Mathematical Methods for Data Analysis

Evženie Uglickich, Ivan Nagy

Faculty of Transportation Sciences
Czech Technical University, Prague

Contents

1 Introduction, probability, system **5**
1.1 Revision of statistics 5
1.2 System and its variables 6

I Stochastic Systems **7**

2 Regression and categorical models **8**
2.1 Regression model 8
2.2 Categorical model 9
2.3 Scilab generations 11

3 Bayes rule **12**

4 Estimation of specific models **15**
4.1 Normal static regression model 15
4.2 Normal regression model 18
4.3 Categorical model 20

5 Output prediction **22**
5.1 Output estimation 22
5.2 One step prediction 22
5.3 Multi-steps prediction 23
5.4 Prediction with discrete model 24

6 State-space model, state estimation **25**
6.1 Model 25
6.2 Estimation 25

7 Nonlinear state estimation **28**
7.1 Nonlinear model 28
7.2 Model with unknown parameters 29

8 Control with regression model **31**
8.1 Derivation in pdf 31
8.2 Derivation for regression model 32

9 Control with categorical model **35**

9.1 Optimization 35

9.2 Application 37

II Clustering and Classification **38**

10 Mixtures **39**

10.1 The basics of clustering and classification 39

10.2 Naive Bayes classification 42

10.3 Classification with learning 43

11 Regression **50**

11.1 Logistic regression 50

11.2 Poisson regression 52

12 Clustering **55**

12.1 K-means algorithm 55

12.2 K-medoids algorithm 57

12.3 Fuzzy clustering 60

12.4 Density based clustering 63

12.5 Hierarchical clustering 67

13 Classification **75**

13.1 K-nearest neighbour 75

13.2 Decision trees 77

13.3 Support vector machines 82

III Supplements **86**

14 Competition to squares **86**

15 Natural conditions of control **86**

16 Bayes rule **87**

17 Categorical distribution **88**

18 Dirichlet distribution 88

19 Normal distribution 89

20 Inverse Gauss-Wishart distribution 90

21 Point estimate with quadratic criterion 90

22 Point estimates of regression model parameters 91

23 Point estimates of categorical model parameters 92

24 Logistic regression in details 93

1 Introduction, probability, system

www.fd.cvut.cz/personal/nagyivan + Mathematical models $\cdots$

1.1 Revision of statistics

- **Variable** × **Random variable** (continuous, discrete)

  Remark: Variables are (i) continuous, (ii) discretized (ordinal), (iii) discrete (nominal) - can be ordered according to something (frequently money, some loss).

- **Distribution** (pf, pdf)

  - discrete: $f(x) \equiv P(X = x)$
  - continuous: $f(x) \equiv \lim P(O_x)/m(O_x)$ for $m(O_x) \to 0$, where $m(O_x)$ is a measure of the neighborhood O_x around the point x

- **Random vector**, joint; marginal; conditional distribution
  draw continuous and discrete uniform distribution for $X = [x_1, x_2]$

$$f(x_1, x_2) = f(x_1) f(x_2|x_1) = f(x_2) f(x_1|x_2)$$

Example

Discrete case

$f(x_1, x_2)$

$x_1\backslash x_2$	1	2	$f(x_1)$
1	0.1	0.3	0.4
2	0.4	0.2	0.6
$f(x_2)$	0.5	0.5	

$f(x_2\|x_1)$	
$\frac{1}{4}$	$\frac{3}{4}$
$\frac{2}{3}$	$\frac{1}{3}$

$f(x_1\|x_2)$	
$\frac{1}{5}$	$\frac{3}{5}$
$\frac{4}{5}$	$\frac{2}{5}$

$f(x_1) f(x_2)$	
0.2	0.2
0.3	0.3

Continuous case

$$f(x_1, x_2) = 6x_1^2 x_2, \;\; x_1, x_2 \in (0,\, 1)$$

$$f(x_1) = \int_0^1 6x_1^2 x_2 dx_2 = 3x_1^2$$

$$f(x_2) = \int_0^1 6x_1^2 x_2 dx_1 = 2x_2$$

$$f(x_1|x_2) = \frac{6x_1^2 x_2}{2x_2} = 3x_1^2$$

$$f(x_2|x_1) = \frac{6x_1^2 x_2}{3x_1^2} = 2x_2$$

As it is $f(x_1, x_2) = f(x_1) f(x_2)$ the variables are independent.

- **Characteristics**

$$E\left[X\right]=\left[\begin{array}{c} E\left[x_1\right]\\ E\left[x_2\right]\end{array}\right],\ C\left[X\right]=\left[\begin{array}{cc} D\left[x_1\right] & \mathbf{cov}\left[x_1,x_2\right]\\ \mathbf{cov}\left[x_1,x_2\right] & D\left[x_2\right]\end{array}\right]$$

- **Random process** is random variable indexed by time

time \ values	discrete	continuous
discrete	Markov chains	random sequences
continuous	queues	x

- **Categorical distribution**

x	1	2	$\cdots$	n
$f\left(x\right)$	p_1	p_2	$\cdots$	p_n

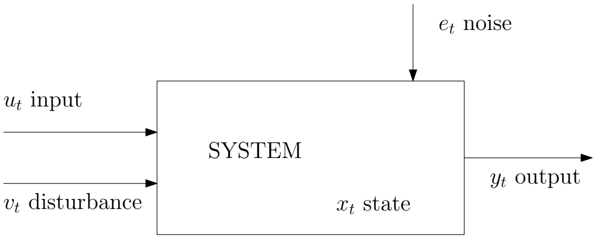
where $p_1 \geq 0$, $\sum p_i = 1$. Each realization has its probability.

- **Normal distribution**

$$f\left(X\right)=\frac{1}{\sqrt{\left(2\pi\right)^n\left|R\right|}}\exp\left\{-\frac{1}{2}\left(x-\mu\right)'R^{-1}\left(x-\mu\right)\right\}$$

1.2 System and its variables

System is a part of reality we are interested in, on which we measure data and which we want to learn about to be able to predict its behavior or influence it.



- output - the modeled variable, after application of the control it can be measured
- input - variable that influences the output and that can be fully manipulated by us
- disturbance - can be measured, cannot be influenced
- state - is influenced by input, influences output, cannot be measured
- noise - can be neither measured nor predicted

Part I

Stochastic Systems

2 Regression and categorical models

2.1 Regression model

$$y_t = \psi_t^{'} \Theta + e_t$$

- y_t modeled variable (output) at time t
- ψ_t regression vector, containing samples of variables influencing the output
- Θ model parameters (regression coefficients θ and noise variance r)
- e_t noise, with zero expectation, constant variance, independent of variables in regression vector = sequence of independent and identically distributed r.v. = i.i.d.

$$\begin{aligned}\psi_t &= [u_t, y_{t-1}, u_{t-1} \cdots y_{t-n}, u_{t-n}, 1]' \\ \theta &= [b_0,\ a_1,\ b_1, \cdots\ a_n,\ b_n,\ k]',\end{aligned}$$

Model in detail

$$y_t = b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + \cdots + a_n y_{t-n} + b_n u_{t-n} + k + e_t$$

Remarks

1. *Number of delayed y and u can be different. Number of delayed y is called* ***model order.***
2. *The term $\psi_t^{'}\theta$ is at time t known constant. Model represents a transformation of e_t to y_t according to the model equation.*
3. *If ψ_t contains no delayed outputs, the model is static. Otherwise, it is dynamic.*
4. *$y_t = \psi_t^{'}\theta$ represents a difference equation.*

A general description of the model as a tool, describing y_t as random variable is model distribution

$$f(y_t|\psi_t, \Theta)$$

Moments of the model are

$$E[y_t|\psi_t, \Theta] = E\left[\psi_t^{'}\theta + e_t\right] = \psi_t^{'}\theta \equiv \hat{y}_t$$

$$D[y_t|\psi_t, \Theta] = D\left[\psi_t^{'}\theta + e_t\right] = D[e_t] = r$$

Normal regression model

$$f(e_t) = \frac{1}{\sqrt{2\pi r}} \exp\left\{-\frac{1}{2r} e_t^2\right\}$$

transformation: $y_t = \hat{y}_t + e_t \rightarrow e_t = y_t - \hat{y}_t$, Jacobian is 1

$$f(y_t|\psi_t, \Theta) = \frac{1}{\sqrt{2\pi r}} \exp\left\{-\frac{1}{2r}\left(y_t - \psi_t^{'}\theta\right)^2\right\}$$

Regression model in the state-space form

The state model is

$$x_t = Mx_{t-1} + Nu_t + w_t.$$

We will demonstrate the transformation for the 2nd order model

$$y_t = b_0u_t + a_1y_{t-1} + b_1u_{t-1} + a_2y_{t-2} + b_2u_{t-2} + k + e_t$$

The state model is

$$\begin{bmatrix} y_t \\ u_t \\ y_{t-1} \\ u_{t-1} \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & k \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ u_{t-1} \\ y_{t-2} \\ u_{t-2} \\ 1 \end{bmatrix} + \begin{bmatrix} b_0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} e_t \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The first row is the regression model, the rest is only one-step time shift.

The advantage of the state-space model lies in recurrent computations. Its memory is only one.

Example

Compute y_3

$$y_1 = b_0u_0 + a_1y_0 + a_2y_{-1}$$
$$y_2 = b_0u_2 + a_1\left(b_0u_0 + a_1y_0 + a_2y_{-1}\right) + a_2y_0$$
$$y_3 = \cdots$$

$$x_1 = Mx_0 + Nu_1$$
$$x_2 = M\left(Mx_0 + Nu_1\right) + Nu_2 = M^2x_0 + MNu_1 + Nu_2$$
$$x_3 = M^3x_0 + M^2Nu_1 + MNu_2 + Nu_3$$

In the state form we even can write a general recurrent formula

$$x_k = M^kx_0 + \sum_{i=2}^{k} M^{k-i}u_i$$

2.2 Categorical model

All variables are discrete - there is a finite number of configurations of data vector $\left[y_t', \psi_t'\right]'$. In the model, each data configuration is assigned its own probability (categorical distribution)

$$f\left(y_t|\psi_t, \Theta\right) = \Theta_{y_t|\psi_t}$$

y_t - output, ψ_t - regression vector, Θ parameter.

For two-valued variables and $\psi_t = \left[u_t', y_{t-1}'\right]'$ the parameters are $\Theta_{y_t|u_t,y_{t-1}}$. The model can be given a form of a table

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
1, 1	$\Theta_{1\vert 11}$	$\Theta_{2\vert 11}$
1, 2	$\Theta_{1\vert 12}$	$\Theta_{2\vert 12}$
2, 1	$\Theta_{1\vert 21}$	$\Theta_{2\vert 21}$
2, 2	$\Theta_{1\vert 22}$	$\Theta_{2\vert 22}$

In the left, there are all configurations of the regression vector. The entries of the table denote all configurations of the data vector, each of them contains its parameter.

It holds:

$$\Theta_{i|jk} \geq 0, \ \sum_i \Theta_{i|jk} = 1, \ \forall jk$$

Remarks

1. *The structure of the model is practically general. It is dynamic and possesses control variable.*
2. *The number of all data configurations is always finite. However, with increasing number of variables and number of values of the variables, its dimension rapidly grows.*

Examples

1. Coin

$y_t = 1$	$y_t = 2$
Θ_1	Θ_2

1. Coin with memory

$$f\left(y_t | y_{t-1}\right), \quad y \in \{1, 2\}$$

y_{t-1}	$y_t = 1$	$y_t = 2$
1	$\Theta_{1\vert 1}$	$\Theta_{2\vert 1}$
1	$\Theta_{1\vert 2}$	$\Theta_{2\vert 2}$

Uncertainty of the regression model is given by the noise variance. Here, it is given by Θ. If its entries are close to 0 or 1, the model is almost deterministic. If they are near to 0.5, the model is very uncertain. E.g.

$$\begin{bmatrix} 0.1, & 0.9 \\ 0.9, & 0.1 \end{bmatrix} \begin{bmatrix} 0.4, & 0.6 \\ 0.6, & 0.4 \end{bmatrix} \text{ or } \begin{bmatrix} 1, & 0 \\ 0, & 1 \end{bmatrix} \begin{bmatrix} 0, & 1 \\ 1, & 0 \end{bmatrix}$$

1. Controlled coin

$$f\left(y_t | u_t\right), \quad y, u \in \{1, 2\}$$

2. Controlled coin with memory

$$f\left(y_t | u_t, y_{t-1}\right), \quad y, u \in \{1, 2\}$$

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
1, 1	0.8	0.2
1, 2	0.7	0.3
2, 1	0.25	0.75
2, 2	0.1	0.9

where y_t mostly obeys u_t

Other examples

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

First: y_t is the bigger from u_t and y_{t-1}, second: y_t is the opposite to u_t.

2.3 Scilab generations

- generate $y \in \{1, 2\}$ so that $P(y = 1) = 0.3$

```
y=(rand(1,1,'u')>0.3)+1      (one value);

y=(rand(1,nd,'u')>0.3)+1      (nd values);
```

- generate $y \in \{1, 2, \cdots, n\}$ so that $P(y = i) = p_i$; $p = [p_1 \cdots p_n]$

```
pp=cumsum(p);

y=sum(rand(1,1,'u')>pp)+1;
```

- number of the row i in the table for combination u_t, $y_{t-1} \in \{1, 2\}$

```
i=2*(u(t)-1)+y(t-1);
```

- generate output y_t from the model $f(y_t | u_t, y_{t-1})$

```
i=2*(u(t)-1)+y(t-1);

pp=cumsum(th(i,:));

y(t)=sum(rand(1,1,'u')>pp)+1;
```

3 Bayes rule

Notation: y_t, $d_t = \{y_t, u_t\}$, $d(t) = \{d_0, d_1, d_2, \cdots, d_t\}$; where d_0 is prior, the rest are measurements.

pdf - probability density function (used for both discrete and continuous random variables).

Bayesian estimation

- classical statistics - parameters are unknown constants
  Bayesian statistics - parameters are random variables (their description is a distribution)
- distributions
  $$\text{model description} \;\; f(y_t|\psi_t, \Theta)$$
  $$\text{parameter description} \;\; f(\Theta|d(t-1)), \; f(\Theta|d(t))$$
- evolution of parameter pdf
  $$f(\Theta|d(0)) \underbrace{\rightarrow}_{d_1=\{u_1,y_1\}} f(\Theta|d(1)) \underbrace{\rightarrow}_{d_2=\{u_2,y_2\}} \cdots \underbrace{\rightarrow}_{d_t=\{u_t,y_t\}} f(\Theta|d(t))$$
- The evolution is governed by the **Bayes rule**
  $$f(\Theta|d(\tau)) \propto f(y_\tau|\psi_\tau, \Theta) f(\Theta|d(\tau-1))$$
  starting from prior pdf $f(\Theta|d(0))$.

Comments

1. Derivation of Bayes rule
   $$\begin{aligned} f(A,B|C) &= f(A|B,C) f(B|C) \\ &= f(B|A,C) f(A|C) \end{aligned}$$
   $\rightarrow$ $\quad f(A|B,C) = \frac{f(B|A,C)f(A|C)}{f(B|C)}$
   where
   $$A \rightarrow \Theta, \;\; B \rightarrow d_t, \;\; C = d(t-1)$$
   and $\{B, C\} = \{d_t, \, d(t-1)\} = d(t)$.
2. Natural conditions of control: The person that estimates also controls. For both actions he uses only information from $d(t-1)$.
   $\rightarrow$
   $f(\Theta|u_t, d(t-1)) = f(\Theta|d(t-1))$ and conversely
   $f(u_t|d(t-1), \Theta) = f(u_t|d(t-1))$
   It applies in estimation with controlled model
   $$f(\Theta|d(t)) \propto f(y_t|\psi_t, \Theta) f(u_t|d(t-1), \cancel{\Theta}) f(\Theta|d(t-1))$$
   which means that $f(u_t|\cdots)$ goes to constant.

3. Self reproducing form of Bayes rule
   B.r. is recursive for functions. To be able to manage functions it is necessary express the pdfs through statistics - e.g. normal distribution is given just by two numbers - expectation and variance, for which the statistics are sum and count. Recursivity requires so that the form of prior pdf (after multiplication by the model) is reproduced in the posterior pdf. E.g. normal pdf → normal pdf, with only statistics recomputed.

   Example (not recursive)

$$f\left(y_t|a\right) = \frac{a}{1+a}\left(\frac{1}{y_t^2} + \exp\left\{-ay_t\right\}\right)$$

   or

$$f\left(y_t|a\right) = \frac{1}{2+\pi a}\left(\sin\left(y_t\right) + a\right)$$

   when computing product of models in measured y_t the number of different terms grows.

   Example (recursive)

$$f\left(y_t|a\right) = a\exp\left\{-ay_t\right\}$$

   Posterior

$$f\left(a|y_1,y_2,y_3\right) \propto a^3\exp\left\{-a\left(y_1+y_2+y_3\right)\right\} =$$
$$a^{\kappa_3}\exp\left\{-aS_3\right\}$$

   where κ and S are statistics, evolving as follows

$$\begin{aligned}\kappa_t &= \kappa_{t-1}+1\\ S_t &= S_{t-1}+y_t\end{aligned}$$

   with initial stats κ_0 and S_0 with the meaning:

   - κ_0 is a virtual number of data samples, from which the prior statistics is constructed.
   - $S_0 = \sum_{i=1}^{\kappa_0} y_i$ from which we have $\bar{y} = \frac{S_0}{\kappa_0}$ i.e. we say that average output is S_0/κ_0.

- Batch estimation
  From Bayes rule it follows

$$f\left(\Theta|d\left(t\right)\right) \propto L_t\left(\Theta\right)f\left(\Theta\right)$$

  where $L_t\left(\Theta\right) = \prod_{\tau=1}^{t} f\left(y_\tau|\psi_\tau,\Theta\right)$ is likelihood and $f\left(\Theta\right) \equiv f\left(\Theta|d\left(0\right)\right)$ is the very prior pdf.

- Results of estimation
  (i) **posterior pdf** $f\left(\Theta|d\left(t\right)\right)$ which brings full information and sometimes can be used as it is - e.g. in prediction

$$f\left(y_t|d\left(t-1\right)\right) = \int_{\Theta^*} f\left(y_t,\Theta|d\left(t-1\right)\right)d\Theta = \int_{\Theta^*} f\left(y_t|\psi_t,\Theta\right)f\left(\Theta|d\left(t-1\right)\right)d\Theta$$

  (ii) **point estimates** computed using posterior pdf

$$\hat{\Theta}_t = E\left[\Theta|d\left(t\right)\right] = \int_{\Theta^*}\Theta f\left(\Theta|d\left(t\right)\right)d\Theta$$

$$\hat{y}_t = E\left[y_t|d\left(t-1\right)\right] = \int_{y^*} y_t\, f\left(y_t|d\left(t-1\right)\right) dy_t =$$

$$\int_{y^*} y_t \left[\int_{\Theta^*} f\left(y_t|\psi_t,\Theta\right) f\left(\Theta|d\left(t-1\right)\right) d\Theta\right] dy_t$$

- Point estimate with **quadratic criterion**

  E.g. for Θ and d - data

  $$J = E\left[\left(\hat{\Theta}-\Theta\right)^2 |d\left(t\right)\right] \to \min$$

  We derive

  $$\min_{\hat{\Theta}} E\left[\left(\hat{\Theta}-\Theta\right)^2 |d\right] = \min_{\hat{\Theta}} E\left[\hat{\Theta^2} - 2\hat{\Theta}\Theta + \Theta^2|d\right] =$$

  $$= \min_{\hat{\Theta}}\left\{\hat{\Theta^2} - 2\hat{\Theta}E\left[\Theta|d\right] + E\left[\Theta^2|d\right]\right\} =$$

  $$= \min_{\hat{\Theta}}\left\{\hat{\Theta^2} - 2\hat{\Theta}E\left[\Theta|d\right] + E\left[\Theta|d\right]^2 \underbrace{-E\left[\Theta|d\right]^2 + E\left[\Theta^2|d\right]}_{D[\Theta]}\right\} =$$

  $$= \min_{\hat{\Theta}}\left\{\hat{\Theta^2} - 2\hat{\Theta}E\left[\Theta|d\right] + E\left[\Theta|d\right]^2\right\} + D\left[\Theta\right] =$$

  $$= \min_{\hat{\Theta}}\left\{\left(\hat{\Theta} - E\left[\Theta|d\right]\right)\right\} + D\left[\Theta\right]$$

  $\to \hat{\Theta} = E\left[\Theta|d\right]$.

