# Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Vienna Science Chair of Bioinformatics
Department of Biotechnology
BOKU University
*peter.sykacek@boku.ac.at*

November 20, 2012





## Expectations of/about this lecture

- Provides a “dinner party talk” introduction to machine learning and an overview of relevant skills.
- Some equations will appear one a few slides for illustrating concepts. Equations are not to be learned by heart and will not be examined!
- The lecture is interactive. My expectation is that we work actively together through theses slides.
- The take home knowledge is in addition documented in a set of questions and prototypical answers. These questions are the only qestions about my lecture you might get asked during the exam.
- All material will be available at `http://www.sykacek.net/teaching.html` (and in moodle).
- Taking notes is a waste of your time here!

# Overview



- Machine learning: data analysis with computers
- What you should know about programming
- A basic understanding of probability
- Concepts in data analysis
- Supervised learning
- Unsupervised learning
- Model fitting and problems associated with it
- Further elective courses on data analysis



# Machine Learning (ML)

Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Introduction

Computer Programming

Probability Theory

Data

Data + Model

Supervised methods

Unsupervised methods

Model Fitting

ML is about implementing and applying computer programs to extract knowledge from data. Knowledge refers to:

- Models which explain data.
- Data summaries of reduced complexity.
- Answers about relations in data (e.g. genes X and Y are involved in process Z).

# Machine Learning: Tasks and Required Skills



Tasks:

- Data extraction and formatting (e.g. extract background information from databases and reformat measurements appropriately.)
- Apply and implement data analysis methods.

Skills:

- One has to learn to program a computer.
- One has to understand concepts in probability theory and statistics.
- Both require a mathematical background, a logical mindset and accuracy.



# Computer Programming



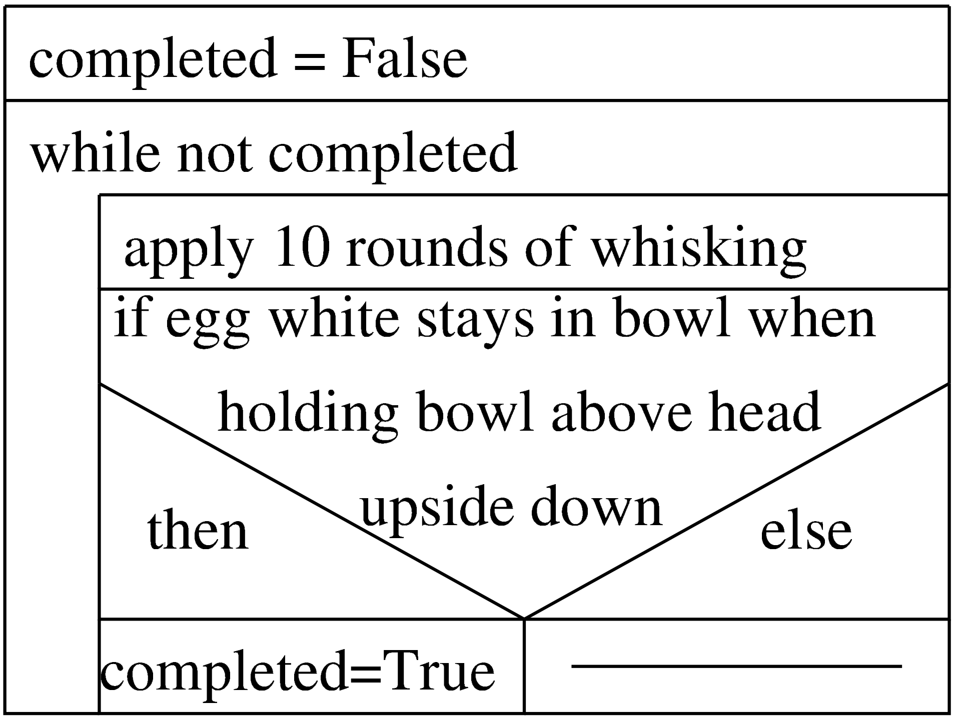
- Computer programs are instructions to convert given “input data” to desired “output data”.
- Similarity with many real world concepts (like cooking a recipe).

Example: produce beaten egg white
Ingredients (input): 1 chicken egg

1. Separate egg white from yolk.
2. Use wire whisk to beat egg white until stiff.
3. Test of success: turn over mixing bowl and look from below into the bowl.

Result: (output): beaten egg white and one yolk.

