

GI01/4C55: Supervised Learning

1. Introduction

October 3, 2005

Massimiliano Pontil

1

Course information

1. When: Mondays, 14:00–17:00
Where: Room 105, 24 Gordon Square
2. Course webpage:
<http://www.cs.ucl.ac.uk/staff/M.Pontil/courses/index-SL05.htm>
3. Mailing list: gi01@cs.ucl.ac.uk
4. Office: 8.05, CS Building, Malet Place

2

Assessment

1. Homework (40%) and Exam (60%)
2. 5 homework assignments
(deliver them on-time, penalty otherwise)
3. To pass the course, you must obtain at least 40% on the homework component and an average of at least 50% when the homework and exam components are weighted together

3

Material

- Lecture notes
 - <http://www.cs.ucl.ac.uk/staff/M.Pontil/courses/index-SL05.htm>
- Reference book
 - Hastie, Tibshirani, & Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001
- Additional material (see webpage for more info)
 - General: Duda, Hart & Stork; Mitchell; Bishop; ...
 - Bayesian methods in ML: Mackay; ...
 - Kernel methods: Shawe-Taylor & Cristianini; Schölkopf & Smola; ...
 - Learning theory: Devroye, Lugosi, & Györfi; Vapnik; ...

4

Prerequisites

- Calculus (real-valued functions, limits, derivatives, Taylor series, integrals,...)
- Elements of probability theory (random variables, expectation, variance, conditional probabilities, Bayes rule,...)
- Fundamentals of linear algebra (vectors, angles, matrices, eigenvectors/eigenvalues,...),
- A bit of optimization theory (convex functions, Lagrange multipliers)

5

Course outline

- (Weeks 1,2) Key concepts (probabilistic formulation of learning from examples, error functionals; loss function, Bayes rule, learning algorithm, overfitting and underfitting, model selection, cross validation); Some basic learning algorithms (linear regression, k -NN, linear discriminant analysis, logistic regression); Discriminative vs. Generative approach
- (Week 3) Hypothesis spaces (Linear models, kernels, neural nets, trees), Optimization algorithms
- (Week 4) Elements of Learning Theory (PAC-learning, Complexity of hypothesis space, VC-bounds)
- (Week 5) Regularization, Kernels

6

Course outline (cont.)

- (Week 6) Support vector machines, Gaussian processes
- (Week 7) Decision trees, Boosting
- (Week 8) Online learning
- (Week 9) Semi-supervised learning
- (Week 10) Multi-task learning

Acknowledgments: Some lectures use material from last year course. Thanks to Fernando Perez-Cruz, Iain Murray and Ed Snelson of the Gatsby Unit at UCL for preparing this!

7

Today plan

- Supervised learning problem
- Regression and classification
- Two learning algorithms: least squares and k -NN
- Probabilistic model, error functional, optimal solutions
- Hypothesis space, overfitting and underfitting
- Choice of the learning algorithm (Model selection)

8

Supervised learning problem

Given a set of **input/output** pairs (**training set**) we wish to compute the functional relationship between the input and the output

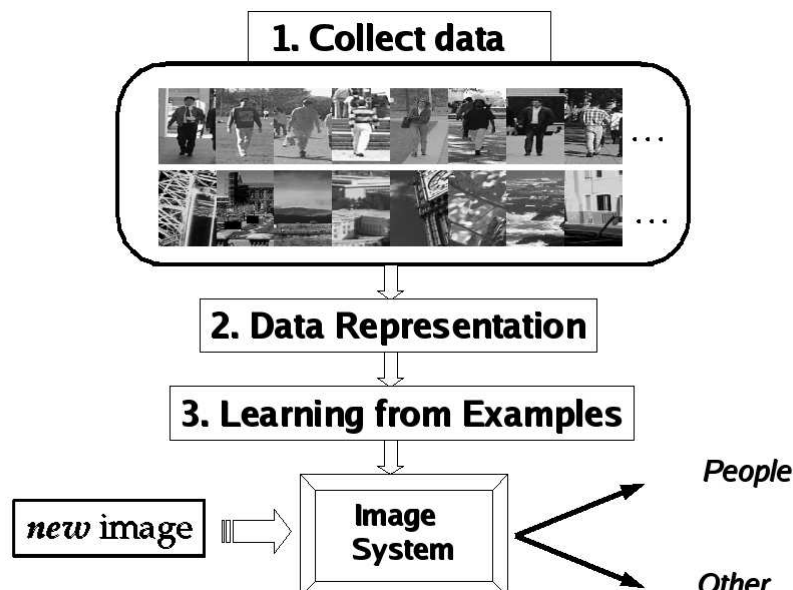
$$x \longrightarrow \boxed{f} \longrightarrow y$$

