The Learning Problem and Regularization

Tomaso Poggio

9.520 Class 02

September 2015

Computational Learning

Statistical Learning Theory

Learning is viewed as a generalization/inference problem from usually **small** sets of **high dimensional**, noisy data.

Today's class is one of the most difficult – because it is abstract. Reasons for it:

- Science of Learning
- Big picture and flavor
- Mathcamp is next
- This classroom is not large enough.

Learning Tasks and Models

There are in principle several "learning problems". The one which is most crisply defined is supervised learning. If the conjecture about Implicit Supervised Examples were correct, then *supervised learning* – together with reinforcement learning – would be the most important building block for the whole of biological learning.

- Supervised
- Semisupervised
- Unsupervised
- Online
- Transductive
- Active
- Variable Selection
- Reinforcement
- .....

In addition one can consider the data to be created in a deterministic, or stochastic or even adversarial way.

Where to Start?

Statistical and Supervised Learning

- Statistical Models are essentially to deal with noise sampling and other sources of uncertainty.
- Supervised Learning is the best understood type of learning problems and may be a building block for most of the others.

Regularization

- Regularization provides a rigorous framework to solve learning problems and to design learning algorithms.
- **In the course we will present a set of ideas and tools which are at the core of several developments in supervised learning and beyond it.**
- We will see the close connection during the last classes between kernel machines and deep networks.

Where to Start?

Statistical and Supervised Learning

- Statistical Models are essentially to deal with noise sampling and other sources of uncertainty.
- Supervised Learning is the best understood type of learning problems and may be a building block for most of the others.

Regularization

- Regularization provides a rigorous framework to solve learning problems and to design learning algorithms.
- **In the course we will present a set of ideas and tools which are at the core of several developments in supervised learning and beyond it.**
- We will see the close connection during the last classes between kernel machines and deep networks.

Remarks on Foundations of Learning Theory

- This class establish our program for the first 10 classes:
  - Main goal of learning is generalization and predictivity *not* explanation
  - Which algorithms to guarantee ensure generalization?
  - We derive "equivalence" of generalization and stability/well-posedness
  - Since it is known that regularization techniques guarantee well-posedness we will use them to guarantee also generalization
  - Notice that they usually result in *computationally "nice"* and *well-posed* constrained optimization problems

Plan

- Part I: Basic Concepts and Notation
- Part II: Foundational Results
- Part III: Algorithms

Learning Problem at a Glance

Given a training set of *input-output* pairs

$$S_n = (x_1, y_1), \ldots, (x_n, y_n)$$

find f_S such that

$$f_S(x) \sim y.$$

e.g. the x's are vectors and the y's discrete labels in classification and real values in regression.

Learning is Inference

For the above problem to make sense we need to assume input and output to be related!

Statistical and Supervised Learning

- Each input-output pairs is a sample from a fixed but unknown distribution $\mu(x, y)$.
- Under some condition we can associate to $\mu(z)$ the probability

$$p(x, y) = p(y|x)p(x).$$

- the training set S_n is a set of **identically and independently distributed** samples drawn from $\mu(z)$.
- It is crucial to note that we view $p(x, y)$ as **fixed** but **unknown**.

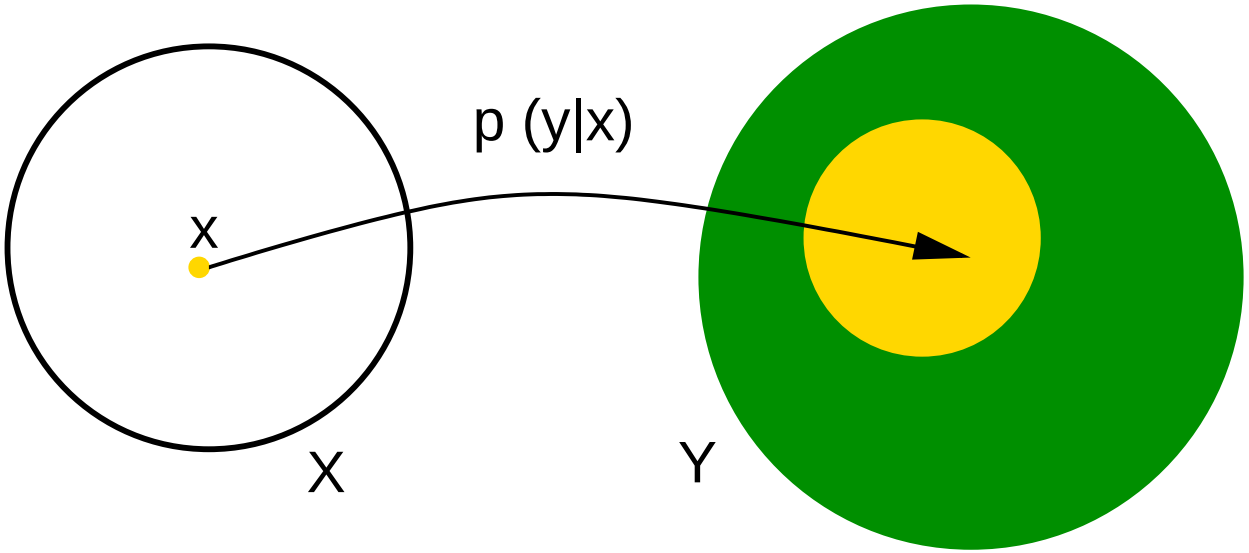
Learning is Inference

For the above problem to make sense we need to assume input and output to be related!

Statistical and Supervised Learning

- Each input-output pairs is a sample from a fixed but unknown distribution $\mu(x, y)$.
- Under some condition we can associate to $\mu(z)$ the probability
$$p(x, y) = p(y|x)p(x).$$
- the training set S_n is a set of **identically and independently distributed** samples drawn from $\mu(z)$.
- It is crucial to note that we view $p(x, y)$ as **fixed** but **unknown**.

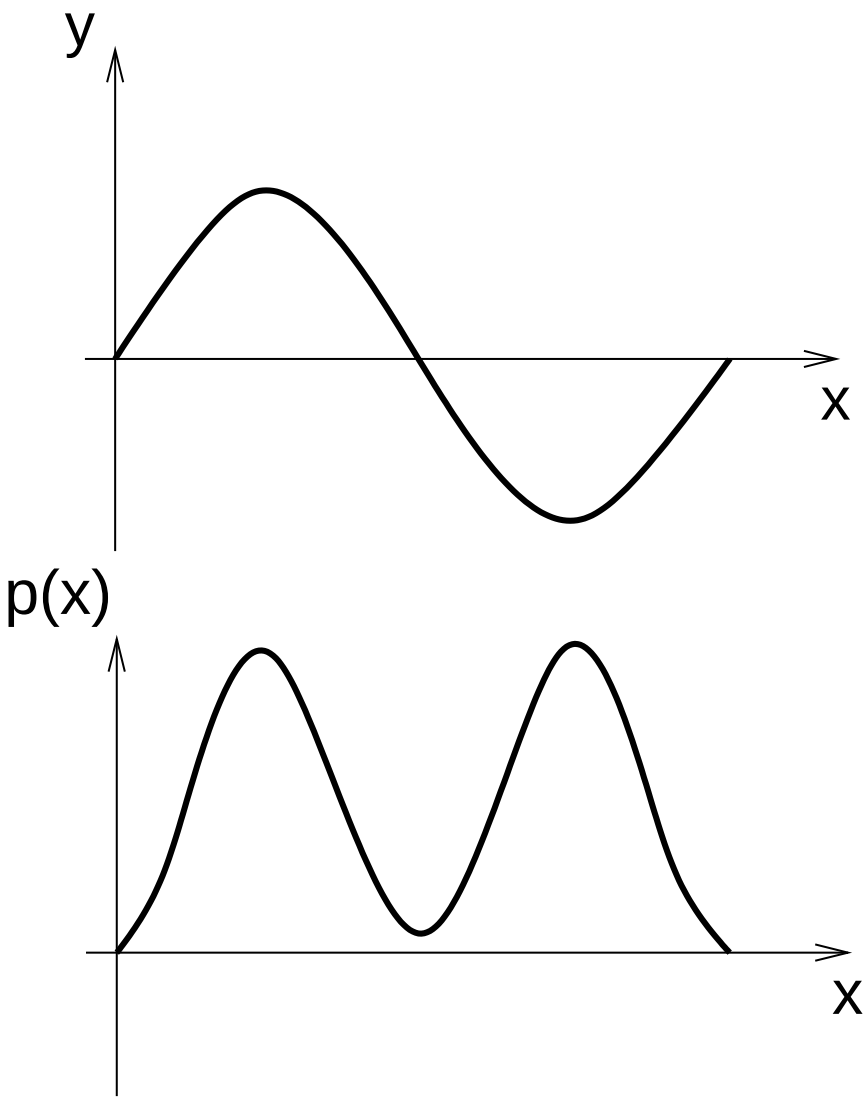
Why Probabilities



the same x can generate different y (according to $p(y|x)$):

- the underlying process is deterministic, but there is **noise** in the measurement of y;
- the underlying process is **not deterministic**;
- the underlying process is deterministic, but only **incomplete** information is available.

Sampling

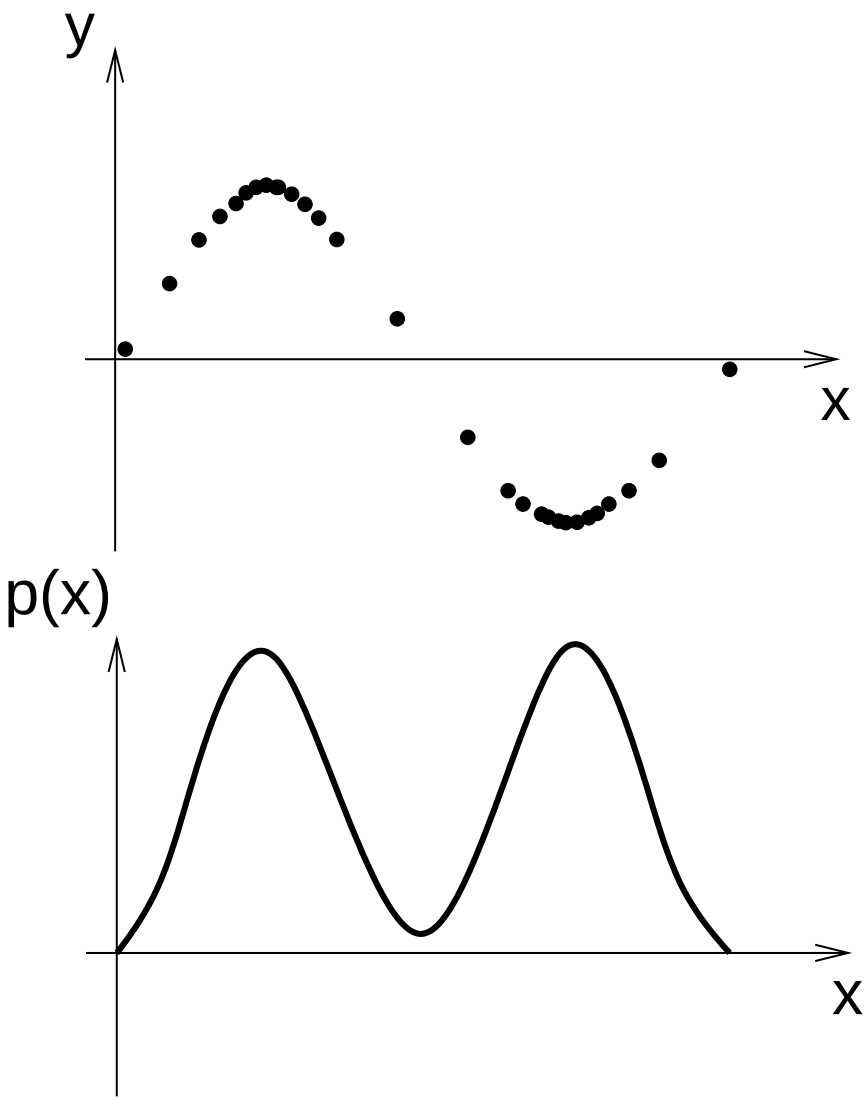


even in a noise free case we have to deal with sampling

the marginal $p(x)$ distribution might model

- errors in the location of the input points;
- discretization error for a given grid;
- presence or absence of certain input instances