# Structogram for beaten egg white




Who wants to volunteer in executing these instructions?





# Computers lack human intuition and creativity

Computers are rather dumb machines. They follow your instructions without applying common sense.

Although logically correct, the structogram will lead to undesired "output" if really followed.

An important aspect of computer programming is thinking about all potential inputs and write instructions that can be followed without creative thinking

A good idea is looking for functions in libraries which contain highly efficient code and will thus speed up execution (similar to replacing the wire whisk with an electrical mixer).

# Programming Paradigms



There are three different programming styles in use:

1. Imperative programming: the solution is described using sequences of instructions, alternatives and loops as was used in the structogram for describing how to obtain beaten egg white.
2. Object oriented programming: solution and data are tied together. Objects are containers of data and "methods" which describe actions how to interact with the objects.
3. Functional programming: the solution is described by its properties and making use of induction.



# Functional Programming



Expression of factorial of $n$ ($n$ is a natural number).

$$n! = \prod_{k=1}^{n} k \quad \text{(corresponds to } 1 * 2 * \ldots * n\text{)}$$

Factorial in Haskell (a purely functional language):

```
factorial :: Integer -> Integer -- type declaration
factorial 0 = 1                 -- pattern matching
factorial n = n * factorial (n - 1)
```

"Functional" factorial in Python (an object oriented language):

```
def factorial(n):
    if n==0:
        return 1
    elif n>=0:
        return n*factorial(n-1)
```

# An object oriented linked list

Introduction to Machine Learning in Bioinformatics

Peter Sykacek



Task: we want a data structure which keeps "words" ordered according to an "alphabet".

```
class LinkList(object):
    def __init__(self):
        self.head=None
    def add2list(self, strval):
        # first we generate a node
        newnode=Node(strval)
        # and now add it:
        if self.head:
            self.head = self.head.addnode(newnode)
        else:
            self.head = newnode
```

LinkList does not do much: It just generates a node object and uses nodes "methods" to add the node to the list.



# An object oriented linked list: the node


Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting


```
class Node(object):
    def __init__(self, strval): # initialise node values
        self.strval=copy.copy(strval)
        self.next=None
    def addnode(self, node):
        if (self.strval < node.strval): # self before node
            if self.next: # node should sit further back
                self.next = self.next.addnode(node)
            else: # self is last node and we append node
                self.next = node
            # List between self and head remains unchanged.
            # We return self to keep the list linked!
            return self
        else:
            # node sits between selfs predecessor and self
            node.next = self
            return node
```

# Probability theory

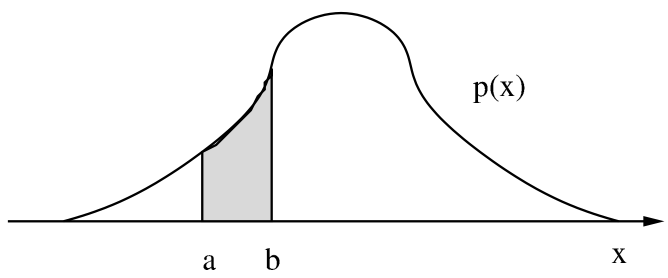


deals with uncertain events using random variables as basic concept.

Variables: $x$ represents deterministic value.

Random Variables: $x$ represents collection of values. Density function $p(x)$ describes relative occurrence of values. Like sand heap specifying occurrence of grain positions.

Area of shaded region: probability observing sample in interval $P(x \in [a, b]) = \int_{x=a}^{b} p(x)dx$.






