Pattern Recognition
Support Vector Machines

Francesco Tortorella

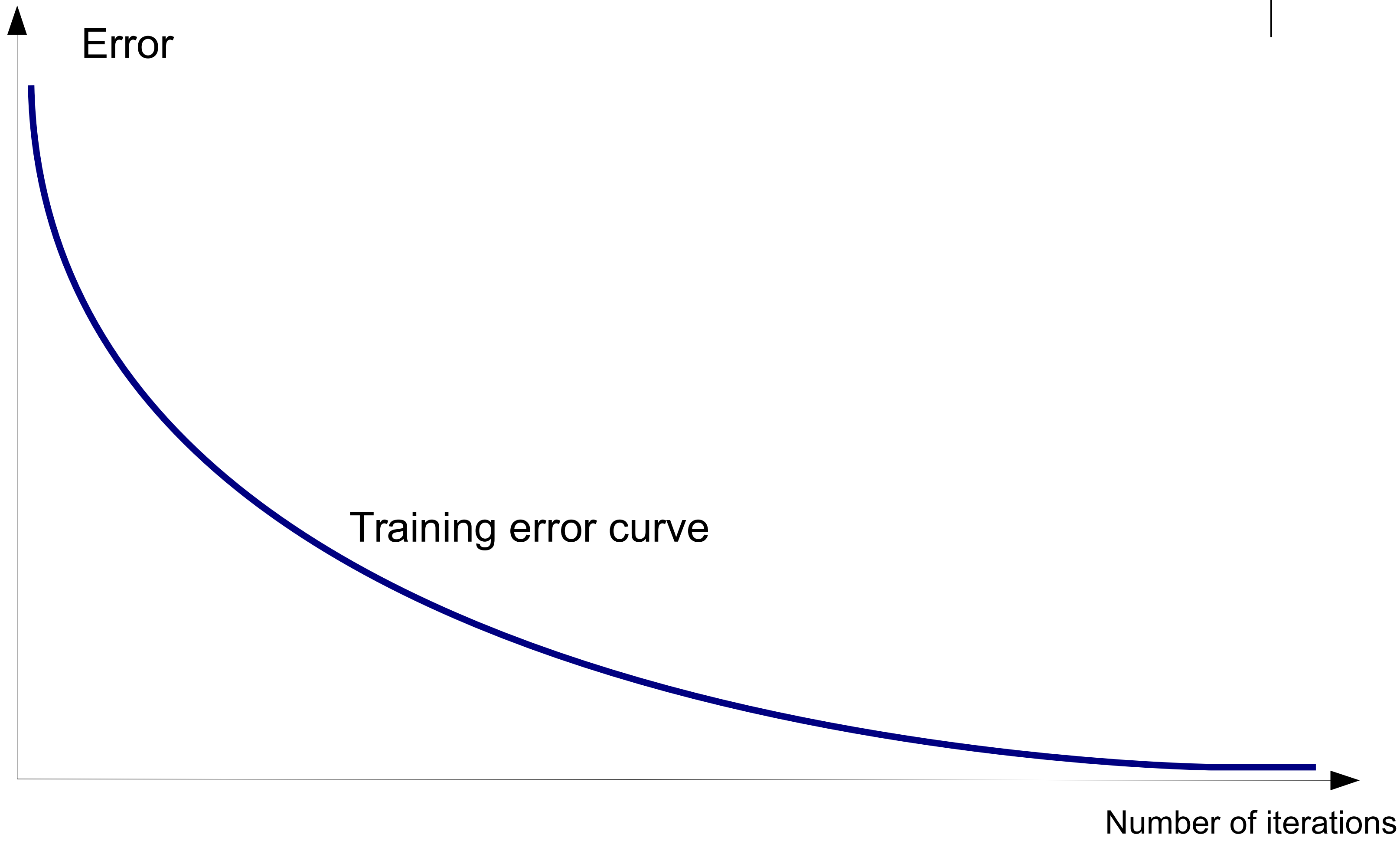
University of Cassino and Southern Latium

Cassino, Italy

Overfitting and underfitting

- The central question is that our learning algorithm must perform well on previously unseen inputs → **Generalization**
- When training a machine learning model, we can compute some error measure on the training set → **Training Error**
- How the training error varies during the training process?

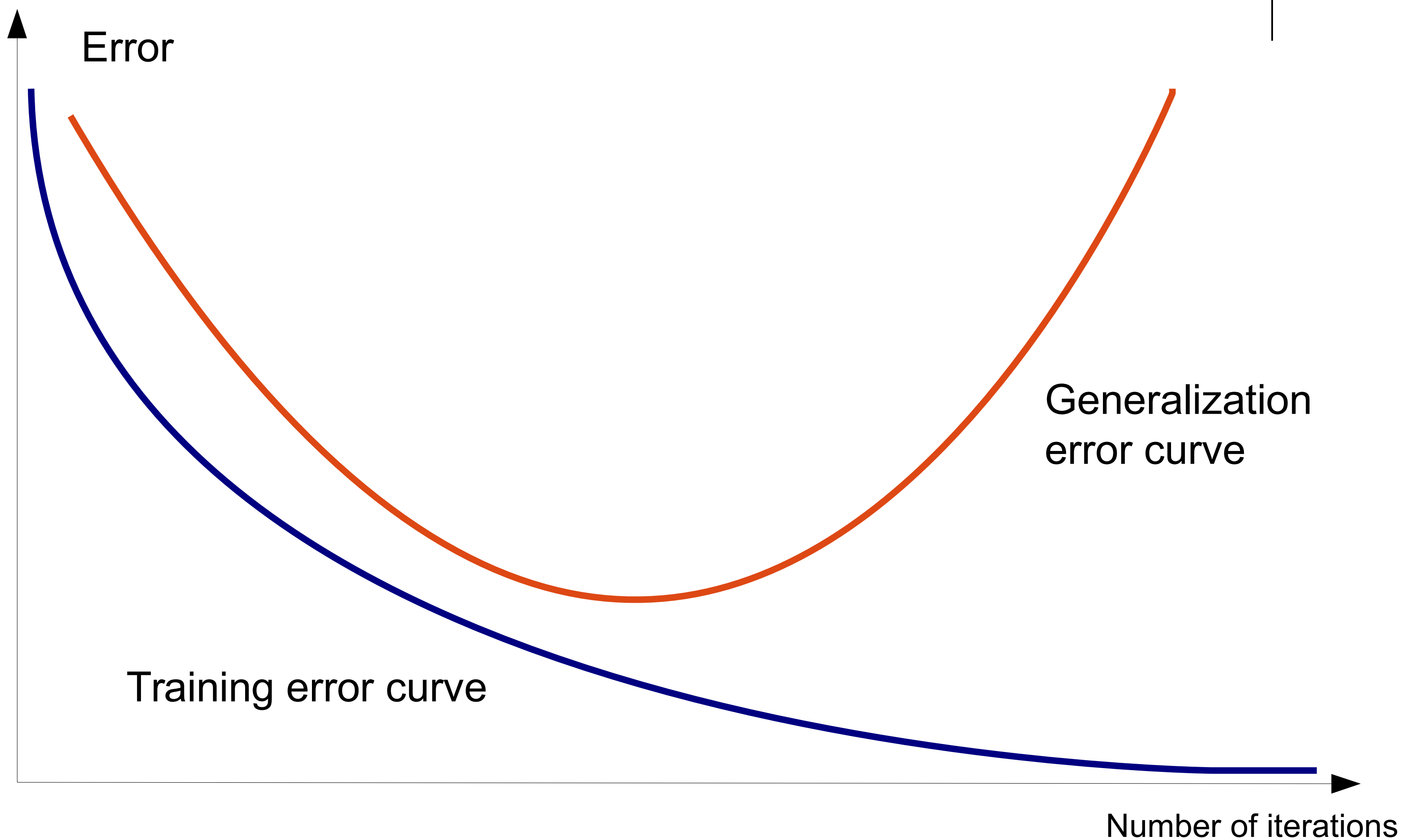
Overfitting and underfitting



Overfitting and underfitting

- We could go on until the training error is low, but we want that the **Test Error** (or **Generalization Error**) must be low as well.
- We tipically estimate the generalization error on a test set containing samples collected separately from the training set.
- What if we observe how the generalization error varies during the training phase?

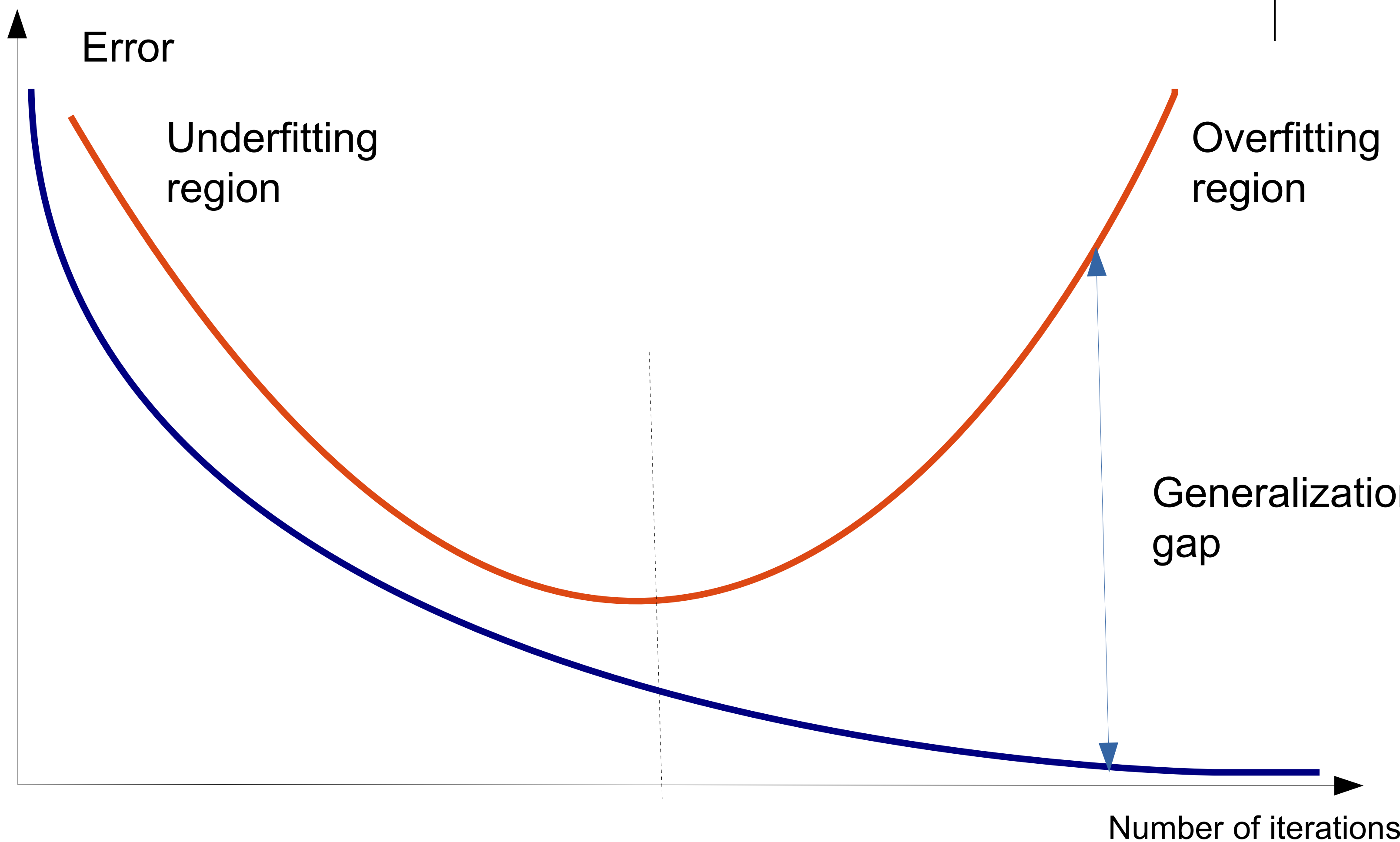
Overfitting and underfitting



Overfitting and underfitting

- We can observe two different problems.
- **Underfitting**: when the model is not able to obtain a sufficently low error on the training set
- **Overfitting**: when the gap between the training error and the test error is too large

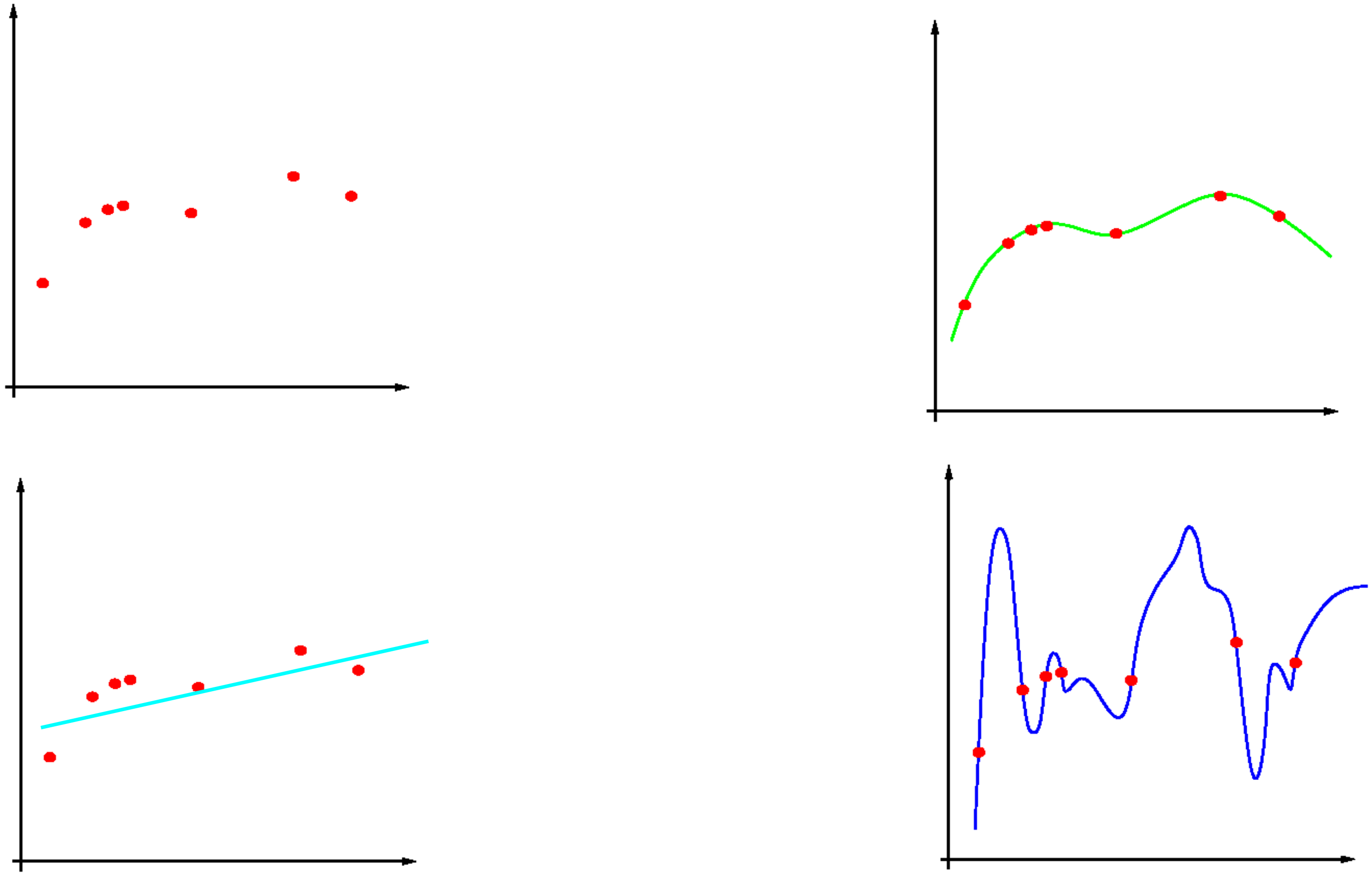
Overfitting and underfitting



Overfitting, underfitting, capacity

- We can control if a model is more likely to underfit or to overfit by altering its capacity.
- Informally, the capacity is the ability of a model to fit a wide variety of functions.
- One way to control the capacity of a learning algortihm is by choosing its hypothesis space, the set of functions that the algorithm is capable to select as the solution.

Overfitting, underfitting, capacity



Measuring the capacity VC Dimension

- Statistical Learning Theory provides some measures fo evaluating the capacity of a model
- The most well known measure is the **Vapnik-Cervonenkis** dimension (VC dimension) that measures the capacity of a binary classifier
- It is defined as the largest possible value of *m* for which there exists some training set of *m* different samples that the classifier can label arbitrarily.

VC Dimension

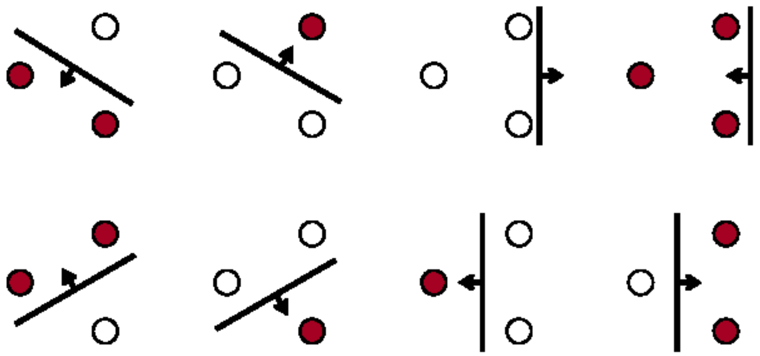
2.1. The VC Dimension

The VC dimension is a property of a set of functions $\{f(\alpha)\}$ (again, we use α as a generic set of parameters: a choice of α specifies a particular function), and can be defined for various classes of function f. Here we will only consider functions that correspond to the two-class pattern recognition case, so that $f(\mathbf{x}, \alpha) \in \{-1, 1\}\ \forall \mathbf{x}, \alpha$. Now if a given set of l points can be labeled in all possible 2^l ways, and for each labeling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that that set of points is *shattered* by that set of functions. The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum number of training points that can be shattered by $\{f(\alpha)\}$. Note that, if the VC dimension is h, then there exists at least one set of h points that can be shattered, but it in general it will not be true that *every* set of h points can be shattered.

C.J.C. Burges
A Tutorial on Support Vector Machines for Pattern Recognition
1998

VC dimension

- Example:
  - Binary classification problem in R^2
  - $f(\alpha)$ family of the oriented hyperplanes (perceptron)
- With *m*=3 there is some set for which it is possible to label arbitrarily all the samples



- With *m*=4 it is not possible for any set of samples (xor problem).
- The VC dimension of $f(\alpha)$ in R^2 is 3.
  - In general, in R^D $VC(f(\alpha))=D+1$

Learning and capacity

- Which link between capacity and learning?
- Our goal is to learn a function $f : X \to \{-1, +1\}$ from the labelled samples of a limited training set
- In other words, we want learn f from the training set

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in X \times \{-1, +1\}$$

Learning and capacity

- If we assume that data is generated from some unknown (but fixed) probability distribution $P(\mathbf{x}, y)$, the goal is to minimize the ***expected error*** (or ***expected risk***) on a test set, also drawn from $P(\mathbf{x}, y)$

$$R[f] = \int \frac{1}{2} |f(\mathbf{x}) - y| dP(\mathbf{x}, y)$$

Learning and capacity

- Actually, we cannot minimize the expected risk, because $P(\mathbf{x}, y)$ is unknown.
- What we could do is to minimize instead the average risk over the training set (***empirical risk***):

$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |f(\mathbf{x}_i) - y_i|$$

Limiting the expected risk

- Minimizing the empirical risk (training error), does not imply a small test error.
- The following bound has been demonstrated (Vapnik) :

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h(log(2N/h)+1) - log(\eta/4)}{N}}$$

with probability $1 - \eta$

- h is the VC dimension of the function f while N is the number of samples in the training set.

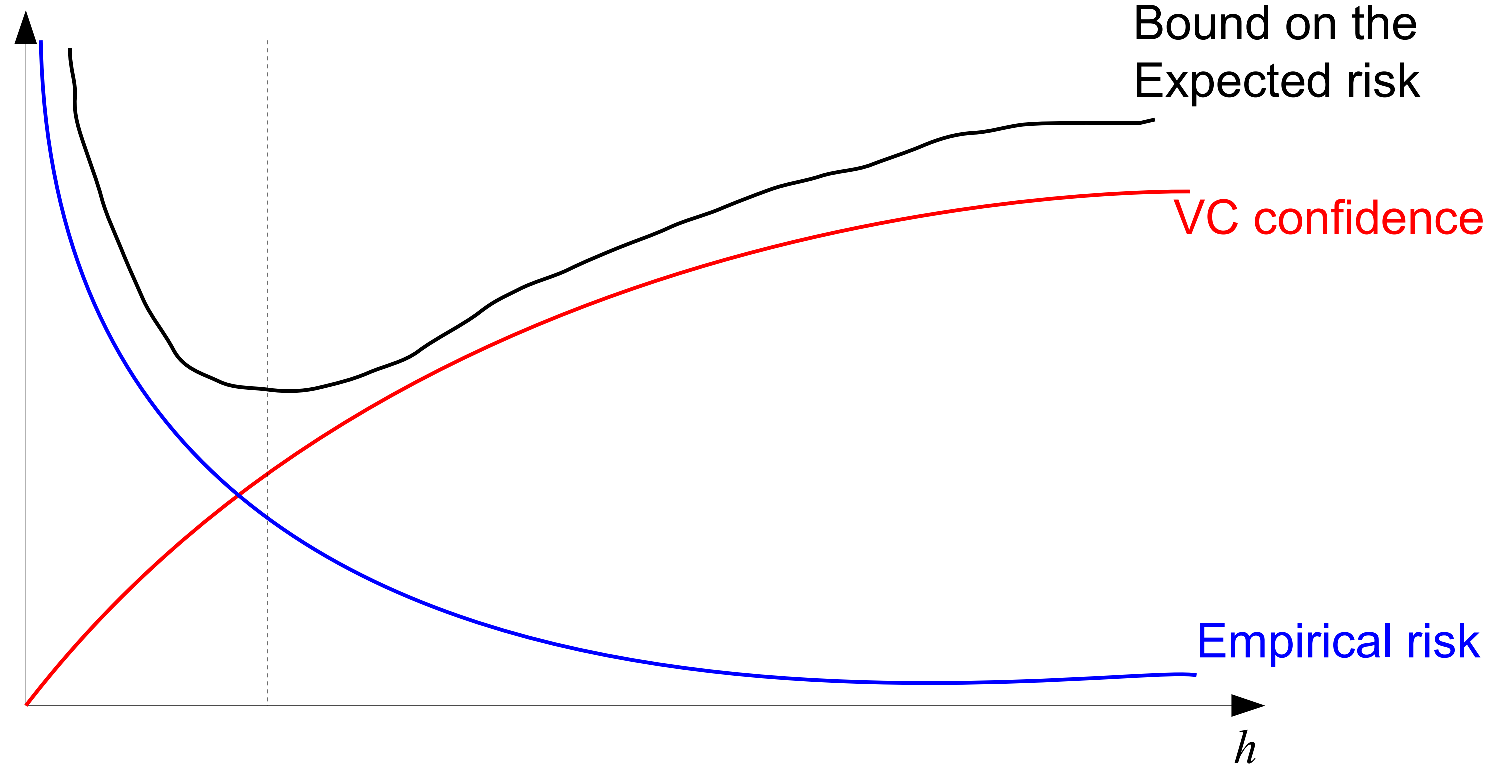
Limiting the expected risk

- The second term on RHS is called VC *confidence* and increases with the VC dimension h.
- In order to limit the expected risk, we have to minimize the sum on RHS, i.e. we have to minimize both
  - Empirical risk
  - VC confidence

Structural Risk Minimization

Limiting the expected risk

Conflicting targets



Limiting the expected risk

- The SRM cannot be directly achieved in many situations
  - VC dimension difficult to evaluate
  - Very difficult optimization problem
  - Very large bound
- Possible in the set of linear models

Support Vector Machines

- Let us consider a 2 class problem for which we have available a training set $S=\{x_i,y_i\}$, where:
  - x_i is the feature vector of the i-th sample
  - y_i is the label of the class to which the i-th sample belongs
- In order to simplify the expressions, assume that the labels are ± 1.
- Assume that the two classes are "linearly separable". This means that it exists an hyperplane $w \cdot x+b=0$ such that:

$$w \cdot x_i+b > 0 \text{ if } y_i=+1$$

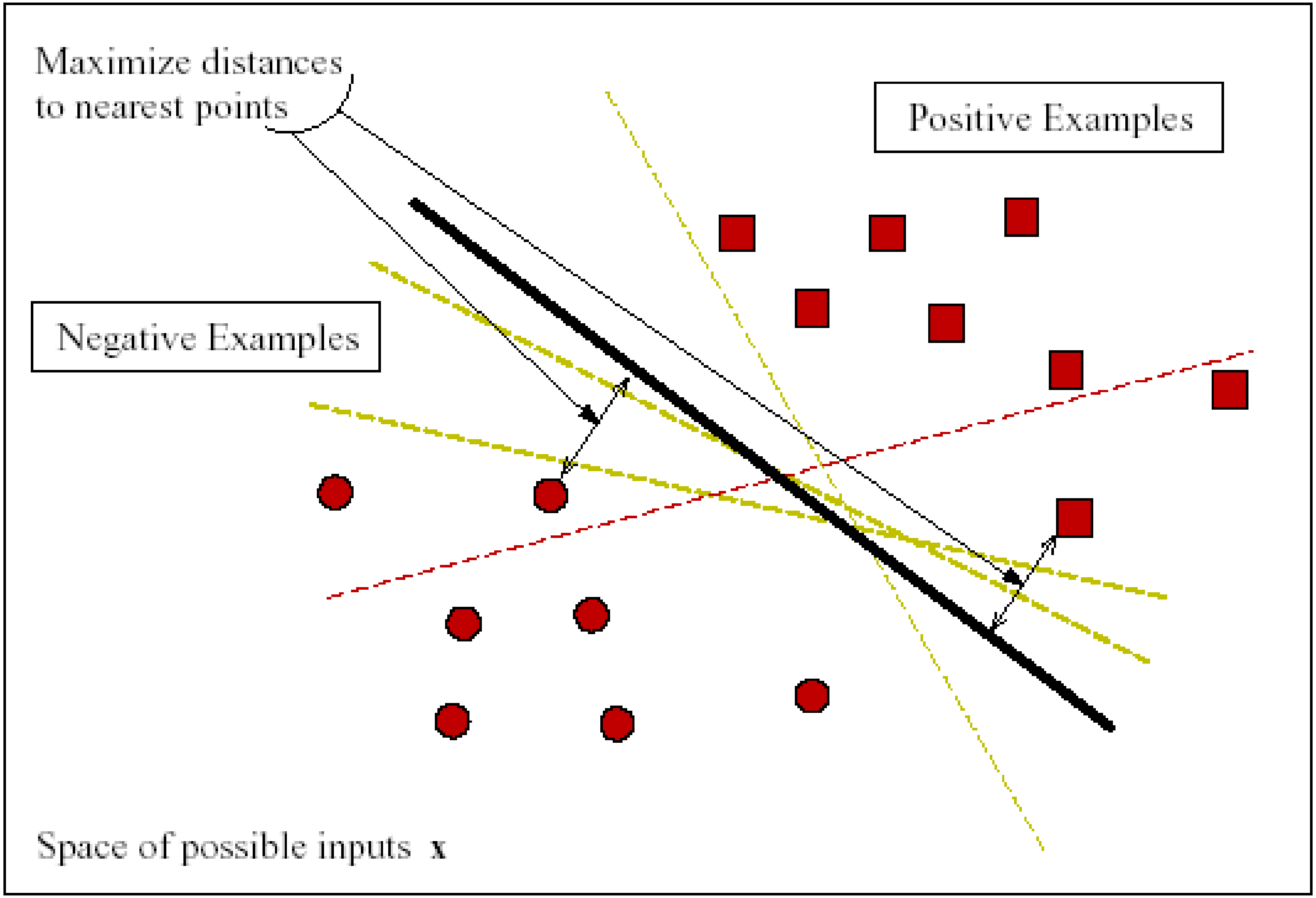
$$w \cdot x_i+b < 0 \text{ if } y_i= -1$$

$$\Longrightarrow \quad (w \cdot x_i+b)\, y_i > 0 \text{ for each } (x_i,y_i)$$

Support Vector Machines

- Consider the point x_+ (x_-) with label +1 (-1) that is the nearest to the hyperplane; call d_+ (d_-) such distance and define *margin* of the hyperplane the sum of the distances $d_+ + d_-$.
- If the two classes are linearly separable, there are several hyperplanes separating the classes.
- We consider the hyperplane with the maximum margin.

The maximum margin hyperplane

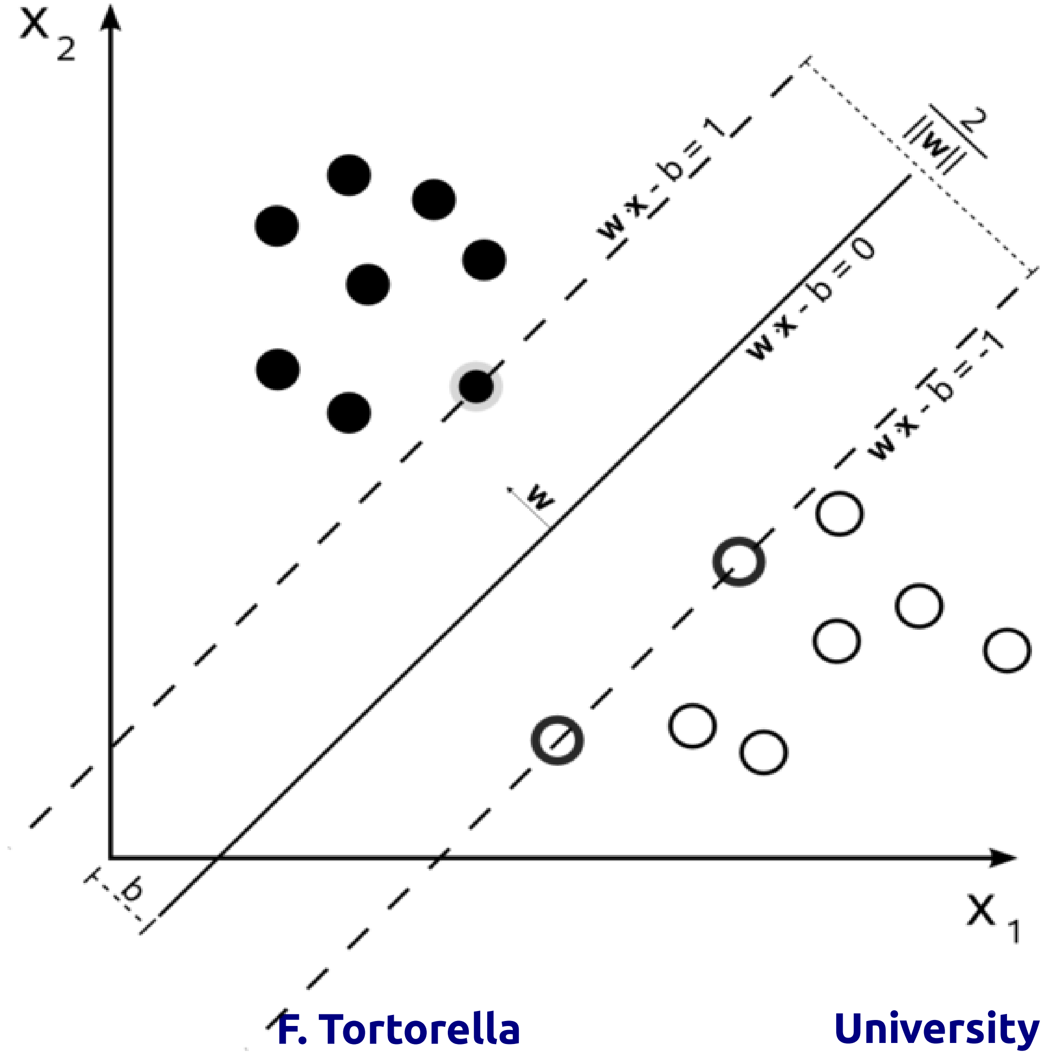


.L.

The maximum margin hyperplane

- We can scale w and b in such a way that, in correspondence of the nearest points, we have $w \cdot x_+ + b = +1$ e $w \cdot x_- + b = -1$.
- In this case $d_+ = d_- = 1/||w||$ and the margin becomes $2/||w||$.

The maximum margin hyperplane



The maximum margin hyperplane

- Vapnik demonstrated that the VC dimension of a separating hyperplane with a margin m is bounded as follows

$$h \leq min\left(\left\lceil \frac{R^2}{m^2} \right\rceil, d\right) + 1$$

where d is the dimensionality of the input space, and R is the radius of the smallest sphere containing all the input vectors

- By maximizing the margin we are thus minimizing the VC dimension.
- Since the separating hyperplane has zero empirical error (it correctly separates all the training examples), maximizing the margin will also minimize the upper bound on the expected risk

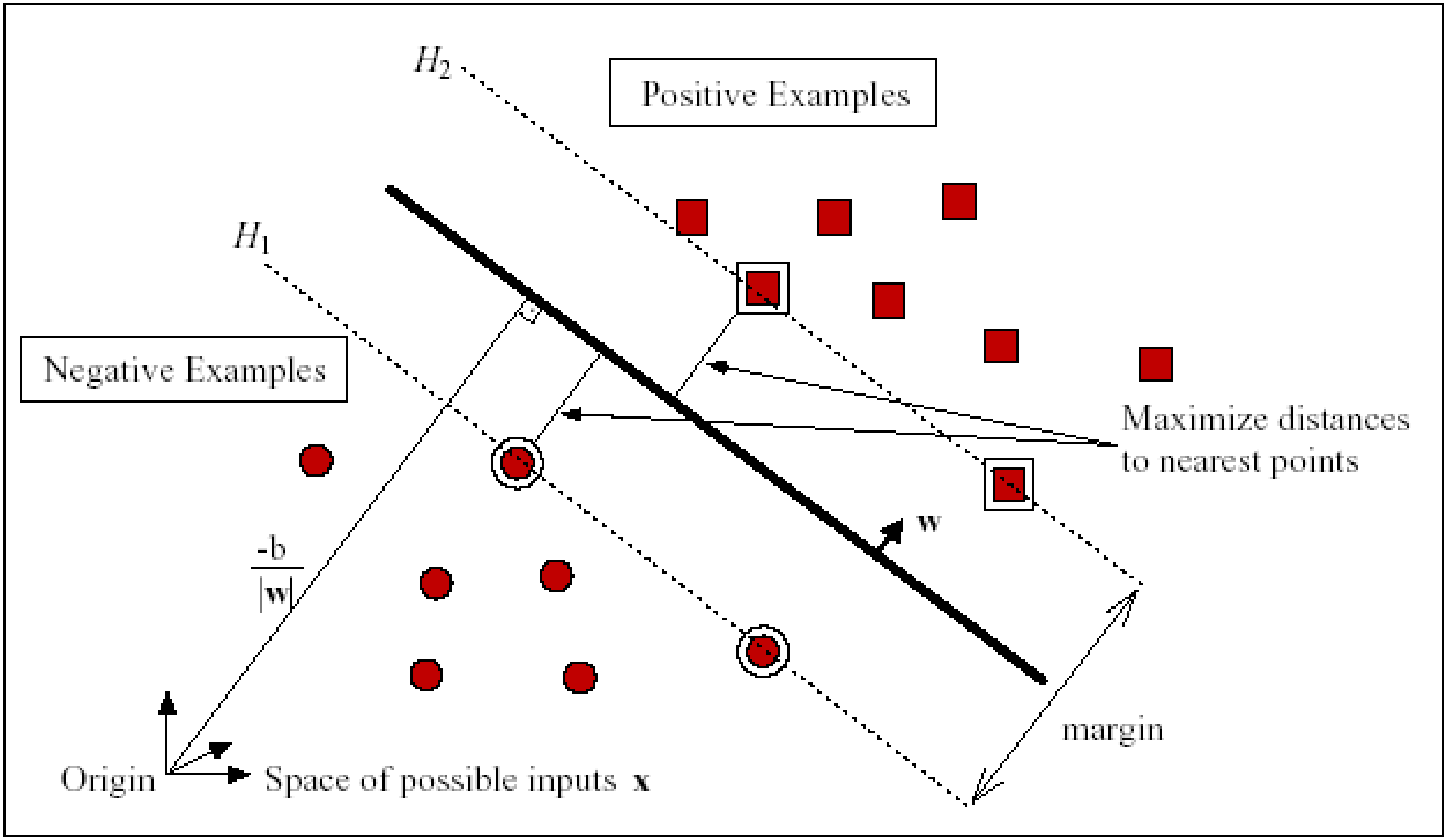
Support Vector Machine

- As a consequence, the best possible generalization is given by the maximum margin hyperplane, that is the *optimal separating hyperplane* (OSH).
- The LDF corresponding to the OSH is called *Support Vector Machine* (SVM).

Support Vector Machine

- By construction, we have $(w \cdot x_i + b)\, y_i - 1 \geq 0$ for each (x_i, y_i).
- The nearest points satisfy the equation $(w \cdot x_i + b)\, y_i - 1 = 0$ that specifies two hyperplanes H_1 and H_2 parallel to the OSH (no man's land).
- The points on H_1 and H_2 are the *support vectors* (SV). If the SVs change, the OSH is modified.

OSH and Support Vectors



Building the OSH

To obtain the OSH we must solve the optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

Margin maximization

subject to $$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \geq 0 \qquad \forall i$$

Empirical risk minimization

This leads to the minimization of the Lagrangian:

$$L_P \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{\ell} \alpha_i y_i \left(\mathbf{w}\cdot\mathbf{x}_i + b\right) + \sum_{i=1}^{\ell} \alpha_i, \qquad \alpha_i \geq 0.$$

Building the OSH

- This is a convex quadratic programming problem with solution :

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

- In the expression above only some Lagrange multipliers α_i will be greater than zero (*sparsness*).
- As a consequence, only the corresponding points of the training set will be support vectors and affect the position of the OSH.

Building the OSH

- At the end, the LDF will be:

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

$$f(\mathbf{x}) = \left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x_i} \right) \cdot \mathbf{x} + b$$

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i \left(\mathbf{x_i} \cdot \mathbf{x} \right) + b$$

Non separable classes

- In principle, the SVM cannot handle problems where the classes are not separable.
- Two possible (not mutually exclusive) approaches:
  - Relaxing the correct classification costraints and accepting a certain number of errors on the training set
  - Using non-linear discriminant functions (???)

Relax!

In the first approach, the correct classification constraints are relaxed by introducing positive *slack variables* ξ_i:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{for } y_i = +1$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{for } y_i = -1$$

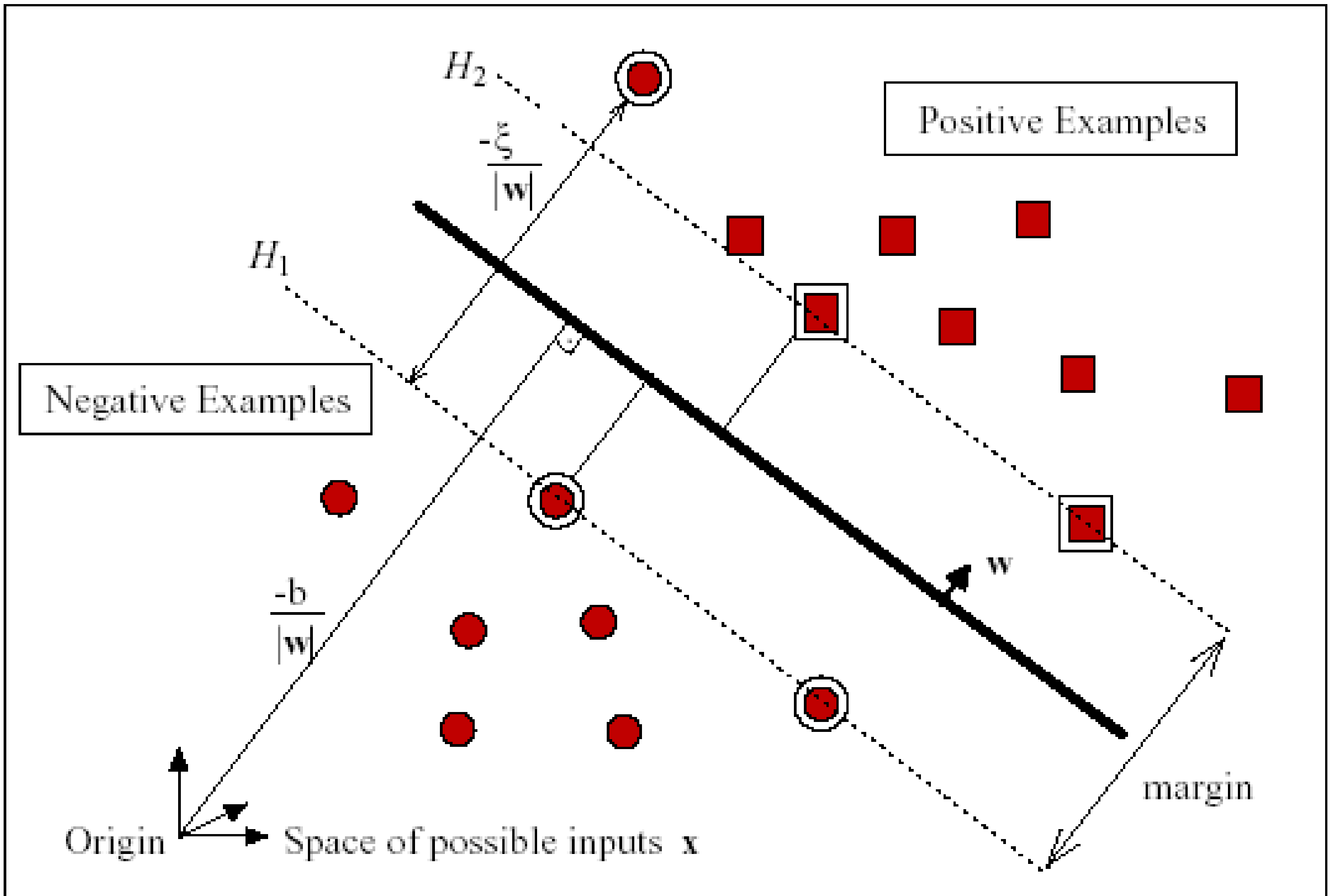
$$\xi_i \geq 0, \qquad \forall\, i.$$

For an error to occur, the corresponding ξ_i must exceed unity.

Relax!

- Depending on the value of the corresponding ξ_i the points of the training set will be
  - placed beyond the hyperplanes H_1 and H_2 and correctly classified ($\xi_i=0$)
  - placed between the hyperplanes H_1 and H_2 and correctly classified ($0<\xi_i<1$)
  - placed beyond the opposite hyperplane and erroneously classified ($\xi_i>1$)

OSH with *soft margins*



S.L.

Building the OSH with soft margins

- When introducing the slack variables, the Lagrangian becomes:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i$$

- The parameter C is chosen by the user to assign a penalty to errors.

Building the OSH with soft margins

- The solution obtained for the OSH is in the same form of the previous case:

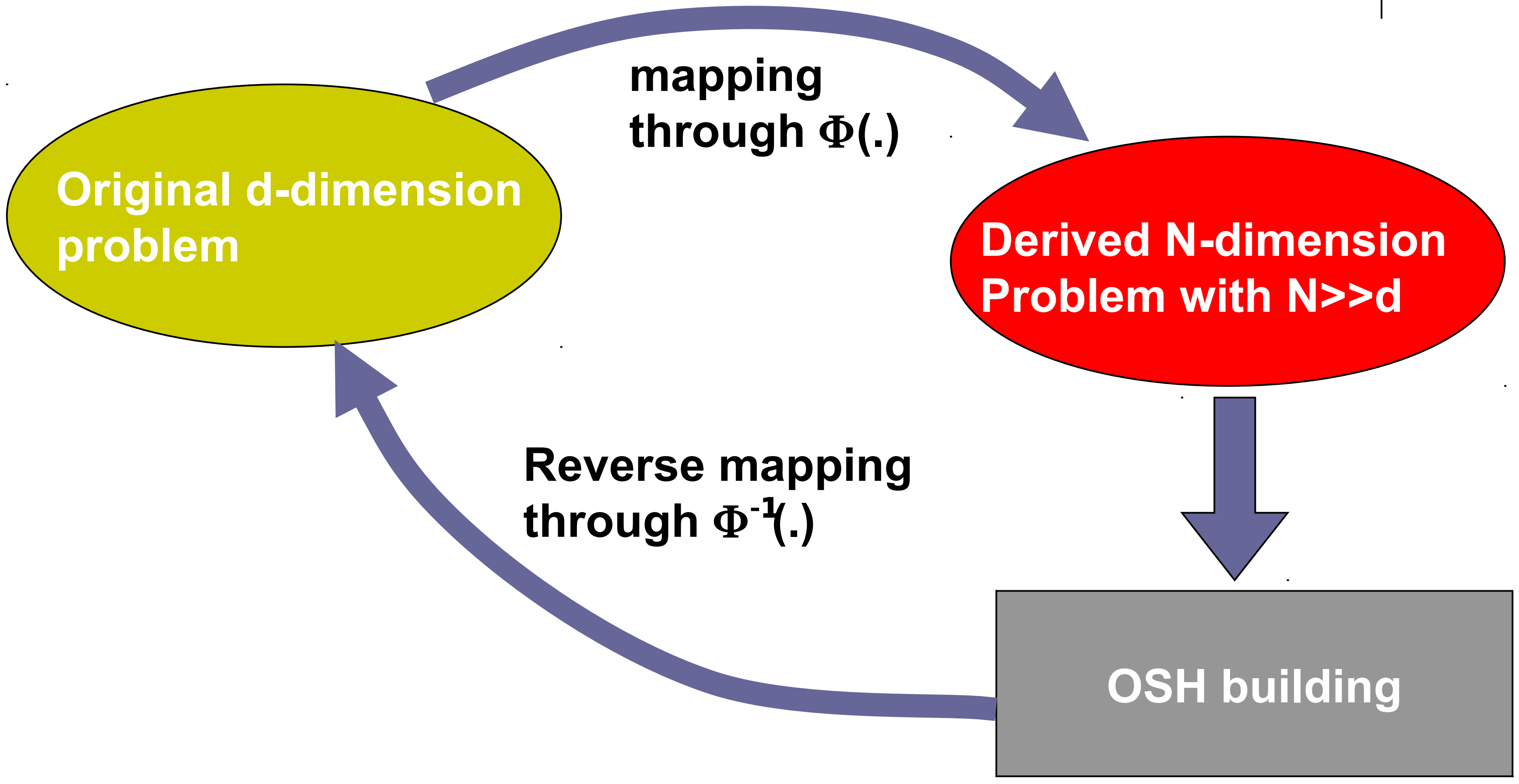
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

- The only difference is in the values of the Lagrange multipliers α_i that are $0 \leq \alpha_i \leq C$.

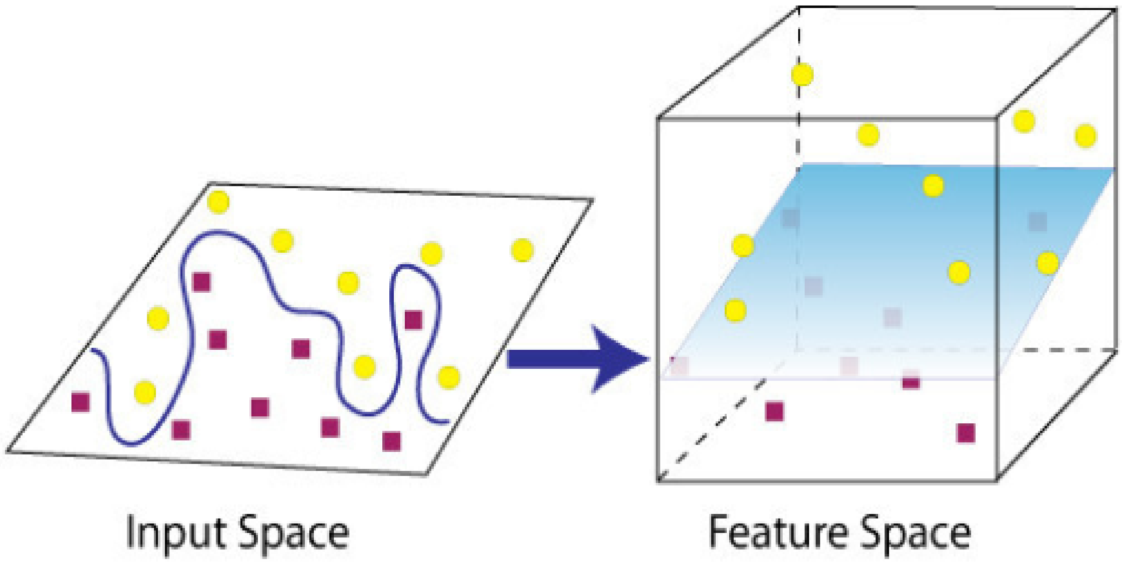
Non-linear SVM (???)

- Another possible approach is to consider a mapping $\Phi(x)$ from the feature space to another space with much higher dimension where the corresponding subsets are linearly separable.
- In this way, the classifier is still linear but in a different space.
- Depending on the dimension of the space in which the original problem was formulated, the mapping can lead to transformed space with very high dimensions ($\sim 10^6$).

Non linear SVM



Non-linear SVM



Feature Transformation

Non-linear SVM

- The Langragian becomes:

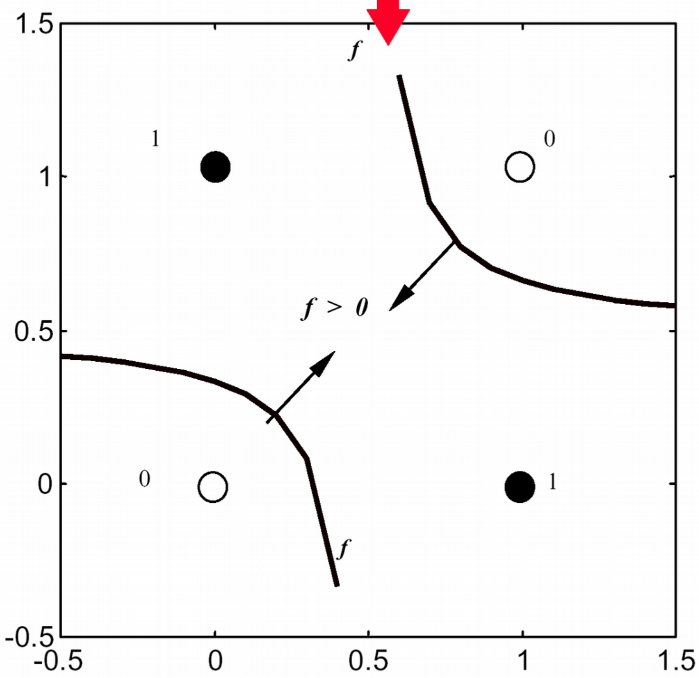
$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \alpha_i \left(y_i \left(w \cdot \Phi(x_i) + b \right) - 1 \right)$$

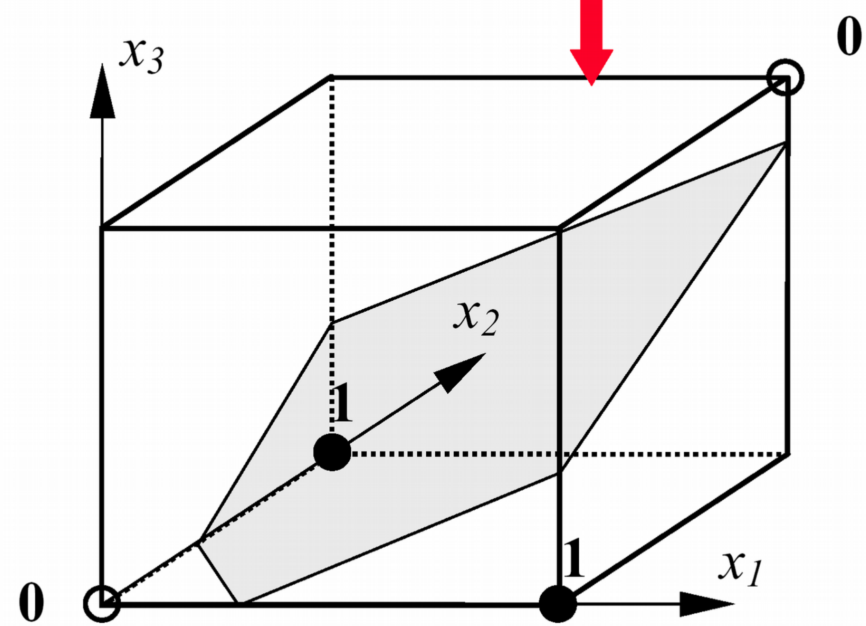
and has solution: $w = \sum_i \alpha_i y_i \Phi(x_i)$

The discriminant function is:

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

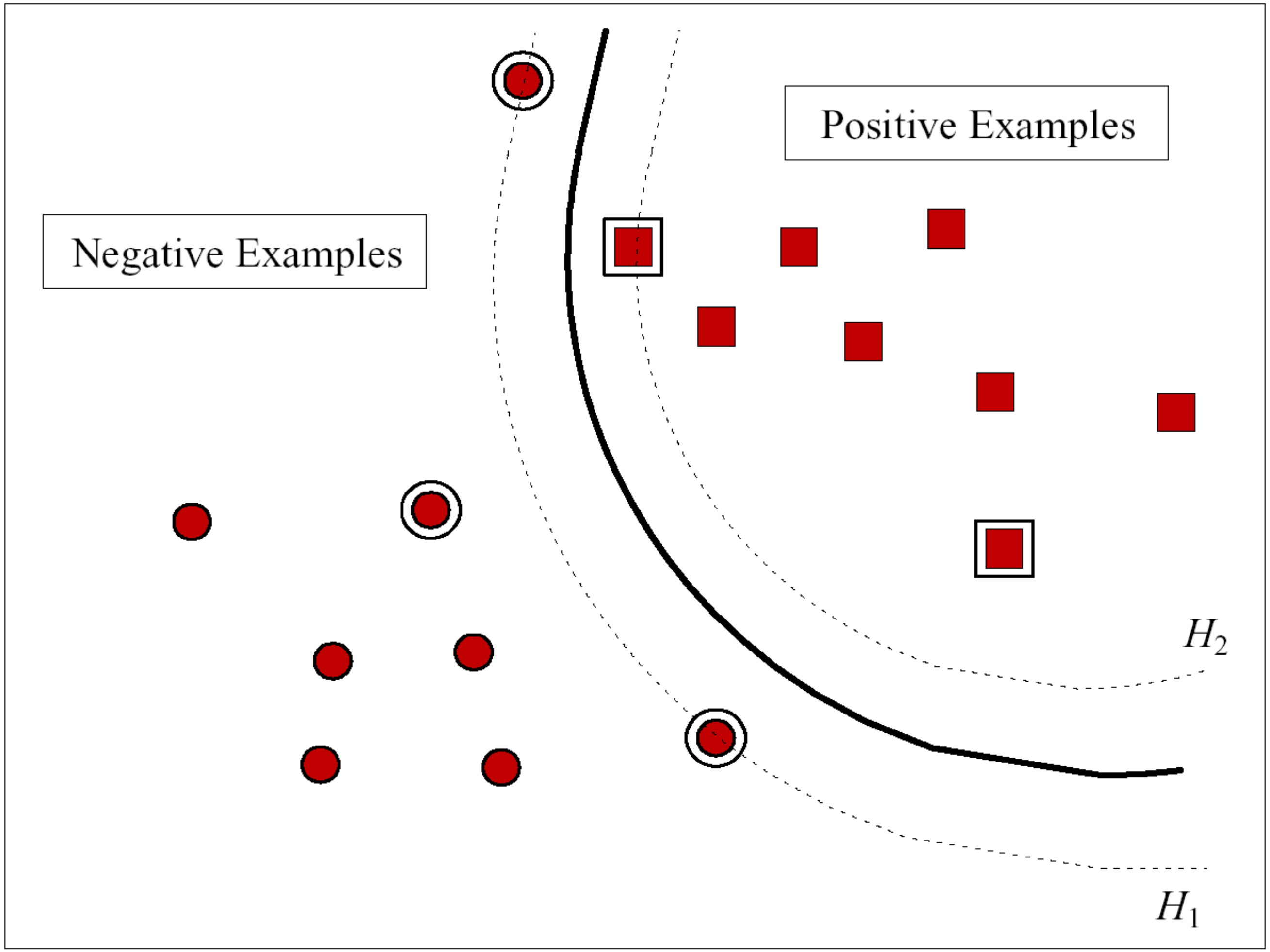
$$f(\mathbf{x}) = x_1 + x_2 - 2\, x_1 x_2 - 1/3\,, \quad x_3 = x_1 x_2\,, \qquad f(\mathbf{x}) = x_1 + x_2 - 2x_3 - 1/3$$





Problema XOR

Non-linear SVM

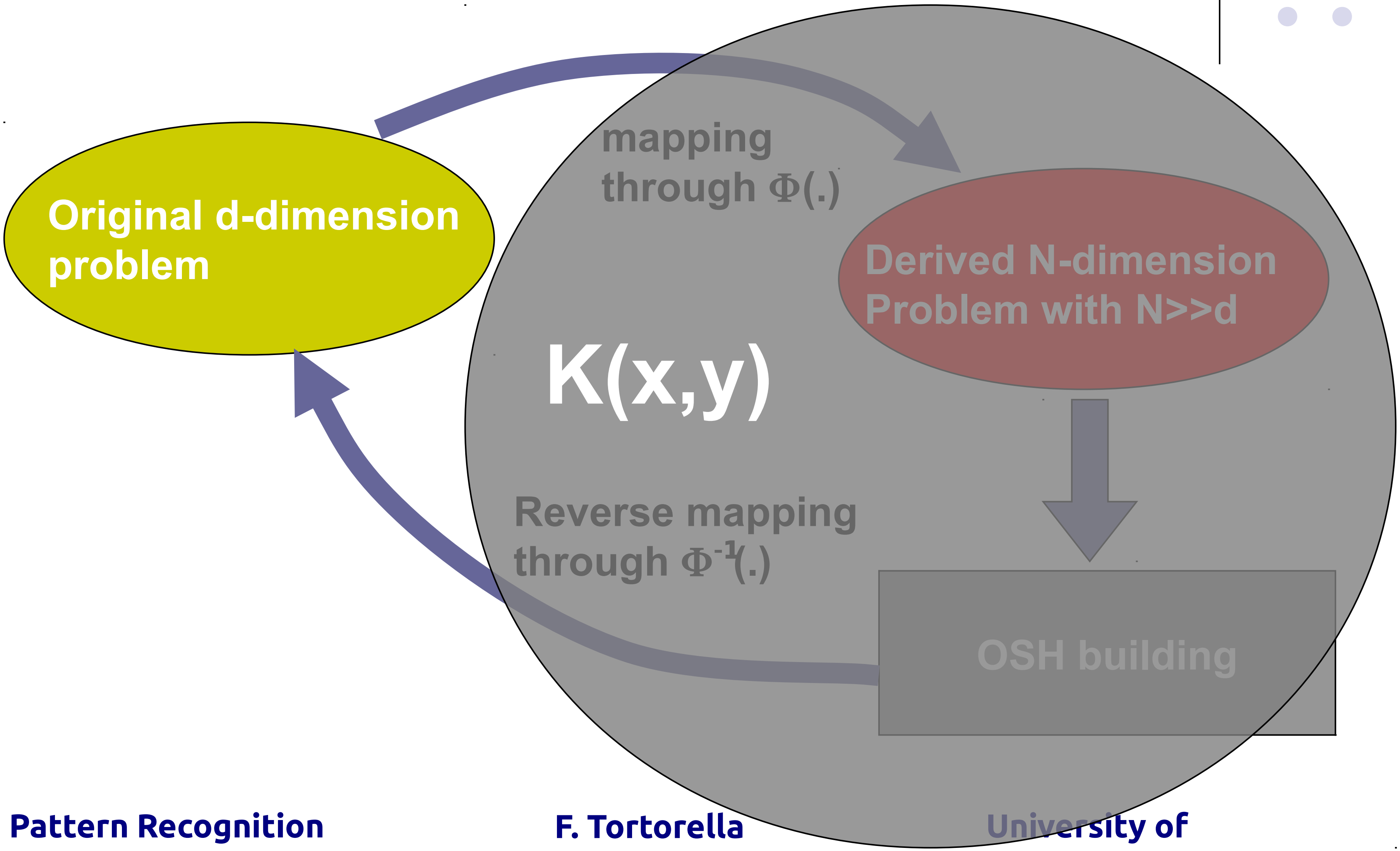


The kernel trick

- Actually it is not necessary to explicitly use the mapping function $\Phi(.)$.
- What is needed for the training stage and the classification stage is the functional form of the dot product $\Phi(x)\cdot\Phi(y)$.
- The Mercer's theorem guarantees that a kernel function $K(x,y)$ exists such that $K(x,y)=\Phi(x)\cdot\Phi(y)$.
- As a consequence, the discriminant function becomes:

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

The kernel trick



Some kernels

- Polynomial

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

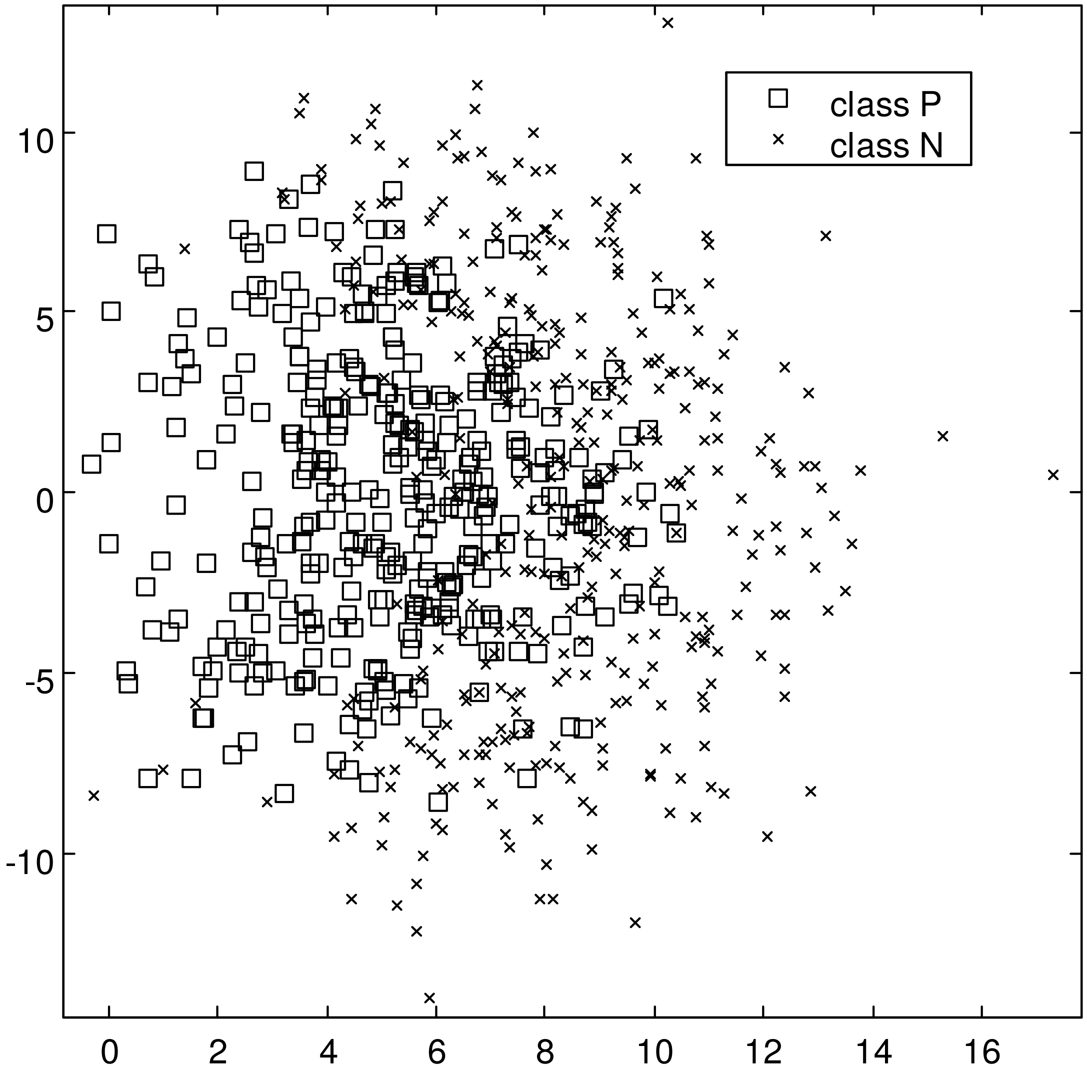
- Gaussian (RBF)

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

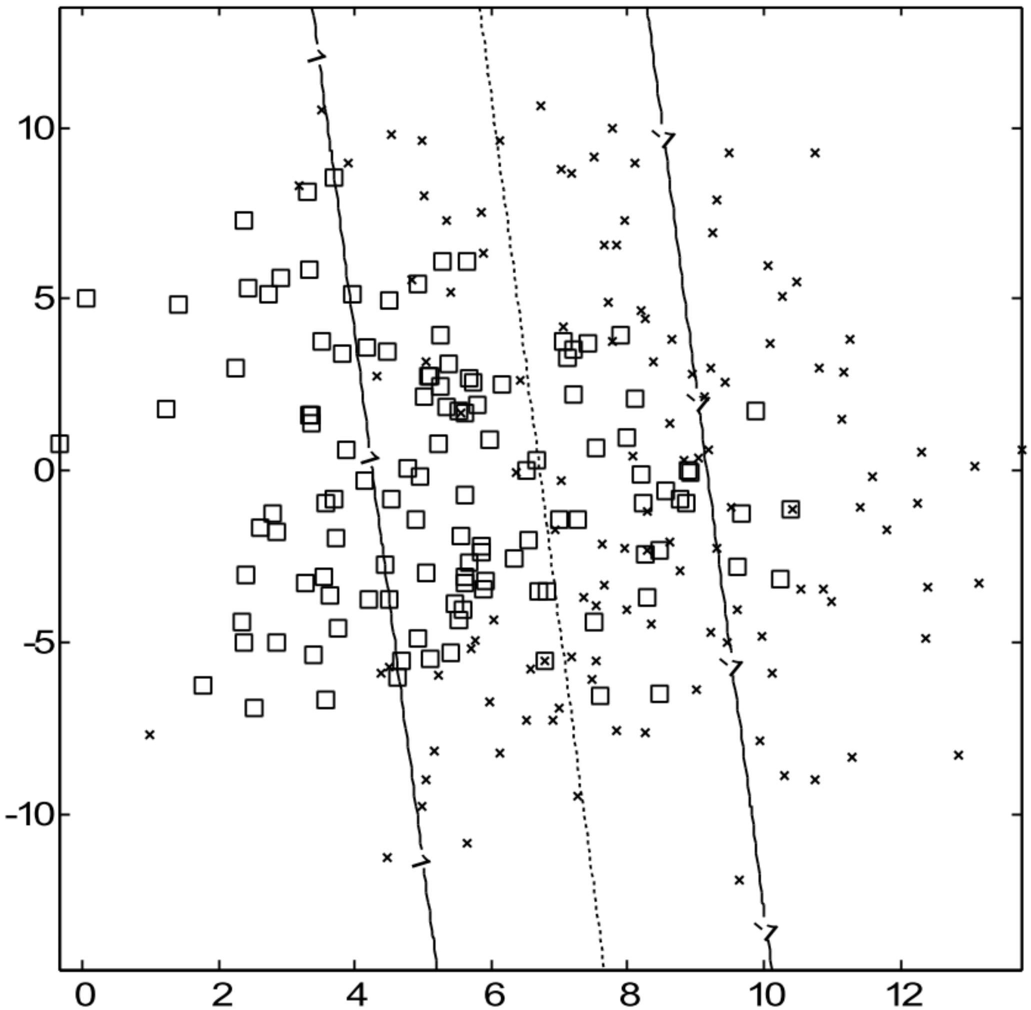
- MLP

$$K(\mathbf{x}, \mathbf{y}) = tanh\left(\kappa \mathbf{x} \cdot \mathbf{y} - \delta\right)$$

Example



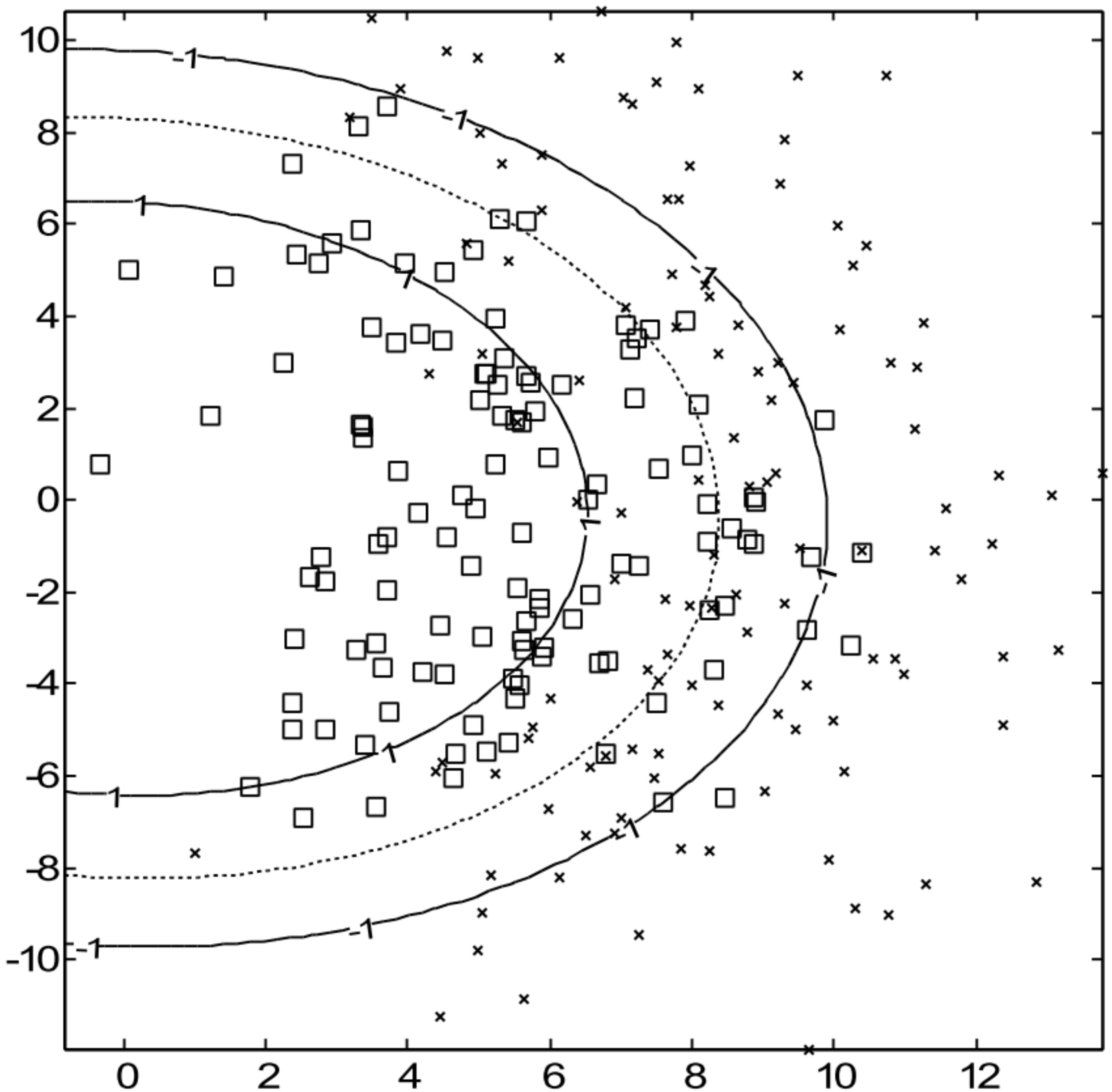
Linear SVM



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

163 SV on 240 training samples

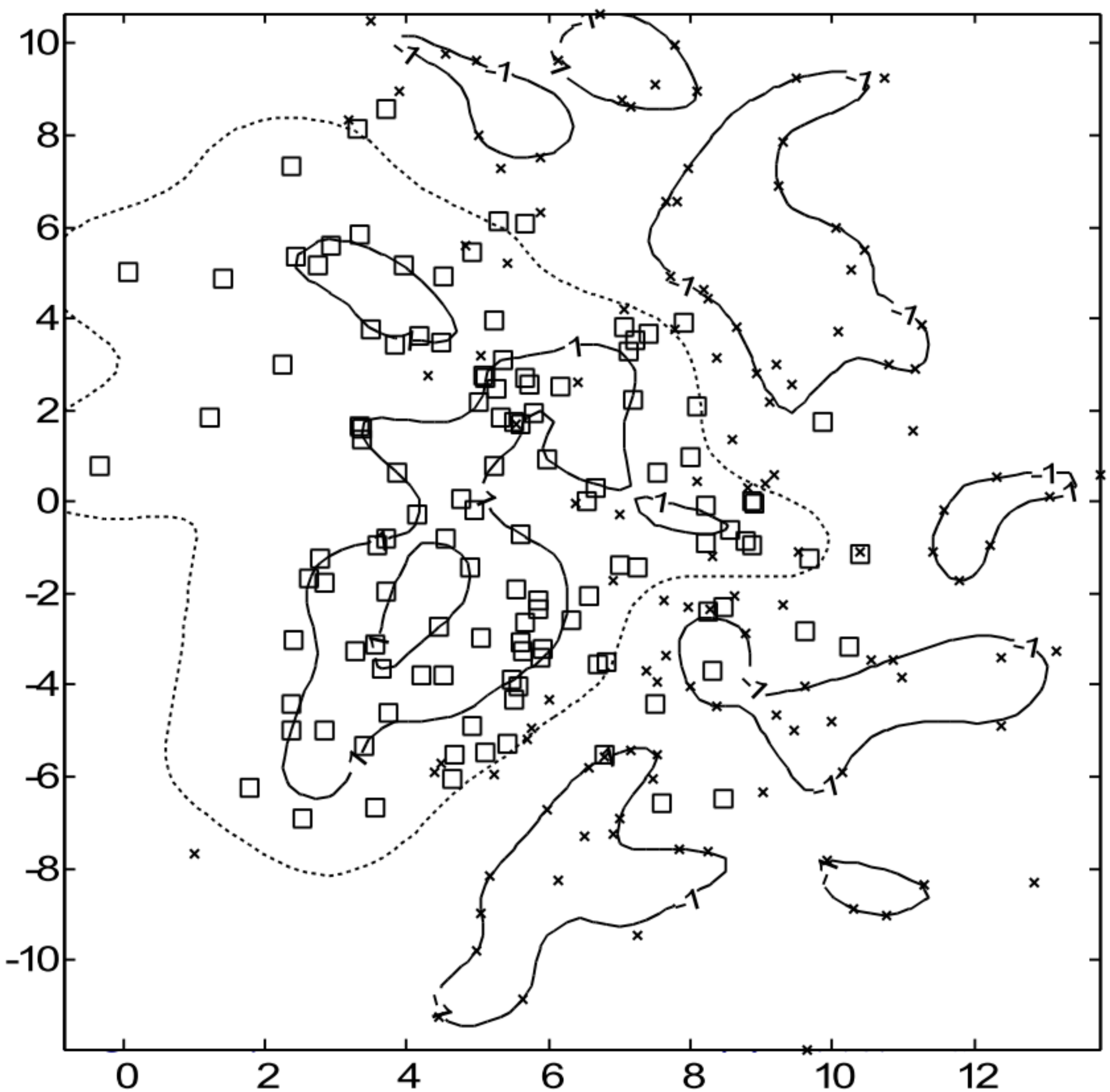
Polynomial SVM



$K(x,y)=(x \cdot y+1)^2$

121 SV on 240 training samples

RBF SVM



$K(x,y)=$
$\exp(-0.5\cdot||x-y||^2)$

190 SV on 240 training samples

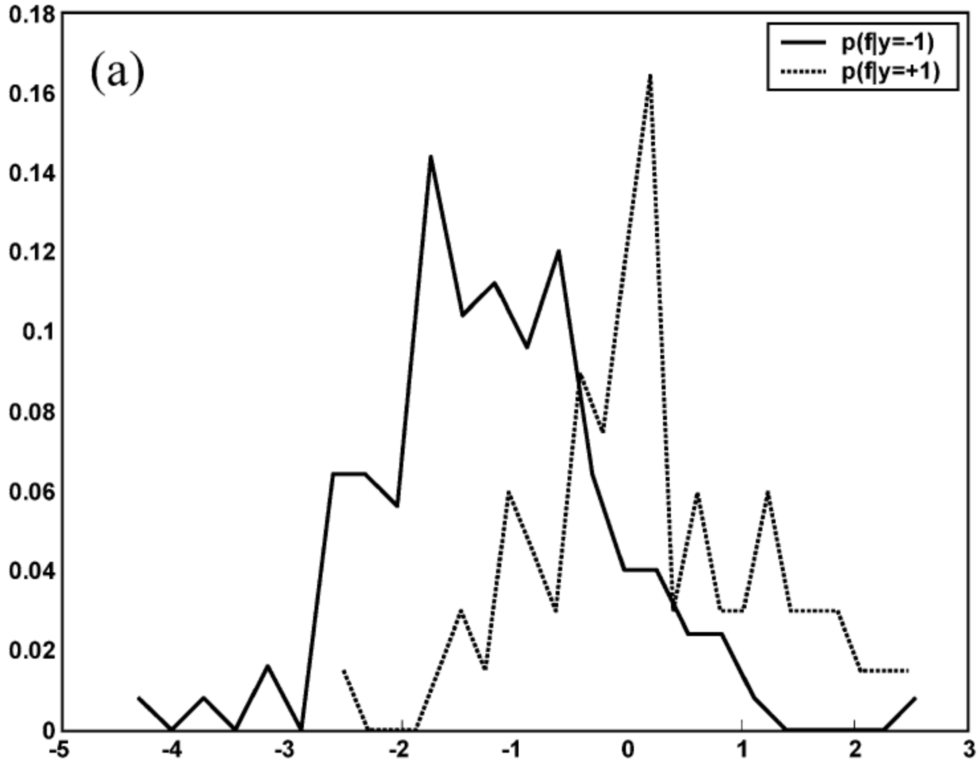
SVM advantages

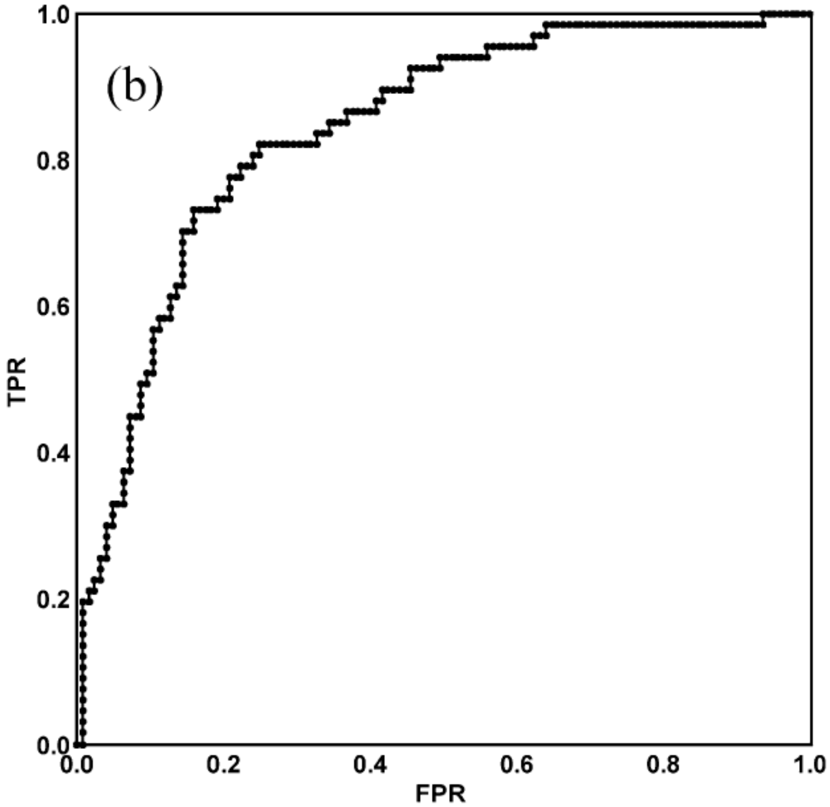
- No problems with local minima
- Optimal solution can be found in polynomial time
- The final results do not depend on random initial weights
- The SVM solution is sparse; it only involves the support vectors
- Excellent generalization capabilities

SVM issues

- Kernel to be selected (any principled way?)
- Model parameters to be selected (C, kernel parameters)
- Optimal data representation?
- SVM as a classifier
  - ROC curve?
  - Confidence measure? Postprobabilities?

ROC curve for SVM





Postprobabilities

- It is possible to transform the outputs of a SVM into a probability distribution over classes.
- ***Platt scaling***

$$P(y=+1|\mathbf{x}) = \frac{1}{1+exp(Af(\mathbf{x})+B)}$$