Selected Topics of Pervasive Computing

Stephan Sigg

Georg-August-University Goettingen, Computer Networks

06.11.2013

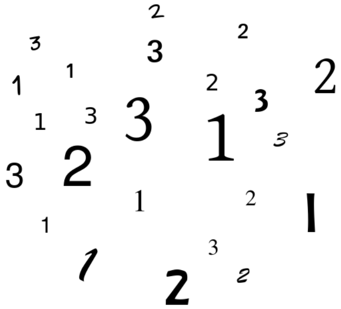
Overview and Structure

30.10.2013	Organisational
30.10.3013	Introduction
06.11.2013	Classification methods (Basic recognition, Bayesian, Non-parametric)
13.11.2013	Classification methods (Linear discriminant, Neural networks)
20.11.2013	–
27.11.2013	–
04.12.2013	–
11.12.2013	Classification methods (Sequential, Stochastic)
18.12.2013	Activity Recognition (Basics, Applications, Algorithms, Metrics)
08.01.2014	Security from noisy data (Basics, Entity, F. Commitment, F. Extractors)
15.01.2014	Security from noisy data (Error correcting codes, PUFs, Applications)
22.01.2014	Context prediction (Algorithms, Applications)
29.01.2014	Networked Objects (Sensors and sensor networks, body area networks)
05.02.2014	Internet of Things (Sensors and Technology, vision and risks)

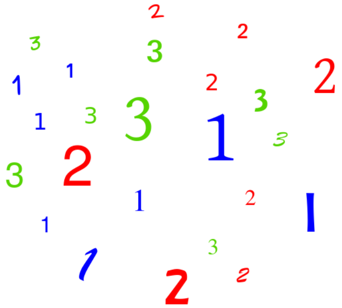
Outline

Pattern recognition vs. Machine learning

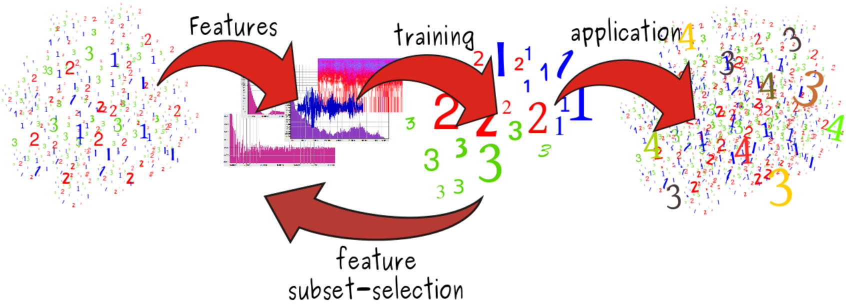
Pattern recognition vs. Machine learning

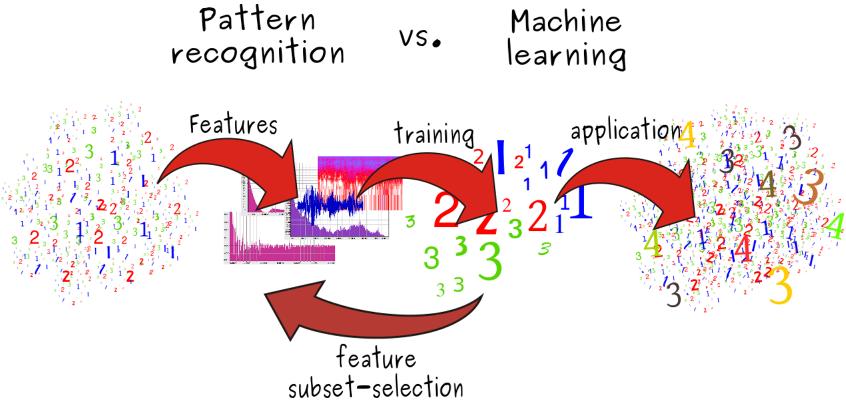


Pattern recognition vs. Machine learning



Pattern recognition vs. Machine learning

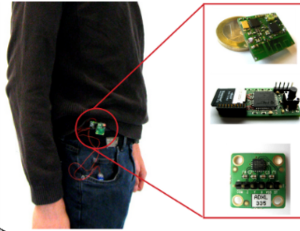




- Mapping of features onto classes by using prior knowledge
- What are characteristic features?
- Which approaches are suitable to obtain these features?

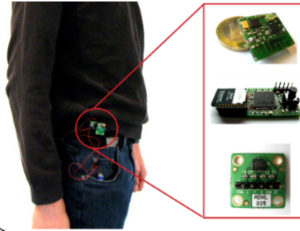
Data sampling

- Record sufficient training data
  - Annotated! (Ground-truth)
  - Multiple subjects
  - Various environmental conditions (time of day, weather, ...)



Data sampling

- Record sufficient training data
  - Annotated! (Ground-truth)
  - Multiple subjects
  - Various environmental conditions (time of day, weather, ...)

