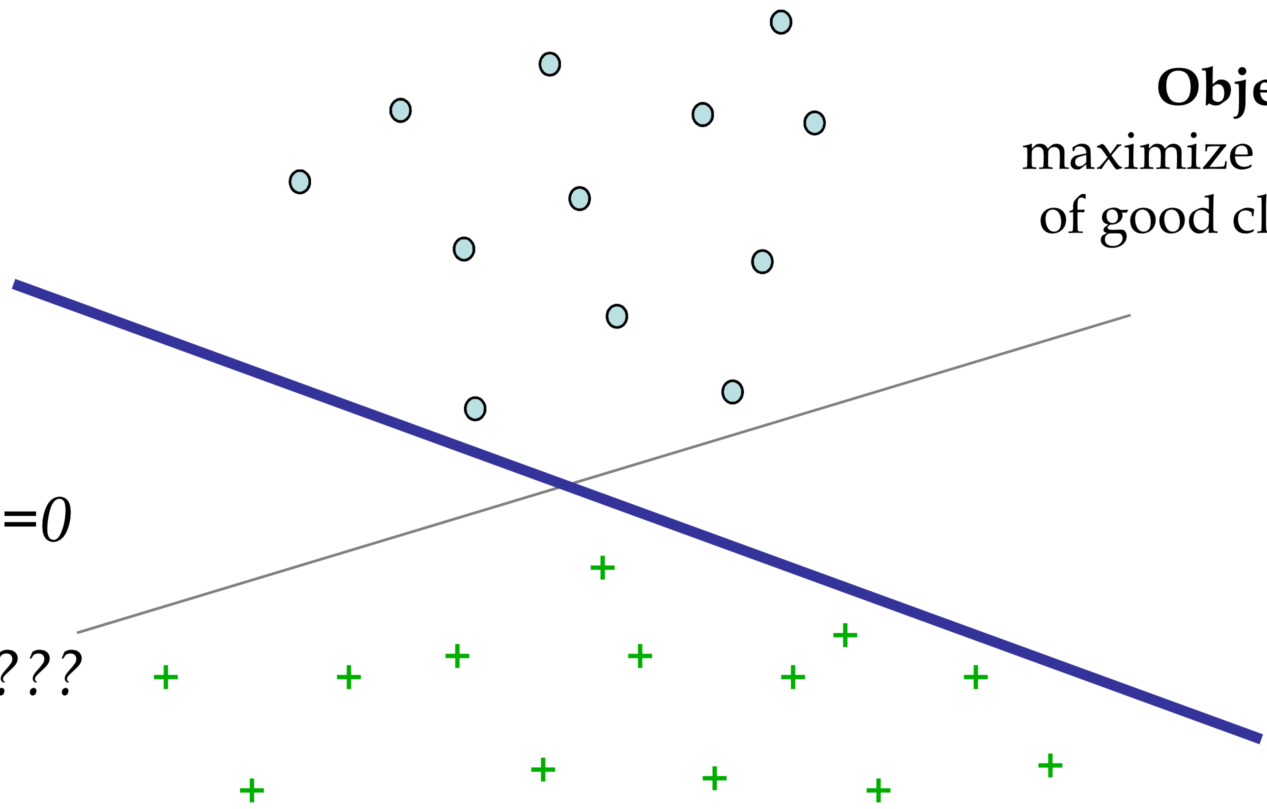
Objective:
maximize the number
of good classification



wx+ b=0

(w,b) ???