4 Estimation of specific models

4.1 Normal static regression model

In this paragraph we are going to tackle estimation of of a static regression model whose pdf is generated by the equation

$$y_t = \theta + e_t \tag{4.1}$$

where θ is a constant and e_t is normal with zero expectation and variance r. Generally the model has two parameters θ and r. As estimation of the noise variance is data demanding and in some tasks it can lead to instability of the process of estimation (e.g. in mixture estimation), sometimes r is chosen fixed and the only parameter to estimate is θ. With r known, the estimation is very easy and the likelihood (as well as the posterior pdf) stay normal.

Direct estimation

For the model equation (4.1), we have

$$E\left[y_t|\theta\right] = E\left[\theta + e_t\right] = E\left[\theta\right] = \theta$$
$$D\left[y_t|\theta\right] = D\left[\theta + e_t\right] = D\left[e_t\right] = r$$

because θ is a constant and expectation of a constant is the constant itself and variance of a constant is zero.

Now, from the classical statistics we know that expectation is estimated by sample average and variance by sample variance. So, it holds

$$\hat{\theta} = \frac{1}{N}\sum_{t=1}^{N} y_t = \bar{y}, \text{ and } \hat{r} = \frac{1}{N-1}\sum_{t=1}^{N}\left(y_t - \bar{y}\right)^2 = s_y^2$$

Maximum likelihood estimation

The model pdf with the equation (4.1) is

$$f\left(y_t|\theta, r\right) = \frac{1}{\sqrt{2\pi}} r^{-0.5} \exp\left\{-\frac{1}{2r}\left(y_t - \theta\right)^2\right\} \propto$$
$$\propto r^{-0.5} \exp\left\{-\frac{1}{2r}\left(y_t^2 - 2y_t\theta + \theta^2\right)\right\}$$

where we omitted the pdf constant and applied the second power.

Now, the likelihood is a product of models for y_1,y_2, etc. It is easy to write

$$L_N = \prod_{t=1}^{N} f\left(y_t|\theta, r\right) \propto \left(r^{-0.5}\right)^N \exp\left\{-\frac{1}{2r}\sum_{t=1}^{N}\left(y_t^2 - 2y_t\theta + \theta^2\right)\right\} =$$
$$= r^{-0.5N} \exp\left\{-\frac{1}{2r}\left(\sum_{t=1}^{N} y_t^2 - 2\theta\sum_{t=1}^{N} y_t + N\theta^2\right)\right\}$$

where we used the fact, that $\prod_t \exp\{A_t\} = \exp\{\sum_t A_t\}$.

So, we can introduce the statistics

$$\begin{aligned} \kappa_t &= \kappa_{t-1} + 1 \;\; \text{data counter} \\ S_t &= S_{t-1} + y_t \; \text{sum of } y \\ R_t &= R_{t-1} + y_t^2 \; \text{sum of squares of } y \end{aligned}$$

and with them to write the likelihood as follows

$$L_N = r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\left(R_N - 2\theta S_N + \kappa_N \theta^2\right)\right\} \tag{4.2}$$

The estimates $\hat{\theta}$ and $\hat{r}$ of the parameters θ and r can be found as arguments maximizing the likelihood L_N.

First, we differentiate with respect to θ:

$$\frac{dL}{d\theta} \propto r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\left(R_N - 2\theta S_N + \kappa_N \theta^2\right)\right\}\left[-\frac{1}{2r}\left(-2S_N + 2\theta\kappa_N\right)\right] = 0$$

from which

$$S_N = \theta\kappa_N \rightarrow \hat{\theta} = \frac{S_N}{\kappa_N} = \frac{\sum_{t-1}^N y_t}{N} = \bar{y}$$

Now, we substitute $\theta = \bar{y}$ and differentiate with respect to r

$$\begin{aligned} \frac{dL}{dr} &\propto \frac{d}{dr} r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right\} = \\ &= \frac{d}{dr}\left(r^{-0.5\kappa_N}\right) \times \exp\left\{-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right\} + \\ &+ r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right\} \times \frac{d}{dr}\left[-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right] = \\ &= -0.5\kappa_N r^{-0.5\kappa_N - 1} \times \exp\left\{-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right\} + \\ &+ r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)\right\} \times \frac{1}{2r^2}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right) = 0 \end{aligned}$$

From it

$$\kappa_N r^{-1} = \frac{1}{r^2}\left(R_N - 2\bar{y}S_N + \kappa_N \bar{y}^2\right)$$

and

$$\hat{r} = \frac{R_N}{\kappa_N} - 2\bar{y}\frac{S_N}{\kappa_N} + \bar{y}^2 = \frac{\sum_{t=1}^N y_t^2}{\kappa_N} - \left(\frac{\sum_{t=1}^N y_t}{\kappa_N}\right)^2 = \overline{y^2} - \bar{y}^2 = s_y^2$$

where we used the formula $D\left[A\right] = E\left[A^2\right] = \left(E\left[A\right]\right)^2$ known from the classical statistics.

Thus we have obtained the same results as in the paragraph Direct estimation.

Another derivation

The derivation demonstrated is rather long and cumbersome. It can be improved in the following way. We take the quadratic form $\left(R_N - 2\theta S_N + \kappa_N \theta^2\right)$ in the exponent of the likelihood (4.2) and complete it to the square

$$\left(R_N - 2\theta S_N + \kappa_N \theta^2\right) = \kappa_N \left(\theta^2 - 2\theta \frac{S_N}{\kappa_N} + \frac{R_N}{\kappa_N}\right) =$$

$$= \kappa_N \left(\theta^2 - 2\theta \frac{S_N}{\kappa_N} + \left(\frac{S_N}{\kappa_N}\right)^2 - \left(\frac{S_N}{\kappa_N}\right)^2 + \frac{R_N}{\kappa_N}\right) =$$

$$= \kappa_N \left[\left(\theta - \frac{S_N}{\kappa_N}\right)^2 + \left(\frac{R_N}{\kappa_N} - \left(\frac{S_N}{\kappa_N}\right)^2\right)\right] =$$

$$= \kappa_N \left[(\theta - \bar{y})^2 + \underbrace{\overline{y^2} - \bar{y}^2}_{s_y^2}\right].$$

Then the likelihood (4.2) is

$$L_N = r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\kappa_N \left[(\theta - \bar{y})^2 + s_y^2\right]\right\}.$$

To maximize it according to θ we look for minimum of the quadratic form which is evidently achieved for $\hat{\theta} = \bar{y}$ and we get the partially maximized likelihood

$$L_N = r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\kappa_N s_y^2\right\}.$$

Now we differentiate according to r (in the same way as we already did)

$$\frac{dL}{dr} = -0.5\kappa_N r^{-0.5\kappa_N - 1} \exp\left\{-\frac{1}{2r}\kappa_N s_y^2\right\} + r^{-0.5\kappa_N} \exp\left\{-\frac{1}{2r}\kappa_N s_y^2\right\} \frac{1}{2r^2}\kappa_N s_y^2$$

from which we obtain the result

$$\hat{r} = s_y^2.$$

Both the results correspond those previously derived.

The resulting lesson

To estimate the static regression model (even in the multivariate form for vector y_t) the regression coefficients can be estimated as the averages of the variables entering the regression and the covariance matrix of noise as their sample covariance matrix.

Example

Let us have the model

$$y_t = \begin{bmatrix} y_{1;t} \\ y_{2;t} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} e_{1;t} \\ e_{2;t} \end{bmatrix}$$

with the dataset

y_1	2.3	3.1	5.6	4.2	8.1	1.3
y_2	12.5	18.3	15.7	17.1	16.5	14.7

Estimate its parameters $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ and $r = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$.

We have

$$\hat{\theta} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \end{bmatrix} = \begin{bmatrix} 4,1 \\ 15.8 \end{bmatrix}$$

and

$$\hat{r} = \begin{bmatrix} \mathrm{var}\,(y_1) & \mathrm{cov}\,(y_1, y_2) \\ \mathrm{cov}\,(y_1, y_2) & \mathrm{var}\,(y_2) \end{bmatrix} = \begin{bmatrix} 5.06, & 1.55 \\ 1.55 & 3.42 \end{bmatrix}$$

And that is all :-)

4.2 Normal regression model

Model

$$f\left(y_t | \psi_t, \Theta\right) = \frac{1}{\sqrt{2\pi}} r^{-0.5} \exp\left\{ -\frac{1}{2r} \left(y_t - \psi_t^{'} \theta \right)^2 \right\}$$

For 1st order $y_t = bu_t + ay_{t-1} + e_t$ it is $\psi_t = \left[u_t,\, y_{t-1}\right]'$. The square in the exponent can be written

$$\left(y_t - bu_t - ay_{t-1}\right)\left(y_t - bu_t - ay_{t-1}\right) =$$

$$= (-1)\left[-1,\, b,\, a\right] \begin{bmatrix} y_t \\ u_t \\ y_{t-1} \end{bmatrix} (-1)\left[y_t,\, u_t,\, y_{t-1}\right] \begin{bmatrix} -1 \\ b \\ a \end{bmatrix} =$$

$$= \left[-1,\, \theta'\right] \underbrace{\begin{bmatrix} y_t \\ \psi_t \end{bmatrix} \left[y_t,\, \psi_t^{'}\right]}_{D_t} \begin{bmatrix} -1 \\ \theta \end{bmatrix}$$

where D_t is data matrix.

Model (in modification)

$$f\left(y_t | \psi_t, \Theta\right) \propto r^{-0.5} \exp\left\{ -\frac{1}{2r} \left[-1,\, \theta'\right] D_t \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\}$$

Prior pdf

In the same form as model

$$f\left(\Theta | d\left(0\right)\right) \propto r^{-0.5\kappa_0} \exp\left\{ -\frac{1}{2r} \left[-1,\, \theta'\right] V_0 \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\}$$

Bayes

$$f\left(\Theta | d\left(1\right)\right) \propto r^{-0.5} \exp\left\{ -\frac{1}{2r} \left[-1,\, \theta'\right] D_t \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\} r^{-0.5\kappa_0} \exp\left\{ -\frac{1}{2r} \left[-1,\, \theta'\right] V_0 \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\} =$$

$$= r^{-0.5\kappa_1} \exp\left\{ -\frac{1}{2r} \left[-1,\, \theta'\right] V_1 \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\}$$

Posterior pdf

$$f\left(\Theta|d\left(t\right)\right) \propto r^{-0.5\kappa_t} \exp\left\{-\frac{1}{2r}\left[-1,\,\theta'\right]V_t\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}$$

Recursion for statistics

$$\begin{aligned}\kappa_t &= \kappa_{t-1}+1\\ V_t &= V_{t-1}+D_t\end{aligned}$$

with κ_0 and V_0 as prior statistics.

Results

(a) **Posterior** - GiW with statistics κ_t and V_t.

(b) **Point estimates** of parameters

$$V_t = \left[\begin{array}{cc} V_y & V_{y\psi}\\ V_{y\psi} & V_\psi\end{array}\right]\cdots\left[\begin{array}{cc}\bullet & --\\ | & \square\end{array}\right]$$

$$\hat{\theta}_t = V_\psi^{-1}V_{y\psi} \quad \text{regression coefficients}$$

$$\hat{r}_t = \frac{V_y - V_{y\psi}'V_\psi^{-1}V_{y\psi}}{\kappa_t} \quad \text{noise variance}$$

Point estimate of output

$$\hat{y}_t = \psi_t\hat{\theta}_{t-1} \qquad (\theta \to \hat{\theta}_{t-1},\ e_t \to 0)$$

Batch estimation

$$y_t = b_0u_t + \cdots a_ny_{t-n} + b_nu_{t-n} + k + e_t$$

for $t = 1, 2, \cdots, N$

$$\begin{gathered}y_1 = b_0u_1 + \cdots a_ny_{1-n} + b_nu_{1-n} + k + e_1\\ y_2 = b_0u_2 + \cdots a_ny_{2-n} + b_nu_{2-n} + k + e_2\\ \cdots\\ y_N = b_0u_N + \cdots a_ny_{N-n} + b_nu_{N-n} + k + e_N\end{gathered}$$

→ matrix form

$$Y = X\theta + E$$

where (for $n = 1$) Y and X are

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}, \quad X = \begin{bmatrix} u_1 & y_0 & u_0 & 1 \\ u_2 & y_1 & u_1 & 1 \\ \cdots & \cdots & \cdots & \cdots \\ u_N & y_{N-1} & u_{N-1} & 1 \end{bmatrix}$$

Optimization - least squares

$$J = \sum e_i^2 = E'E = (Y - X\theta)' (Y - X\theta) = Y'Y - 2\theta' X'Y + \theta' X'X\theta$$

$$\frac{\partial}{\partial \theta} J = -2X'Y + 2X'X\theta$$

$$X'X\theta = X'Y \quad \rightarrow \quad \hat{\theta}_t = (X'X)^{-1} X'Y$$

4.3 Categorical model

Product form of the model

$$f(y_t|\psi_t, \Theta) = \Theta_{y_t|\psi_t} = \prod_{y|\psi} \Theta_{y|\psi}^{\delta(y|\psi;\, y_t|\psi_t)}$$

i.e. product over all possible configurations of $y|\psi$; but only $y_t|\psi_t$ is chosen.

Prior / posterior pdf

$$f(\Theta|d(t)) \propto \prod_{y|\psi} \Theta_{y|\psi}^{\nu_{y|\psi;t}}$$

where $\nu_{y|\psi;t}$ for all configurations of $y|\psi$ is statistics; $\nu_{y|\psi;0}$ is the prior one.

Statistics update

From Bayes rule

$$\nu_{y|\psi;t} = \nu_{y|\psi;t-1} + \delta(y|\psi;\, y_t|\psi_t)$$

for all configurations of $y|\psi$ (or $\nu_{y_t|\psi_t;t} = \nu_{y_t|\psi_t;t-1} + 1$ for actual data)

Point estimate

$$\hat{\theta}_{y|\psi;t} = \frac{\nu_{y|\psi;t}}{\sum_i \nu_{i|\psi;t}}$$

which is normalization of the statistic matrix in rows.

Example

Model (of a coin)

$$f\left(y|p\right) = p_y, \;\; y = 1, 2\,; \, p = [p_1, \, p_2]'$$

Product form

$$f\left(y|p\right) = p_1^{\delta(y,1)} p_2^{\delta(y,2)}$$

Posterior

$$f\left(p|d\left(t\right)\right) \propto p_1^{\nu_{1;t}} p_2^{\nu_{2;t}}$$

Statistics

$$\nu_t = [\nu_{1;t}, \, \nu_{2;t}]$$

Update

– for $y = 1$

$$\nu_{1;t} = \nu_{1;t-1} + 1$$

– for $y = 2$

$$\nu_{2;t} = \nu_{2;t-1} + 1$$

For the data

t	1	2	3
y_t	1	1	2

and zero initial statistics

t	0	1	2	3
ν_1	0	1	2	2
ν_2	0	0	0	1
p_1	x	1	1	$\frac{2}{3}$
p_2	x	0	0	$\frac{1}{3}$

With initial statistics 10

t	0	1	2	3
ν_1	10	11	12	12
ν_2	10	10	10	11
p_1	x	0.524	0.546	0.522
p_2	x	0.476	0.454	0.478

The ratio $\frac{\nu_{1;0}}{\nu_{1;0}+\nu_{2;0}}$ expresses the value of p_1

The magnitude of ν expresses our belief in .our guess.

Output estimate

$$f\left(y_t|d\left(t-1\right)\right) = f\left(y_t|\psi_t, \Theta = \hat{\Theta}_{t-1}\right)$$

y_t	1	2	3	$\cdots$	n
$f\left(y_t\vert\psi_t\hat{\Theta}_{t-1}\right)$	$P\left(y_t = 1\right)$	$P\left(y_t = 2\right)$	$P\left(y_t = 3\right)$		$P\left(y_t = n\right)$

Point estimate is the value of y_t with the biggest probability.

5 Output prediction

Estimation of the future output y_{t+k} with the knowledge of data up to $t-1$ for $k=0,1,\cdots$. For $k=0$ we speak about output estimation or zero-step prediction.

5.1 Output estimation

It can be also called zero-step prediction. Here we estimate y_t at time t which, however, was not yet measured. E.g. for the 1st order regression model without control variable $f\left(y_t|y_{t-1},\Theta\right)$

$$f\left(y_t|y\left(t-1\right)\right)=\int_{\Theta^*}f\left(y_t,\Theta|y\left(t-1\right)\right)d\Theta=$$

$$\begin{aligned}(i)\quad &= \int_{\Theta^*}f\left(y_t|y_{t-1},\Theta\right)f\left(\Theta|y\left(t-1\right)\right)d\Theta \ \text{ posterior of } \Theta\\ (ii)\quad &\doteq f\left(y_t|y_{t-1},\hat{\Theta}_{t-1}\right)\ \text{ point estimate of } \Theta\end{aligned}$$

where (ii) is achieved by replacing $f\left(\Theta|y\left(t-1\right)\right)\rightarrow\delta\left(\Theta,\hat{\Theta}_{t-1}\right)$ and

$$\int_{\Theta^*}f\left(y_t|y_{t-1},\Theta\right)f\left(\Theta|y\left(t-1\right)\right)d\Theta\doteq$$

$$\doteq\int_{\Theta^*}f\left(y_t|y_{t-1},\Theta\right)\delta\left(\Theta,\hat{\Theta}_{t-1}\right)d\Theta=f\left(y_t|y_{t-1},\hat{\Theta}_{t-1}\right)$$

where $\hat{\Theta}_{t-1}=E\left[\Theta|y\left(t-1\right)\right]=\int_{\Theta^*}\Theta f\left(\Theta|y\left(t-1\right)\right)d\Theta$ is point estimate of Θ based on the data $y\left(t-1\right)$.

Remarks

1. *In $f\left(y_t|y\left(t-1\right)\right)$ the parameter Θ is missing. We need to supply it.*
2. *This type of prediction is useful for validation of estimated model. After prediction $\hat{y}_t$ we measure the value of the true output $y_t$and compute prediction error $\hat{e}_t=y_t-\hat{y}_t$, $\forall t$. Good model should have the sum of prediction errors small.*

5.2 One step prediction

$$f\left(y_{t+1}|y\left(t-1\right)\right)=\int_{\Theta^*}\int_{y_t^*}f\left(y_{t+1},y_t,\Theta|y\left(t-1\right)\right)dy_td\Theta=$$

$$\begin{aligned}(i)\quad &= \int_{\Theta^*}\int_{y_t^*}f\left(y_{t+1}|y\left(t\right),\Theta\right)f\left(y_t|y_{t-1},\Theta\right)f\left(\Theta|y\left(t-1\right)\right)dy_td\Theta\\ (ii)\quad &\doteq f\left(y_{t+1}|\hat{y}_t,\hat{\Theta}_{t-1}\right)\end{aligned}$$

where for (ii) we lay $f\left(\Theta|y\left(t-1\right)\right)\rightarrow\delta\left(\Theta,\hat{\Theta}_{t-1}\right)$ and $f\left(y_t|y\left(t-1\right)\right)\rightarrow\delta\left(y_t,\hat{y}_t\right)$ with $\hat{\Theta}_{t-1}$ and $\hat{y}_t$ being point estimates.

Remark

- *Here, both Θ and y_t are missing. We must supply both.*
- *Comparing (i) and (ii) we can see the basic principle of Bayesian estimation. Basically, value of the missing unknown variable (Θ and y_t) is substituted (into the pdfs) and it is weighted by its probability (prior pdf + integration). In the second variant (ii) first point estimates are computed and then substituted for the unknown variables.*

5.3 Multi-steps prediction

Regression model with known parameters and point estimation

For a 1st order regression model $y_t = ay_{t-1}+bu_t+e_t$ with known parameters and point prediction we have

$$\begin{aligned}
y_t &= ay_{t-1}+bu_t+e_t \\
\hat{y}_t &= ay_{t-1}+bu_t \\
\hat{y}_{t+1} &= a\hat{y}_t+bu_{t+1} = a\left(ay_{t-1}+bu_t\right)+bu_{t+1} = \\
&= a^2y_{t-1}+abu_t+bu_{t+1} \\
\hat{y}_{t+2} &= a\hat{y}_{t+1}+bu_{t+2} = \\
&= a^3y_{t-1}+a^2bu_t+abu_{t+1}+bu_{t+2} \\
& etc.
\end{aligned}$$

The point prediction can be achieved by a simple repetitive substitution of the model. For simulation, directly last estimates can be used.

