Data Mining: from theory, to applications, to business opportunities

Theodoros Evgeniou
Assistant Professor
Technology Management
INSEAD

SETN 2002

Plan

Part I: Statistical Learning Theory (1.5 h)
a) Learning from examples and Empirical Theories
b) Underline theory and VC complexity

Part II: Learning Methods (1.25 h)
a) Support Vector Machines and Regularization Networks
b) Kernels and Kernel Machines

Part III: Applications (1 h)

Part IV: Business Issues (.5 h)

Data Mining, Learning, Statistics, AI, etc

Data Mining consists of many steps (i.e. define the problem, choose the data, choose data representations, clean the data, measure basic statistics, develop learning models, evaluate the models, - etc - then iterate)

Learning (supervised): once you have clean data describing a relation between inputs and outputs of a process, develop a model that captures this relation.

"Learning is at the core of intelligence", T. Poggio

Learning (as function approximation) from examples

The basic goal of **supervised learning** is to use a training set S consisting of ℓ samples (datapoints, examples, data) (x_i, y_i) drawn i.i.d. from an unknown probability distribution $P(x, y)$, namely $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$, to construct a function f_S that given a new x_{new} predicts the associated value of y:

$$y_{pred} = f_S(x_{new})$$

It is crucial to note that we view $P(x, y)$ as **fixed** but **unknown**.

Regression and Classification

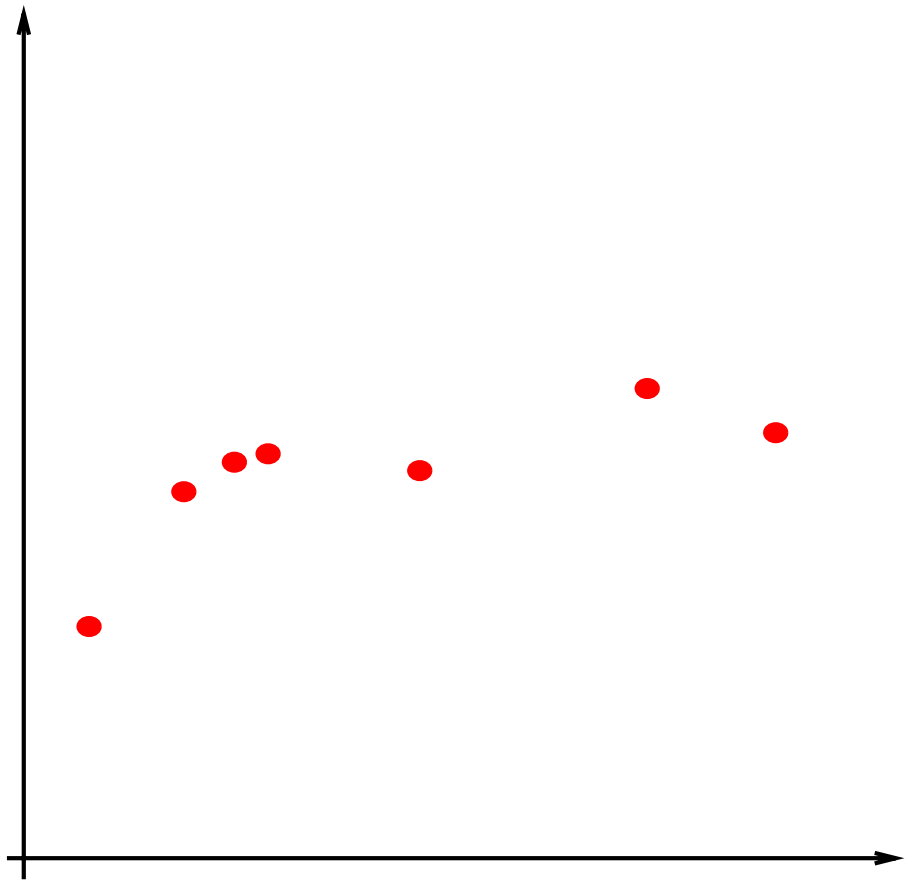
If y is a real-valued random variable, we have **regression**.

If y takes values from a finite set, we have **pattern classification**. In two-class pattern classification problems, it is often mathematically convenient to assign one class a y value of 1, and the other class a y value of -1.

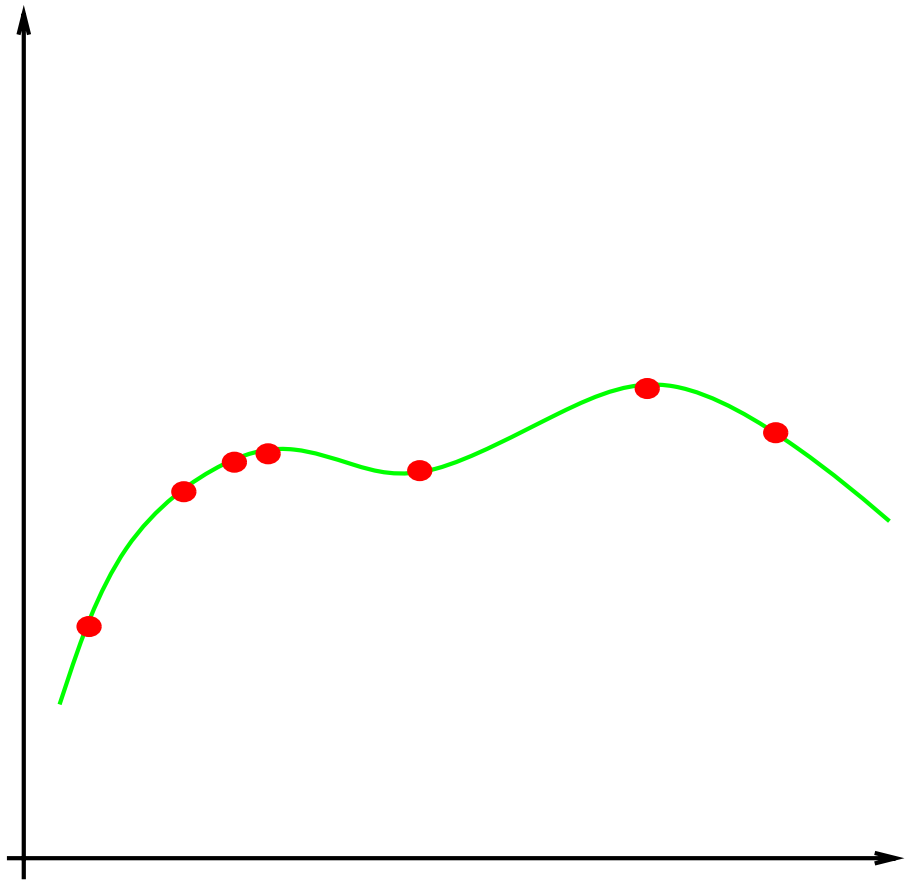
Some Examples

- $\mathbf{x}$ is a vector representing an image (i.e. pixel values), y is $+1$ if the image is that of a person, -1 otherwise. Learn how humans look.
- $\mathbf{x}$ is a vector representing an email (i.e. frequency of words), y is $+1$ if the email is about a complaint, -1 if it is about a request.
- $\mathbf{x}$ is a vector of measurements from a chemical or biological process, y is the time until a certain event occurs.

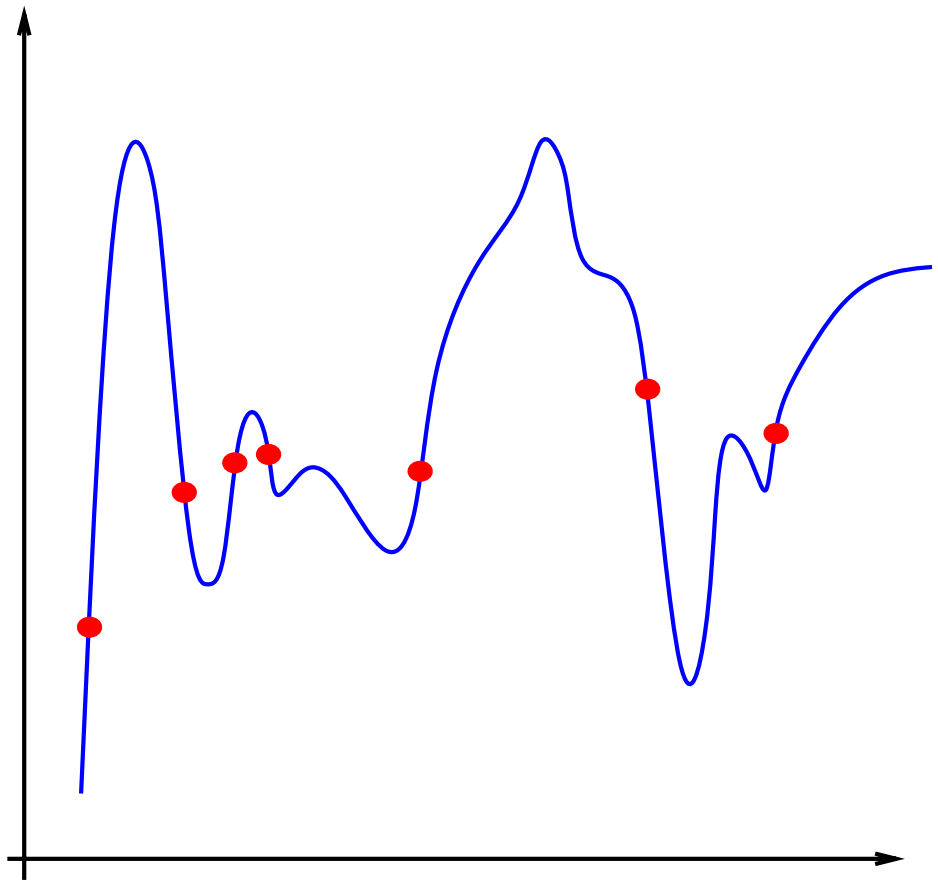
Given a certain number of examples...



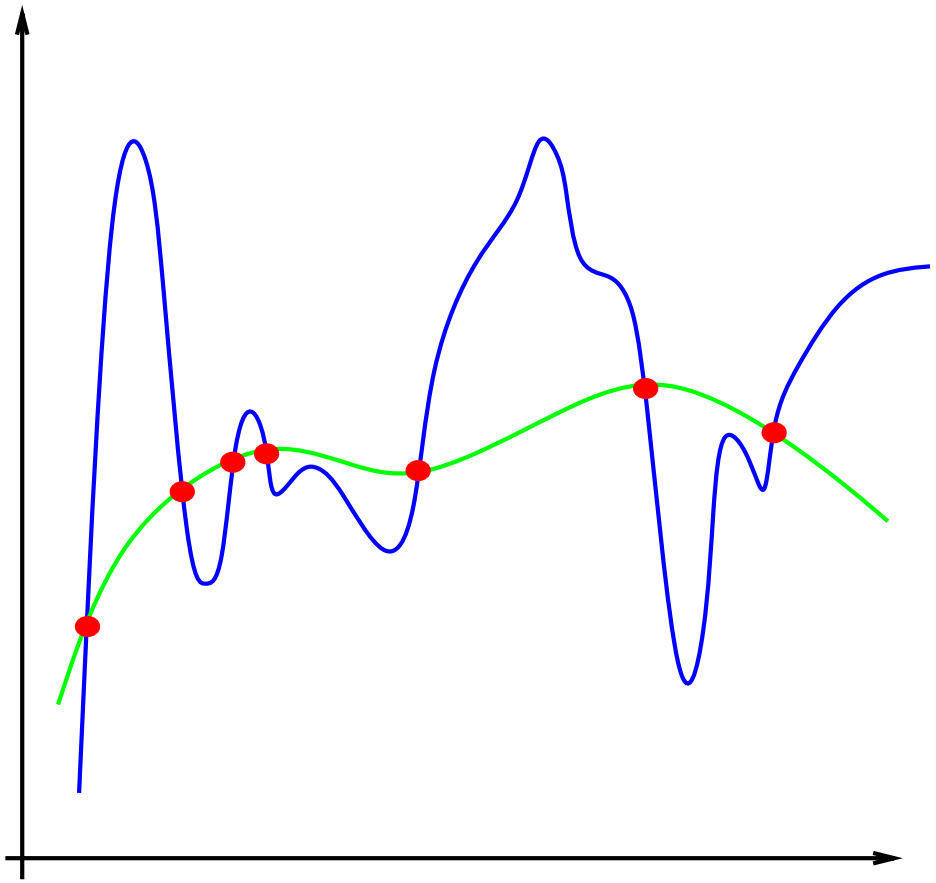
here is one possible solution...