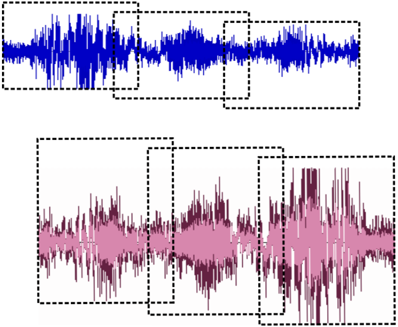


Example

- Electric supply data over 15 years covers 5000 days but only 15 christmas days
- Especially critical events like accidents (e.g. plane, car, earthquake) are scarce

Feature subset-selection

- Pre-process data
  - Framing
  - Normalisation



Feature subset-selection

- Pre-process data
  - Framing
  - Normalisation

Domain knowledge?
-> better set of ad-hoc features

Features commensurate?
-> normalise

Pruning of input required?
-> if no, create disjunctive features or weithted sums of features

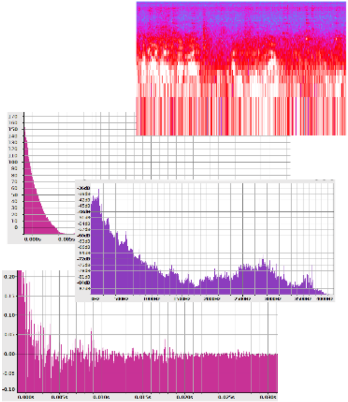
Independent features?
-> construct conjunctive features or products of features

Is the data noisy?
-> detect outlier examples

Do you know what to do first?
-> If not, use a linear predictor

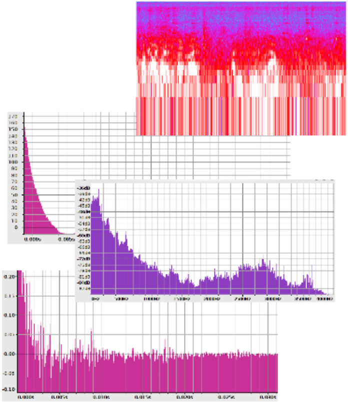
Feature extraction

- Identify meaningful features
  - remove irrelevant/redundant features



Feature extraction

- Identify meaningful features
  - remove irrelevant/redundant features
- Features can be contradictory!



Feature subset-selection

Simple ranking of features with correlation coefficients

Example: Pearson Correlation Coefficient

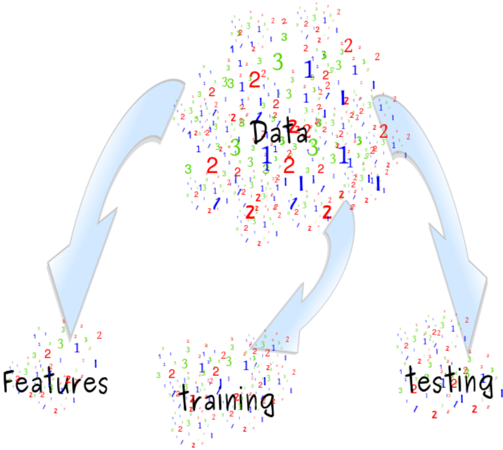
$$\varrho(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} \tag{1}$$

- Identifies linear relation between input variables x_i and an output y

Feature subset-selection

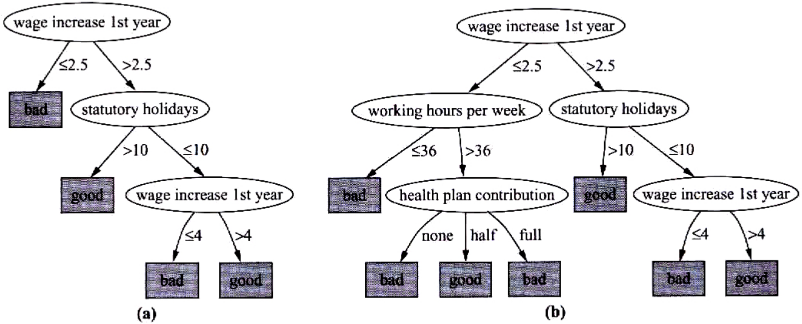
How to do reasonable feature selection

- Utilise dedicated test- and training- data-sets
- Pay attention that a single raw-data sample could not impact features in both these sets
- Don't train the features on the training- or test- data-set



Training of the classifier

A decision tree classifier



Training of the classifier

Evaluation of classification performance

k-fold cross-validation

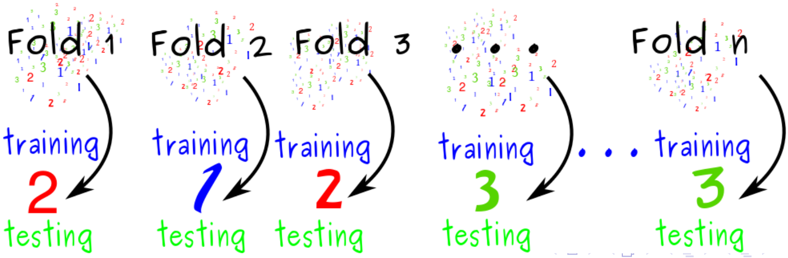
- Standard: k=10

Training of the classifier

Evaluation of classification performance

Leave-one-out cross-validation

- n-fold cross validation where n is the number of instances in the data-set
- Each instance is left out once and the algorithm is trained on the remaining instances
- Performance of left-out instance (success/failure)

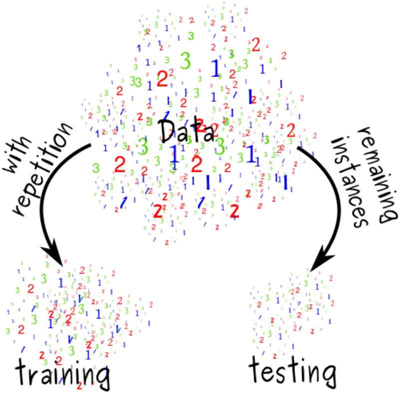


Training of the classifier

Evaluation of classification performance

0.632 Bootstrap

- Form training set by choosing n instances from the data-set with replacement
- All not picked instances are used for testing
- Probability to pick a specific instance:
  $1-\left(1-\frac{1}{n}\right)^{n} \approx 1-e^{-1} \approx 0.632$



Training of the classifier

Evaluation of classification performance

Classification accuracy

- Confusion matrices
- Precision
- Recall

	Classification							
	Aw	No	To	Sb	Sl	Sr	St	Σ
Aw	**52**		3	6	0	17	22	100
No		**436**	25	7	6	17	9	500
To		40	**59**				1	100
Sb	15	22		**32**	4	22	5	100
Sl	12	11	1	6	**48**	8	14	100
Sr	4	15		6	1	**67**	7	100
St	3	18	1	1	24	10	**43**	100
Σ	92	551	86	65	94	129	83	

	Classification							
	Aw	No	To	Sb	Sl	Sr	St	recall
Aw	**.58**	.09		.13	.11	.05	.04	.58
No		**.872**	.05	.014	.012	.034	.018	.872
To		.4	**.59**				.01	.59
Sb	.15	.22		**.32**	.04	.22	.05	.32
Sl	.12	.11	.01	.06	**.48**	.08	.14	.48
Sr	.04	.15		.06	.01	**.67**	.07	.67
St	.03	.18	.01	.01	.24	.1	**.43**	.43
prec	.630	.791	.686	.492	.511	.519	.518	

Training of the classifier

Evaluation of classification performance

Information score

Let C be the correct class of an instance and $\mathcal{P}(C)$, $\mathcal{P}'(C)$ be the prior and posterior probability of a classifier
We define:[1]

$$I_i = \begin{cases} -\log(\mathcal{P}(C)) + \log(\mathcal{P}'(C)) & \text{if } \mathcal{P}'(C) \geq \mathcal{P}(C) \\ -\log(1-\mathcal{P}(C)) + \log(1-\mathcal{P}'(C)) & \text{else} \end{cases} \tag{2}$$

The information score is then

$$\text{IS} = \frac{1}{n}\sum_{i=1}^{n} I_i \tag{3}$$

[1] I. Kononenko and I. Bratko: Information-Based Evaluation Criterion for Classifier's Performance, Machine Learning, 6, 67-80, 1991.

Training of the classifier

Evaluation of classification performance

Brier score

The Brier score is defined as

$$\text{Brier} = \sum_{i=1}^{n} (t(x_i) - p(x_i))^2 \tag{4}$$

where

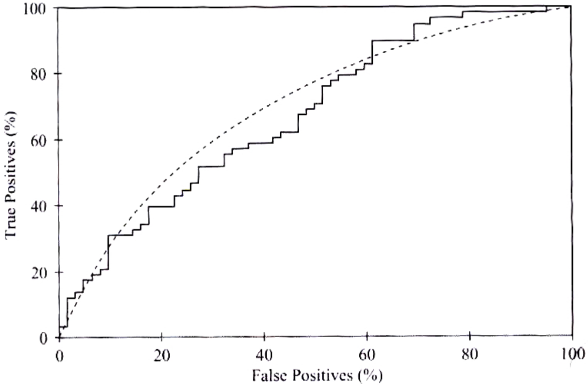
$$t(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is the correct class} \\ 0 & \text{else} \end{cases} \tag{5}$$

and $p(x_i)$ is the probability the classifier assigned to the class x_i.

Training of the classifier

Evaluation of classification performance

Area under the receiver operated characteristic (ROC) curve (AUC)



Rank	Predicted	Actual Class
1	0.95	*yes*
2	0.93	*yes*
3	0.93	*no*
4	0.88	*yes*
5	0.86	*yes*
6	0.85	*yes*
7	0.82	*yes*
8	0.80	*yes*
9	0.80	*no*
10	0.79	*yes*
11	0.77	*no*
12	0.76	*yes*
13	0.73	*yes*
14	0.65	*no*
15	0.63	*yes*
16	0.58	*no*
17	0.56	*yes*
18	0.49	*no*
19	0.48	*yes*
...	...	...

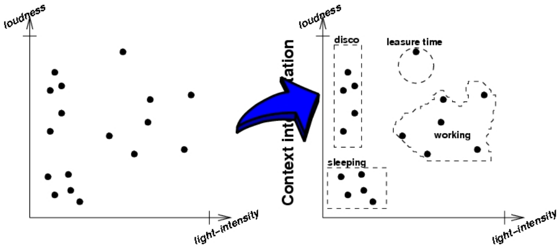
Pattern recognition and classification

Data mining frameworks

- Orange Data Mining (http://orange.biolab.si/)
- Weka Data Mining (http://www.cs.waikato.ac.nz/ml/weka/)

Pattern recognition and classification

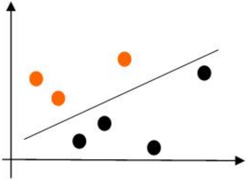
- From features to context
  - Measure available data on features
  - Context reasoning by appropriate method
    - Syntactical (rule based – e.g. RuleML)
    - Bayesian classifier
    - Non-parametric
    - Linear discriminant
    - Neural networks
    - Sequential
    - Stochastic



