Statistical Learning Teory and Support Vector Machines

Marcello Pelillo

University of Venice, Italy

Artificial Intelligence

a.y. 2017/18

Ca' Foscari University of Venice

eclt European Centre for Living Technology

The formal setup

SLT deals mainly with **supervised learning** problems.

Given:

- ✓ an input (feature) space: $\mathcal{X}$
- ✓ an output (label) space: $\mathcal{Y}$ (typically $\mathcal{Y} = \{ -1, +1 \}$)

the question of learning amounts to estimating a functional relationship between the input and the output spaces:

$$f : \mathcal{X} \to \mathcal{Y}$$

Such a mapping f is called a **classifier**.

In order to do this, we have access to some (labeled) training data:

$$(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathcal{X} \times \mathcal{Y}$$

A **classification algorithm** is a procedure that takes the training data as input and outputs a classifier f.

Assumptions

In SLT one makes the following assumptions:

- ✓ there exists a joint probability distribution P on $\mathcal{X} \times \mathcal{Y}$
- ✓ the training examples (X_i, Y_i) are sampled independently from P (iid sampling).

In particular:

1. No assumptions on P
2. The distribution P is unknown at the time of learning
3. Non-deterministic labels due to label noise or overlapping classes
4. The distribution P is fixed

Losses and risks

We need to have some measure of "how good" a function f is when used as a classifier. A *loss function* measures the "cost" of classifying instance $X \in \mathcal{X}$ as $Y \in \mathcal{Y}$.

The simplest loss function in classification problems is the **0-1 loss** (or misclassication error):

$$\ell(X, Y, f(X)) = \begin{cases} 1 & \text{if } f(X) \neq Y \\ 0 & \text{otherwise.} \end{cases}$$

The *risk* of a function is the average loss over data points generated according to the underlying distribution P:

$$R(f) := E(\ell(X, Y, f(X)))$$

The *best classifier* is the one with the smallest risk $R(f)$.

Bayes classifiers

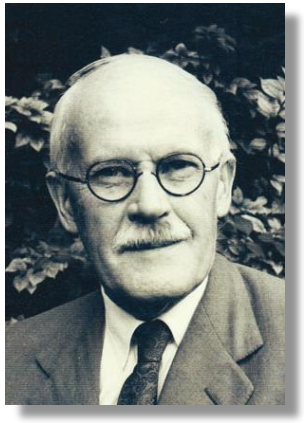
Among all possible classifiers, the "best" one is the *Bayes classifier*:

$$f_{Bayes}(x) := \begin{cases} 1 & \text{if } P(Y = 1 \mid X = x) \geq 0.5 \\ -1 & \text{otherwise.} \end{cases}$$

In practice, it is impossible to directly compute the Bayes classifier as the underlying probability distribution *P* is unknown to the learner.

The idea of estimating *P* from data doesn't usually work ...

Bayes' theorem



«[Bayes' theorem] is to the theory of probability what Pythagoras' theorem is to geometry.»

Harold Jeffreys
Scientific Inference (1931)

$$P(h \mid e) = \frac{P(e \mid h)P(h)}{P(e)} = \frac{P(e \mid h)P(h)}{P(e \mid h)P(h) + P(e \mid \varnothing h)P(\varnothing h)}$$

- ✓ $P(h)$: prior probability of hypothesis h
- ✓ $P(h \mid e)$: posterior probability of hypothesis h (in the light of evidence e)
- ✓ $P(e \mid h)$: "likelihood" of evidence e on hypothesis h

The classification problem

Given:

- ✓ a set training points $(X_1,Y_1), \ldots, (X_n,Y_n) \in \mathcal{X} \times \mathcal{Y}$ drawn iid from an *unknown* distribution P
- ✓ a loss functions

Determine a function $f : \mathcal{X} \to \mathcal{Y}$ which has risk $R(f)$ as close as possible to the risk of the Bayes classifier.

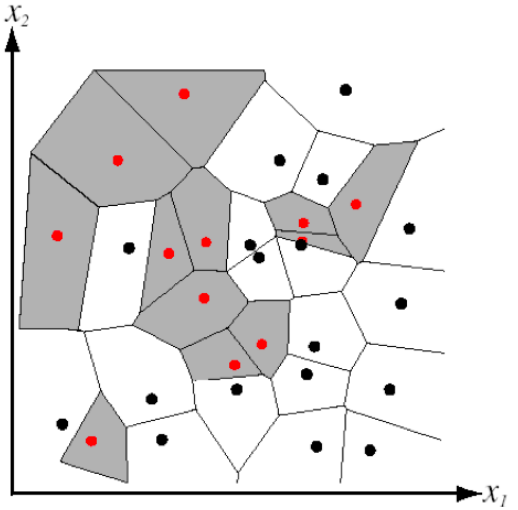
Caveat. Not only is it impossible to compute the Bayes error, but also the risk of a function f cannot be computed without knowing P.

A desperate situation?

An example: The nearest neighbor (NN) rule

«Early in 1966 when I first began teaching at Stanford, a student, Peter Hart, walked into my office with an interesting problem. He said that Charles Cole and he were using a pattern classification scheme which, for lack of a better word, they described as the **nearest neighbor procedure**.
This scheme assigned to an as yet unclassified observation the classification of the nearest neighbor. Were there any good theoretical properties of this procedure?»

Thomas Cover (1982)



How good is the NN rule?

Cover and Thomas showed that:

$$R(f_{Bayes}) \leq R_{\infty} \leq 2R(f_{Bayes})$$

where R_{∞} denotes the expected error rate of NN when the sample size tends to infinity.

We cannot say anything stronger as there are probability distributions for which the performance of the NN rule achieves either the upper or lower bound.

Variations:

- ***k*-NN rule:** use the k nearest neighbors and take a majority vote
- ***k_n*-NN rule:** the same as above, for k_n growing with n

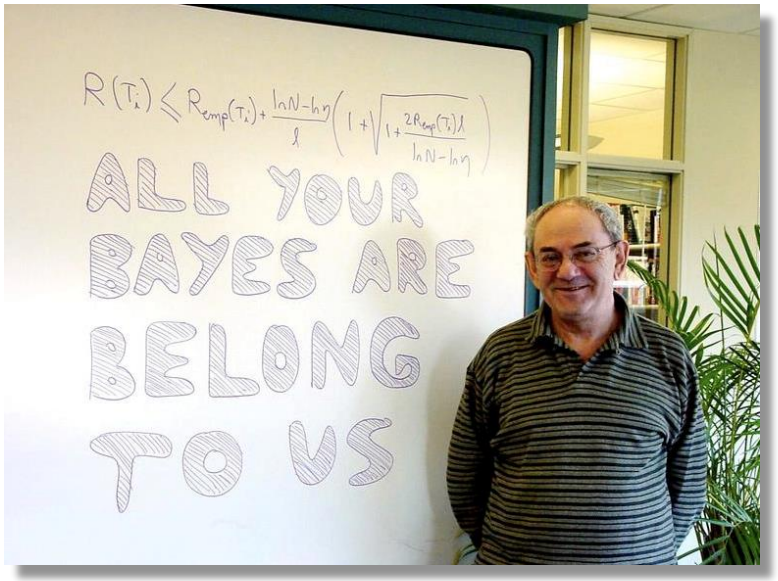
Theorem (Stone, 1977) If $n \to \infty$ and $k \to \infty$, such that $k/n \to 0$, then for all probability distributions $R(k_n\text{-NN}) \to R(f_{Bayes})$ (that is, the k_n-NN rule is "universally Bayes consistent").

Empirical Risk Minimization

«At the end of the 1960's, the theory of Empirical Risk Minimization (ERM) for the pattern recognition problem was constructed.

This theory included both (a) the general *qualitative theory* of generalization that described the necessary and sufficient conditions for consistency of the ERM induction principle [...];

and (b) the general *quantitative theory* that described the bounds on the probability of the (future) test error.»



Vladimir Vapnik
Statistical Learning Theory (1998)

The ERM principle

Instead of looking for a function which minimizes the true risk $R(f)$, we try to find one which minimizes the *empirical risk*:

$$R_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(X_i, Y_i, f(X_i))$$

Given training data $(X_1,Y_1), \ldots, (X_n,Y_n) \in \mathcal{X} \times \mathcal{Y}$, a function space $\mathcal{F}$, and a loss function, we define the classifier f_n as:

$$f_n := \operatorname*{argmin}_{f \in \mathcal{F}} \mathrm{R}_{\text{emp}}(f)$$

This approach is called the *empirical risk minimization* (ERM) induction principle, the motivation of which comes from the law of large numbers.

Note. Same as least-squares/ML methods (but… binary *vs.* real functions!).

A key question

What has to be true of the function class $\mathcal{F}$ so that, no matter what the unknown background probability distribution, ERM eventually does as well as possible with respect to the rules in $\mathcal{F}$?

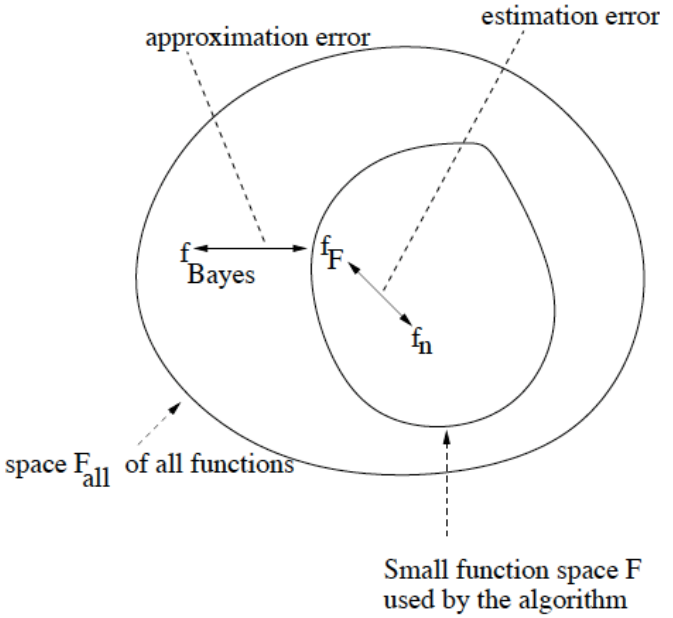
A fundamental result of SLT is that the set of rules in $\mathcal{F}$ cannot be too rich, where the richness of $\mathcal{F}$ is measured by its VC dimension.

Estimation *vs.* approximation

Ideally we want to make $R(f_n) - R(f_{Bayes})$ as small as possible, as $n \to \infty$.