... but here is another (and very different) one!



Which one should we pick?



Central (Philosophical) Question

How to develop a "theory" (be it a function, model, or lifestyle theory) from "empirical data" (be it training data, observations, experiences from the past) that is "good".

The Demarcation Problem

"What is the distinction between scientific empirical theories and metaphysical ones?" *(Immanuel Kant,* 18^{th} *century)*

Falsifiability

"A necessary condition for the correctness (scientific value) of an empirical theory is the possibility of its falsification" *(Karl Popper, 20^{th} century)*

Occam's Razor

"One should not increase, beyond what is necessary, the number of entities required to explain anything" ("Keep it simple stupid - but not too simple") *(William Occam, 14^{th} century)*

Complexity and Information

The more data you have, the more complex a theory you can "safely" build.

Statistical Learning Theory:
The 3 forces for good theories

1. Falsifiability

2. Complexity

3. Amount of information

Part I: Loss Functions

In order to measure the goodness of our function, we need a **loss function** V. In general, we let $V(f(x), y^*)$ denote the price we pay when we see x and guess that the associated y value is $f(x)$ when it is actually y^* - when we make an error (false).

Common Loss Functions For Regression

For regression, the most common loss function is the square (L_2) loss:

$$V(f(x), y) = (f(x) - y)^2$$

We could also use the absolute value (L_1) loss:

$$V(f(x), y) = |f(x) - y|$$

For reasons we will see later, we sometimes use Vapnik's ϵ-insensitive L_1 loss function:

$$V(f(x), y) = |f(x) - y|_\epsilon$$

Common Loss Functions For Classification

For binary classification, the most intuitive loss is the 0-1 loss:

$$V(f(x), y) = \Theta(-yf(x))$$

where Θ is the Heavyside function.

For tractability and other reasons, we often use the L_1-hinge loss in binary classification:

$$V(f(x), y) = |1 - y \cdot f(x)|_+$$

The True Error

Given a function f, a loss function V, and a probability distribution P over $X \times Y$, we can define the **true error (risk)** (or just error (risk)) of f as:

$$R(f) = \int V(f(x), y) dP$$

Note that $R(f)$ is also the **expected loss** on a new example drawn at random from the unknown distbution P.

We would like to make $R(f)$ small. Unfortunately, because P is unknown, we cannot measure $R(f)$.

The Empirical Error

Given a function f, a loss function V, and a training set S consisting of ℓ datapoints, we can define the **empirical error (risk)** of f as:

$$R_\ell(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} V(f(x_i), y_i)$$

Notice: Emprirical error = falsifiability

(Will the empirical error of a function be similar to its true error?)

Part II: Target Space, Hypothesis Space

The **target space** $\mathcal{T}$ is a space of functions that is assumed to contain the "true" function that minimizes the error. We can safely assume that $\mathcal{T}$ is all functions in L_2, or all differentiable functions.

The **hypothesis space** $\mathcal{H}$ is the space of functions that we allow our algorithm to search. *It is often chosen with respect to the amount of data available.*

(Note: a hypothesis space that "at the limit" is dense in L_2 is a desired property of any approximation scheme.)

Empirical Risk Minimization Method

Given a training set S and a hypothesis space $\mathcal{H}$, empirical risk minimization is the approach of finding a function $\widehat{f}_{\mathcal{H},S}$ that minimizes the empirical error over all functions $f \in \mathcal{H}$:

$$\begin{aligned} \widehat{f}_{\mathcal{H},S} &= \arg\min_{f} \frac{1}{\ell} \sum_{i=1}^{\ell} V(f(x_i), y_i) \\ \text{s.t.} \quad & f \in \mathcal{H}. \end{aligned}$$

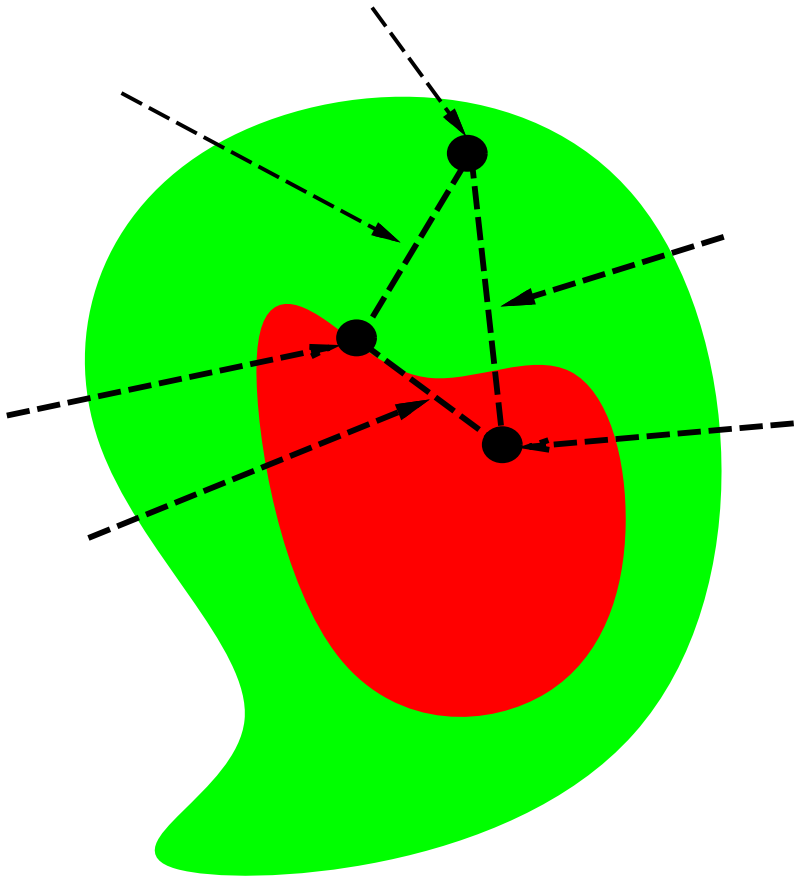
Approximation Error and Estimation Error

Let f_0 be the function in $\mathcal{T}$ with the smallest true error.
Let $f_{\mathcal{H}}$ be the function in $\mathcal{H}$ with the smallest true error.

The **generalization error** of our empirical error minimizer is the sum of the **estimation error** and the **approximation error**:

$$R(\widehat{f}_{\mathcal{H},\mathcal{S}}) - R(f_0) = (R(\widehat{f}_{\mathcal{H},\mathcal{S}}) - R(f_{\mathcal{H}})) + (R(f_{\mathcal{H}}) - R(f_0))$$

We want to minimize the generalization error.



Consistency of the empirical risk minimization method

The empirical risk minimization method is said to be **consistent** for the set $\{f \in \mathcal{H}\}$, the loss function V, and for the probability distribution $P(x, y)$ if

$$\lim_{\ell \to \infty} R[f_{S,\mathcal{H}}] \stackrel{p}{=} R[f_{\mathcal{H}}]$$

and

$$\lim_{\ell \to \infty} \widehat{R}_{emp}[f_{S,\mathcal{H}}] \stackrel{p}{=} R[f_{\mathcal{H}}]$$

That is, the expected error of the minimizer of the empirical error converges to the best possible expected error in the hypothesis space $\mathcal{H}$.

Uniform convergence

Two sided uniform convergence in probability is defined as:

$$\lim_{\ell \to \infty} P \left\{ \sup_{f \in \mathcal{H}} |R(f) - \widehat{R}(f)| > \epsilon \right\} = 0 \quad \forall \epsilon > 0$$

Uniform convergence is a necessary and sufficient condition for the empirical risk minimization approach to be consistent in a hypothesis space $\mathcal{H}$

Uniform convergence for one function

If our $\mathcal{H}$ consisted of one function f_1 we can show that:

$$P\left\{|R(f_1) - \widehat{R}(f_1)| > \epsilon\right\} < \exp(-\epsilon^2 \ell C).$$

If our loss function is bounded $0 \leq V(f_1(x), y) \leq B$ then we can use Hoeffdings inequality which says: let X be a set and D a distribution on X and let $f : X \to [a, b]$ be a function. Then

$$P\left\{\left|\frac{1}{\ell}\sum_{i=1}^{\ell} f(x_i) - E_D f(x)\right| \geq \epsilon\right\} \leq 2\exp(-2\epsilon^2 \ell/(a-b)^2).$$

Applying the inequality results in

$$P\left\{|R(f_1) - \widehat{R}(f_1)| \geq \epsilon\right\} \leq 2\exp(-\epsilon^2 \ell/B^2).$$

Uniform convergence for k functions

If our $\mathcal{H}$ consisted of k functions $f_1, ..., f_k$ we need to show

$$P\left\{\sup_{f_i : i=1,..,k} \left|R(f_i) - \hat{R}(f_i)\right| > \epsilon\right\} < \exp(-\epsilon^2 \ell C).$$

We know that for one function f_i

$$P\left\{|R(f_i) - \hat{R}(f_i)| \geq \epsilon\right\} \leq 2\exp(-\epsilon^2 \ell / B^2).$$

We need the above to hold for all k functions. So we apply the union bound

$$P(a \cup b \cup c) \leq P(a) + P(b) + P(c)$$

so

$$P\left\{\sup_{f_i : i=1,..,k} |R(f_i) - \hat{R}(f_i)| \geq \epsilon\right\} \leq 2k\exp(-\epsilon^2 \ell / B^2).$$

Uniform convergence for a general hypothesis space

In general our hypothesis space is not a finite set of functions (i.e. all linear functions in R^d).

For continuous functions, we can count the number of functions in a hypothesis space using the topological notion of a minimum ϵ-net. That is, for any r (= the "ϵ" of the ϵ-net) we compute what is the minimum number N of functions $g_1, ..., g_N \in \mathcal{H}$ that we can find such that for any $f \in \mathcal{H}$ we have that $\sup_x |f(x) - g_i(x)| < r$ for at least one g_i.

Now we can say

$$P\left\{\sup_{f\in\mathcal{H}} |R(f) - \hat{R}(f)| \geq \epsilon\right\} \leq 2\mathcal{N}(\mathcal{H}, r(\epsilon)) \exp(-\epsilon^2 \ell / B^2).$$

Counting classification functions

For classification functions, we can do something else:

Given ℓ points $\{(x_1, y_1), ..., (x_\ell, y_\ell)\}$, for every $f \in \mathcal{H}$ we get different "labelings" $\{\Theta(-y_1 f(x_1)), ..., \Theta(-y_\ell f(x_\ell))\}$