Sampling

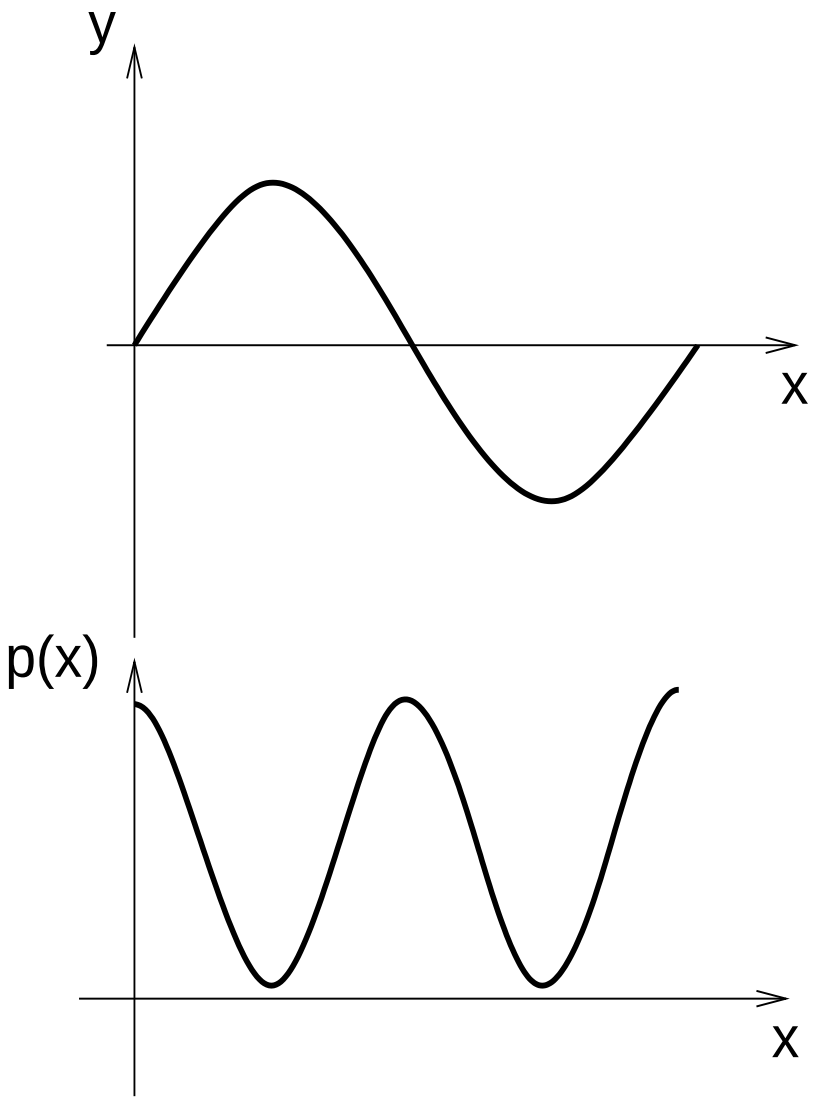


even in a noise free case we have to deal with sampling

the marginal $p(x)$ distribution might model

- errors in the location of the input points;
- discretization error for a given grid;
- presence or absence of certain input instances

Sampling

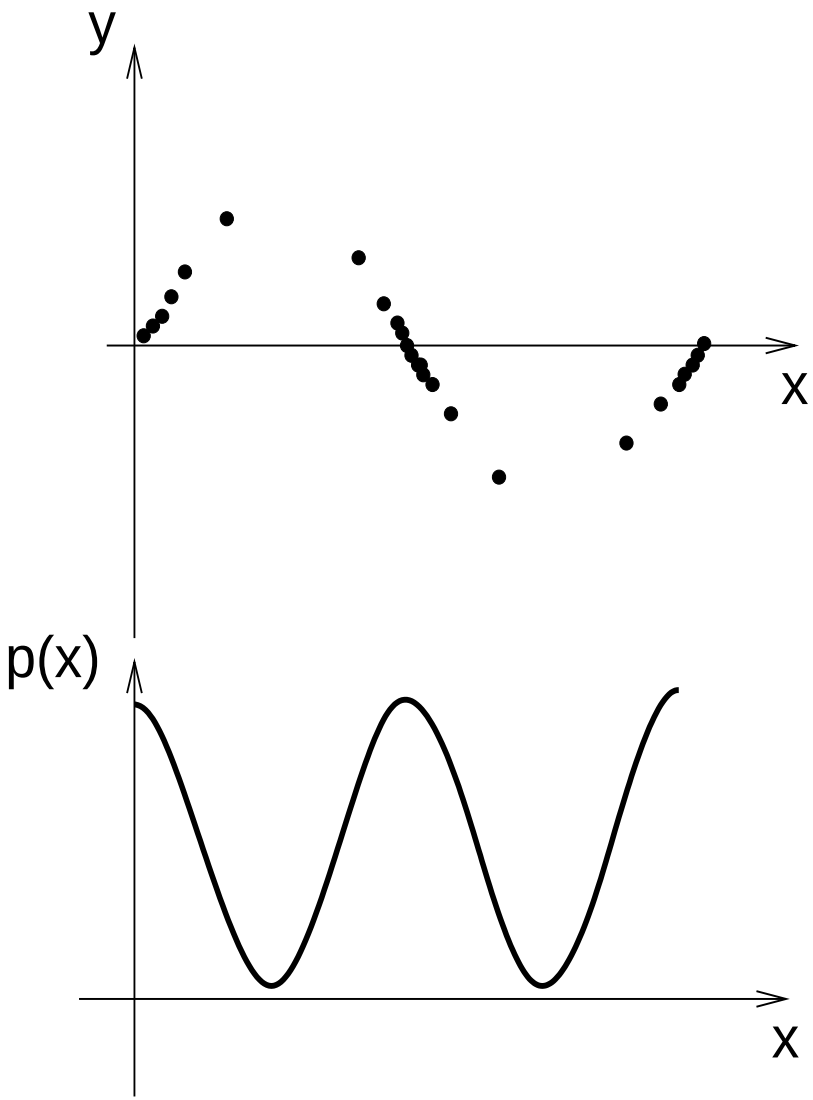


even in a noise free case we have to deal with sampling

the marginal $p(x)$ distribution might model

- errors in the location of the input points;
- discretization error for a given grid;
- presence or absence of certain input instances

Sampling



even in a noise free case we have to deal with sampling

the marginal $p(x)$ distribution might model

- errors in the location of the input points;
- discretization error for a given grid;
- presence or absence of certain input instances

Learning Problem at a Glance

Given a training set of *input-output* pairs

$$S_n = (x_1, y_1), \ldots, (x_n, y_n)$$

find f_S such that

$$f_S(x) \sim y.$$

e.g. the x's are vectors and the y's discrete labels in classification and real values in regression.

Learning, Generalization/Prediction

Predictivity or Generalization

Given the data, the goal is to learn how to make decisions/predictions about future data / data not belonging to the training set. **Generalization** is the key requirement emphasized in Learning Theory: generalization is a masure of predictivity. This emphasis makes it different from Bayesian or traditional statistics (especially explanatory statistics).

The problem is often: **Avoid overfitting!!**

Loss functions

In order to define generalization we need to define and measure errors.

Loss function

A loss function $V : \mathbf{R} \times Y$ determines the price $V(f(x), y)$ we pay, predicting $f(x)$ when in fact the true output is y.

Loss functions

In order to define generalization we need to define and measure errors.

Loss function

A loss function $V : \mathbf{R} \times Y$ determines the price $V(f(x), y)$ we pay, predicting $f(x)$ when in fact the true output is y.

Loss functions for regression

- The most common is the **square loss** or L_2 **loss**

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

- **Absolute value** or L_1 **loss**:

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

- Vapnik's ϵ-**insensitive loss**:

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

Loss functions for (binary) classification

- The most intuitive one: $0-1$**-loss**:

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

(θ is the step function)

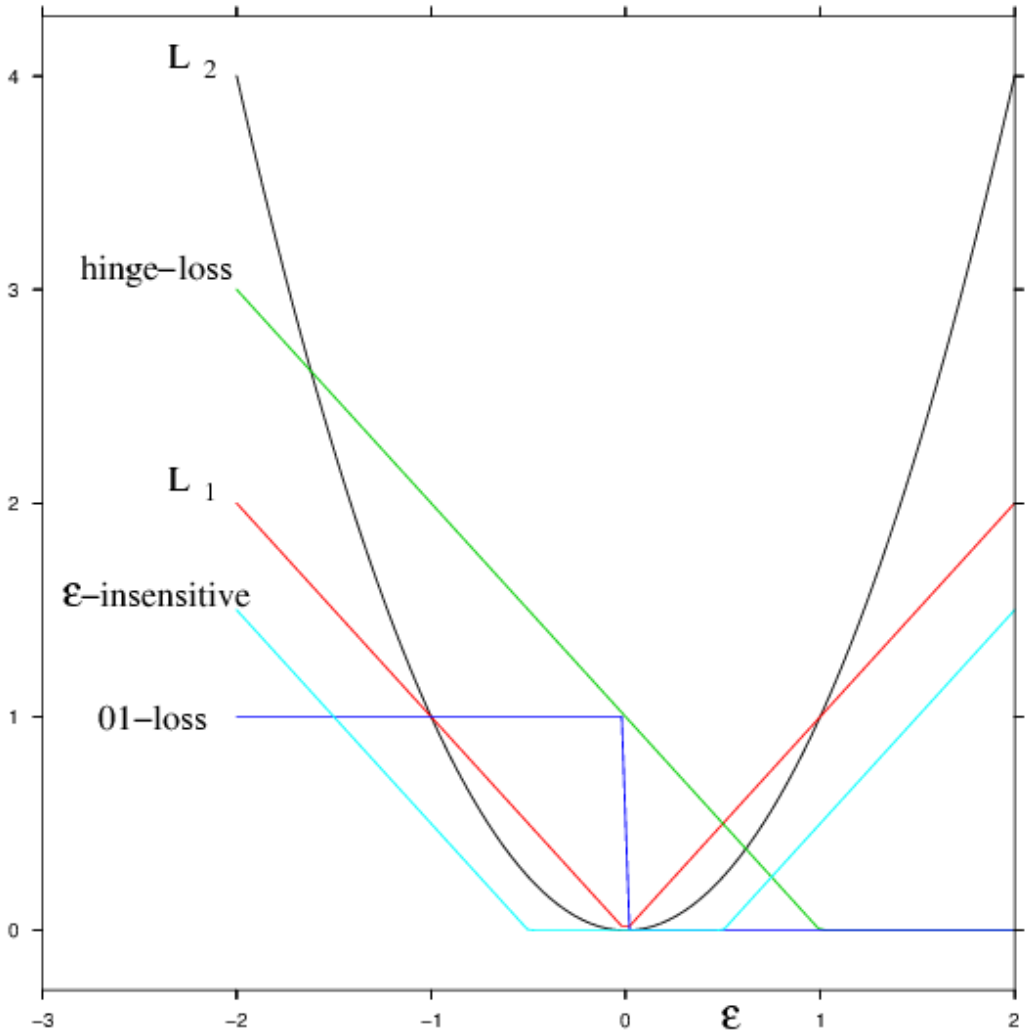
- The more tractable **hinge loss**:

$$V(f(x), y) = (1 - yf(x))_+$$

- And again the **square loss** or L_2 **loss**

$$V(f(x), y) = (1 - yf(x))^2$$

Loss functions



Expected Risk

A good function – we will speak of function or *hypothesis* – should incur in only *a few* errors. We need a way to quantify this idea.

Expected Risk

The quantity

$$I[f] = \int_{X \times Y} V(f(x), y) p(x, y) dxdy.$$

is called the expected error and measures the loss averaged over the unknown distribution.

A good function should have small expected risk.

Expected Risk

A good function – we will speak of function or *hypothesis* – should incur in only *a few* errors. We need a way to quantify this idea.

Expected Risk

The quantity

$$I[f] = \int_{X \times Y} V(f(x), y) p(x, y) dxdy.$$

is called the expected error and measures the loss averaged over the unknown distribution.