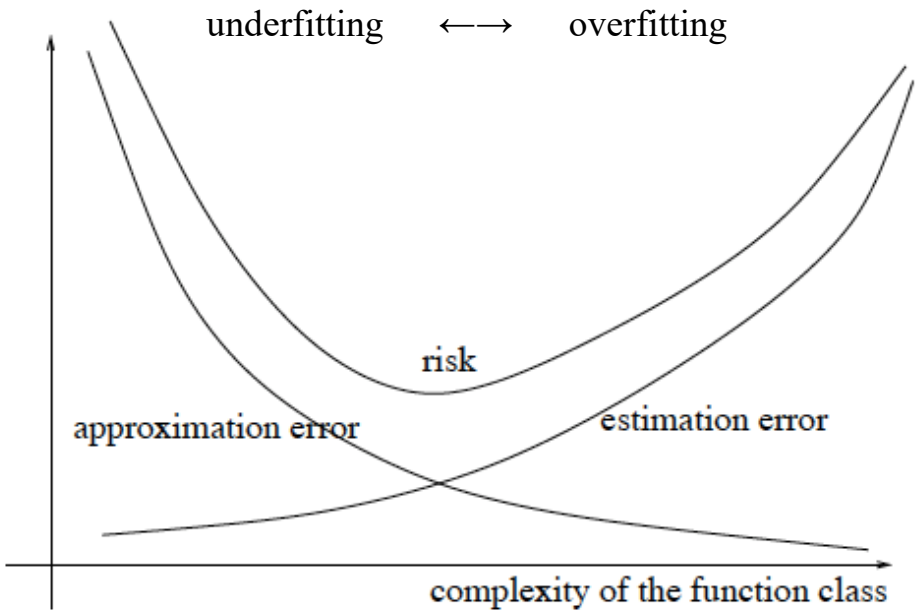
Denoting by $f_{\mathcal{F}}$ the best classifier in $\mathcal{F}$, the difference can be decomposed as:

$$R(f_n) - R(f_{Bayes}) = \underbrace{\Big(R(f_n) - R(f_{\mathcal{F}})\Big)}_{\text{estimation error}} + \underbrace{\Big(R(f_{\mathcal{F}}) - R(f_{Bayes})\Big)}_{\text{approximation error}}$$



[von Luxburg and Schölkopf, 2008]

Underfitting *vs.* overfitting

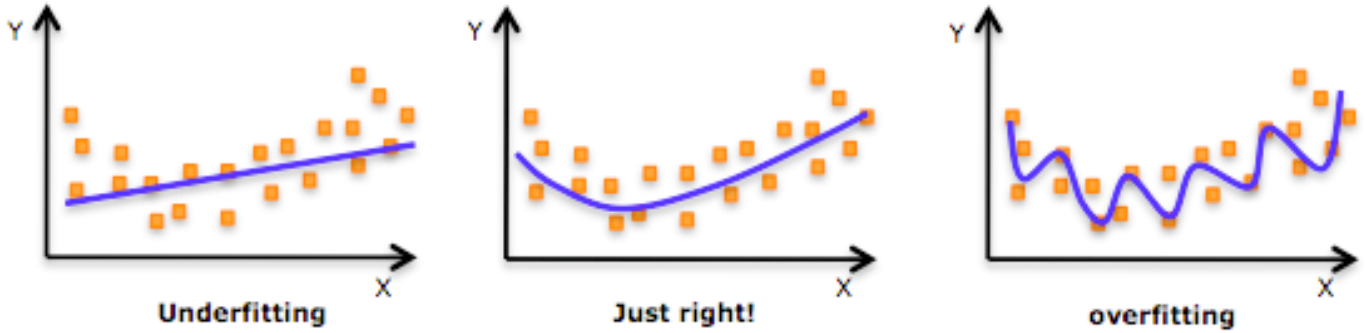


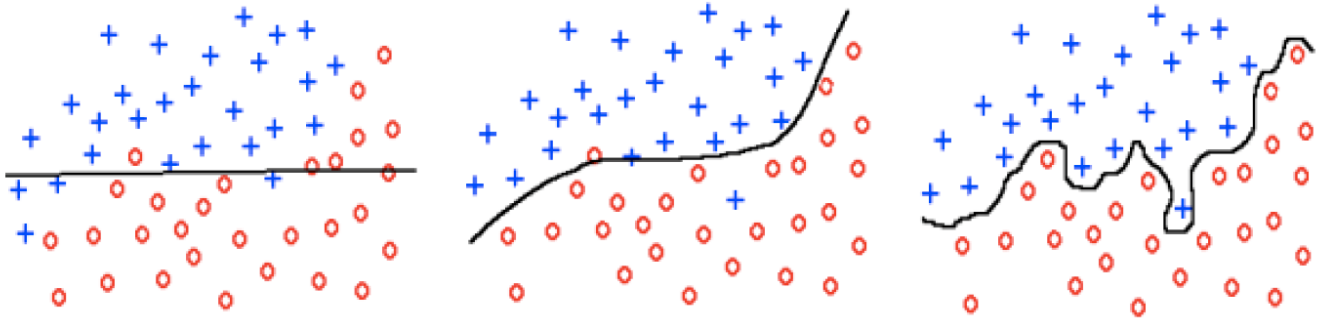
- ✓ small complexity of $\mathcal{F}$ ⇒ small estimation error, large approximation error (*underfitting*)
- ✓ large complexity of $\mathcal{F}$ ⇒ large estimation error, small approximation error (*overfitting*)

The best overall risk is achieved for "moderate" complexity

[von Luxburg and Schölkopf, 2008]

Model selection





underfit fit overfit

Shattering

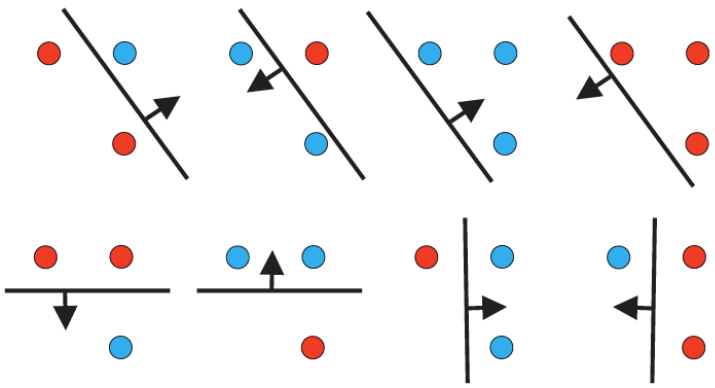
A set of n instances $X_1,\ldots, X_n$ from the input space $\mathcal{X}$ is said to be *shattered* by a function class $\mathcal{F}$ if all the 2^n labelings of them can be generated using functions from $\mathcal{F}$.

Example.

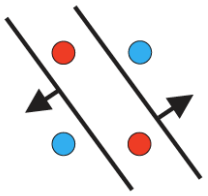
$\mathcal{F}$ = linear decision functions (straight lines) in the plane

(a) Any set of 3 non-collinear points shatters $\mathcal{F}$

(b) No set of 4 points can shatter $\mathcal{F}$



(a)



(b)

The VC dimension

The *VC dimension* of a function class $\mathcal{F}$, denoted $\mathrm{VC}(\mathcal{F})$, is the largest integer h such that *there exists* a sample of size h which is shattered by $\mathcal{F}$.

If arbitrarily large samples can be shattered, then $\mathrm{VC}(\mathcal{F}) = \infty$.

Examples.

- ✓ $\mathcal{F}$ = linear decision functions in $\mathbf{R}^2$ $\Rightarrow$ $\mathrm{VC}(\mathcal{F}) = 3$
- ✓ $\mathcal{F}$ = linear decision functions (hyperplanes) in $\mathbf{R}^n$ $\Rightarrow$ $\mathrm{VC}(\mathcal{F}) = n + 1$
- ✓ $\mathcal{F}$ = multi-layer perceptrons with W weights $\Rightarrow$ $\mathrm{VC}(\mathcal{F}) = O(W \log W)$
- ✓ $\mathcal{F}$ = nearest neighbor classifiers $\Rightarrow$ $\mathrm{VC}(\mathcal{F}) = \infty$

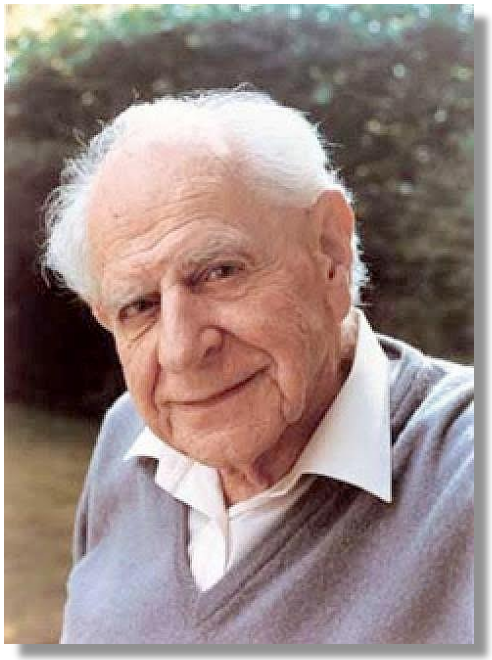
VC dimension *vs.* number of parameters

«In algebraic representation, the dimension of the set of curves depends upon the number of *parameters* whose values we may freely choose.

We can therefore say that the number of freely determinable parameters of a set of curves by which a theory is represented is characteristic for the degree of falsifiability (or testability) of that theory.»

Karl Popper

The Logic of Scientific Discovery (1959)



Note. The VC dimension is in general not related to the number of free parameters of a model (e.g., $f_\alpha(x) = \mathrm{sgn}(\sin(\alpha x))$: 1 parameter, VCdim = ∞).

Fundamental results

For all $f \in \mathcal{F}$, with probability at least $1 - \delta$, we have:

$$R(f) \leq R_{\text{emp}}(f) + \sqrt{\frac{h(\log(2n/h)+1) - \log(\delta/4)}{n}}$$

where $h = \text{VC}(\mathcal{F})$, and n is the sample size.

With probability approaching 1, no matter what the unknown probability distribution, given more and more data, the expected error for the functions that ERM endorses at each stage eventually approaches the minimum value of expected error of the functions in $\mathcal{F}$ *if and only if* $\mathcal{F}$ has finite VC dimension.

Structural risk minimization

ERM takes only care of the *estimation* error (variance) but it is not concerned with the *approximation* error (bias).

The optimal model is found by striking a balance between the empirical risk and the capacity of the function class F (e.g., the VC dimension).

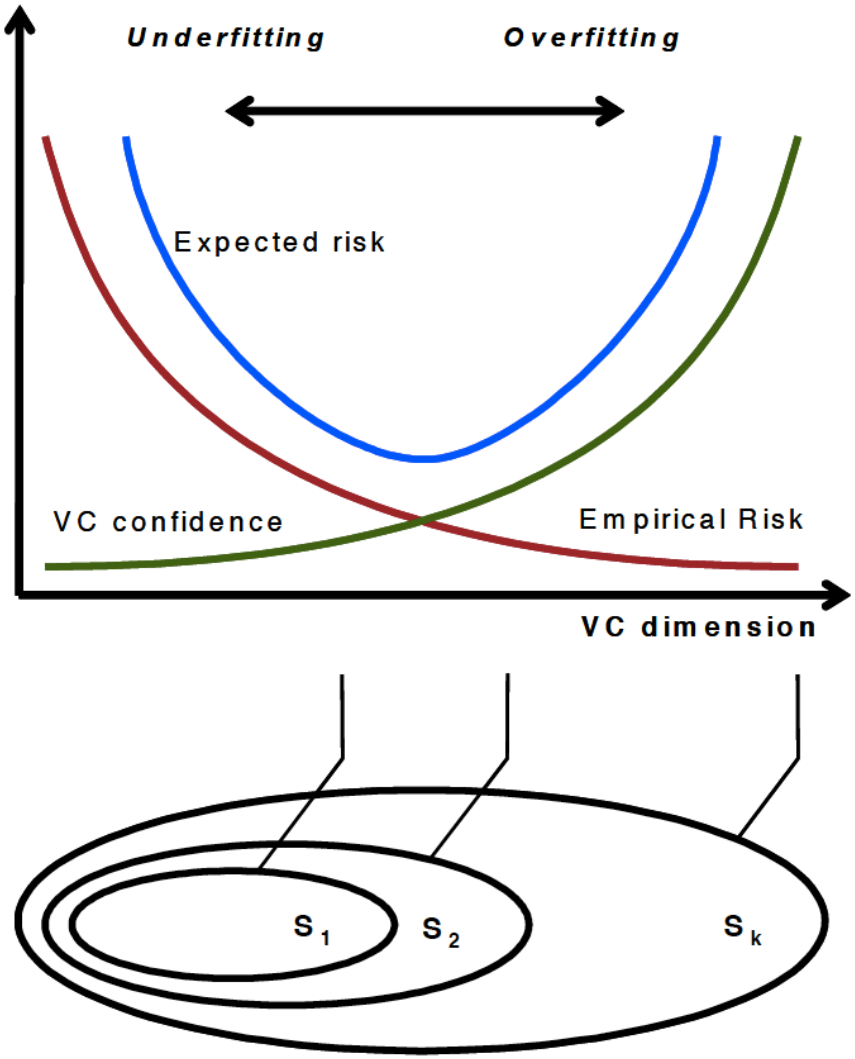
Basic idea of *Structural Risk Minimization* (SRM):

1. Construct a nested structure for family of function classes $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots$ with non-decreasing VC dimensions ($VC(\mathcal{F}_1) \leq VC(\mathcal{F}_2) \leq \ldots$)