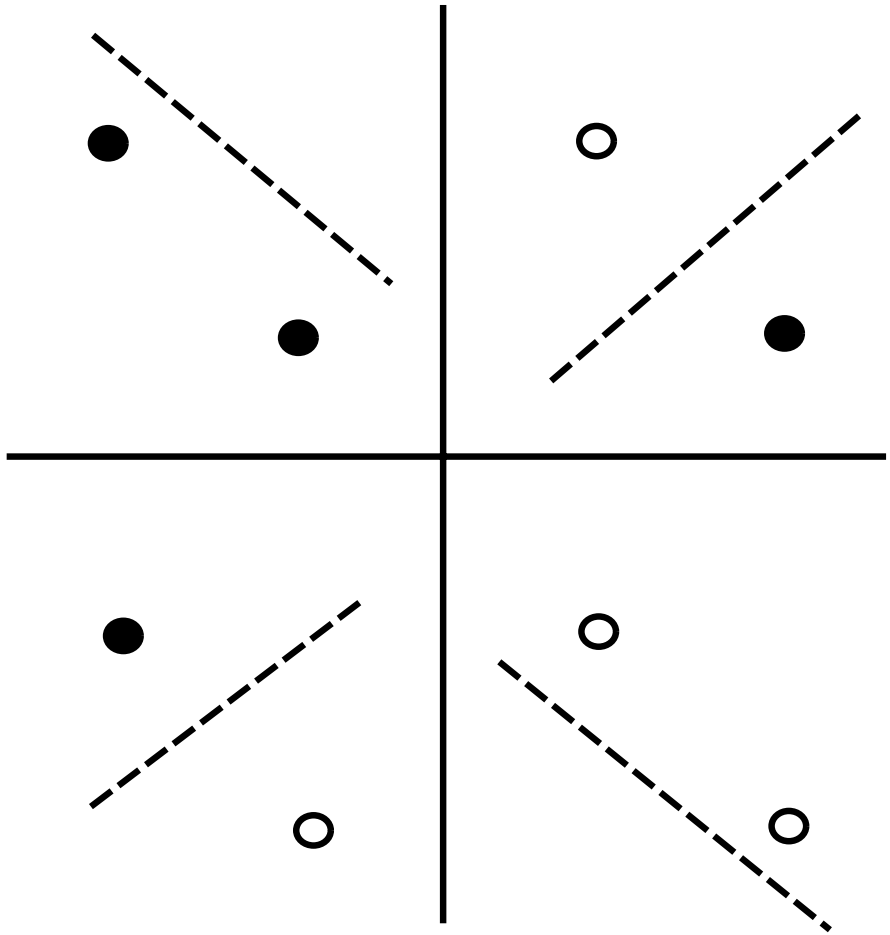
We define the random VC entropy of a hypothesis space $\mathcal{H}$ as the number of labelings that can be implemented over $f \in \mathcal{H}$ written as

$$N^{\mathcal{H}}((x_1, y_1), ..., (x_\ell, y_\ell)).$$

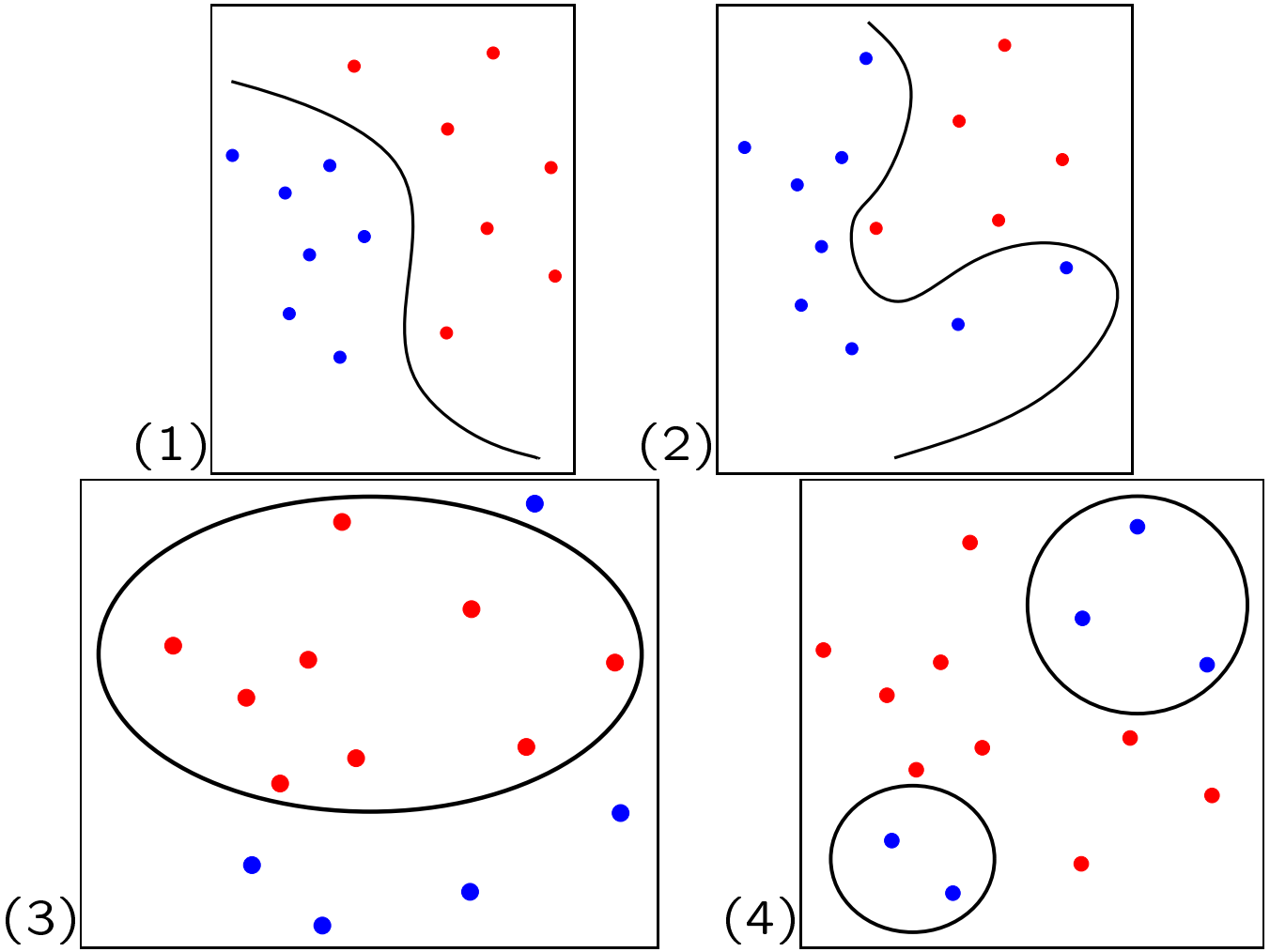
An obvious property of $N^{\mathcal{H}}((x_1, y_1), ..., (x_\ell, y_\ell))$ is:

$$N^{\mathcal{H}}((x_1, y_1), ..., (x_\ell, y_\ell)) \leq 2^\ell.$$

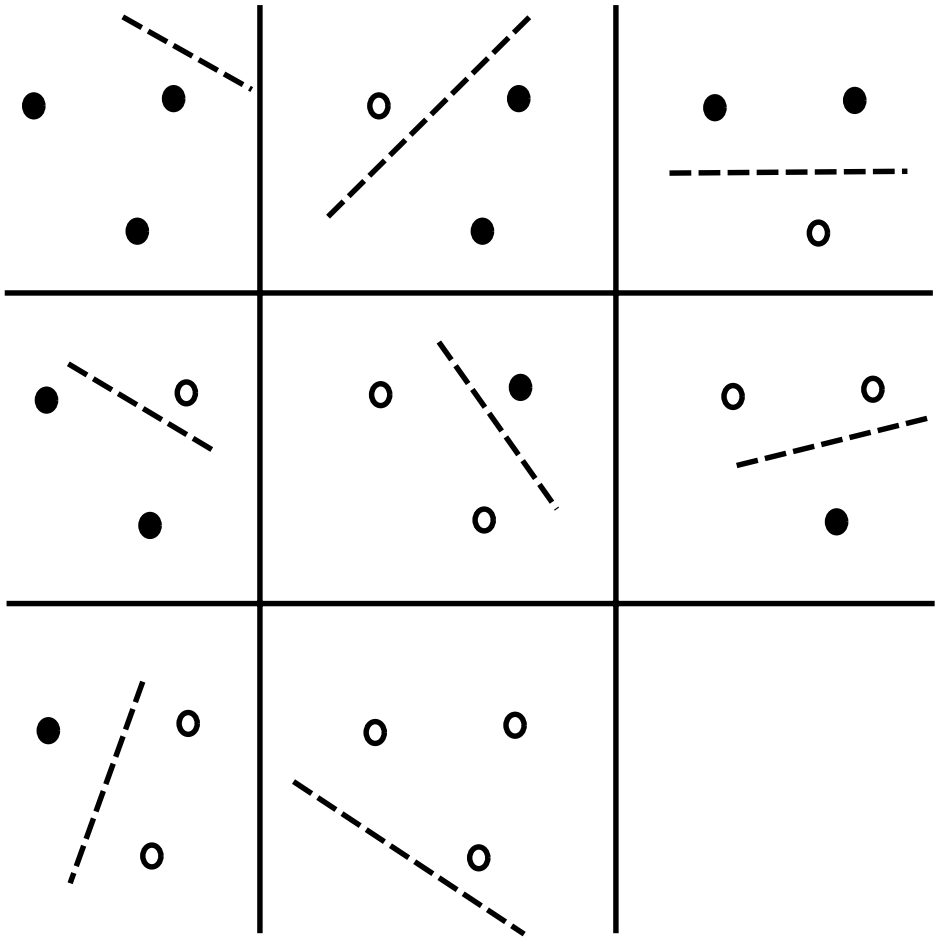
Implementation of different labelings



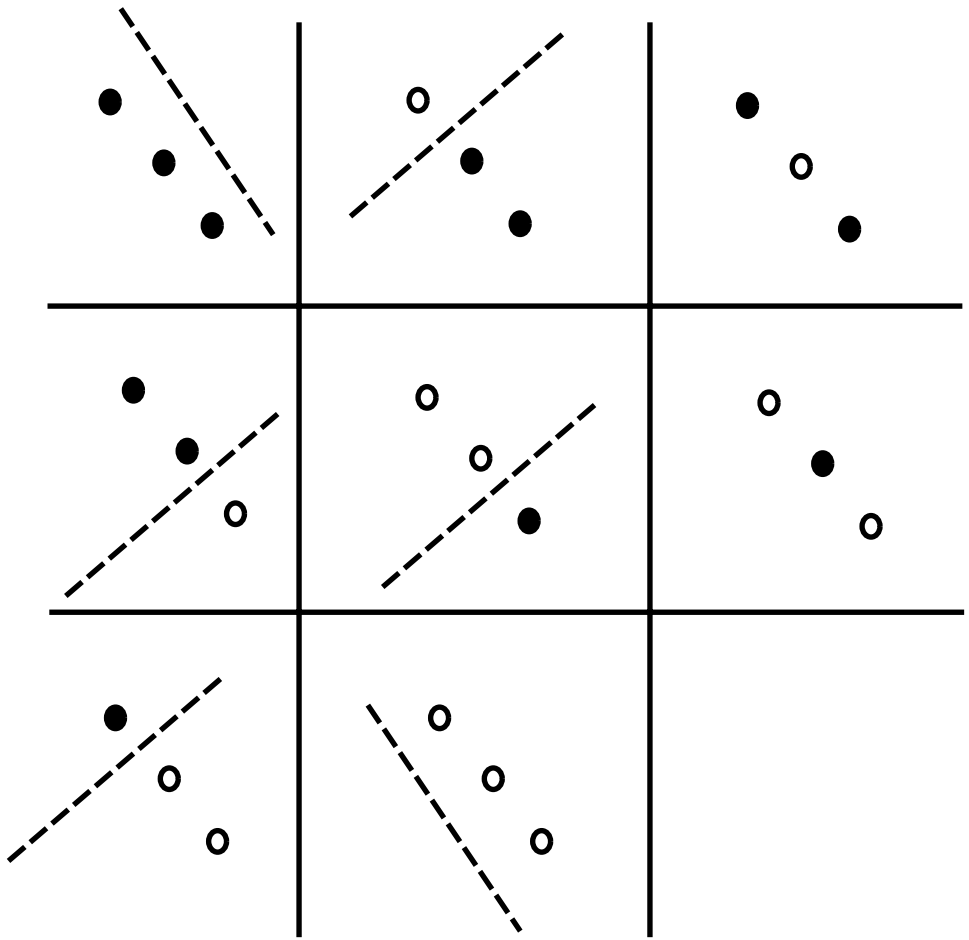
Implementation of different labelings



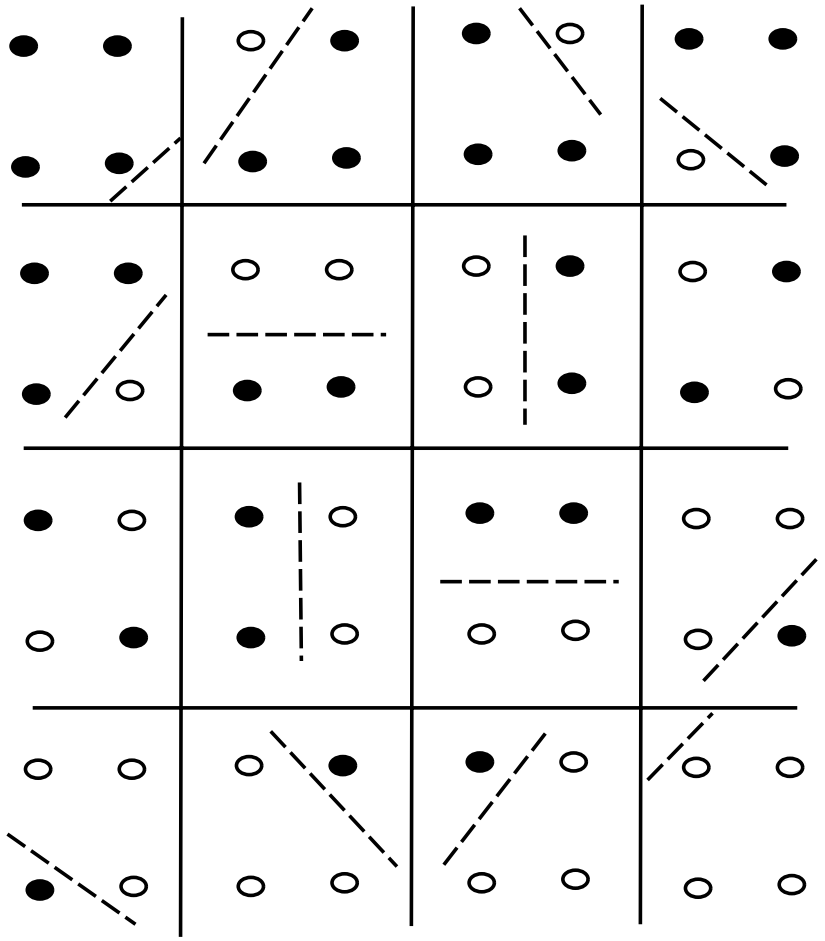
The 8 possible labelings of 3 points in 2D using lines