A good function should have small expected risk.

Learning Algorithms and Generalization

A learning algorithm can be seen as a map

$$S_n \to f_n$$

from the training set to the a set of candidate functions.

Basic definitions

- $p(x, y)$ probability distribution,
- S_n training set,
- $V(f(x), y)$ loss function,
- $I_n[f] = \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i)$, empirical risk,
- $I[f] = \int_{X \times Y} V(f(x), y) p(x, y) dxdy$, expected risk.

Reminder

Convergence in probability

Let $\{X_n\}$ be a sequence of bounded random variables. Then

$$\lim_{n\to\infty} X_n = X \quad \text{in probability}$$

if

$$\forall \epsilon > 0 \quad \lim_{n\to\infty} \mathbb{P}\{|X_n - X| \geq \epsilon\} = 0$$

Convergence in Expectation

Let $\{X_n\}$ be a sequence of bounded random variables. Then

$$\lim_{n\to\infty} X_n = X \quad \text{in expectation}$$

if

$$\lim_{n\to\infty} \mathbb{E}(|X_n - X|) = 0$$

. Convergence in the mean implies convergence in probability.

Reminder

Convergence in probability

Let $\{X_n\}$ be a sequence of bounded random variables. Then

$$\lim_{n\to\infty} X_n = X \quad \text{in probability}$$

if

$$\forall \epsilon > 0 \quad \lim_{n\to\infty} \mathbb{P}\{|X_n - X| \geq \epsilon\} = 0$$

Convergence in Expectation

Let $\{X_n\}$ be a sequence of bounded random variables. Then

$$\lim_{n\to\infty} X_n = X \quad \text{in expectation}$$

if

$$\lim_{n\to\infty} \mathbb{E}(|X_n - X|) = 0$$

. Convergence in the mean implies convergence in probability.

Consistency and Universal Consistency

A requirement considered of basic importance in classical statistics is for the algorithm to get better as we get more data (in the context of machine learning consistency is less immediately critical than *generalization*)...

Consistency

We say that an algorithm is consistent if

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Universal Consistency

We say that an algorithm is universally consistent if for all probability p,

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Consistency and Universal Consistency

A requirement considered of basic importance in classical statistics is for the algorithm to get better as we get more data (in the context of machine learning consistency is less immediately critical than *generalization*)...

Consistency

We say that an algorithm is consistent if

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Universal Consistency

We say that an algorithm is universally consistent if for all probability p,

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Consistency and Universal Consistency

A requirement considered of basic importance in classical statistics is for the algorithm to get better as we get more data (in the context of machine learning consistency is less immediately critical than *generalization*)...

Consistency

We say that an algorithm is consistent if

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Universal Consistency

We say that an algorithm is universally consistent if for all probability p,

$$\forall \epsilon > 0 \quad \lim_{n \to \infty} \mathbb{P}\{I[f_n] - I[f_*] \geq \epsilon\} = 0$$

Sample Complexity and Learning Rates

The above requirements are asymptotic.

Error Rates

A more practical question is, how fast does the error decay?
This can be expressed as

$$\mathbb{P}\{I[f_n] - I[f_*]\} \leq \epsilon(n, \delta)\} \geq 1 - \delta.$$

Sample Complexity

Or equivalently, 'how many point do we need to achieve an error ϵ with a prescribed probability δ?'
This can expressed as

$$\mathbb{P}\{I[f_n] - I[f_*] \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Sample Complexity and Learning Rates

The above requirements are asymptotic.

Error Rates

A more practical question is, how fast does the error decay?
This can be expressed as

$$\mathbb{P}\{I[f_n] - I[f_*]\} \leq \epsilon(n, \delta)\} \geq 1 - \delta.$$

Sample Complexity

Or equivalently, 'how many point do we need to achieve an error ϵ with a prescribed probability δ?'
This can expressed as

$$\mathbb{P}\{I[f_n] - I[f_*] \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Sample Complexity and Learning Rates

The above requirements are asymptotic.

Error Rates

A more practical question is, how fast does the error decay?
This can be expressed as

$$\mathbb{P}\{I[f_n] - I[f_*]\} \leq \epsilon(n, \delta)\} \geq 1 - \delta.$$

Sample Complexity

Or equivalently, 'how many point do we need to achieve an error ϵ with a prescribed probability δ?'
This can expressed as

$$\mathbb{P}\{I[f_n] - I[f_*] \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Sample Complexity and Learning Rates

The above requirements are asymptotic.

Error Rates

A more practical question is, how fast does the error decay? This can be expressed as

$$\mathbb{P}\{I[f_n] - I[f_*]\} \leq \epsilon(n, \delta)\} \geq 1 - \delta.$$

Sample Complexity

Or equivalently, 'how many point do we need to achieve an error ϵ with a prescribed probability δ?'

This can expressed as

$$\mathbb{P}\{I[f_n] - I[f_*] \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Sample Complexity and Learning Rates

The above requirements are asymptotic.

Error Rates

A more practical question is, how fast does the error decay?
This can be expressed as

$$\mathbb{P}\{I[f_n] - I[f_*]\} \leq \epsilon(n, \delta)\} \geq 1 - \delta.$$

Sample Complexity

Or equivalently, 'how many point do we need to achieve an error ϵ with a prescribed probability δ?'
This can expressed as

$$\mathbb{P}\{I[f_n] - I[f_*] \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Empirical risk and Generalization

How do we design learning algorithms that work? One of the most natural ideas is ERM...

Empirical Risk

The empirical risk is a natural proxy (how good?) for the expected risk

$$I_n[f] = \frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i).$$

Generalization Error

How good a proxy is captured by the generalization error,

$$\mathbb{P}\{|I[f_n] - I_n[f_n]| \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Empirical risk and Generalization

How do we design learning algorithms that work? One of the most natural ideas is ERM...

Empirical Risk

The empirical risk is a natural proxy (how good?) for the expected risk

$$I_n[f] = \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i).$$

Generalization Error

How good a proxy is captured by the generalization error,

$$\mathbb{P}\{|I[f_n] - I_n[f_n]| \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Empirical risk and Generalization

How do we design learning algorithms that work? One of the most natural ideas is ERM...

Empirical Risk

The empirical risk is a natural proxy (how good?) for the expected risk

$$I_n[f] = \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i).$$

Generalization Error

How good a proxy is captured by the generalization error,

$$\mathbb{P}\{|I[f_n] - I_n[f_n]| \leq \epsilon\} \geq 1 - \delta,$$

for $n = n(\epsilon, \delta)$.

Some (Theoretical and Practical) Questions

- How do we go from here to an actual class of algorithms?
- Is minimizing the empirical error – error on the data – a good idea?
- Under which conditions is the empirical error a good proxy for the expected error?

Some (Theoretical and Practical) Questions

- How do we go from here to an actual class of algorithms?
- Is minimizing the empirical error – error on the data – a good idea?
- Under which conditions is the empirical error a good proxy for the expected error?

Some (Theoretical and Practical) Questions

- How do we go from here to an actual class of algorithms?
- Is minimizing the empirical error – error on the data – a good idea?
- Under which conditions is the empirical error a good proxy for the expected error?

Plan

- Part I: Basic Concepts and Notation
- **Part II: Foundational Results**
- Part III: Algorithms

No Free Lunch Theorem Devroye et al.

Universal Consistency

Since classical statistics worries so much about consistency let us start here even if I do not think it is a practically important concept. Can we learn consistently any problem? Or equivalently do universally consistent algorithms exist?

YES! Neareast neighbors, Histogram rules, SVM with (so called) universal kernels...

No Free Lunch Theorem

Given a number of points (and a confidence), can we always achieve a prescribed error?

NO!

The last statement can be interpreted as follows: inference from finite samples can effectively performed if and only if the problem satisfies some a priori condition.

No Free Lunch Theorem Devroye et al.

Universal Consistency

Since classical statistics worries so much about consistency let us start here even if I do not think it is a practically important concept. Can we learn consistently any problem? Or equivalently do universally consistent algorithms exist?
YES! Neareast neighbors, Histogram rules, SVM with (so called) universal kernels...

No Free Lunch Theorem

Given a number of points (and a confidence), can we always achieve a prescribed error?
NO!

The last statement can be interpreted as follows: inference from finite samples can effectively performed if and only if the problem satisfies some a priori condition.

No Free Lunch Theorem Devroye et al.

Universal Consistency

Since classical statistics worries so much about consistency let us start here even if I do not think it is a practically important concept. Can we learn consistently any problem? Or equivalently do universally consistent algorithms exist?
YES! Neareast neighbors, Histogram rules, SVM with (so called) universal kernels...

No Free Lunch Theorem

Given a number of points (and a confidence), can we always achieve a prescribed error?
NO!

The last statement can be interpreted as follows: inference from finite samples can effectively performed if and only if the problem satisfies some a priori condition.

No Free Lunch Theorem Devroye et al.

Universal Consistency

Since classical statistics worries so much about consistency let us start here even if I do not think it is a practically important concept. Can we learn consistently any problem? Or equivalently do universally consistent algorithms exist?
YES! Neareast neighbors, Histogram rules, SVM with (so called) universal kernels...

No Free Lunch Theorem

Given a number of points (and a confidence), can we always achieve a prescribed error?
NO!

The last statement can be interpreted as follows: inference from finite samples can effectively performed if and only if the problem satisfies some a priori condition.

Hypotheses Space

In many learning algorithms (not all!) we need to choose a suitable space of hypotheses $\mathcal{H}$.

The **hypothesis space** $\mathcal{H}$ is the space of functions that we allow our algorithm to "look at". For many algorithms (such as optimization algorithms) it is the space the algorithm is allowed to search. As we will see in future classes, it is often important to choose the hypothesis space as a function of the amount of data n available.

Hypotheses Space

In many learning algorithms (not all!) we need to choose a suitable space of hypotheses $\mathcal{H}$.

The **hypothesis space** $\mathcal{H}$ is the space of functions that we allow our algorithm to "look at". For many algorithms (such as optimization algorithms) it is the space the algorithm is allowed to search. As we will see in future classes, it is often important to choose the hypothesis space as a function of the amount of data n available.

Hypotheses Space

Examples: linear functions, polynomial, RBFs, Sobolev Spaces...

Learning algorithm

A learning algorithm A is then a map from the data space to $\mathcal{H}$,

$$A(S_n) = f_n \in \mathcal{H}.$$

Hypotheses Space

Examples: linear functions, polynomial, RBFs, Sobolev Spaces...

Learning algorithm

A learning algorithm A is then a map from the data space to $\mathcal{H}$,

$$A(S_n) = f_n \in \mathcal{H}.$$

Empirical Risk Minimization

ERM

A prototype algorithm in statistical learning theory is Empirical Risk Minimization:

$$\min_{f \in \mathcal{H}} I_n[f].$$

How do we choose $\mathcal{H}$? How do we design A?

Reminder: Expected error, empirical error

Given a function f, a loss function V, and a probability distribution μ over Z, the **expected or true error** of f is:

$$I[f] = \mathbb{E}_z V[f, z] = \int_Z V(f, z) d\mu(z)$$

which is the **expected loss** on a new example drawn at random from μ.

We would like to make $I[f]$ small, but in general we do not know μ.

Given a function f, a loss function V, and a training set S consisting of n data points, the **empirical error** of f is:

$$I_S[f] = \frac{1}{n} \sum V(f, z_i)$$

Reminder: Generalization

A natural requirement for f_S is distribution independent **generalization**

$$\lim_{n\to\infty} |I_S[f_S] - I[f_S]| = 0 \ \textit{in probability}$$

This is equivalent to saying that for each n there exists a ε_n and a $\delta(\varepsilon)$ such that

$$\mathbb{P}\left\{|I_{S_n}[f_{S_n}] - I[f_{S_n}]| \geq \varepsilon_n\right\} \leq \delta(\varepsilon_n), \tag{1}$$

with ε_n and δ going to zero for $n \to \infty$.
In other words, the training error for the solution must converge to the expected error and thus be a "proxy" for it. Otherwise the solution would not be "predictive".
A desirable additional requirement is **consistency**

$$\varepsilon > 0 \ \lim_{n\to\infty} \mathbb{P}\left\{ I[f_S] - \inf_{f\in\mathcal{H}} I[f] \geq \varepsilon \right\} = 0.$$

A learning algorithm should be well-posed, eg stable

In addition to the key property of generalization, a "good" learning algorithm should also be *stable*: f_S should depend continuously on the training set S. In particular, changing one of the training points should affect less and less the solution as n goes to infinity. Stability is a good requirement for the learning problem and, in fact, for any mathematical problem. We open here a small parenthesis on stability and well-posedness.