2. For each class $\mathcal{F}_i$, find the solution f_i that minimizes the empirical risk

3. Choose the function class $\mathcal{F}_i$, and the corresponding solution f_i that minimizes the risk bound (= empirical risk + VC confidence)

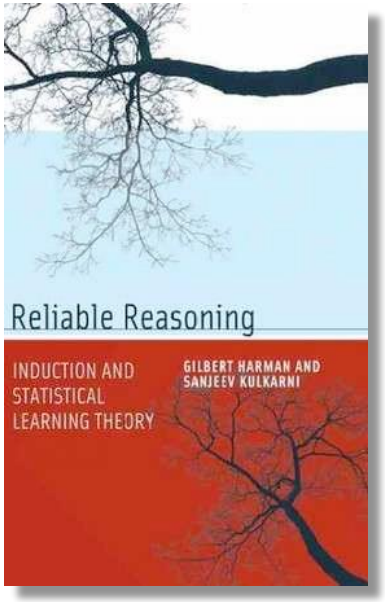
Structural risk minimization



Further readings

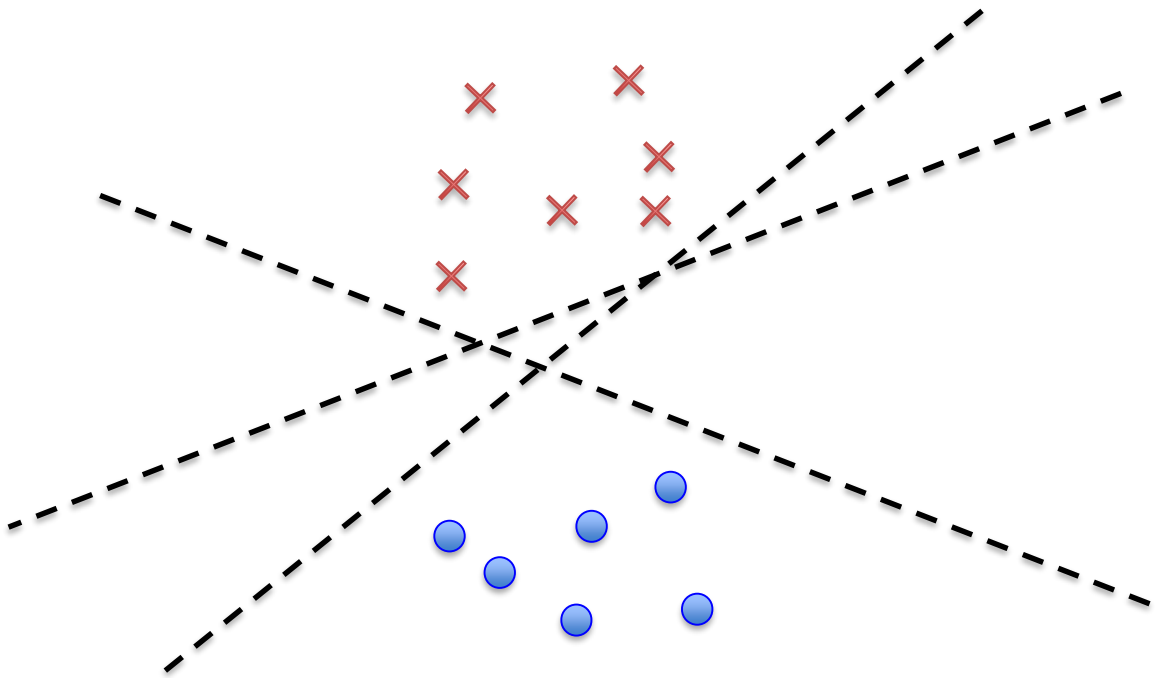
U. von Luxburg and B. Schölkopf. *Statistical learning theory: Models, concepts and results* (2008).

S. Kulkarni and G. Harman. *Statistical learning theory: A tutorial* (2011).



Support Vector Machines

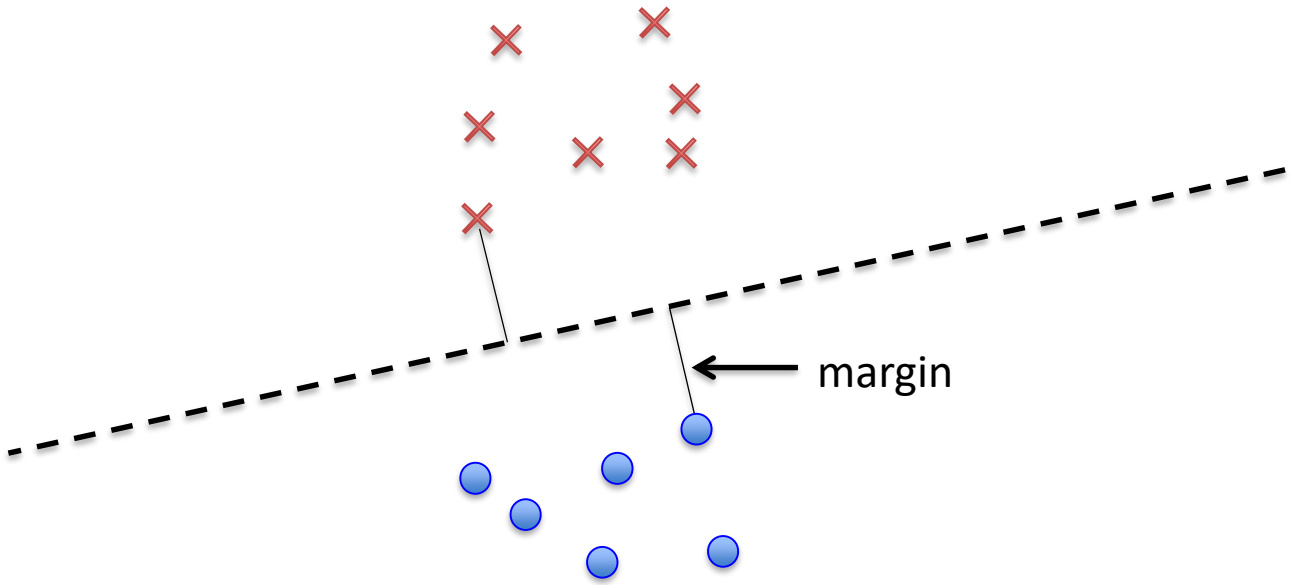
Several possible decision boundaries



All get 100% accuracy on this training set!

Several possible decision boundaries

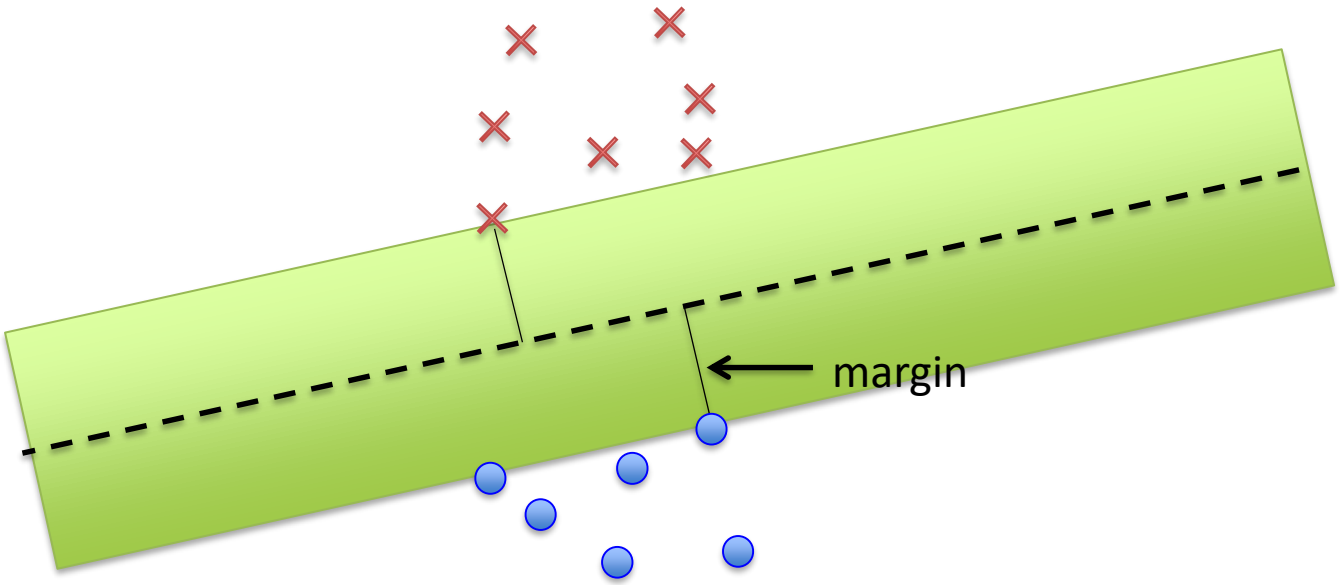
The SVM finds this one – the boundary furthest from the two clusters



Distance to the closest training point is called **the margin**
(equal on both sides of the boundary)

Several possible decision boundaries

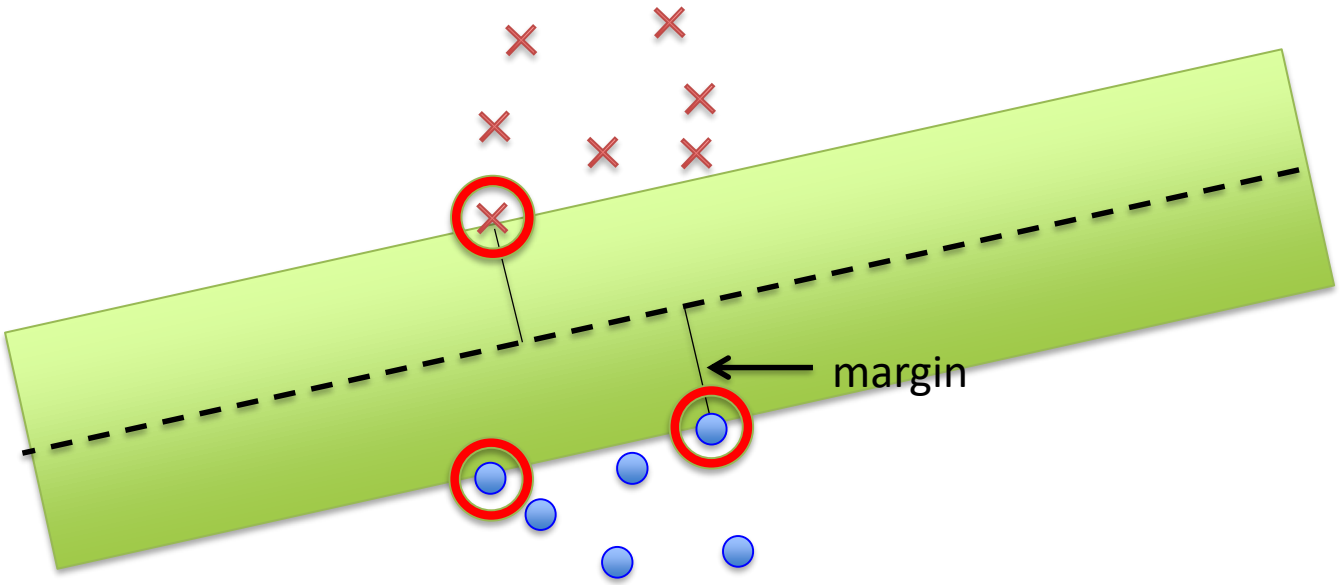
The SVM finds this one – the boundary furthest from the two clusters



Distance to the closest training point is called **the margin**
(equal on both sides of the boundary)