# Rolling a Dice

Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Introduction

Computer Programming

Probability Theory

Data

Data + Model

Supervised methods

Unsupervised methods

Model Fitting

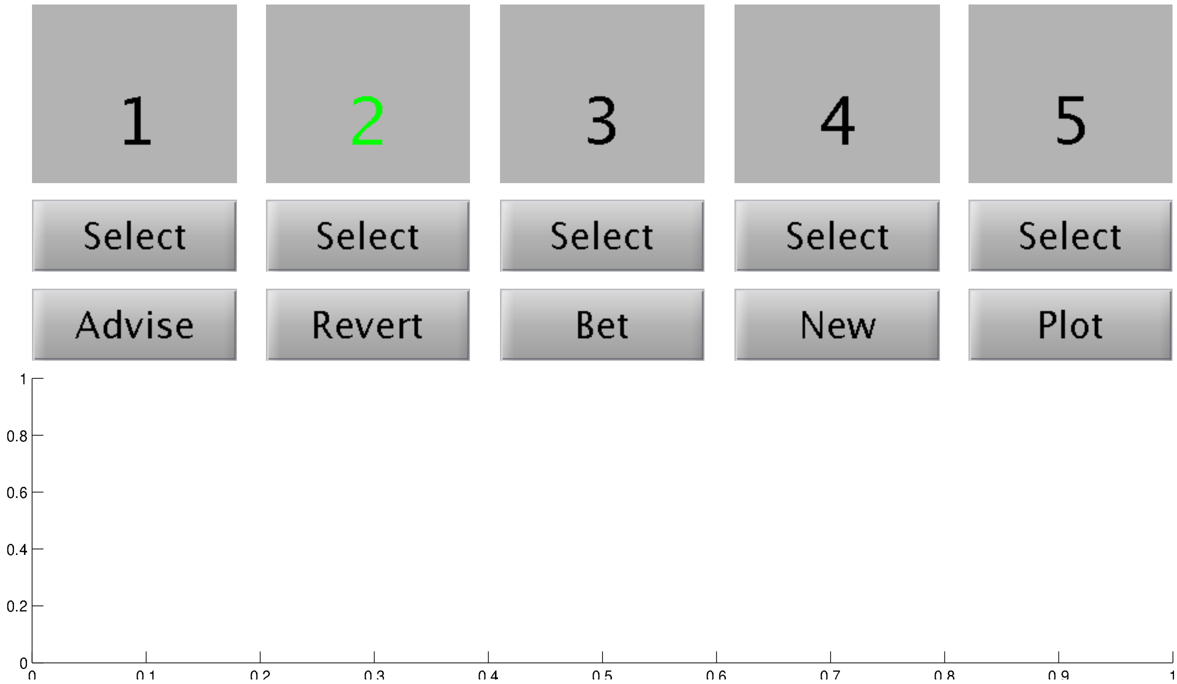
A dice is an example of a discrete random variable:

- We do not know which side will occur next.
- Occurrence of a particular outcome (e.g. rolling a 3) does not tell us anything about the next time we roll the dice.
- The dice as a random variable is fully described by the occurrence probability of the different sides.

A fair dice would have $P_1 = P_2 = ...P_6 = 1/6$, other parametrisation are possible. Example for biased distribution in biology: distribution over amino acids where some might be preferred in a particular class of proteins.