General definition of Well-Posed and Ill-Posed problems

A problem is **well-posed** if its solution:

- exists
- is unique
- depends continuously on the data (e.g. it is *stable*)

A problem is **ill-posed** if it is not well-posed. In the context of this class, well-posedness is mainly used to mean *stability* of the solution.

More on well-posed and ill-posed problems

Hadamard introduced the definition of ill-posedness. Ill-posed problems are typically inverse problems.
As an example, assume g is a function in Y and u is a function in X, with Y and X Hilbert spaces. Then given the linear, continuous operator L, consider the equation

$$g = Lu.$$

The direct problem is is to compute g given u; the inverse problem is to compute u given the data g. In the learning case L is somewhat similar to a "sampling" operation and the inverse problem becomes the problem of finding a function that takes the values

$$f(x_i) = y_i, i = 1, \ldots n$$

The inverse problem of finding u is well-posed when

- the solution exists,
- is unique and
- is *stable*, that is depends continuously on the initial data g.

Given a training set S and a function space $\mathcal{H}$, empirical risk minimization as we have seen is the class of algorithms that look at S and select f_S as

$$f_S = \arg\min_{f \in \mathcal{H}} I_S[f].$$

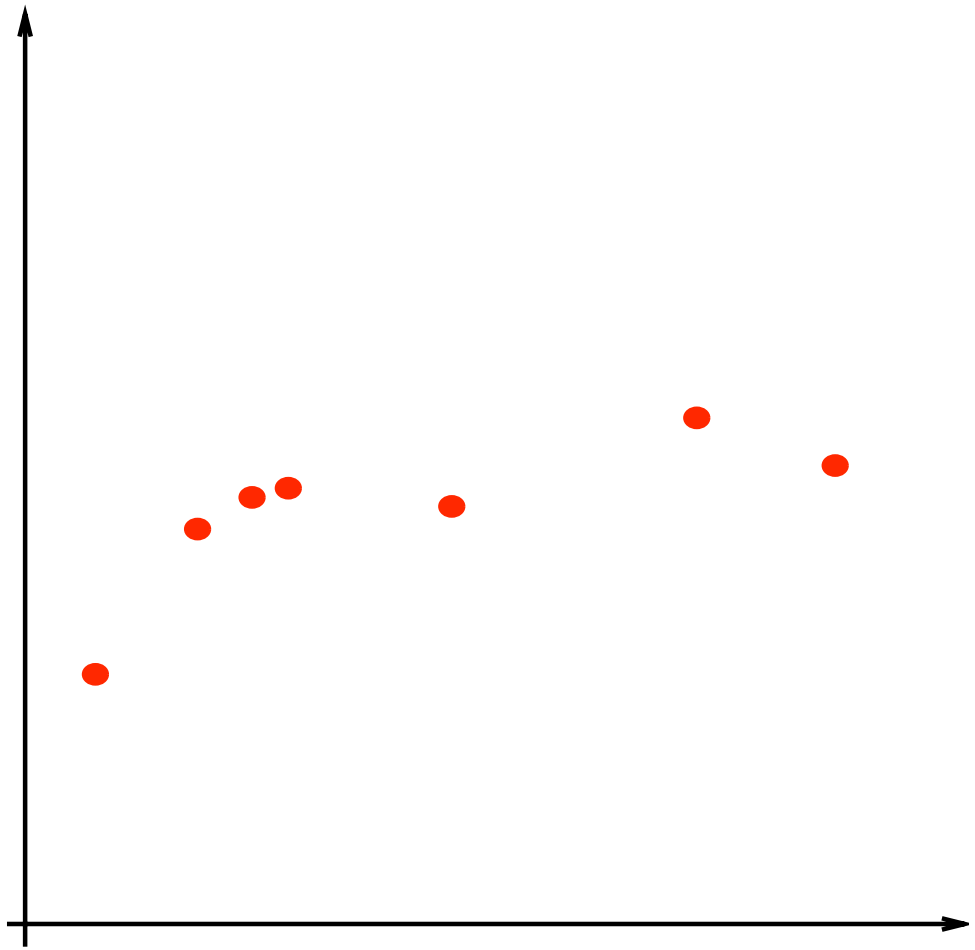
For example linear regression is ERM when $V(z) = (f(x) - y)^2$ and H is space of linear functions $f = ax$.

Generalization and Well-posedness of Empirical Risk Minimization

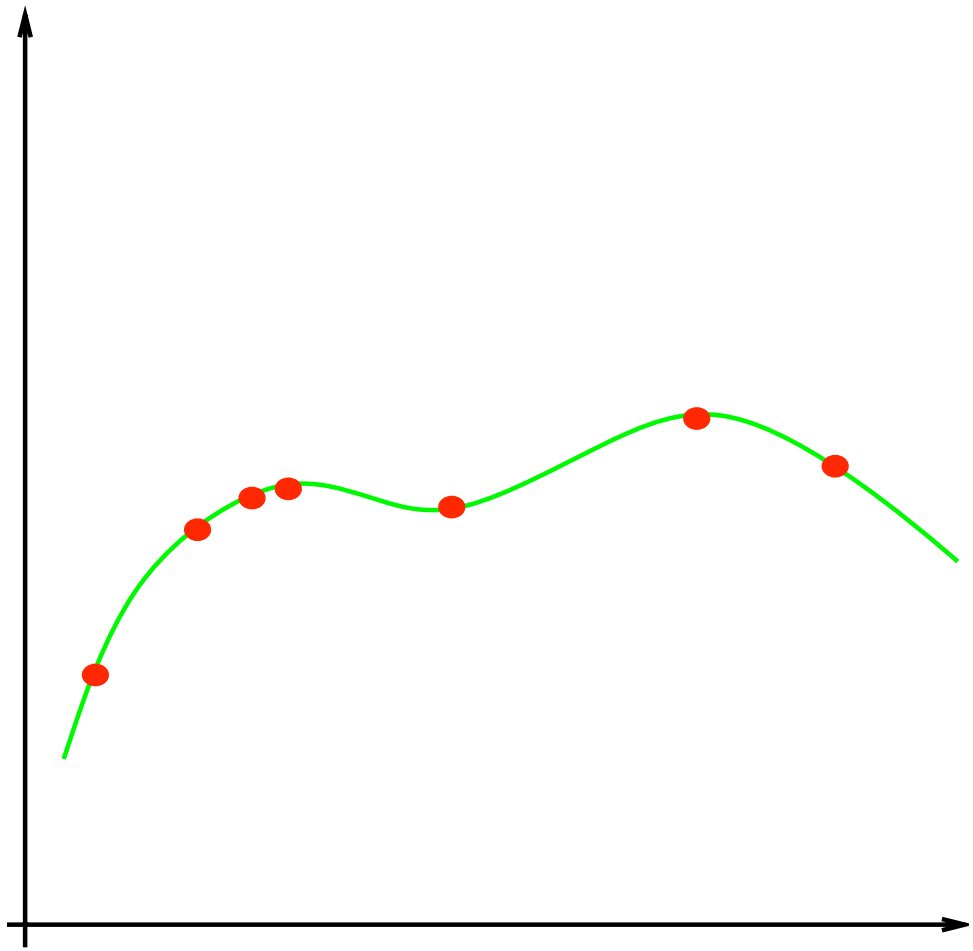
For ERM to represent a “good” class of learning algorithms, the solution should

- *generalize*
- exist, be unique and – especially – be *stable* (well-posedness), according to *some* definition of stability.

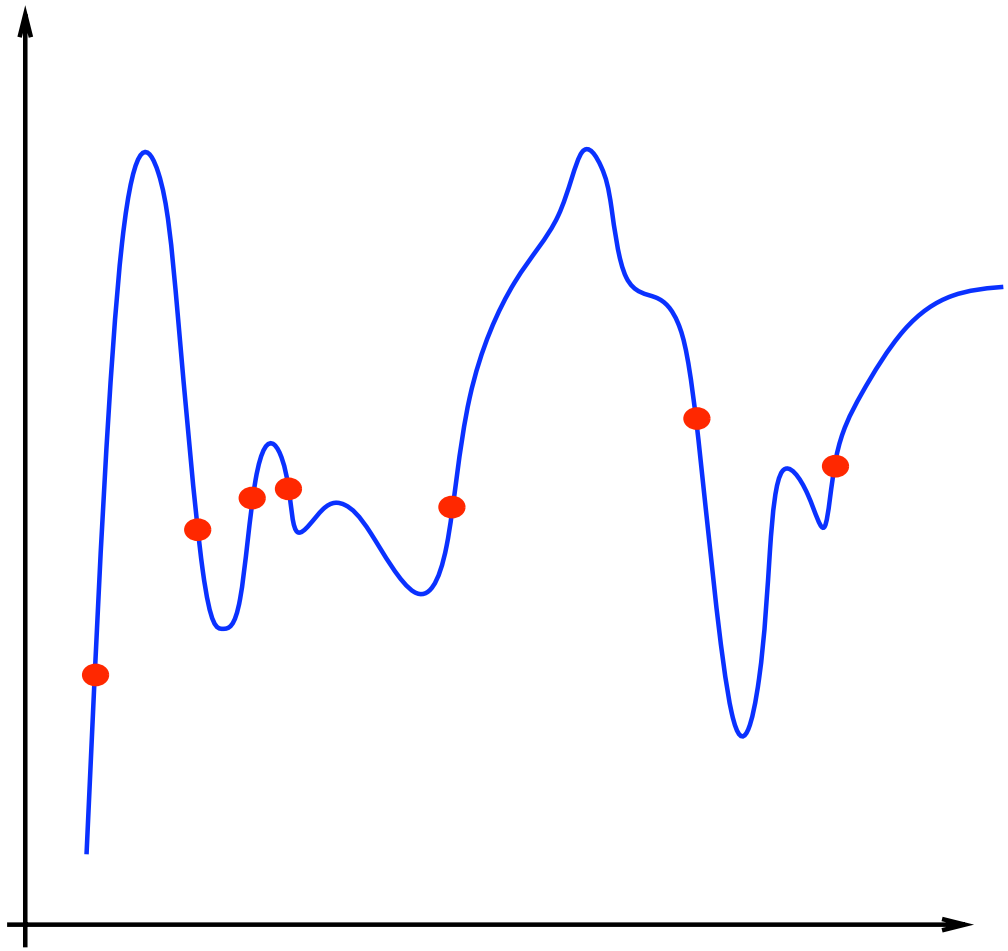
ERM and generalization: given a certain number of samples...