- **Ex1** (people detection) given an image we wish to say if it depicts a person or not
The output is one of 2 possible categories
- **Ex2** (pose estimation) we wish to predict the pose of a face image
The output is a continuous number (here a real number describing the face rotation angle)

In both problems the input is a high dimensional vector x representing pixel intensity/color

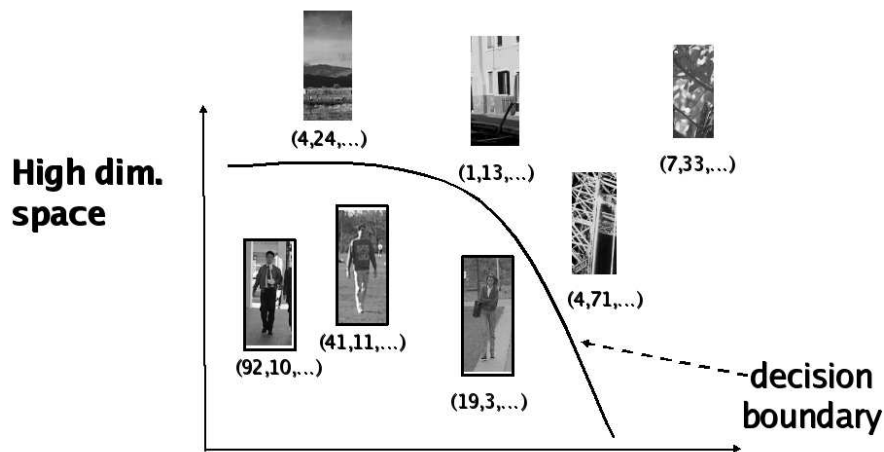
9

People detection example



10

People detection example (cont.)



Data are sparse! Risk for overfitting!

11

Notation

- $\mathcal{X}$: input space (eg, $\mathcal{X} \subseteq \mathbb{R}^d$), with elements $\mathbf{x}, \mathbf{x}', \mathbf{x}_i, \dots$
- $\mathcal{Y}$: output space, with elements $y, y', y_i, \dots$
 - **Classification** (qualitative output) $\mathcal{Y} = \{c_1, \dots, c_k\}$
Binary classification: $k = 2$
 - **Regression** (quantitative output) $\mathcal{Y} \subseteq \mathbb{R}$
- $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$: training set (set of I/O examples)

Supervised learning problem: compute a function which “best describes” I/O relationship

12

Learning algorithm

Training set: $S = \{(\mathbf{x}_i, y_i)_{i=1}^m\} \subseteq \mathcal{X} \times \mathcal{Y}$

A **learning algorithm** is a mapping $S \mapsto f_S$

A new input $\mathbf{x}$ is predicted as $f_S(\mathbf{x})$

In the course we mainly deal with deterministic algorithms but we'll also comment on some randomized ones

Today: we describe two simple learning algorithms:
linear regression and k -nearest neighbors

13

Some important questions

- How is the data **collected**? (need assumptions!)
- How do we **represent** the inputs? (may require preprocessing step)
- How **accurate** is f_S on new data (study of **generalization error**) / How do we **evaluate performance** of the learning algorithm on unseen data?
- How “**complex**” is a learning task? (computational complexity, sample complexity)
- Given two different learning algorithms, f_S and f'_S which one should we choose? (**model selection** problem)

14

Some difficulties/aspects of the learning process

- New inputs **differ** from the ones in the training set (look up tables do not work!)
- Inputs are measured with **noise**
- Output is **not deterministically** obtained by the input
- Input is **high dimensional** but some components/variables may be irrelevant
- Whenever **prior knowledge** is available it should be used

15

More examples / applications

- Optical digit recognition (useful for identifying the numbers in a ZIP code from a digitalized image) (Computer Vision)
- Predicting house prices based on sq. feet, number of rooms, distance from central London,... (Marketing)
- Estimate amount of glucose in the blood of a diabetic person (Medicine)
- Detect spam emails (Information retrieval)
- Predict protein functions / structures (Bioinformatics)
- Speaker identification / sound recognition (Speech recognition)