Pattern recognition and classification

- Allocation of sensor value by defined function
  - Correlation of various data sources
  - Several methods possible – simple approaches
  - Template matching
  - Minimum distance methods
  - 'Integrated' feature extraction
    - Nearest Neighbour
    - Neural Networks
- Problem
  - Measured raw data might not allow to derive all features required
  - Therefore often combination of sensors

Not enough features



Sufficient feature count

Pattern recognition and classification

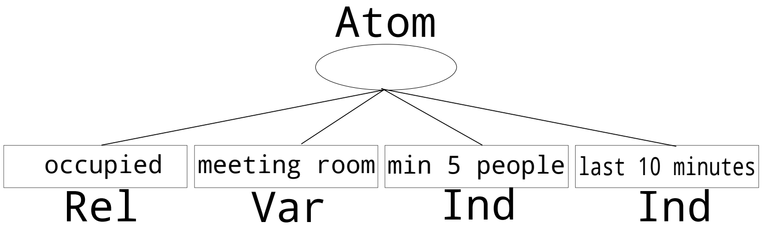
- Methods – Syntactical (Rule based)
  - Idea: Description of Situation by formal Symbols and Rules
  - Description of a (agreed on?) world view
  - Example: RuleML
- Comment
  - Pro:
    - Combination of rules and identification of loops and impossible conditions feasible

    Contra:
    - Very complex with more elaborate situations
    - Extension or merge of rule sets typically not possible without contradictions

Pattern recognition and classification

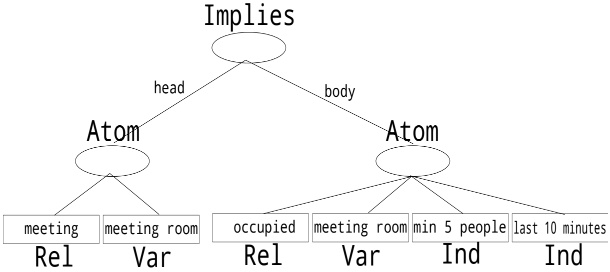
- Rule Markup Language: Language for publishing and sharing rules
- Hierarchy of rule-sub-languages (XML, RDF, XSLT, OWL)
- Example:
  - A meeting room was occupied by min 5 people for the last 10 minutes.

```
<Atom>
     <Rel> occupied </Rel>
     <Var> meeting room </Var>
     <Ind> min 5 persons </Ind>
     <Ind> last 10 minutes </Ind>
</Atom>
```



Pattern recognition and classification

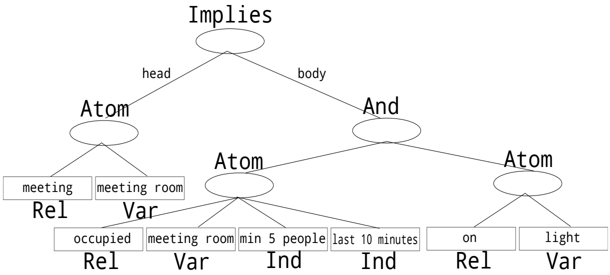
- Also conditions can be modelled
  - A Meeting is taking place in a meeting room when it was occupied by min 5 people for the last 10 minutes.




```
<Implies>
  <head>
     <Atom>
          <Rel> meeting </Rel>
          <Var> meeting room </Var>
     </Atom>
  </head>
  <body>
     <Atom>
          <Rel> occupied </Rel>
          <Var> meeting room </Var>
          <Ind> min 5 persons </Ind>
          <Ind> last 10 minutes </Ind>
     </Atom>
  </body>
</Implies>
```

Pattern recognition and classification

- Logical combination of conditions
  - A Meeting is taking place in a meeting room when it was occupied by min 5 people for the last 10 minutes and the light is on.




```
<Implies>
   <head>
      <Atom>
         <Rel> meeting </Rel>
         <Var> meeting room </Var>
      </Atom>
   </head>
   <body>
      <And>
         <Atom>
            <Rel> on </Rel>
            <Var> light </Var>
         </Atom>
         <Atom>
            <Rel> occupied </Rel>
            <Var> meeting room </Var>
            <Ind> min 5 persons </Ind>
            <Ind> last 10 minutes </Ind>
         </Atom>
      </And>
   </body>
</Implies>
```

Outline

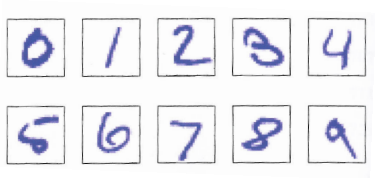
Recognition of patterns

Patterns can be described by a sufficient number of rules

Samples are inaccurate

Tremendous amount of rules to model all variations of one class

Therefore: Consider machine learning approaches



Recognition of patterns

Training set $x_1 \dots x_N$ of a large number of N samples is utilised

Classes $t_1 \dots t_N$ of all samples in this set known in advance

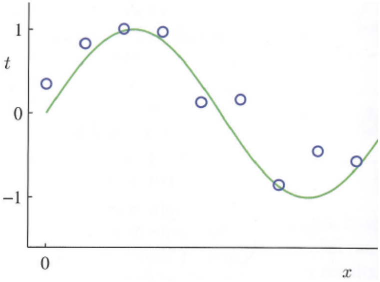
Machine learning algorithm computes a function $y(x)$ and generates a new target t‘