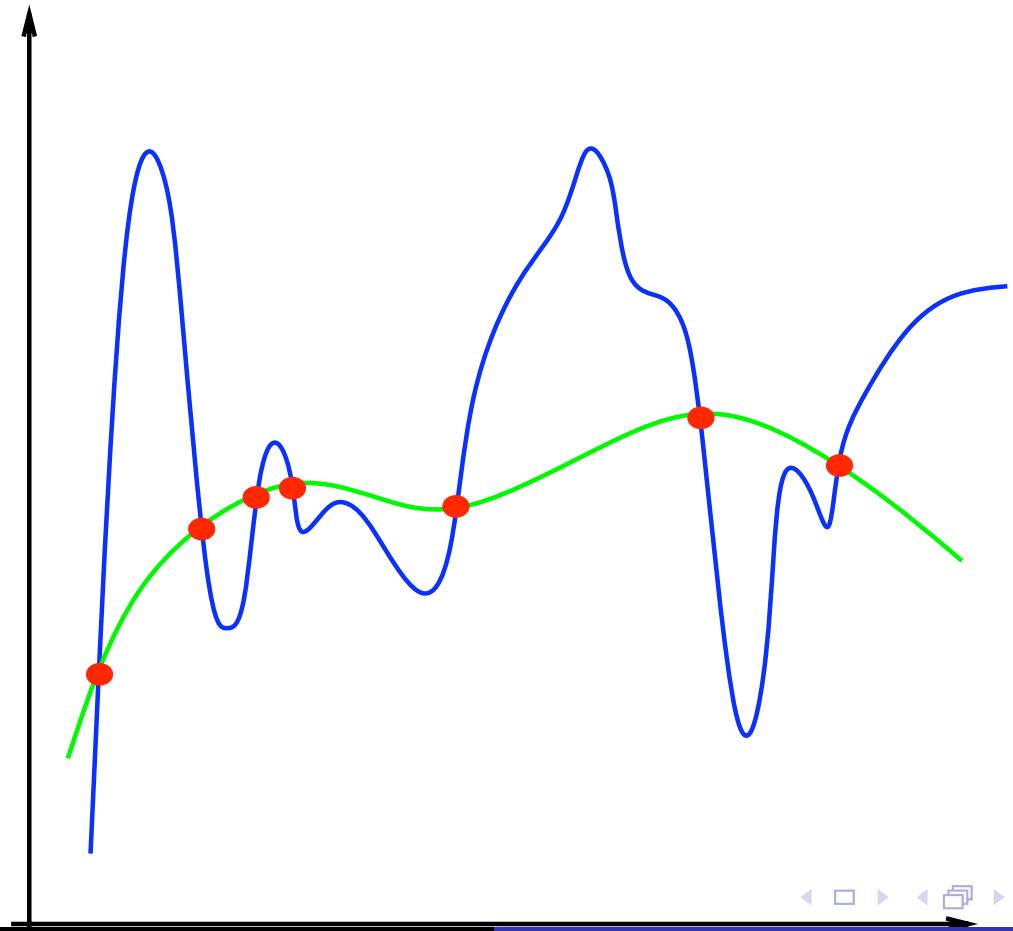
...suppose this is the “true” solution...



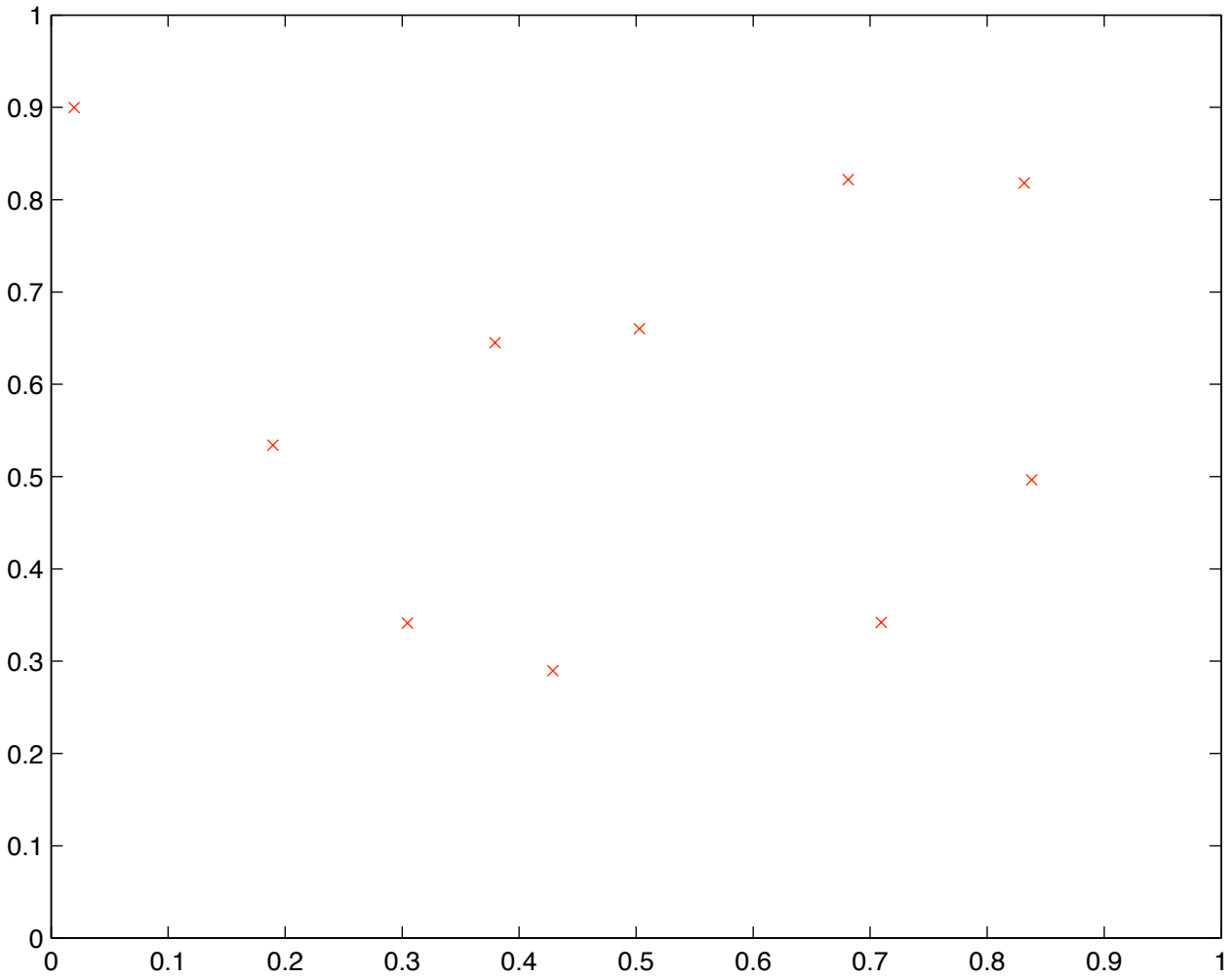
... but suppose ERM gives this solution.



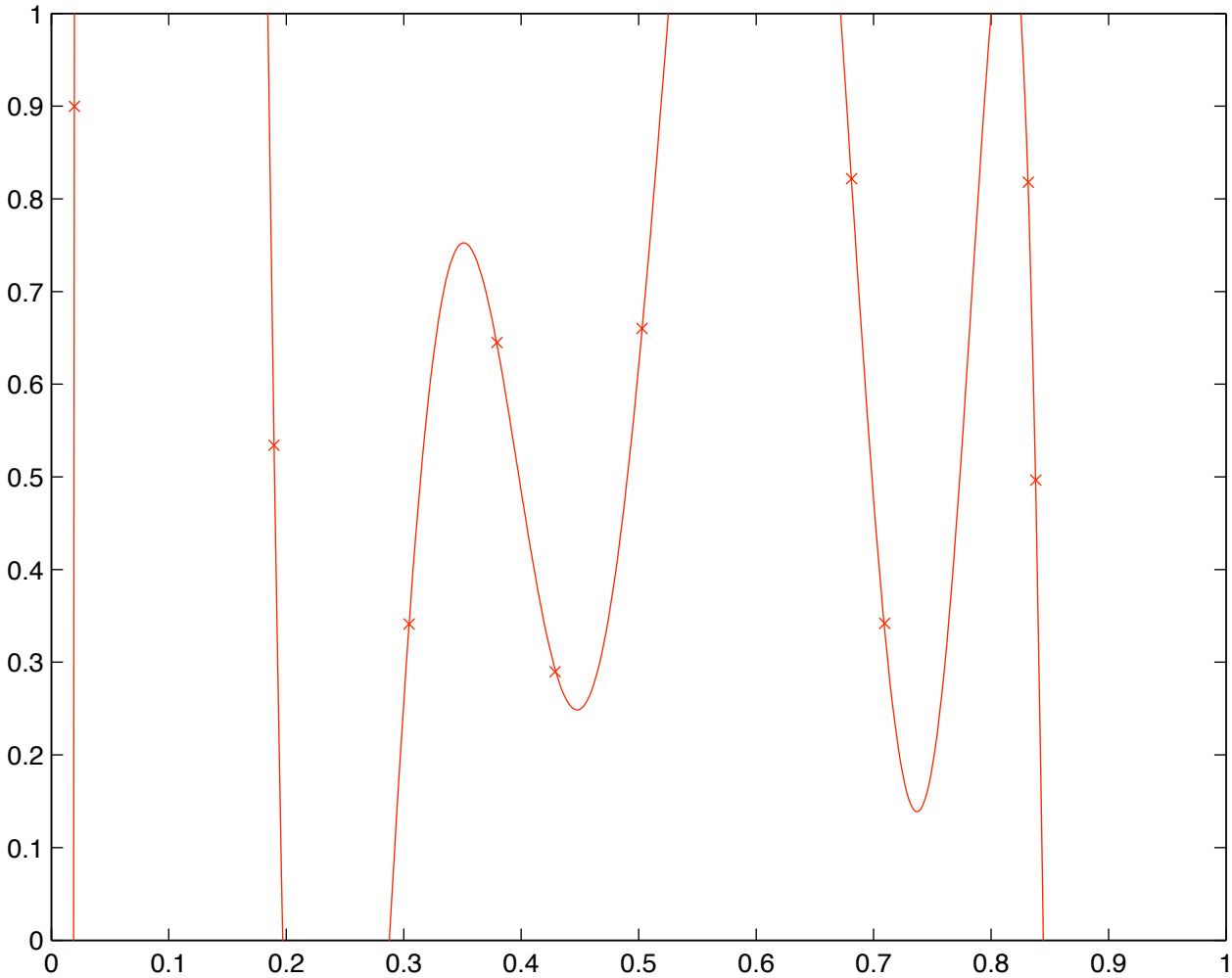
Under which conditions the ERM solution converges with increasing number of examples to the true solution? In other words...what are the conditions for generalization of ERM?



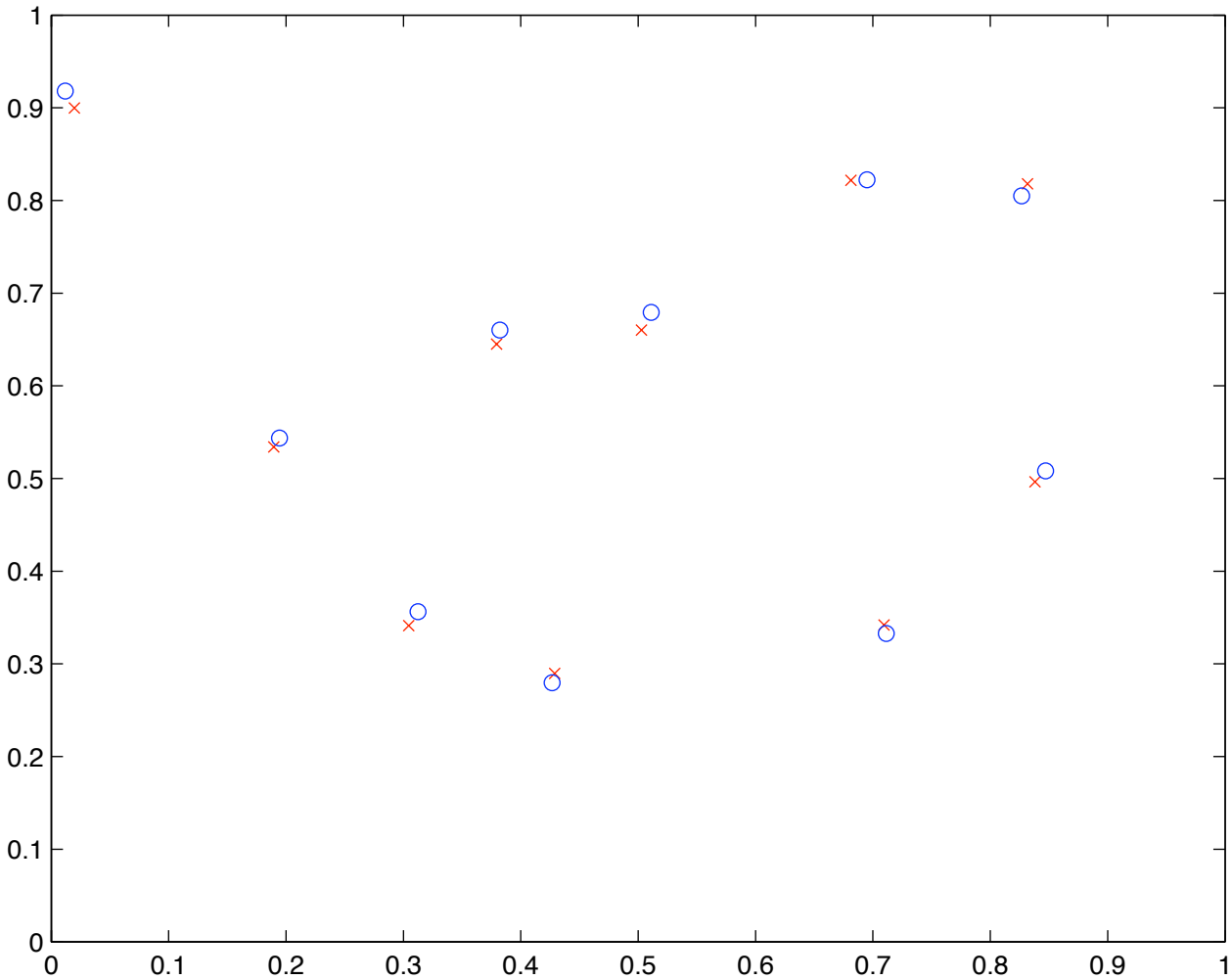
ERM and stability: given 10 samples...



