A Machine Learning Inspired Introduction to Data Analysis for Applications in Bioinformatics

Peter Sykacek[1]

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

Overview

- Fundamental Problems of Analysing Data
- Concepts for Data Analysis
- Supervised Learning
- Unsupervised Learning
- Matrices and Linear Regression (connection with Comp. Math. part!)
- Empirical Approaches for Diagnosing Models
- Maximum Likelihood
- Bayesian Inference

The Nature of Data

- Discrete valued observations (e.g. class labels).
- Continuous valued observations (e.g. measurements).

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

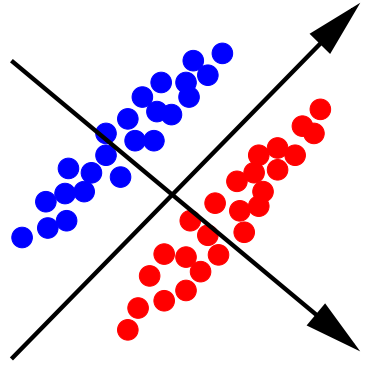
- Measurement noise.
- Wrong classifications (e.g. disease state).
- Simplified Models.

Individual data points do thus not reflect ground truth. Data analysis uses replicates to remove the noise and model the remaining aspects as good as possible.

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!

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

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

Good Statistical Practise I

Avoid ad hoc rules and match data analysis to the application domain.

Example: Proove hypothetical genes by measurements.

A simple rule for "proving genes": An RNA mix of K biological states, is hybridised on N microarrays. We declare all genes as verified, if $n < N$ arrays show expression above a threshold δ. Aspects worth considering:

1)Motivation of n - why for example $n = 6$ and not one more or less?

2)Motivation of δ - how is it specified?

3)We know a-priori that certain genes (e.g.transcription factors) tend to showing smaller expression than others. The required expression level will bias the proof towards highly expressed genes!

Approach does not fit the objective. $->$ Benchmark your ideas! (e.g. Do you produce more false negatives among certain candidates?)

Good Statistical Practise II

Avoid the "hammer and nail" syndrom and use methods which indeed answer the biological questions.

Example: identify cancer genes from microarrays obtained from cancer and wild type tissues.

We could propose using an SVM (support vector machine) and a greegy search strategy to find a gene set which is optimal for cancer predition. Aspects worth considering:

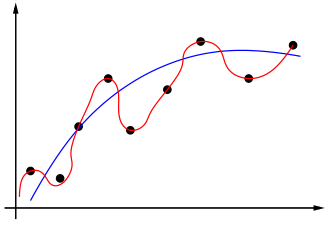
1) In most data sets, the SVM does not outperform a much simpler linear classifier using a single gene.

2) Greedy search provides a set which will work well for the classification task but does certainly not allow claiming having a complete cancer gene set.

3) The gene set provides no ranking w.r.t. functionally important genes.

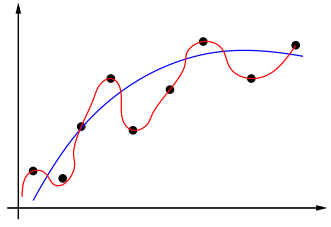
An otherwise useful approach (as a diagnostic tool) fails here answering the biological question!

Fundamental Principle in Data Analysis

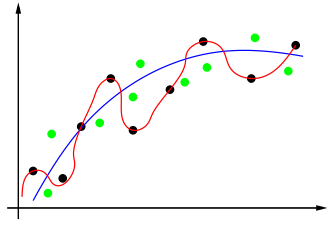


Which is the better model? Why is that the case?

Fundamental Principle in Data Analysis



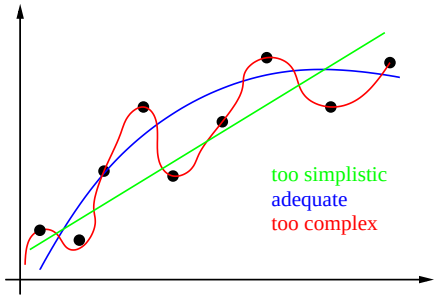
Which is the better model? Why is that the case?



Find (or abstract from) the underlying model that generated the data.

Adequate models

Capture underlying structure and avoid overfitting. “Fiddle parameters” affecting model complexity can have adverse effects.

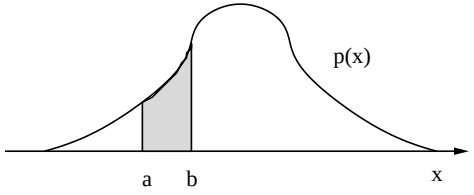


Idea: overfitting is a result of tuning the model towards the training data. Over or under-complex models that do not capture the underlying data generating mechanism will perform worse on novel data obtained from the generating model than an appropriate model.

Random Variable and PDF

Data analysis is inherently connected with the concept of random variables. A random variable is a non deterministic quantity where repeated observations differ and are generated according to some overall property. Properties of random variables are for example captured by an associated probability density function (pdf), $p(x)$.

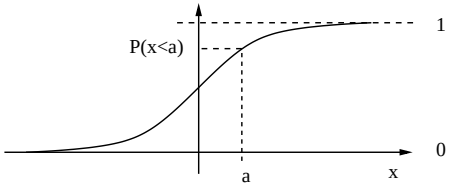
The pdf allows deducing the probability that a new realisation falls into a particular set, $P(x \in [a,b]) = \int_{x=a}^{b} p(x)dx$.



Cumulative Distribution Function

An equivalent characterisation for univariate random variables is provided by the so called cumulative distribution function (cdf), $F(x)$.

The cdf $F(x)$ denotes the probability that a realisation of the random variable is smaller than x. $F(a)$ is thus the probability $P(x < a)$.



> the pdf $p(x) = \frac{dF(\xi)}{d\xi}|_{\xi=x}$ is the derivative of the cdf at x.

Essential Rules of Probability Calculus

- If A and B represent mutually exclusive events with probabilities $P(A)$ and $P(B)$, the probability that either event occurs is $P(A) + P(B)$.
- The joint probability over A and B is: $P(A,B) = P(A)P(B|A) = P(B)P(A|B)$. If A and B are independent we have $P(B|A) = P(B)$ and $P(A|B) = P(A)$.
- Given $P(A,B,C) = P(A)P(B|A)P(C|A,B)$, we obtain $P(C,A) = \int_B P(A,B,C)dB$ by integration (here also named marginalisation).

Why Bother With Data Analysis?

Moore's Law:

PC 1984 5 MB Hard Drive

PC 2007 2 TB Hard Drive (4*500 GB) $\approx$ 400 Euro

How much paper on one PC in 2007 assuming 10.000 (single byte) characters per page ?

Why Bother With Data Analysis?

Moore's Law:

PC 1984 5 MB Hard Drive

PC 2007 2 TB Hard Drive (4*500 GB) $\approx$ 400 Euro

How much paper on one PC in 2007 assuming 10.000 (single byte) characters per page ?

It is actually a stack of paper 20 km high!

2 TB $\approx$ $2 * 10^{12}$ byte

$= 2 * 10^8$ pages, assuming 1000 pages = 10 cm

a stack $2 * 10^5 * 10$ cm = $2 * 10^4$ m = 20 km

What About Data Generation?

Medical monitoring 1:

20 channels EEG+physiological signals 8 hours sleep at 200 Hz and 16 Bit :

$20 * 8 * 3600 * 200 * 2 \approx 230,410^6$ byte ≈ 250 MB.

A single sleep lab with 8 recording units, operated at nights only, will generate one TB in just over a year.

What About Data Generation?

Medical monitoring 1:

20 channels EEG+physiological signals 8 hours sleep at 200 Hz and 16 Bit :

$20 * 8 * 3600 * 200 * 2 \approx 230,410^6$ byte ≈ 250 MB.

A single sleep lab with 8 recording units, operated at nights only, will generate one TB in just over a year.

Medical monitoring 2:

An FMRI scanner, 1dm^3 volume, 10s temporal and 1mm^3 spatial resolution, 16 bit.

One scanner generates $10^6 * 360 * 2$ byte ≈ 720 MB per hour which fills 1 TB in about 58 days.

What About Data Generation?

Medical monitoring 1:
20 channels EEG+physiological signals 8 hours sleep at 200 Hz and 16 Bit :
$20 * 8 * 3600 * 200 * 2 \approx 230,410^6$ byte ≈ 250 MB.
A single sleep lab with 8 recording units, operated at nights only, will generate one TB in just over a year.

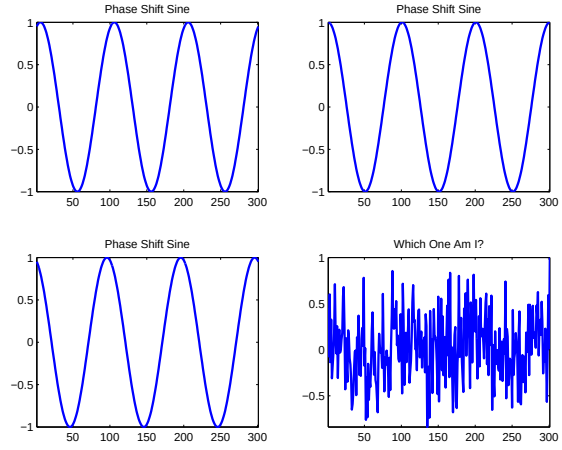
Medical monitoring 2:
An FMRI scanner, 1dm^3 volume, 10s temporal and 1mm^3 spatial resolution, 16 bit.
One scanner generates $10^6 * 360 * 2$ byte ≈ 720 MB per hour which fills 1 TB in about 58 days.

High throughput molecular biology:
A small lab produces up to 12 slides per 24 hours. One slide can contain up to 30.000 probes with ≈ 300 pixels/probe at 16 bit. Since we scan the entire array this is about 240 MB per 24 hours.

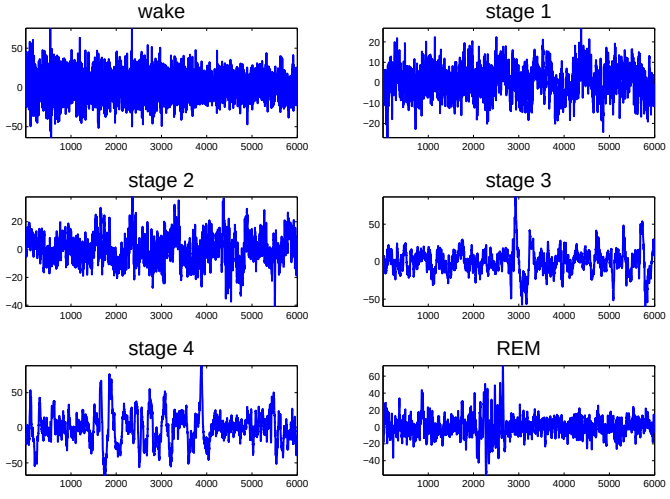
Such data can for two reasons not be analysed manually:
Amount and “Noise”

Manual Analysis Task

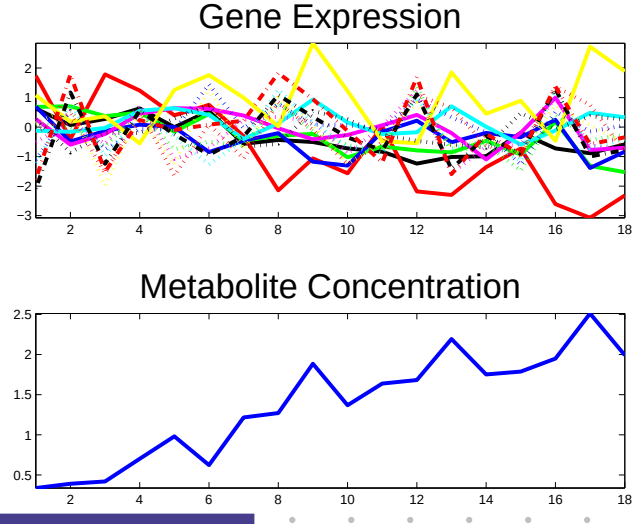
Which sine wave has the correct phase?



Example: Sleep EEG



Example: Metabolomics



Analysis Strategies

All data analysis problems can be grouped into two categopries:

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

Regression Problems

Regression is concerned with data sets of size N of the type $\mathcal{Z} = \{(y_1, x_1), (y_2, x_2), ...(y_N, x_N)\}$, with the tuples (y_n, x_n) drawn from an unknown joint pdf $p(y, x)$. The learning task is modelling the dependent variable, y, as a function of the independent variable x.

Typical regression models:
Linear regression: $p(y|x) = \mathcal{N}(kx + d, \lambda)$ and $y \in \Re$.
Logistig regression: $P(y|x) = \text{cdf}_{\text{logistig}}(kx + d)$ and $y \in \{0, 1\}$.

Exploratory Data Analysis

Exploratory data analysis searches for unknown structure in a data set of size N of the type $\mathcal{X} = \{x_1, x_2, ..., x_N\}$, with the x_n drawn from an unknown pdf $p(x)$. The learning task is modelling x as a function of an unobserved (latent) variable t, which provides a summary of the data.

Typical models for exploratory data analysis:
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 Models

Gaussian mixture model: $p(x|t = k) = \mathcal{N}(x; \mu_k, \lambda_k)$, i.e. a (possibly multivariate) Gaussian density function.
K-means clustering: can be regarded as a mixture density model with $P(t = k) = 1/K$ and $p(x|t = k)$ being K uniform densities with domains emerging from the Voronoi tesselation defined by the K cluster centers.
Hidden Markov Model: assumes a one dimensional ordering (e.g. time) among the latent variables t_n. We have thus a more complicated prior: $P(t_n|t_{n-1})$.

These models infer as summary information which mixture component generated the data point x_n $->$ Clustering.

Continuous latent variable models

Common aspect: $x = [m+]Wt[+\epsilon]$, $W : [d \times k]$ dimensional coefficients matrix, m, ϵ : optional mean and noise term.
PCA (principle component analysis): $t \sim \mathcal{N}(t; 0, \Lambda)$, $\Lambda : [d \times d]$ diagonal cov. matrix, $\epsilon : 0$, m: data mean.
Factor analysis: $t \sim \mathcal{N}(t; 0, \Lambda)$, $\Lambda : [d \times d]$ general cov. matrix, $\epsilon_k \sim N(\epsilon_k; 0, \lambda_k)$ and m: data mean.
ICA (independent component analysis): $t \sim \prod_d p(t_d|\theta_d)$ and $p(t_d|\theta_d)$: univariate density functions, at maximum one Gaussian, m, ϵ : implementation dependent.

These models provide as summary a lower dimensional representation of the data $->$ dimensionality reduction.

Matrices for Data Analysis

Important to simplify notation!
Definition n-dimensional Euclidian Space $\mathbb{R}^n$:

$$\mathbb{R}^n = \underbrace{\mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ times}} \; -> \text{Cartesian product}$$

Definition of a matrix (n rows, m columns):

$$\boldsymbol{M} = \begin{pmatrix} m_{1,1} & \cdots & m_{1,m} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,m} \end{pmatrix} = (\boldsymbol{m}_1, \cdots, \boldsymbol{m}_m) \text{ and } \boldsymbol{m}_i \in \mathbb{R}^n$$

MatLab: $>> M = [[a, b, c]; [d, e, f]; ...];$ What are the $\boldsymbol{m}_i$?

Matrix Operations

Transposition: $\boldsymbol{B} = \boldsymbol{A}^T$, $\forall n, m : b_{m,n} = a_{n,m}$
MatLab: $>> B = A';$

Addition ($\boldsymbol{A}$, $\boldsymbol{B}$ equal size):
$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$, $\forall n, m : c_{n,m} = a_{n,m} + b_{n,m}$
MatLab: $>> C = A + B;$
Associative and commutative?

Matrix times constant:
$\boldsymbol{B} = \lambda \boldsymbol{A}$ $\forall n, m : b_{n,m} = \lambda a_{n,m}$
MatLab: $>> B = lambda * A;$
Associative and commutative?

Matrix Multiplication

Matrix Inner Product ($\boldsymbol{A}$'s column no. equals $\boldsymbol{B}$'s row no.): $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ $\forall n, m :$ $c_{n,m} = \sum_i a_{n,i} b_{i,m}$
MatLab: $>> C = A * B;$
Associative and commutative?
Note: $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T \boldsymbol{A}^T$
However: $(\boldsymbol{A} + \boldsymbol{B})^2 = \boldsymbol{A}^2 + \boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A} + \boldsymbol{B}^2$

Hadamard Product ($\boldsymbol{A}$, $\boldsymbol{B}$ equal size):
$\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{B}$, $\forall n, m : c_{n,m} = a_{n,m} b_{n,m}$
MatLab: $>> C = A. * B;$
Associative and commutative?

Matrices and Data Analysis

Just to make sure that you see the connection between matrices and data analysis, here an example:

Assume N samples of k "input" measurements collected in $\boldsymbol{x}_n$ and one dependent variable y_n, which we intend modelling as a function $f(\boldsymbol{x}_n, \boldsymbol{\Theta})$ parameterised by $\boldsymbol{\Theta}$. This type of modelling is called *regression*.

We can only move on deciding on a particular $f(\boldsymbol{x}_n, \boldsymbol{\Theta})$.

For simplicity we assume that the best guess of y_n is obtained as linear combination of $\boldsymbol{x}_n$. This allows writing:

$$y_n = \sum_k \boldsymbol{x}_n[k]\boldsymbol{\Theta}[k], \text{ or } y_n = \boldsymbol{x}_n^T\boldsymbol{\Theta}$$

Model fitting I

is often done by minimising least squares differences

$$\hat{\boldsymbol{\Theta}} = \text{argmin}_{\boldsymbol{\Theta}} \left(\sum_n \left(y_n - \boldsymbol{x}_n^T\boldsymbol{\Theta} \right)^2 \right)$$

Importance of thinking in terms of matrices:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} \text{ and } \boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{pmatrix}$$

Model fitting II

We can then immediately write

$$\text{lsd} = \sum_n \left(y_n - \boldsymbol{x}_n^T\boldsymbol{\Theta} \right)^2 = (\boldsymbol{X}\boldsymbol{\Theta} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{\Theta} - \boldsymbol{y})$$

which contains no sums any more and is an extremely convenient method for deriving model fitting procedures and code for numerical tools like MatLab.

MatLab:$>> y\_d = X * theta - y;$
$>> LSD = y\_d' * y\_d;$
two lines to be more efficient!

The Least Squares Optimum

is, as previously discussed, obtained as $\hat{\boldsymbol{\Theta}} = \text{argmin}_{\boldsymbol{\Theta}}(\text{lsd})$. Similar to unconditional optimisation of functions:

- We take the derivative with respect to the quantity we want to optimise for.
- and set the derivative equal to zero.

$$\text{lsd} = \boldsymbol{\Theta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\Theta} - 2\boldsymbol{\Theta}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y}$$

Special: we take derivatives w.r.t the vector $\boldsymbol{\Theta}$!

Gradient Vectors of LSD Expression

$$\nabla_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\Theta}) = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\Theta}$$

$$\nabla_{\boldsymbol{\Theta}}(-2\boldsymbol{\Theta}^T\boldsymbol{X}^T\boldsymbol{y}) = -2\boldsymbol{X}^T\boldsymbol{y}$$

and

$$\nabla_{\boldsymbol{\Theta}}(\boldsymbol{y}^T\boldsymbol{y}) = \mathbf{0}$$

The gradient vector w.r.t. $\boldsymbol{\Theta}$ is thus given by:

$$\nabla_{\boldsymbol{\Theta}}(\text{lsd}) = 2(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\Theta} - \boldsymbol{X}^T\boldsymbol{y}).$$

Solution: $2(\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\Theta}} - \boldsymbol{X}^T\boldsymbol{y}) = 0$ or $\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\Theta}} = \boldsymbol{X}^T\boldsymbol{y}$

$->$ we have to remove $\boldsymbol{X}^T\boldsymbol{X}$ from the left side of the equality.

Square Matrices

Rank of a matrix $r(\boldsymbol{A})$: number of linearly independent columns (or rows, that's the same) of $\boldsymbol{A}$.

Square matrix $\boldsymbol{A}$ is a square matrix if no. rows equals no. cols, that is: $n = m$.

Square matrix $\boldsymbol{A}$ is non-singular if rank $r(\boldsymbol{A}) = n$.

The Determinant and the inverse are defined for square matrices.

Note: $\boldsymbol{X}^T\boldsymbol{X}$ from the previous slide is a square matrix!

The Determinant

The determinant $|\boldsymbol{A}|$ (MatLab: $>> det(A)$) converts a square matrix to a real number.
$|\boldsymbol{A}| = 0$ implies that $\boldsymbol{A}$ is singular
$|\boldsymbol{A}| = \prod_n \lambda_n$, where λ_n are the eigenvalues of $\boldsymbol{A}$

if $|\boldsymbol{A}|$ is an upper or lower diagonal matrix:
$|\boldsymbol{A}| = \prod_{i=1}^{n} a_{i,i}$

Recursive definition w.r.t j-th row (highschool!):

$$|\boldsymbol{A}| = \sum_i (-1)^{i+j} a_{i,j} |\boldsymbol{A}_{i,j}|$$

$|\boldsymbol{A}_{i,j}|$ is the determinant of the submatrix when removing the j-th row and i-th column.

Diagonal and Identity Matrices

Diagonal matrix: $\boldsymbol{A} = \text{diag}(a_{1,1}, .., a_{n,n})$, defines a matrix with the only non zero elements located on the main diagonal
MatLab: $>> A = diag(a);$ % places $\boldsymbol{a}$ into main diagonal of $\boldsymbol{A}$
$>> a = diag(A);$ % places main diagonal of $\boldsymbol{A}$ into $\boldsymbol{a}$

Identity matrix: $\boldsymbol{I} = \text{diag}(1, .., 1)$
neutral element of matrix multiplication:
$\boldsymbol{I}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{I} = \boldsymbol{A}$
MatLab: $>> I = eye(n);$ % generates an $[n \times n]$ identity matrix.

Inverse Matrix

If matrix $\boldsymbol{A}$ is non-singular, we get a non-singular matrix $\boldsymbol{B} = \boldsymbol{A}^{-1}$, such that, $\boldsymbol{BA} = \boldsymbol{AB} = \boldsymbol{I}$.
Matrix $\boldsymbol{B}$ is the inverse of $\boldsymbol{A}$

MatLab:$>> B = A\hat{}(-1)$

Remarks: $(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1}$ and $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$
Matrix $\boldsymbol{A}$ is orthonormal if $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, hence $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$
Examples: projection to principal axis (PCA)
Permutation matrix $\boldsymbol{P}$ in every row and column one 1 entry, otherwise 0

Using Inverse Matrices

Consider:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

with $\boldsymbol{A}$ square $[n \times n]$, then:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

Such operations are often found in data analysis, e.g. in finding least squares solutions.

Most important: Unlike in one dimensional operations, you must here multiply from the left! About 80% get this wrong in the exam!

Moore Penrose Pseudo Inverse

Inverting ill conditioned (close to singular) matrices involves quantities close to machine precision. Results derived from such inverse matrices result in large numerical errors.

Pracatical rule - never use matrix inversion, always use the Moore Penrose pseudo inverse.

$$\boldsymbol{A}^{+} = \lim_{\delta \to 0}(\boldsymbol{A}^T\boldsymbol{A} + \delta\boldsymbol{I})^{-1}\boldsymbol{A}^T$$

MatLab: $>> A\_plus = pinv(A);$
If $\boldsymbol{A}$ square and not ill-conditioned:
$\boldsymbol{A}^{+}\boldsymbol{A} = \boldsymbol{A}^{-1}(\boldsymbol{A}^T)^{-1}\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$

Solution for Least Squares Optimum

We have to solve:

$$\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\Theta}} = \boldsymbol{X}^T\boldsymbol{y}$$

for $\hat{\boldsymbol{\Theta}}$. We thus multiply both sides of the equality from the left with $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$. Left hand slide:

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X})\hat{\boldsymbol{\Theta}} = \boldsymbol{I}\hat{\boldsymbol{\Theta}} = \hat{\boldsymbol{\Theta}}$$

The solution is thus:

$$\hat{\boldsymbol{\Theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

in MatLab: $\text{theta\_hat} = \text{pinv}(X' * X) * X' * y;$

Regression Scenarios in Life Sciences

1. Given measurements $\boldsymbol{x}_n$ and some corresponding dependent information y_n, we might ask: How are they related?

Regression Scenarios in Life Sciences

1. Given measurements $\boldsymbol{x}_n$ and some corresponding dependent information y_n, we might ask: How are they related?
2. Given two sets of measurements $\boldsymbol{x}_n$ and $\boldsymbol{z}_n$, we might ask: Which of those are closer related to some corresponding dependent information y_n?

Regression Scenarios in Life Sciences

1. Given measurements $\boldsymbol{x}_n$ and some corresponding dependent information y_n, we might ask: How are they related?
2. Given two sets of measurements $\boldsymbol{x}_n$ and $\boldsymbol{z}_n$, we might ask: Which of those are closer related to some corresponding dependent information y_n?

$->$ two instances of “inference” commonly found in applied life sciences.

First Scenario

Suppose a life science experiment provided some noisy data $\mathcal{Z} = \{(y_1, \boldsymbol{x}_1), ..., (y_N, \boldsymbol{x}_N)\}$. Note: $\boldsymbol{x}_n$ possibly multivariate i.e. vectors.

Based on $\mathcal{Z}$, we have an inference problem of finding an “optimal” relation between $\boldsymbol{x}$ and y:

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

First Scenario

Suppose a life science experiment provided some noisy data $\mathcal{Z} = \{(y_1, \boldsymbol{x}_1), ..., (y_N, \boldsymbol{x}_N)\}$. Note: $\boldsymbol{x}_n$ possibly multivariate i.e. vectors.

Based on $\mathcal{Z}$, we have an inference problem of finding an "optimal" relation between $\boldsymbol{x}$ and y:

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

Noise requires a deterministic and a random component.

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

Inference

Parameter Inference:

Implies knowing $f(\boldsymbol{x}; \boldsymbol{\theta})$ and the noise model $\epsilon(\lambda)$ up to unknown parameters ($\boldsymbol{\theta}$ and λ) which we will be inferring from data.

Inference

Parameter Inference:

Implies knowing $f(\boldsymbol{x}; \boldsymbol{\theta})$ and the noise model $\epsilon(\lambda)$ up to unknown parameters ($\boldsymbol{\theta}$ and λ) which we will be inferring from data.

Model Inference:

A more realistic assumption is that the model class is unknown and we will be inferring model class and parameters.

Assessing Model Parameters

Idea: subtract the deterministic part from y_n:

$$\epsilon_n = y_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})$$

For convenience introduce $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ and $\mathcal{D} = \{y_1, ..., y_N\}$. Assuming that ϵ_n are i.i.d samples, we get the likelihood function:

$$p(\mathcal{D}|\boldsymbol{\theta}, \lambda, \mathcal{X}) = \prod_n p(y_n|\boldsymbol{\theta}, \lambda, \boldsymbol{x}_n)$$

which is a suitable objective function to be maximized for $\boldsymbol{\theta}$ and λ.

Likelihood and Linear Regression

Assuming N samples, we have:

$$p(y_n|\boldsymbol{x}_n;\boldsymbol{\theta},\lambda) = (2\pi)^{-0.5}\lambda^{0.5}\exp(-0.5\lambda(y_n-\boldsymbol{x}_n^T\boldsymbol{\theta})^2) \text{ and}$$
$$p(\mathcal{D}|\mathcal{X};\boldsymbol{\theta},\lambda) = (2\pi)^{-\frac{N}{2}}\lambda^{\frac{N}{2}}\exp(-0.5\lambda\sum_n(y_n-\boldsymbol{x}_n^T\boldsymbol{\theta})^2)$$

Taking the $\log$, we get the log likelihood:

$$\text{llh} = \frac{N}{2}(\log(\lambda)-\log(2\pi))-0.5\lambda(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})$$

which, if we consider maximising for $\boldsymbol{\theta}$ only, is a familiar expression.

$>$ minimising least squares assumes Gaussian noise!

Likelihood and Classification

"Classification" often used synonymously for regression with discrete outcomes. Likelihood of regression model:

$$P(\mathcal{D}|\mathcal{X};\boldsymbol{\theta}) = \prod_n P(y_n|\boldsymbol{x}_n,\boldsymbol{\theta})$$

To enforce $\sum_{y_n} P(y_n|\boldsymbol{x}_n,\boldsymbol{\theta})$ is 1, we apply a suitable output transformation, e.g. the cdf of the logistig distribution:

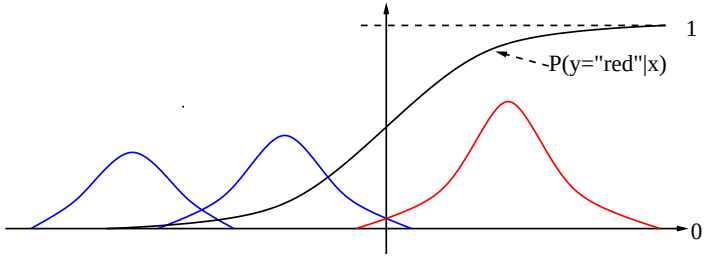
$$P(y_n|\boldsymbol{x}_n^T\boldsymbol{\theta}) = \frac{1}{1+\exp((2y_n-1)\boldsymbol{x}_n^T\boldsymbol{\theta})}$$

Probabilities are certainty measures about classes to avoid ignorant decisions:
Surgeon: Amputate or not?
Nurse: The SVM says +1.

Classification and Sampling Paradigm

$$P(y_n|\boldsymbol{x}_n) = \frac{P(y_n)p(\boldsymbol{x}_n|y_n)}{p(\boldsymbol{x}_n)}$$

$->$ Bayes theorem suggests that we can also model class priors $P(y_n)$ and class conditional densities $p(\boldsymbol{x}_n|y_n)$.



vantage: a useful density model, disadvantage: more complicated

Practical Data Analysis

The quality of the inferred model will most often depend on the chosen data representation and settings of "fiddle parameters" like model order or smothness inducing coefficients.

This suggests considering

- transformations of the data (for good representations and filling in missing information)
- measures (quantify adequacy of models)
- and methods for assessing models (determine adequacy of models)

Missing Values

Models that do not represent the joint distribution of the data, can not deal with missing observations. Solution:

- Modelling based on joint input output distributions (e.g. classification in the sampling paradigm) and marginalising over all missing values during inference.
- Fill in missing values a-priori (e.g. k-nearest neighbour imputing) and fit model of choice to completed data.

The former is more principled though more involved and limited to using certain models. The latter remains ad hoc.

Data Transformations

Match modelling assumptions:

Measurements from life sciences sometimes provide strictly positive quantities (e.g. RNA concentration). Strictly positive quantities are necessarily non Gaussian. Modelling based on Gaussianity assumptions (e.g. logistic regression) could thus benefit from transforming measurements to $\mathbb{R}$, which is in this case for example obtained by taking logs.

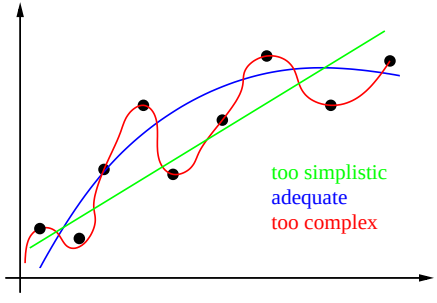
Improve modelling:

Modelling can benefit from simple linear transformations like adjusting inputs to zero mean and unit std. deviations (in the context of Bayesian priors), or from projecting to principal component directions (i.e. zero covariance among all variables).

Transformations must be information preserving, otherwise they are dangerous. In theory neither of these transformations makes a difference!

Adequate models

Capture underlying structure and avoid overfitting. Fiddle parameters affecting model complexity can have adverse effects.



Over or under-complex models that do not capture the underlying data generating mechanism will perform worse on novel data obtained from the generating model than an appropriate model. Model classes (just examples):

$y = kx + d + \epsilon$

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

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

all cases: $\epsilon \sim \mathcal{N}(\epsilon; 0, \lambda)$.

Keeping data for validation and test purpose allows diagnosis!

Model Diagnostic Measures

The mean square generalisation error (MSE_{test}) allows assessing regression models.

$$\mathrm{MSE}_{test} = \frac{1}{N}(\boldsymbol{y}_{test} - \boldsymbol{f}(\boldsymbol{X}_{test}; \boldsymbol{\theta}_{train}))^T(\boldsymbol{y}_{test} - \boldsymbol{f}(\boldsymbol{X}_{test}; \boldsymbol{\theta}_{train}))$$

Classification aims at labeling novel samples correctly. This is tested by the generalisation accuracy acc_{test}.

$$\forall n\ \hat{\boldsymbol{y}}_{test}[n] = \mathrm{argmax}_k(P(y = k|\boldsymbol{X}_{test}[n,:]\boldsymbol{\theta}_{train}))$$

$$\mathrm{acc}_{test} = \frac{1}{N}\sum_n \delta(\boldsymbol{y}_{test}[n], \hat{\boldsymbol{y}}_{test}[n])$$

We classify such that the most probable class wins and estimate the fraction of correctly classified test cases.

Estimating Diagnostic Measures

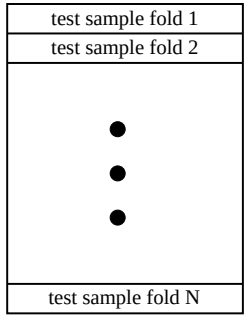
Trade-off: reliable inference requires large "training sets" (i.e. many samples for model fitting); unbiased diagnostics require large "test sets" (i.e. many novel samples for assessing the model).

Diagnostic quantities are only unbiased if we leave test samples untouched! Test samples must not be used for any modelling decisions.

$->$ Solution: reuse samples by iterating over model fitting and testing.

N-Fold Cross Testing

Sketch and MatLab like pseudo code




```
allres=[]; allpred=[];
for n=1:n_folds
    % split into training and test data
    [train, test]=foldsplit(orig_data, n_folds, n);
    % model inference
    [model]=trainfunc(train, fiddleparams);
    % store this folds true targets and predictions
    [res]=truetarg(test);
    [pred]=predtarg(test, model);
    allres=[allres; res];
    allpred=[allpred; pred];
end
```

Leave one out has as many folds as samples. An alternative by resampling with replacement is called Bootstrapping.

More on Assessing Classifiers

What is the implication of predicting the label which got largest posterior probability?

Consider an individual prediction and that we know the correct posterior $P(y|\boldsymbol{x})$: deciding for label $y = 1$ implies being correct with probability $P(y = 1|\boldsymbol{x})$ and being wrong with probability $1 - P(y = 1|\boldsymbol{x})$.

If we decide for the label $k = \text{argmax}_k(P(y = k|\boldsymbol{x}))$, we will thus minimise the overall number of missclassifications.

What if the missclassifications cost is class dependant?

Missclassification Cost

Classifying correctly induces zero cost. A false negative for class k induces cost α_k.
Predicting $y = t$ results in an expected cost (deciding that the true y is probably t, this is the cost we expect to suffer):

$$C = \sum_{k \neq t} P(y = k|\boldsymbol{x})\alpha_k$$

Missclassification cost C is thus minimised by classifying $y = \text{argmax}_k(P(y = k|\boldsymbol{x})\alpha_k)$.

This structure is commonly found in life science applications. In cancer screening a false negative is obviously much worse than a false positive.

ROC Curve I

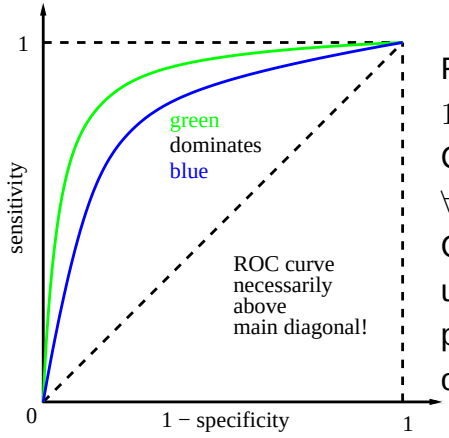
For unknown missclassification cost, the Receiver Operating Characteristic (ROC) curve provides means for assessing binary classifiers.

$$\text{Sensitivity: } \mathrm{s}(\gamma) = \frac{\sum_{n|y_n=1} \delta(P(y_n = 1|\boldsymbol{x}_n) > \gamma)}{N_+}$$

$$\text{Specificity: } \mathrm{p}(\gamma) = \frac{\sum_{n|y_n=0} \delta(P(y_n = 1|\boldsymbol{x}_n) < \gamma)}{N_-}$$

depend on a detection threshold γ; $\delta()$ maps "true" to 1 and "false" to 0; N_+ is the number of positive and N_- the number of negative samples.

ROC Curve II



ROC curve: $s(\gamma)$ over $1 - p(\gamma)$

Classifier a dominates b if $\forall\, s_a, s_b = s : p_b(s_b^{-1}) < p_a(s_a^{-1})$

Given domination, the area under the ROC curve (AUC) provides a quality measure of the classifier.

AUC and survival probability: $\mathrm{AUC} = P(P(y = 1|x \sim p(x|1)) > P(y = 1|x \sim p(x|0)))$.

Comparing Classifiers

Verification particular choices like the chosen model and fiddle parameters requires performance comparisons.

Default Accuracy:

The most trivial competitor predicts for every sample it's prior probability and thus always the majority class.

Competing Classifiers:

Since it is not at all clear, whether a particular model works well in the case at hand, one should use a set of different approaches. A simple, yet still powerful approach is the so called k nearest neighbour classifier.

Given several performance measures, we should investigate, whether differences are significant.

Mc Nemars Test

The idea of Mc Nemars test is that differences in classifiers manifest themselves in differently classified samples. This allows prodcing a 2 by 2 contingency table:

	C_{2w}	C_{2c}
C_{1c}	n_a	n_{bthc}
C_{1w}	n_{bthw}	n_b

C_{xc} and C_{xw} refer to samples correctly / wrongly labeled by classifier x. Differences manifest in the number of samples, n_a, which C_1 labels correctly and C_2 labels wrongly and the number of samples, n_b, where the situation is vice versa.

If both classifiers are equal, (n_a, n_b) is a sample of drawing $n_a + n_b$ times from a Binomial distrbution with probability 0.5. The null hypothesis of Mc Nemars test is the Binomial $\mathcal{B}n(n_a + n_b, 0.5)$ and we obtain the p-value by calculating the tail probability from (n_a, n_b) onwards.

A Major Problem

True model - linear regression, Gaussian noise:

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

$f(\boldsymbol{x};\boldsymbol{\theta}) = [1, \boldsymbol{x}^T]\boldsymbol{\theta}$ and $\epsilon(\lambda) = \mathcal{N}(\epsilon; 0, \lambda)$, with λ denoting "precision" (i. e. inverse variance).

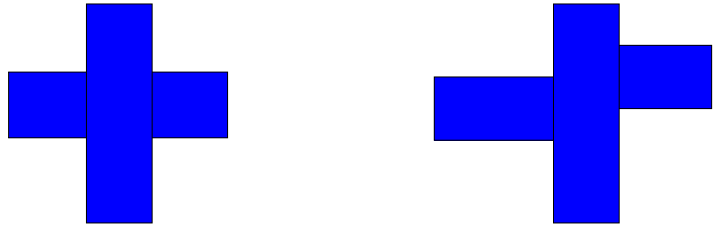
Finite sample size and different model classes: What is the maximum of the likelihood?

Think "phone book": Perfect memorizing of all y_n, modelling error 0, $\lambda -> \infty$, $p(\mathcal{D}|\boldsymbol{\theta}, \lambda, \mathcal{X}) -> \infty$.

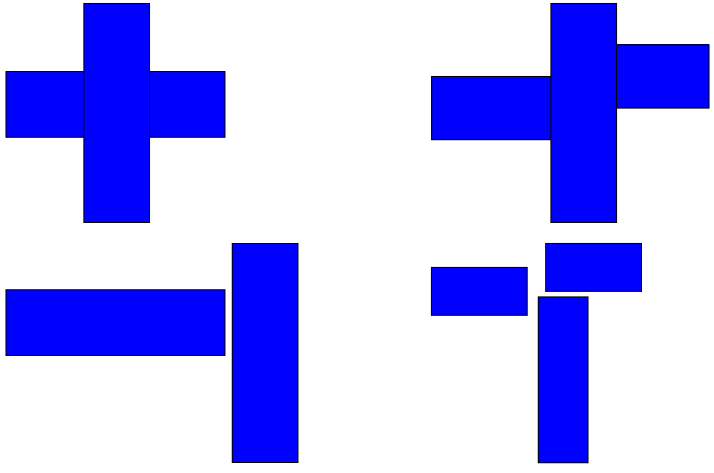
-> likelihood unsuitable objective for model inference!

Why is memorizing useless?

Guess the Correct "Model"



Guess the Correct "Model"



Comparing models requires complexity penalties on top of the likelihood! (AIC, BIC, etc.)

Occam's Razor

We implicitly apply Occam's Razor

William of Occam (or Ockham) (1288 - 1348)

Entia non sunt multiplicanda sine necessitate: Entities are not to be multiplied without necessity.

Interpretation: One should always opt for an explanation in terms of the fewest possible number of causes, factors, or variables.

Material from `http://en.wikipedia.org/wiki/William_of_Ockham`.

Bayesian Inference

Thomas Bayes (1701 - 1763)

Occam's Razor built in!
Two important consequences for "learning from data". Inference based on a decision theoretic framework

Bayesian Inference

Thomas Bayes (1701 - 1763)

Occam's Razor built in!
Two important consequences for "learning from data". Inference based on a decision theoretic framework

$$p(I|\mathcal{D}) = \frac{p(\mathcal{D}|I)p(I)}{p(\mathcal{D})}$$

1) Revise beliefs by Bayes theorem

Bayesian Inference

Thomas Bayes (1701 - 1763)

Occam's Razor built in!
Two important consequences for "learning from data". Inference based on a decision theoretic framework

$$p(I|\mathcal{D}) = \frac{p(\mathcal{D}|I)p(I)}{p(\mathcal{D})}$$

1) Revise beliefs by Bayes theorem

$\alpha_{opt} = \text{argmax}_\alpha < u(\alpha) >$, where
$< u(\alpha) >= \int_G u(\alpha, I)p(I|\mathcal{D})dI.$

2) Decisions by maximising expected utility

More on Data Analysis

Data Analysis is a very important topic in modern life sciences. I offer thus three elective courses on data analysis (all given in English, providing theoretical concepts and practical hands on experience in the computer lab).

- *Efficient Microarray Data Analysis with R and FSPMA (793.403)* 1.0 HRS SS 2009 This is a two day blocked lecture (18th and 19th of May) which will be held in the Computer Lab in the 6th floor. This lecture will be moved to WS and should happen in WS 09/10 again. See details in BLIS - registration is still open.
- *Neural Networks and Pattern Recognition in Bioinformatics (793.404)* 2.0 WS (planned for WS 2009/2010 - theoretical part and MatLab based practical in the computer lab, no BLIS entry yet).
- *Bayesian Data Analysis in the Life Sciences (793.402)* 3.0 HRS (will be moved from WS to SS thus SS 2010). This lecture consists of a theoretical part and a 3 days blocked MatLab based practical in the computer lab (see BLIS).

App.: Bayesian Dice Model - Likelihood

Goal: inferring probabilities observing sides of a dice, i.e. $\pi = \{\pi_1, .., \pi_5, 1 - \sum_{k=1}^{5} \pi_k\}$
Data: N observations from rolling the dice.

App.: Bayesian Dice Model - Likelihood

Goal: inferring probabilities observing sides of a dice, i.e. $\pi = \{\pi_1, .., \pi_5, 1 - \sum_{k=1}^{5} \pi_k\}$
Data: N observations from rolling the dice.

We need a likelihood function:
Throwing the dice once results in a multinomial one distribution over sides, i.e.
$P(I_n|\pi) = \prod_{k=1}^{6} \pi_k^{\delta(I_n=k)}$, where $I_n \in \{1, .., 6\}$.

Independence assumption $->$ likelihood:
$p(\mathcal{D}|\pi) = \prod_n P(I_n|\pi)$, where $\mathcal{D} = \{I_1, ..., I_N\}$ denotes the N outcomes.
What is the final expression of the likelihood?

App.: Bayesian Dice Model - Prior

We typically use a conjugate prior: a convenient choice to remain within a functional family which is a known distribution. The Multinomial suggests a Dirichlet prior over π:

$$p(\pi) = \frac{\Gamma(\sum_{k=1}^{6} \alpha_k)}{\prod_{k=1}^{6} \Gamma(\alpha_k)} \prod_{k=1}^{6} \pi_k^{\alpha_k - 1}$$

$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} \exp(-x) dx$ is known as gamma function.
Write the definition of $\Gamma(\alpha)$ down! You will need it later during the lecture!

The α_k are hyper parameters of our model.
What is their logical meaning?

App.: Bayesian Dice Model - Posterior

Multiplying prior and likelihood and renormalising gives the posterior distribution over π as the result of Bayesian inference of the dice model:

$$p(\pi|\mathcal{D}) = \frac{1}{p(\mathcal{D})} \frac{\Gamma(\sum_{k=1}^{6} \alpha_k)}{\prod_{k=1}^{6} \Gamma(\alpha_k)} \prod_{k=1}^{6} \pi_k^{\alpha_k + n_k - 1}$$

where $p(\mathcal{D}) = \int_{\pi_1,..,\pi_6} p(\pi, \mathcal{D}) d\pi$ denotes the marginal likelihood, which is useful for model selection.
What is the functional form of the marginal likelihood ?

App.: Iterative Inference

Given prior counts $\{\alpha_1, ..\alpha_k\}$ and data sets $\mathcal{D}_1 = \{I_1, ..., I_N\}$ and $\mathcal{D}_2 = \{I_{N+1}, ..., I_{N+M}\}$, using $p(\pi|\mathcal{D}_1)$ as prior for $\mathcal{D}_2$ will result in the same posterior $p(\pi|\mathcal{D}_1, \mathcal{D}_2)$ we get from the original prior and the pooled data $\mathcal{D} = \{I_1, .., I_{N+M}\}$:

$$p(\pi|\mathcal{D}_1) = \frac{\Gamma(\sum_k(\alpha_k + n_k))}{\prod_k \Gamma(\alpha_k + n_k)} \prod_k \pi_k^{\alpha_k + n_k - 1}$$

$$p(\pi|\mathcal{D}_1, \mathcal{D}_2) = \frac{\Gamma(\sum_k(\alpha_k + n_k + m_k))}{\prod_k \Gamma(\alpha_k + n_k + m_k)} \prod_k \pi_k^{\alpha_k + n_k + m_k - 1}$$

Since $n_k + m_k$ is the overall number of observations of side k this is equivalent to $p(\pi|\mathcal{D})$.

App.: Applied Bayesian Decision Theory

Horse betting: bet x; choice α; uncertain outcome of race I. Bookmakers "odds" r_A and r_B (one + odds ratio) imply utility function $u(\alpha, I)$:

$\alpha \backslash I$	"A" wins	"B" wins
bet "A"	xr_A	0
bet "B"	0	xr_B
no bet	x	x

Need probability of $I = [A, B]$ i.e. respective horse wins. From previous observations (races) $\mathcal{D}$: $P(I = A|\mathcal{D}) = 0.7$ and $P(I = B|\mathcal{D}) = 0.3$.

App.: Horse Betting ctd.

Calculate expected utility $u(\alpha) = \sum_I u(\alpha, I)P(I|\mathcal{D})$:

bet "A"	bet "B"	no bet
$0.7xr_A$	$0.3xr_B$	x

Maximise expected utility!

case	I	II	III
r_A	1.4	1.9	1.3
r_B	3.2	2.5	4.5

What are your decisions?

App.: Horse Betting ctd.

Calculate expected utility $u(\alpha) = \sum_I u(\alpha, I)P(I|\mathcal{D})$:

bet "A"	bet "B"	no bet
$0.7xr_A$	$0.3xr_B$	x

Maximise expected utility!

case	I	II	III
r_A	1.4	1.9	1.3
r_B	3.2	2.5	4.5

What are your decisions?

Can we earn money?