Polynomial curve fitting

Example

A curve shall be approximated by a machine learning approach

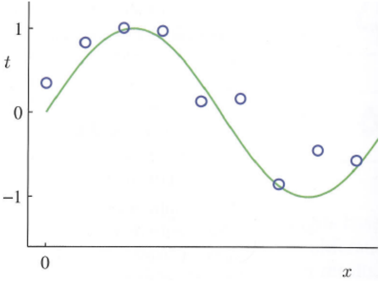
Sample points are created for the function $\sin(2\pi x) + \mathcal{N}$ where $\mathcal{N}$ is a random noise value



Polynomial curve fitting

We will try to fit the data points into a polynomial function:

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



Polynomial curve fitting

We will try to fit the data points into a polynomial function:

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

This can be obtained by minimising an error function that measures the misfit between $y(x, \overrightarrow{w})$ and the training data set:

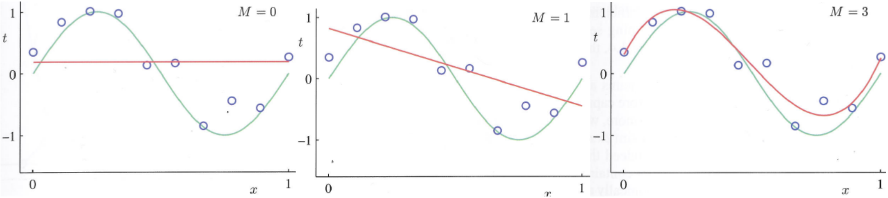
$$E(\overrightarrow{w}) = \frac{1}{2} \sum_{i=1}^{N} \left[y(x_i, \overrightarrow{w}) - t_i \right]^2$$

$E(\overrightarrow{w})$ is non-negative and zero if and only if all points are covered by the function

Polynomial curve fitting

One problem is the right choice of the dimension M

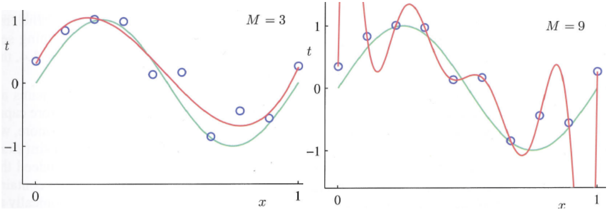
When M is too small, the approximation accuracy might be bad



Polynomial curve fitting

However, when M becomes too big, the resulting polynomial will cross all points exactly

When M reaches the count of samples in the training data set, it is always possible to create a polynomial of order M that contains all values in the data set exactly.



Polynomial curve fitting

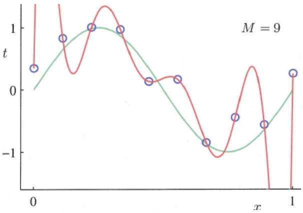
This event is called overfitting

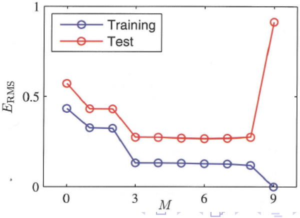
The polynomial now trained too well to the training data

It will therefore perform badly on another sample of test data for the same phenomenon

We visualise it by the Root of the Mean Square (RMS) of $E(\overrightarrow{w})$

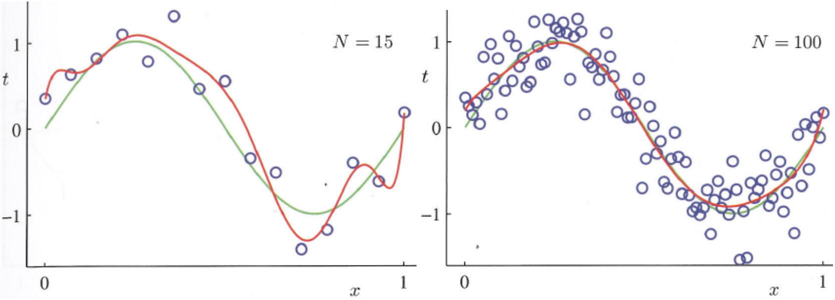
$$E_{RMS} = \sqrt{\frac{2E(\overrightarrow{w})}{N}}$$





Polynomial curve fitting

With increasing number of data points, the problem of overfitting becomes less severe for a given value of M



Polynomial curve fitting

One solution to cope with overfitting is regularisation

A penalty term is added to the error function

This term discourages the coefficients of $\overrightarrow{w}$ from reaching large values

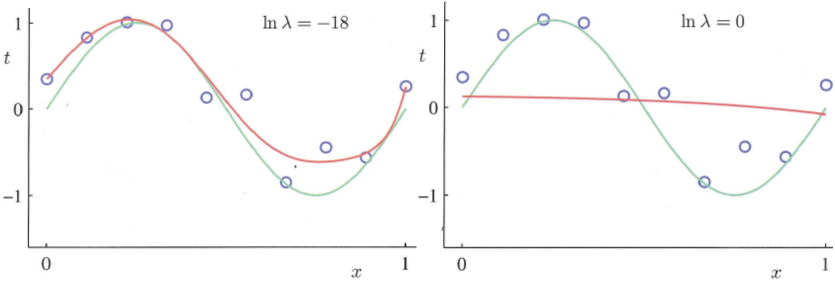
$$\overline{E}(\overrightarrow{w}) = \frac{1}{2}\sum_{i=1}^{N}\left[y(x_i, \overrightarrow{w}) - t_i\right]^2 + \frac{\lambda}{2}||\overrightarrow{w}||^2$$

with

$$||\overrightarrow{w}||^2 = \overrightarrow{w}^T\overrightarrow{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