2-class problem

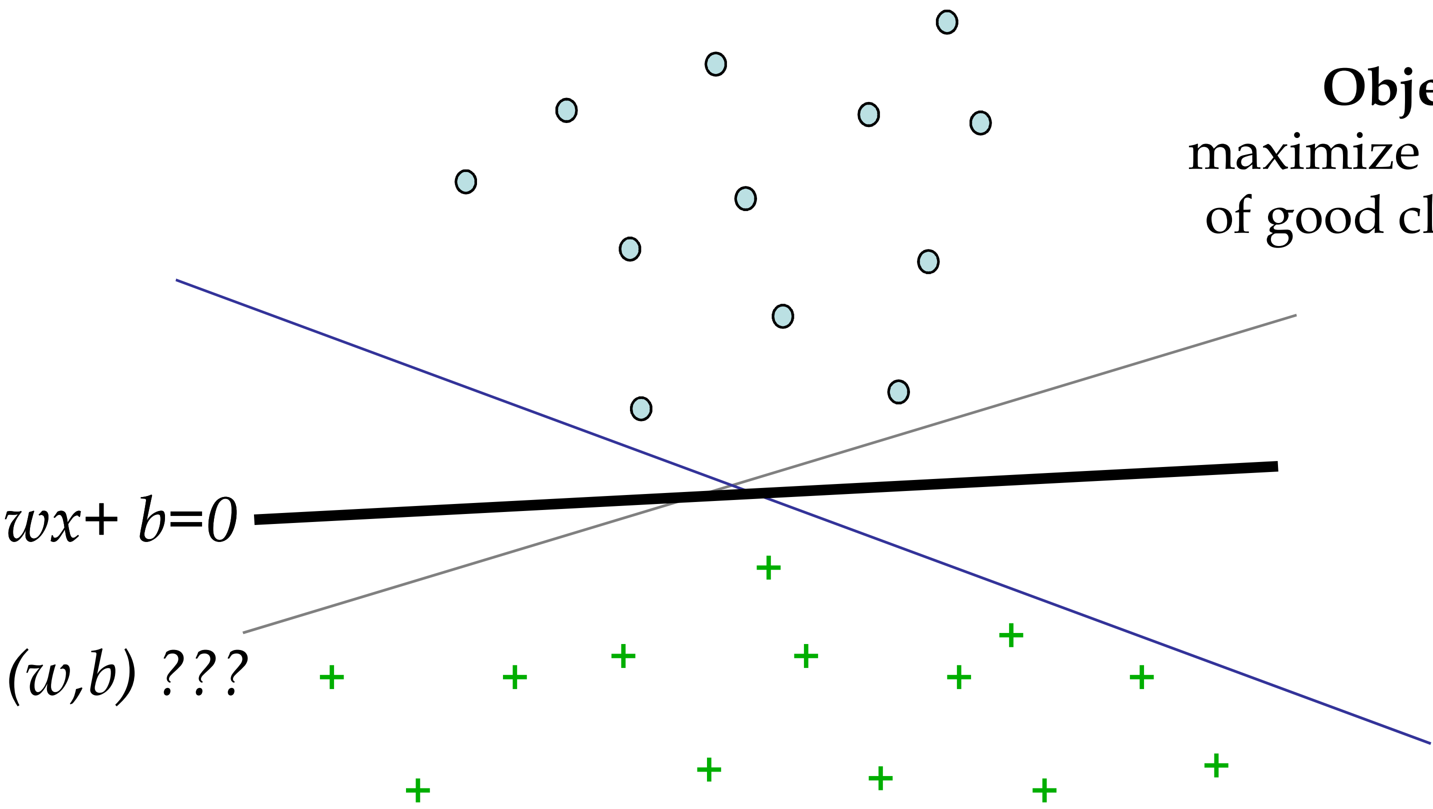


Objective:
maximize the number
of good classification

$wx+b=0$

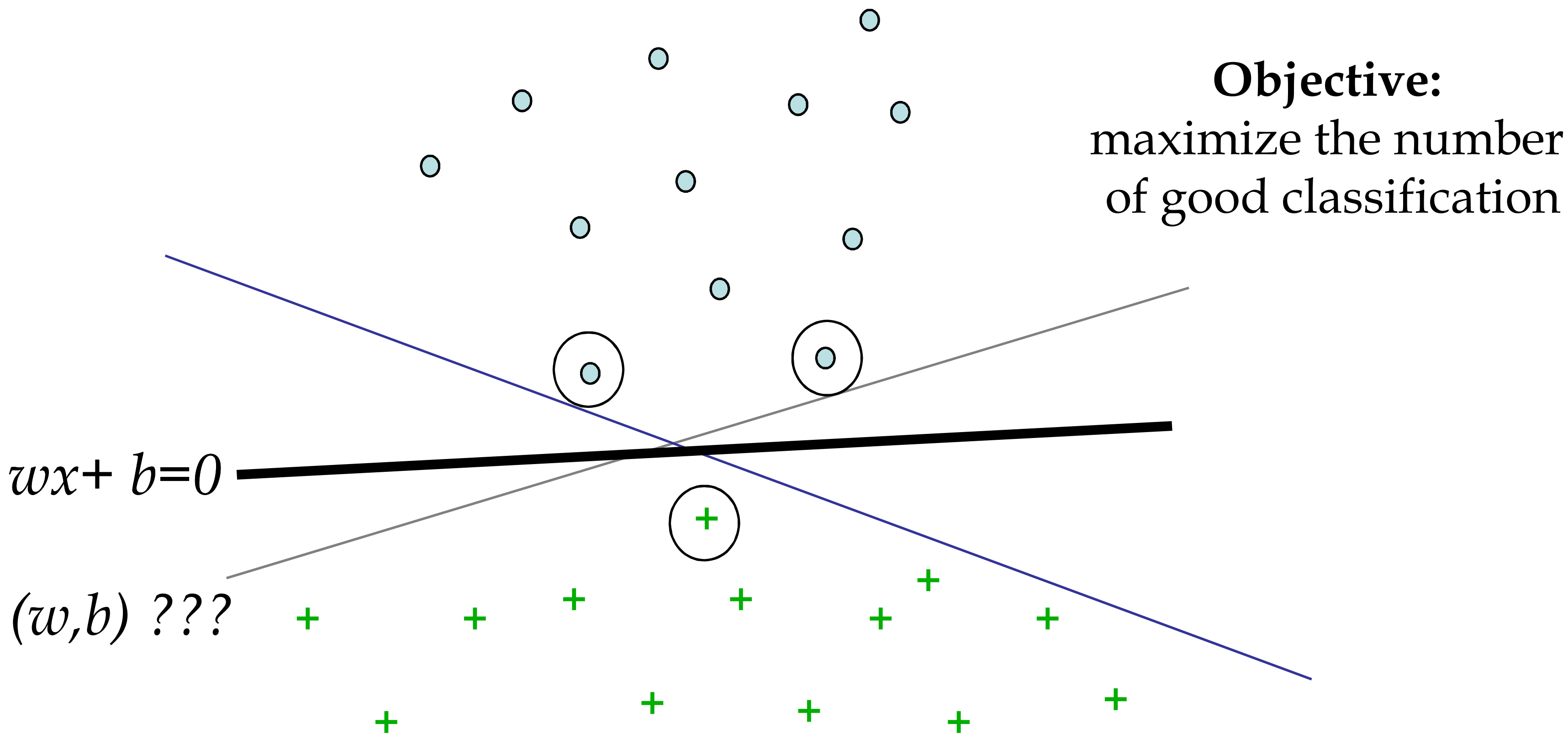
(w,b) ???