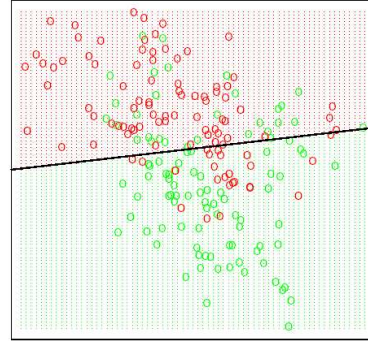
16

Binary classification: an example

We describe two basic learning algorithms/models for classification which can be easily adapted to regression as well. We choose: $\mathcal{X} = \mathbb{R}^2$, $\mathbf{x} = (x_1, x_2)$ and $\mathcal{Y} = \{green, red\}$

Our first learning algorithm computes a **linear function** (perceptron), $\mathbf{w}^\top \mathbf{x} + b$ and classifies an input $\mathbf{x}$ as

$$f(\mathbf{x}) = \begin{cases} \text{red} & \mathbf{w}^\top \mathbf{x} + b > \frac{1}{2} \\ \text{green} & \mathbf{w}^\top \mathbf{x} + b \leq \frac{1}{2} \end{cases}$$



17

Least squares

How do we compute the parameters $\mathbf{w}$ and b ?

We do not lose generality if we set $b = 0$ (we can change the data representation as $\mathbf{x}^\top \rightarrow (\mathbf{x}^\top, 1)$ and $\mathbf{w}^\top \rightarrow (\mathbf{w}^\top, b)$)

We find $\mathbf{w}$ by minimizing the residual sum of squares on the data

$$R(\mathbf{w}) = \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

To compute the minimum we need to solve the system of equations

$$\nabla R(\mathbf{w}) = 0 \quad \text{recall that : } \nabla = \left(\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2} \right)^\top$$

18

Normal equations

$$R(\mathbf{w}) = \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

Note that:

$$\frac{\partial R}{\partial w_k} = 2 \sum_{i=1}^m (\mathbf{w}^\top \mathbf{x}_i - y_i) \frac{\partial (\mathbf{w}^\top \mathbf{x}_i)}{\partial w_k} = 2 \sum_{i=1}^m (\mathbf{w}^\top \mathbf{x}_i - y_i) x_{ik}$$

Hence, to find $\mathbf{w} = (w_1, w_2)^\top$ we need to solve the **linear system** of equations

$$\sum_{i=1}^m (x_{ik} x_{i1} w_1 + x_{ik} x_{i2} w_2) = \sum_{i=1}^m x_{ik} y_i, \quad k = 1, 2$$

19

Normal equations (cont.)

In vector notations:

$$\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w} = \sum_{i=1}^m \mathbf{x}_i y_i$$

In matrix notation:

$$\mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

where

$$\mathbf{X}^\top = \begin{bmatrix} x_{11} & \cdots & x_{m1} \\ \vdots & \ddots & \vdots \\ x_{1d} & \cdots & x_{md} \end{bmatrix} \equiv [\mathbf{x}_1, \cdots, \mathbf{x}_m], \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Note: here $d = 2$ but, clearly, the result holds for any $d \in \mathbb{N}$

20

Least square solution

$$\mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

For the time being we will assume that the matrix $\mathbf{X}^\top \mathbf{X}$ is invertible, so we conclude that

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Otherwise, the solution may not be unique... (we will deal with the case that $\mathbf{X}^\top \mathbf{X}$ is not invertible later)

21

Going back to “b”

Substituting $\mathbf{x}^\top$ by $(\mathbf{x}^\top, 1)$ and $\mathbf{w}^\top$ by $(\mathbf{w}^\top, b)$, the above system of equations can be expressed in matrix form as (exercise):

$$\begin{aligned} (\mathbf{X}^\top \mathbf{X}) \mathbf{w} + \mathbf{X}^\top \mathbf{1} b &= \mathbf{X}^\top \mathbf{y} \\ \mathbf{1}^\top \mathbf{X} \mathbf{w} + m b &= \mathbf{1}^\top \mathbf{y} \end{aligned} \quad \text{that is} \quad \begin{bmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{X}^\top \mathbf{1} \\ \mathbf{1}^\top \mathbf{X} & m \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top \mathbf{y} \\ \mathbf{1}^\top \mathbf{y} \end{bmatrix}$$

where $\mathbf{1} = (1, 1, \dots, 1)^\top$, $m \times 1$ vector of “ones”