# Winning a prize in a quiz show





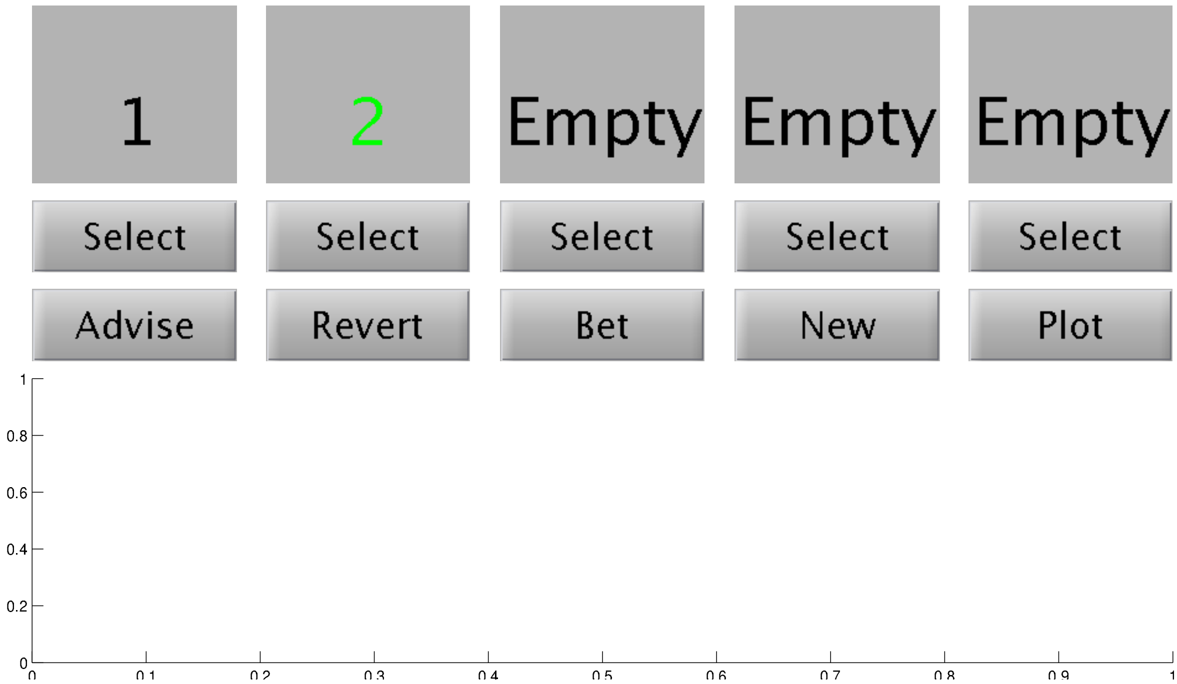

Select one of five doors behind which you might find your prize.

Probability of winning?



# A kind show master

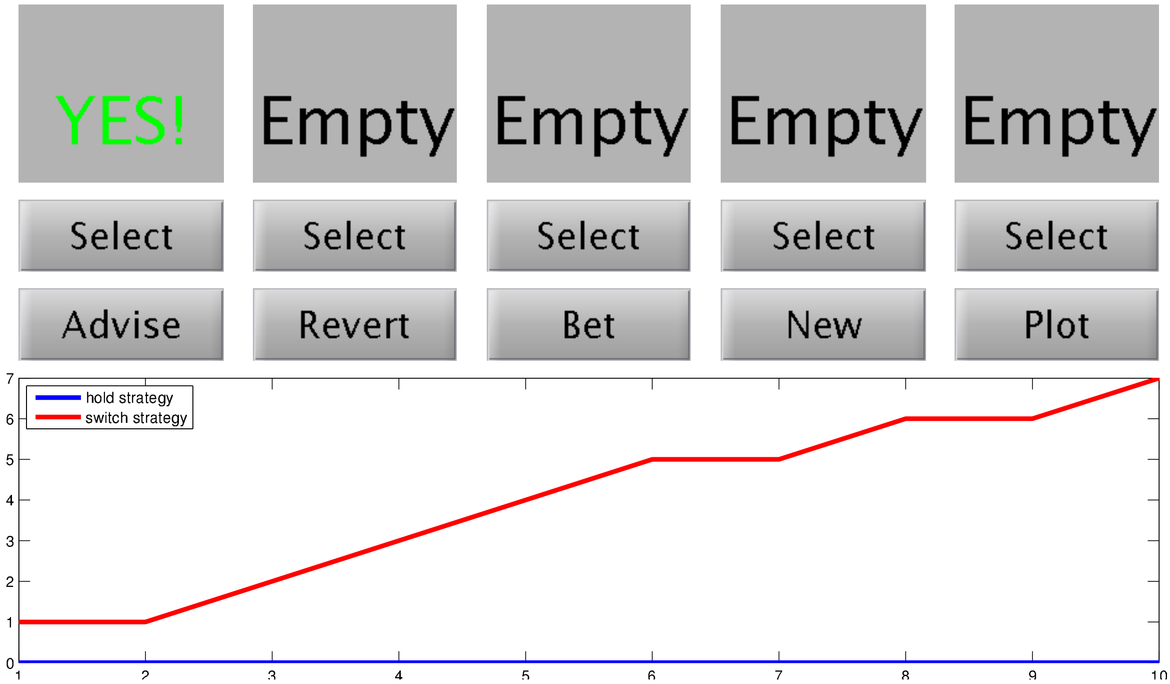
Introduction to Machine Learning in Bioinformatics
Peter Sykacek
Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting




Two strategies: A) Keep selection B) Alter selection

We need 10 volunteers for each strategy!

# Twenty shows later




Switching is clearly the better strategy. The probability of winning without switching is proportional to the initial selection. The probability of the other door winning is thus 80%.





# Nature of Data

- Data type (discrete vs. continuous)
- Observation is different from ground truth.

Discrete data: phenotype, genotype, age group,... Ordering among labels can be exploited. Continuous data: length, temperature, weight, pressure, mRNA expression,...

Measurement and ground truth: Measuring pencil length of 15.3cm $\neq$ true length of 15.3cm!
Why? Repeated measurements differ (15.2cm, 15.4cm, etc.)

$->$ measurement errors!

# Origin of Noise



Measurement processes involve errors which arise from noise (fluctuations) that are or can not be captured:

- Measurement noise.
- Missclassifications (e.g. wrong phenotype).
- Simplified Models.

Data analysis uses replicates to remove the noise and model the remaining aspects as good as possible.



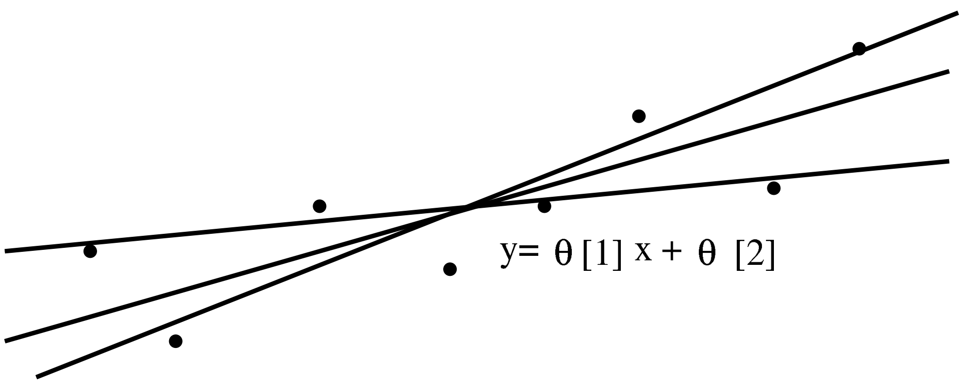
# Refresher: Scalar Product

Introduction to Machine Learning in Bioinformatics
Peter Sykacek
Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting

$K$ measurements $\mathbf{x}^T = [\mathbf{x}[1], ..., \mathbf{x}[K]]$ (row vector) represent variable $y$ as linear function (parameter $\boldsymbol{\theta}$). $->$ *linear regression* Express $y$:

$$y = \sum_k \mathbf{x}[k]\boldsymbol{\theta}[k], \text{ or } y = \mathbf{x}^T\boldsymbol{\theta} \text{ and equivalently } y = \boldsymbol{\theta}^T\mathbf{x}$$

$->$ vector dot product or scalar product

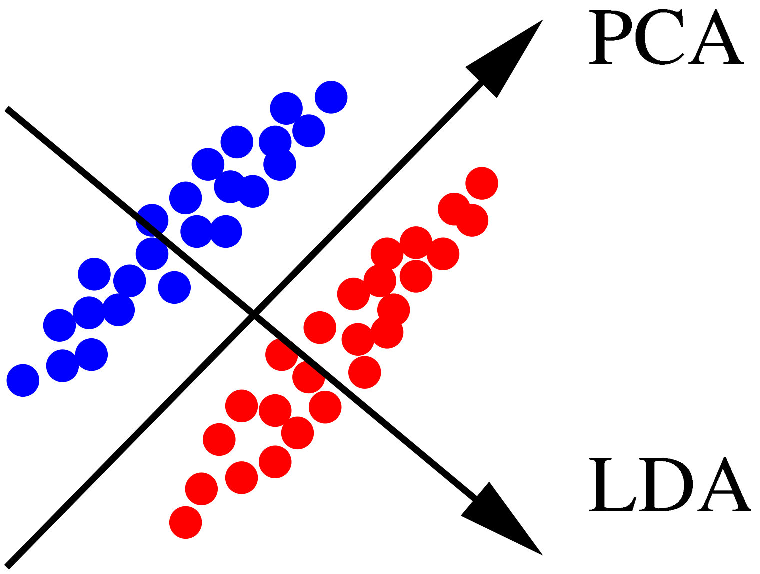



Peter Sykacek Introduction to Machine Learning in Bioinformatics

# Why Understand Data Analysis?

Result = Data + Model!

Linear discriminant (LDA) and principle component analysis (PCA) give different projections of the same data.




Both use linear projections!

$$t_{\mathrm{PCA}} = \theta_{\mathrm{PCA}}^T x$$

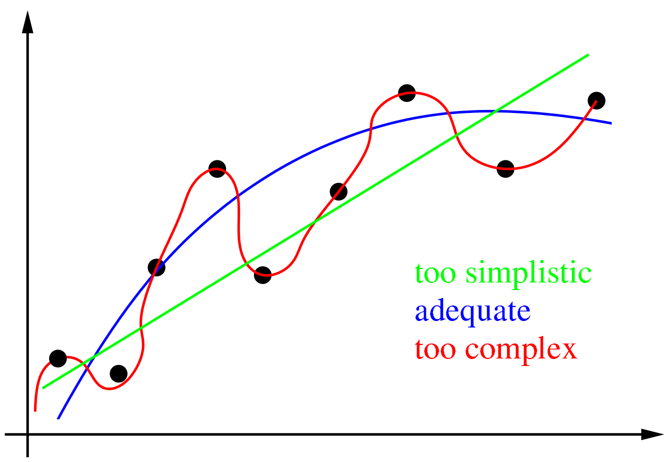
$$t_{\mathrm{LDA}} = \theta_{\mathrm{LDA}}^T x$$





# Adequate models

Capture underlying structure and avoid overfitting. Adjust "fiddle parameters" $->$ avoid too simple and too complex.




Overfitting memorises training data including uninteresting noise. To "learn something useful from data" we have to get complexity right.

Keeping data for validation and test purpose allows diagnosis!

# Analysis Strategies



All data analysis problems can be grouped into two categories:

1. Supervised Learning methods are used for regression problems.
2. Unsupervised Learning methods are used for exploratory data analysis.



# Supervised Method: Regression

Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting

Noisy Data from life science experiment
$\mathcal{Z} = \{(y_1, \mathbf{x}_1), ..., (y_N, \mathbf{x}_N)\}$ with $\mathbf{x}_n$ denoting vectors.

Regression fits based on $\mathcal{Z}$ an "optimal" function relating $\mathbf{x}$ and $y$:

$$y = f(\mathbf{x}; \boldsymbol{\theta}) + \epsilon(\lambda)$$

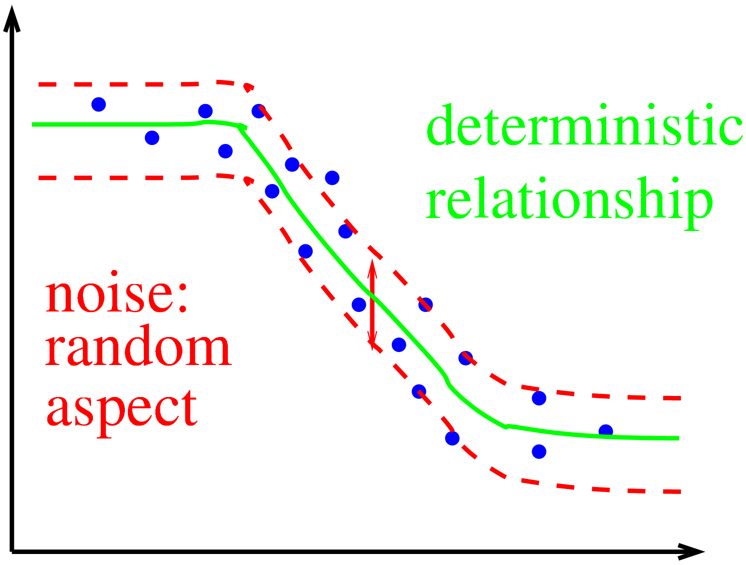
Noise requires a deterministic and a random component.

$->$ Inherent uncertainty, $y$ is a random variable!

# Implication of Randomness



Best we can do: Predict expected $y$ values from $x$ (local average). Complete description of data includes noise characteristics.




Red error bars represent the standard deviation which is the complete description in case of Gaussian noise.

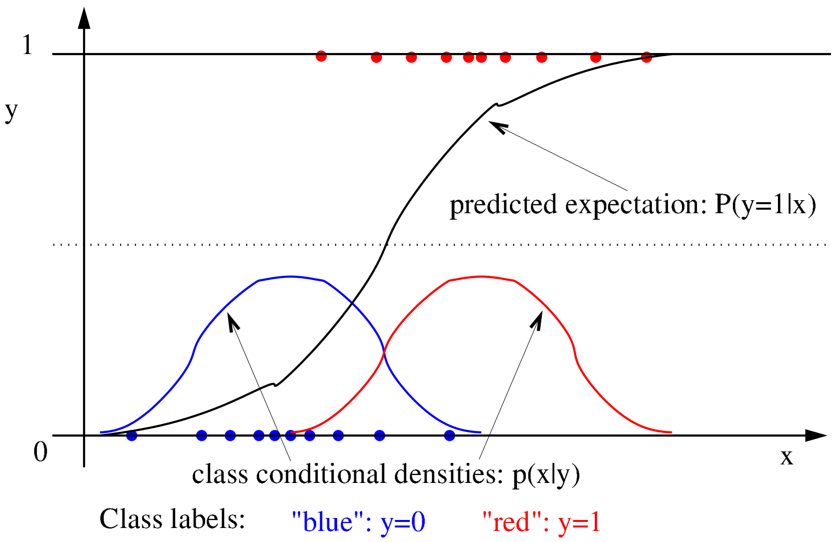


# Supervised Method: Classification



Classification is instance of regression, with predicted values (i.e. the $y$) being discrete.

Two classes: $y = \{0, 1\}$. Predicted expectations are the class probabilities $P(y|x)$.



# Unsupervised methods



Search of unknown structure in a data set $\mathcal{X} = \{x_1, x_2, ..., x_N\}$, $x_n$ distributed according to unknown pdf $p(x)$.

Learning task: summarise $x$ by an unobserved variable $t$.

Typical models:

Mixture density models: $p(x) = \sum_k P(t = k)p(x|t = k)$, and $t \in \{1, .., K\}$.

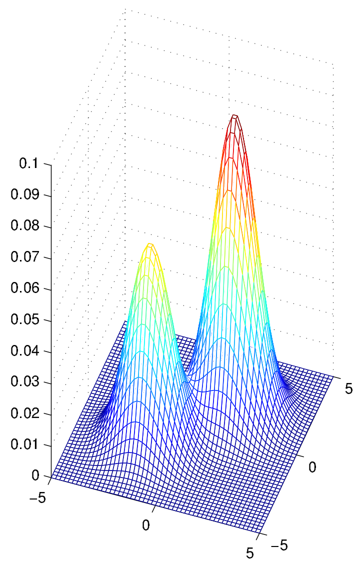
Continuous latent variable models: $p(x) = \int_t p(t)p(x|t)dt$, $x \in \Re^k$, $t \in \Re^d$ and $k > d$.

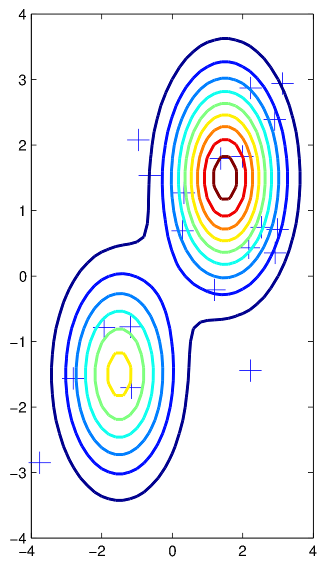


# Mixture Density Model

Introduction to Machine Learning in Bioinformatics
Peter Sykacek
Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting

Example: Gaussian mixture model $p(x|t = k) = \mathcal{N}(x; \mu_k, \lambda_k)$ - a Gaussian density function.




Summary: the $k$ which generated the observation $x$ $->$ Clustering.

Peter Sykacek Introduction to Machine Learning in Bioinformatics

# Continuous Latent Variable Model

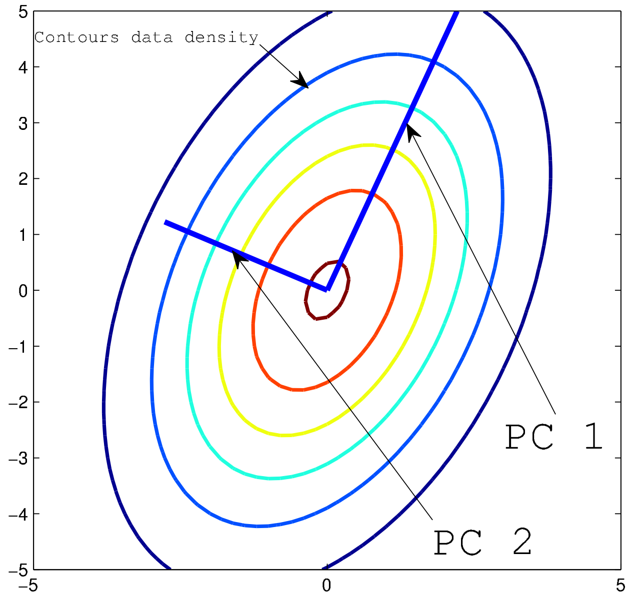


Example - PCA (principle component analysis):

$x = m + w_1 t_1 + \ldots + w_d t_d,\ x \in \Re^k,\ t = [t_1, \ldots, t_d] \in \Re^d$

$w_d : [k \times 1]$ $d$-th eigenvector of sample covariance matrix

$t \sim \mathcal{N}(t; 0, \Lambda)$, with $\Lambda : [d \times d]$ diagonal cov. matrix
Summary: lower dimensional continuous representation $->$ dimensionality reduction






# Analysis Tasks and Methods

Introduction to Machine Learning in Bioinformatics

Peter Sykacek

Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting

| Task | $->$ | Method |
|---|---|---|
| predict continuous $y$ from input data | $->$ | Regression |
| predict discrete $y$ from input data | $->$ | Classification |
| find unknown groups in input data | $->$ | Clustering (e.g. k-means, mixture models) |
| find low dimensional representation for input data | $->$ | Dimensionality reduction (PCA, ICA) |

# Assessing Model Parameters



Goal: tune $\boldsymbol{\theta}$ such that $f(\mathbf{x}_n; \boldsymbol{\theta})$ represents all $(y_n \mathbf{x}_n)$ pairs well.

Need expression we may optimise (maximise, minimise) for good fit of all $n$ "training" samples.

Possible choice: sum of squared errors (SSE). Idea: subtract deterministic part from $y_n$: $\epsilon_n = y_n - f(\mathbf{x}_n; \boldsymbol{\theta})$ + summation

$$\mathrm{SSE} = \sum_n \epsilon_n^2 = \sum_n (y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2$$

Several objective functions e.g. (log)-likelihood



# Major Problem

Introduction to Machine Learning in Bioinformatics
Peter Sykacek
Introduction
Computer Programming
Probability Theory
Data
Data + Model
Supervised methods
Unsupervised methods
Model Fitting

True model - linear regression:

$$y_n = \mathbf{x}_n^T \boldsymbol{\theta} + \epsilon_n$$

Finite sample size and arbitrarily complex models: What is the minimum of the SSE?

Think "phone book": Perfect memorising of all $y_n$, modelling error 0, SSE $->0$

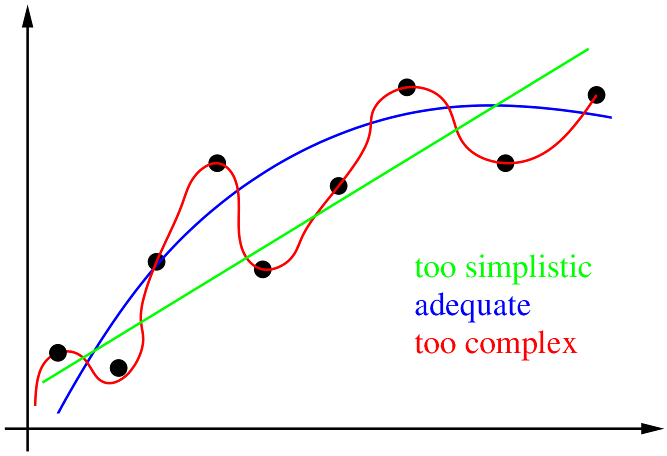
$->$ SSE unsuitable for model selection! (likelihood likewise!)

Peter Sykacek Introduction to Machine Learning in Bioinformatics

# Adequacy of models



Optimise model structure and avoid overfitting.




Wrong model class does not capture "truth" and performs worse in applications.

Some model classes:

$y = kx + d + \epsilon$

$y = lx^2 + kx + d + \epsilon$

$y = \sum_{j=0}^{J} (x^j k_j) + \epsilon$

How get complexity right ?
See my master courses on Machine Learning !



# In Depth Courses



Data Analysis is an important topic in modern life sciences. Three elective courses provide more advanced topics (In English, providing theoretical concepts and practical experience in the computer lab).

- *Efficient Microarray Data Analysis using R and FSPMA (793.403)* 1.0 HRS, winter term, 1.5 ETCS, A two day blocked lecture held entirely in computer lab.
- *Bayesian Data Analysis in the Life Sciences (793.302)* 3.0 HRS, winter term, 4.5 ETCS - theoretical part and 3 days blocked MatLab practical in the computer lab.
- *Machine Learning and Pattern Recognition for Bioinformatics (793.304)* 3.0 HRS, summer term, catalogued elective course with 4.5 ETCS - theoretical part and MatLab based practical in the computer lab.

Further details at `http://www.sykacek.net/teaching.html`