2-class problem

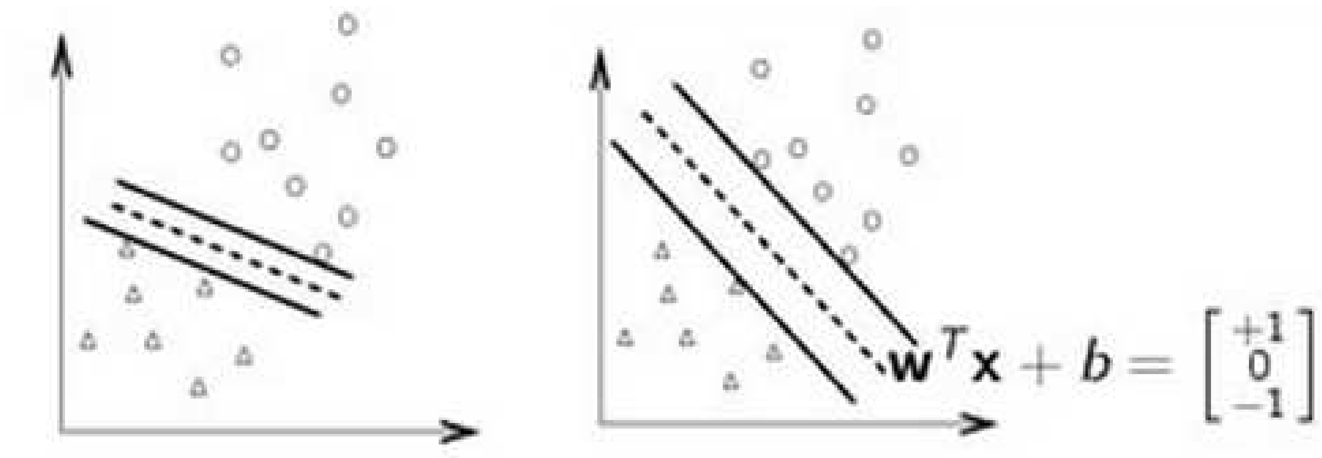


Objective:
maximize the number
of good classification

Particular examples in classification



2-class problem



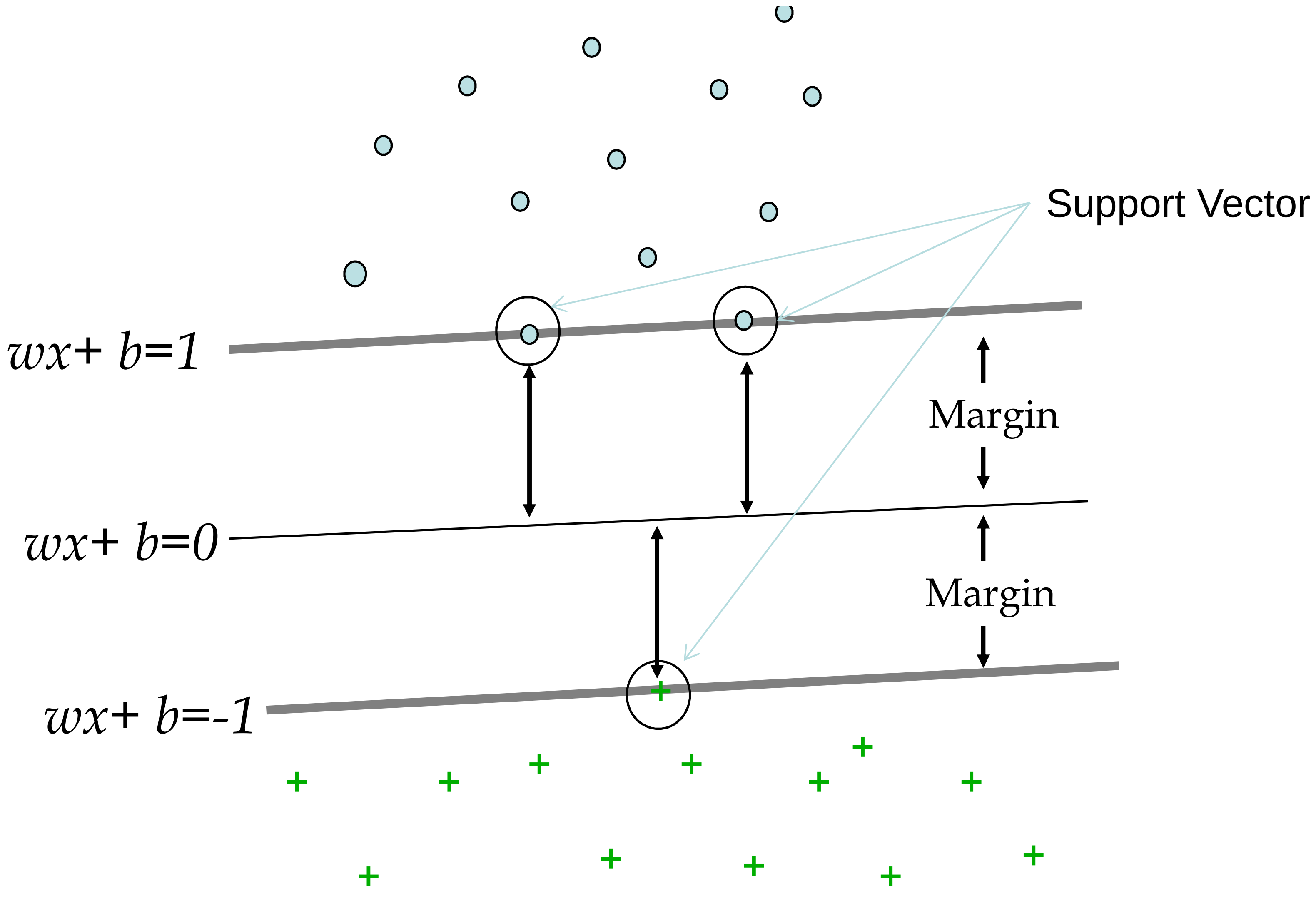
- A separating hyperplane: $\mathbf{w}^T\mathbf{x} + b = 0$