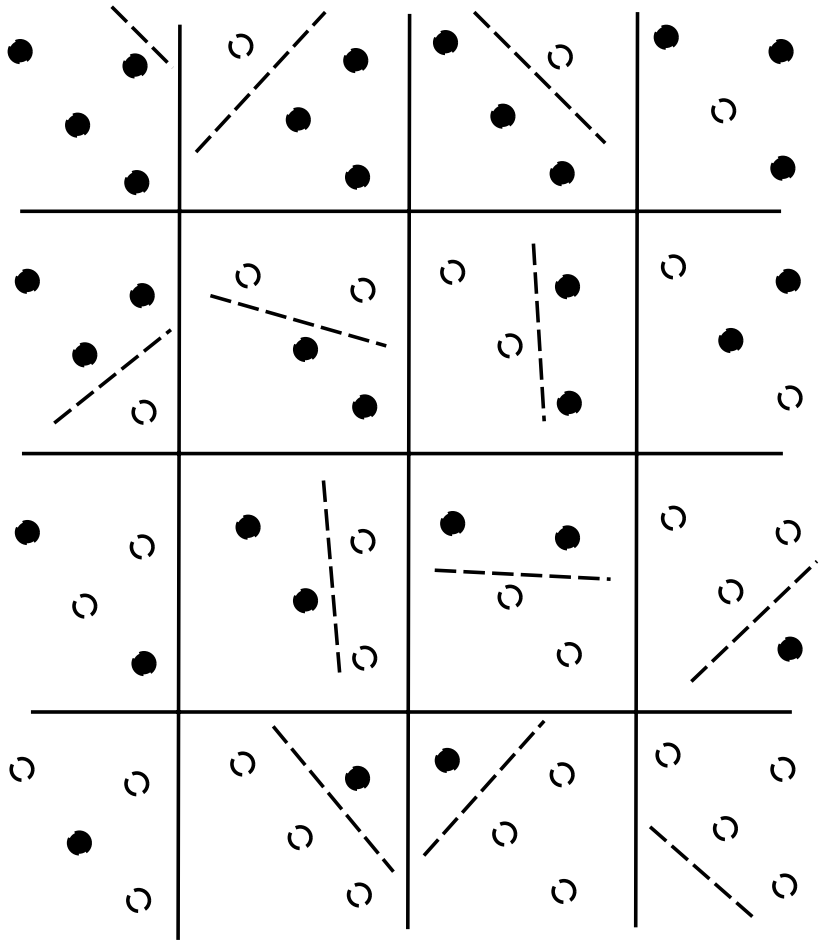
Example



Example



Example



How Many Labelings?
Sauer's Lemma

If the hypothesis space can separate h points in all possible (2^h ways), then $\ell > h$ points can be labelled in

$$\sum_{i=1}^{h} \binom{\ell}{i} < \left(\frac{e\ell}{h}\right)^h$$

possible ways and

$$\sum_{i=1}^{h} \binom{\ell}{i} < 2^\ell.$$

VC-dimension

The VC-dimension of a set of binary functions is h if and only if

- There is **at least one set of h points** that can be labeled in all possible ways;
- there is **no set of $h+1$ points** that can be labeled in all possible ways;

VC-dimension and free parameters

The VC-dimension is proportional, but not necessarily equal, to the number of parameters.

- For Multilayer Perceptrons with hard thresholds $h \propto n \ln n$ (Maass, 1994);

- For Multilayer perceptrons with standard sigmoid thresholds $h \propto n^2$ (Koiran and Sontag, 1995);

- For classification functions of the form $\theta(-y \sin(\alpha x))$ the VC-dimension is infinite;

Classification

The finiteness of the VC-dimension of the set of functions $f \in \mathcal{H}$ for the classification loss is a **necessary and sufficient** for the uniform convergence of the empirical risk minimization in a bounded function class for arbitrary probability distributions with a fast rate of convergence.

VC type Bounds

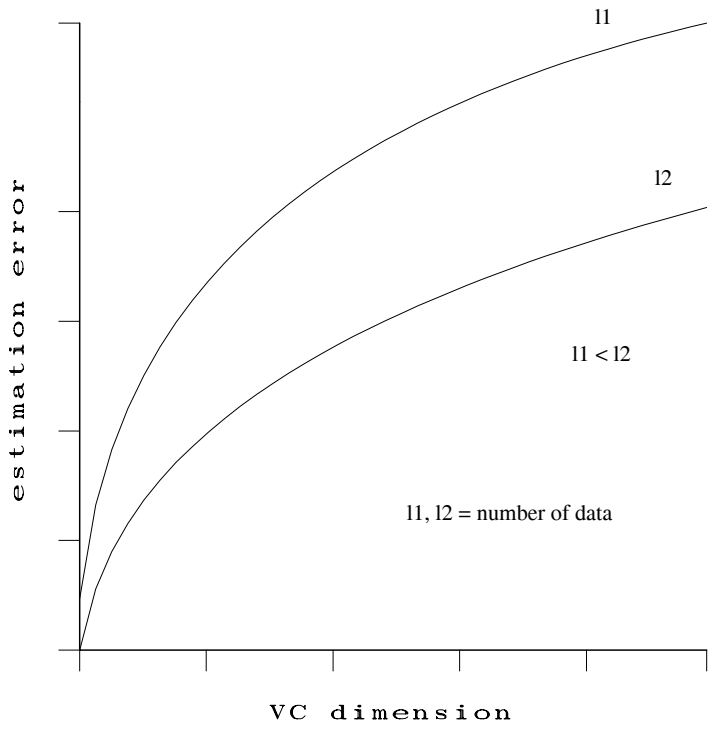
In the case of classification:

$$P\left\{\sup_{f\in\mathcal{H}}\left|R(f)-\widehat{R}(f)\right|>\epsilon\right\}<2\left(\frac{e\ell}{h}\right)^{h}\exp(-2\epsilon^{2}\ell).$$

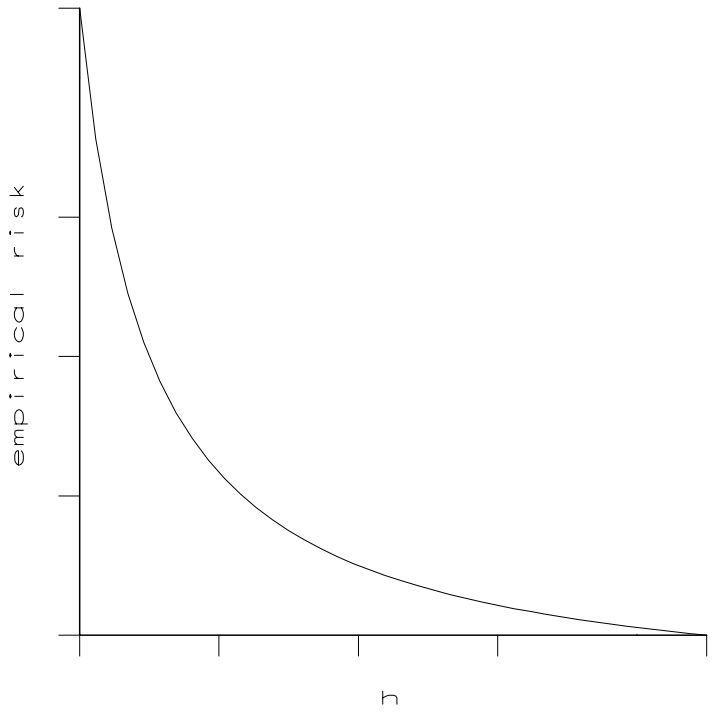
Which allows us to state that with probability $1-\delta$

$$R(f)\leq\widehat{R}(f)+\sqrt{\frac{h\ln(e\ell/h)-\ln(\delta/2)}{\ell}}.$$

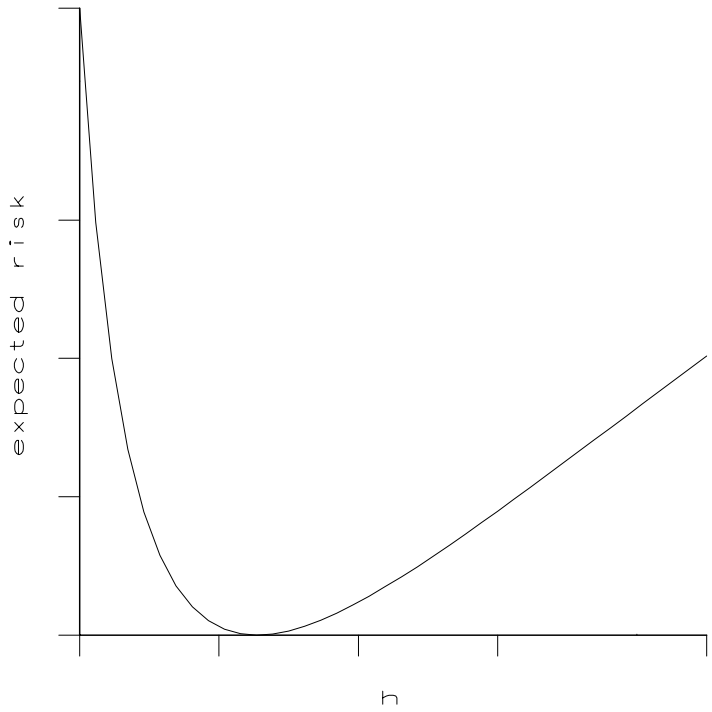
There are many such bounds for classification *and* for regression. They relate *the empirical error, the expected error, the number of data, and the complexity of the hypothesis space.*



Estimation error as a function of the VC dimension for different number of data points ℓ_1 and ℓ_2, with $\ell_1 < \ell_2$.



Empirical error as a function of the VC-dimension h (typical behavior).



Guaranteed error (that is, bound on the expected error) as a function of the VC-dimension h (typical behavior).

A new induction principle

$$\text{Guaranteed risk} = R_{emp}(f) + \sqrt{\frac{h\left(\ln\frac{2\ell}{h}+1\right)-\ln\left(\frac{\delta}{4}\right)}{\ell}}$$

Expected risk $\leq$ Guaranteed risk

$\Downarrow$

Minimize the guaranteed risk, rather than the empirical risk!

This implies that the VC-dimension (generally the complexity of the hypothesis space) must be one of the variables of the problem (the others being the empirical error and the number of data - REMEMBER THE 3 FORCES FOR GOOD THEORIES?).

Structural risk minimization (SRM)

Let S_h an hypothesis space of VC-dimension h, so that

$$S_1 \subset S_2 \subset \ldots \subset S_h \subset \ldots$$

SRM consists in solving a problem of the *form*

$$\min_{S_h} \left[R_{emp}(f) + \sqrt{\frac{h\left(\ln \frac{2\ell}{h} + 1\right) - \ln\left(\frac{\delta}{4}\right)}{\ell}} \right]$$

Structural risk minimization: examples

- $S_h \equiv \{\text{RBF nets with } h \text{ centers}\}$
- $S_h \equiv \{\text{MLP nets with } h \text{ units}\}$
- $S_h \equiv \{\text{Polynomials of degree } h\}$
- $S_h \equiv \{f \mid \phi[f] \leq h\}$

Structural risk minimization "Type" Methods

Typically (for computational reasons) one solves the optimization problem:

$$\min_f \sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)) + \lambda(\ complexity\ of\ \ f)$$

Choosing V and "complexity/hypothesis space" we get a very large family of learning machines (i.e. Regularizatin Networks, Support Vector Machines, etc) that we will discuss next.