Polynomial curve fitting

Depending on the value of λ, overfitting is controlled



$$\overline{E}(\overrightarrow{w}) = \frac{1}{2}\sum_{i=1}^{N}\left[y(x_i, \overrightarrow{w}) - t_i\right]^2 + \frac{\lambda}{2}||\overrightarrow{w}||^2$$

Outline

Bayesian decision theory

With probability theory, the probability of events can be estimated by repeatedly generating events and counting their occurrences

When, however, an event only very seldom occurs or is hard to generate, other methods are required

Example:

Probability that the Arctic ice cap will have disappeared by the end of this century

In such cases, we would like to model uncertainty

In fact, it is possible to represent uncertainty by probability

Conditional probability

Conditional probability

The conditional probability of two events χ_1 and χ_2 with $P(\chi_2) > 0$ is denoted by $P(\chi_1|\chi_2)$ and is calculated by

$$P(\chi_1|\chi_2) = \frac{P(\chi_1 \cap \chi_2)}{P(\chi_2)}$$

$P(\chi_1|\chi_2)$ describes the probability that event χ_2 occurs in the presence of event χ_2.

Bayesian decision theory

With the notion of conditional probability we can express the effect of observed data $\overrightarrow{t} = t_1, \ldots, t_N$ on a probability distribution of $\overrightarrow{w}$: $P(\overrightarrow{w})$.

Thomas Bayes described a way to evaluate the uncertainty of $\overrightarrow{w}$ after observing $\overrightarrow{t}$

$$P(\overrightarrow{w}|\overrightarrow{t}) = \frac{P(\overrightarrow{t}|\overrightarrow{w})P(\overrightarrow{w})}{P(\overrightarrow{t})}$$

$P(\overrightarrow{t}|\overrightarrow{w})$ expresses how probable a value for $\overrightarrow{t}$ is given a fixed choice of $\overrightarrow{w}$

Bayesian decision theory

A principle difference between Bayesian viewpoint and frequentist viewpoint is that prior assumptions are provided

Example:

Consider a fair coin that scores heads in three consecutive tosses

Classical maximum likelihood estimate will predict head for future tosses with probability 1

Bayesian approach includes prior assumptions on the probability of events and would result in a less extreme conclusion

Bayesian curve fitting

In the curve fitting problem, we are given $\overrightarrow{x}$ and $\overrightarrow{t}$ together with a new sample x_{M+1}

The task is to find a good estimation of the value t_{M+1}

This means that we want to evaluate the predictive distribution

$$p(t_{M+1}|x_{M+1}, \overrightarrow{x}, \overrightarrow{t})$$

To account for measurement inaccuracies, typically a probability distribution (e.g. Gauss) is underlying the sample vector $\overrightarrow{x}$

Bayesian curve fitting

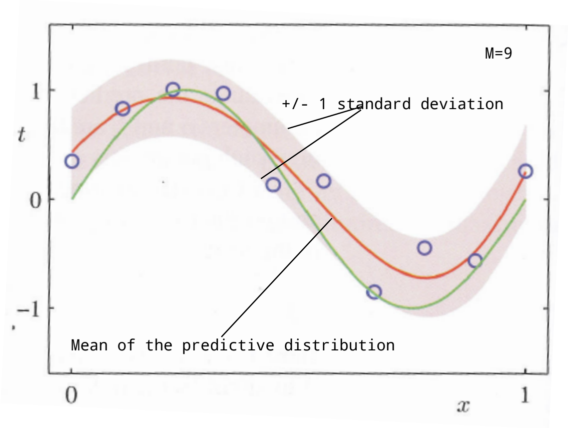
This means that we want to evaluate the predictive distribution

$$p(t_{M+1}|x_{M+1}, \overrightarrow{x}, \overrightarrow{t})$$

After consistent application of the sum and product rules of probability we can rewrite this as

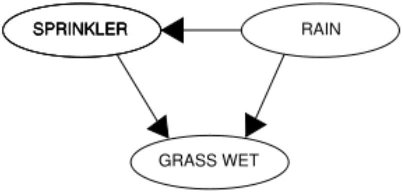
$$p(t_{M+1}|x_{M+1}, \overrightarrow{x}, \overrightarrow{t}) = \int p(t_{M+1}|x_{M+1}, \overrightarrow{w})p(\overrightarrow{w}|\overrightarrow{x}, \overrightarrow{t})d\overrightarrow{w}$$

Bayesian curve fitting



Example

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Outline

Histogram methods

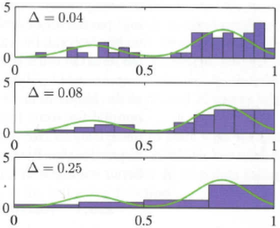
Alternative approach to function estimation: histogram methods

In general, the probability density of an event is estimated by dividing the range of N values into bins of size Δ_i

Then, count the number of observations that fall inside bin Δ_i

This is expressed as a normalised probability density

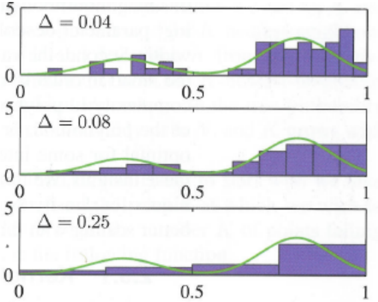
$$p_i = \frac{n_i}{N\Delta_i}$$



Histogram methods

Accuracy of the estimation is dependent on the width of the bins

Approach well suited for big data since the data items can be discarded once the histogram is created

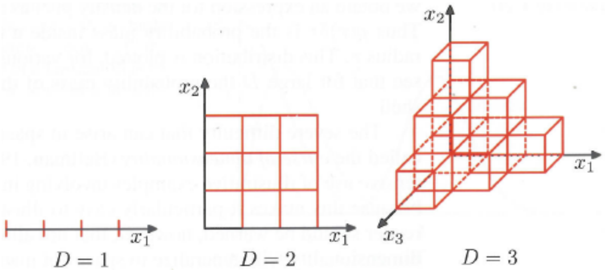


Histogram methods

Issues:

Due to the edges of the bins, the modelled distribution is characterised by discontinuities not present in the underlying distribution observed

The method does not scale well with increasing dimension (Curse of dimensionality)



Parzen estimator methods

Assume an unknown probability density $p(\cdot)$

We want to estimate the probability density $p(\overrightarrow{x})$ of $\overrightarrow{x}$ in a $\mathcal{D}$-dimensional Euclidean space

We consider a small region $\mathcal{R}$ around $\overrightarrow{x}$:

$$P = \int_{\mathcal{R}} p(\overrightarrow{x}) d\overrightarrow{x}$$

Parzen estimator methods

We utilise a data set of N observations

Each observation has a probability of P to fall inside $\mathcal{R}$

With the binomial distribution we can calculate the count K of points falling into $\mathcal{R}$:

$$\mathrm{Bin}(K|N,P) = \frac{N!}{K!(N-K)!}P^K(1-P)^{N-K}$$

Parzen estimator methods

We utilise a data set of N observations

Each observation has a probability of P to fall inside $\mathcal{R}$

With the binomial distribution we can calculate the count K of points falling into $\mathcal{R}$:

$$\mathrm{Bin}(K|N,P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}$$

For large N we can show

$$K \approx NP$$

With sufficiently small $\mathcal{R}$ we can also show for the volume V of $\mathcal{R}$

$$P \approx p(\vec{x})V$$

Therefore, we can estimate the density as

$$p(\vec{x}) = \frac{K}{NV}$$

Parzen estimator methods

We assume that $\mathcal{R}$ is a small hypercube

In order to count the number K of points that fall inside $\mathcal{R}$ we define

$$k(\overrightarrow{u}) = \begin{cases} 1, & |u_i| \leq \frac{1}{2}, \quad i = 1, \ldots, D, \\ 0, & \text{otherwise} \end{cases}$$

This represents a unit cube centred around the origin

This function is an example of a kernel-function or Parzen window

Parzen estimator methods

$$k(\overrightarrow{u}) = \begin{cases} 1, & |u_i| \leq \frac{1}{2}, \quad i = 1, \ldots, D, \\ 0, & \text{otherwise} \end{cases}$$

When the measured data point $\overrightarrow{x_n}$ lies inside a cube of side h centred around $\overrightarrow{x}$, we have

$$k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right) = 1$$

The total count K of points that fall inside this cube is

$$K = \sum_{n=1}^{N} k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

Parzen estimator methods

The total count K of points that fall inside this cube is

$$K = \sum_{n=1}^{N} k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

When we substitute this in the density estimate derived above

$$p(\overrightarrow{x}) = \frac{K}{NV}$$

with volume $V = h^D$ we obtain the overall density estimate as

$$p(\overrightarrow{x}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{h^D}\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

Parzen estimator methods

$$p(\overrightarrow{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

Again, this density estimator suffers from artificial discontinuities

(Due to the fixed boundaries of the cubes)

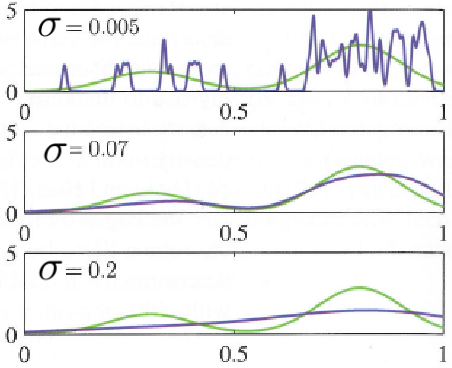
Problem can be overcome by choosing a smoother kernel function

(A common choice is a Gaussian kernel with a standard deviation σ)

$$p(\overrightarrow{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} e^{-\frac{||\overrightarrow{x} - \overrightarrow{x_n}||^2}{2\sigma^2}}$$

Parzen estimator methods

Density estimation for various values of σ

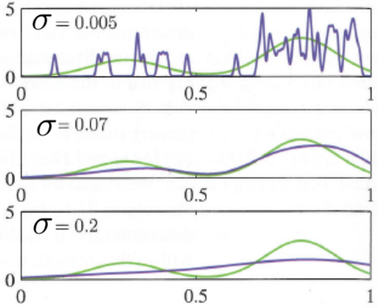


Nearest neighbour methods

A problem with Parzen estimator methods is that the parameter governing the kernel width (h or σ) is fixed for all values $\vec{x}$

In regions with

...high density, a wide kernel might lead to over-smoothing
...low density, the same width may lead to noisy estimates



Nearest neighbour methods

NN-methods address this by adapting width to data density

Parzen estimator methods fix V and determine K from the data
Nearest neighbour methods fix K and choose V accordingly

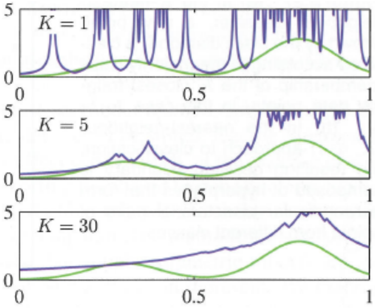
Again, we consider a point $\overrightarrow{x}$ and estimate the density $p(\overrightarrow{x})$

The radius of the sphere is increased until K data points (the nearest neighbours) are covered

Nearest neighbour methods

The value K then controls the amount of smoothing

Again, an optimum value for K exists



Nearest neighbour methods

Classification: Apply KNN-density estimation for each class

Then, utilise Bayes theorem

Assume data set of N points with N_k points in class C_k

To classify sample $\overrightarrow{x}$, draw a sphere containing K points around $\overrightarrow{x}$

Sphere can contain other points regardless of their class

Assume sphere has volume V and contains K_k points from C_k

Nearest neighbour methods

Assume: Sphere of volume V contains K_k points from class C_k

We estimate the density of class C_k as

$$p(\overrightarrow{x}|C_k) = \frac{K_k}{N_k V}$$

The unconditional density is given as

$$p(\overrightarrow{x}) = \frac{K}{NV}$$

The probability to experience a class C_k is given as

$$p(C_k) = \frac{N_k}{N}$$

With Bayes theorem we can combine this to achieve

$$p(C_k|\overrightarrow{x}) = \frac{p(\overrightarrow{x}|C_k)p(C_k)}{p(\overrightarrow{x})} = \frac{K_k}{K}$$

Nearest neighbour methods

$$p(C_k|\overrightarrow{x}) = \frac{p(\overrightarrow{x}|C_k)p(C_k)}{p(\overrightarrow{x})} = \frac{K_k}{K}$$

To minimise the probability of misclassification, assign $\overrightarrow{x}$ to class with the largest probability

This corresponds to the largest value of

$$\frac{K_k}{K}$$

Nearest neighbour methods

To classify a point, we identify the K nearest points

And assign the point to the class having most representatives in this set

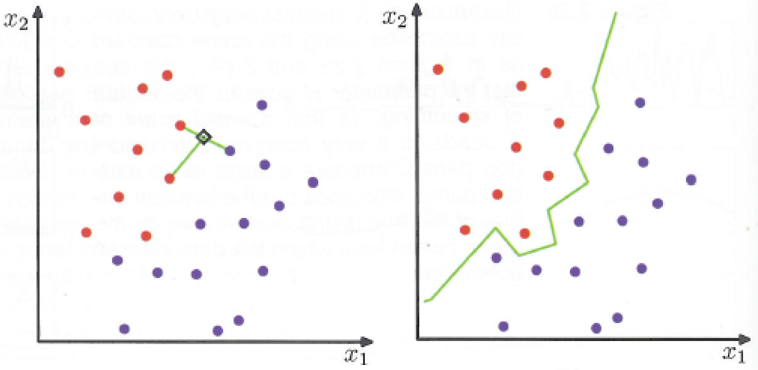
Choice $K = 1$ is called nearest neighbour rule

For this choice, the error rate is never more than twice the minimum achievable error rate of an optimum classifier[2]

[2] T. Cover and P. Hart: Nearest neighbour pattern classification. IEEE Transactions on Information Theory, IT-11, 21-27, 1967

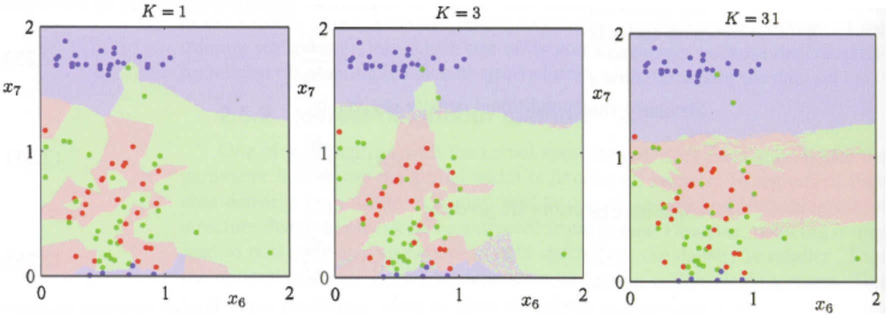
Nearest neighbour methods

Classification of points by the K-nearest neighbour classifier



Nearest neighbour methods

Classification of points by the K-nearest neighbour classifier



Nearest neighbour methods

The KNN-method and the Parzen-method are not well suited for large data sets since they require the entire data set to be stored

Outline