$$
\begin{array}{ll}
(\mathbf{w}^T\mathbf{x}_i) + b > 0 & \text{if } y_i = 1 \\
(\mathbf{w}^T\mathbf{x}_i) + b < 0 & \text{if } y_i = -1
\end{array}
$$

- Decision function $f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$, $\mathbf{x}$: test data

  Many possible choices of $\mathbf{w}$ and b

Maximal Margin



Maximal Margin

- Margin = distance between the support vector lines
- Linear$\rightarrow$ $f(x) = w.x + b = 0$
- Distance between a point and the hyperplane separator

  $$d(x) = |w.x + b|/||w||$$
- Margin size: $2/||W||$
- Maximize the margin = minimize $||w||$

Maximal Margin

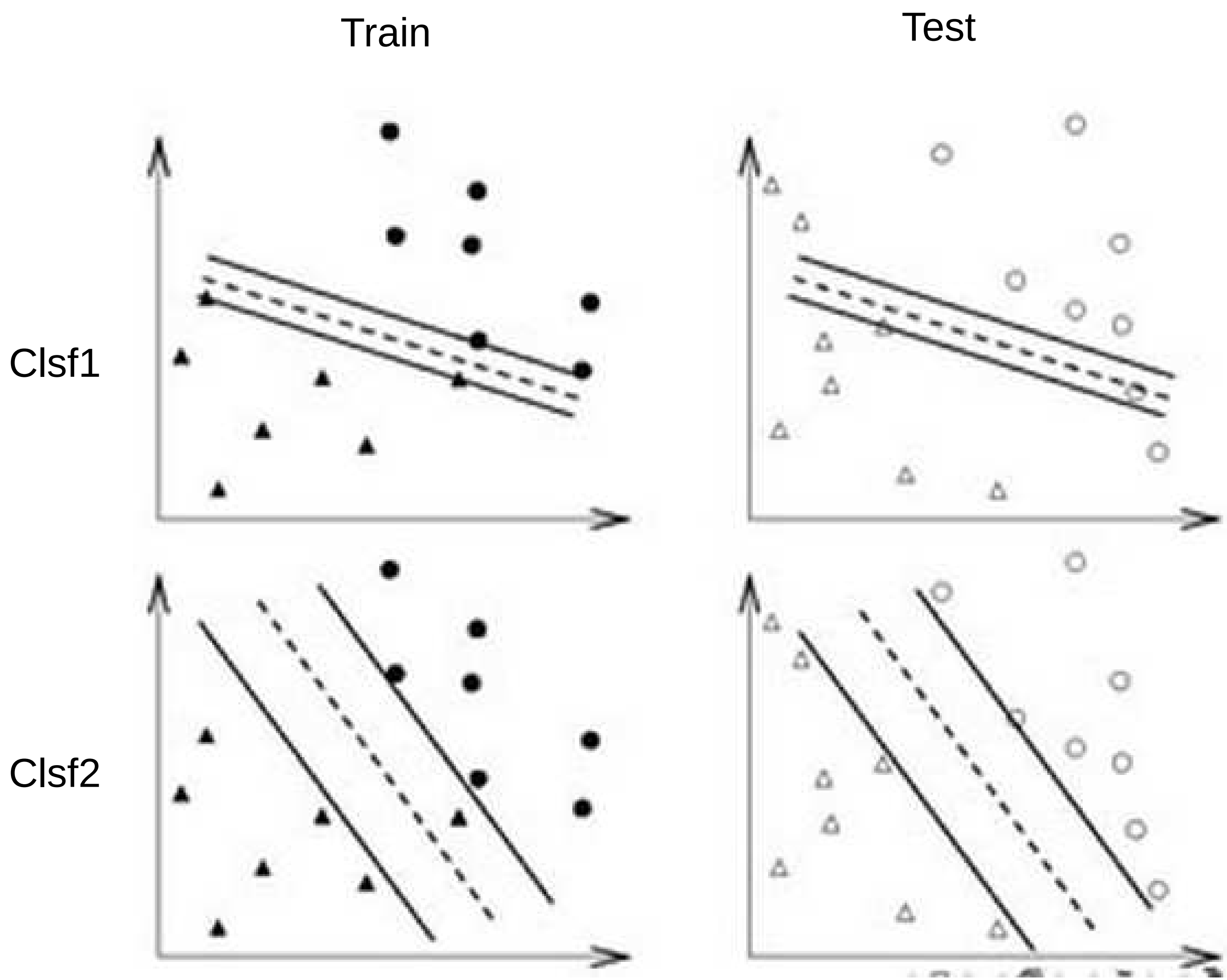
- Distance between $\mathbf{w}^T\mathbf{x} + b = 1$ and -1:

$$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}}$$

- A quadratic programming problem [Boser et al., 1992]