Summary of Part I

- In Statistical Learning Theory the problem of learning from examples is cast in a probabilistic framework where one ideally wants to minimize the expected error of the solution;Since the minimization of the expected error is not feasible, one attempts to minimize the empirical error.

- The three forces of good theories: empirical error, complexity, number of data. Statistical learning theory links the three through the "VC-bounds" (Expected error $\leq$ empirical error + $\frac{\mathrm{C}omplexity}{\mathrm{n}umber\ of\ data}$)

- The important concepts of hypothesis and target spaces, and generalization, approximation, and estimation errors have been introduced. VC-dimension and complexity measures have been introduced.

- Structural risk minimization (SRM) as an approach to learning has been developed. This leads to a family of learning methods.

Tikhonov Regularization and SRM

In Tikhonov regularization, we trade off the training error and the complexity of the hypothesis:

$$\widehat{f}_{H,S} = \arg\min_{f} \sum_{i=1}^{\ell} V(f(x_i), y_i) + \lambda \|f\|_K^2$$

What is V? What is $\|f\|_K^2$?

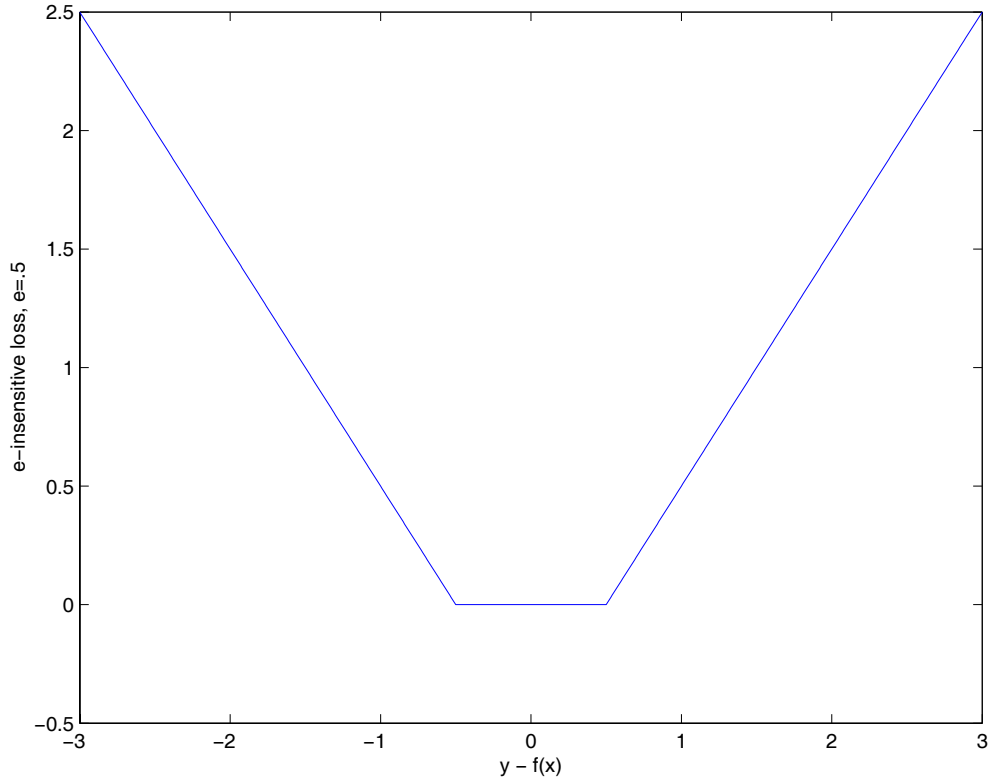
Three important techniques

$$
\begin{aligned}
V &= (y - f(\mathbf{x}))^2 \quad \text{for RN,} \\
V &= |y - f(\mathbf{x})|_\epsilon \quad \text{for SVMr, and} \\
V &= |1 - yf(\mathbf{x})|_+ \quad \text{for SVMc,}
\end{aligned}
$$

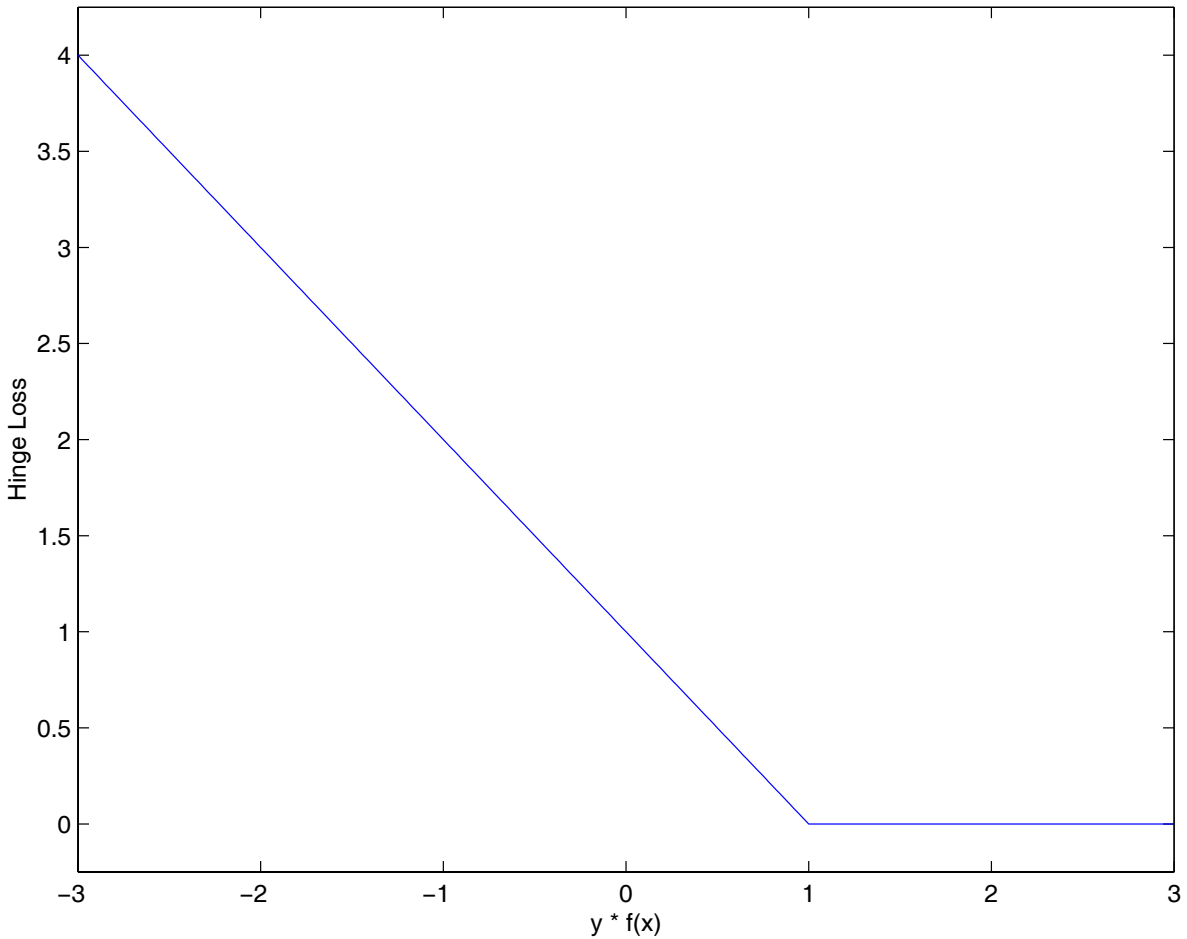
RN: Regularization Networks
SVMr: Support Vector Machines Regressions
SVMc: Support Vector Machines Classification

What about the functions used? Let's use $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ for the moment.

The ϵ-Insensitive Loss Function



The SVMc Loss Function



Substituting in the square loss

Using the square loss, our problem becomes

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{\ell} ((\mathbf{w}\mathbf{x}_i) - y_i)^2 + \lambda \|w\|^2$$

The Representer Theorem

Theorem. The solution to the Tikhonov regularization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} V(y_i, \mathbf{w}\mathbf{x}_i) + \lambda \|\mathbf{w}\|^2$$

can be written in the form

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{\ell} a_i (\mathbf{x} \cdot \mathbf{x}_i).$$

This theorem is exceedingly useful — it says that to solve the Tikhonov regularization problem, we need only find the best function of the form $f(\mathbf{x}) = \sum_{i=1}^{\ell} a_i (\mathbf{x} \cdot \mathbf{x}_i)$. Put differently, all we have to do is find the a_i.

Solving for α_i's

It turns out that for the L_2 loss function the α_i's are:

$$\alpha = (Q + I\lambda)^{-1}\mathbf{y}$$

(α is the vector of α_i's) and

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i(\mathbf{x}_i \cdot \mathbf{x}).$$

Where, Q is the square matrix defined by

$$Q_{ij} = (\mathbf{x}_i \cdot \mathbf{x}_j).$$

Least-Squares Regularized Regression

- The matrix $(Q + I\lambda)$ is guaranteed to be invertible (if $0 < \lambda < \infty$). As $\lambda \to 0$, the regularized least-squares solution goes to the standard least-squares solution which minimizes the empirical error. As $\lambda \to \infty$, the solution goes to $f(\mathbf{x}) = 0$.
- In pratice, we don't actually invert $(Q+I\lambda)$, but instead use an algorithm for solving linear systems.
- In order to use this approach, we need to compute and store the entire matrix Q. This makes it impractical for use with very large training sets.

Support Vector Machines Classification

With the L_1 hinge loss, the learning method becomes

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} |1 - y_i \mathbf{w} \mathbf{x}_i|_+ + \lambda ||\mathbf{w}||^2$$

Slack Variables (Optimization Theory)

We introduce slack variables ξ_i, to make the problem easier to work with:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \textstyle\sum_{i=1}^{\ell} \xi_i + \lambda ||\mathbf{w}||^2 & \\ \text{subject to :} \quad & y_i \mathbf{w} \mathbf{x}_i \geq 1 - \xi_i & i = 1, \ldots, \ell \\ & \xi_i \geq 0 & i = 1, \ldots, \ell \end{aligned}$$

The Dual Program

Simple optimization theory gives that the problem is equivalent to solving the following (dual) **Quadratic Optimization Problem with box constraints:**

$$\begin{aligned} \max_{\alpha \in \mathcal{R}^\ell} \quad & \sum_{i=1}^{\ell} \alpha_i - \alpha^T Q \alpha \\ \text{subject to :} \quad & 0 \leq \alpha_i \leq \tfrac{1}{\lambda} \qquad i = 1, \ldots, \ell \end{aligned}$$

Where, Q is the matrix defined by

$$Q_{ij} = y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j).$$

NOTE: if we look for $f(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$ (added a threshold b), then we also have the constraint: $\sum_{i=1}^{\ell} y_i \alpha_i = 0$

Support Vectors

The solution

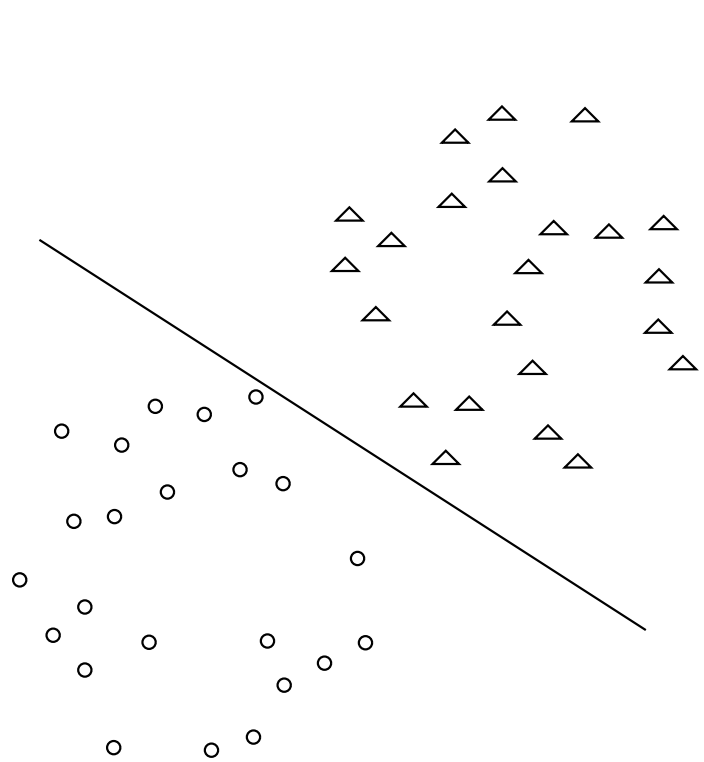
$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i (\mathbf{x}_i \cdot \mathbf{x}).$$

is sparse in the sense that a coefficient α_i will be non-zero only if $y_i f(\mathbf{x_i}) \leq 1$. In other words, points with no "training" error do not contribute to the function, and are not (so called) **support vectors**.