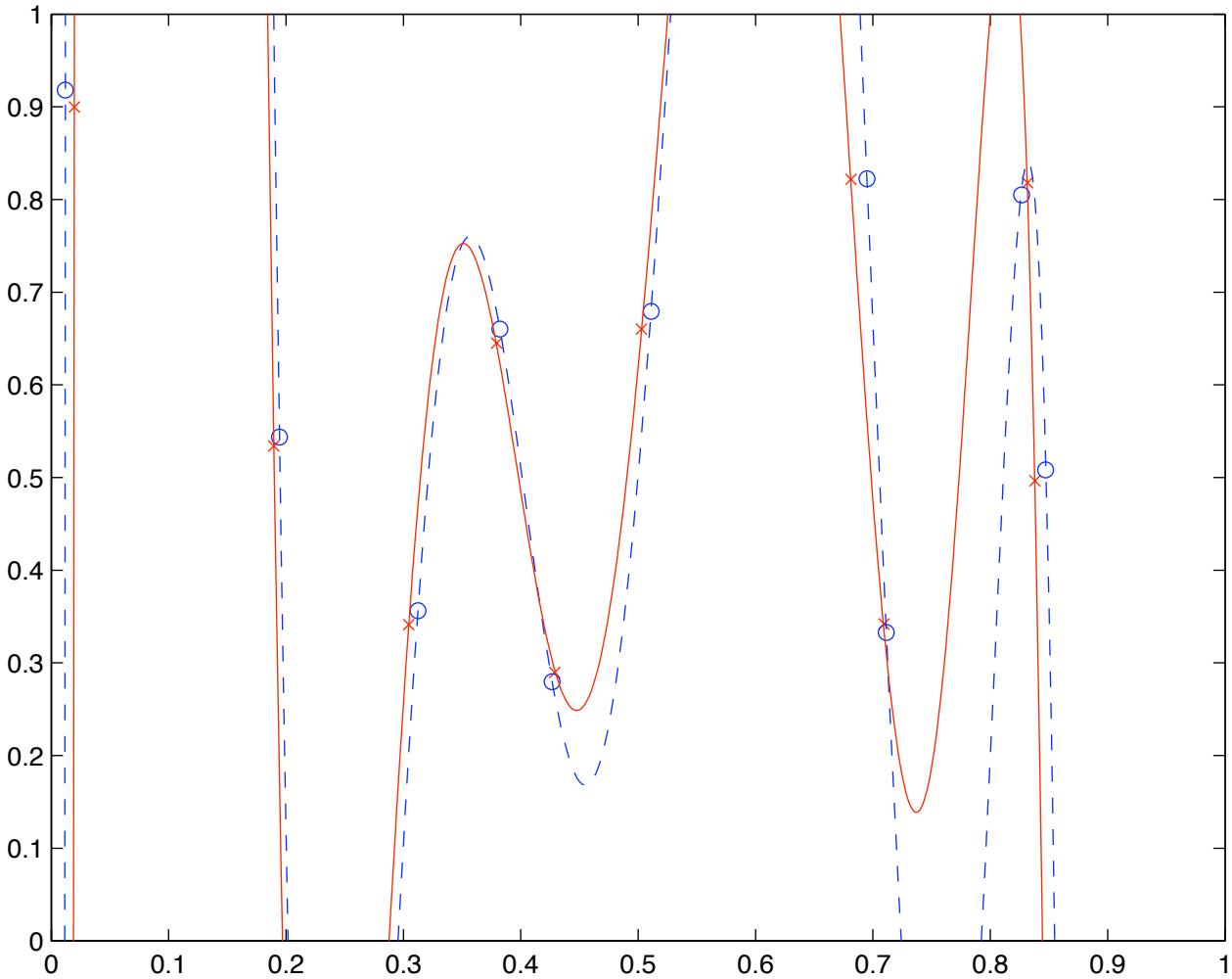
...we can find the smoothest interpolating polynomial (which degree?).



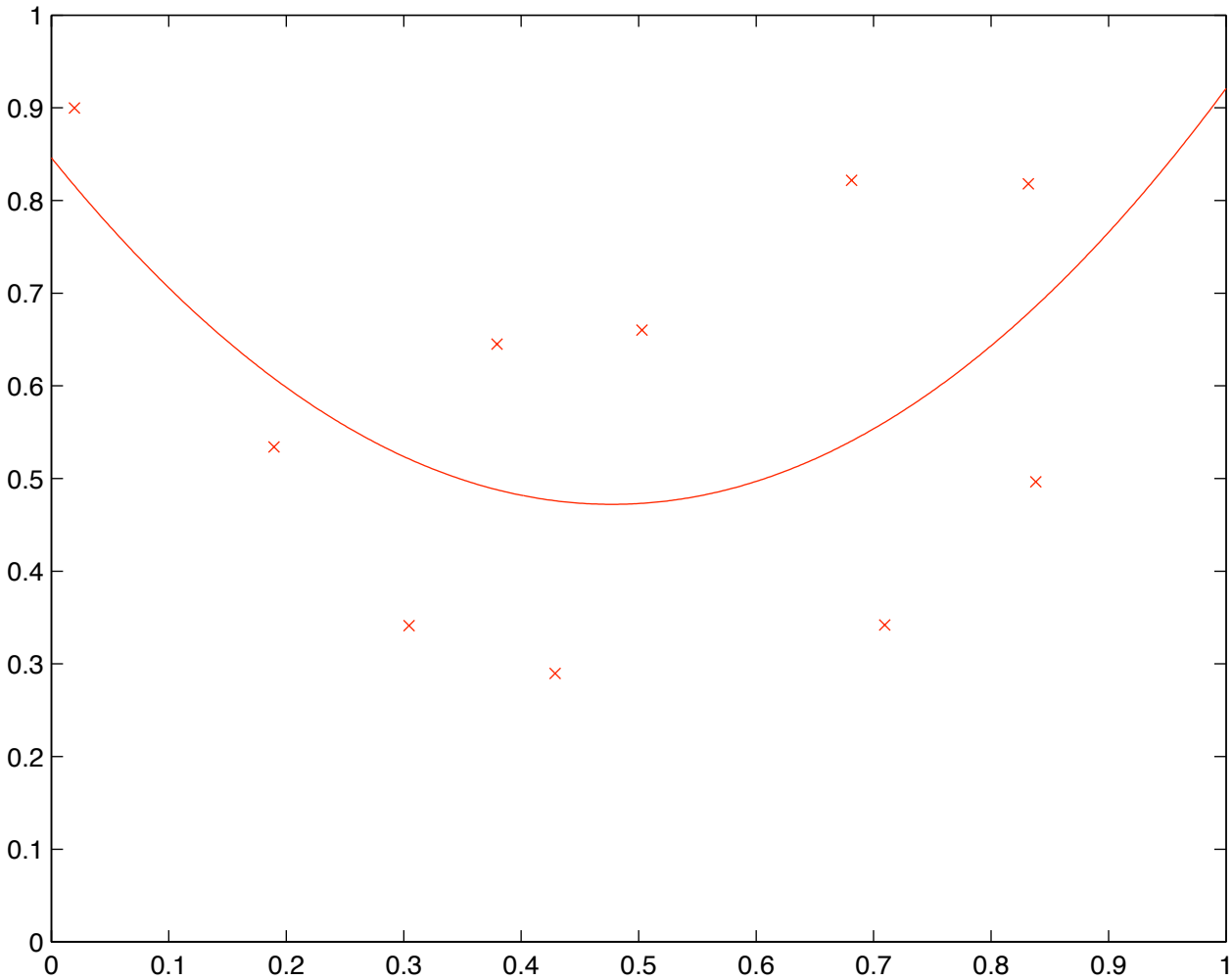
But if we perturb the points slightly...



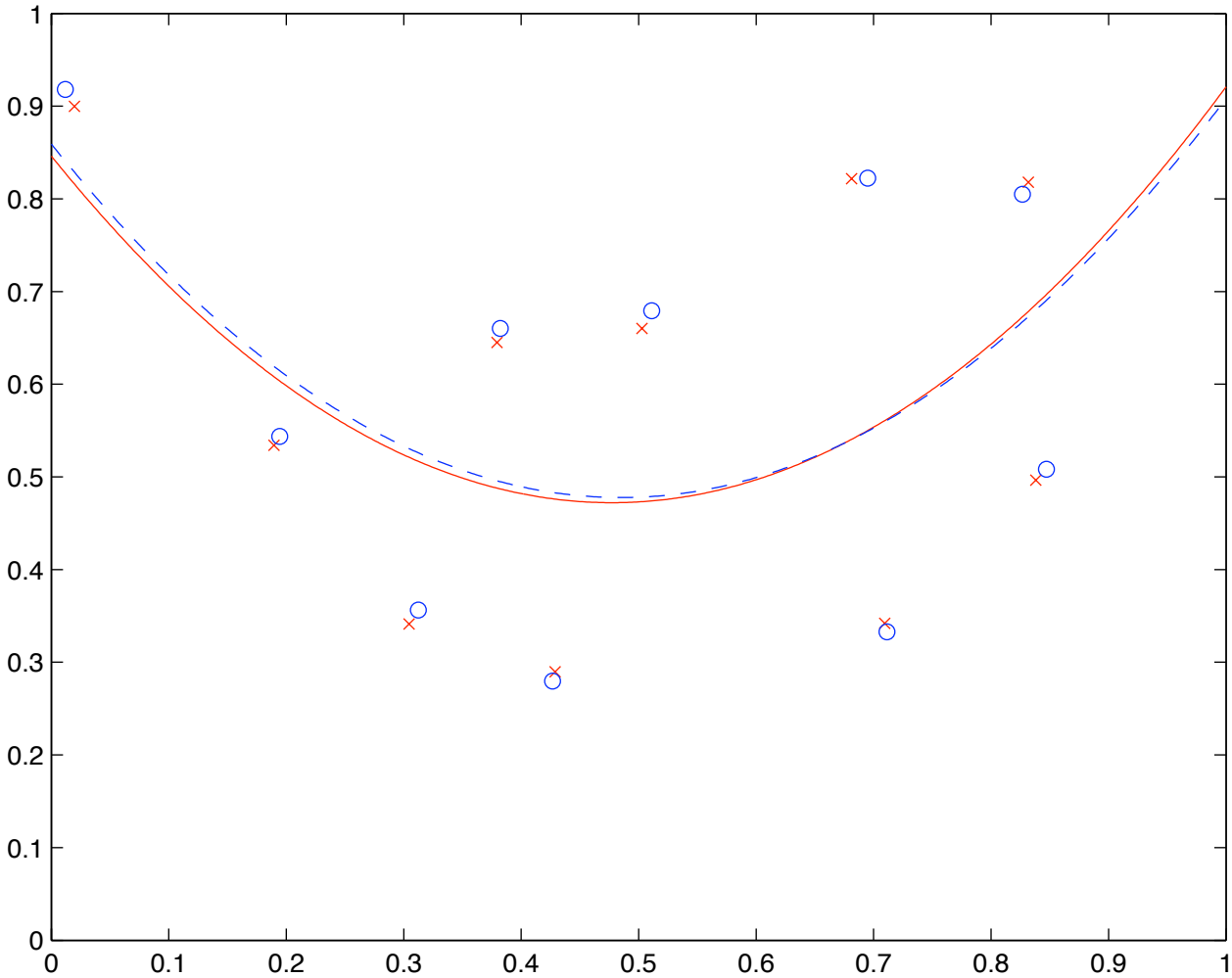
...the solution changes a lot!