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \\ & i = 1, \ldots, l. \end{aligned}$$

Problem: not optimal



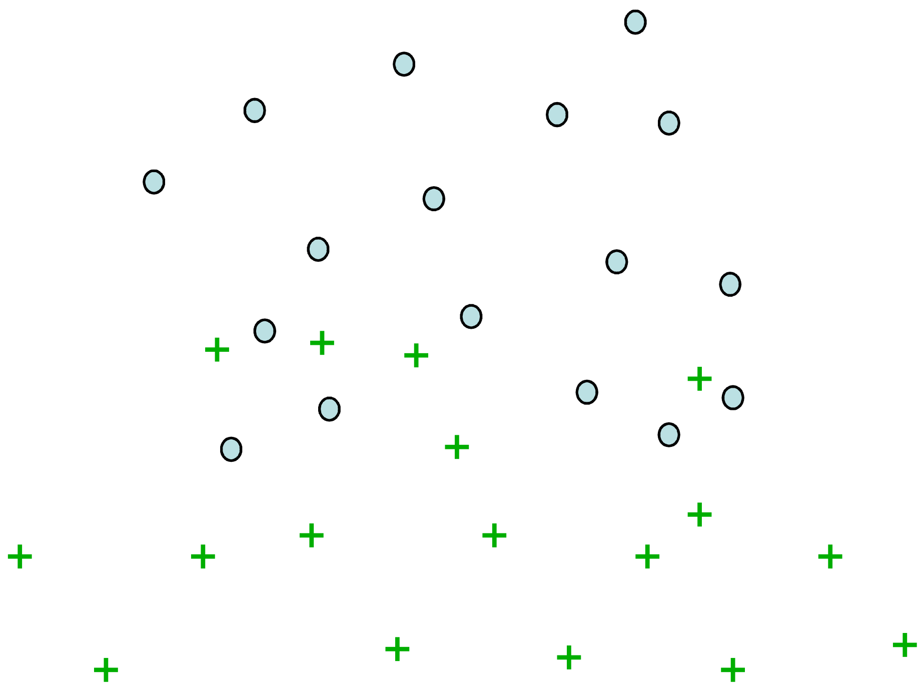
Slacking Variables

- It may be not possible or not optimal to separate perfectily the two classes in trainning set.
- We add slacking variables to make the model tolerant to the errors.
- C defines the tradeoff between margin and constraint.

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

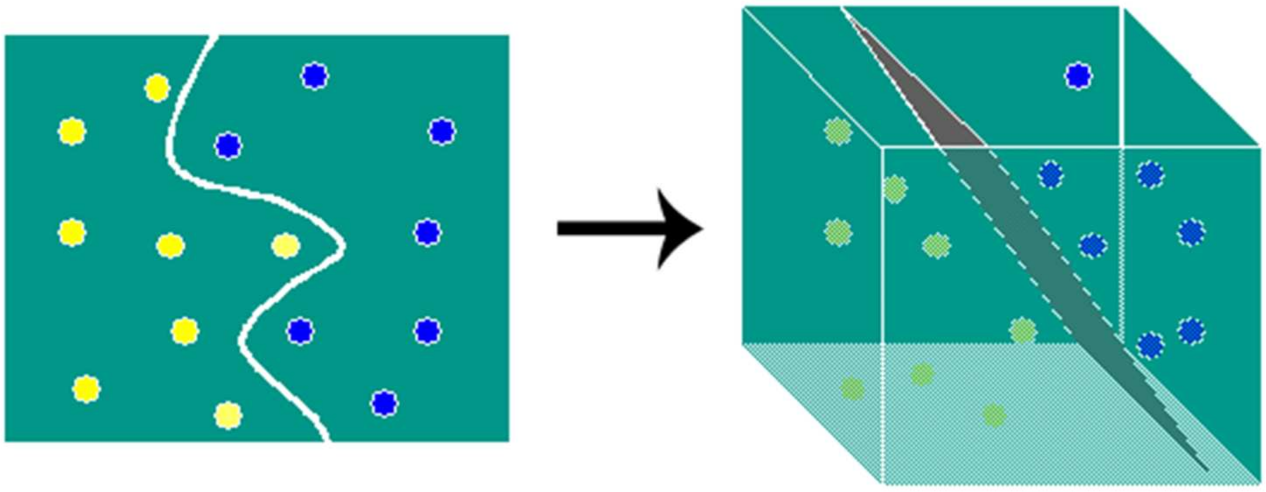
$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \ i = 1, \ldots, l.$$

Problem: not linearly separable



Kernel Trick

- Solving a non linear problem ?
- We change the dimensionality in using a kernel :
- The data are projected in a new higher dimension space
- A linear classifier is consructed in this kernel space



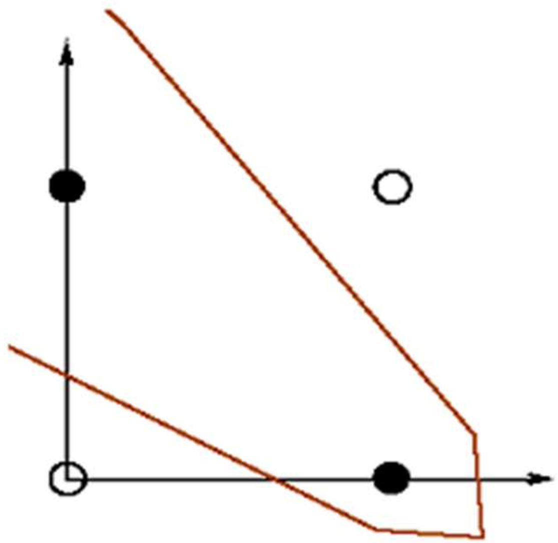
Kernel Trick: example

Example of the XOR problem:

The XOR problem is not linearly separable.