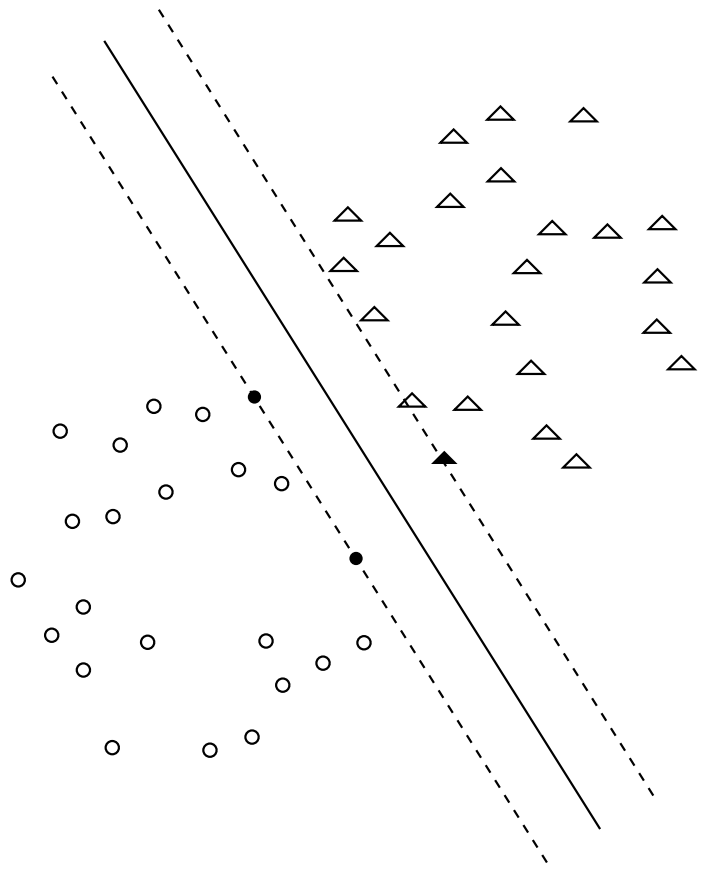
The Geometric Approach to SVMc

The "traditional" approach to developing the mathematics of SVM is to start with the concepts of *separating hyperplanes* and *margin*. Defining the margin as the distance from the hyperplane to the nearest example, the basic observation is that intuitively, we expect a hyperplane with larger margin to generalize better than one with smaller margin.

Large and Small Margin Hyperplanes



(a)



(b)

Classification With Hyperplanes

We denote our hyperplane by $\mathbf{w}$, and we will classify a new point $\mathbf{x}$ via the function

$$f(x) = \text{sign } (\mathbf{w} \cdot \mathbf{x}). \tag{1}$$

Given a separating hyperplane $\mathbf{w}$ we let $\mathbf{x}$ be a datapoint closest to $\mathbf{w}$, and we let $\mathbf{x}^{\mathbf{w}}$ be the unique point on $\mathbf{w}$ that is closest to x. Obviously, finding a maximum margin $\mathbf{w}$ is equivalent to maximizing $||\mathbf{x} - \mathbf{x}^{\mathbf{w}}||$ for all our training data ...

THIS LEADS TO THE SAME OPTIMIZATION PROBLEM FOR SVMc

Support Vector Machines For Regression

If we write a regularized regression problem with the ϵ-insensitive loss, and do some math, we end up with a quadratic programming, and a regression function of the form $f(x) = \sum_{i=1}^{\ell} \alpha_i (\mathbf{x} \cdot \mathbf{x_i})$ again.

The solution is sparse again in the sense that a coefficient α_i will be zero if $y_i - \epsilon < f(\mathbf{x_i}) < y_i + \epsilon$. In other words, points with no "training" error do not contribute to the function, and are not support vectors.

Historical Perspective

SVM for classification was defined in terms of maximizing the margin. With slack variables, this is no longer a strong justification. SVM for regression was an outgrowth of classification that preserved the idea of solving using a quadratic program and sparsity of the solution.

The SVM formulations have attracted a lot of attention in the literature. SVM classification, in particular, was often compared to other non-complexity-controlled methods for classification, and found to give very strong results. It is not however clear that SVM is better than RN, although this view is not prevalent in the literature.

Moreover, the perspective of viewing the choice of SVM vs RN as the choice of the more appropriate **computational** strategy is not prevalent in the literature.

Non-linear Functions

Nonlinear functions are also linear!

Consider the space of functions that is spanned by the "basis functions" $\phi_p(\mathbf{x})$,

$$\mathcal{H} = \{f \mid f(\mathbf{x}) = \sum_{p=1}^{N} c_p \phi_p(\mathbf{x})\}.$$

(where N can be infinite)

The idea is to map the data into a higher dimensional space called a *feature space* ($\Phi : \mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$) and to look for a linear function in this space.

Kernels and dot products

We can often define the inner product between two points mapped into feature space as

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

and the norm of a linear function in the feature space as:

$$\|f\|_K^2 = \sum_{n=1}^{N} c_n^2$$

K is called a **kernel** and the space of functions defined this way is called a **Reproducing Kernel Hilbert Space**. K basically defines a dot product in a high-dimensional feature space defined by the ϕ's.

Kernels: Examples

$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$)corresponds to polynomial functions of degree d)

$K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2)$ (Gaussian RBF)

$K(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^{2n+1}$ (Thin plate splines)

$K(\mathbf{x}, \mathbf{z}) = \tanh(\mathbf{x} \cdot \mathbf{z} - \theta)$ (only for some values of θ - Multi Layer Perceptron)

RKHS can be very rich

Depending on the coise of the kernel (or feature functions ϕ) an RKHS becomes dense in L_2.
This means is that these hypothesis spaces are "big", since they can approximate arbitrarily well a very rich class of functions (namely all L_2).

Moreover, it is possible to create "controlled" hypothesis spaces where we control the size of this space by using the RKHS norm as a "knob"

$$\|f\|_K^2 \leq A.$$

Which justifies RN and SVM within Statistical Learning Theory.

Why Kernels Work

We can use the kernel "trick" because for all the methods we developed so far, the final function is of the form $f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i (\mathbf{x} \cdot \mathbf{x}_i)$, that is, to evaluate it we only need to consider dot products between the example data and a new point. Moreover, to find the α_i we need to have *only* the dot products between our data.

With kernels our function is always of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

General Learning Methods

In general you can:

- Define your favorite loss function for the problem in hand
- Start by using linear functions (i.e. $f(x) = w \cdot x$) and formulate an optimization problem that is of the form of minimizing (over a set of parameters) empirical error $+ \lambda$ complexity
- Then simply use kernels in order to get non-linear functions

Key Practical Issues

This way we can get "advanced" (i.e. as non-linear and as complex models are we like) learning methods **as long as**:

1. It is computationally efficient to solve the optimization problem

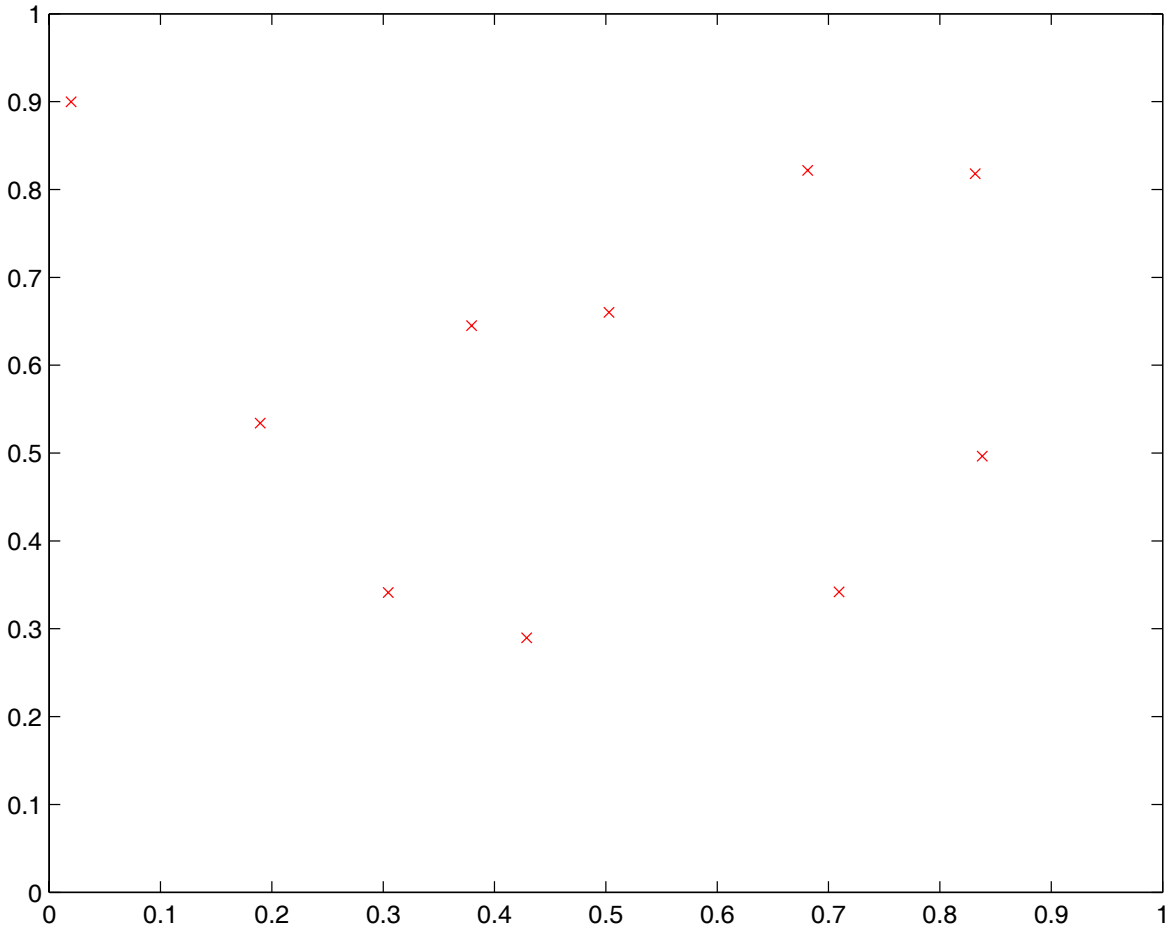
2. The methods are "kernelizable": this is the case ONLY if solving the optimization problems requires only dot-product calculations