If we restrict ourselves to degree two polynomials...



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



ERM: conditions for well-posedness (stability) and predictivity (generalization)

Since Tikhonov, it is well-known that a generally ill-posed problem such as ERM, can be guaranteed to be well-posed and therefore *stable* by an appropriate choice of $\mathcal{H}$. For example, compactness of $\mathcal{H}$ guarantees stability.
It seems intriguing that Vapnik's (see also Cucker and Smale) *classical conditions for consistency of ERM* – thus quite a different property – consist of appropriately restricting $\mathcal{H}$. It seems that the same restrictions that make the approximation of the data stable, may provide solutions that generalize...

ERM: conditions for well-posedness (stability) and predictivity (generalization)

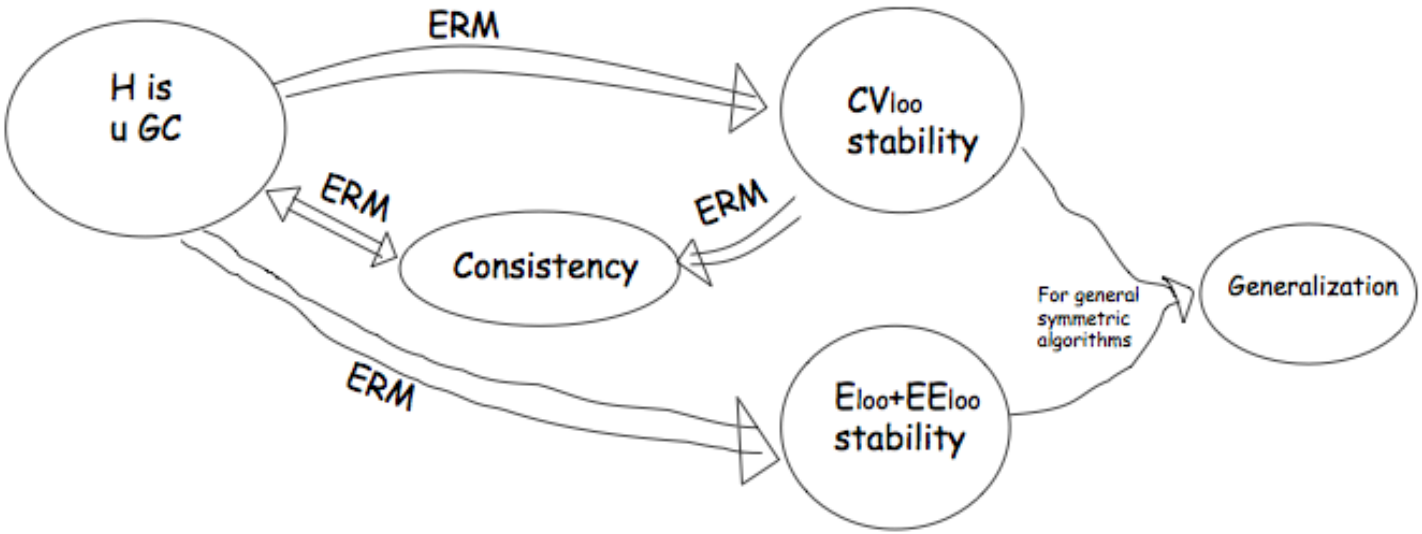
We would like to have a hypothesis space that yields generalization. Loosely speaking this would be a H for which the solution of ERM, say f_S is such that $|I_S[f_S] - I[f_S]|$ converges to zero in probability for n increasing.
Note that the above requirement is NOT the law of large numbers; the requirement for a fixed f that $|I_S[f] - I[f]|$ converges to zero in probability for n increasing IS the law of large numbers.

ERM: conditions for well-posedness (stability) and predictivity (generalization) in the case of regression and classification

- The theorem (Vapnik et al.) says that a proper choice of the hypothesis space $\mathcal{H}$ ensures generalization of ERM (and consistency since for ERM generalization is necessary and sufficient for consistency and viceversa). Other results characterize uGC classes in terms of measures of complexity or capacity of H (such as VC dimension).
- A separate theorem (Niyogi, Mukherjee, Rifkin, Poggio) says that stability (defined in a specific way) of (supervised) ERM is sufficient and necessary for generalization of ERM. Thus with the appropriate definition of stability, *stability and generalization are equivalent for ERM; stability and H uGC are also equivalent.*

Thus the two desirable conditions for a supervised learning algorithm – generalization and stability – are equivalent (and they correspond to the same constraints on $\mathcal{H}$).

Key Theorem(s) Illustrated



L

Regularization

The “equivalence” between generalization and stability gives us a an approach to predictive algorithms. It is enough to remember that regularization is the classical way to restore well posedness. Thus regularization becomes a way to ensure generalization. Regularization in general means retricting H, as we have in fact done for ERM. There are two standard approaches in the field of ill-posed problems that ensure for ERM *well-posedness* (and *generalization*) by constraining the hypothesis space $\mathcal{H}$. The direct way – minimize the empirical error subject to f in a ball in an appropriate $\mathcal{H}$ – is called *Ivanov regularization*. The indirect way is *Tikhonov regularization* (which is not strictly ERM).

Ivanov and Tikhonov Regularization

ERM finds the function in $(\mathcal{H})$ which minimizes

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

which in general – for arbitrary hypothesis space $\mathcal{H}$ – is *ill-posed*.

- Ivanov regularizes by finding the function that minimizes

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

  while satisfying $\mathcal{R}(f) \leq A$.

- Tikhonov regularization minimizes over the hypothesis space $\mathcal{H}$, for a fixed positive parameter γ, the regularized functional

$$\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i) + \gamma\mathcal{R}(f). \tag{2}$$

$\mathcal{R}(f)$ is the regulirizer, a penalization on f. In this course we will mainly discuss the case $\mathcal{R}(f) = \|f\|_K^2$ where $\|f\|_K^2$ is the norm in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, defined by the kernel K.

Tikhonov Regularization

As we will see in future classes

- Tikhonov regularization ensures well-posedness eg existence, uniqueness and especially *stability* (in a very strong form) of the solution
- Tikhonov regularization ensures generalization
- Tikhonov regularization is closely related to – but different from – Ivanov regularization, eg ERM on a hypothesis space $\mathcal{H}$ which is a ball in a RKHS.

Remarks on Foundations of Learning Theory

Intelligent behavior (at least learning) consists of optimizing under constraints. Constraints are key for solving computational problems; constraints are key for prediction. Constraints may correspond to rather general symmetry properties of the problem (eg time invariance, space invariance, invariance to physical units (pai theorem), universality of numbers and metrics implying normalization, etc.)

- Key questions at the core of learning theory:
  - generalization and predictivity *not* explanation
  - probabilities are unknown, only data are given
  - which constraints are needed to ensure generalization (therefore which hypotheses spaces)?
  - regularization techniques result usually in *computationally "nice"* and *well-posed* optimization problems

Statistical Learning Theory and Bayes

Unlike statistical learning theory the Bayesian approach does not emphasize

- the issue of generalization (following the tradition in statistics of explanatory statistics);
- that probabilities are not known and that only data are known: assuming a specific distribution is a very strong – unconstrained by any Bayesian theory – seat-of-the-pants guess;
- the question of which priors are needed to ensure generalization;
- that the resulting optimization problems are often *computationally intractable* and possibly ill-posed optimization problems (for instance not unique).

Plan

- Part I: Basic Concepts and Notation
- Part II: Foundational Results
- Part III: Algorithms

INSTEAD....

Appendix: Target Space, Sample and Approximation Error

In addition to the hypothesis space $\mathcal{H}$, the space we allow our algorithms to search, we define...
The **target space** $\mathcal{T}$ is a space of functions, chosen a priori in any given problem, that is assumed to contain the "true" function f_0 that minimizes the risk. Often, $\mathcal{T}$ is chosen to be all functions in L_2, or all differentiable functions. Notice that the "true" function if it exists is defined by $\mu(z)$, which contains all the relevant information.

Sample Error (also called Estimation Error)

Let $f_{\mathcal{H}}$ be the function in $\mathcal{H}$ with the smallest true risk.
We have defined the **generalization error** to be $I_S[f_S] - I[f_S]$.
We define the **sample error** to be $I[f_S] - I[f_{\mathcal{H}}]$, the difference in true risk between the best function in $\mathcal{H}$ and the function in $\mathcal{H}$ we actually find. This is what we pay because our finite sample does not give us enough information to choose to the "best" function in $\mathcal{H}$. We'd like this to be small. *Consistency* – defined earlier – is equivalent to the sample error going to zero for $n \to \infty$.
A main goal in classical learning theory (Vapnik, Smale, ...) is "bounding" the generalization error. Another goal – for learning theory *and* statistics – is bounding the sample error, that is determining conditions under which we can state that $I[f_S] - I[f_{\mathcal{H}}]$ will be small (with high probability).
As a simple rule, we expect that if $\mathcal{H}$ is "well-behaved", then, as n gets large the sample error will become small.

Approximation Error

Let f_0 be the function in $\mathcal{T}$ with the smallest true risk.
We define the **approximation error** to be $I[f_{\mathcal{H}}] - I[f_0]$, the difference in true risk between the best function in $\mathcal{H}$ and the best function in $\mathcal{T}$. This is what we pay when $\mathcal{H}$ is smaller than $\mathcal{T}$. We'd like this error to be small too. In much of the following we can assume that $I[f_0] = 0$.
We will focus less on the approximation error in 9.520, but we will explore it.
As a simple rule, we expect that as $\mathcal{H}$ grows bigger, the approximation error gets smaller. If $\mathcal{T} \subseteq \mathcal{H}$ – which is a situation called *the realizable setting* –the approximation error is zero.

Error

We define the **error** to be $I[f_S] - I[f_0]$, the difference in true risk between the function we actually find and the best function in $\mathcal{T}$. We'd really like this to be small. As we mentioned, often we can assume that the **error** is simply $I[f_S]$.
The error is the sum of the sample error and the approximation error:

$$I[f_S] - I[f_0] = (I[f_S] - I[f_{\mathcal{H}}]) + (I[f_{\mathcal{H}}] - I[f_0])$$

If we can make both the approximation and the sample error small, the error will be small. There is a tradeoff between the approximation error and the sample error...

The Approximation/Sample Tradeoff

It should already be intuitively clear that making $\mathcal{H}$ big makes the approximation error small. This implies that we can (help) make the error small by making $\mathcal{H}$ big.
On the other hand, we will show that making $\mathcal{H}$ small will make the sample error small. In particular for ERM, if $\mathcal{H}$ is a uGC class, the generalization error and the sample error will go to zero as $n \to \infty$, but how quickly depends directly on the "size" of $\mathcal{H}$. This implies that we want to keep $\mathcal{H}$ as small as possible. (Furthermore, $\mathcal{T}$ itself may or may not be a uGC class.)
Ideally, we would like to find the optimal tradeoff between these conflicting requirements.

Generalization, Sample Error and Approximation Error

Generalization error is $I_S[f_S] - I[f_S]$.
Sample error is $I[f_S] - I[f_{\mathcal{H}}]$
Approximation error is $I[f_{\mathcal{H}}] - I[f_0]$
Error is $I[f_S] - I[f_0] = (I[f_S] - I[f_{\mathcal{H}}]) + (I[f_{\mathcal{H}}] - I[f_0])$

Plan

- Part I: Basic Concepts and Notation
- Part II: Foundational Results
- Part III: Algorithms

Hypotheses Space

We are going to look at hypotheses spaces which are reproducing kernel Hilbert spaces.