Let see in a 2-D space:

Points : (0,0) ; (0,1) ; (1,0) ; (1,1)



Kernel Trick: example

We take a polynomial kernel : (x , y) -> (x , y , x.y)

We project the example from a 2-D space to a 3-D space

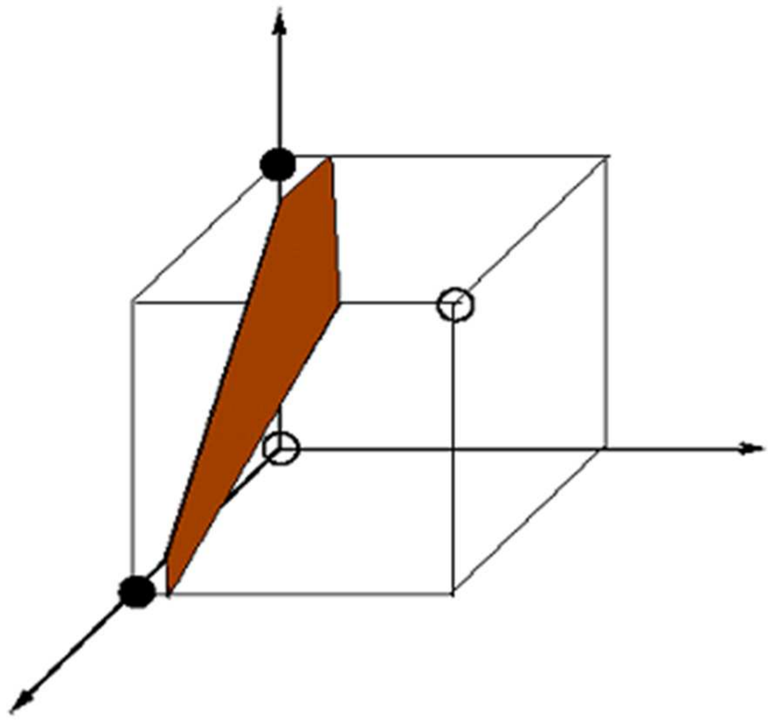
The class are linearly separable in this new space:

(0,0) -> (0,0,0)

(0,1) -> (0,1,0)

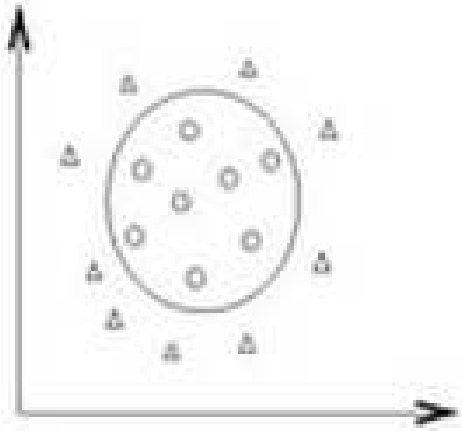
(1,0) -> (1,0,0)

(1,1) -> (1,1,1)



Kernel Trick

- An example:



- Allow training errors
- Higher dimensional (maybe infinite) feature space

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots).$$

SVM Formulation

- Standard SVM [Cortes and Vapnik, 1995]

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \dots, l. \end{aligned}$$

- Example: $\mathbf{x} \in R^3, \phi(\mathbf{x}) \in R^{10}$

$$\begin{aligned} \phi(\mathbf{x}) \ = \ & (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\ & x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3) \end{aligned}$$

Finding the decision function

- $\mathbf{w}$: maybe infinite variables
- The dual problem

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\
\text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l \\
& \mathbf{y}^T \boldsymbol{\alpha} = 0,
\end{aligned}
$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$

- At optimum

$$\boxed{\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)}$$

- A finite problem: #variables = #training data

Decision function

- At optimum

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)$$

- Decision function

$$\begin{aligned} & \mathbf{w}^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

- Only $\phi(\mathbf{x}_i)$ of $\alpha_i > 0$ used $\Rightarrow$ support vectors

Kernel function

- The problem and the solution depend only on the dot product $\Phi(x).\Phi(x')$
- In practice we do not choice Φ but K
  K: Xx$X \rightarrow R$ such that $k(x,x') = \Phi(x).\Phi(x')$
- K is called kernel function
- Pb: How to choice the kernel function

Mercer Condition

- K is a kernel ⇔ (k(xi,xj))i,j is a definite positive matrix
- K is symetric
- In this case the function Φ tq: k(x,x')= Φ(x).Φ(x') is defined
- Probleme:
  - These conditions are difficult to check
  - Φ stays unknown
  - Do not help for the choice of the kernel

Kernel example

General Kernel:

- Linéaire

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}'$$

- Polynomial

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}')^d \text{ ou } (c + \boldsymbol{x} \cdot \boldsymbol{x}')^d$$