3. Complexity of the solution is well-controlled: can lead from very simple to very complex functions.

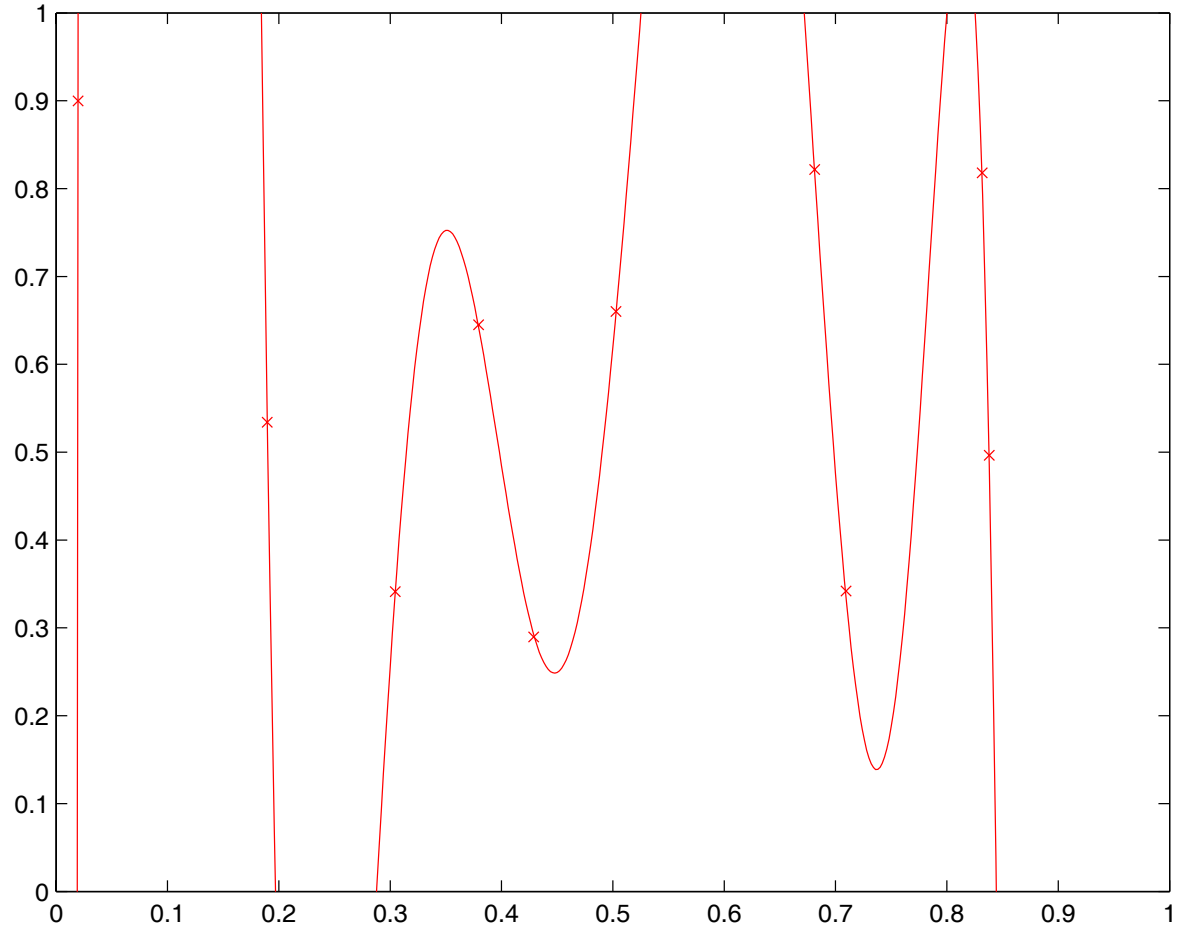
Another "theory of theories": Stability and Regularization

The goal is to have an algorithm for fitting the data that **is stable with respect to changes to the data** (i.e. how much will our solution change if we perturb the data a little or if we remove one training point?). (This is also one of the basic ideas behind regularization theory developed well before V. Vapnik did his PhD in Moscow).

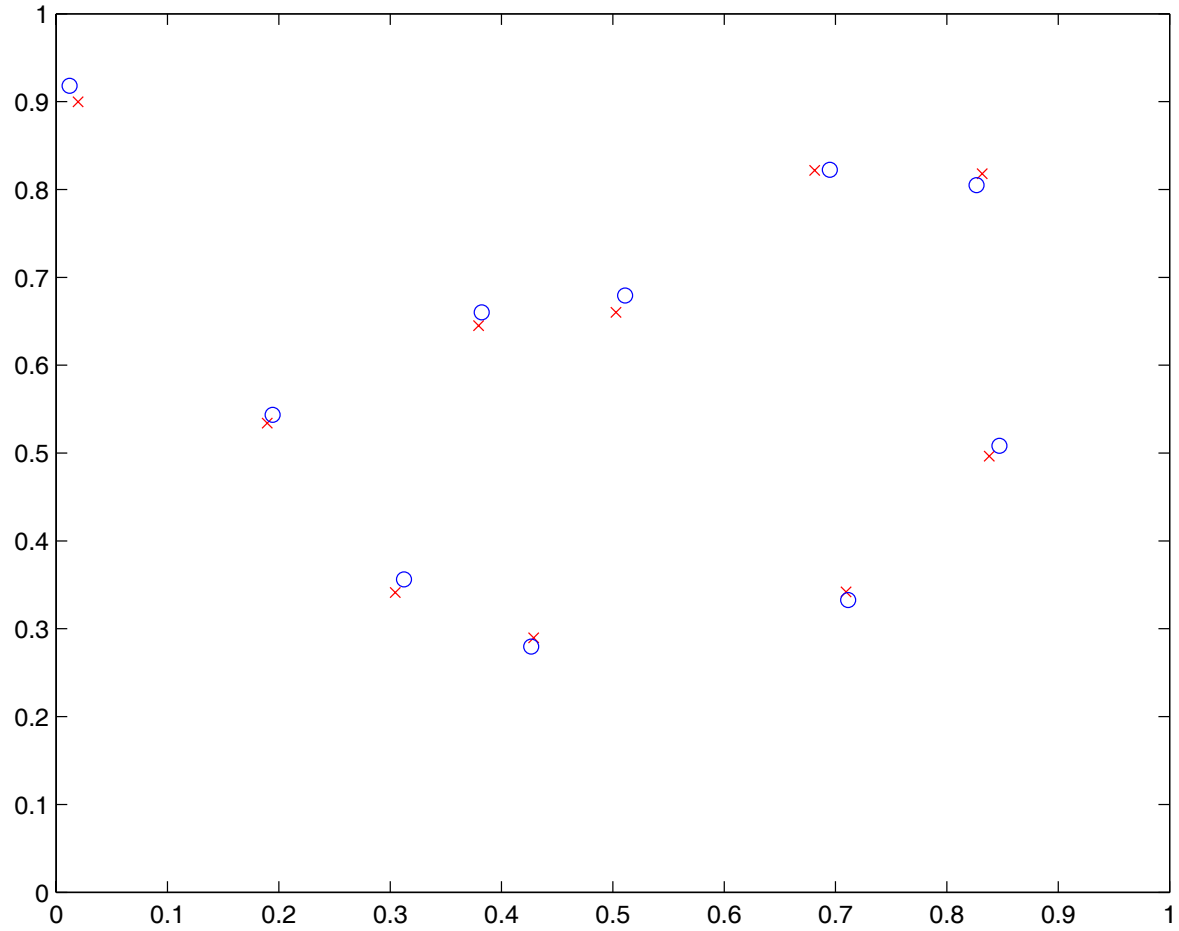
Given 10 samples...



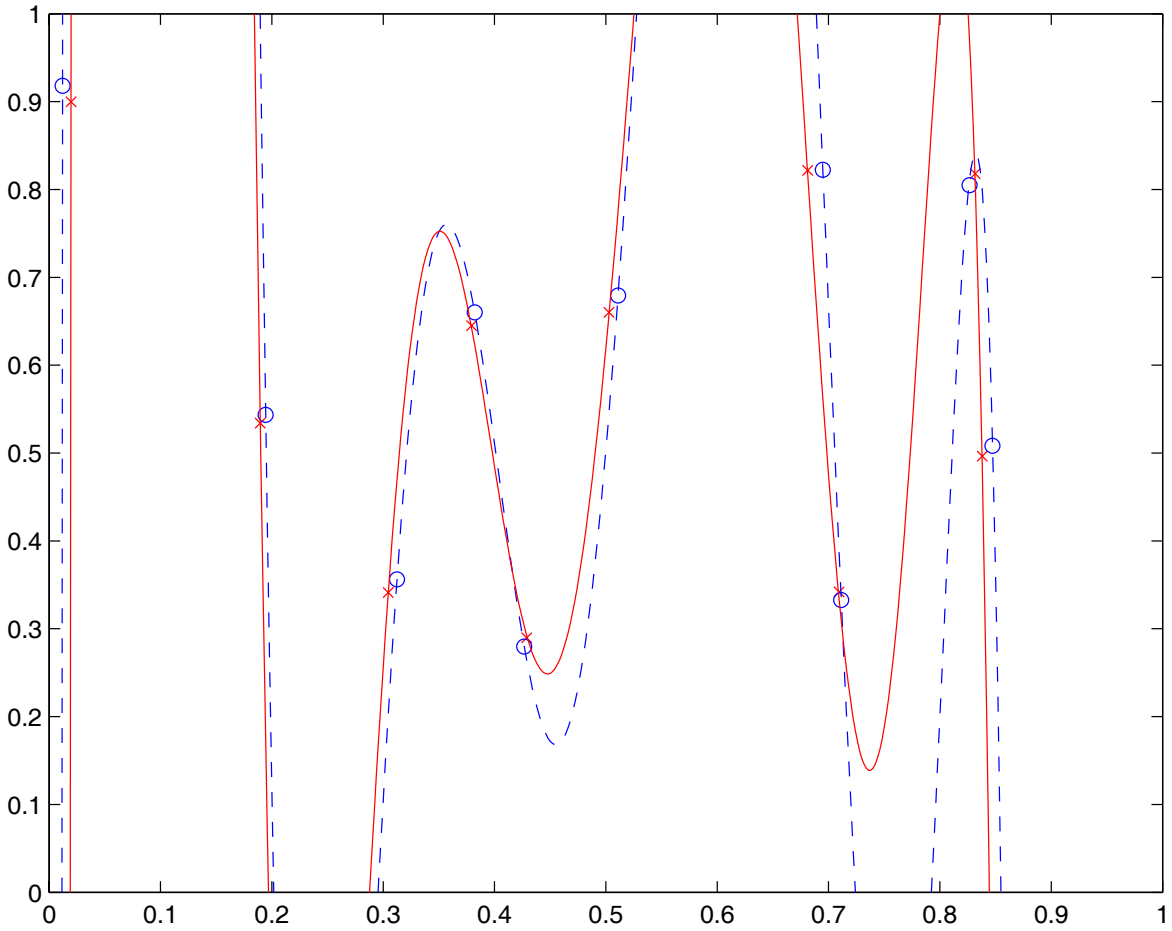
...we can find the smoothest interpolating polynomial.



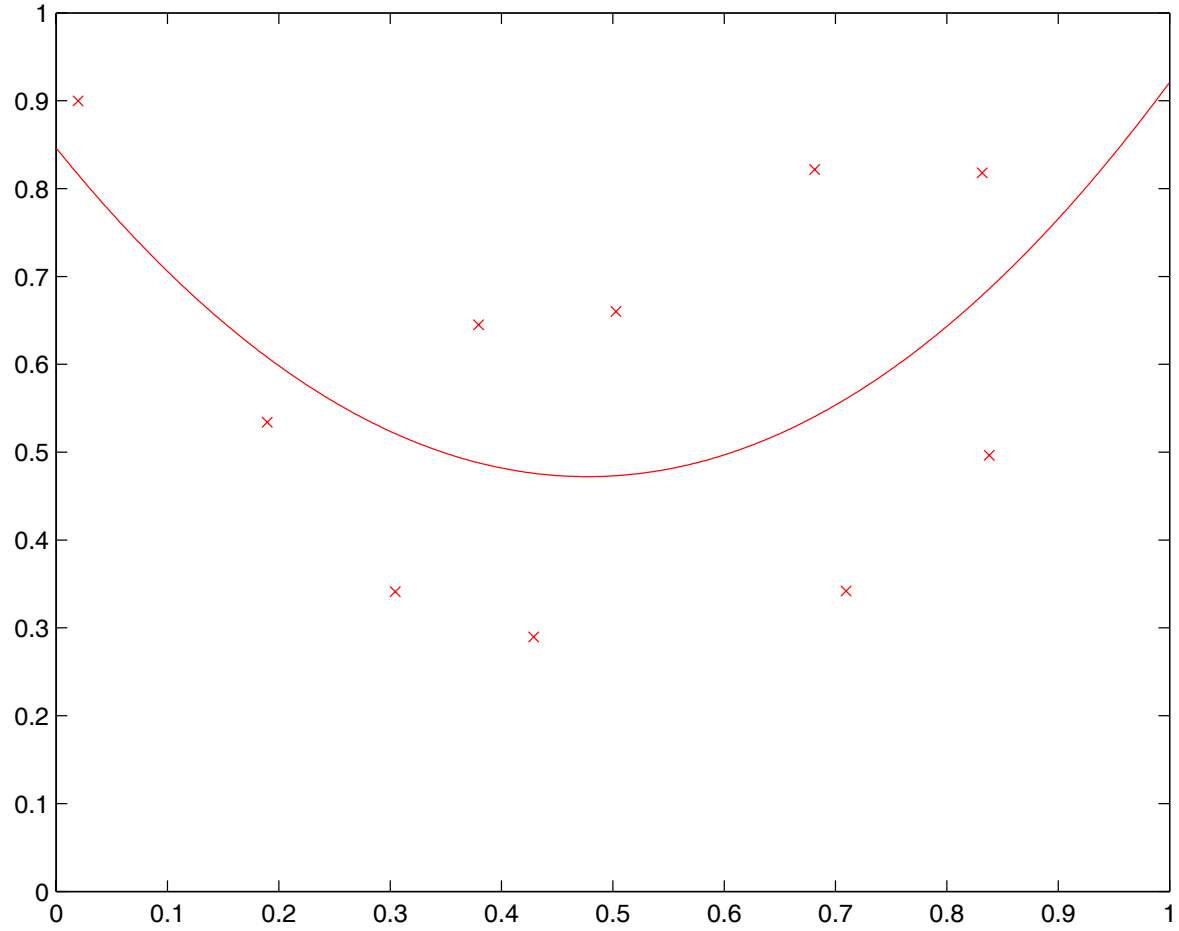
But if we perturb the points slightly...