22

A different approach: k -nearest neighbors

Let $N(\mathbf{x}; k)$ be the set of k nearest training inputs to $\mathbf{x}$ and $I_{\mathbf{x}} = \{i : \mathbf{x}_i \in N(\mathbf{x}; k)\}$ the corresponding index set

$$f(\mathbf{x}) = \begin{cases} \text{red} & \text{if } \frac{1}{k} \sum_{i \in I_{\mathbf{x}}} y_i > \frac{1}{2} \\ \text{green} & \text{if } \frac{1}{k} \sum_{i \in I_{\mathbf{x}}} y_i \leq \frac{1}{2} \end{cases}$$

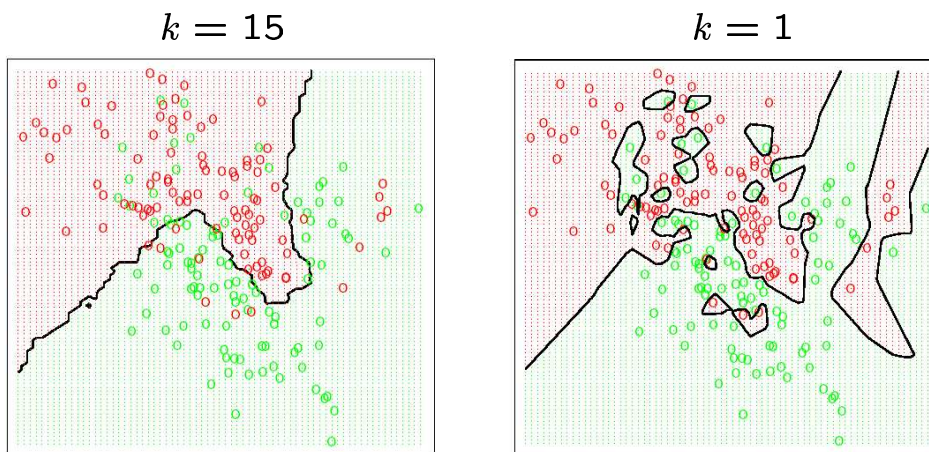
- Closeness is measured using a metric (eg, Euclidean dist.)
- Local rule (compute local majority vote)
- Decision boundary is non-linear

Note: for regression we set $f(\mathbf{x}) = \frac{1}{k} \sum_{i \in I_{\mathbf{x}}} y_i$ (a “local mean”)

23

k -NN: the effect of k

- The smaller k the more irregular the decision boundary



- How to choose k ? later...

24

Linear regression vs. k -NN (informal)

- Global vs. local
- Linear vs. non-linear
- Bias / variance considerations:
 - LR relies heavily on linear assumption (may have large bias) k -NN does not
 - LR is stable (solution does not change much if data are perturbed) 1-NN isn't!
- k -NN sensitive to input dimension d : if d is high, the inputs tends to be far away from each other!

25

Supervised learning model (back to Q1)

We assume that the data is obtained by sampling **i.i.d.** from a **fixed but unknown** probability density $P(\mathbf{x}, y)$

Expected error:

$$\mathcal{E}(f) := \mathbf{E} \left[(y - f(\mathbf{x}))^2 \right] = \int (y - f(\mathbf{x}))^2 dP(\mathbf{x}, y)$$

Our goal is to minimize $\mathcal{E}$

Optimal solution: $f^* := \operatorname{argmin}_f \mathcal{E}(f)$

But: in order to compute f^* we need to know P !

Note: for binary classification with $\mathcal{Y} = \{0, 1\}$ and $f : \mathcal{X} \rightarrow \mathcal{Y}$, $\mathcal{E}(f)$ counts the average number of mistakes of f (aka expected misclassification error)

26

Regression function...