Several possible decision boundaries

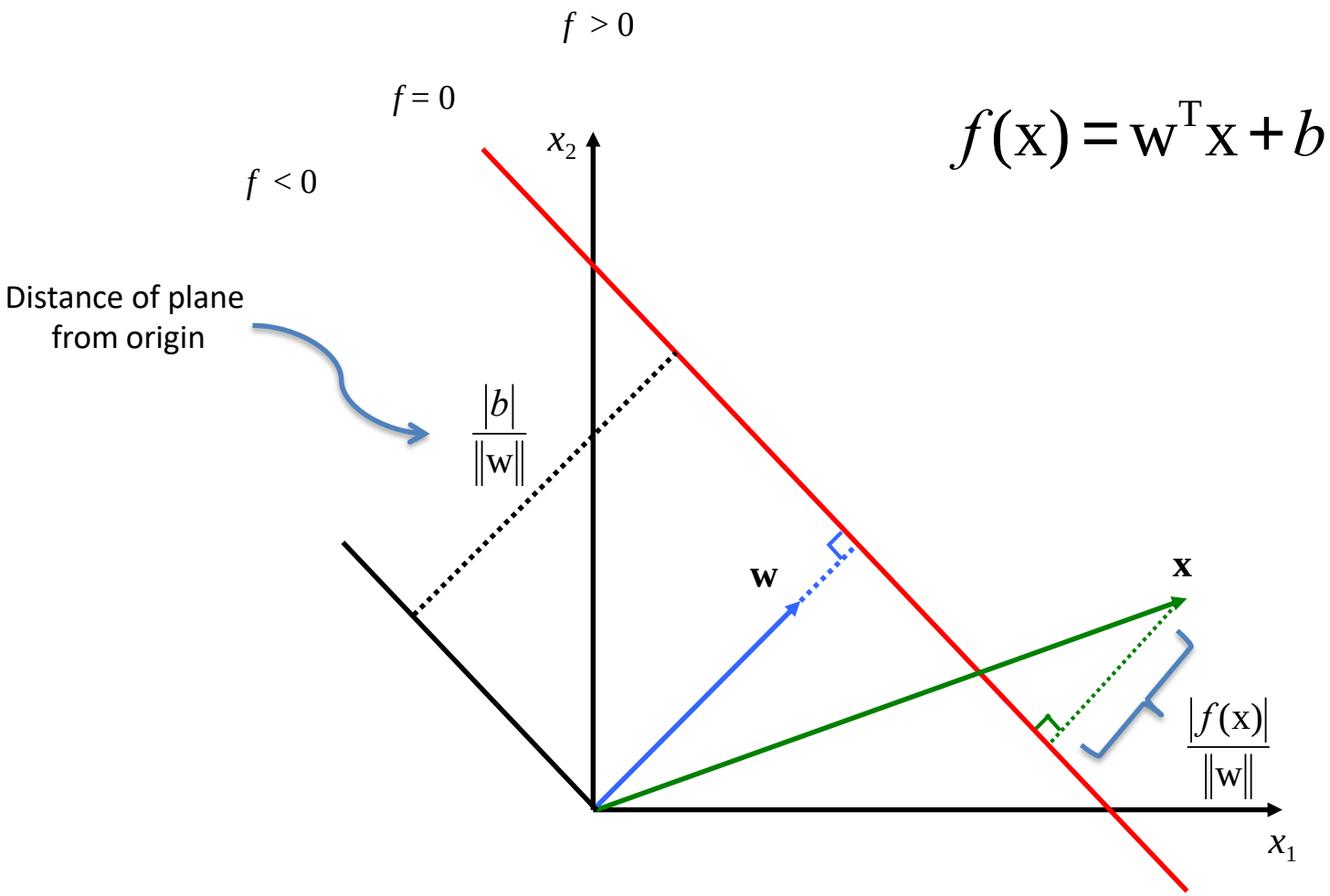
The SVM finds this one – the boundary furthest from the two clusters



The circled points are called
SUPPORT VECTORS

All other points can move freely.
Solution only dependent on SVs.

Basic geometric facts



Proof

$$\mathbf{x} = \mathbf{x}_\wedge + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Which yields:

$$\mathbf{x}_\wedge = \mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

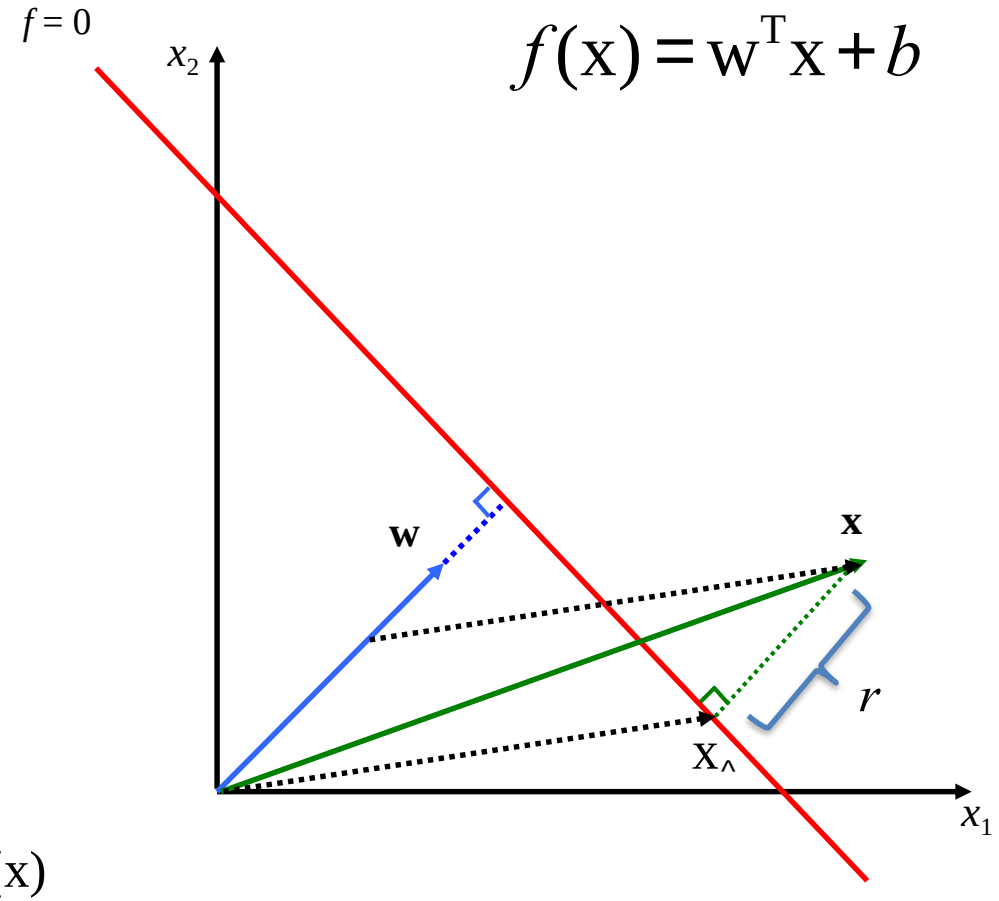
Since $\mathbf{x}_\wedge$ belongs to the plane:

$$\mathbf{w}^T\mathbf{x}_\wedge + b = 0$$

$$\mathbf{w}^T\left(\mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b = 0$$

$$\mathbf{w}^T\mathbf{x} - r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} + b = 0$$

$$\mathbf{w}^T\mathbf{x} - r\|\mathbf{w}\| + b = 0 \quad \Rightarrow \quad r = \frac{f(\mathbf{x})}{\|\mathbf{w}\|}$$



Normalizing the weights

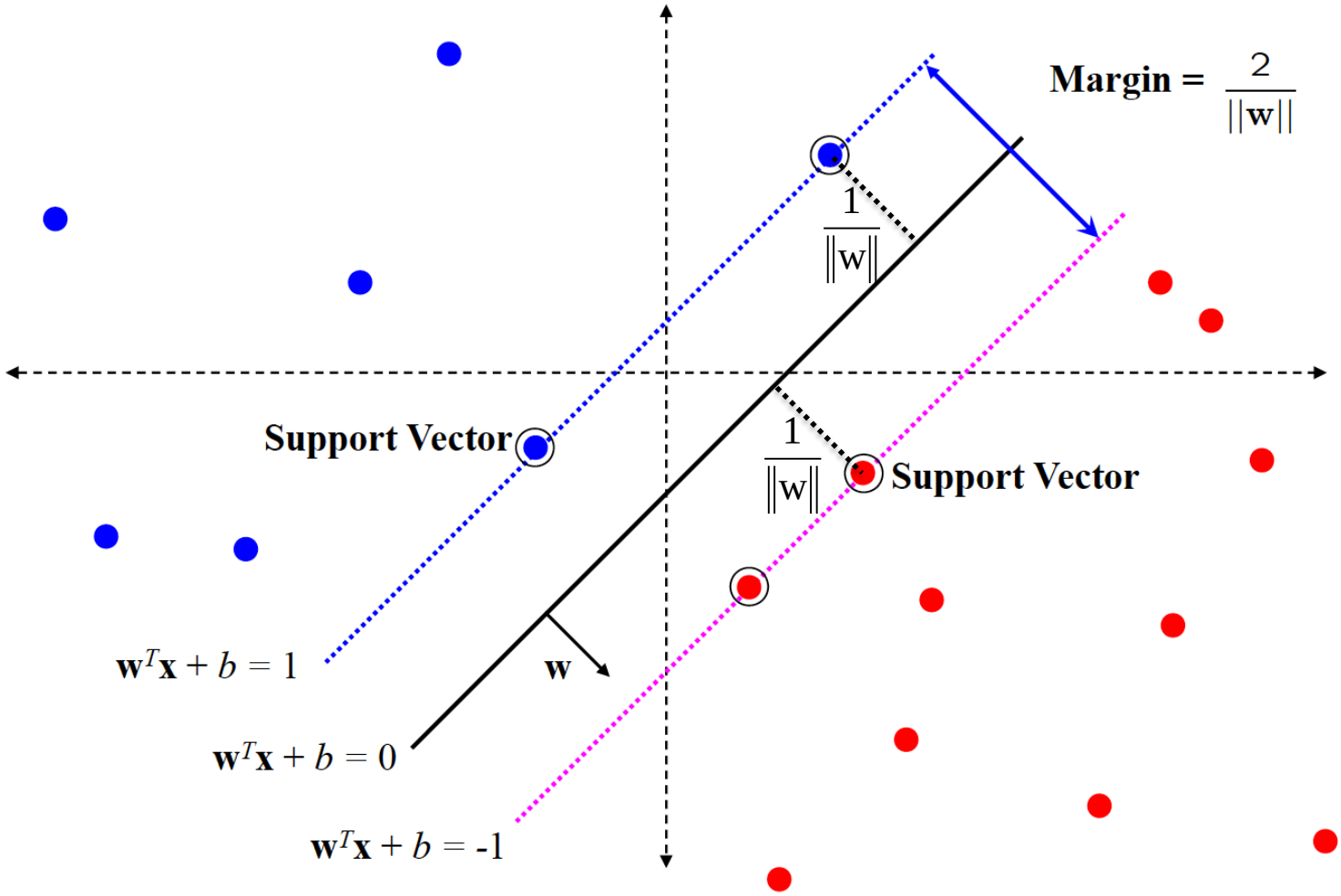
Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$ and $c(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = 0$ define the same plane.

Hence we have the freedom to chose the normalization of $\mathbf{w}$ and b.

Choose normalization such that (canonical form):

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = +1$ for the *positive* support vectors
- $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = -1$ for the *negative* support vectors

Support Vector Machines



Learning SVM's

Learning the SVM can be formulated as an optimization problem:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{l} \geq 1 \quad \text{if } y_i = +1 \\ \leq -1 \quad \text{if } y_i = -1 \end{array} \text{ for } i = 1 \ldots N$$

or, equivalently:

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \ldots N$$

This is a (convex) quadratic optimization problem subject to linear constraints and **there is a unique minimum!**

The problem of margin maximization

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to} \quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b) \geq 1, \quad i = 1, \ldots, N$$

To solve this constrained optimization problem, we introduce N Lagrange multipliers $\lambda_i \geq 0$ (one for each constraint) giving the **Lagrangian** function:

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left[y_i \left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \right) - 1 \right]$$

where $\boldsymbol{\Lambda} = (\lambda_1, \ldots, \lambda_N)$ is the vector of Lagrange multipliers.