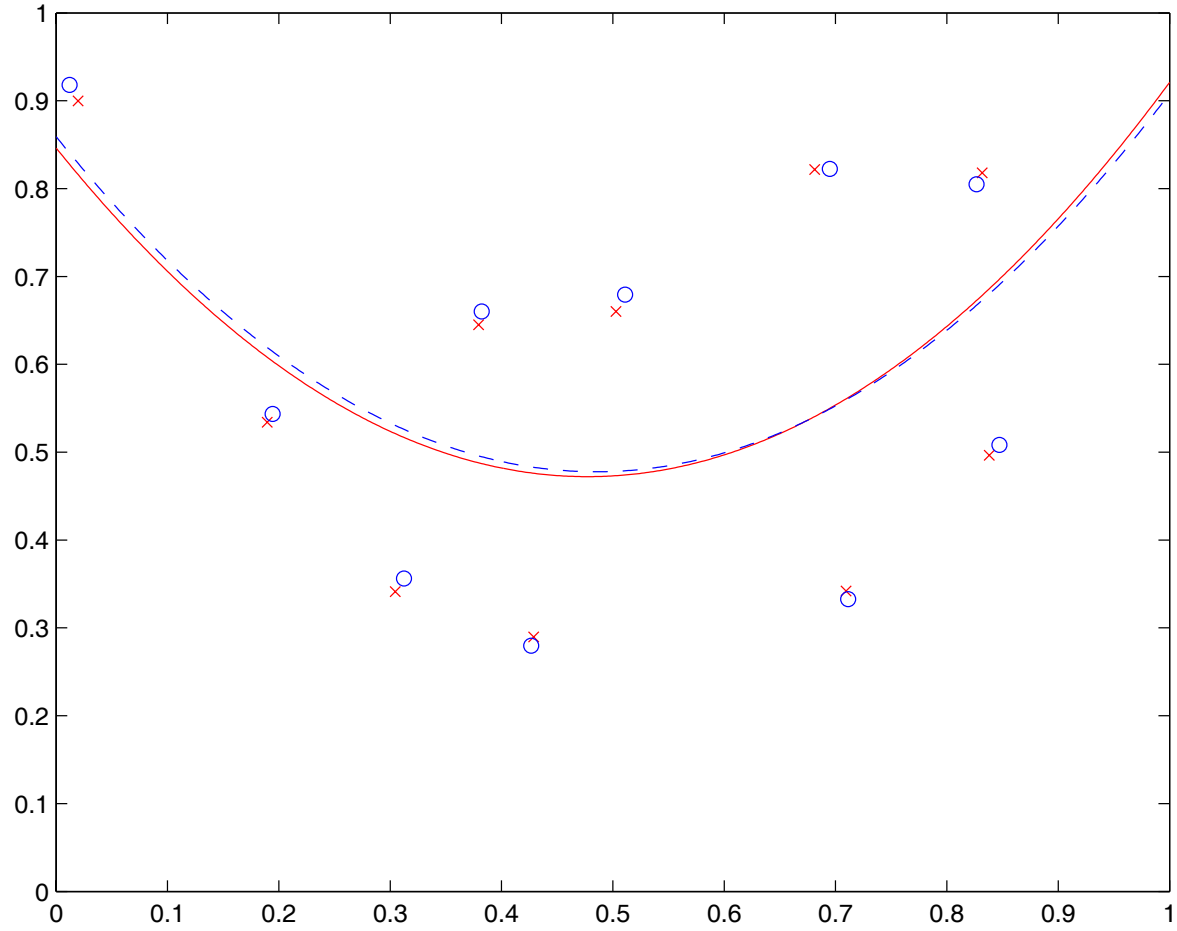
...the solution changes a lot.



If we restrict ourselves to degree two polynomials...



...the solution varies only a small amount under a small perturbation.



Stability and Generalization Performance

Let β be the stability of a learning algorithm (i.e. an upper bound on how much the solution can change if a training point changes). There exist bounds of the form:

With probability $1 - \delta$

$$R(f) \leq \hat{R}(f) + \beta + \ell\beta\sqrt{\frac{2\ln(2/\delta)}{\ell}}$$

so if $\beta = \frac{k}{\ell}$ for some constant k we get bounds of the form:

$$R(f) \leq \hat{R}(f) + \frac{k}{\ell} + k\sqrt{\frac{2\ln(2/\delta)}{\ell}}$$

See (Bousquet and Elisseeff, 2002 JMLR) for more information.

Applications

SVM and Kernel Machines with complexity control are very appropriate for *very large dimensional datasets* **(NO CURSE OF DIMENSIONALITY)** with very few example data.

- Object detection and recognintion in images (see demo)
- Bioinformatics
- Internet and business applications

Bioinformatics Problems

Types of data: DNA data, gene expression data, protein data, etc
Types of tasks: diagnostics, gene function prediction, protein folding, drug discovery, etc

Example:
Predict the functional class of genes given data of gene expression profiles (genes make proteins by producing RNA...). This is a very high dimensional classification problem with very few data. SVM have been shown to be helpful.

Modeling Preferences

For this problem the data are of the following form:

For $i = 1, \ldots n$ we are given 2 "products" (vectors) $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}$ and we are told that (without loss of generality) product $\mathbf{x}_{i1}$ is preferred.

Problem: Develop a utility function $f(\mathbf{x})$ such that when given a new set of products $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k\}$ function f takes the largest value for the preferred product.

The term "utility function" is a standard term in economics.

General Learning Methods (As before)

In general you can:

- Define your favorite loss function for the problem in hand
- Start by using linear functions (i.e. $f(x) = w \cdot x$) and formulate an optimization problem that is of the form of minimizing (over a set of parameters) empirical error $+ \lambda$ complexity
- Then simply use kernels in order to get non-linear functions

Key Practical Issues (As before)

This way we can get "advanced" (i.e. as non-linear and as complex models are we like) learning methods **as long as**:

1. It is computationally efficient to solve the optimization problem

2. The methods are "kernelizable": this is the case ONLY if solving the optimization problems requires only dot-product calculations

3. Complexity of the solution is well-controlled: can lead from very simple to very complex functions.

Modeling Preferences

Consider linear utility functions $f(\mathbf{x}) = \mathbf{wx}$. Simple empirical risk minimization leads to the method:

$$\begin{array}{ll} \min\limits_{\mathbf{w}} & \sum_{i=1}^{n} \xi_i \\ \text{s.t.} & \mathbf{wx}_{i1} \geq \mathbf{wx}_{i2} - \xi_i \quad for\ i \in \{1, 2, \ldots n\} \\ & \xi_i \geq 0 \end{array}$$

Modeling Preferences: SRM approach

$$\begin{aligned} \min_{\mathbf{w}} \quad & \textstyle\sum_{i=1}^{n} \xi_i + \lambda \|w\|^2 \\ \text{s.t.} \quad & \mathbf{w}(\mathbf{x}_{i1} - \mathbf{x}_{i2}) \geq 1 - \xi_i \quad for\ i \in \{1, 2, \ldots n\} \\ & \xi_i \geq 0 \end{aligned}$$

(Does this look like SVM classification of vector differences?)

Modeling Preferences: the kernel trick

$$\begin{aligned} \min_{\mathbf{f}} \quad & \textstyle\sum_{i=1}^{n} \xi_i + \lambda \|f\|_K^2 \\ \text{s.t.} \quad & f(\mathbf{x}_{i1}) - f(\mathbf{x}_{i2}) \geq 1 - \xi_i \quad for\ i \in \{1, 2, \ldots n\} \\ & \xi_i \geq 0 \end{aligned}$$

with dual form:

$$\begin{aligned} \max_{\alpha \in \mathcal{R}^\ell} \quad & \textstyle\sum_{i=1}^{\ell} \alpha_i - \alpha^T Q \alpha \\ \text{subject to :} \quad & 0 \leq \alpha_i \leq \tfrac{1}{\lambda} \qquad i = 1, \ldots, \ell \end{aligned}$$

Where, Q is the matrix defined using again the kernel $K(x_i, x_j)$ (is it directly this kernel? - "homework")

Other Problems

1. Anomaly detection can be treated similarly (kind of)

2. Optimal control problems

3. Density estimation problems

4. ??????

A Reality Check

- 95% of the time for Data Mining is spent on preparing the data. Less than 5% of the time is spent on using SVM or whatever other method.

- Even if we do data mining well, most of the time at most 3 people in the world know it (the author and the 2 reviewers) - and even fewer care about it.

- Yet lately many data miners (machine learners, AIers, etc) thought of themselves as gold-miners - but guess what?

Are there opportunities? What are the key issues for business success?

The Data World

- The average novel is 10^{-6} TB (1 TerraByte = 1000 GB). The US library of Congress has 20 TB.
- The "visible" WWW is around 30 TB.
- In 1999 there were 1 trillion emails sent in the world.
- It took 10 years for a big bank in the 80's to reach 10 TB of data. Only 9 months in the 90's online.
- The average online company is expected to manage 120 TB of customer data by 2003.

$\Rightarrow$ There is so much damn data

The 3 Big Areas (currently)

1. Customer Relationship Management (i.e. e.piphany.com, SAS, etc)

2. Financial markets (many)

3. Knowledge Management (i.e. Autonomy.com)

Data Mining and Business Value Chain

Supply Chain Management $-->$ Internal comnunications $-->$ Customer Relationship management

There is data to be analyzed (to cut costs, increase revenues, create new markets) at all points in these "business chains", especially because of the **virtual value chain** created due to IT (including the Internet).

Example: Openratings.com and supply chain management.

Other areas

- Bio/medical
- Fraud detection
- Risk management
- Preventive maintenance
- Industry specific opportunities

The market for business intelligence TOOLS is growing at an annual rate of 50%, and is estimated to about 150 billion Euros (more than the GNP of Greece). Analytics is a small part of it (but analytics based applications are key).

Factors of Success: warnings

1. People (i.e. ease of use)
2. People (i.e. behavioral constraints)
3. People (i.e. incentives to use)
4. Strategy (Yes it helps, but does it help with what **we want** to achieve?)
5. Bottom line effects (reduce costs or increase revenues)
6. The right data, legal constraints (i.e. privacy), etc

Conclusions

We talked about two worlds:

- Philosophy, Mathematics, Engineering (**research** applications)

- Applications adopted by companies, and used in every day life

In the information age, data mining is about issues in both worlds.

References

- T. Evgeniou, M. Pontil, T. Poggio. Regularization Networks and Support Vector Machines. Advances in Computational Mathematics, 2000.
- F. Cucker and S. Smale. On The Mathematical Foundations of Learning. Bulletin of the American Mathematical Society, 2002.
- www.kernel-machines.org
- www.kdnuggets.com
- www.sims.berkeley.edu/resources/infoecon/

Special thanks to S. Mukherjee, M. Pontil, T. Poggio, R. Rifkin, MIT CBCL