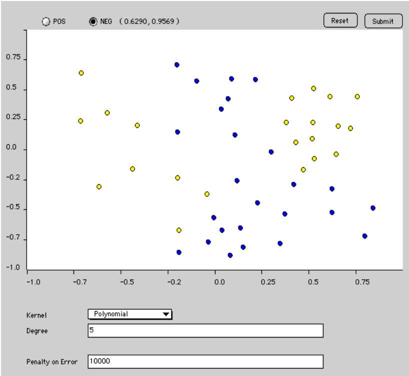
- Gaussien

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/\sigma}$$

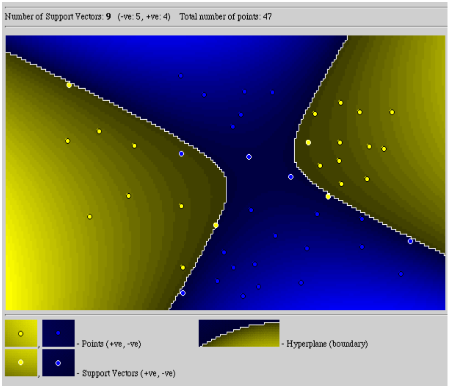
- Laplacien

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|_1/\sigma}$$

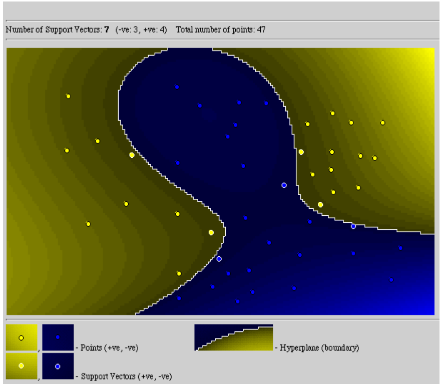
Kernel example



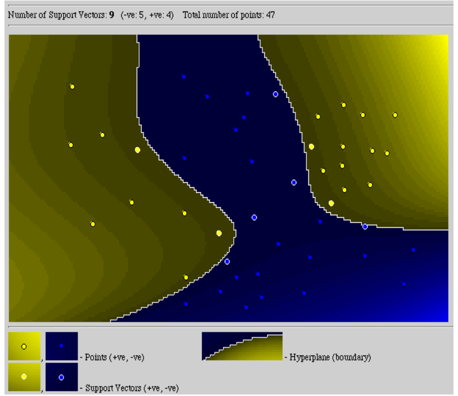
- 47 examples (22 +, 25 -)



(5-, 4+)

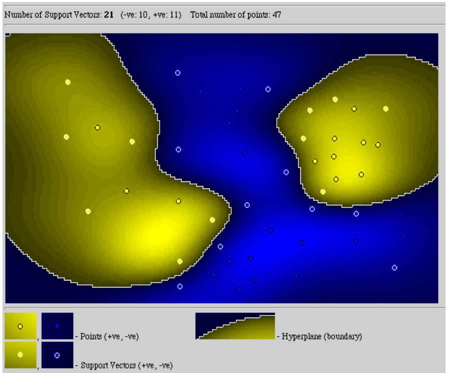


(3-, 4+)

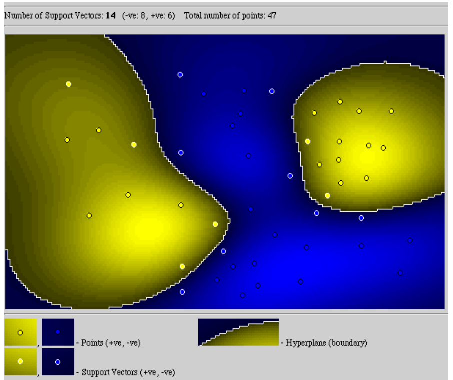


(5-, 4+)

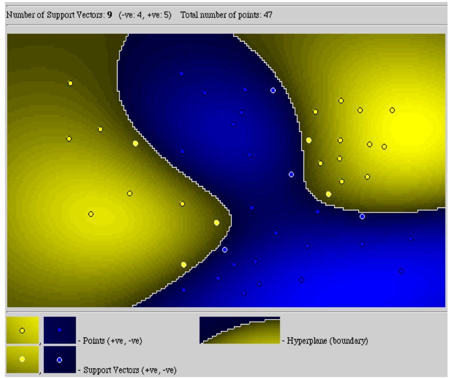
Polynomial function, degree=*2, 5*, 8 **and** *C* = *10000*



(10-, 11+)



(8-, 6+)



(4-, 5+)

Guassian function σ = *2, 5*, 10 **and** *C* = *10000*

Selecting Kernels

- For beginners, use Gaussian (RBF) first
- Linear Kernel: special case of RBF
- Polinomial: numerical difficulties

  More parameters than RBF
- Researchers design many kernels specialized to different domains
- We can combine different kernel to construct a more complex kernel