Let us compute the optimal solution f^* for regression ($\mathcal{Y} = \mathbb{R}$) and binary classification ($\mathcal{Y} = \{0, 1\}$)

Using the decomposition $P(y, \mathbf{x}) = P(y|\mathbf{x})P(\mathbf{x})$ we have

$$\mathcal{E}(f) = \int_{\mathcal{X}} \left\{ \int_{\mathcal{Y}} (y - f(\mathbf{x}))^2 dP(y|\mathbf{x}) \right\} dP(\mathbf{x})$$

so we see that

- for regression, f^* (called the regression function) is

$$f^*(\mathbf{x}) = \operatorname{argmin}_{c \in \mathbb{R}} \int_{\mathcal{Y}} (y - c)^2 dP(y|\mathbf{x}) = (\text{exercise}) = \int_{\mathcal{Y}} y dP(y|\mathbf{x})$$

27

... and Bayes classifier

- for binary classification, f^* (called the Bayes classifier) is

$$f^*(\mathbf{x}) = \begin{cases} 1 & \text{if } P(1|\mathbf{x}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

To see this note, as before, that

$$f^*(\mathbf{x}) = \operatorname{argmin}_{c \in \{0,1\}} \left\{ \int_{\mathcal{Y}} (y - c)^2 dP(y|\mathbf{x}) = (1 - c)^2 P(1|\mathbf{x}) + c^2 P(0|\mathbf{x}) \right\}$$

and use

$$P(0|\mathbf{x}) + P(1|\mathbf{x}) = 1$$

28

k -NN revised

k -NN attempts to estimate/approximate $P(1|\mathbf{x})$ as $\frac{1}{k} \sum_{i \in I_{\mathbf{x}}} y_i$

- Expectation is replaced by averaging over sample data
- Conditioning at $\mathbf{x}$ is relaxed to conditioning on some region close to $\mathbf{x}$

One can show that $\mathcal{E}(f_S) \rightarrow \mathcal{E}(f^*)$ as $m \rightarrow \infty$ provided that: $k(m) \rightarrow \infty$ and $\frac{k(m)}{m} \rightarrow 0$

Weakness: the approximation (rate of convergence) depends critically on the input dimension...

29

Least squares revisited

$P(\mathbf{x}, y)$ is unknown $\Rightarrow$ cannot compute $f^* = \operatorname{argmin}_f \mathcal{E}(f)$

We are only given a sample (training set) from P

A natural approach: we approximate the expected error $\mathcal{E}(f)$ by the empirical error

$$\mathcal{E}_{\text{emp}}(f) := \frac{1}{m} \sum_{i=1}^m (y_i - f(\mathbf{x}_i))^2$$

- If we minimize $\mathcal{E}_{\text{emp}}$ over all possible functions, we can always find a function with zero empirical error!

Note: Sometimes the (empirical) error is called (empirical) risk

30

Least squares revisited (cont.)

Proposed solution: we introduce a restrictive class of functions $\mathcal{H}$ called **hypothesis space**

We minimize $\mathcal{E}_{\text{emp}}$ within $\mathcal{H}$. That is, our learning algorithm is:

$$f_S = \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}_{\text{emp}}(f)$$

This approach is usually called **empirical error minimization**

Linear regression: $\mathcal{H} = \{f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$

31

Simplified scenario

Let us discuss the case that $\mathcal{X}$ is a finite set, $\mathcal{X} = \{\mathbf{t}^{(1)}, \dots, \mathbf{t}^{(N)}\} \subset \mathbb{R}^d$, and $\mathcal{Y} = \{0, 1\}$. Moreover, assume that

- $P(\mathbf{x})$ is the uniform distribution
- the I/O relationship is deterministic: $P(y|\mathbf{x}) = \delta(y, f^*(\mathbf{x}))$

$$\mathcal{E}(f) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{y \in \mathcal{Y}} (y - f(\mathbf{x}))^2 P(\mathbf{x}, y) = \frac{1}{N} \sum_{n=1}^N (f^*(\mathbf{t}^{(n)}) - f(\mathbf{t}^{(n)}))^2$$

f^* is the Bayes solution and the Bayes error, $\mathcal{E}(f^*)$, is zero

Note: we may write $\mathbf{x}_i = \mathbf{t}^{(n(i))}$ where $n(i) \in \{1, \dots, N\}$

32

Simplified scenario (cont)

$$\mathcal{E}(f) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{y \in \mathcal{Y}} (y - f(\mathbf{x}))^2 P(\mathbf{t}^{(n)}, y) = \frac{1}{N} \sum_{n=1}^N (f^*(\mathbf{t}^n) - f(\mathbf{x}))^2$$

$$\mathcal{E}_{\text{emp}}(f) = \sum_{i=1}^m (y_i - f(\mathbf{x}_i))^2$$

If hypothesis space $\mathcal{H}$ is large, $\mathcal{E}_{\text{emp}}(f) = 0$ for many classifiers $f \in \mathcal{H}$ but $\mathcal{E}(f)$ may be bounded away from zero! (risk for **overfitting**)

Warning: approximation $\mathcal{E}_{\text{emp}}(f) \approx \mathcal{E}(f)$ may be too optimistic!

33

Summary

- Data S sampled i.i.d from P (fixed but unknown)
- f^* is what we want, f_S is what we get
- Different approaches to attempt to estimate/approximate f^* :
 - Minimize $\mathcal{E}_{\text{emp}}$ in some restricted space of functions (eg, linear)
 - Compute local approximation of f^* (k -NN)
 - Estimate P and then use Bayes rule... (we'll discuss this next week)

34

Perspectives

Theoretical / methodological aspects involved in supervised learning

- Function representation and approximation – to describe $\mathcal{H}$
- Optimization/numerical methods – to compute f_S
- Probabilistic methods – to study generalization error of f_S

35

Additive noise model

Consider the regression problem. Assume that the output is computed as

$$y = f(\mathbf{x}) + \epsilon$$

where ϵ is a zero mean r.v. Hence we can write

$$P(y, \mathbf{x}) = P(y|\mathbf{x})P(\mathbf{x}) = P_\epsilon(y - f(\mathbf{x}))P(\mathbf{x})$$

where $\mathbb{E}[\epsilon] = 0$

Noise free model: y deterministically computed from $\mathbf{x}$ ($\epsilon \equiv 0$)

36

Additive noise model (cont.)

$$y = f(\mathbf{x}) + \epsilon$$

$$P(y, \mathbf{x}) = P(y|\mathbf{x})P(\mathbf{x}) = P_\epsilon(y - f(\mathbf{x}))P(\mathbf{x})$$

The training data is obtained, for $i = 1, \dots, m$ as following

- sample $\mathbf{x}_i$ from $P_{\mathbf{x}}$
- sample ϵ_i from P_ϵ
- set $y_i = f(\mathbf{x}_i) + \epsilon_i$

Note: $P(\mathbf{x}) \equiv P_{\mathbf{x}}(\mathbf{x})$ (just use different notation when needed)

37

Additive noise model (cont.)

$$P(y, \mathbf{x}) = P(y|\mathbf{x})P_{\mathbf{x}}(\mathbf{x}) = P_\epsilon(y - f(\mathbf{x}))P_{\mathbf{x}}(\mathbf{x})$$

So, since ϵ has zero mean, we have that

$$f^*(\mathbf{x}) := \int y dP(y|\mathbf{x}) = \int y dP_\epsilon(y - f(\mathbf{x})) = \int (f(\mathbf{x}) + \epsilon) dP_\epsilon(\epsilon) = f(\mathbf{x})$$

A common choice for the noise distribution P_ϵ is a Gaussian:

$$P_\epsilon(\epsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

38

Maximum likelihood

What is the probability of the data S ?

$$\begin{aligned} P(S) &= \prod_{i=1}^m P(y_i, \mathbf{x}_i) = \prod_{i=1}^m P(y_i | \mathbf{x}_i) P(\mathbf{x}_i) \\ &= \prod_{i=1}^m P(\mathbf{x}_i) \prod_{i=1}^m P(y_i | \mathbf{x}_i) = A \prod_{i=1}^m P(y_i | \mathbf{x}_i) \end{aligned}$$

where

$$A = \prod_{i=1}^m P(\mathbf{x}_i) = P(\mathbf{x}_1, \dots, \mathbf{x}_m)$$

We define the **likelihood** of f as

$$L(f; S) = P(S; f)$$

39

Maximum likelihood (cont.)

Maximum likelihood principle: compute f by maximizing $L(f; S)$

If we use a linear functions and additive Gaussian noise, we have

$$L(\mathbf{w}; S) = A \prod_{i=1}^m (2\pi\sigma^2)^{-\frac{1}{2}} \exp \left\{ -\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2} \right\}$$

In particular the log likelihood is (note: since the log function is strictly increasing maximizing L or $\log L$ is the same)

$$\log L(\mathbf{w}; S) = -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \text{const}$$

Hence maximizing the likelihood is equivalent to least squares!

40

Supervised learning as function approximation

Given the training data $y_i = f^*(\mathbf{x}_i) + \epsilon_i$, the goal is to compute an “approximation” of f^* in some region of $\mathbb{R}^d$.

We look for an approximant of f^* within a prescribed hypothesis space $\mathcal{H}$

- Unless prior knowledge is available on f^* (eg, f^* is linear) we cannot expect $f^* \in \mathcal{H}$
- Choosing $\mathcal{H}$ “very large” leads to overfitting! (we’ll see an example of this in a moment)

41

Polynomial fitting

As an example of hypothesis spaces of increasing “complexity” consider regression in one dimension

$$H_0 = \{f(x) = b : b \in \mathbb{R}\}$$

$$H_1 = \{f(x) = ax + b : a, b \in \mathbb{R}\}$$

$$H_2 = \{f(x) = a_1x + a_2x^2 + b : a_1, a_2, b \in \mathbb{R}\}$$

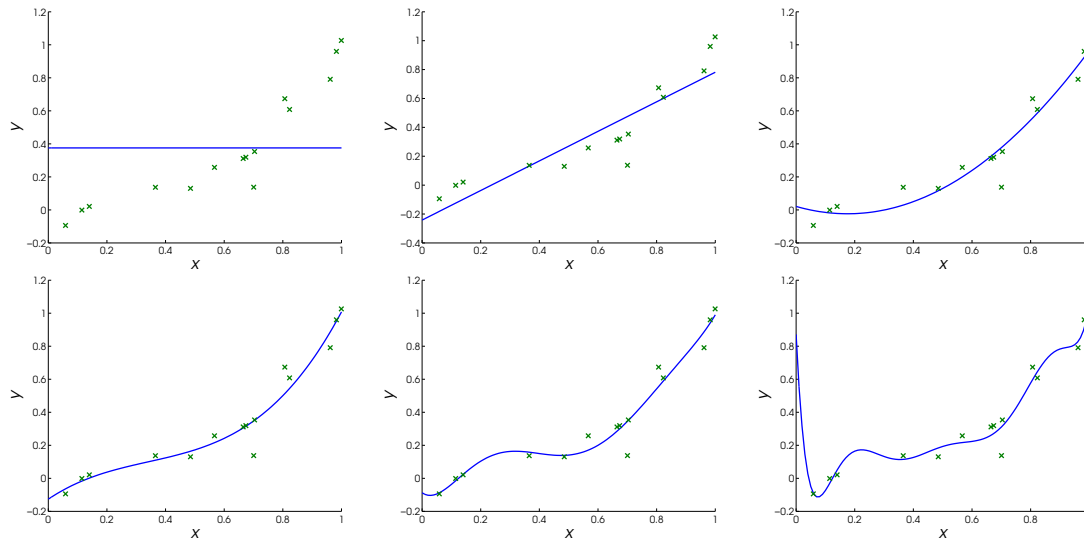
⋮

$$H_n = \left\{ f(x) = \sum_{\ell=1}^n a_\ell x^\ell + b : a_1, \dots, a_n, b \in \mathbb{R} \right\}$$

Let us minimize the empirical error in $\mathcal{H}_r$ (r = polynomial degree)

42

Polynomial fitting (simulation)

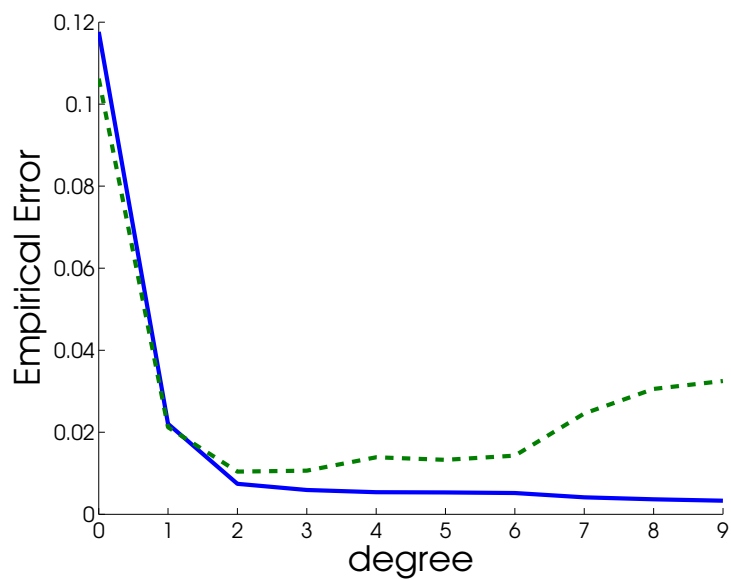


$r = 0, 1, 2, 3, 4, 5$. As r increases the fit to the data improves (empirical error decreases)

43

Overfitting vs. underfitting

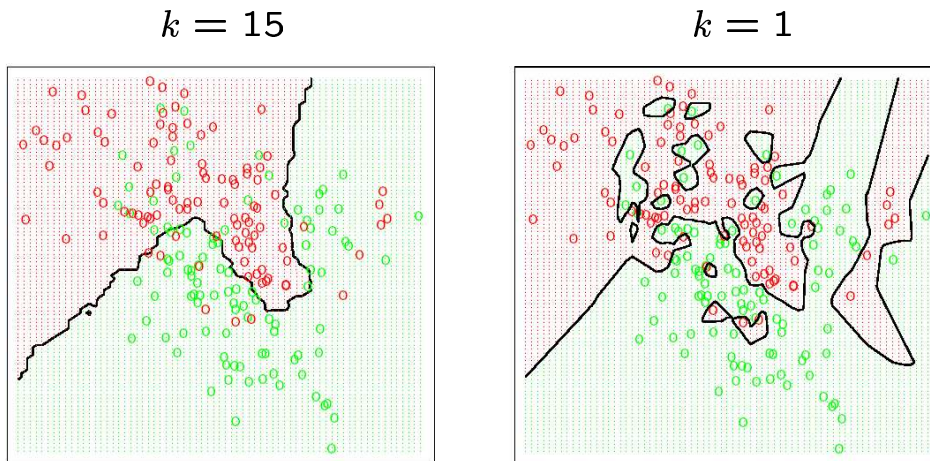
- Compare the empirical error (solid line) with expected error (dashed line)
 - r small: underfitting
 - r large: overfitting
- The larger r the lower the empirical error of f_S !
 $\Rightarrow$ We cannot rely on the training error!



44

k -NN: the effect of k

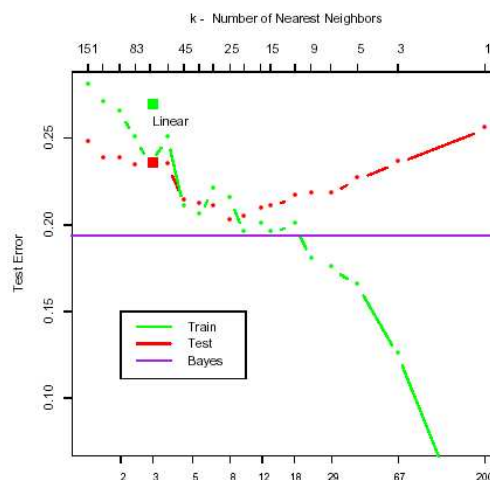
- The smaller k the more irregular the decision boundary and the smaller the empirical error



45

k -NN: the effect of k

- $\frac{m}{k}$ large: overfitting
- $\frac{m}{k}$ small: underfitting



46

Model selection

How to choose k in k -NN?

How to choose the degree r for polynomial regression?

The simplest approach is to use part of the training data (say 2/3) for training and the rest as a validation set

Another approach is K -fold cross-validation: we split the data in K parts (of roughly equal sizes) and repeatedly train on $K - 1$ parts and test on the part “left out”

- We will discuss the model selection problem later in the course

47

Other learning paradigms

Supervised learning is not the only learning setup!

- **Semi-supervised learning:** the ‘learning environment’ may give us access to many input examples but only few of them are labeled
- **Active learning:** we are given many inputs and we can choose which ones to ask the label for
- **Unsupervised learning:** We have only input examples. Here we may want to find data clusters, estimate the probability density of the data, find important features/variable (dimensionality reduction problem), detect anomalies, etc.

48