The problem of margin maximization

Setting the derivatives of $L(\mathbf{w}, b, \boldsymbol{\Lambda})$ to zero we obtain:

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\Lambda})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\Lambda})}{\partial b} = \sum_{i=1}^{N} \lambda_i y_i = 0$$

The dual representation

Eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \boldsymbol{\Lambda})$ using these conditions, gives the **dual representation** of the maximum margin problem:

$$\text{maximize} \quad L_D(\lambda_1, \ldots, \lambda_N) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \text{ for all } i = 1 \ldots N$$

Note: the training vectors $\mathbf{x}_i$ appear only as dot products (useful later on!).

The dual representation

If $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_N)$ is the solution of the dual optimization problem, then:

- The weight vector of the maximum margin hyperplane is:

$$\mathbf{w} = \sum_i y_i \lambda_i \mathbf{x}_i$$

- The corresponding discriminant function is:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \\ &= \sum_i y_i \lambda_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b \end{aligned}$$

- The linear SVM classifier $g : \mathbf{R}^n \rightarrow \{ -1, +1 \}$ is:

$$g(\mathbf{x}) = \mathrm{sgn}(\sum_i y_i \lambda_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b)$$

Note: vector b is implicitly given by the constraints.

The role of support vectors

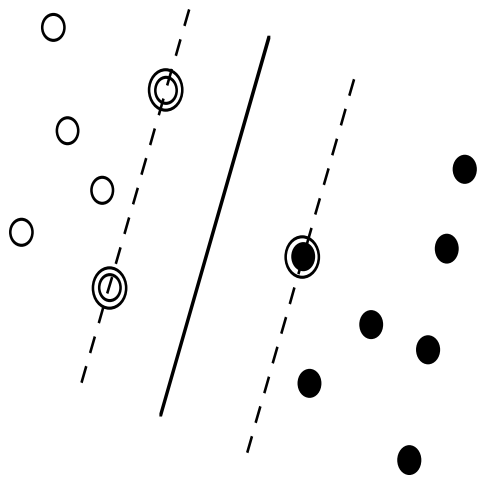
In the dual representation, the maximum margin hyperplane is given by:

$$\sum_{i=1}^{N} y_i \lambda_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b = 0$$

At first sight the dual form appears to have the disadvantage of *k*-NN classifiers — it requires the training data points $\mathbf{x}_i$.

However, many of the λ_i's are zero (sparse solution).

The coefficients $\lambda_i > 0$ are the **support vectors!**



Finding the *b* parameter

For support vectors we have

$$y_i \left(\sum_i y_i \lambda_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b \right) = 1$$

which yields

$$b = 1 / y_i - \sum_i y_i \lambda_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}$$

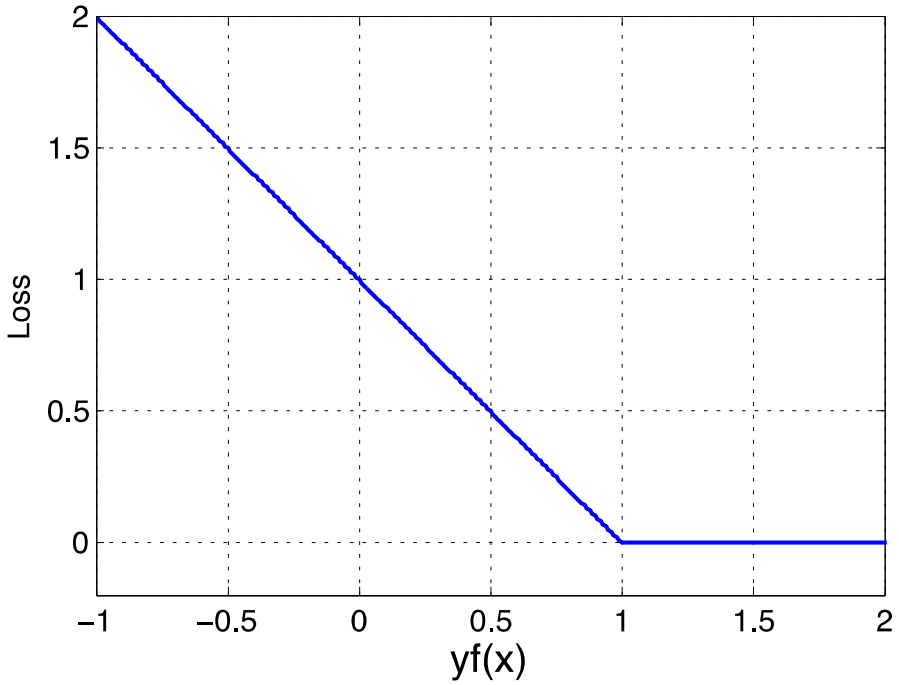
Considering *all* support vectors we obtain a more stable solution:

$$b = \frac{1}{|SV|} \sum_{i \in SV} \left(y_i - \sum_{j=1}^{N} y_j \lambda_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i \right)$$

where SV is the set of support vectors.

Hinge loss function

$$L_{hinge} = \max\left\{0, 1 - y_i f(\mathbf{x}_i)\right\}$$



SVM error function = hinge loss + 1/margin

$$E = \sum_{i=1}^{N} \max\left\{0, 1 - y_i f(\mathbf{x}_i)\right\} + \frac{1}{2}\sum_{j=1}^{d} w_j^2$$

hinge loss
(summed over data points)

proportional to the inverse
of the margin

Gradient descent not possible.

But, this is a quadratic function with linear constraints.
So we can use "**quadratic programming" (QP)** algorithms.

QP solvers are standard and efficient – like "sqrt(x)",
nobody programs it themselves.

SVM's and the VC dimension

Theorem (Vapnik)

Consider hyperplanes $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$ in canonical form, that is such that:

$$\min_{1 \leq i \leq N} \left| \mathbf{w}^{\mathrm{T}} \mathbf{x}_i + b \right| = 1$$

Then the set of decision functions $g(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0)$ that satisfy the constraint $\|\mathbf{w}\| < \gamma$ has a VC dimension h satisfying:

$$h \leq R^2 \gamma^2$$

where R is the smallest radius of the sphere around the origin containing all the training points.

Note: Dropping the condition $\|\mathbf{w}\| < \gamma$ leads to a VC dimension equal to $n+1$. Hence, the constraint allows us to work in high-dimension spaces.

SVM's and SLT

From the previous theorem and from

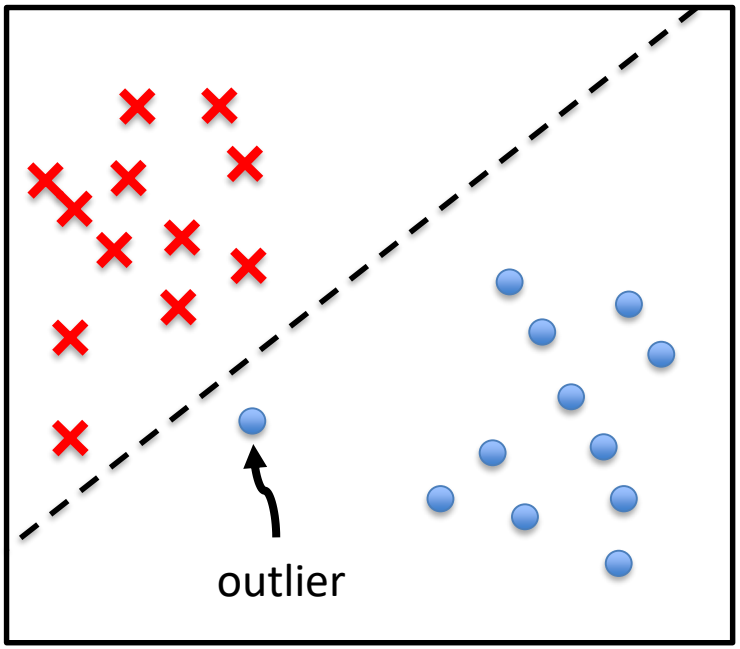
$$R(f) \pounds R_{\text{emp}}(f) + \sqrt{\frac{h(\log(2n/h)+1) - \log(d/4)}{n}}$$

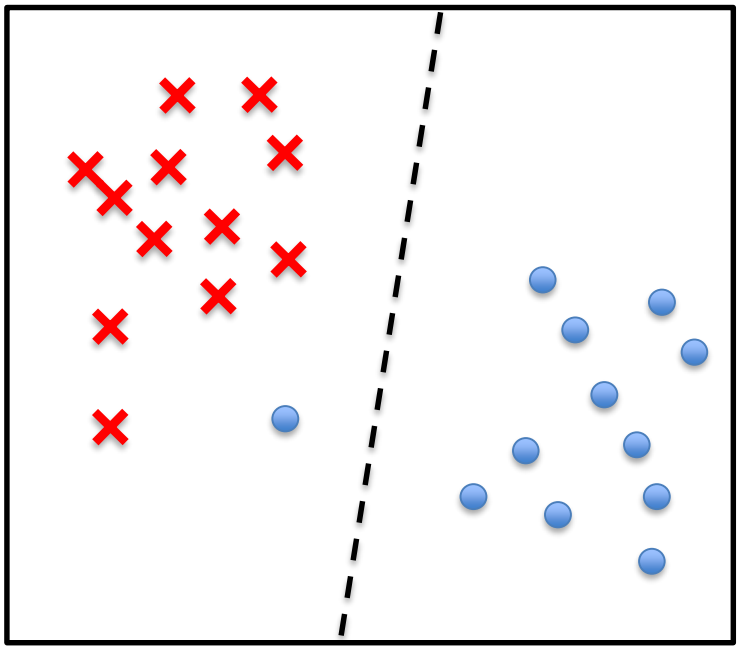
we have:

- By maximizing the margin, or equivalently by minimizing $\|\mathbf{w}\|$, we are in fact minimizing the VC dimension of the SVM.
- The minimization of the expected risk depends on both minimizing the empirical risk and the confidence interval
- The confidence interval depends mainly on the ratio h/n
- SVM algorithm minimizes both the empirical risk and the confidence interval
- SVM directly implements the structural risk mimimization principle

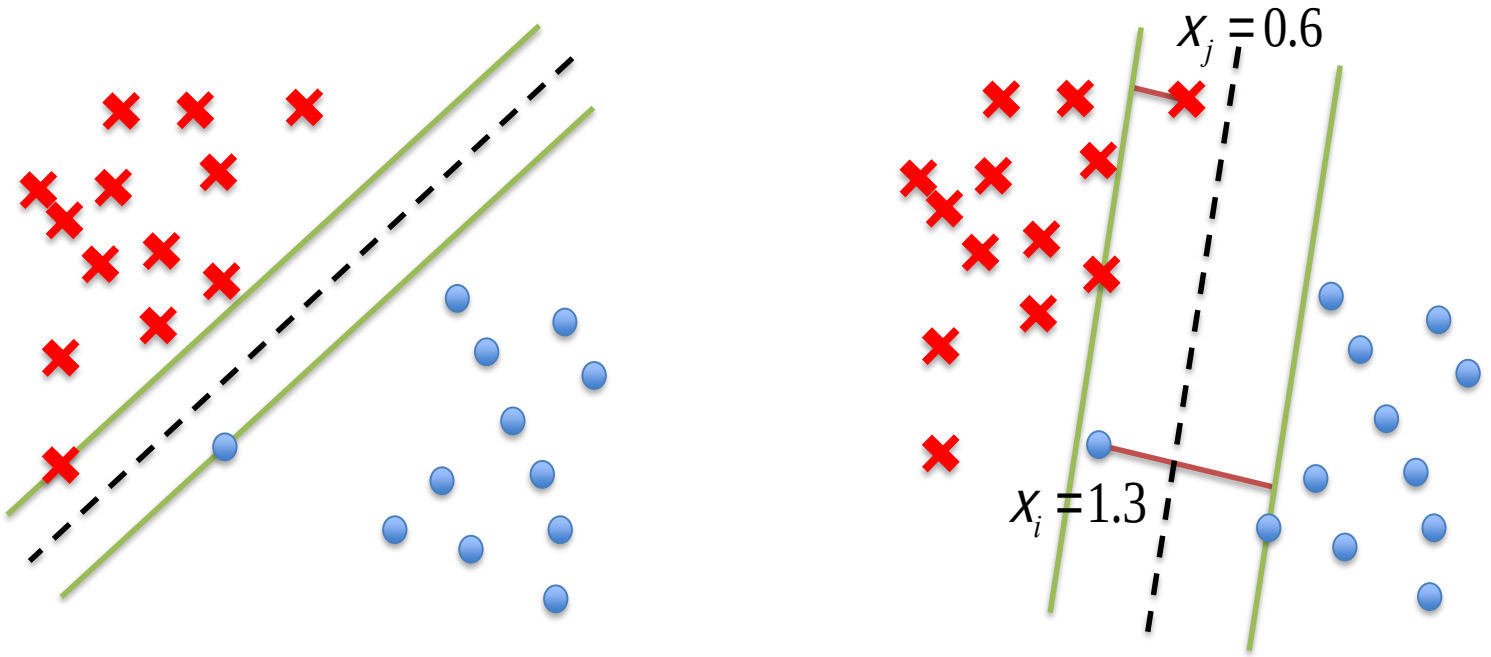
How to manage outliers: Slack variables

(aka soft margins)





How to manage outliers: Slack variables
(aka soft margins)



How to manage outliers: Slack variables
(aka soft margins)

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b) \geq 1 - \xi_i \qquad i = 1, \ldots, N$$

$$\xi_i \geq 0$$

The only parameter C controls the tradeoff between the accuracy w.r.t. to the training data and the maximization of the margin.

It can be interpreted also as a regularization term:

- small C allows constraints to be easily ignored $\rightarrow$ large margin
- large C makes constraints hard to ignore $\rightarrow$ narrow margin
- $C = \infty$ enforces all constraints: hard margin

The dual representation

$$\text{maximize} \quad L_D(\lambda_1,\ldots,\lambda_N) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$0 \leq \lambda_i \leq C, \text{ for all } i = 1\ldots N$$

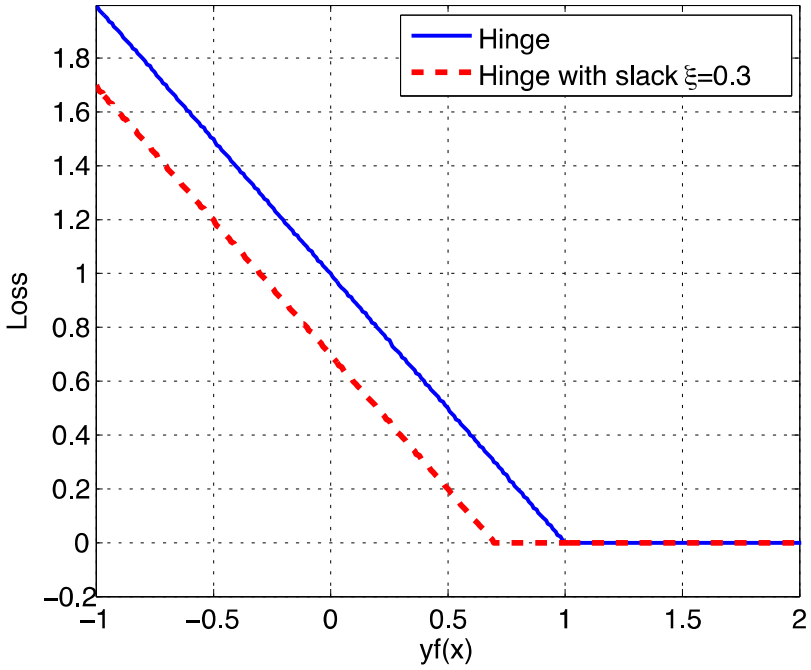
Notes: The hyperplane whose weight vector solves this quadratic oprimization problem is called the **soft margin hyperplane**

The soft-margin optimization problem is equivalent to that of the maximum margin hyperplane with the additional constraint $\lambda_i \leq C$ (Box constraints).

This approach limits the effect of outliers (for which λ_i tends to be large).

Hinge loss (again)

$$L_{hinge} = \max\left\{0, 1 - y_i f(\mathbf{x}_i) - \xi_i\right\},$$

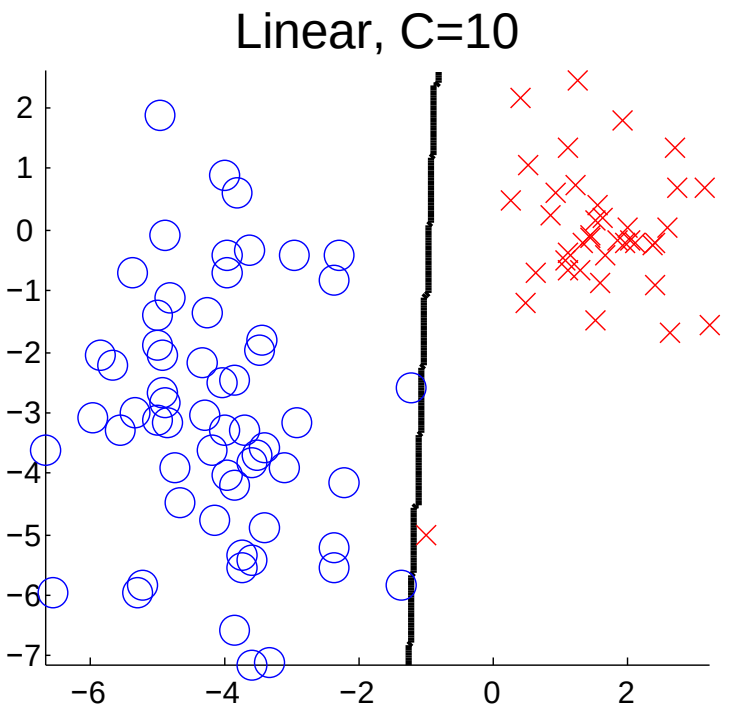


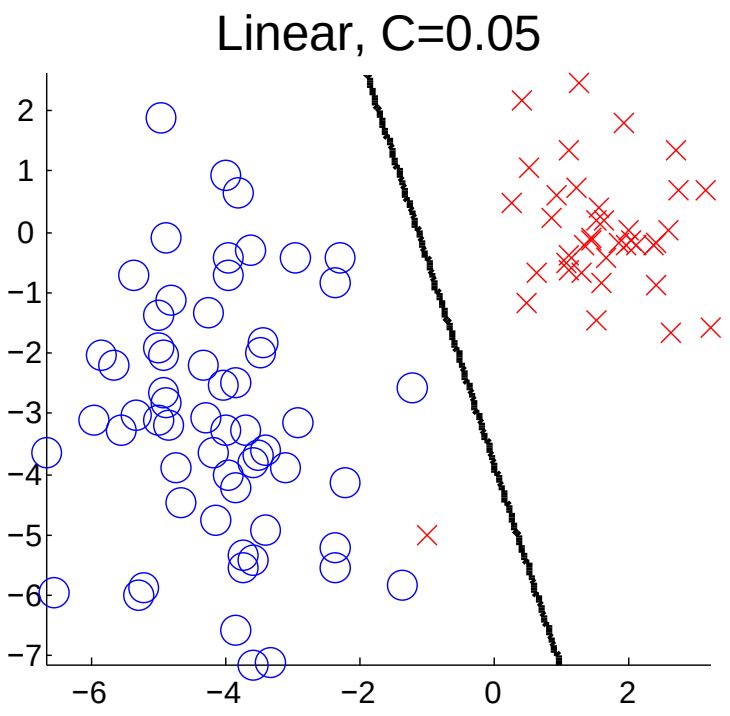
$$E = \sum_{i=1}^{N} \max\left\{0, 1 - y_i f(\mathbf{x}_i) - \xi_i\right\} + \frac{1}{2}\sum_{j=1}^{d} w_j^2 + C\sum_{i=1}^{N} \xi_i$$

slack

penalty for using slack

Example





Application: Pedestrian detection

Goal: Detect (localize) standing humans in images

- reduces object detection to binary classification

- does an image window contain a person or not?

Method: the HOG detector

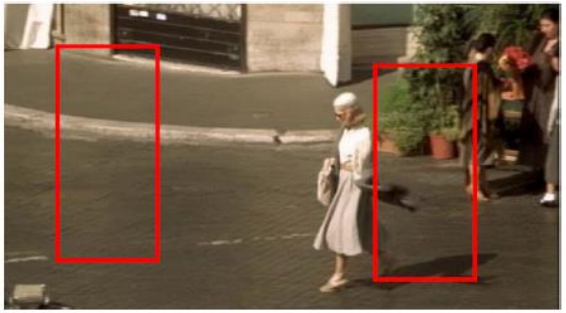
From: A. Zisserman

Training data

- Positive data – 1208 positive window examples




- Negative data – 1218 negative window examples (initially)

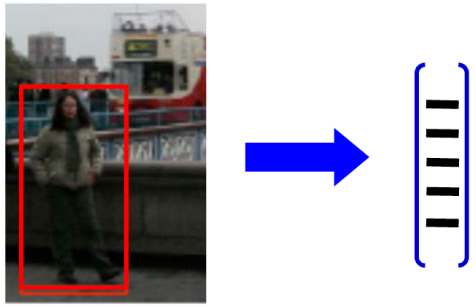



From: A. Zisserman

The algorithm

Learning phase

- Represent each example window by a HOG (Histogram of Oriented Gradients) feature vector:



$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

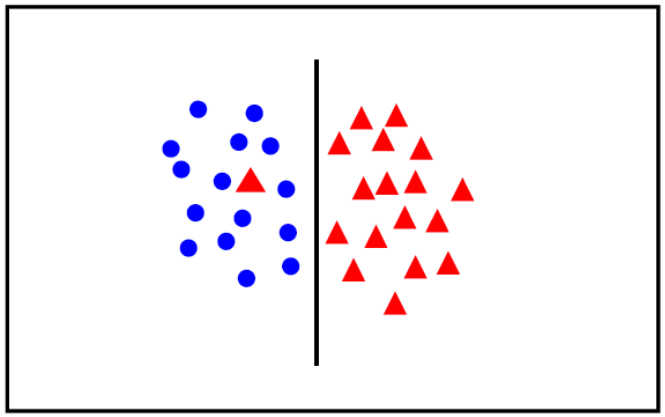
- Train a linear SVM classifier

Testing (Detection)

- Sliding window SVM

From: A. Zisserman

Nonlinear SVM's

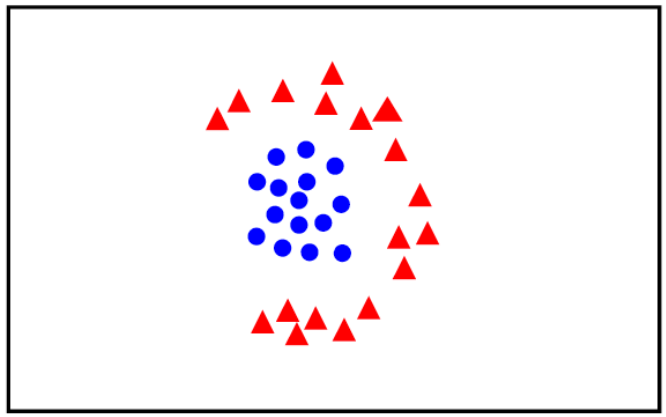


- introduce slack variables

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

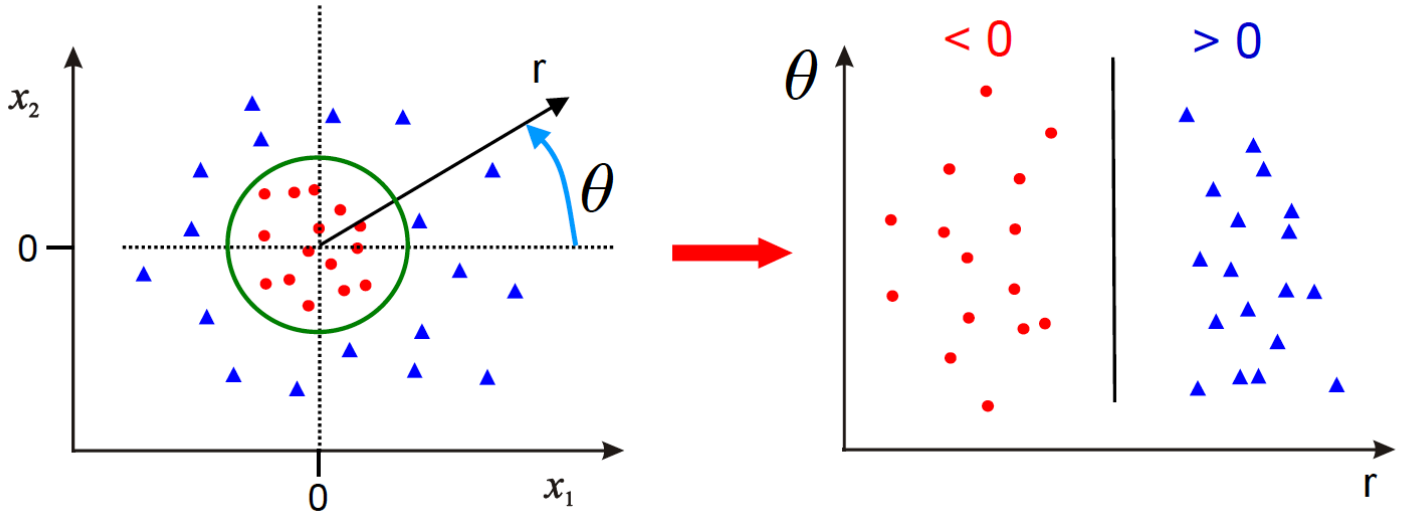


- linear classifier not appropriate

??

From: A. Zisserman

An example



- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

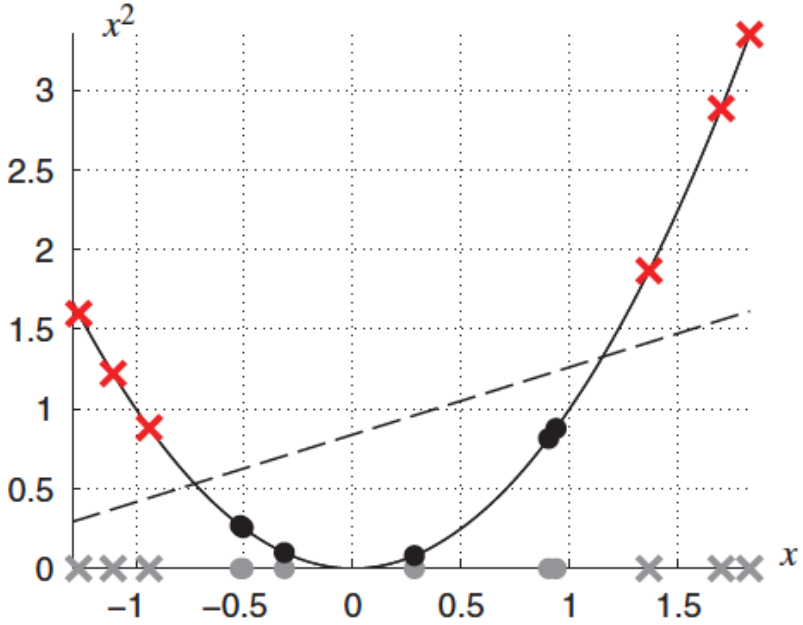
$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

From: A. Zisserman

Cover's theorem

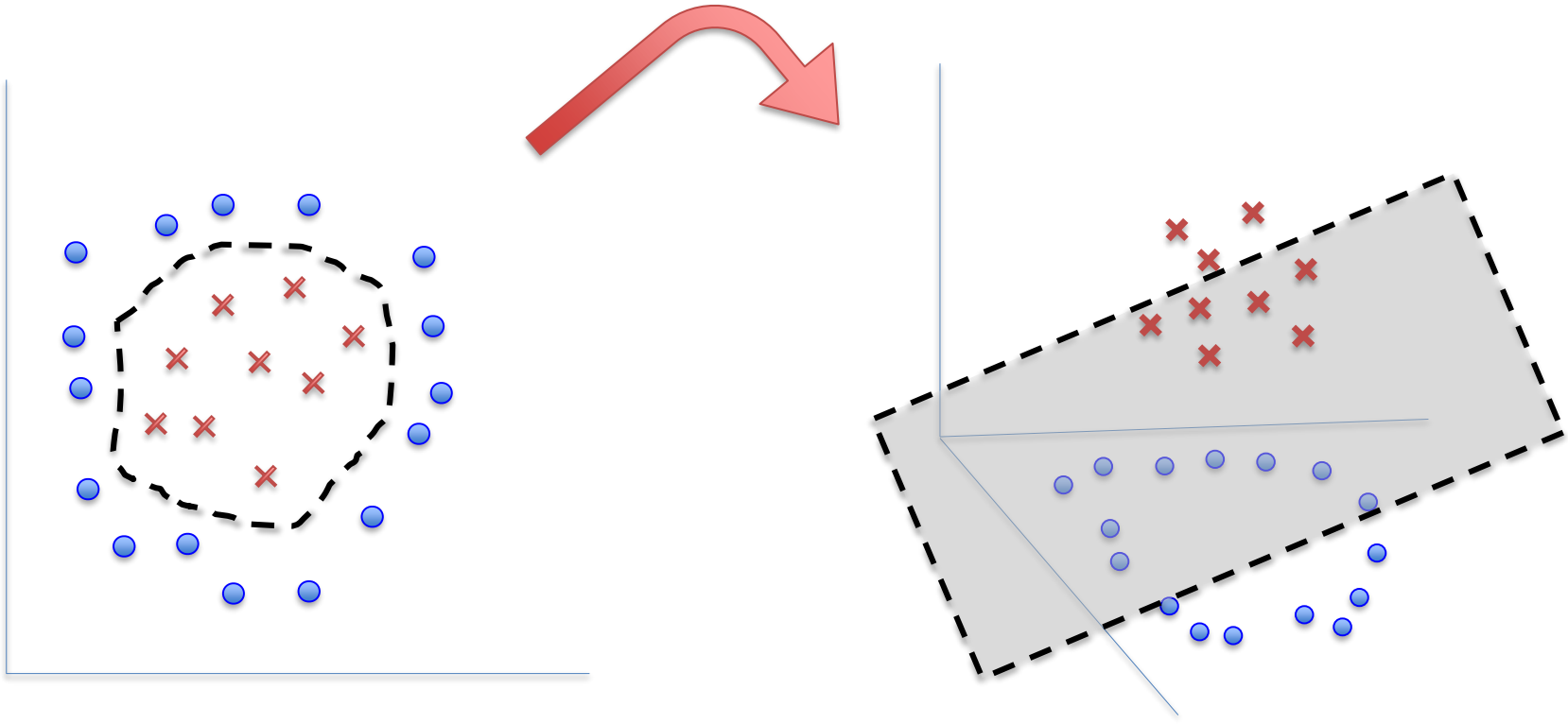
"A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space"

Thomas Cover (1965)



Main idea

Project into high dimensional space, and solve with a linear model

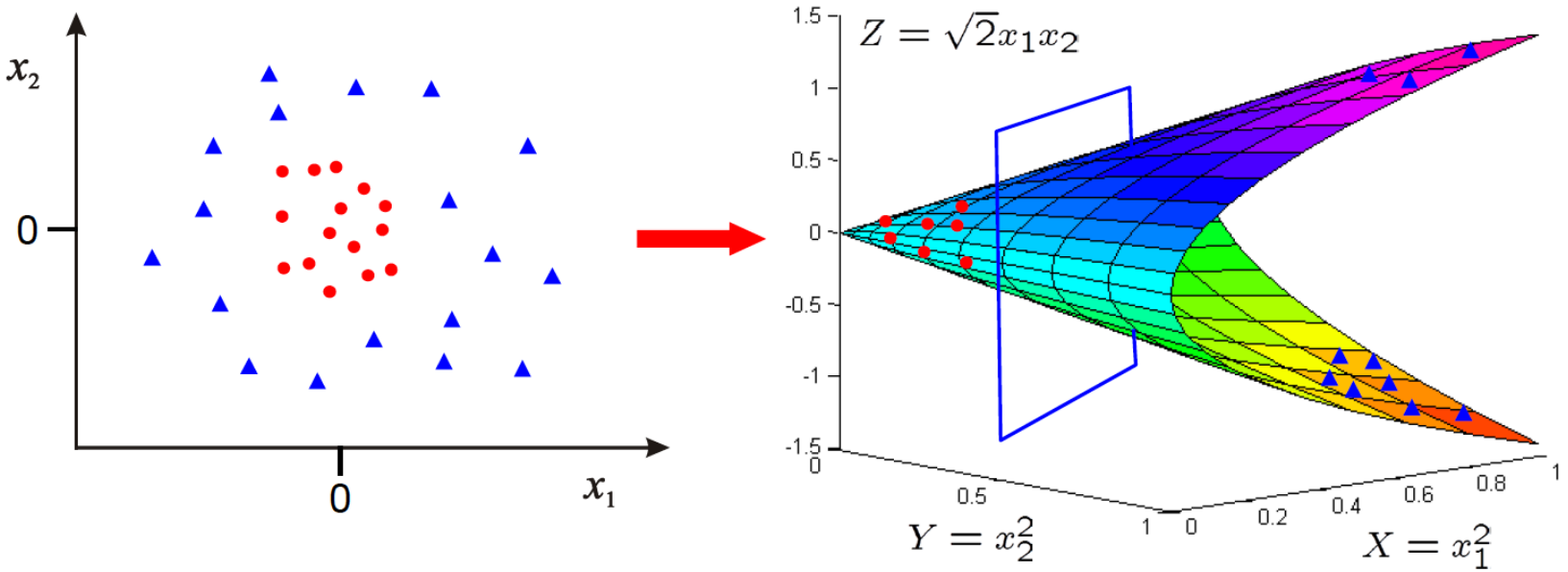


2D

3D

An example

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

From: A. Zisserman

SVM's and Cover's theorem

The power of SVM's resides in the fact that they represent a robust and efficient implementation of Cover's theorem

Nonlinear SVM's operate in two stages:

- Perform a (typically implicit) non-linear mapping of the feature vector $\mathbf{x}$ onto a high-dimensional space that is hidden from the inputs or the outputs

- Construct an optimal separating hyperplane in the high-dim space

The "kernel trick"

Note that in the dual representation of SVM's the inputs appears **only in a dot-product form**:

$$\text{maximize} \quad L_D(\alpha_1,\ldots,\alpha_N) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$$

$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \text{ for all } i = 1\ldots N$$

The discriminant function obtained from the solution is:

$$f(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b$$

The "kernel trick"

Suppose we first mapped the data to some other (possibly infinite dimensional) Euclidean space, using a mapping:

$$\mathbf{x} \rightarrow \Phi(\mathbf{x})$$

By setting $K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{y})$, we obtain:

$$\text{maximize } L_D(\lambda_1,\ldots,\lambda_N) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$0 \leq \lambda_i \leq C, \text{ for all } i = 1\ldots N$$

The discriminant function obtained from the solution becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{N} y_i \lambda_i K(\mathbf{x}_i, \mathbf{x}) + b$$

No need to compute $\Phi(\mathbf{x})$!

Example kernels

Linear kernel $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{y}$

Polynomial kernel $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x}^{\mathrm{T}}\mathbf{y})^{d}$ (for any $d > 0$)

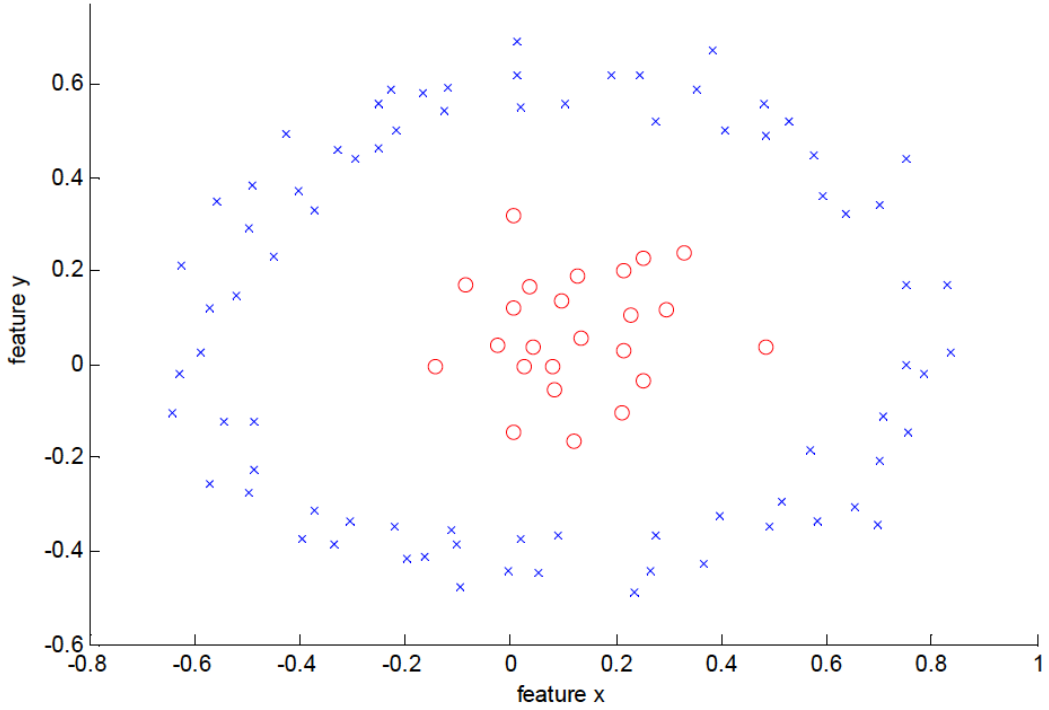
Gaussian kernel $K(\mathbf{x},\mathbf{y}) = \exp\{ -\| \mathbf{x} - \mathbf{y} \|^2 / 2\sigma^2 \}$ (for $\sigma > 0$)

For some kinds of kernel functions, we have:

$$K(\mathrm{x}_i, \mathrm{x}') = \phi(\mathrm{x}_i)^T \phi(\mathrm{x}')$$ **Mercer's condition**

SVM's with Gaussian kernels
(aka Radial Basis Function SVM's)

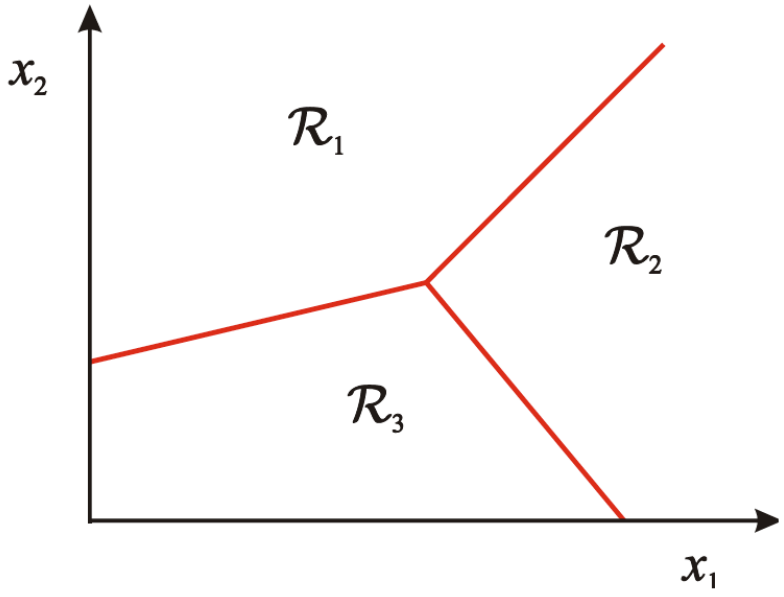
$$f(\mathbf{x}) = \sum_i y_i \lambda_i \exp\{ -\| \mathbf{x} - \mathbf{x}_i \|^2 / 2\sigma^2 \} + b$$



Multi-class problems

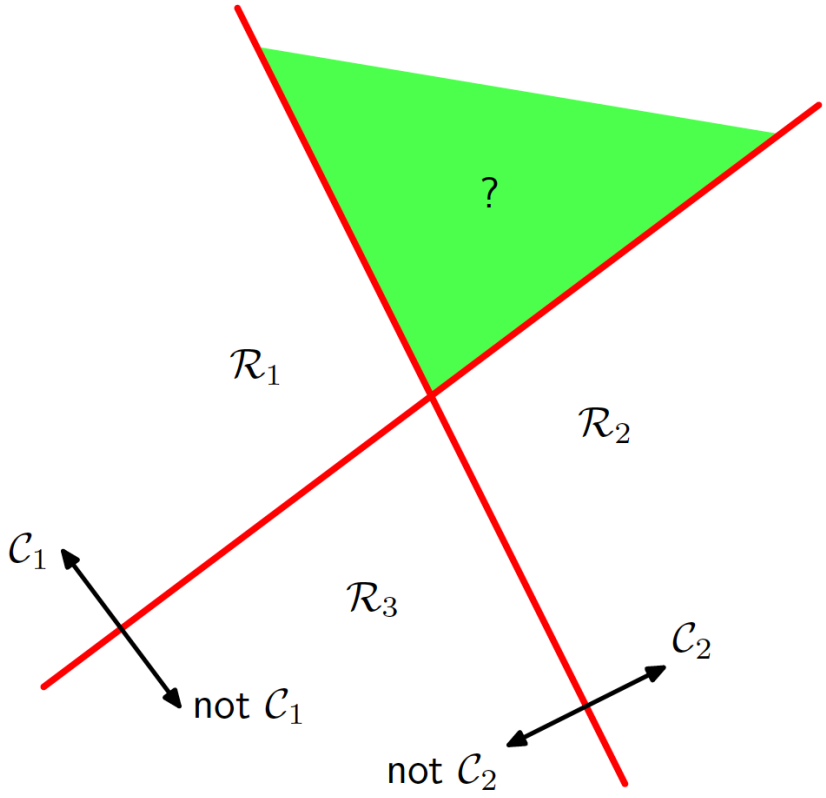
Assign input vector to one of K classes

Goal: a decision rule that divides input space into K decision regions separated by decision boundaries



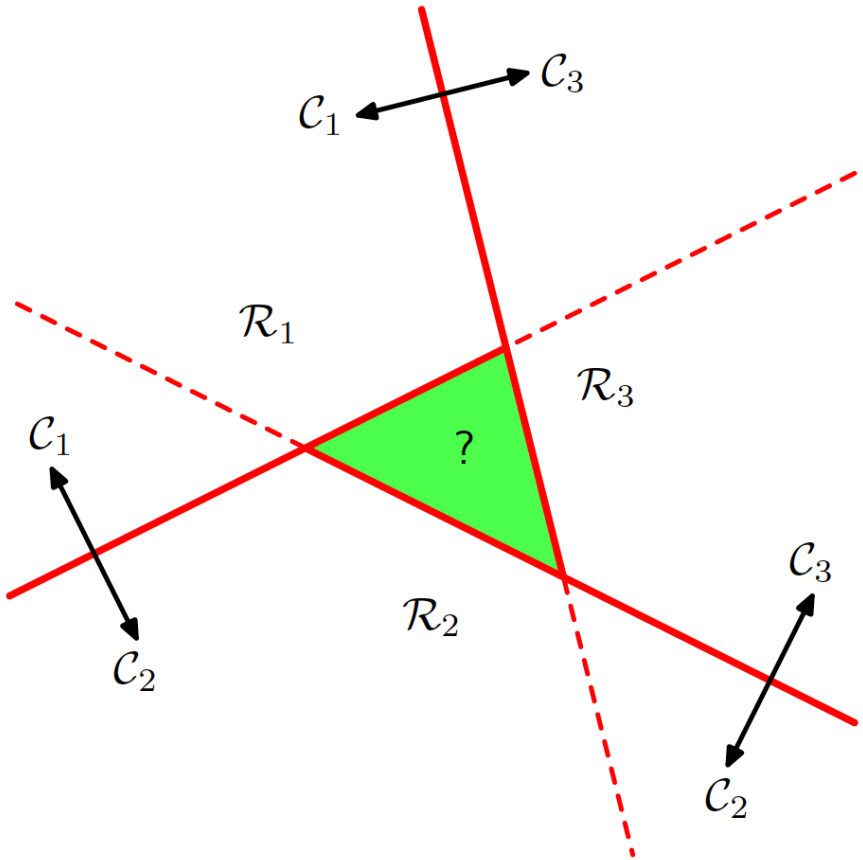
One-vs-the-rest classifiers

Train K−1 classifiers each of which solves a two-class problem of separating points in a particular class from points not in that class



One-vs-one classifiers

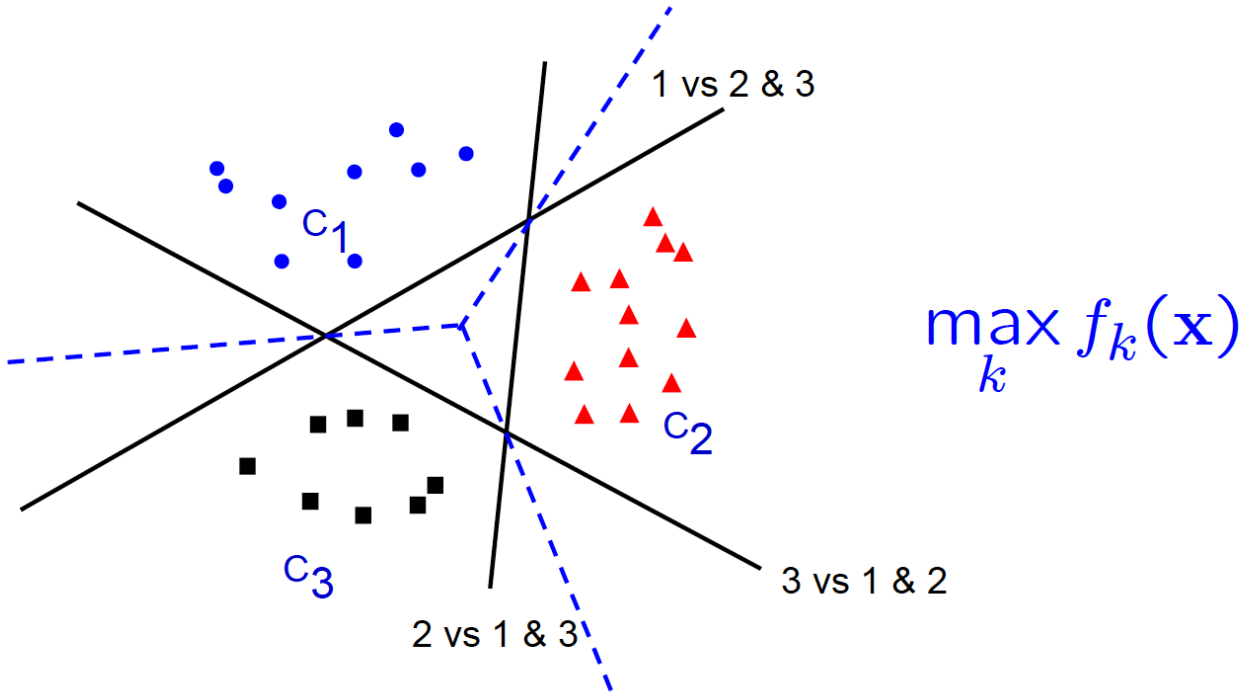
Train K(K−1)/2 binary classifiers, one for every possible pairs of classes.



The typical approach

Learn: Train K *one-vs-the rest* classifiers

Classification: choose the class with the most positive score



References

- C. Burges, A tutorial on support vector machines for pattern recogniton. *Data Mining and Knowledge Discovery* 2(2):121-167 (1998)

- K. Muller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopk. An introduction to kernel-based learning methods. *IEEE Transactions on Neural Networks*, 12(2) (2001)

- N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel Based Learning Methods*. Cambridge University Press (2000)

- J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK (2004)