Parameters selection

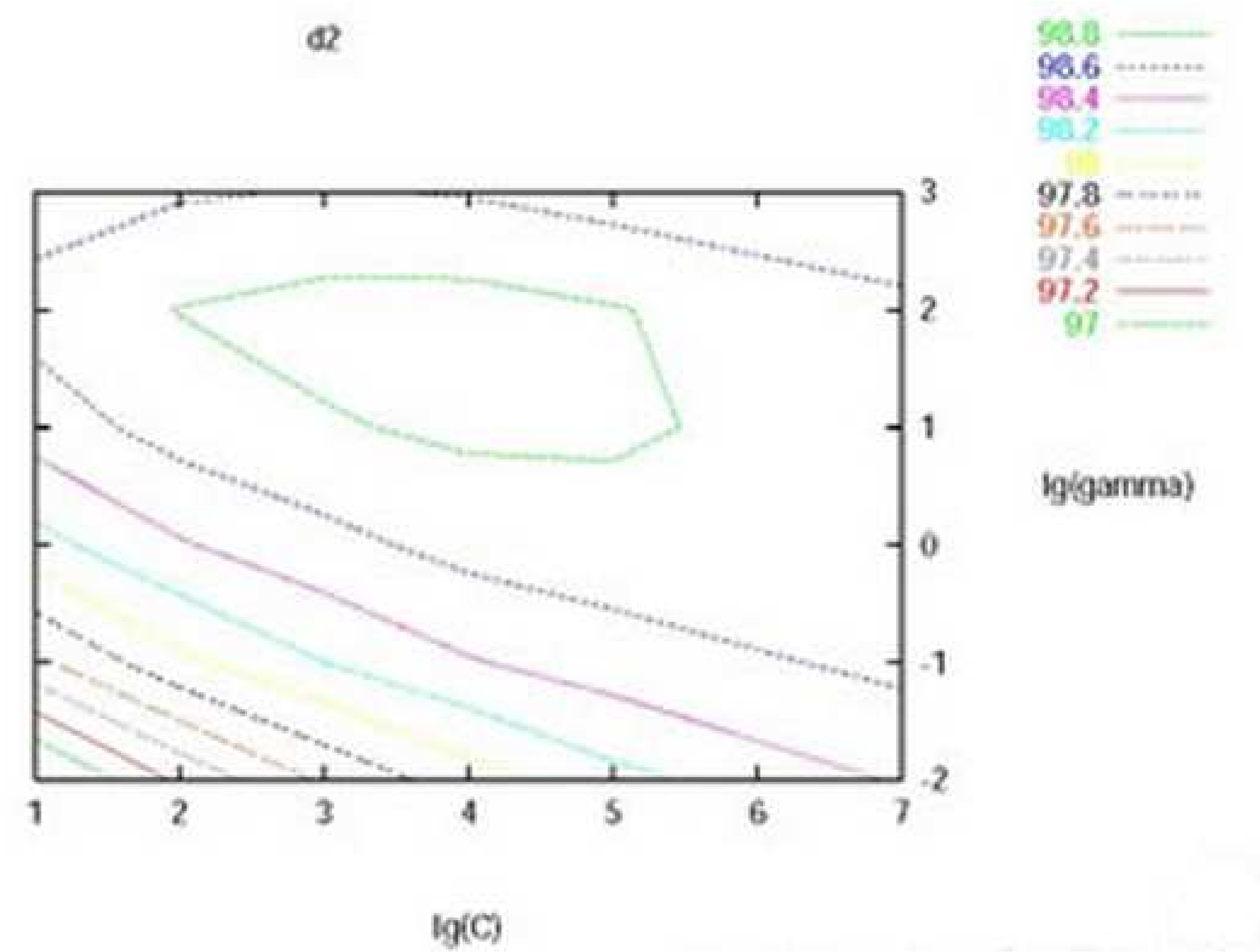
- Is important
- Now parameters are
  C, kernel parameters
- Example:

$$\gamma \text{ of } e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

$$a, b, d \text{ of } (\mathbf{x}_i^T \mathbf{x}_j / a + b)^d$$

- How to select them?
  So performance better?

Parameters selection



Parameters selection

- In practice
  Available data $\Rightarrow$ training and validation
- Train the training
- Test the validation
- k-fold cross validation:
  Data randomly separated to k groups
  Each time $k-1$ as training and one as testing
- Using CV on training + validation
- Predict testing with the best parameters from CV

Data scaling

- Problem: Features in greater numeric range may dominate the construction of the separator
- Scale the features to the range [0,1]

$$x^j{}_i = \frac{x^j{}_i - x^{min}{}_i}{x^{max}{}_i - x^{min}{}_i}$$

- Scalling generally helps but not always
- Same scaling factor have to be used on training and testind set

A simple procedure

1. Conduct simple scaling on the data
2. Consider RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$
3. Use cross-validation to find the best parameter C and γ
4. Use the best C and γ to train the whole training set
5. Test

For beginners only, you can do a lot more

Computing time

- n = Number of training example
- d = Number of features
- $dn^2 \leq$ complexity $\leq dn^3$
- Kernel matrix size= n^2
- => SVM can deal with problem up to 100.000 examples

SVM with R

- Function svm() du package e1071

```
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
    "radial", degree = 3, coef0 = 0, cost = 1, nu = 0.5,
    class.weights = NULL, cachesize = 40, tolerance =
    0.001, epsilon = 0.1, shrinking = TRUE, cross = 0,
    probability = FALSE, fitted = TRUE, ..., subset, na.action
    = na.omit)
```

SVM with R

x: a data matrix.

y: a response vector with one label for each row/component of x

Scale: A logical vector indicating the variables to be scaled. If scale is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally (both x and y variables) to zero mean and unit variance.

Type: svm can be used as a classification machine, as a regression machine. (C-classification,eps-regression)

SVM with R

Kernel: the kernel used in training and predicting.

- linear: *u'*v*
- polynomial: *(gamma*u'*v + coef0)^degree*
- radial basis: *exp(-gamma*|u-v|^2)*
- sigmoid: *tanh(gamma*u'*v + coef0)*

Degree, gamma, coef0: parameter needed for kernel

Cost: cost of constraints violation (default: 1)—it is the ‘C’-constant of the regularization term in the Lagrange formulation.

SVM with R

Class.weights: a named vector of weights for the different classes, used for asymetric class sizes.

Cross: if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model

Probability: logical indicating whether the model should allow for probability predictions.

Demo: http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml