NN2: Statistical Pattern Recognition

Dr. Wojtek Kowalczyk

wojtek@liacs.nl

Course Overview

- Part One: Classical Neural Networks
  - Basics of statistical pattern recognition
  - Linear models: Perceptron, Adaline, Logistic Regression, Support Vector Machines *(Assignment 1; 15 February)**
  - Multi-layer Perceptron and Backpropagation
  - RBF-networks
- Part Two: Deep Learning
  - Convolutional Networks *(Assignment 2; 8 March)**
  - GPU computing for Deep Learning: *Theano/TensorFlow/Keras*
  - Recurrent and Recursive Networks *(Assignment 3; 29 March)**
  - Autoencoders
  - Restricted Boltzmann Machines & Deep Belief Networks

deadline: 3 weeks later

Statistical Pattern Recognition

1. What is Pattern Recognition?
2. The “a or b?” example
3. Bayes Formula
4. Optimal Decision Boundary
5. Misclassification Costs and Minimizing Risk
6. Classification and Regression
7. Model Complexity and Overfitting; Regularization

What is pattern recognition?

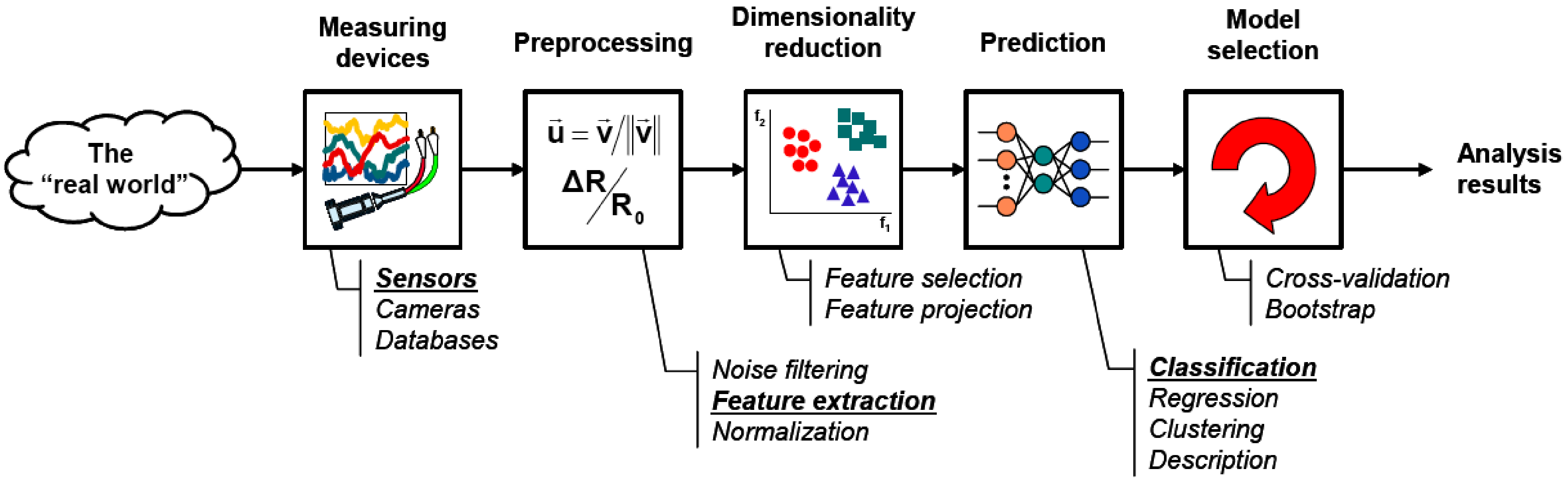
“The assignment of a physical object or event to one of several prespecified categories” -- Duda & Hart

- A **pattern** is an object, process or event that can be given a name.
- A **pattern class** (or category) is a set of patterns sharing common attributes and usually originating from the same source.
- During **recognition** (or **classification**) given objects are assigned to prescribed classes.
- A **classifier** is a machine which performs classification.

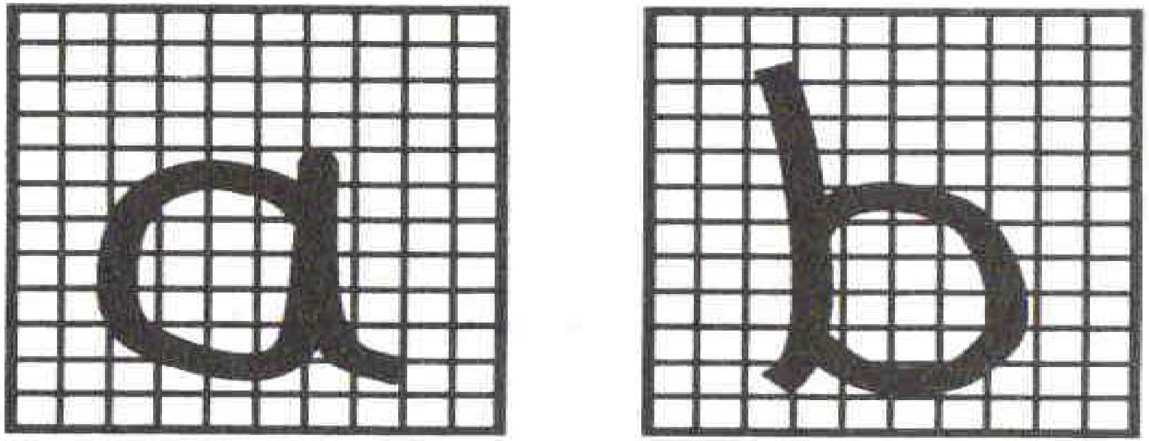
Examples of applications

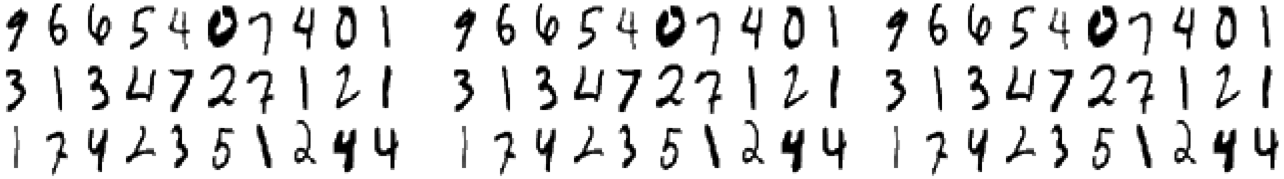
- **Optical Character Recognition (OCR)**
  - Handwritten: sorting letters by postal code, input device for PDA‘s.
  - Printed texts: reading machines for blind people, digitalization of text documents.
- **Biometrics**
  - Face recognition, verification, retrieval.
  - Finger prints recognition.
  - Speech recognition.
- **Diagnostic systems**
  - Medical diagnosis: X-Ray, EKG analysis.
  - Machine diagnostics, waster detection.
- **Military applications**
  - Automated Target Recognition (ATR).
  - Image segmentation and analysis (recognition from aerial or satelite photographs).

Components of a PR System



Example: “a” or “b” ?





Task

- Classify images of handwritten **a**’s (C_1) and **b**’s (C_2)

Let us consider one feature only:

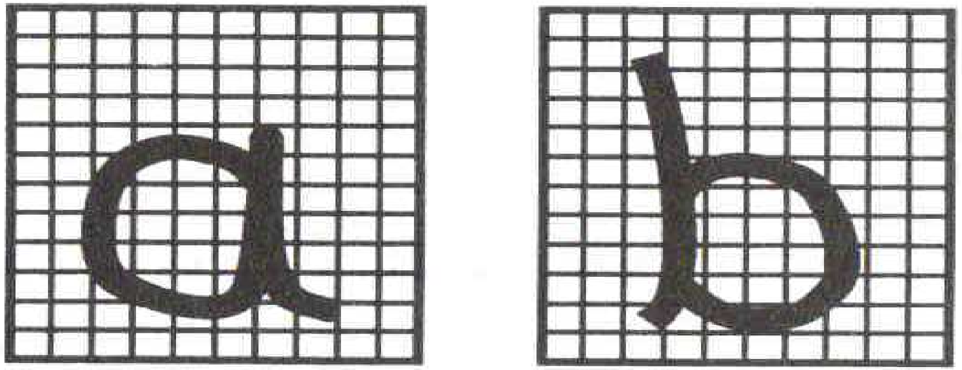
X=height/width

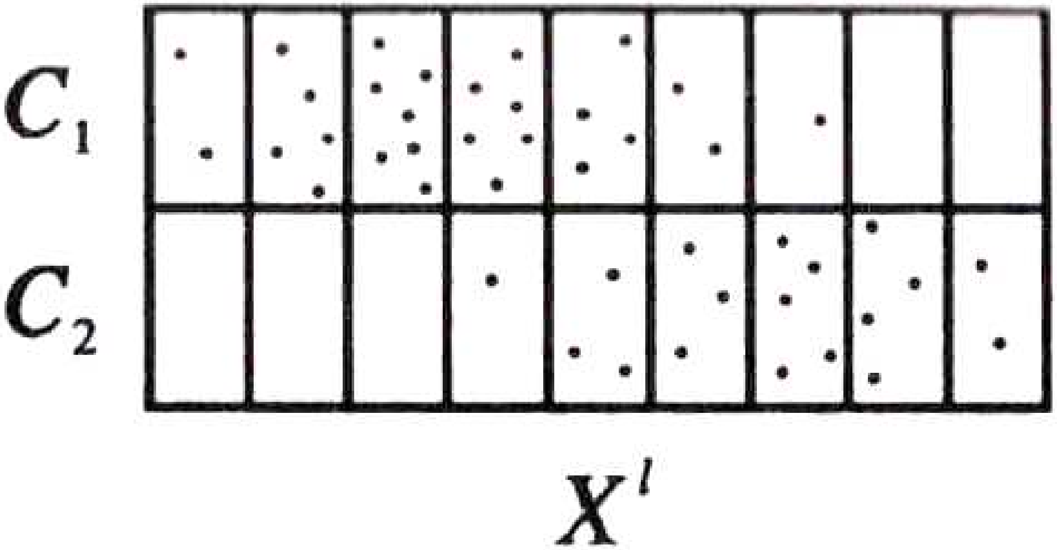
1) How would you decide if you don’t know the value of X?
   - Choose C_1 if $P(C_1) > P(C_2)$:prior probabilities
   - Choose C_2 otherwise

2) How about if you know the value of X?
   – Estimate probabilities $P(C_1|X)$ and $P(C_2|X)$, and take bigger one!

How could we estimate $P(C_1|X)$ and $P(C_2|X)$ from data?

Example

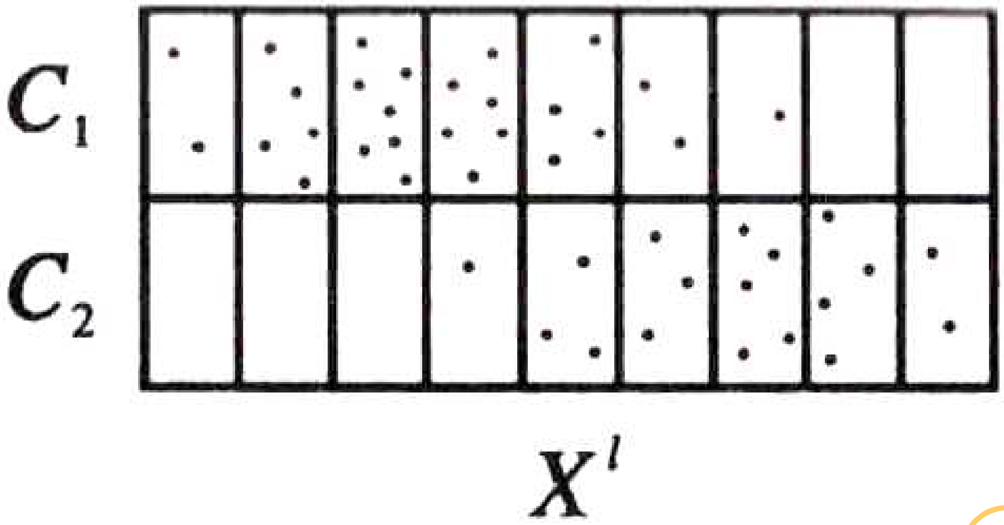




a

b

Bayes' Theorem



multiply

$$P(C_1,X=x) = \frac{\text{num. samples in corresponding box}}{\text{num. all samples}}$$

//joint probability of C_1 and X

$$P(X=x|C_1) = \frac{\text{num. samples in corresponding box}}{\text{num. of samples in } C_1\text{-row}}$$

//class-conditional probability of X

$$P(C_1) = \frac{\text{num. of of samples in } C_1\text{-row}}{\text{num. all samples}}$$

//prior probability of C_1

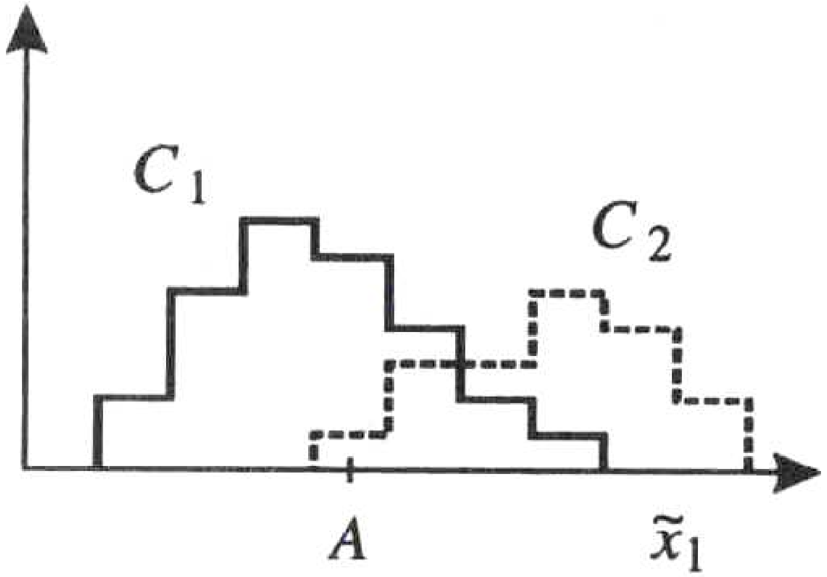
$$P(C1, X=x) = P(X=x|C_1)\ P(C_1)$$

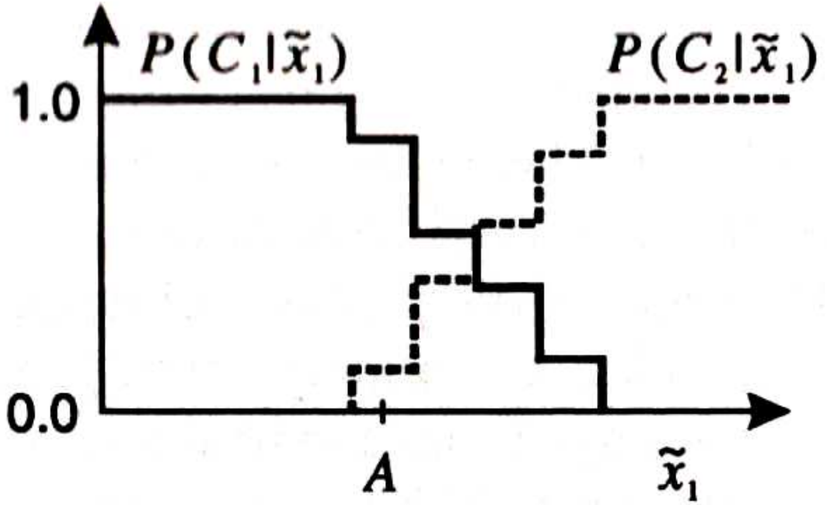
$$P(C1, X=x) = P(C_1|X=x)\ P(X=x)$$

$$P(C1|X=x) = P(X=x|C_1)\ P(C_1)/\ P(X=x)$$

$$\text{posterior} = \frac{\text{likelihood x prior}}{\text{normalization factor}}$$

Histograms





Bayes Theorem

$$P(C_k \mid X) = \frac{P(X \mid C_k)P(C_k)}{P(X)}$$

$\text{scaling factor}:$

$$P(X) = P(X \mid C_1)P(C_1) + P(X \mid C_2)P(C_2)$$

$\text{priors}: P(C_k)$

$\text{class-conditional}: P(X \mid C_k)$

$\text{posteriors}: P(C_k \mid X)$

Posterior Probability Distribution

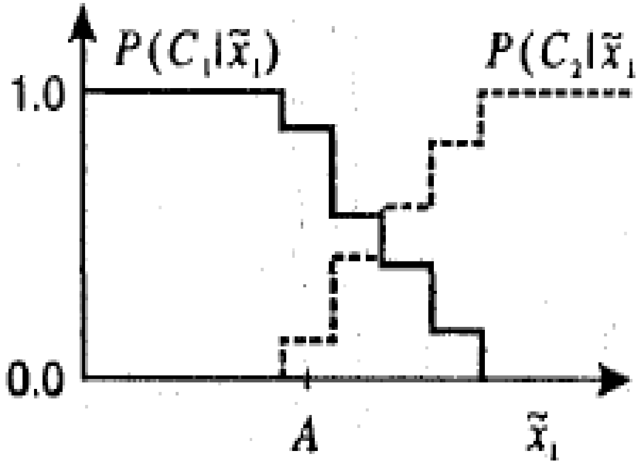


Figure 1.14. Histogram plot of posterior probabilities, corresponding to the histogram of observations in Figure 1.2, for prior probabilities $P(C_1) = 0.6$ and $P(C_2) = 0.4$.

Optimal Decision Boundary

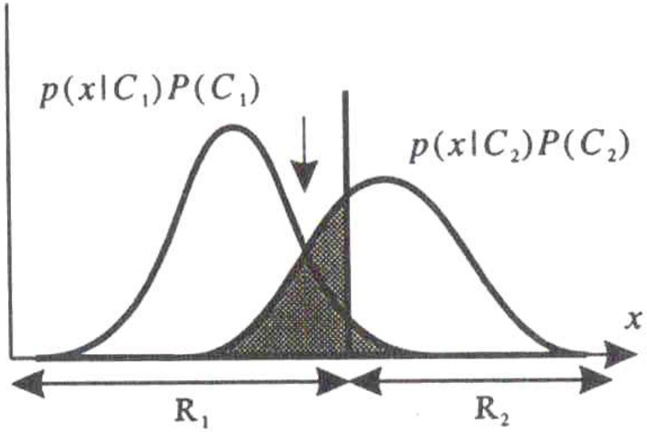


Figure 1.15. Schematic illustration of the joint probability densities, given by $p(x, \mathcal{C}_k) = p(x|\mathcal{C}_k)P(\mathcal{C}_k)$, as a function of a feature value x, for two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. If the vertical line is used as the decision boundary then the classification errors arise from the shaded region. By placing the decision boundary at the point where the two probability density curves cross (shown by the arrow), the probability of misclassification is minimized.

Optimal Decision Boundaries

- In general, assign a pattern x to C_k for k making $P(C_k|x))$ biggest

- In other words, assign a pattern x to C_k if:

$$P(C_k|\mathbf{x}) > P(C_j|\mathbf{x}) \qquad \text{for all } j \neq k.$$

- This generates c decision regions $R_1 \ldots R_c$ such that a point falling in region R_k is assigned to class C_k.

- Note that each of these regions need not be contiguous, but may itself be divided into several disjoint regions all of which are associated with the same class. The boundaries between these regions are known as *decision surfaces* or *decision boundaries*.

Discriminant Functions

- Instead of *posterior probabilities* we could use other *discriminant functions* $y_1(x),\ldots., y_c(x)$ such that an input vector $\mathbf{x}$ is assigned to class C_k if:

  $y_k(x) > y_j(x)$ for all $j \neq k$

- Examples:
  - $y_k(x)=P(C_k|x)$ (posterior probability)
  - $y_k(x)=P(x|C_k)*P(C_k)$ (normalization not needed)
  - $y_k(x)= \ln(P(x|C_k)+\ln P(C_k)$ (monotonic transformation doesn't affect final decisions!!!)

- When there are only two classes (C_1 and C_2) we often use $y(x)=y_1(x)-y_2(x)$ as a discriminant; then the sign of y(x) decides on class:

  if $y(x)>0$ then x in C_1 else x in C_2

Loss Matrix and Risk

- In some situations misclassifying “a” as “b” may cost much more than misclassifying “b” as “a” (think about “fraud” and “non-fraud” or “cancer” and “no-cancer” problems…).

- Usually we use then a Loss Matrix (or Cost Matrix) that quantify these costs. For example:

  $L_{12}=2; L_{21}=1; L_{11}=L_{22}=0,$

  where L_{ij} denotes the cost of “classifying” an object from class i as object from class j (correctly or incorrectly).

- Then, instead of minimizing the number of misclassified cases, we are interested in minimizing the EXPECTED COSTS of misclassified cases.

Minimizing Risk in our Example

Loss Matrix: L_{ab}=2; L_{ba}=1;L_{aa}=L_{bb}=0,

Consider a pattern x. How should we label it, given P(C1|x) and P(C2|x), to minimize expected loss on x?

Each x is either:
“a” with probability P(C1|x), or
“b” with probability P(C2|x), thus:

if we say “a” then the expected loss is: P(C1|x)*L_{aa} + P(C2|x)*L_{ba}
if we say “b” then the expected loss is: P(C1|x)*L_{ab} + P(C2|x)*L_{bb}

TAKE THE CHEAPER ALTERNATIVE!

Continuous valued attributes

$$P(x \in [a,b]) = \int_a^b p(x)\,dx.$$

We use an upper-case P for *probabilities* and
a lower-case p for *probability densities*

For continuous variables, the class-conditional probabilities introduced earlier become class-conditional probability density functions (pdf's), which we write in the form $p(x|C_k)$.

Multiple attributes

- If there are *d* variables/attributes $x_1,\ldots,x_d$, we may group them into a vector $\mathbf{x} = [x_1,\ldots ,x_d]^T$ corresponding to a point in a d-dimensional space.

- The distribution of values of x can be described by probability density function p(**x**), such that the probability of x lying in a region *R of the* x-space is given by

$$P(\mathbf{x} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x})\, d\mathbf{x}.$$

Bayes Theorem

- For continuous variables, the prior probabilities can be combined with the class conditional densities to give the posterior probabilities $P(C_k|x)$ using Bayes theorem:

$$P(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)}.$$

- Note that you can show (and generalize to k classes):

$$p(x) = p(x|C_1)P(C_1) + p(x|C_2)P(C_2).$$

Minimizing Risk in general case

- This risk is minimized if each pattern x is assigned to a class j with the smallest expected loss, i.e., for each another class i, the corresponding expected risk is bigger:

$$\sum_{k=1}^{c} L_{kj}\, p(\mathbf{x}|C_k) P(C_k) < \sum_{k=1}^{c} L_{ki}\, p(\mathbf{x}|C_k) P(C_k) \qquad \text{for all } i \neq j$$

THE OPTIMALITY OF BAESIAN CLASSIFIER:

Once we know prior & class conditional probability distributions we can assign class labels in an **optimal** way, i.e., minimizing “classification error” or “loss”.

Proof not required (check the book).

Key Points

1. Pattern, feature, class, classifier, decision boundary, discriminant function, loss matrix, risk (expected misclassification cost), error (expected misclassification rate), train and test sets, over-fitting, regularization
2. Bayes Formula, prior, posterior, conditional probabilities, likelihood, P (probability) and p (probability density)
3. Optimal Decision Rules; Expected Risk;

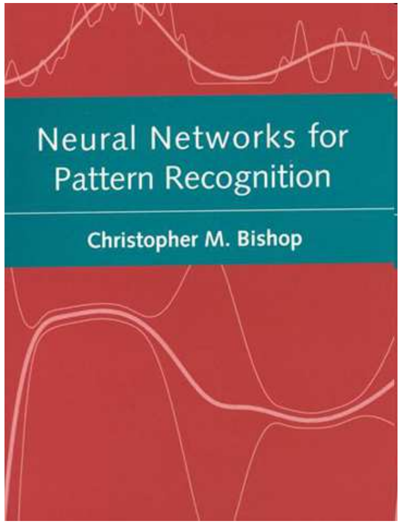
If we could only estimate probabilities we would be done …

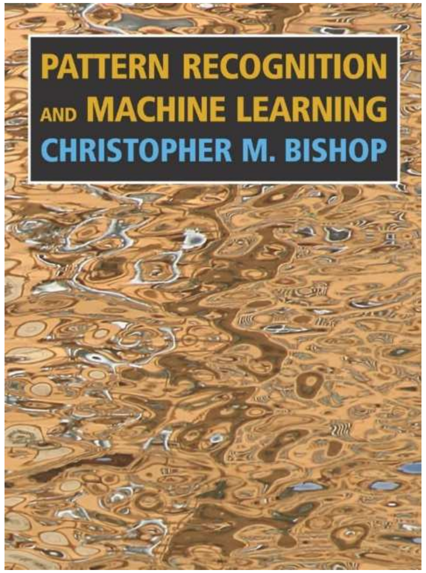
Estimating probabilities => next lecture

If we can’t estimate probs => just find decision boundaries …

Sources

1. The “a” or “b” example comes from C. Bishop’s book: “Neural Networks for Pattern Recognition” (Chapter 1)
2. The example with polynomial approximations comes from C. Bishop’s book: “Pattern Recognition and Machine Learning” (the first 20 pages)





Refresher:

Refresh (?) your memory by studying:

1. Linear Algebra: 2.1 up to 2.5
2. Probability Theory: 3.1 up to 3.9.3
3. Machine Learning Basics: 5.1 up to 5.3

Optionally:

1. Make Assignment 0+ and Assignment 0++ (see next slides)

   (you may submit them to Bas if you want them to be checked).

A0+: Is it an apple or a banana?

Task: design a classifier that can classify **B**ananas and **A**pples!

Input: just one feature - fruit color **- Y**ellow or **G**reen

Moreover, we know that:

the majority of bananas (70%) is yellow,
while apples are usually green (90%).

apples are more frequent than bananas:
only 30% of input patterns represent bananas,
while the remaining 70% represent apples.

Calculate P(B|Y), P(B|G), P(A|Y), P(A|G)

(use Python, so you could play with various scenario’s)

Is it an apple or a banana?

1. How do you label **yellow** fruits: Apple or Banana? And what about **green** fruits?

2. What is the accuracy of your system?

3. Suppose your system was applied to an unusual sample of fruits: 500 bananas and 500 apples. How many errors do you expect?

   Extra: *estimate the 95% confidence interval of the number of errors (binomial distribution).*

4. Suppose that for misclassifying an apple as a banana you pay 5$ while for misclassifying a banana as an apple you pay 15$. What is the expected loss (per one fruit) in the “normal scenario” (30:70) and in the “unusual scenario” (50:50)? Would you change your classifier to minimize the expected losses?

Review of A0

“x’s are distributed uniformly at random (np.random.uniform(n))”

OR

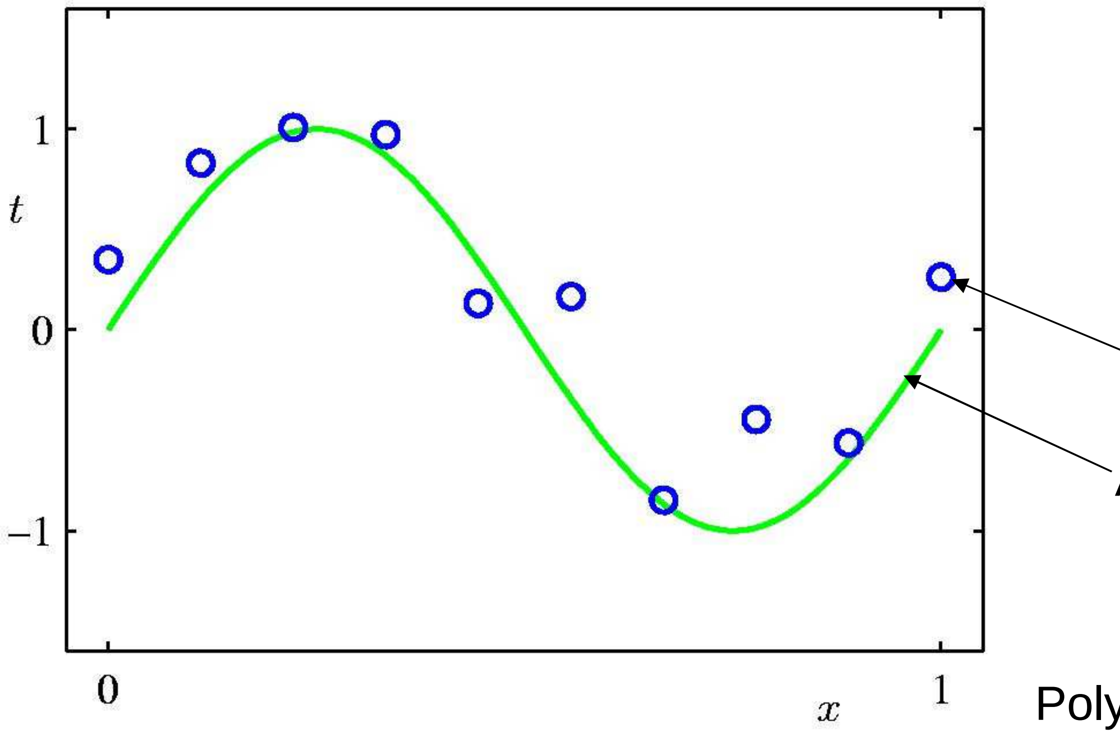
“x’s are distributed uniformly over [0, 1] (np.linspace(0, 1, n)”

The first scenario (randn()) leads to more “noise” in the data;

the second one (linspace()) usually doesn’t have much sense:
the domain of the test set is the same as of the training set!

To see the difference, modify your code replacing “uniform” by “linspace”, and reproduce (several times) all the figures.

Regression: Polynomial Curve Fitting



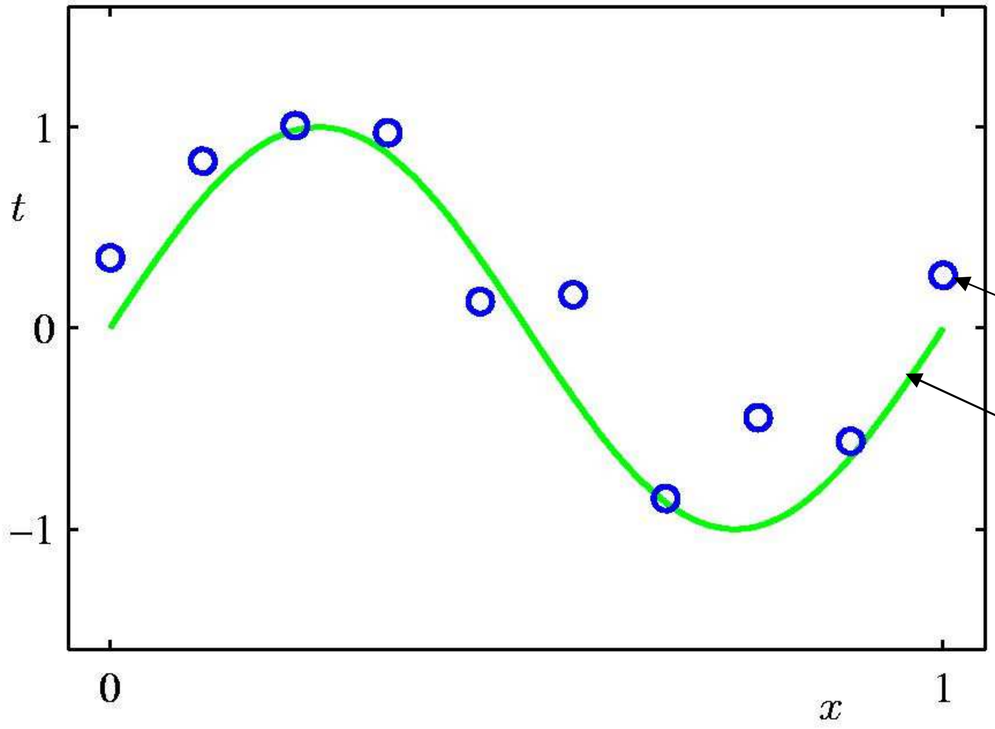
Target, training set

Actual/Target function

Polynomial of degree M

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Regression: Polynomial Curve Fitting



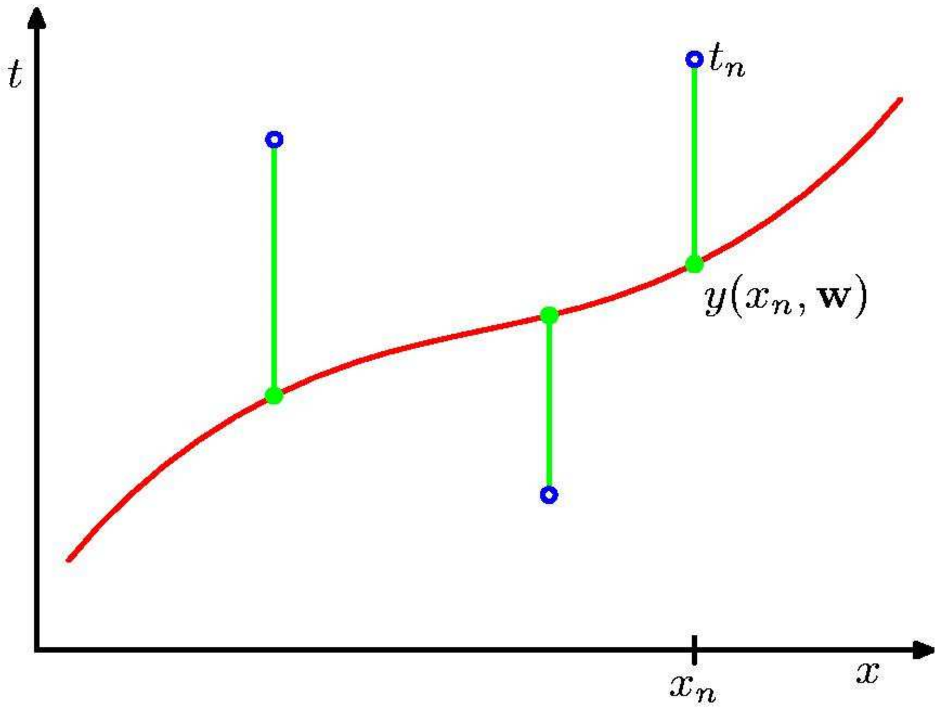
Target, training set

Actual/Target function

Polynomial of degree M

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Through optimization,
we need to find the model (red)
y(x,w) that minimizes the
error function.

E_{SSE}(w): Sum squared error: 2E(w)

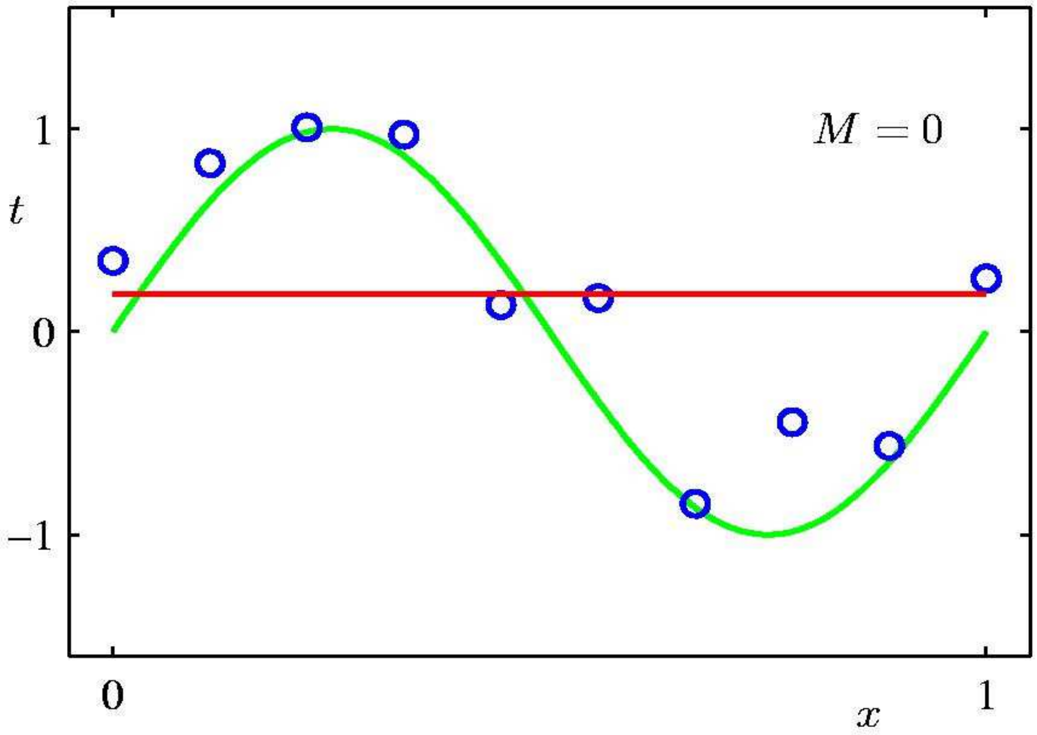
E_{RMS}(w): Root mean squared error

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$$

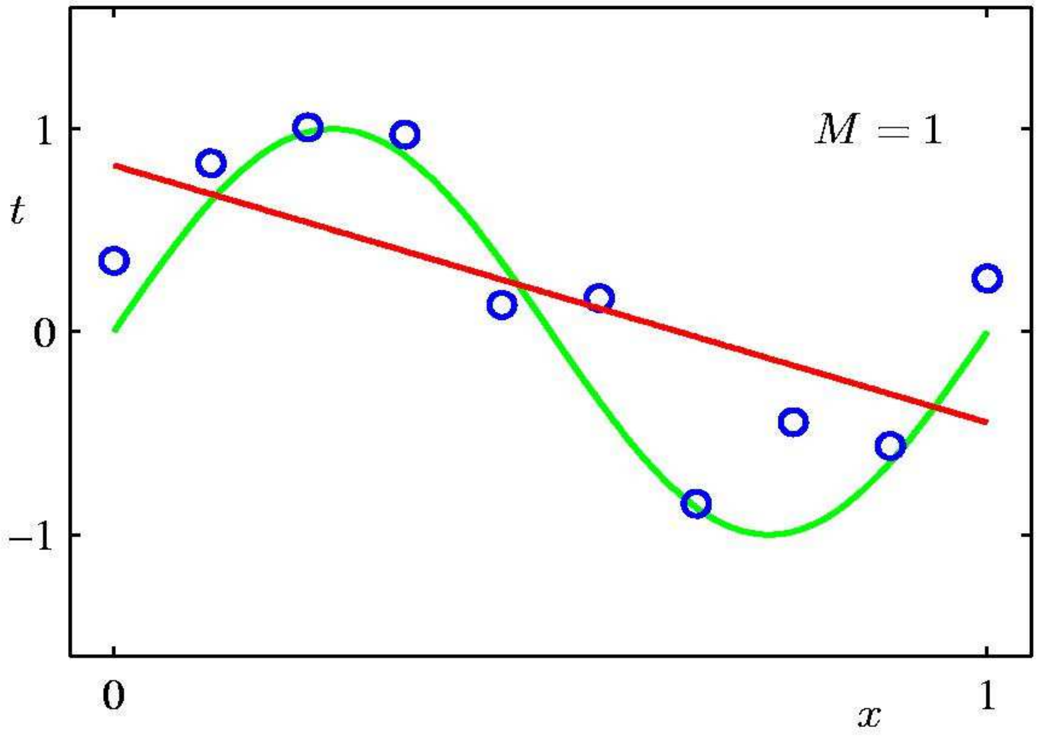
Finding optimal coefficients:
the ***polyfit.m*** function.

|x-y| instead of (x-y)2?
Then error wouldn’t be “smooth”…

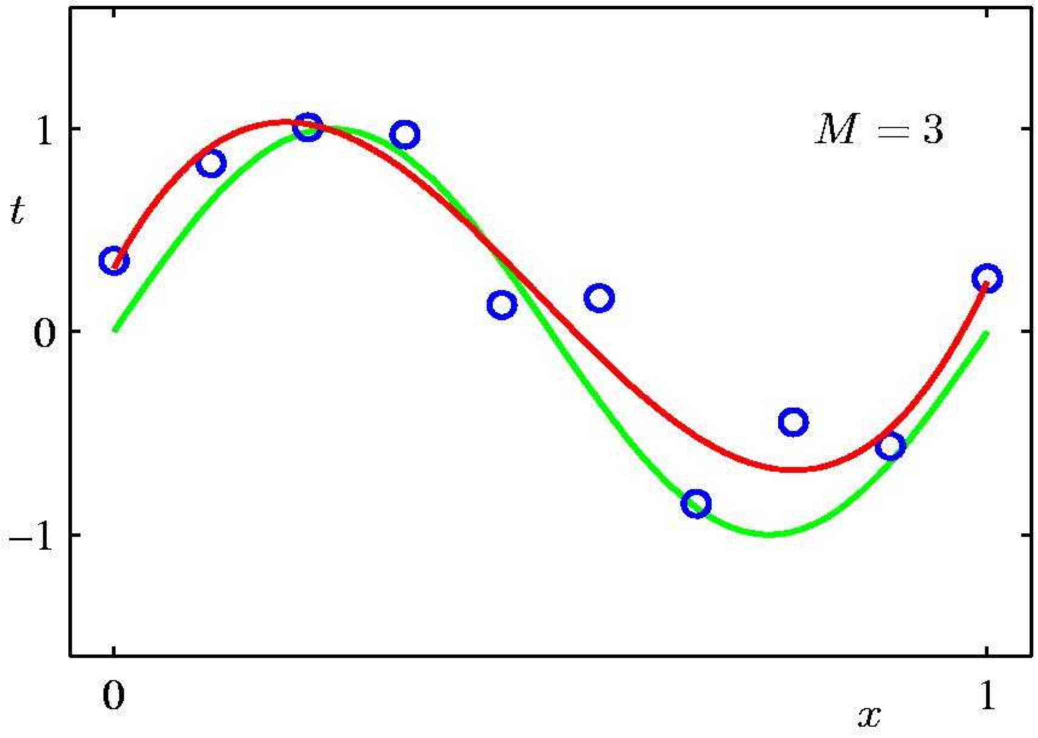
0^{th} Order Polynomial



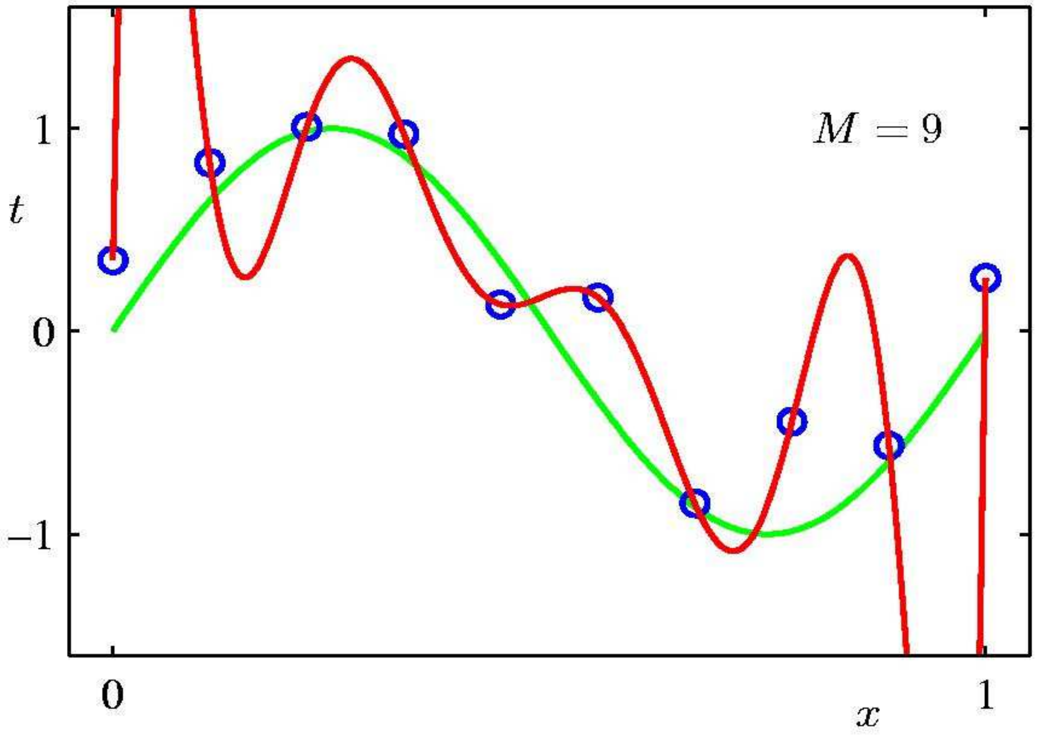
1^{st} Order Polynomial



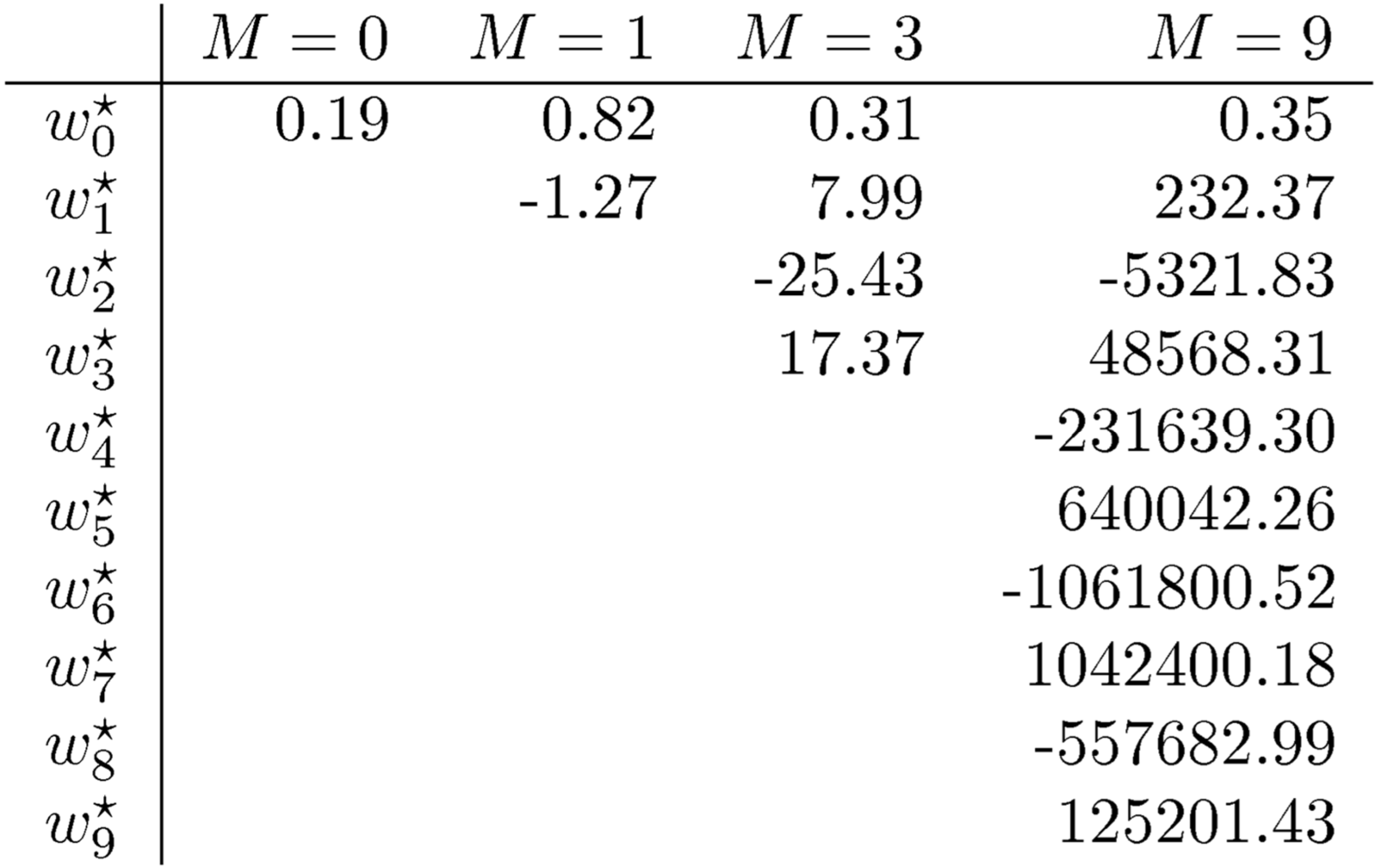
3^{rd} Order Polynomial



9^{th} Order Polynomial (over-fitting)



Polynomial Coefficients



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

REGULARIZATION

Regularization is a powerful technique of limiting overfitting.

The key idea: somehow enforce the absolute values of model parameters to be relatively small (in our case: coefficients of the polynomial).

For example: add to the error function an extra term: "the sum of squared coefficients of your model":

$$\lambda \sum_{i=0}^{M} w_i^2$$

where λ a tunable parameter that controls the size "punishment" for too big values of coefficients.

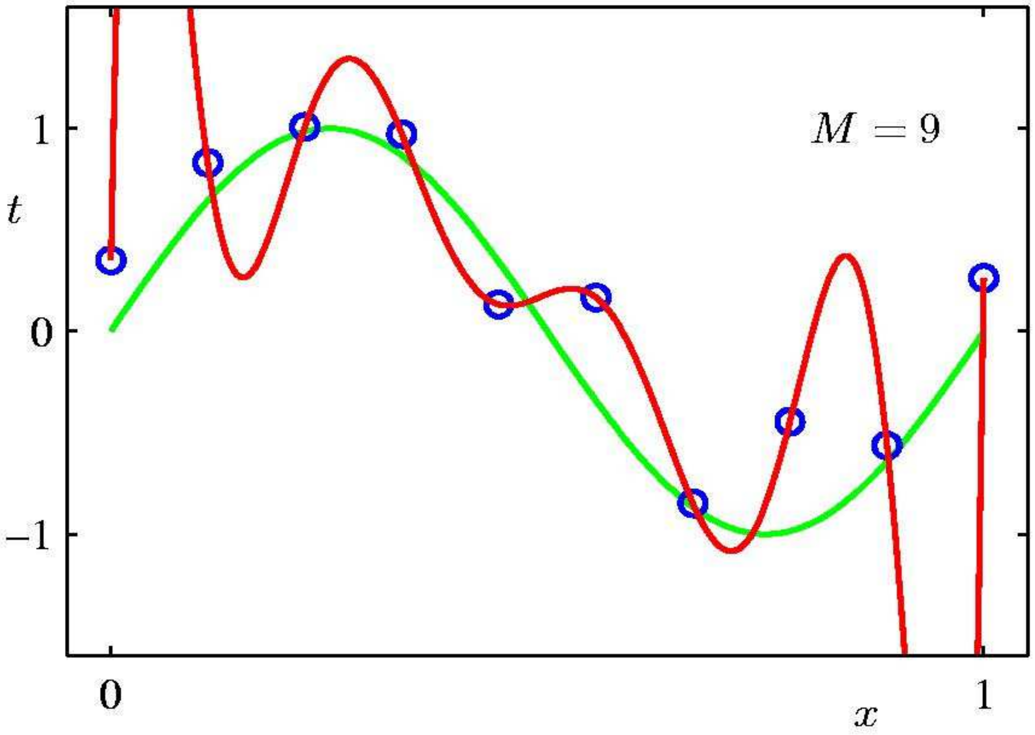
Regularization

- **Penalize large coefficient values**

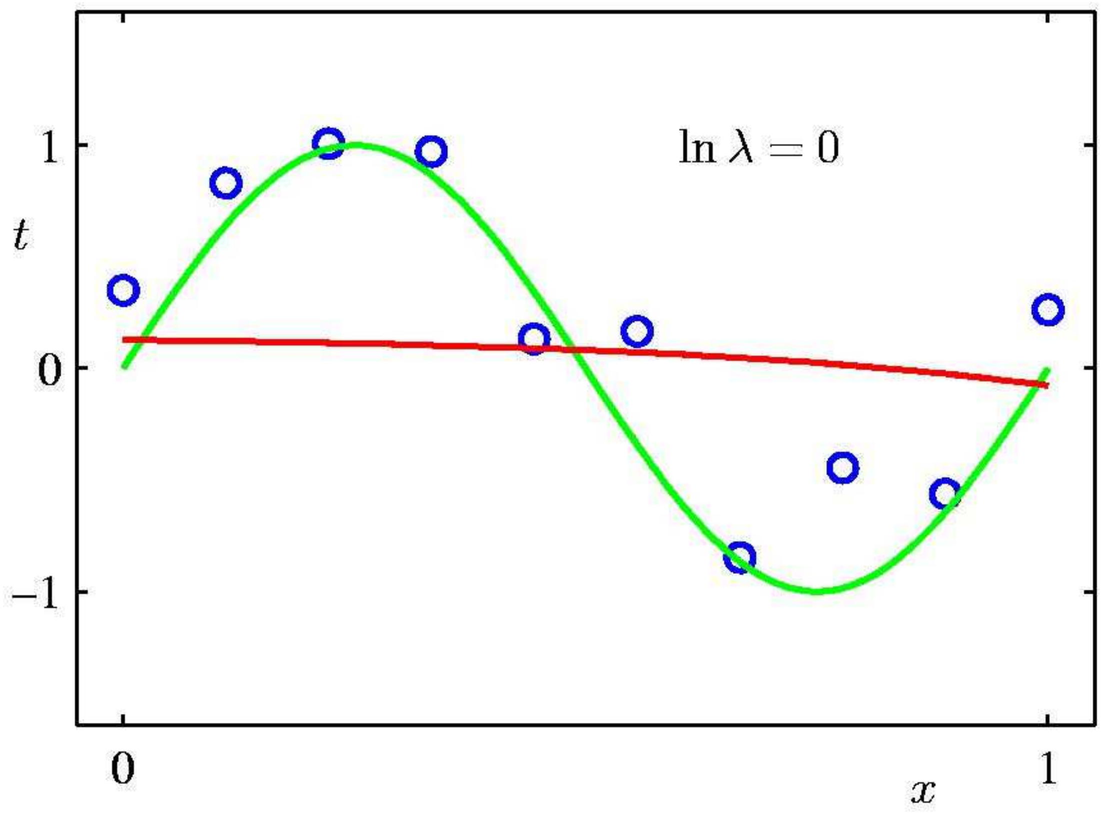
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- **Minimize training error while keeping the weights small. This is known as:**
  - shrinkage,
  - ridge regression,
  - weight decay (neural networks)

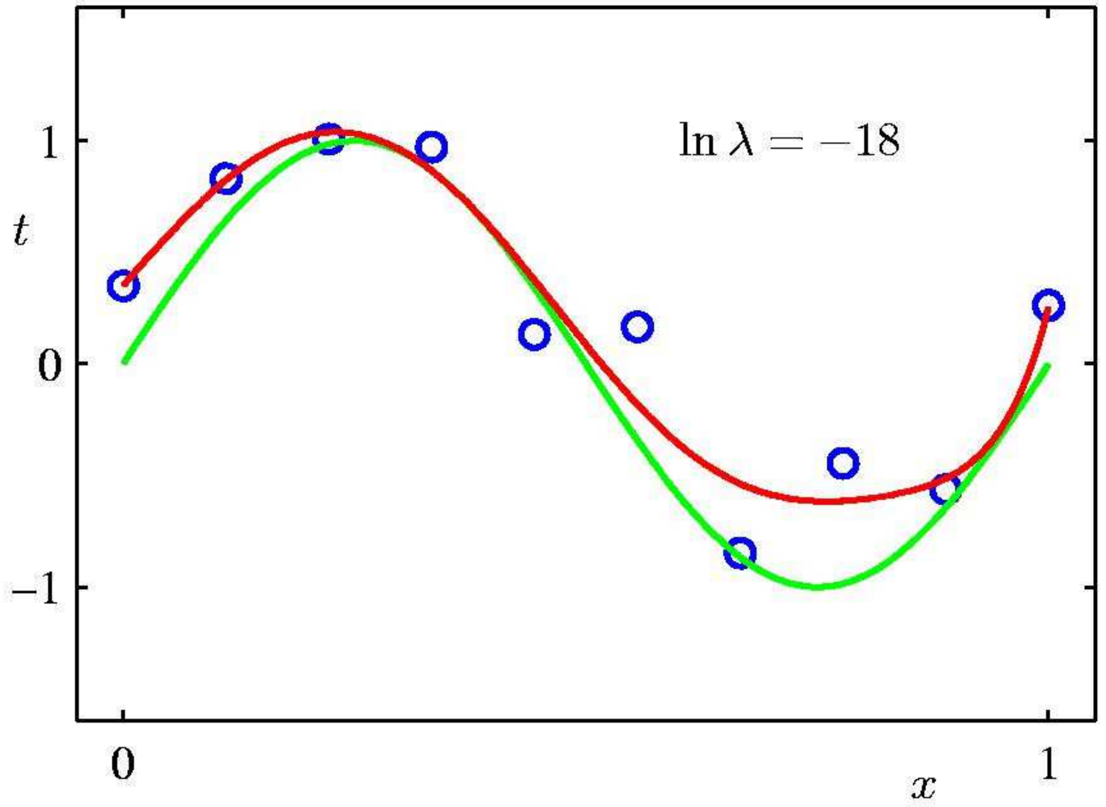
Regularization (M=9; λ=0)



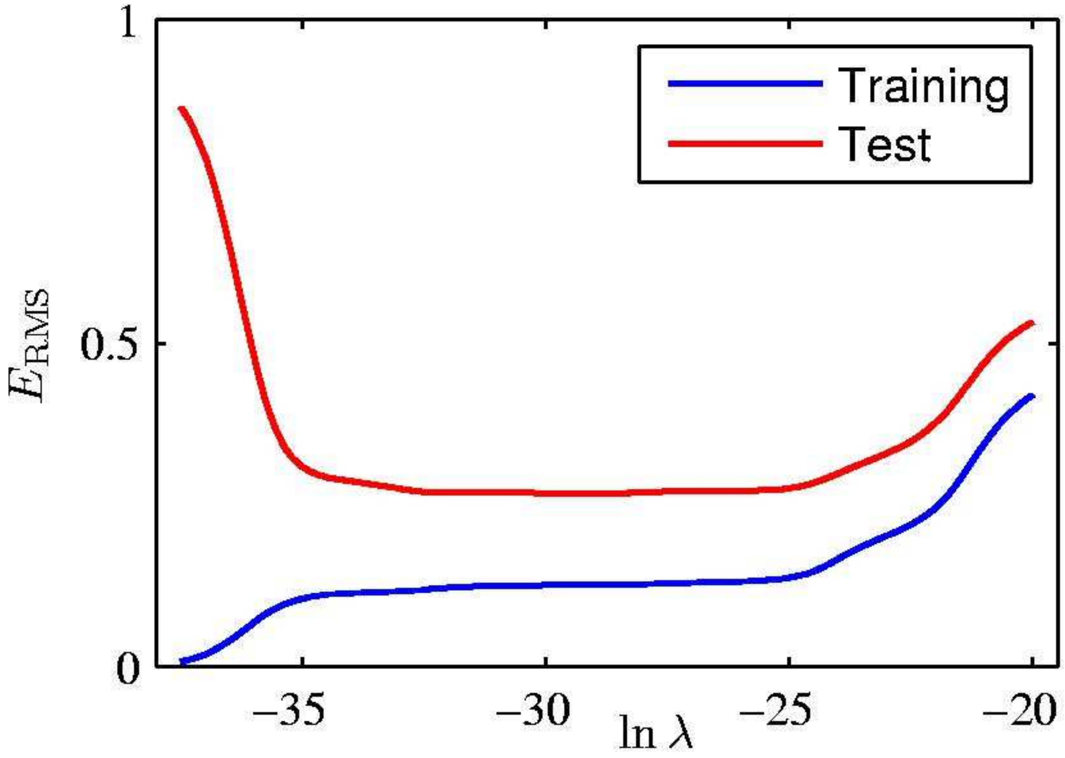
Regularization (d=9; λ=1)



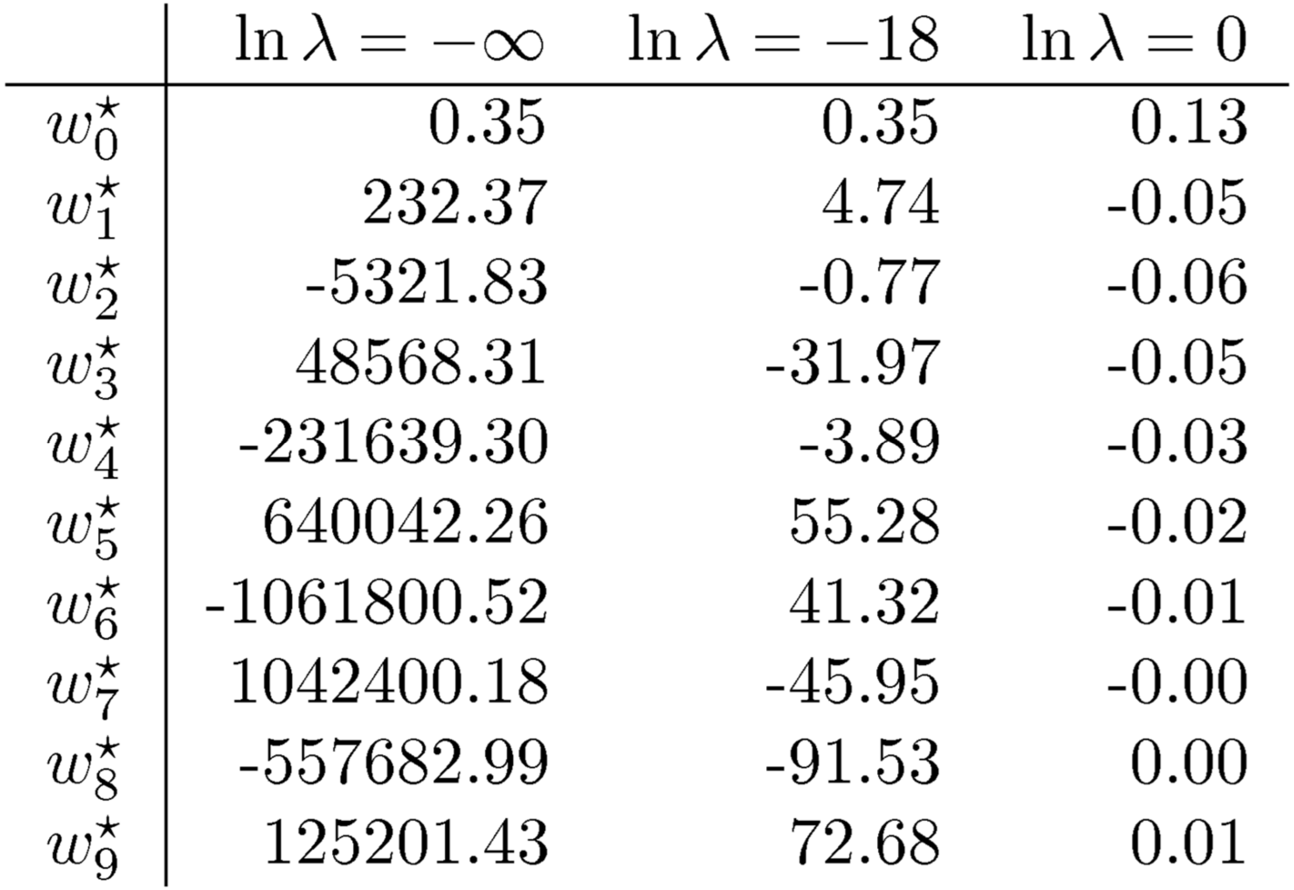
Regularization (M=9; λ=1.5230e-08)



Regularization: error vs. lambda



Polynomial Coefficients



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Assignment 0++

A) Modify your code from Assignment 0 to reproduce the table from slide 36. This time use linspace(0, 1, 10) to produce 10 equally spaced 10 values between 0 and 1. Of course, your values of coefficients will be different than those on slide 36 – y's contain random noise.

B) Think, or search in the literature, how to minimize the error function that includes the regularization term (slide 38).
Hint: the error function is a quadratic function of weights, so its gradient is given by linear functions of weights. Therefore, finding the minimum of the error function reduces to solving a system of linear equations.

C) Implement your optimization procedure and run a number of experiments to reproduce slide 43 (or the plot from slide 42).