- RKHS are **Hilbert spaces** of **point-wise defined** functions.
- They can be defined via a **reproducing kernel**, which is a symmetric positive definite function.

$$\sum_{i,j=1}^{n} c_i c_j K(t_i, t_j) \geq 0$$

for any $n \in \mathbb{N}$ and choice of $t_1, ..., t_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$.

- functions in the space are (the completion of) linear combinations

$$f(x) = \sum_{i=1}^{p} K(x, x_i) c_i.$$

- the norm in the space is a natural measure of complexity

$$p$$

Hypotheses Space

We are going to look at hypotheses spaces which are reproducing kernel Hilbert spaces.

- RKHS are **Hilbert spaces** of **point-wise defined** functions.
- They can be defined via a **reproducing kernel**, which is a symmetric positive definite function.

$$\sum_{i,j=1}^{n} c_i c_j K(t_i, t_j) \geq 0$$

for any $n \in \mathbb{N}$ and choice of $t_1, ..., t_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$.

- functions in the space are (the completion of) linear combinations

$$f(x) = \sum_{i=1}^{p} K(x, x_i) c_i.$$

- the norm in the space is a natural measure of complexity

$$p$$

Hypotheses Space

We are going to look at hypotheses spaces which are reproducing kernel Hilbert spaces.

- RKHS are **Hilbert spaces** of **point-wise defined** functions.
- They can be defined via a **reproducing kernel**, which is a symmetric positive definite function.

$$\sum_{i,j=1}^{n} c_i c_j K(t_i, t_j) \geq 0$$

for any $n \in \mathbb{N}$ and choice of $t_1, ..., t_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$.

- functions in the space are (the completion of) linear combinations

$$f(x) = \sum_{i=1}^{p} K(x, x_i) c_i.$$

- the norm in the space is a natural measure of complexity

Hypotheses Space

We are going to look at hypotheses spaces which are reproducing kernel Hilbert spaces.

- RKHS are **Hilbert spaces** of **point-wise defined** functions.
- They can be defined via a **reproducing kernel**, which is a symmetric positive definite function.

$$\sum_{i,j=1}^{n} c_i c_j K(t_i, t_j) \geq 0$$

for any $n \in \mathbb{N}$ and choice of $t_1, ..., t_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$.

- functions in the space are (the completion of) linear combinations

$$f(x) = \sum_{i=1}^{p} K(x, x_i) c_i.$$

- the norm in the space is a natural measure of complexity

$$p$$

Examples of pd kernels

Very common examples of symmetric pd kernels are

- **Linear kernel**

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

- **Gaussian kernel**

$$K(x, x') = \mathrm{e}^{-\frac{\|x-x'\|^2}{\sigma^2}}, \qquad \sigma > 0$$

- **Polynomial kernel**

$$K(x, x') = (x \cdot x' + 1)^d, \qquad d \in \mathbb{N}$$

For specific applications, designing an effective kernel is a challenging problem.

Kernel and Features

Often times kernels, are defined through a dictionary of features

$$\mathcal{D} = \{\phi_j,\ i = 1, \ldots, p \mid \phi_j : X \to \mathbb{R},\ \ \forall j\}$$

setting

$$K(x, x') = \sum_{i=1}^{p} \phi_j(x)\phi_j(x').$$

Ivanov regularization

We can regularize by explicitly restricting the hypotheses space $\mathcal{H}$ — for example to a ball of radius R.

Ivanov regularization

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i)$$

subject to

$$\|f\|_{\mathcal{H}}^2 \leq R.$$

The above algorithm corresponds to a constrained optimization problem.

Tikhonov regularization

Regularization can also be done implicitly via penalization

Tikhonov regularizarion

$$\arg\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i) + \lambda \|f\|_{\mathcal{H}}^2 .$$

λ is the regularization parameter trading-off between the two terms.

The above algorithm can be seen as the Lagrangian formulation of a constrained optimization problem.

The Representer Theorem

An important result

The minimizer over the RKHS $\mathcal{H}$, f_S, of the regularized empirical functional

$$I_S[f] + \lambda \|f\|_{\mathcal{H}}^2,$$

can be represented by the expression

$$f_n(x) = \sum_{i=1}^{n} c_i K(x_i, x),$$

for some $(c_1, \ldots, c_n) \in \mathbb{R}$.

Hence, minimizing over the (possibly infinite dimensional) Hilbert space, *boils down to minimizing over* $\mathbb{R}^n$.

SVM and RLS

The way the coefficients $\mathbf{c} = (c_1, \ldots, c_n)$ are computed depend on the loss function choice.

- RLS: Let Let $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{K}_{i,j} = K(x_i, x_j)$ then $\mathbf{c} = (\mathbf{K} + \lambda n I)^{-1}\mathbf{y}$.
- SVM: Let $\alpha_i = y_i c_i$ and $\mathbf{Q}_{i,j} = y_i K(x_i, x_j) y_j$

$$\max_{\alpha \in \mathbb{R}^n} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2}\alpha^T \mathbf{Q}\alpha$$

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

$$0 \leq \alpha_i \leq C \qquad i = 1, \ldots, n$$

SVM and RLS

The way the coefficients $\mathbf{c} = (c_1, \ldots, c_n)$ are computed depend on the loss function choice.

- RLS: Let Let $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{K}_{i,j} = K(x_i, x_j)$ then $\mathbf{c} = (\mathbf{K} + \lambda n I)^{-1}\mathbf{y}$.
- SVM: Let $\alpha_i = y_i c_i$ and $\mathbf{Q}_{i,j} = y_i K(x_i, x_j) y_j$

$$\max_{\alpha \in \mathbb{R}^n} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2}\alpha^T \mathbf{Q}\alpha$$

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

$$0 \leq \alpha_i \leq C \qquad i = 1, \ldots, n$$

Bayes Interpretation

- **Empirical risk minimization** is ML.

$$p(\mathbf{Y}|f,\mathbf{X}) \propto e^{-\frac{1}{2}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

- **Linear RLS** is MAP.

$$p(\mathbf{Y},f|\mathbf{X}) \propto e^{-\frac{1}{2}\sum_{i=1}^{N}(y_i - \langle x_i,\theta\rangle)^2} \cdot e^{-\frac{\lambda}{2}\theta^T\theta}$$

- **Kernel RLS** is also MAP.

$$p(\mathbf{Y},f|\mathbf{X}) \propto e^{-\frac{1}{2}\sum_{i=1}^{N}(y_i - f(x_i)^2} \cdot e^{-\frac{\lambda}{2}\|f\|_{\mathcal{H}}^2}$$

Regularization approach

More generally we can consider:

$$I_n(f) + \lambda R(f)$$

where, $R(f)$ is a regularizing functional.

- Sparsity based methods
- Manifold learning
- Multiclass
- ...

Summary

- statistical learning as a foundational framework to predict from data
- a proxy for predictivity is the empirical error *iff* generalization holds for the class of algorithms
- stability and generalization are equivalent
- regularization as a fundamental tool in learning algorithm to ensure stability and generalization

Generalization, Sample Error and Approximation Error

Generalization error is $I_S[f_S] - I[f_S]$.
Sample error is $I[f_S] - I[f_{\mathcal{H}}]$
Approximation error is $I[f_{\mathcal{H}}] - I[f_0]$
Error is $I[f_S] - I[f_0] = (I[f_S] - I[f_{\mathcal{H}}]) + (I[f_{\mathcal{H}}] - I[f_0])$

Final (optional) Remarks

Remarks: constrained optimization

Intelligent behavior (at least learning) consists of optimizing under constraints. Constraints are key for solving computational problems; constraints are key for prediction. Constraints may correspond to rather general symmetry properties of the problem (eg time invariance, space invariance, invariance to physical units (π theorem), universality of numbers and metrics implying normalization, etc.)

ERM: conditions for well-posedness (stability) and predictivity (generalization) in the case of regression and classification

Theorem [Vapnik and Červonenkis (71), Alon et al (97), Dudley, Giné, and Zinn (91)]

A (necessary) and sufficient condition for generalization (and consistency) of ERM is that $\mathcal{H}$ is uGC.

Definition

$\mathcal{H}$ is a (weak) uniform Glivenko-Cantelli (uGC) class if

$$\forall \varepsilon > 0 \lim_{n \to \infty} \sup_{\mu} \mathbb{P}_S \left\{ \sup_{f \in \mathcal{H}} |I[f] - I_S[f]| > \varepsilon \right\} = 0.$$

Key Theorem(s)

Uniform Glivenko-Cantelli Classes

We say that $\mathcal{H}$ is a uniform Glivenko-Cantelli (uGC) class, if for all p,

$$\forall \epsilon > 0 \lim_{n \to \infty} \mathbb{P} \left\{ \sup_{f \in \mathcal{H}} |I[f] - I_n[f]| > \epsilon \right\} = 0.$$

A necessary and sufficient condition for consistency of ERM is that $\mathcal{H}$ is uGC.

See: [Vapnik and Červonenkis (71), Alon et al (97), Dudley, Giné, and Zinn (91)].

In turns the UGC property is equivalent to requiring $\mathcal{H}$ to have finite capacity: V_γ dimension in general and VC dimension in classification.

Key Theorem(s)

Uniform Glivenko-Cantelli Classes

We say that $\mathcal{H}$ is a uniform Glivenko-Cantelli (uGC) class, if for all p,

$$\forall \epsilon > 0 \lim_{n \to \infty} \mathbb{P} \left\{ \sup_{f \in \mathcal{H}} |I[f] - I_n[f]| > \epsilon \right\} = 0.$$

A necessary and sufficient condition for consistency of ERM is that $\mathcal{H}$ is uGC.
See: [Vapnik and Červonenkis (71), Alon et al (97), Dudley, Giné, and Zinn (91)].

In turns the UGC property is equivalent to requiring $\mathcal{H}$ to have finite capacity: V_γ dimension in general and VC dimension in classification.

Stability

notation: S training set, $S^{i,z}$ training set obtained replacing the i-th example in S with a new point $z = (x, y)$.

Definition

We say that an algorithm $\mathcal{A}$ has **uniform stability** β (is β-stable) if

$$\forall (S, z) \in \mathcal{Z}^{n+1}, \ \forall i, \ \sup_{z' \in Z} |V(f_S, z') - V(f_{S^{i,z}}, z')| \leq \beta.$$

CV_{loo} Stability

$z = (x, y)$
$S = z_1, ..., z_n$
$S^i = z_1, ..., z_{i-1}, z_{i+1}, ...z_n$

CV Stability

A learning algorithm A is CV_{loo} *stable* **if** for each n there exists a $\beta_{CV}^{(n)}$ and a $\delta_{CV}^{(n)}$ such that for all p

$$\mathbb{P}\left\{|V(f_{S^i}, z_i) - V(f_S, z_i)| \leq \beta_{CV}^{(n)}\right\} \geq 1 - \delta_{CV}^{(n)},$$

with $\beta_{CV}^{(n)}$ and $\delta_{CV}^{(n)}$ going to zero for $n \to \infty$.

Kernel and Data Representation

In the above reasoning the kernel and the hypotheses space define a representation/parameterization of the problem and hence play a special role.

Where do they come from?

- There are a few off the shelf choices (Gaussian, polynomial etc.)
- Often they are the product of problem specific engineering.

Are there principles– applicable in a wide range of situations– to design effective data representation?

Kernel and Data Representation

In the above reasoning the kernel and the hypotheses space define a representation/parameterization of the problem and hence play a special role.

Where do they come from?

- There are a few off the shelf choices (Gaussian, polynomial etc.)
- Often they are the product of problem specific engineering.

Are there principles– applicable in a wide range of situations– to design effective data representation?

Kernel and Data Representation

In the above reasoning the kernel and the hypotheses space define a representation/parameterization of the problem and hence play a special role.

Where do they come from?

- There are a few off the shelf choices (Gaussian, polynomial etc.)
- Often they are the product of problem specific engineering.

Are there principles– applicable in a wide range of situations– to design effective data representation?

Kernel and Data Representation

In the above reasoning the kernel and the hypotheses space define a representation/parameterization of the problem and hence play a special role.

Where do they come from?

- There are a few off the shelf choices (Gaussian, polynomial etc.)
- Often they are the product of problem specific engineering.

Are there principles– applicable in a wide range of situations– to design effective data representation?