Full prediction with regression model under condition of normality

Prediction with normal model with known parameters preserves normality. If e_t is normal, all predictions are normal, too.

$$\begin{aligned}
y_t &= ay_{t-1}+bu_t+e_t \\
y_{t-1} &= ay_t+bu_{t+1}+e_{t+1} = \\
&= a\left(ay_{t-1}+bu_t+e_t\right)+bu_{t+1}+e_{t+1} = \\
&= a^2y_{t-1}+abu_t+bu_{t+1}+ae_t+e_{t+1} \\
y_{t+2} &= ay_{t+1}+bu_{t+2}+e_{t+2} = \\
&= a^3y_{t-1}+a^2bu_t+abu_{t+1}+bu_{t+2}+a^2e_t+ae_{t+1}+e_{t+2}
\end{aligned}$$

$\rightarrow$

$E\left[y_{t+2}|y\left(t-1\right)\right]=a^{3}y_{t-1}+a^{2}bu_{t}+abu_{t+1}+bu_{t+2}$
$D\left[y_{t+2}|y\left(t-1\right)\right]=D\left[a^{2}e_{t}+ae_{t+1}+e_{t+2}\right]=\left(a^{4}+a^{2}+1\right)r$
Predictive pdf

$$f\left(y_{t+2}|y\left(t-1\right)\right)=N_{y_{t+2}}\left(E\left[y_{t+2}|y\left(t-1\right)\right],\,D\left[y_{t+2}|y\left(t-1\right)\right]\right)$$

Remark

- Normal distribution is preserved during computation (prove it[1]).
- Normal distribution is determined by its expectation and variance.

5.4 Prediction with discrete model

For a model $f\left(y_{t}|y_{t-1},\Theta\right)$ we have

Zero step prediction

It is given directly by the model

$$f\left(y_{t}|y_{t-1},\Theta\right)=\Theta_{y_{t}|y_{t-1}}$$

Multi-steps prediction

For the model in a form of square table (matrix), the prediction is

$$f\left(y_{t+k}|y\left(t-1\right)\right)=\left(\Theta^{k+1}\right)_{y_{t+k}|y_{t-1}}$$

Example

Two steps prediction

$$\begin{aligned}f\left(y_{t+2}|y\left(t-1\right)\right)&=\sum_{y_{t+1}}\sum_{y_{t}}f\left(y_{t+2}|y_{t-1}\right)f\left(y_{t+1}|y_{t}\right)f\left(y_{t}|y_{t-1}\right)=\\&=\sum_{y_{t+1}}\Theta_{y_{t+2}|y_{t+1}}\sum_{y_{t}}\Theta_{y_{t+1}|y_{t}}\Theta_{y_{t}|y_{t-1}}=\left(\Theta^{3}\right)_{y_{t+2}|y_{t-1}}\end{aligned}$$

For square

$$\Theta=\left[\begin{array}{cc}0.4, & 0.6\\0.8, & 0.2\end{array}\right]$$

$$f\left(y_{t+2}|y\left(t-1\right)\right)=\left[\begin{array}{cc}0.4, & 0.6\\0.8, & 0.2\end{array}\right]^{3}=\left[\begin{array}{cc}0.544, & 0.456\\0.608, & 0.392\end{array}\right]$$

→

for $y_{t-1}=1$ we have $f\left(y_{t+2}|1\right)=\left[0.544,\,0.456\right]$

for $y_{t-1}=2$ we have $f\left(y_{t+2}|2\right)=\left[0.608,\,0.392\right]$

Point prediction either can be constructed either as MAP prediction or it can be computed from the estimated model as if in simulation.

[1] The computations are based on square completion in exponents of Gaussian densities.

6 State-space model, state estimation

6.1 Model

$$\begin{array}{ll} f\left(x_t|x_{t-1},u_t\right) & \text{model of the state} \\ f\left(y_t|x_t,u_t\right) & \text{model of the output} \end{array}$$

is generated by the equations

$$\begin{aligned} x_t &= Mx_{t-1}+Nu_t+w_t \\ y_t &= Ax_t+Bu_t+v_t \end{aligned}$$

where M, N, A, B are matrices, w_t and v_t white noises with covariance matrices r_w and r_v.

6.2 Estimation

State description

$$f\left(x_{t-1}|d\left(t-1\right)\right) \underbrace{\rightarrow}_{\text{prediction}} f\left(x_t|d\left(t-1\right)\right) \underbrace{\rightarrow}_{\text{filtration}} f\left(x_t|d\left(t\right)\right)$$

Evolution

$$\begin{aligned} f\left(x_t|d\left(t-1\right)\right) &= \int_{x_{t-1}^*} f\left(x_t|x_{t-1},u_t\right) f\left(x_{t-1}|d\left(t-1\right)\right) \quad \text{prediction} \\ f\left(\underbrace{x_t}_{\Theta}|d\left(t\right)\right) &\propto \underbrace{f\left(y_t|x_t,u_t\right)}_{\text{model}} f\left(\underbrace{x_t}_{\Theta}|d\left(t-1\right)\right) \quad \text{Bayes} \end{aligned}$$

! In the above derivation Natural Conditions of Control are used !

Kalman filter

For normal model and normal prior state distribution the normality is preserved. Functional recursion becomes algebraic one for expectations and covariance matrices.

Notation

$$\begin{aligned} f\left(x_t|x_{t-1},u_t\right) &= N_{x_t}\left(Mx_{t-1}+Nu_t,\, r_w\right) \\ f\left(y_t|x_t,u_t\right) &= N_{y_t}\left(Ax_t+Bu_t,\, r_v\right) \end{aligned}$$

and

$$\begin{aligned} f\left(x_{t-1}|d\left(t-1\right)\right) &= N_{x_{t-1}}\left(x_{t-1|t-1},\, R_{t-1|t-1}\right) \\ f\left(x_t|d\left(t-1\right)\right) &= N_{x_t}\left(x_{t|t},\, R_{t|t}\right) \\ f\left(x_t|d\left(t\right)\right) &= N_{x_t}\left(x_{t|t},\, R_{t|t}\right) \end{aligned}$$

Substitution into the evolution equations gives Kalman filter (KF)

Kalman filter

$x_{t|t-1} = Mx_{t-1|t-1} + Nu_t$ state prediction

$R_{t|t-1} = r_x + MR_{t-1|t-1}M'$

$y_p = Ax_{t|t-1} + Bu_t$ output prediction

$R_p = r_y + AR_{t|t-1}A'$

$R_{t|t} = R_{t|t-1} - R_{t|t-1}A'R_p^{-1}AR_{t|t-1}$

$K = R_{t|t}A'r_y^{-1}$ Kalman gain

$x_{t|t} = x_{t|t-1} + K\left(y_t - y_p\right)$ state correction

The filter starts with prior $x_{0|0}$ and $R_{0|0}$, uses data y_t, u_t, $t = 1, 2, \cdots, N$ and currently computes $x_{t|t}$ and $R_{t|t}$. The result is either point state estimate $x_{t|t}$ or the full distribution of the state $f\left(x_t|u_t, d\left(t\right)\right) = N_{x_t}\left(x_{t|t},\, R_t\right)$.

Program together with subroutine Kalman.sci is here

```
clc, clear, close
[u,t,n]=file();               // find working directory
chdir(dirname(n(1)));         // set working directory
getd('.')

nd=100;
M=[.5 .4; .3 .6];
N=[1; 1];
A=[.4 1.2];
Rw=10*eye(2,2);
Rv=1;

// Simulace
xt=zeros(2,nd);
xt(:,1)=50*ones(2,1);
ut=rand(1,nd,'n');
yt=zeros(1,nd);
for t=1:nd
  xt(:,t+1)=M*xt(:,t)+N*ut(t)+Rw*rand(2,1,'n');
  yt(t)=A*xt(:,t)+Rv;
end

// Filtration
Rx=1e3*eye(2,2);
x=zeros(2,1);
for t=1:nd
```

```
  [x,Rx,yp]=Kalman(x,yt(t),ut(t),M,N,A,0,Rw,Rv,Rx);
  xe(:,t+1)=x;
end

// Results
s=1:nd;
scf(1);
subplot(211)
plot(s,xt(1,s)',s,xe(1,s)','.')
subplot(212)
plot(s,xt(2,s)',s,xe(2,s)','.')
```

and the subroutine

```
function [xt,Rx,yp]=Kalman(xt,yt,ut,M,N,A,B,Rw,Rv,Rx)
  // [xt,xf,Rx,yp]=Kalman(xt,yt,ut,M,N,A,B,Rw,Rv,Rx)
  // Kalman filter for state estimtion with the model
  //                             xt = M*xt + N*ut + w
  //                             yp = A*xt + B*ut + e
  // xt    state
  // Rx    state estimate covariance matrix
  // yp    output prediction
  // yt    output
  // ut    input
  // M,N   state model parameters
  // A,B   output model parameters
  // Rw    state model covariance
  // Rv    output model covariance

  // prediction
  xt=M*xt+N*ut;                        // rime update of the state
  Rx=Rw+M*Rx*M';                       // time updt. of state covariance

  // filtration
  yp=A*xt+B*ut;                        // output prediction
  Ry=Rv+A*Rx*A';                       // noise covariance update
  Rx=Rx-Rx*A'*inv(Ry)*A*Rx;            // state est. coariance update
  ey=yt-yp;                            // prediction error
  KG=Rx*A'*inv(Rv);                    // Kalman gain
  xt=xt+KG*ey;                         // data update of the state
endfunction
```

7 Nonlinear state estimation

7.1 Nonlinear model

$$x_t = g\left(x_{t-1}, u_t\right) + w_t$$
$$y_t = h\left(x_t, u_t\right) + v_t$$

Example

For

$$x_t = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t, \quad u_t, \quad y_t$$

the model is

$$\begin{aligned} x_{1;t} &= \exp\left\{-x_{1;t-1} - x_{2;t-1}\right\} + u_t + w_t \\ x_{2;t} &= x_{1;t-1} - 0.3u_t + w_{2;t} \\ y_t &= x_{2;t} + v_t \end{aligned}$$

Linearization

Is done using first two terms of Taylor expansion of nonlinear functions at the point of last point estimate. For the state equation it is $\hat{x}_{t-1}$ and for the output equation it is $\hat{x}_t$.

Generally, i.e. for a general value x the expansion reads

$$g\left(x, u_t\right) \doteq g\left(\hat{x}_{t-1}, u_t\right) + g'\left(\hat{x}_{t-1}, u_t\right)\left(x - \hat{x}_{t-1}\right)$$
$$h\left(x, u_t\right) \doteq h\left(\hat{x}_t, u_t\right) + h'\left(\hat{x}_t, u_t\right)\left(x - \hat{x}_t\right)$$

Remarks

1. x_t and x_{t-1} are random variables. x is their general value, $\hat{x}_t$ and $\hat{x}_{t-1}$ are special values: $\hat{x}_t$ is the point estimate of x_t and $\hat{x}_{t-1}$ is point estimate of x_{t-1}.
2. *Linearization can be applied only to nonlinear parts of the model. The linear parts can stay as they are.*

The derivatives g' and h' are

$$g'\left(\hat{x}_{t-1}, u_t\right) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ & & \cdots & \\ \frac{\partial g_n}{\partial x_1} & & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix}_{|x=\hat{x}_{t-1}}, \quad h'\left(\hat{x}_t, u_t\right) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ & & \cdots & \\ \frac{\partial h_m}{\partial x_1} & & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}_{|x=\hat{x}_t}$$

After substitution the linearization into the model, we have

and for $x = \hat{x}_{t-1}$ in the case of the state equation and $x = \hat{x}_t$ for output equation we obtain the linearized model

$$\begin{aligned} x_t &= \bar{M}x_{t-1} + F + w_t \\ y_t &= \bar{A}x_t + G + v_t \end{aligned}$$

where

$$\bar{M} = g'\left(\hat{x}_{t-1}, u_t\right), \qquad F = g\left(\hat{x}_{t-1}, u_t\right) - g'\left(\hat{x}_{t-1}, u_t\right)\hat{x}_{t-1},$$
$$\bar{A} = h'\left(\hat{x}_t, u_t\right), \qquad G = h\left(\hat{x}_t, u_t\right) - h'\left(\hat{x}_t, u_t\right)\hat{x}_t.$$

Example (continuation) - $\cdots$ only first equation is nonlinear

$$g_1\left(x, u_t\right) = \exp\left\{-x_1 - x_2\right\} + u_t$$

$$g_1^{'}\left(x, u_t\right) = \left[\frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_2}\right] = \left[-\exp\left\{-x_1 - x_2\right\}, -\exp\left\{-x_1 - x_2\right\}\right]$$

Fully linearized model is

$$\begin{aligned} x_{1;t} &= g_1^{'}\left(\hat{x}_{t-1}, u_t\right)x_{t-1} + g_1\left(\hat{x}_{t-1}, u_t\right) - g_1^{'}\left(\hat{x}_{t-1}, u_t\right)\hat{x}_{t-1} + w_t \\ x_{2;t} &= [1,\,0]\,x_{t-1} - 0.3u_t + w_{2;t} \\ y_t &= [0,\,1]\,x_t + v_t \end{aligned}$$

where

$$\bar{M} = \left[\begin{array}{c} g_1^{'}\left(\hat{x}_{t-1}, u_t\right) \\ [1,0] \end{array}\right], \quad F = \left[\begin{array}{c} g_1\left(\hat{x}_{t-1}, u_t\right) - g_1^{'}\left(\hat{x}_{t-1}, u_t\right)\hat{x}_{t-1} \\ -0.3u_t \end{array}\right],$$

$$N = \left[\begin{array}{c} 0 \\ 0 \end{array}\right], \quad \bar{A} = [0,\,1], \quad G = 0, \quad B = 0.$$

With this, we can use subroutine Kalman

[xt,Rx,yp]=Kalman(xt,yt,ut,$\bar{M}$,N,F,$\bar{A}$,B,G,Rw,Rv,Rx)

7.2 Model with unknown parameters

The unknown parameters of the model are added to the state a and estimated. However, the model becomes nonlinear - model matrices contain state entries and they are multiplied by state. So, the technique of linearization must be used, again.

Example

Model

$$\begin{aligned} x_t &= \exp\left\{-a x_{t-1}\right\} + b u_t + w_t \\ y_t &= x_t + v_t, \end{aligned}$$

where a and b are unknown.

We define new state

$$X_t = \left[x_t^{'},\, a,\, b\right]', \quad X_{t-1} = \left[x_{t-1}^{'},\, a,\, b\right]'$$

and obtain new model

$$\begin{aligned} X_t &= \begin{bmatrix} \exp\{-X_{2;t-1}X_{1;t-1}\} + X_{3;t-1}u_t \\ X_{2;t-1} \\ X_{3;t-1} \end{bmatrix} + \underbrace{\begin{bmatrix} w_t \\ \epsilon_{2;t} \\ \epsilon_{3;t} \end{bmatrix}}_{W_t} \\ y_t &= [1,\, 0,\, 0]\, X_t + v_t \end{aligned}$$

Only the first equation is nonlinear, however, we will treat the whole model as nonlinear (it is well possible)

$$g = \begin{bmatrix} \exp\{-X_{2;t-1}X_{1;t-1}\} + X_{3;t-1}u_t \\ X_{2;t-1} \\ X_{3;t-1} \end{bmatrix}_{X_{t-1}=\hat{X}_{t-1}}$$

$$g' = \begin{bmatrix} -X_{2;t-1}\exp\{-X_{2;t-1}X_{1;t-1}\}\,, & -X_{1;t-1}\exp\{-X_{2;t-1}X_{1;t-1}\}\,, & u_t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{X_{t-1}=\hat{X}_{t-1}}$$

model

$$\begin{aligned} X_t &= \underbrace{g'}_{\bar{M}}\, X_{t-1} + \underbrace{g - g'\hat{X}_{t-1}}_{F} + W_t \\ y_t &= \underbrace{[1,\, 0,\, 0]}_{\bar{A}}\, X_t + v_t \end{aligned}$$

and $N = [0,\, 0,\, 0]'$, $B = 0$, $G = 0$.

[x, Rx, yp] = Kalman(x, y, u, $\bar{M}$, N, F, $\bar{A}$, B, G, Rw, Rv, Rx)

8 Control with regression model

8.1 Derivation in pdf

Criterion

Optimal control needs criterion. We will use summation one

$$J = \sum_{t=1}^{N} J_t$$

where J_t is a penalization for time t. Mostly it is $J_t = y_t^2 + \omega u_t^2$.

We want to set $u_t,\ t = 1, 2, \cdots, N$ that minimizes J. But, J is a random variable, due to the output y_t. As random variable can take many different values it is not possible to speak about its minimization. So, we must minimize its estimate (which is expectation). So the minimized criterion is

$$E\left[J|d\left(0\right)\right] = E\left[\sum_{t=1}^{N} J_t|d\left(0\right)\right]$$

where in condition of the expectation is our preliminary knowledge - prior data.

Remark

For $N = 1$ we obtain one-step control. Here, we optimize control only for the next output. This control is dangerous, because the controller does not take into account future evolution of the system and to act best in one step it can generate too beg output. This can excite the system so much that it is not possible even to stabilize it in the future and the control fails.

Sequential minimization

$$\min_{u_{1:N}} E\left[\varphi_{N+1}^* + \sum_{t=1}^{N} J_t|d\left(0\right)\right] =$$

$$= \min_{u_{1:(N-1)}} E\left[\min_{u_N} \underbrace{E\left[\varphi_{N+1}^* + J_N|u_N, d\left(N-1\right)\right]}_{\varphi_N^*} + \sum_{t=1}^{N-1} J_t \middle| d\left(0\right)\right] =$$

$$= \min_{u_{1:(N-1)}} E\left[\min_{u_N} \varphi_N + \sum_{t=1}^{N-1} J_t|d\left(0\right)\right] = \min_{u_{1:N}} E\left[\varphi_N^* + \sum_{t=1}^{N-1} J_t|d\left(0\right)\right]$$

which reproduces the initial form, only with $N \to N-1$ and where (due to the reproduction in general form)

Bellman equations

$$\varphi_t = E\left[\varphi_{t+1}^* + J_t|u_t, d\left(t-1\right)\right] \quad \text{expectation}$$

$$\varphi_t^* = \min_{u_t} \varphi_t \quad \text{minimization}$$

for $t = N, N-1, N-2, \cdots, 1$. Each minimization gives the formula for optimal control - it is $u_t = \arg\min \varphi_t \left(d\left(t-1\right)\right)$. However, ti cannot be used immediately, because the data $d\left(t-1\right)$ is not known, yet. Only at time $t = 1$ we need data $d\left(0\right)$ and the control can start to be generated.

Remark

The derivation of the control law in the operator of expectation is brief but not explicit. We will show its integral form:

$$\min_{u_{1:N}} E\left[\varphi^*_{N+1} + \sum_{t=1}^{N} J_t | d\left(0\right)\right] =$$

$$= \min_{u_{1:N}} \int \cdots \int \left(\varphi^*_{N+1} + \sum_{t=1}^{N} J_t\right) f\left(y\left(N\right), u\left(N\right) | d\left(0\right)\right) dy\left(N\right) du\left(N\right) =$$

$$= \min_{u_{1:N}} \int \cdots \int \int \int \left(\left[\varphi^*_{N+1} + J_N\right] + \sum_{t=1}^{N-1} J_t\right) f\left(y_N | u_N, d\left(N-1\right)\right) f\left(u_N | d\left(N-1\right)\right) \times$$

$$\times f\left(y\left(N-1\right), u\left(N-1\right) | d\left(0\right)\right) dy\left(N\right) du\left(N\right) =$$

$$= \min_{u_{1:(N-1)}} \left\{ \int \cdots \int \min_{u_N} \int \underbrace{\int \left(\varphi^*_{N+1} + J_t\right) f\left(y_N | u_N, d\left(N-1\right)\right) dy_N}_{\varphi_N\left(u_N, d\left(N-1\right)\right)} f\left(u_N | d\left(N-1\right)\right) du_N + \right.$$

$$\left. \sum_{t=1}^{N-1} J_t f\left(y\left(N-1\right), u\left(N-1\right) | d\left(0\right)\right) dy\left(N-1\right) du\left(N-1\right) \right\}$$

Minimum over u_N

$$\min_{u_N} \int \underbrace{\int \left(\varphi^*_{N+1} + J_t\right) f\left(y_N | u_N, d\left(N-1\right)\right) dy_N}_{\varphi_N\left(u_N, d\left(N-1\right)\right)} f\left(u_N | d\left(N-1\right)\right) du_N =$$

$$= \min_{u_N} \int \varphi_N\left(u_N, d\left(N-1\right)\right) f\left(u_N | d\left(N-1\right)\right) du_N$$

→ $u^*_N = \arg\min_{u_N} \varphi_N$ - formula for optimal control law.

8.2 Derivation for regression model

Regression model can be converted to state-space form (see lecture 2 - Regression model in state-space form).

$$x_t = M x_{t-1} + N u_t + w_t$$

where $x_t = \left[y_t, u_t, y_{t-1}, u_{t-1}, \cdots y_{t-n+1}, u_{t-n+1}\right]'$.

The penalty can be written as

$$y_t^2 + \omega u_t^2 = x_t' \Omega x_t \tag{8.1}$$

where Ω is a diagonal matrix

$$\Omega = \begin{bmatrix} 1 & & & & \\ & \omega & & & \\ & & 0 & & \\ & & & \cdots & \\ & & & & 0 \end{bmatrix}$$

Now the model and criterion is used in general Bellman equations, where we guess the form of $\varphi^*_{t+1} = x_t^{'} R_{t+1} x_t$

$$E\left[x_t^{'} R_{t+1} x_t + x_t^{'} \Omega x_t | u_t, d\left(t-1\right)\right] = E\left[x_t^{'} U x_t\right] =$$

$$= \left(M x_{t-1} + N u_t\right)' U \left(M x_{t-1} + N u_t\right) + \rho =$$

$$= x_{t-1}^{'} \underbrace{M'UM}_{C} x_{t-1} + 2 u_t^{'} \underbrace{N'UM}_{B} x_{t-1} + u_t^{'} \underbrace{N'UN}_{A} u_t + \rho =$$

$$= u_t^{'} A u_t + 2 u_t^{'} A \underbrace{A^{-1} B}_{S_t} x_{t-1} + x_{t-1}^{'} S_t^{'} A S_t x_{t-1} +$$

$$+ \underbrace{x_{t-1}^{'} C x_{t-1} - x_{t-1}^{'} S_t^{'} A S_t x_{t-1}}_{x_{t-1} R_t x_{t-1}} + \rho =$$

$$= \left(u_t + S_t x_{t-1}\right)' A \left(u_t + S_t x_{t-1}\right) + x_{t-1}^{'} R_t x_{t-1} + \rho$$

Optimal $u_t = S_t x_{t-1}$.

Recursion

```
$R_{N+1} = 0$
for $t = N,\ N-1,\ \cdots,\ 1$
    $U = R_{t+1} + \Omega$
    $A = N'UN$
    $B = N'UM$
    $C = M'UM$
    $S_t = A^{-1}B$
    $R_t = C - S_t^{'} Q S_t$
    $u_t = S_t x_{t-1}$.
end
```

Remark

The penalty function (8.1) can be very easily extended to the following form

$$\left(y_t - s_t\right)^2 + \omega u_t^2 + \lambda \left(u_t - u_{t-1}\right)^2$$

where the first term leads to the following the output y_t the prescribed set-point s_t and the last term introduces penalization of increments of the control variable. Penalizing the control increments calms control behavior and at the same time it does not result to steady-state deviation of the output and the set-point as it is when penalizing the whole control variable.

The solution how to introduce the above requirements for the control lies in construction of the penalization matrix as follows

$$\Omega = \begin{bmatrix} 1 & & & & & & -1 \\ & \omega+\lambda & & -\lambda & & & \\ & & 0 & & & & \\ & -\lambda & & \lambda & & & \\ & & & & \dots & & \\ & & & & & 0 & \\ -1 & & & & & & 1 \end{bmatrix}$$

which is evident if we take into account that the criterion is

$$x_t^{'}\Omega x_t$$

and $x_t = [y_t,\, u_t,\, y_{t-1},\, u_{t-1},\, \cdots, 1]$.

9 Control with categorical model

We will show the synthesis for the model of controlled coin with memory

$$\begin{array}{ll} \text{model} & f\left(y_t|u_t, y_{t-1}\right) \\ \text{penalty} & J_{y_t|u_t,y_{t-1}} \end{array}$$

and for three steps control, i.e. for $t = 1, 2, 3$. The corresponding model and penalization are

model (Θ)

$u_3,\ y_2$	$y_3 = 1$	$y_3 = 2$
1, 1	0.7	0.3
1, 2	0.2	0.8
2, 1	0.9	0.1
2, 2	0.4	0.6

penalty (J)

$u_3,\ y_2$	$y_3 = 1$	$y_3 = 2$
1, 1	0	1
1, 2	1	0
2, 1	1	2
2, 2	2	1

9.1 Optimization

Step for $t = 3$: $\varphi_4^* = 0$

Expectation

$$\varphi_3 = E\left[J|u_3, d\left(2\right)\right] = \sum_{y_3=1}^{2} J_{y_3|u_3,y_2}\Theta_{y_3|u_2,y_2} =$$

$$= \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} .* \begin{bmatrix} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} .* \begin{bmatrix} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 1.1 \\ 1.4 \end{bmatrix} \begin{array}{l} \cdots \quad u_3 = 1,\ y_2 = 1 \\ \cdots \quad u_3 = 1,\ y_2 = 2 \\ \cdots \quad u_3 = 2,\ y_2 = 1 \\ \cdots \quad u_3 = 2,\ y_2 = 2 \end{array}$$

Minimization

$$\text{for}: \quad y_2 = 1 \rightarrow \min\left\{0.3,\ 1.1\right\} = 0.3 \ \text{ for } u_3 = 1$$

$$\text{for}: \quad y_2 = 2 \rightarrow \min\left\{0.2,\ 1.4\right\} = 0.2 \ \text{ for } u_3 = 1$$

→

$$u_3 = \begin{cases} 1 & \text{for } y_2 = 1 \\ 1 & \text{for } y_2 = 2 \end{cases}$$

and reminder after minimization

$$\begin{array}{cc} y_2 = 1 & y_2 = 2 \\ \hline 0.3 & 0.2 \end{array} \ \forall u_2, y_1 \ \rightarrow \ \begin{array}{c|cc} u_2,\ y_1 & y_2 = 1 & y_2 = 2 \\ \hline 1,\ 1 & 0.3 & 0.2 \\ 1,\ 2 & 0.3 & 0.2 \\ 2,\ 1 & 0.3 & 0.2 \\ 2,\ 2 & 0.3 & 0.2 \end{array} \ = \varphi_3^*$$

Step for $t = 2$:

Expectation

$$\varphi_2 = E\left[J + \varphi_3^* | u_2, d\left(1\right)\right] = \sum_{y_2=1}^{2} \left(J_{y_2|u_2,y_1} + \varphi^*_{3;y_2|u_2,y_1}\right) \Theta_{y_2|u_2,y_1} =$$

$$= \left(\begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}\right) .\varphi_2^* = 0 * \begin{bmatrix} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{bmatrix} + \left(\begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}\right) .* \begin{bmatrix} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.8 \\ 0.7 \\ 1.6 \\ 1.9 \end{bmatrix} \begin{matrix} \cdots & u_2 = 1,\, y_1 = 1 \\ \cdots & u_2 = 1,\, y_1 = 2 \\ \cdots & u_2 = 2,\, y_1 = 1 \\ \cdots & u_2 = 2,\, y_1 = 2 \end{matrix}$$

Minimization

$$\text{for}: \quad y_1 = 1 \rightarrow \min\left\{0.8,\, 1.6\right\} = 0.8 \ \text{ for } u_2 = 1$$

$$\text{for}: \quad y_1 = 2 \rightarrow \min\left\{0.7,\, 1.9\right\} = 0.7 \ \text{ for } u_2 = 1$$

→

$$u_2 = \begin{cases} 1 & \text{for } y_1 = 1 \\ 1 & \text{for } y_1 = 2 \end{cases}$$

and reminder after minimization

$$\begin{array}{cc} y_1 = 1 & y_1 = 2 \\ \hline 0.8 & 0.7 \end{array} \ \forall u_1, y_0 \ \rightarrow \ \begin{array}{c|cc} u_1,\, y_0 & y_1 = 1 & y_1 = 2 \\ \hline 1,\, 1 & 0.8 & 0.7 \\ 1,\, 2 & 0.8 & 0.7 \\ 2,\, 1 & 0.8 & 0.7 \\ 2,\, 2 & 0.8 & 0.7 \end{array} = \varphi_2^*$$

Step for $t = 1$:

Expectation

$$\varphi_1 = E\left[J + \varphi_2^* | u_1, d\left(0\right)\right] = \sum_{y_1=1}^{2} \left(J_{y_1|u_1,y_0} + \varphi^*_{2;y_1|u_1,y_0}\right) \Theta_{y_1|u_1,y_0} =$$

$$= \left(\begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.8 \\ 0.8 \\ 0.8 \\ 0.8 \end{bmatrix}\right) .* \begin{bmatrix} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{bmatrix} + \left(\begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.7 \\ 0.7 \\ 0.7 \\ 0.7 \end{bmatrix}\right) .* \begin{bmatrix} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{bmatrix} =$$

$$= \begin{bmatrix} 1.8 \\ 1.7 \\ 2.6 \\ 2.9 \end{bmatrix} \begin{matrix} \cdots & u_1 = 1,\, y_0 = 1 \\ \cdots & u_1 = 1,\, y_0 = 2 \\ \cdots & u_1 = 2,\, y_0 = 1 \\ \cdots & u_1 = 2,\, y_0 = 2 \end{matrix}$$

Minimization

$$\text{for}: \quad y_0 = 1 \rightarrow \min\{1.8,\, 2.6\} = 1.8 \;\; \text{for}\, u_1 = 1$$

$$\text{for}: \quad y_1 = 2 \rightarrow \min\{1.7,\, 2.9\} = 1.7 \;\; \text{for}\, u_1 = 1$$

→

$$u_1 = \begin{cases} 1 & \text{for } y_0 = 1 \\ 1 & \text{for } y_0 = 2 \end{cases}$$

and reminder after minimization

$y_0 = 1$	$y_0 = 2$
1.8	1.7

9.2 Application

For $t = 0$ let us have $y_0 = 2$.

For $y_0 = 2$ we have $u_1 = 1$; → [1, 2] $\Theta_{1,2}$ =[0.2, 0.8] let us obtain $y_1 = 2$

For $y_1 = 2$ we have $u_2 = 1$; → [1, 2] $\Theta_{1,2}$ =[0.2, 0.8] let us obtain $y_2 = 1$

For $y_2 = 1$ we have $u_3 = 1$; → [1, 1] $\Theta_{1,1}$ =[0.7, 0.3] let us obtain $y_3 = 2$

The final value of criterion is $J_{2|12} + J_{1|12} + J_{2|11} = 0 + 1 + 1 = 2$.

Part II

Clustering and Classification

10 Mixtures

One of the prominent approaches in data mining is based on data modeling. The model describes density of data points in the data space and gives a possibility to detect the areas with high density to which a newly measured data record (point in the space) belongs.

First we will inspect models connected with this area and derive simple but general clustering and classification tool corresponding to estimation of mixture models.

Then we will show the same procedure as before but with a simplifying assumption of independence of variables in the regression vector. This approach is known as Naive Bayes.

In the end, we will extend the previous attitude for models with unknown parameters. Then the parameters must be estimated from data.

10.1 The basics of clustering and classification

Our approach to clustering and classification here is based on modeling. Generally, we consider a data space X of finite vectors $x = [x_1, x_2, \cdots, x_n]$. These vectors represent points in the multivariate data space and we suppose, these points are somehow grouped with respect to their density (or spatial probability of occurrence).

These groups of data vectors (represented as areas of dense points in data space) are called **clusters**. Our task is

1. to detect these groups in the data space (clustering),
2. to decide, which class a newly measured vector belongs to (classification).

The groups ale labeled - each of them has its own flag (value of pointer). In our case, the flags will be integers $c = 1, 2, \cdots, n_c$. In classification, we measure a vector x and want to classify it. As the true class to which the vector belongs is unknown, the pointer c will be described by a discrete random variable with its probability function $f(c|x)$ where x is the vector to be classified.

Individual groups have their models $f(x|c) = f_c(x)$ which is a probabilistic description of vectors x belonging to the class c.

Models

As we have seen, in connection with the tasks of clustering and classification, we have two models (for now, with known parameters).

Model of data

$$f(x|c)$$

Model of classification

$$f(c|x)$$

These two models are connected via Bayes rule

$$f(c|x) = f(x|c)\frac{f(c)}{f(x)} \propto f(x|c)\, f(c)$$

where $\propto$ denotes proportionality.

We will demonstrate these models in the following example.

Example

Let the joint model $f(c,x) = f(x|c)\,f(c)$ be described trough the conditional pdf

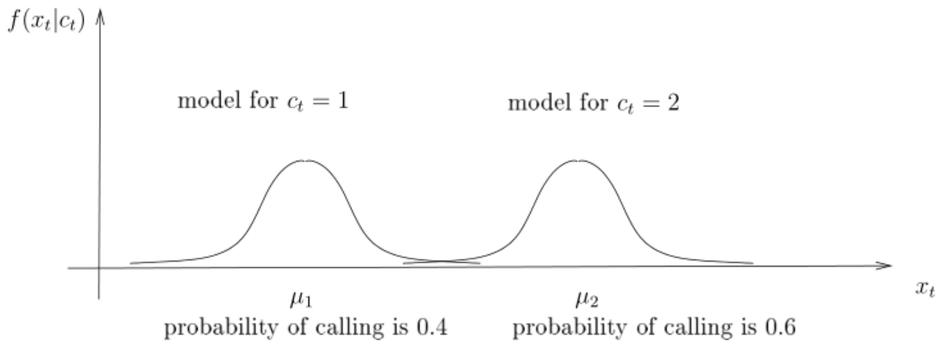
$$f(x|c=1) = f_1(x) = N_x(\mu_1, 1) \tag{10.1}$$

$$f(x|c=2) = f_2(x) = N_x(\mu_2, 1) \tag{10.2}$$

and the marginal

c	1	2
$f(c)$	0.4	0.6

Notice that each model is of a different type. The data are continuous, so their model type is also continuous (regression model), while the pointer is discrete and its description is a discrete probability function. As the data model depends on the pointer, we have to define two regression models - for each pointer value one model. They differ in parameters. The first component has expectation μ_1 while the second μ_2. The model can be demonstrated in the picture



In the left part of the figure, the model of the first component is shown. It is active (generates data) in the 40%. In the right one there is the model of the second component, active at 60% of cases.

Now, the joint model $f(x,c)$, is given by a product of the conditional data model and the marginal pointer model; it is

$$f(x,c) = \begin{cases} 0.4N_x(\mu_1, 1) & \text{for } c = 1, \\ 0.6N_x(\mu_2, 1) & \text{for } c = 2, \end{cases} \text{for } x \in R$$

This is how a mixed model (i.e. model with both continuous and discrete variables) can be expressed.

Having the joint distribution of the model, we can express arbitrary conditional or marginal model. Conditional one of the data and marginal one for the pointer we already have - we have defined them above. Now, we are going to determine the remaining two models.

Marginal data model

Is obtained by summing the joint model over all values of the pointer, i.e. for $c = 1, 2$. We get

$$f\left(x\right) = 0.4 N_x\left(\mu_1, 1\right) + 0.6 N_x\left(\mu_2, 1\right)$$

which is a weighted sum of two Gaussian distributions.

Classification model

Can be computed as

$$f\left(c|x\right) = \frac{f\left(c, x\right)}{f\left(x\right)} \text{ or } f\left(x|c\right) \frac{f\left(c\right)}{f\left(x\right)} \propto \underbrace{f\left(x|c\right)}_{\text{component model}} \underbrace{f\left(c\right)}_{\text{component prior}} \tag{10.3}$$

$$f\left(c|x\right) = \begin{cases} \frac{0.4N_x(\mu_1,1)}{0.4N(\mu_1,1)+0.6N(\mu_1,2)} & \text{for } c = 1 \\ \frac{0.6N_x(\mu_2,1)}{0.4N(\mu_1,1)+0.6N(\mu_1,2)} & \text{for } c = 2 \end{cases} \propto$$

$$\propto \begin{cases} 0.4 N_x\left(\mu_1, 1\right) & \text{for } c = 1 \\ 0.6 N_x\left(\mu_2, 1\right) & \text{for } c = 2 \end{cases}$$

which is obvious (joint pdf is proportional to joint one), however, this is very important result claiming that:

Result: *The probability that x is to be classified into the class c is proportional to the value of the model of this component with the vector x inserted.*

The classification can run like this:

1. Measure new data vector x
2. Compute values of all component models with inserted vector x
3. Multiply them by probabilities of individual classes
4. Assign the vector to the component corresponding to the greatest computed value.

Remark

This holds for known models of components and pointer. If the parameters of these models are unknown, they have to be estimated and their point estimates can be used instead of the true parameters. It is an approximation but very good one. We will tackle this problem in more details later.

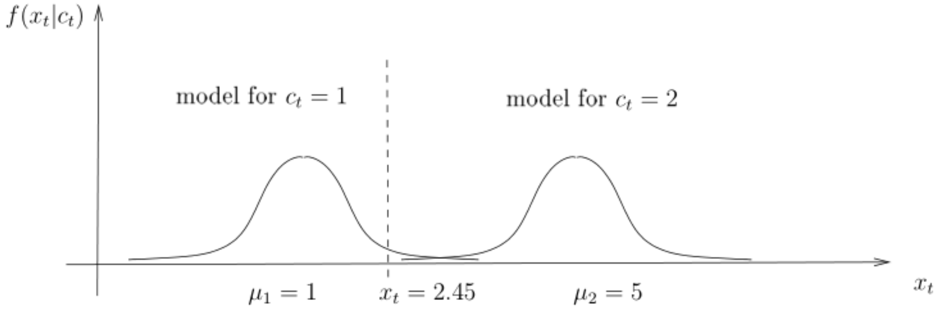
In our example we have.

Let $\mu_1 = 1,\ \mu_2 = 5$ and the measured vector $x = 2.45$. The the values of the component models are

$$f_1 = 0.4 \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(2.45 - 1\right)^2\right\} = 0.056$$

$$f_2 = 0.6\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{1}{2}\left(2.45-5\right)^2\right\} = 0.009.$$

The first value is greater, the point $x = 2.45$ belongs to the first component. The situation in this simple example is clearly visible from the following picture



The point x lies closer to the first model, so the value of the model in x is greater.

Remark

The influence of the pointer model is not so important and is neglected in the picture. The main effect is caused by the component models and their values will be called proximity *as they express the closeness of the measured point to the centers of individual components.*

10.2 Naive Bayes classification

This method is nothing but the previous case plus assumption of conditional independence of data variables, i.e. entries of the data vector x. With this assumption we have

$$f\left(x|c\right) = \prod_{i=1}^{n} f\left(x_{i;t}|c\right).$$

Remarks

1. *This formula can be explained by the assumed fact that the data in one cluster differ only by noise and thus are independent.*
2. *The independence brings considerable savings - instead of multidimensional model we can use only several one-dimensional ones. For normal components, instead of large covariance matrix we need only several (namely n) scalar variances.*

The pointer model (10.3) has now the form

$$f\left(c|x\right) \propto f\left(x|c\right)f\left(c\right) = f\left(c\right)\prod_{i=1}^{n} f\left(x_{i;t}|c\right)$$

where $f\left(x_{i;t}|c\right)$ are scalar models of individual variables within the class c.

10.3 Classification with learning

Classification 1 - known components

In this case(not very realistic from application viewpoint) we assume that the component models $f(x|c=i), i=1,2,\cdots\nu$, and switching probabilities $f(c), i=1,2,\cdots\nu$ are known. We have measured one data record $x=\xi$ and we are to estimate the most probable component to which the record belongs (i.e. the value of c). The decision is described by the pf

$$f(c|\xi) \propto f(\xi|c) f(c)$$

which is (e.g. for $\nu=3$)

$$f(c=1|\xi) = kf(\xi|c=1) f(c=1)$$
$$f(c=2|\xi) = kf(\xi|c=2) f(c=2)$$
$$f(c=3|\xi) = kf(\xi|c=3) f(c=3)$$

The result of the decision is given by selecting the value of c with the greatest probability.

Remark

Similarly we can continue with the next measured records.

Classification 2 - known pointer for learning

In this case we assume, that the components as well as the pointer probabilities are unknown, However, we have a learning data sample $x_1, x_2, \cdots x_N$ together with the corresponding pointer values $c_1, c_2, \cdots c_N$ for learning. After learning we get $x=\xi$ and we are to classify it.

Solution

Divide the learning sample x into clusters C_c, $c=1,2,\cdots,n_c$ according to the learning sample of pointer values. Then, for all $x_t \in C_c$ update the statistics S_{c_t} and finally construct components.

Now, follow the previous case.

Remark

If the components are normal, the estimation consists in computing expectations $\bar{x}_c$ and covariance matrix Σ_c within each set C_c.

For categorical components the component parameters are normalized histograms of x within all C_c.

Classification 3 - EM-like algorithm

Here we assume that both component models and pointer values are unknown. It means that we do not know to which component the data records x_t, $t=1,2,\cdots,T$ belong, i.e. we do not know which component is to be updated. Updating of all components with the same data has no sense. All components would converge to the same model from their initial setting. So, we have to initialize the components, for this setting we determine the pointer values (i.e. to each record from dataset x we determine to which component it belongs) and then estimate components for the created pointer. And this is repeated: for given components determine pointer values and for this pointer update parameters of the components. The end of iterations can be indicated by the fact, that the pointer stays the same.

Algorithm

1. Take a dataset $X = [x_1, x_2, \cdots, x_N]$ for estimation
2. Set initial components and switching probabilities.
3. Determine pointer values $f(c|X) \propto \alpha_c f(x_t|\theta_c)$
   For $t = 1 : N$

$$\begin{gathered} f(c=1|X) \propto \alpha_1 f(x_t|\theta_1) \\ f(c=2|X) \propto \alpha_2 f(x_t|\theta_2) \\ \cdots \\ f(c=\nu|X) \propto \alpha_\nu f(x_t|\theta_\nu) \end{gathered}$$

4. Recompute component parameters and switching probabilities
   For $i = 1 : \nu$ do
   (a) select subset of dataset whose records correspond to pointer value i
   (b) use this subset for estimation of parameters of the i-th component $f(x|\theta_i)$ for normal components - average and variance
   (c) switching probabilities are relative frequencies of the pointer values
5. If the pointer changes, go to 3

Classification 4 - mixture estimation

The final and and in applications most required salutation is when the mixture parameters are unknown and we continuously get data records and we have to on-line estimate the mixture and at the same time to classify the data. Then, we have to proceed as follows:

For a data record x_t and the current estimate (or prior estimate) of the model parameters determine the weights of the measured x_t with respect to all components using the probability function $f(c_t|x_t)$ (its constitution is shown later).

The data record x_t is added to all statistics with its weight

$$S_{j;t} = S_{j;t-1} + w_j x_t,$$

$$\kappa_{j;t} = \kappa_{j;t-1} + w_j.$$

With new statistics the point estimates of the parameters are constructed.

Algorithm

Initial setting: Set initial parameters of component $(\theta, r,)$ switching probabilities (α) and corresponding statistics.

for $t = 1 : nd$

```
    1. measure data record $x_t$
    2. determine weights $w$
       for $j = 1 : nc$
          (a) $q_j = f\left(x_t|\theta_j\right)$ - proximity
          (b) $w_j = \aleph\left(q_j\alpha_j\right)$ - where $\aleph$ means normalization to sum equal to 1
       end
    3. recompute statistics and parameters
       for $j = 1 : nc$
          (a) $S_{j;t} = S_{j;t} + w_j x_t$
          (b) $\kappa_{j;t} = \kappa_{j;t} + w_j$
          (c) $\gamma_{j;t} = \gamma_{j;t-1} = w_j$
          (d) $\theta_j = \frac{S_{j;t}}{\kappa_{j;t}}$
          (e) $\alpha_j = \aleph\left(\gamma\right)$
       end
end
```

Remarks

1. The derivation can be found in the textbook.
2. For component parameters, the point estimates have been used.
3. There are two main points used
   (a) pointer estimation for new data record - the basis is $f\left(c|x\right)$
   (b) update of statistics with the weight
   - standard update: $S = S + x$
   - for two identical x and x it is: $S = S + 2x$ (weight)
   - similarly for x valid with probability w it is: $S = S * wx$ (again weight)
   and similarly for other statistics.

Derivation of mixture estimation

As in the beginning of this section, we looked for the classification pdf $f\left(c|x\right)$. To be able to construct it, now, we must introduce the model parameters θ and α for the models $f\left(x|c=j,\theta_j\right),\ j = 1, 2, \cdots, n_c$ and $f\left(c|\alpha\right)$

We start with pdf of all unknown objects $f\left(c,\theta,\alpha|x\right)$ and perform its factorization

$$f\left(c,\theta,\alpha|x\right) \propto \underbrace{f\left(x,c,\theta,\alpha\right)}_{\text{joint pdf}} = f\left(x|c,\theta,\alpha\right) f\left(c|\theta,\alpha\right) f\left(\theta,\alpha\right) =$$

$$= \underbrace{f\left(x|c,\theta_c\right)}_{\text{component model}} \underbrace{f\left(c|\alpha_c\right)}_{\text{pointer model}} \underbrace{f\left(\theta,\alpha\right)}_{\text{prior}}$$

where the first two pdfs are parameterized models of data and pointer, the last one is a prior description of parameters which is updated to posterior with the information carried by the data vector x. From this relation for the joint pdf we can obtain all needed

1. The classification (determination of the weights w)
$$
\begin{aligned}
f\left(c|x\right) &= \int_{\theta^*}\int_{\alpha^*} f\left(x,c,\theta,\alpha\right)d\alpha d\theta = \\
&= \int_{\theta^*}\int_{\alpha^*} f\left(x|c,\theta_c\right)f\left(c|\alpha_c\right)f\left(\theta,\alpha\right)d\alpha d\theta \doteq \qquad (10.4)\\
&= \begin{cases} f\left(x|c=1,\hat{\theta}_1\right)\hat{\alpha}_1 & \text{for } c=1 \\ f\left(x|c=2,\hat{\theta}_1\right)\hat{\alpha}_2 & \text{for } c=2 \\ \cdots & \cdots \\ f\left(x|c=n_c,\hat{\theta}_{n_c}\right)\hat{\alpha}_{n_c} & \text{for } c=n_c \end{cases}
\end{aligned}
$$
where $\hat{\theta}_1,\ \hat{\theta}_2,\cdots\hat{\theta}_{n_c}$ and $\hat{\alpha}_1,\ \hat{\alpha}_2,\ \cdots,\ \hat{\alpha}_{n_c}$ are current (or prior) point estimates of the parameters.

2. The estimation (
$$
\begin{aligned}
f\left(\theta,\alpha|x\right) &= \sum_c f\left(x,c,\theta,\alpha\right) = \\
&= \sum_c \left[f\left(x|c,\theta_c\right)f\left(c|\alpha_c\right)\right]f\left(\theta,\alpha\right) \qquad (10.5)
\end{aligned}
$$

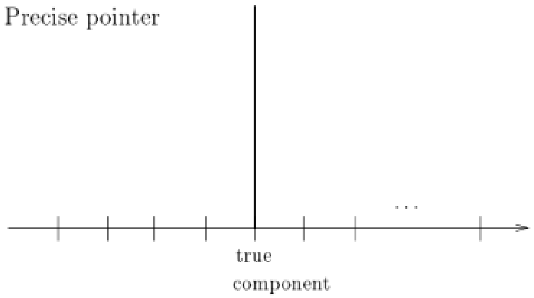
However, the summation form of the model in recursive estimation a serious trouble. As the Bayes rule is a product of pdfs and the model is a sum, its repetitive calling produces the posterior pdf in a form which gets more and more complex and its evaluation and storing in memory is unfeasible. So, we must proceed as follows:

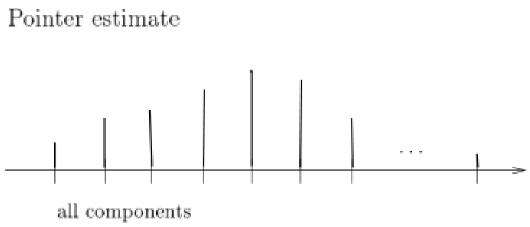
Suppose, we know the true component to which the measured data record belongs. Then we can define a pointer
$$
\delta\left(c_t,\hat{c}_t\right) = \begin{cases} 1 & \text{for } c_t=\hat{c}_t \\ 0 & \text{elsewhere} \end{cases}
$$
where c_t is random variable and $\hat{c}_t$ its realization (the label of the true component). Thus, at each time instant t the pointer denotes the component that is really true (active - the data record x_t belongs to it). However, in reality, we do not know the active component. So, we must estimate the pointer as an expectation
$$
E\left[\delta\left(c_t,\hat{c}_t\right)|x\left(t\right)\right] = \sum_{c\epsilon c_t}\delta\left(c,\hat{c}_t\right)f\left(c|x\left(t\right)\right) = P\left(c=\hat{c}_t|x\left(t\right)\right) \text{ for } c=1,2,\cdots,n_c
$$
which is a vector of probabilities that the c-th component is active. We will call that vector actual components weights and denote it by $w_t=\left[w_{1;t},w_{2;t},\cdots,w_{n_c;t}\right]$ where
$$
w_{i;t} = P\left(c_t=i|x\left(t\right)\right),\ i=1,2,\cdots,n_c
$$

Remark

Notice that w_t depends on the actually measured data record x_t. It is the difference between it and the pointer model $f\left(c_t|\alpha\right)$. The pointer model expresses only historical knowledge about the activities of the component while w_t takes into account also x_t which is most important for the actual classification.

The effect of the approximation is following: Formerly, we needed to know the true active component. Now, we only need to know the probabilities that each individual component is active. The knowledge of the true active component is not required. It is like in the following picture





The pointer, now, is nothing but the classification pdf $f\left(c_t|x\left(t\right)\right)$. It has been determined formerly in (10.4).

Example

We will continue with the same example like in the preceding sections. We will:

1. Simulate a mixture with two static Gaussian components

$$\begin{aligned} &f_1\left(x_t|\mu_1\right), \quad \mu_1 = 1 \\ &f_1\left(x_t|\mu_2\right), \quad \mu_2 = 5 \end{aligned}$$

   with known variances equal to 1 and pointer model

$$f\left(c_t|\alpha\right), \;\; \alpha = [0.4, \, 0.6] \, .$$

2. Estimate the mixture with initial parameters

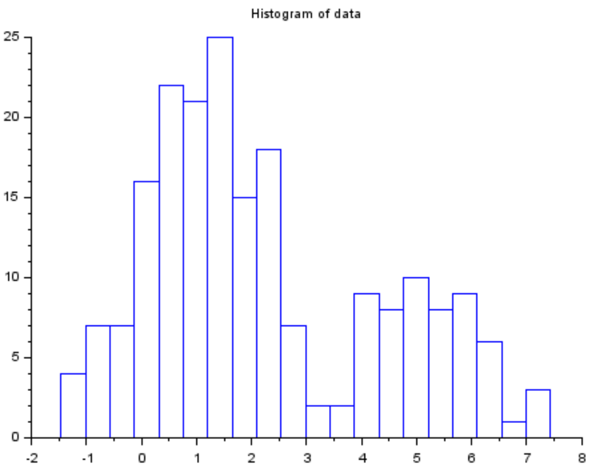
$$\hat{\mu}_{1;0} = 2, \;\; \hat{\mu}_{2;0} = 3, \;\; \hat{\alpha} = [0.5, \, 0.5] \, .$$

The program is here

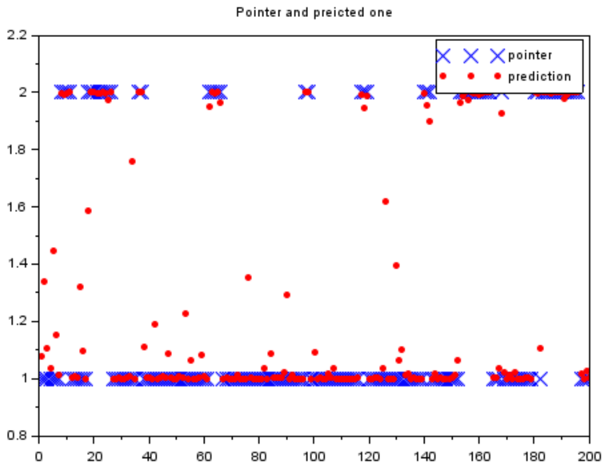
The program is described inside. Only some notes are necessary:

1. Simulation: first the pointer value is generated and according it a corresponding component is used for data generation.
2. The second part is estimation.
   (a) First, the initial parameters m and al are specified. K is the counter. Its initial value expresses the strength of prior information (the fictive number of data from which the information has been extracted).
   (b) Then, in the time loop, weights are computed. The computation is performed in logarithms, then it is roughly normalized by subtracting maximum, then exponent is taken and multiplication with al is performed and finally normalized to sum equal to one.
   (c) In the end of the loop, statistics are updated by weighted data and point estimates computed.

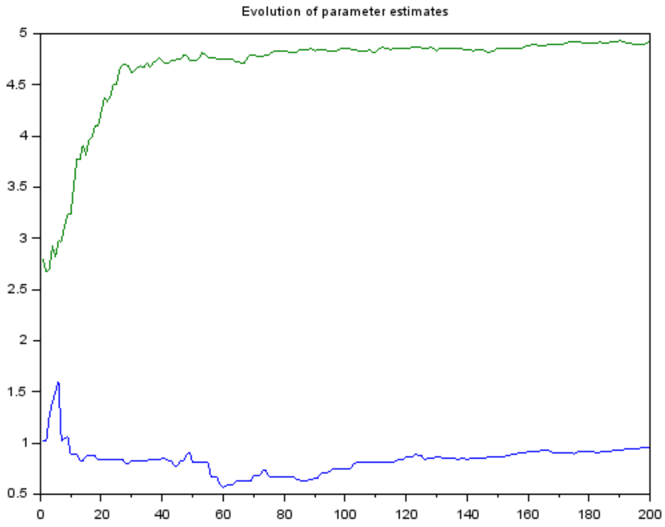
The results of estimation (classification) are in the following pictures



Here the histogram of data sample is plotted. It can be seen, that the components are slightly overlapping. The classification is not trivial.



Here, the simulated (blue) pointer values and the predicted (magenta) ones are plotted. The prediction is finally classified to the class which is closer to it. It can be seen that at the beginning, when the learning just started there are some errors. Gradually it improves and in the end all classifications are correct.



And here is supplementary information - evolution of expectation estimated during estimation. The initial estimates are gradually improved till they reach practically correct values (1 and 5).

Remark

The approach presented for the last time is practically equivalent to mixture estimation.

11 Regression

Here, we will demonstrate the logistic and Poisson regression. They are both very similar:

1. They use nonlinear models with unknown parameters.
2. Their estimation is performed off-line using numerical optimization. It has two phases: learning and testing.
3. They need to cope with non-negativity of estimated parameters.

11.1 Logistic regression

Model for variable c_t with Bernoulli distribution

$$f\left(c_t|p\right) = p^{c_t}\left(1-p\right)^{1-c_t}$$

with $c_t = 0,\,1$ is dichotomous model output $p \in (0,1)$ is the probabilistic parameter: $p = P\left(c_t = 1\right)$.

The expectation of c_t is

$$E\left[c_t|p\right] = p$$

Now, we would like to extend this model so that its expectation will be modeled by regression in the form

$$p \to x'b = b_0 + b_1 x_1 + \cdots + b_m x_m$$

However, there are problems. $p \in (0,1)$, i.e. it is nonnegative and bounded from above.

1. The solution with respect to bounding is: instead of p to model $\frac{p}{1-p}$ which is from the interval $(0,\infty)$
2. Nonnegativity of $\frac{p}{1-p}$ can be solved by taking logarithm $\ln\frac{p}{1-p}$. This variable is called *logit*

$$logit\left(p\right) = \ln\left(\frac{p}{1-p}\right)$$

This logit will be modeled by regression

$$\ln\left(\frac{p}{1-p}\right) = x_t b$$

The final model $f\left(c_t|b\right)$ can be derived from the above expression and it has the form

$$f\left(c_t|b\right) = p = \begin{cases} \frac{\exp\{x_t b\}}{1+\exp\{x_t b\}} & \text{for } c_t = 1 \\ \frac{1}{1+\exp\{x_t b\}} & \text{for } c_t = 0 \end{cases}$$

and using the fact that $c_t \in \{0,1\}$ we can write the model as

$$f\left(c_t|b\right) = \frac{\exp\left\{c_t x_t b\right\}}{1+\exp\left\{x_t b\right\}}.$$

Note, that both the mentioned demands are fulfilled - $p \in (0,1)$, and nonnegative, indeed.

For estimation of the parameter p we will construct the likelihood function

$$L_N(p) = \prod_{t=1}^{N} \frac{\exp\{c_t x_t b\}}{1 + \exp\{x_t b\}}$$

where we used a trick for writing the model in a unified form. For $c_t = 1$ the nominator in the model will be $\exp\{x_t b\}$ and for $c_t = 0$ it will be 1.

The log-likelihood is

$$\ln L_N(p) = \sum_{t=1}^{N} \left[c_t x_t b - \ln\left(1 + \exp\{x_t b\}\right)\right]$$

As the first and second derivatives of this expression can be computed analytically, the Newton method for numerical maximization is very suitable. It is quick and has fast convergence.

Program

```
// DM_LogisReg.sce
// Example: Logistic regression with two independen variables
// ------------------------------------------
clc, clear, close, mode(0), warning(’off’)
getd _func
function LL=logLL(b,par)
  // log-likelihood of logistic regression
  x=par.x;                         // data x
  y=par.y;                         // data y
  Li=y.*(x*b)-log(1+exp(x*b));     // vector of log-models
  LL=-sum(Li);                     // log-likelihood
endfunction
function [f,g,ind]=fun(b,ind,par)
  // auxiliary function
  f=logLL(b,par);             // log-likelihood
  g=numderivative(logLL,b);   // gradient
endfunction

// SIMULATION ==================================================================
nd=200;                       // number of data
bS=[4 8 -1]’;                 // simulated parameter
sd=1;                         // regression noise  z=x*b+sd*rand
x=[ones(nd,1) rand(nd,1,’n’) 5-rand(nd,1,’n’)];
z=x*bS+sd*rand(nd,1,’n’);
p=exp(z)./(1+exp(z));
y=round(p);

// LOGISTIC REGRESSION --------------------------------------
b0=[0 0 0]’;                  // initial estimates of parameters (including ones)
par.x=x;                      // data x
par.y=y;                      // data y
```

```
// estimation
fce=list(fun,par);
[LLopt, b,  gopt, work, iters, evals, err]..
= optim (fce,b0,iprint=2,'ar',1e8,1e8);    // optimization
b,err

z=par.x*b;                    // regression
p=exp(z)./(1+exp(z));         // p=P(y=1|x)
yp=round(p);                  // rounding  <.5 ->0, >.5 -> 1

wrong=sum(y~=yp)              // number of wrong classification

// RESULTS
set(scf(),'position',[800 10 500 300]);
plot(1:nd,y,'bx',1:nd,yp,'r.')

Ep=variance(y-yp)/variance(y) // relative prediction error

scf();
plot(jiggle(y),jiggle(yp),'.','markersize',3)
title 'y against yp - jiggled'
```

11.2 Poisson regression

Model with Poisson distribution

$$f\left(c_t|\lambda\right)=\exp\left\{-\lambda\right\}\frac{\lambda^{c_t}}{c_t!} \tag{11.1}$$

with $c_t=0,1,2,\cdots,\infty$, $\lambda>0$ it the expectation (average number of events per time unit). Again, the expectation should be expanded by regression. The condition of upper limit is nor demanded, but the non-negativity remains and is solved in the same way as for logistic regression - by expanding logarithm of λ instead of λ itself

$$\ln\left(\lambda\right)=x_tb=b_0+b_1x_1+\cdots+b_mx_m.$$

Thus, for λ we have

$$\lambda=\exp\left\{x_tb\right\}.$$

The final model $f\left(c_t|\lambda\right)$ will be (11.1) with the above substitution - for log-likelihood we express directly its logarithm

$$\ln\left\{f\left(c_t|b\right)\right\}=-\exp\left\{x_tb\right\}+c_tx_tb-\ln\left(c_t!\right)$$

Log-likelihood is

$$\ln L_N\left(b\right)=\sum_{t=1}^{N}\left[-\exp\left\{x_tb\right\}+c_tx_tb-\ln\left(c_t!\right)\right]$$

and it is maximized numerically.

Program to the Poisson regression is here

```
// DM_LogisReg.sce
// Example: Logistic regression with two independen variables
// ------------------------------------------
clc, clear, close, mode(0), warning('off')
getd _func
function LL=logLL(b,par)
  // log-likelihood of logistic regression
  x=par.x;                         // data x
  y=par.y;                         // data y
  Li=y.*(x*b)-log(1+exp(x*b));     // vector of log-models
  LL=-sum(Li);                     // log-likelihood
endfunction
function [f,g,ind]=fun(b,ind,par)
  // auxiliary function
  f=logLL(b,par);             // log-likelihood
  g=numderivative(logLL,b);   // gradient
endfunction

// SIMULATION ==================================================================
nd=200;                       // number of data
bS=[4 8 -1]';                 // simulated parameter
sd=1;                         // regression noise  z=x*b+sd*rand
x=[ones(nd,1) rand(nd,1,'n') 5-rand(nd,1,'n')];
z=x*bS+sd*rand(nd,1,'n');
p=exp(z)./(1+exp(z));
y=round(p);

// LOGISTIC REGRESSION --------------------------------------
b0=[0 0 0]';                  // initial estimates of parameters (including ones)
par.x=x;                      // data x
par.y=y;                      // data y

// estimation
fce=list(fun,par);
[LLopt, b,  gopt, work, iters, evals, err]..
= optim (fce,b0,iprint=2,'ar',1e8,1e8);    // optimization
b,err

z=par.x*b;                    // regression
p=exp(z)./(1+exp(z));         // p=P(y=1|x)
yp=round(p);                  // rounding  <.5 ->0, >.5 -> 1

wrong=sum(y~=yp)              // number of wrong classification

// RESULTS
set(scf(),'position',[800 10 500 300]);
plot(1:nd,y,'bx',1:nd,yp,'r.')

Ep=variance(y-yp)/variance(y) // relative prediction error
```

```
scf();
plot(jiggle(y),jiggle(yp),'.','markersize',3)
title 'y against yp - jiggled'
```

12 Clustering

The task of clustering consists in dividing the data space into several subspaces whose data are somehow similar. Mostly the similarity is given by the distance of the points. We demand that the points in a cluster are as close as possible and on the other hand the points between different clusters are as remote as possible. However, the sorting can be governed also by other rules as e.g. color or shape of “data points”.

For us the clustering according to the distance will be decisive. The distance is mainly Euclidean but it can also be some other, like Manhattan or Minkowski ones.

12.1 K-means algorithm

Let us have a data sample $X = [x_1, x_2, \cdots, x_N]$ where $x_t = [x_{1;t}, x_{2;t}, \cdots, x_{n;t}]$ is a data record (point) and N is total number of data records. The algorithm of clustering is as follows

0. Determine the number of clusters ans set their initial centers.
1. For each data point measure the distance to all cluster centers and assign the point to the nearest center. The points form clusters.
2. Compute the centroid (average) of points in each cluster and set it as its new center.
3. Check, if the centers changed. If yes, go to 1. If not, the algorithm ends.

Program

```
// DM_kmeans.sce
// K-means
// ------------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction

// SIMULATION
m=list();
m(1)=[0 1]';
m(2)=[5 2]';
m(3)=[3 8]';
n=[15 30 20]*10;
sd=1.5;
ny=length(m(1));
```

```
y=[];
for i=1:3
  for t=1:n(i)
    y=[y m(i)+sd*rand(ny,1,’n’)]; // data generation
  end
end

// ALGORITM
nd=size(y,2);            // number of data
nc=length(m);            // number of clusters
yc=list(); cL=list();
// inicialization of centers
mi=min(y,’c’);
ma=max(y,’c’);
for j=1:nc
  C(j).c0=(mi+ma)/2+rand(ny,1,’n’);     // initial centers of clusters
  C(j).c=C(j).c0;             // first centers are initial ones
end

for it=1:1000
  for j=1:nc
    C(j).cd=[];               // initialization of clusters
  end

  // distances of data from nodes
  for i=1:nd
    for j=1:nc
      d(j)=distance(C(j).c,y(:,i));     // distances of point from centers
    end
    [xxx,k]=min(d);           // minimal distance point from the k-th center
    C(k).cd=[C(k).cd y(:,i)];
  end

  df=0;
  for j=1:nc
    C(j).cs=C(j).c;           // remember centers from last step
    C(j).c=mean(C(j).cd,2);   // new centers
    df=df+sum(abs(C(j).c-C(j).cs));     // shift of centers
  end
  if df<.1
    break                     // end of iterations
  end
end

// RESULTS
k1=1:n(1);
k2=k1($)+1:k1($)+n(2);
k3=k2($)+1:k2($)+n(3);
set(scf(),’position’,[800 200 600 400]);
```

```
// data
plot(y(1,k1),y(2,k1),'kd','markersize',12)
plot(y(1,k2),y(2,k2),'ks','markersize',12)
plot(y(1,k3),y(2,k3),'ko','markersize',12)
// clusters
plot(C(1).cd(1,:),C(1).cd(2,:),'r.','markersize',3)
plot(C(2).cd(1,:),C(2).cd(2,:),'b.','markersize',3)
plot(C(3).cd(1,:),C(3).cd(2,:),'g.','markersize',3)
title('Data and found clusters','fontsize',4)

disp(it,'number of iterations')
```

Description of the program

Definition of the distance

Simulation

Three centers m, standard deviation of data in clusters sd are set. Two dimensional data generated in loop. In the i-th cluster $n(i)$ data points are simulated.

Algorithm

Structure variable C is defined. It has items $.c0$ - initial centers, $.c$ - new centers, $.cs$ - centers from previous step, $.cd$ - points in a cluster. It runs according to the list above.

12.2 K-medoids algorithm

This algorithm is similar to k-means with the difference, that centers (medoids) are always data points. The algorithm is:

0. Randomly select m data points - medoids. The rest of points are called non-medoids.

0. To each medoid find the non-medoids that are closest to it. They form clusters.

0. Determine overall distance of non-medoids from their medoids.

1. Randomly select one medoid and one non-medoid and swap them.

2. Re-construct clusters and determine overall distance.

3. If the distance is repeatedly not smaller, stop the algorithm othervise continue by 1.

Program

```
// DM_cmedoids.sce
// c-medoids (simple - like genetic alg.)
// ------------------------------------------
clc, clear, close, mode(0)
```

```
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction
function d=distXY(X,Y)
  // Distance of vectors X and Y
  nX=size(X,2);
  nY=size(Y,2);
  d=zeros(nX,nX);
  for i=1:nX
    for j=1:nY
      d(i,j)=distance(X(:,i),Y(:,j));
    end
  end
endfunction
function dc=updateCls(md,s,u,y)
  // update of all distances after update of medoids
  // dc   distances points from individual medoids: matrix md X nd
  // md   nuber of clusters
  // s    indexes of medoids
  // u    indexes of non-medoids
  // construction of new clusters
  d0=distXY(y(:,s),y(:,u));   // distances between medoids and non-medoids
  [xxx,ic]=min(d0,'r');       // ic(k) is label of cluster
  c=list();                   //        to which y(:,k) belongs
  for j=1:md
    c(j)=find(ic==j);         // c(k) is vector of indxes of y
  end                         //        which belong to cluster k
  // evaluation of new clusters

  for j=1:md
    dc(j)=sum(distXY(y(:,s(j)),y(:,c(j))));
  end                         // sum of distances data from medoids
endfunction                   //        = optimality criterion
// ====================================================================

// SIMULATION
m=list();
m(1)=[1 1]';                  // data centers
m(2)=[5 2]';
m(3)=[3 8]';
sd=.5;                        // std of data
al=fnorm([1 3 2]);            // prabalilities of modes
ny=length(m(1));              // dimension of y
nc=length(al);                // number of modes
```

```
nd=200;                        // length of data
md=3;                          // number of initial centers (points)

for t=1:nd
  i=sum(rand(1,1,'u')>cumsum(al))+1;
  y(:,t)=m(i)+sd*rand(ny,1,'n');   // data generation
end

// CLUSTERING - first step
s=samwr(1,md,1:nd);            // first medoids
u=setdiff(1:nd,s);             // first non-medoids
dc=updateCls(md,s,u,y);        // distances within initial clusters
d0=sum(dc);

dd=d0;
dd0=d0;
ss=s';

// CLUSTERING - iterations
for ite=1:1000
  s0=s;                        // remember medoids from last step
  u1=samwr(1,1,u);             // choice of one non-medoid
  s1=samwr(1,1,s);             // choice of one medoid
  // swap one medoid and one non-medoid
  s=setdiff(s,s1);
  s=[s,u1];                    // new medoids
  u=setdiff(1:nd,s);           // remaining non-medoids

  dc=updateCls(md,s,u,y);      // new distances in clusters
  d=sum(dc);

  if abs(d-d0)<.001            // test of end of iterations
    printf(' Počet kroků  %d\n\n',ite)
    break
  end

  if d<d0                      // test in the end of iteration (go on / go back)
    d0=d;                      // crit OK - remember its value
  else
    s=s0;                      // crit is not OK - go back to original medoids
  end

  // remember
  dd=[dd d];
  dd0=[dd0 d0];
  ss=[ss s'];
end
chk=[dd0;dd;ss];
```

```
// RESULTS
C=y(:,s);
scf();
plot(y(1,:),y(2,:),'.')
plot(C(1,:),C(2,:),'rx','markersize',12)
```

Program description

Function definition

- updateCls recomputes centers and evaluates the overall distance of points from medoids within individual clusters.

Simulation

Two dimensional data y are generated. nd is number of data, md is number of clusters.

Initialization

Select medoids, the rest of points are non-medoids. Compute the overall distance.

Iterations

Chose one medoid and one non-medoid. Swap them. Compute the overall distance and compare with the previous one. Check for the end.

12.3 Fuzzy clustering

C-means algorithm

In the c-means algorithm we minimize criterion

$$J = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^k \|x_i - c_j\|^2, \ k \geq 1$$

where u_{ij} is a degree of membership of the point x_i to cluster c_j and $\|\cdot\|$ is a norm.

The update of weights u_{ij} is performed as follows

- determine the centers (weighted average - follows from the criterion)

$$c_j = \frac{\sum_{i=1}^N u_{ij}^k x_i}{\sum_{i=1}^N u_{ij}^k}$$

- weights (are given as membership functions)

$$u_{ij} = \frac{1}{\sum_{z=1}^m \left(\frac{\|x_i - c_j\|}{\|x_i - c_z\|} \right)^{\frac{2}{k-1}}} \tag{12.1}$$

Algorithm

0. Compute the initial matrix of membership U.
1. Construct the centers c_j with existing matrix U.
2. Update the matrix U.
3. If $\|U_{new} - U_{old}\| < \epsilon$, END otherwise go to 1.

Program

```
// DM_cmeans.sce
// c-means (fuzzy)
// Remark: weights are computed in the function CMupdt
// -----------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction
// -----------------------------------------
function d=distXY(X,Y,p)
  // Distance of vectors X and Y
  if argn(2)<3, p=1; end
  nX=size(X,2);
  nY=size(Y,2);
  d=zeros(nX,nY);
  for i=1:nX
    for j=1:nY
      d(i,j)=distance(X(:,i),Y(:,j),p);
    end
  end
endfunction
// -----------------------------------------
function [c,d]=CMupdt(c,y)
  // computation of weights and centers
  // c    clusters
  // y    data

  // distances  d
  d=distXY(c,y);               // distances of points and medoids

  // weights  u
  v=ones(d)./(d+1e-8);        // membership function
  u=fnorm(v,1);               // normoalization over clusters

  // centers  c
  un=fnorm(u,2);              // normalization over points
  for j=1:size(c,2)
    c(:,j)=y*un(j,:)';
  end
endfunction
// -----------------------------------------
function c=clusters(dn)
```

```
  // indexes of points for individual clusters
  // dn   normed distances
  // c    list of indexes of points for clusters
  [xxx,ic]=min(dn,'r');
  c=list();
  for j=1:size(dn,1)
    c(j)=find(ic==j);         // clusters
  end
endfunction
// ------------------------------------------

// SIMULATION
cS=list();
cS(1)=[1 1]';                           // centers for simulation
cS(2)=[5 2]';
cS(3)=[3 8]';
sd=.8;                                  // stdev of points
al=fnorm([1 3 2]);                      // prabablities of modes
ny=length(cS(1));                       // dimension of y
nc=length(al);                          // number of modes
nd=50;                                  // length of data
md=3;                                   // number of initial centers (points)

// SIMULATION
for t=1:nd
  i=sum(rand(1,1,'u')>cumsum(al))+1;
  y(:,t)=cS(i)+sd*rand(ny,1,'n');
end

// CLUSTERING - first step
p=2;                                    // distance  [(p-q)'*(p-q)]^(p/2)
j=fix(nd*rand(1,md,'u'))+1;             // indexes of initial centers
c=y(:,j);                               // initial centers
[c,d0]=CMupdt(c,y);                     // first update of centers
sd0=sum(d0);

// CLUSTERING - iteration
for ite=1:1000
  [c,d]=CMupdt(c,y);                    // new centers (medoids)

  sd1(ite)=sum(d);
  if abs(sd1(ite)-sd0)<.001             // test of the end of iterations
    printf(' Počet iterací  %d\n',ite)
    break
  end
  sd0=sd1(ite);
end

cL=clusters(d);                          // indexes of points in clusters
```

```
// RESULTS
tx=['r.';'m.';'g.'];
scf();
plot(y(1,:),y(2,:),'x','markersize',7)
for j=1:md
  if ~(isempty(y(1,cL(j))) | isempty(y(2,cL(j))))
  plot(y(1,cL(j)),y(2,cL(j)),tx(j),'markersize',6)
  end
end
plot(c(1,:),c(2,:),'s')
```

Program description

Function definitions

- CMupdt computes distances of points from centers. First normalizes over clusters and then over points. Finally creates clusters using the u-weights.

- clusters constructs clusters according to the distances m.

Simulation - standard

Initialization - updating of clusters (new centers)

Iterations - update of clusters (new clusters). Check for end of the algorithm.

12.4 Density based clustering

Dbscan

We have a set of data $X = \{x_1, x_2, \cdots, x_N\}$, where $x_i \in R^n$

We define:

- **Distance** of two points x and y and denote it by $d(x, y)$.
- ϵ-**neighborhood** of point x
$$O_\epsilon(x) = \{x \in X : d(x, y) < \epsilon\}.$$
- **Inner point** is such one that has in its neighborhood at least given number of points.
- A point y is **accessible** from the point x, if a sequence of inner points from x to y exists.
- A **connection** between points x a y exists, it both these points are accessible from some inner point.

Algorithm of clustering

1. For each point from X find its ϵ-neighborhood.
2. Define variables “clus” and “buff” (for storing points).

3. To “clus” put a single inner point and to “buff” its neighborhood.
4. Select one point (e.g. the first one) from “buff”. Add it to “cluss” and its neighborhood add to “buff”.
5. From “buff” remove all points that have already been used (those that are in some cluster).
6. Repeat from 4. until “buff” is not empty. Otherwise continue.
7. Remember the created cluster “clus” and prepare the variable for new one.
8. If there exists another free inner point, put it to “clus” and go to 4. If not, stop the algorithm.

Clusters are formed by points that are connected.

Program

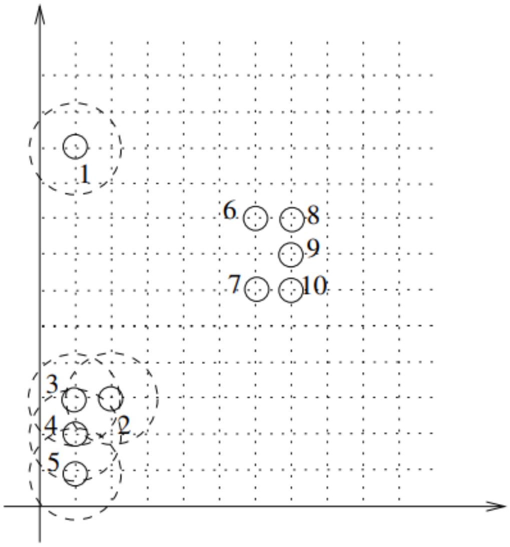
```
// DM_dbscan.sce
// Dbscan
// ------------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction
function b=board(x)
  // boards for graph
  b=[min(x(1,:))-.2 max(x(1,:))+.2 min(x(2,:))-.2 max(x(2,:))+.2];
endfunction

// SIMULATION
p=[.1 .2 .1 .4 .2];                 // switchin parameter
th=[0 0; 0 3; 1 2; 2 1; 3 3]';      // centers
nd=100;                             // number of data
for i=1:nd
  j=sum(randu(1,1)>cumsum(p))+1;
  x(:,i)=.3*randn(2,1)+th(:,j);
end
bo=board(x);

// CLUSTERING
eP=.5;                              // radius of neighbourhood
mP=3;                               // minimum of points

// marking of inner points
V=[];                               // inner points
X=list();                           // neighbourhood of inner points
```

```
for i=1:nd
  X(i)=[];
  for j=setdiff(1:nd,i)
    if distance(x(:,i),x(:,j))<eP
      X(i)=[X(i) j];                // indexes of neighbourhood
    end
  end
  if length(X(i))>=mP
    V=[V i];                        // inner points
  end
end

// creation of
C=list();                 // clusters
b=V(1);                   // auxiliary variable
M=[];                     // already used points
k=1;                      // label of actual cluster
for h=1:100               // cycle for various clusters
  CC=[];                  // actual cluster
  while ~isempty(b)       // cycle for one cluster
    b1=b(1);              // one inner point
    CC=[CC b1];           // new point to cluster
    b=union(b,X(b1));     // add neighbourhood to b (auxiliary var.)
    b=setdiff(b,CC);      // removing just used point from b
  end
  if isempty(CC)
    break                 // end of algorithm
  end
  M=[M CC];               // remembering points from a cluster
  Vr=setdiff(V,M);        // inner points that are still not used
  if 1
    C(k)=gsort(CC,'g','i'); // actual cluster(with border)
  else
    C(k)=intersect(V,CC);   // actual cluster(without border)
  end
  k=k+1;                  // next cluster
  b=Vr(1);                // still not used point -> b
end
nC=length(C);             // number of clusters

// RESULTS
tx=['.r';'.b';'.g';'.m';'.k'];
set(scf(),'position',[600 100 900 400])
subplot(121)
plot(x(1,:),x(2,:),'c.')            // data
set(gca(),'data_bounds',bo)
title Data
subplot(122)
for i=1:nC
```

```
  plot(x(1,C(i)),x(2,C(i)),tx(i))  // clusters
end
set(gca(),'data_bounds',bo)
title Clusters
```

Example

Let us have 10 points as demonstrated in the picture



Points are circles and are plotted in a net with unit step. Parameter *eps* = 1.1, minimum number of points is $mp = 2$. Then points

- 3, 4, 8, 9, 10 are inner points
- 2, 5, 6, 7 are border points
- 1 is noise points.

Cluster construction

If the points are two-dimensional, the best way is to draw them in a plane (as in the picture above) and to select the clusters manually. Start with arbitrary free inner point and add to it all connected points. Repeat until all points are classified.

Here the result is:

Cluster1 = {2, 3, 4, 5} a Cluster2 = {6, 7, 8, 9, 10}.

The point 1 is noise.

12.5 Hierarchical clustering

Agglomerative clustering

There is a lot of variances of this method. We will show here one of them which is very simple. The algorithm is here:

1. All data points are denoted as clusters on the level 1 (with only one point).
2. Find two nearest clusters and join them together in one cluster. Its level is equal to the number of points in joined clusters.
3. The coordinates of the cluster lie on a connecting line of the coordinates of clusters to be joined in the proportion of their levels (the higher level the nearer).
4. Remember the clusters from which the new ones have been created (hierarchy).
5. Repeat from 2 until only one cluster remains.

Remarks

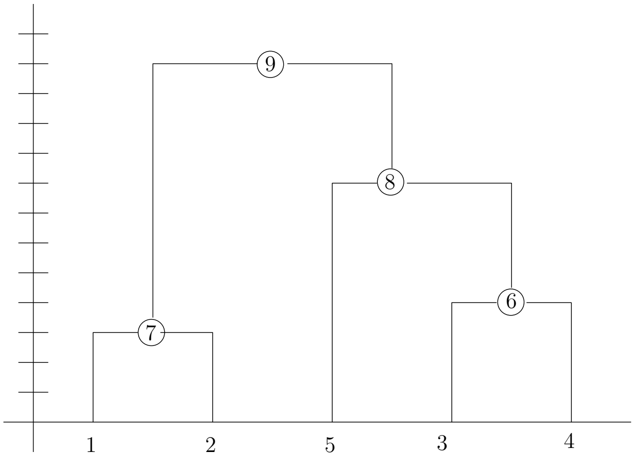
1. *The distance is Euclidean. It is computed between coordinates of clusters.*
2. *Coordinates of clusters on the level 1 are those of the points. For clusters generated by joining clusters with coordinates with levels h_i a h_j are coordinates given as follows:*
   The line connecting coordinates of the two clusters is

   $$x = x_i + t\,(x_j - x_i)\,,\ t \in (0,1)$$

   The point in the ratio of the levels (nearer to the cluster with higher level) is given by the parameter $t = \frac{h_j}{h_i+h_j}$. So

   $$x = x_i + \frac{h_j}{h_i + h_j}\,(x_j - x_i) = \frac{h_i x_i + h_j x_j}{h_i + h_j}$$

3. *Dendrogram is a special graph that shows the structure of hierarchical clustering as shown in the picture*



The resulting clusters can be determined on the basis of the dendrogram which can be drawn *manually. The program gives the matrix C, where in each row the number of cluster, the distance of the coordinates of parents, and numbers of the parents can be found. The drawing will start in the cluster with the highest number (the last row of the matrix). In the graph, in the middle of the axis x and in the level of the distance (the second column in the matrix) on the axis y, we draw a circle and write a number of the cluster inside it. In the matrix C, find the parents of the node and draw the circles with their numbers in a corresponding levels on the axis y (the position on the axis x is arbitrary). We repeat this procedure until we exhaust all clusters that have been created by joining, only clusters with level one remain.*

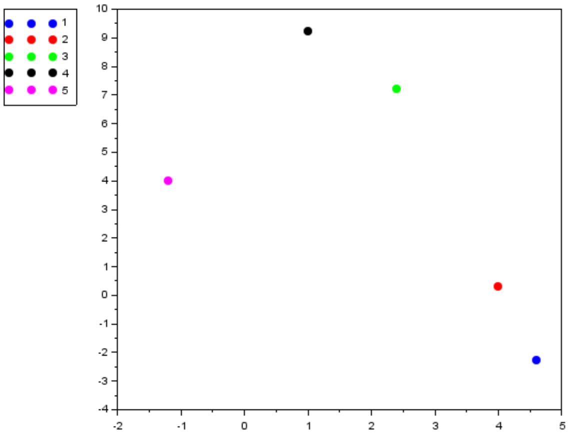
According to the desired number we can proceed as follows in determining the clusters :

We draw a horizontal line that intersect vertical lines of the dendrogram. The line can be shifted up or down. The number of intersections of the horizontal line with the vertical ones gives the number of created clusters. The points belonging to individual clusters are in the axes x below the intersection.

Example

We have 5 points

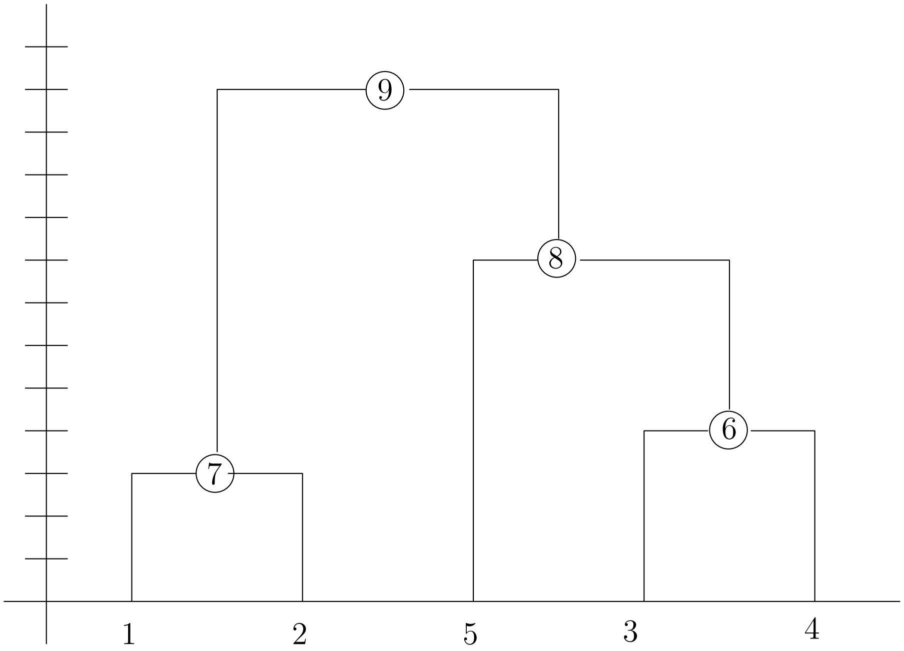
i	1	2	3	4	5
y_1	4.6	4.0	2.4	1.0	-1.2
y_2	-2.3	0.3	7.2	9.2	4.0



The matrix C is

$$C = \begin{bmatrix} 6, & 2.44, & 3, & 4 \\ 7, & 2.67, & 1, & 2 \\ 8, & 5.10, & 5, & 6 \\ 9, & 8.58, & 7, & 8 \end{bmatrix}$$

Construction of dendrogram starts with the last row (the cluster 9). We draw a circle in the middle of the axis x and in the height 8.58. Its parents are clusters 7 and 8. Those can be drawn to the left and right from the node 9 in heights 2.67 and 5.10. We continue in this way until we obtain the dendrogram according to the following picture



If we cut the dendrogram so that we obtain three intersections, we obtain the clusters

$$C_1 = \{1,2\}\,,\ C_2 = \{5\}\,,\ C_3 = \{3,4\}\,.$$

A comparison with the data plot confirms the clusters created.

Two clusters would be $C_1 = \{1,2\}$ a $C_2 = \{3,4,5\}$.

Program

```
// DM_hierAgl.sce
// Hierarchical clustering (agglomerative)
// ----------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y)
  x=x(:); y=y(:);
  e=x-y;
  d=sqrt(e'*e);
endfunction
// ============================================================

// SIMULATION -------------------------------------------------
```

```
m=list();
m(1)=[1 1]';                       // centers
m(2)=[5 2]';
m(3)=[3 8]';
sd=2.5;                            // stdev of points
al=fnorm([1 3 2]);                 // switching parametwr
ny=length(m(1));                   // number of variables
nc=length(al);                     // number of modes
nd=5;                              // number of data

for t=1:nd
  i=sum(rand(1,1,'u')>cumsum(al))+1;
  y(:,t)=m(i)+sd*rand(ny,1,'n');   // simulation
  c(1,t)=i;
end

// structre variable definition
for i=1:nd
  cL(i).y=y(:,i);   // data point or cluster
  cL(i).n=1;        // numb. of points in cluster
  cL(i).p=[];       // parents
  cL(i).v=[];       // distance
end

// ALGORITHM
nc=nd;
cc=1:nd;
for ite=1:(nd-1)
  lc=length(cc);
  d=zeros(lc,lc);
  for i=1:lc
    for j=1:lc
      if i<j
        d(i,j)=distance(cL(cc(i)).y,cL(cc(j)).y); // distances between points
      else
        d(i,j)=%inf; // symmetrical entries
      end
    end
  end

  // grouping points
  [v,ii]=min(d);    // nearest point
  i1=ii(1);
  i2=ii(2);

  nc=nc+1;
  n1=cL(cc(i1)).n;
  n2=cL(cc(i2)).n;
  y1=cL(cc(i1)).y;
  y2=cL(cc(i2)).y;
  cL(nc).y=(n1*y1+n2*y2)/(n1+n2);  // joining two points (clusters)
  cL(nc).n=n1+n2;
  cL(nc).p=cc(ii);
  cL(nc).v=v;
  cc(ii)=[];
  cc=[cc nc];
end

// determining clusters
C=[];
for i=(nd+1):(2*nd-1)
  C=[C; [i cL(i).v cL(i).p]];
end

// RESULTS
set(scf(),’position’,[800 100 600 500])
tx=[’.r’;’.b’;’.g’;’.k’;’.m’;’.y’];
tn=[’1’,’2’,’3’,’4’,’5’];
for i=1:nd
  plot(y(1,i),y(2,i),tx(i,:),’markersize’,8)
end
legend(tn(1:nd),-2);

disp(’ node  distance    p1   p2  (p = parents)’)
disp(C)
```

Divisive clustering

In divisive clustering we proceed from top to bottom. We start with one cluster that contains all data points and subsequently we divide clusters so that there would be minimal point distances in clusters and maximal distances between clusters. For a given definition of the distance $D(x,y)$ we introduce following notions

Big cluster C_T - is a cluster to be divided.

Left and right cluster C_L a C_R - clusters created by division

Distance between clusters C_L and C_R - I_{LR}

Distance inside clusters - U_L, $U_{\acute{R}}$

Distance of the divided cluster - $U_T = I_{LR} + U_L + U_R$ (it is sum of distances from each point from C_L to each point from C_R - it is independent on division)

Task: to find C_L a C_R so that

$$H_{LR} = (1-\alpha)\underbrace{I_{LR}}_{H_1} - \alpha\underbrace{[U_L + U_R]}_{H_2} \rightarrow \min$$

This task is combinatorial and it is *np*-hard. For its approximative numerical solution we will use the method called

Avalanche method.

We have a cluster C_T (in the beginning the whole data sample), which is to be divided.

We introduce C_L as an empty set and C_R as the whole cluster C_T.

1. In C_R we find anti-medoid - i.e. the point which is maximally remote from all other points in the cluster C_R.
2. We shift anti-medoid into the cluster C_L a compute the value of the criterion H_{LR}.
3. We try to add another point that is closer to the previously added.
4. If the value of the criterion increases we leave the point in C_L and we go to the point 3. If it dos not increase, the algorithm ends.

Program

```
// DM_hierDiv.sce
// Hierarchical clustering (divisive)
// ------------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction
// ------------------------------------------
function d=distXY(X,Y,p)
  // Distance of vectors X and Y
  if argn(2)<3, p=1; end
  nX=size(X,2);
  nY=size(Y,2);
  d=zeros(nX,nY);
  for i=1:nX
    for j=1:nY
      d(i,j)=distance(X(:,i),Y(:,j),p);
    end
  end
endfunction
function h=H1(cL,cR,y)
  // sum of mutual distances of points
  h=0;
  for i=cL
    for j=cR
      h=h+distance(y(:,i),y(:,j));
    end
  end
```

```
endfunction

// ===================================================================
// DATA
nd=120;
py=[.3 .5 .2];
th=[-2 6; 6 3; 8 15]';
cv=3;
for i=1:nd
  iy=sum(rand(1,1,'u')>cumsum(py))+1;
  y(:,i)=th(:,iy)+cv*rand(2,1,'n');
end

// ALGORITHM
c=list();
cL=1:size(y,2);                // data indexes
for itA=1:100                  // iterations between clusters
  cR=[];

  // initialization
  D=distXY(y(:,cL),y(:,cL));
  Da=sum(D,2);
  [xxx,cc1]=max(Da);             // cc1 - pointer to anti-medoid
  c1=cL(cc1);                    // c1  - index of anti-meoid
  ci=cc1;                        // ci  - storing of used clusters

  cL=setdiff(cL,c1);            // old (all)
  cR=union(cR,c1);              // new (is added)

  h2=H1(cL,c1,y);

  // iterations in one cluster
  for ite=1:100
    Dn=zeros(cL);
    for i=1:length(cL)
      Dn(i)=distance(y(:,c1),y(:,cL(i)));
    end
    [xxx,cc2]=min(Dn);
    c2=cL(cc2);
    ci=[ci cc2];                   // is added for trial use

    cL=setdiff(cL,c2);
    cR=union(cR,c2);
    h1=H1(cL,cR,y);

    c1=c2; cc1=cc2;

    if h1<=h2
      ci=setdiff(ci,cc2);          // if not used, it is removed
      break
    end
      h2=h1;
  end
  c(itA)=cR;
  if isempty(cL), break, end
end

// RESULTS
tx=['.r';'.b';'.g';'.k';'.m';'.y';'*r';'*b';'*g'];
set(scf(),'position',[800 100 600 500])
for i=1:length(c)
  plot(y(1,c(i)),y(2,c(i)),tx(i))
end
c
```

13 Classification

By classification we mean assignment of a data record (point) to some cluster or more clusters each with its probability. Here, we mostly assume, that clusters have already been created by some clustering method.

13.1 K-nearest neighbour

It is a basic form of classification.

We have data $X = \{x_i\}_{i=1}^N$ with detected clusters. We can get them using some method of clustering. The task is: for a newly measured data point y, to assign it to some cluster.

The procedure of classification is following:

1. Compute the distance of the point y from all points from $x_i \in X$.
2. Determine k points x_i, $i = 1, 2, \cdots, k$ nearest to y.
3. Assign y to the cluster to which majority of the k nearest points belongs.

Remark

If there are more than one such cluster, take the first of them.

Program

```
// DM_knearest.sce
// K nearest neighbour
// ----------------------------------------
clc, clear, close, mode(0)
getd _func
function d=distance(x,y,p)
  // Euclidean distance (for p=1)
  if argn(2)<3, p=1; end
  x=x(:); y=y(:);
  e=x-y;
  d=(e'*e)^(p/2);
endfunction
// ----------------------------------------
function d=distXY(X,Y,p)
  // Distance of vectors X and Y
  if argn(2)<3, p=1; end
  nX=size(X,2);
  nY=size(Y,2);
  d=zeros(nX,nY);
  for i=1:nX
    for j=1:nY
      d(i,j)=distance(X(:,i),Y(:,j),p);
    end
  end
endfunction
function tx=scfmark()
  // marks for plot
  tx=['.b';'.r';'.g';'.k';'.m';
      'xb';'xr';'xg';'xk';'xm';
      'db';'dr';'dg';'dk';'dm';
      'sb';'sr';'sg';'sk';'sm';
      '*b';'*r';'*g';'*k';'*m';
      'pb';'pr';'pg';'pk';'pm';
      '+b';'+r';'+g';'+k';'+m';
      'ob';'or';'og';'ok';'om'];
endfunction
function [h,f]=vals(a)
  // [h f]=vals(a)  find different values of a variable
  //                and their frequencies
  // h       values and frequencies  [vals;abs_freq]
  // f       relative frequencies

  a=a(:)';
  b=gsort(a,'g','i');
  [v,m]=unique(b);
  dm=diff(m);
  n1=length(b)+1;
  n=[dm n1-m($)];
  f=n/sum(n);
  h=[v(:)';n];

  if sum(n)~=max(size(a))
    disp('Error: in vals.sci')
    return
  end
endfunction
// =================================================================

// SIMULATION
m=list();
m(1)=[1 1]';
m(2)=[5 2]';
m(3)=[3 8]';
sd=2.5;
al=fnorm([1 3 2]);
ny=length(m(1));
nc=length(al);
nd=130;

for t=1:nd
  i=sum(rand(1,1,'u')>cumsum(al))+1;
  y(:,t)=m(i)+sd*rand(ny,1,'n');   // data generation
  c(1,t)=i;
end

// ALGORITHM
k=15;                              // k nearest neighbour (this is k)
i=sum(rand(1,1,’u’)>cumsum(al))+1;
z=m(i)+sd*rand(ny,1,’n’);          // choice of a point
ic=i;

d=distXY(z,y);
[ds,j]=gsort(d,’g’,’i’);
jk=j(1:k);                         // the nearest k points
ck=c(jk)
v=vals(ck);
[xxx,i]=max(v(2,:));
cz=v(1,i)

// RESULTS
tx=[’.r’;’.b’;’.g’;’.k’;’.m’;’.y’;’*r’;’*b’;’*g’];
scf();
for j=1:length(m)
  i=find(c==j);
  plot(y(1,i),y(2,i),tx(j),’markersize’,3)
end
legend(’1’,’2’,’3’);
plot(z(1),z(2),’ko’,’markersize’,8)
```

13.2 Decision trees

Let us have discrete data records $x_t = [x_1, x_2, \cdots, x_n]_t$, $t = 1, 2, \cdots, N$ and a pointer variable $c_t \in \{1, 2, \cdots, m\}$ which is a label of the class (cluster) to which the record x_t belongs.

The principle of tree construction if following:

We construct a matrix from the data records and add the pointer variable c_t as its last column. We have matrix $N \times (m + 1)$

$$X = [x_{ti}, \, c_t] \, , \, t = 1 : N, \, i = 1 : m$$

We chose some variable x_i and according to its values we sort the remaining parts of the matrix into groups. Then, in each group we again select a variable and do the same. We repeat this procedure until each group contains only one value of the pointer. If some final group has more than one pointer value, the decision is probabilistic.

It is clear that the subsequent choice of variables is very important for a success of the task. However, the proper choice is a combinatorial task for which we need to use some heuristic methods. One of them is illustrated in the following example.

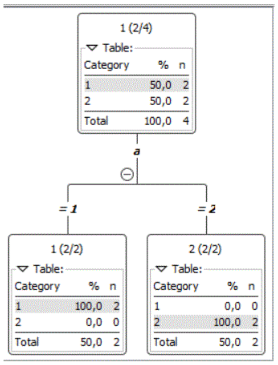
Example

Let us have the following data

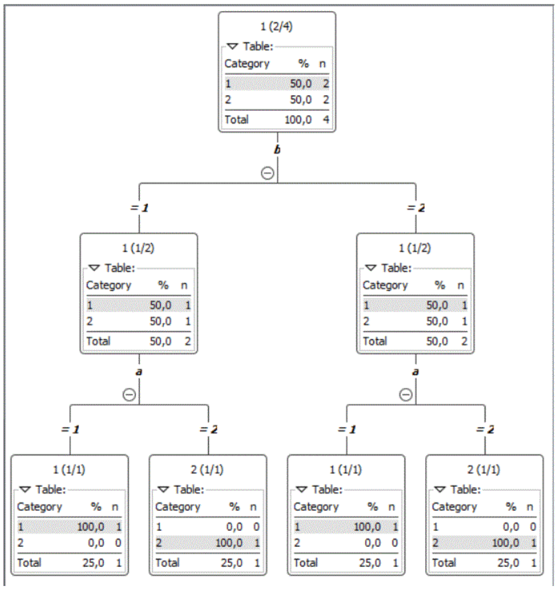
t	x_1	x_2	c
1	1	1	1
2	1	2	1
3	2	1	2
4	2	2	2

where x_1, x_2 are data records and c is pointer variable.

It is evident, the variable x_1 decides about the classification (on the basis of only the variable x_1 we can decide about classes of all records). The tree for the order of variables x_1 - x_2 is



If we swap the order of variables to x_2 - x_1 we get the tree longer and more complex

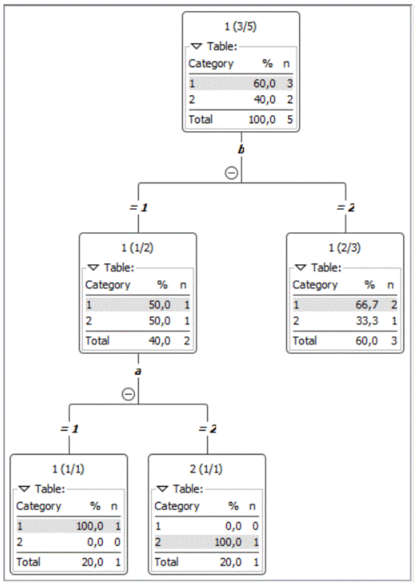


However, both the trees led to deterministic decision making (the final percent are 100%).

If we supply the data by one more record (the last row of the table)

t	x_1	x_2	c
1	1	1	1
2	1	2	1
3	2	1	2
4	2	2	2
5	2	2	1

which is in contradiction with the others, the thee will be like this



1 (3/5)

▽ Table:

Category	%	n
1	60,0	3
2	40,0	2
Total	100,0	5

1 (1/2)

▽ Table:

Category	%	n
1	50,0	1
2	50,0	1
Total	40,0	2

1 (2/3)

▽ Table:

Category	%	n
1	66,7	2
2	33,3	1
Total	60,0	3

1 (1/1)

▽ Table:

Category	%	n
1	100,0	1
2	0,0	0
Total	20,0	1

2 (1/1)

▽ Table:

Category	%	n
1	0,0	0
2	100,0	1
Total	20,0	1

In the second layer, the decision is probabilistic..

Implementation of the task in KNIME

We take an example from web https://tanthiamhuat.files.wordpress.com/2015/10/decision-tree-tutorial-by-kardi-teknomo.pdf

Example

The data bring information about the ways in which people go to work.

sex	has a car?	fare	income	**way**
M	0	L	N	**B**
M	1	L	S	**B**
Z	1	L	S	**V**
Z	0	L	N	**B**
M	1	L	S	**B**
M	0	S	S	**V**
Z	1	S	S	**V**
Z	1	D	V	**A**
M	1	D	S	**A**
Z	1	D	V	**A**

(13.1)

kde “sex”, “car”, “fare” and “income” data records and “way” is a value of the pointer variable.

The values of variables are:

sex: M = man, Z = woman;

car: 0 - does not have, 1 - has

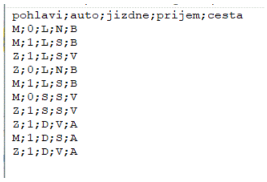
fare: L - low, S = medium, V - high;

way: B - bus, V - train, A - car.

The task is to decide about the way (B, V, A) on the basis of the values in data records.

We are going to show the solution in KNIME.

1. Data can be set into table e.g. in EXCEL and exported as csv table to disk.

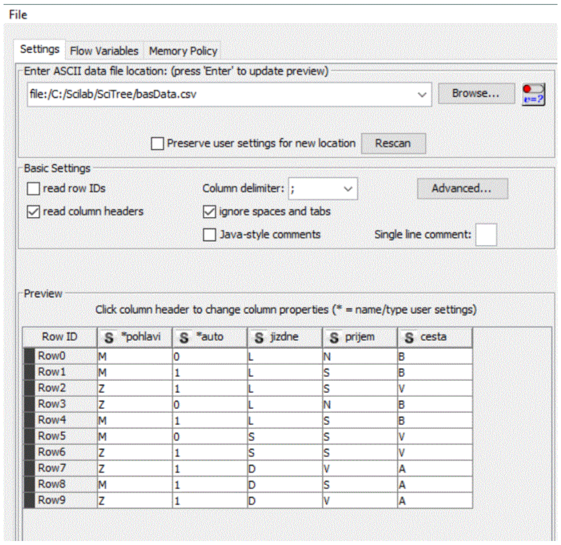


```
pohlavi;auto;jizdne;prijem;cesta
M;0;L;N;B
M;1;L;S;B
Z;1;L;S;V
Z;0;L;N;B
M;1;L;S;B
M;0;S;S;V
Z;1;S;S;V
Z;1;D;V;A
M;1;D;S;A
Z;1;D;V;A
```

2. In KNIME we open a New KNIME workflow (icon new).

3. In KNIME in the left side there is a window Node Repository (here icons of various tasks are found).

   (a) In IO we find Read and File reader and drag it by mouse to the working area. An icon of the Reader appears. We click on it by left mouse button (or twice by the right) and we obtain menu Configuration



Enter ASCII data file location: (press 'Enter' to update preview)

file:/C:/Scilab/SciTree/basData.csv

□ Preserve user settings for new location

Basic Settings

□ read row IDs

☑ read column headers

Column delimiter: ;

☑ ignore spaces and tabs

□ Java-style comments

Single line comment:

Preview

Click column header to change column properties (* = name/type user settings)

Row ID	S *pohlavi	S *auto	S jizdne	S prijem	S cesta
Row0	M	0	L	N	B
Row1	M	1	L	S	B
Row2	Z	1	L	S	V
Row3	Z	0	L	N	B
Row4	M	1	L	S	B
Row5	M	0	S	S	V
Row6	Z	1	S	S	V
Row7	Z	1	D	V	A
Row8	M	1	D	S	A
Row9	Z	1	D	V	A

Here (up) we can set the name of the data csv file. Most of the rest is set automatically.
But **important** !!!

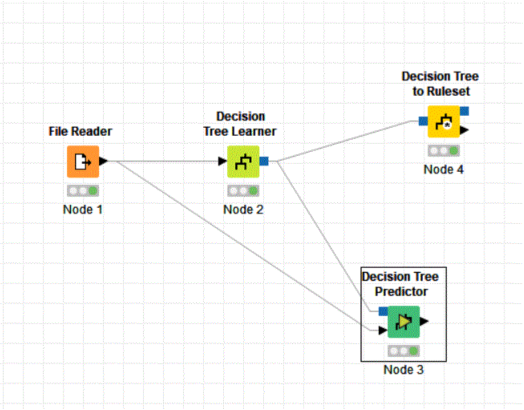
- The pointer variable must be set as string. The rest of variables can stay as they are.
- Strings are sorted by values the other by intervals.
- The change of the variable type can be done in the menu which can be obtained by clicking at the title of the variable in the data table below. After a click a menu window appears in which the type can be selected.
- W click once again at the icon of the task and select Execute (or press F7).

(b) Next, in the window Node Repository open the folder Analytics and Mining and select the tool Decision Tree Learner, drag it to working area and by mouse connect it with the Reader (by the black small triangles).
Press F7.

(c) Further, we can choose the tool Decision Tree Prediction, and possibly Decision Tree to Ruleset. The small triangles are always connected to Reader, small blue rectangles subsequently with the new tool (they generate the model of the task).

4. The results can be stored by the tool IO/Write/CCV Writer or directly checked by clicking by the left mouse and opening
in Learner the menu Decision tree view
in Prediction the menu Classified Data
in Ruleset the menu Rules table

The overall view on the task in KNIME is following



Remark

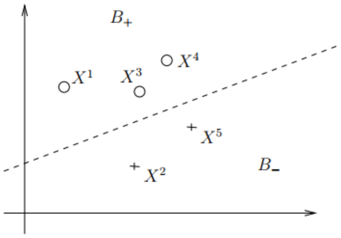
If the tree ends prematurely, it is necessary to set Number of records per node = 1 in the menu Configure in the tool Decision Tree Learner. It means that the decision rule can be derived from only one data record.

13.3 Support vector machines

In this task, we are going to find hyperplane in the data space that separates the space into two sub-spaces, one with $y = 1$ and second with $y = -1$. If the points are linearly separable, the result will be without errors. In addition, we demand so that the hyperplane would separate the points optimally. It means that the points should lay as far as possible from the hyperplane.

Theory

We will demonstrate the task in a plane (with two variables). The data sample is $X = \{x_1, x_2, \cdots, x_N\}$ where $x_i = [x_1, x_2]_i$ is i-th data record. In this case, the hyperplane will be a line as indicated in the picture



Here we have a sample of five points x_1, x_2, x_3, x_4 and x_5. The separating line is drawn dashed and it separates the points whose attributes are "circles" (up) and "crosses" (down). The attributes can be expressed numerically by 1 and -1 as values of a variable y

x	x_1	x_2	x_3	x_4	x_5
y	1	-1	1	1	-1

The points with $y = 1$ form the set B_+, those with $y = -1$ the set B_-. So, it is

$$B_+ = \{x_1, x_3, x_4\}, \text{ and } B_- = \{x_2, x_5\}.$$

The task is to find a line which separates the points and maximizes the distance of points from itself.

Let us denote the separating line as $\alpha' x + \beta = 0$. The parallel line above it is $\alpha' x + \beta + \delta = 0$ and below it $\alpha' x + \beta - \delta = 0$ for any $\delta > 0$. All these equations are over-parameterized, i.e. can be divided by some nonzero number. We will divide them by δ and get

separating line

$$w' x + b = 0$$

lines above and below

$$w' x + b \pm 1 = 0$$

For all x_1 above the above line we have the condition

$$w' x + b + 1 > 0$$

and below the below line the condition is

$$w' x + b - 1 < 0.$$

The second condition can be multiplied by -1

$$-(w' x + b) + 1 > 0$$

and using the fact that $y_i = -1$ for all x_i below and $y_i = 1$ for x_i above, we have

$$y_i (w' x_i + b) + 1 > 0$$

this single condition for all the points x_i (compare the original condition above and the modified condition below). The equality holds for parallels as borders of the above and below area.

Now, we want the above and below lines would be as far as possible one from the other. The distance of parallel lines is measured as a distance of intersections of the lines and a vertical to them. Such a vertical has equation

$$x = m + t \frac{w}{|w|}$$

where m is a fixed point, x is arbitrary point on the vertical and t is a parameter. $|w|$ is the length of w and thus $\frac{w}{|w|}$ is a unit vector. In this case the distance of the points x and m is

$$|x - m| = t \frac{|w|}{|w|} = t,$$

and it is directly equal to t. Now, we choose that x is a point on the parallel and m lies on the separating line. Then x must fulfill the equation for the parallel and m for the separating line. Tu this end we multiply the previous equation by w', add b to both sides and we obtain

$$|\underbrace{w' x + b + 1}_{=0\,(\text{parallel})} - 1 - \underbrace{w' m + b}_{=0\,(\text{separ.})}| = +t \frac{w' w}{|w|}$$

and the result is

$$1 = t\frac{w'w}{|w|} = t|w|$$

The distance is

$$t = \frac{1}{|w|}$$

which is to be maximized. From it the task is

$$|w| \to \min$$

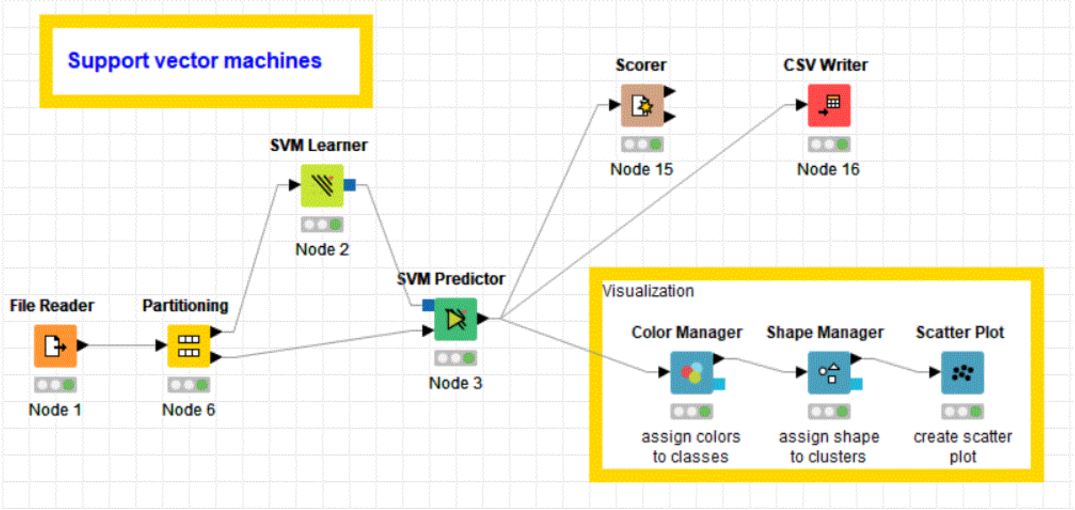
on condition that

$$y_i\left(w'x_i + b\right) + 1 > 0$$

As both w and b are to be optimized, the task is nonlinear and the solution rather complex.

Program KNIME

Create tho following program scheme



Block 1: Reading data.

Block 6: Division of data to learning and training parts.

Block 2: Estimation (learning).

Block 3: Prediction (classification).

Block 15: Frequencies of classification (table: from / to).

Block 16: Write results to disk.

Block in the yellow frame: Show graph of the found clusters .

Remarks

1. The results can be found after clicking on the task icon down in the menu.

2. The data file can be changed directly on disk. If there are new variables (not only values), it is necessary to perform new Configuration otherwise only to run Execute.

3. If the results are stored on disk, we have a possibility to investigate them in some other program - probably in Excel. To this end it is necessary to:

   (a) Set semicolon as data delimiter - in menu menu of the icon of CSV Writer, in the item Configure / Advanced.

   (b) In the menu Configure / Settings it is good to set Overwrite in the item If file exists ...

Scatter plot

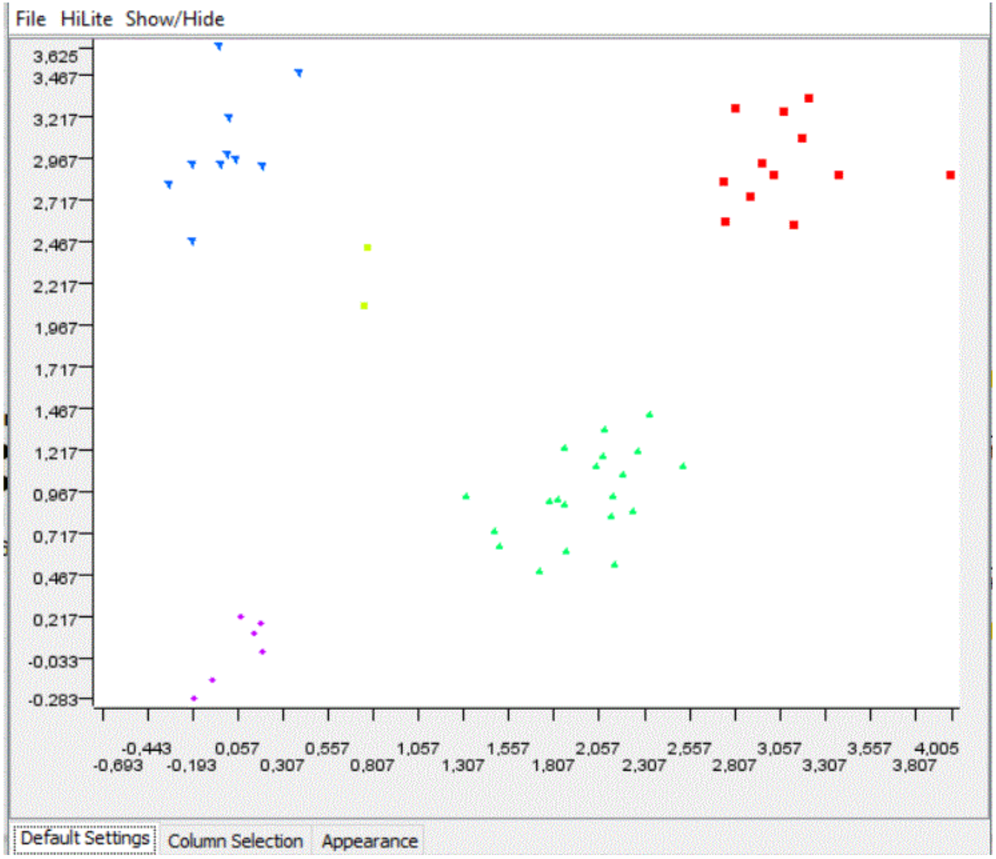
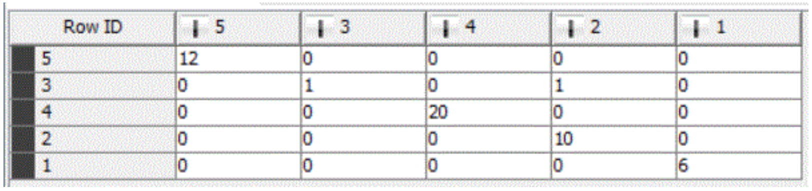


Table of classifications



Row ID	5	3	4	2	1
5	12	0	0	0	0
3	0	1	0	1	0
4	0	0	20	0	0
2	0	0	0	10	0
1	0	0	0	0	6

Part III

Supplements

14 Competition to squares

For normal distribution it is equivalent to decomposition of joint pdf to conditional and marginal ones. Is useful for integration.

Scalar case

For scalar variables x a y and constants a, b, c it holds

$$ax^2 + 2bxy + cy^2 = a\left[x^2 + 2x\frac{b}{a}y + \left(\frac{b}{a}y\right)^2 - \left(\frac{b}{a}y\right)^2\right] + cy^2 =$$

$$a\left(x + \frac{b}{a}y\right)^2 + cy^2 - \frac{b^2}{a}y^2 = a\left(x + \frac{b}{a}y\right)^2 + \frac{ac - b^2}{a}y^2.$$

Vector case

For variables x a y (column vectors) and constant matrices A, B, C with corresponding dimensions, A a C symmetric, it holds

$$x'Ax + 2x'By + y'Cy = x'Ax + 2x'AA^{-1}By + \left(A^{-1}By\right)' AA^{-1}By - \left(A^{-1}By\right)' AA^{-1}By + y'Cy =$$

$$= \underbrace{\left(x + A^{-1}By\right)' A\left(x + A^{-1}By\right)}_{\text{kvadrát}} + \underbrace{y'\left(C - B'A^{-1}B\right)y}_{\text{zbytek}}.$$

Remark

For vectors it holds ax^2 corresponds to $x'Ax$.

15 Natural conditions of control

Assumption, that the information about unknown parameter Θ and used for construction of new control variable u_t is the same and it is gained only from data $d(t-1) = \{y(t-1), u(t-1)\}$. Then it holds

$$f(\Theta|u_t, d(t-1)) = f(\Theta|d(t-1)).$$

I.e. now new information about Θ can be extracted from u_t that that, which already is in $d(t-1)$.

Using Bayes rule, we can also write

$$f(u_t|\Theta, d(t-1))) = f(u_t|d(t-1))$$

Remark

Natural Conditions of Control are abbreviates as NCC.

16 Bayes rule

Derivation

$$f(A,B|C) = \begin{cases} f(A|B,C)\, f(B|C) & \text{z jedné strany, nebo} \\ f(B|A,C)\, f(A|C) & \text{z druhé strany.} \end{cases}$$

By comparison of both right hand sides we get

$$f(A|B,C)\, f(B|C) = f(B|A,C)\, f(A|C)\,.$$

From it

$$f(B|A,C) = \frac{f(A|B,C)\, f(B|C)}{f(A|C)}. \tag{16.1}$$

which can also be written as

$$f(A|B,C) \propto f(A|B,C)\, f(B|C)$$

where the constant is hidden in the proportional sign $\propto$.

Application

In estimation we have

- A is the output y_t,
- B are parameters Θ and
- Cold data $d(t-1)$ (or $\{u_t,\, d(t-1)\}$).

In this way, the Bayes rule reads

$$f(\Theta|d(t)) = \frac{f(y_t|\psi_t, \Theta)\, f(\Theta|d(t-1))}{f(y_t|d(t-1))}$$

Remarks

1. *For model it holds* $f(y_t|u_t, d(t-1), \Theta) = f(y_t|\psi_t, \Theta)$.
2. *The natural conditions* $f(\Theta|u_t, d(t-1)) = f(\Theta|d(t-1))$ *are applied.*

17 Categorical distribution

The probability function of categorical distribution is

y	1	2	$\cdots$	n_l
$f(y)$	p_1	p_2	$\cdots$	p_{n_l}

,

where p_i are probabilities, $p_i \geq 0$, $i = 1, 2, \cdots, n_l$ a $\sum_{i=1}^{n_l} p_i = 1$.

Alternative form for the pf is

$$f(y) = p_y, \; y = 1, 2, \cdots, n_l.$$

Model of discrete system is

$$f(y|\psi, \Theta) = \Theta_{y|\psi}.$$

and it can be expressed in the form of table (for $y \in \{1, 2\}$ and $\psi = [u, \, v]'$, where $u, v \in \{1, 2\}$

$f(y|u, v)$

$[u, v]$	$y = 1$	$y = 2$
1, 1	$\Theta_{1\|11}$	$\Theta_{2\|11}$
1, 2	$\Theta_{1\|12}$	$\Theta_{2\|12}$
2, 1	$\Theta_{1\|21}$	$\Theta_{2\|21}$
2, 2	$\Theta_{1\|22}$	$\Theta_{2\|22}$

,

$\Theta_{i|jk}$ are conditional probabilities $\Theta_{i|jk} \geq 0$, $\forall i, j, k$, $\sum_{i=1}^{2} \Theta_{i|jk} = 1$, $\forall j, k$.

For the purpose of estimation it is useful to express the model in so called **product form**

$$f(y|\psi, \Theta) = \prod_{i \in y^*} \prod_{\varphi \in \psi^*} \Theta_{i|\varphi}^{\delta(i|\varphi, y|\psi)}, \tag{17.1}$$

where i is index, φ is multiindex (vector index), y^*, ψ^* domains of variables and $\delta(i|\varphi, y|\psi)$ is Dirac function, i.e. it is one for $i|\varphi = y|\psi$ and zero otherwise.

18 Dirichlet distribution

A suitable distribution of model parameters in the case when model is categorical, is the Dirichlet one.

$$f(\Theta|d(t)) = \frac{1}{B(\nu_t)} \prod_{i \in y^*} \prod_{\varphi \in \psi^*} \Theta_{i|\varphi}^{\nu_{i|\varphi;t}}, \tag{18.1}$$

Here

ν_t is the statistics (with the same structure as the model has) ,

$B(\nu)$ is a multivariate beta function

$$B(\nu)=\prod_{\varphi\in\psi^*}\frac{\prod_{i\in y^*}\Gamma\left(\nu_{i|\varphi}\right)}{\Gamma\left(\sum_{i\in y^*}\nu_{i|\varphi}\right)}, \tag{18.2}$$

where $\Gamma(\cdot)$ is gamma function defined by the formula

$$\Gamma(x)=\int_0^\infty t^{x-1}\exp(-t)\,dt, \tag{18.3}$$

for which it holds

$$\Gamma(x+1)=x\Gamma(x)\,,\ x\in R^+. \tag{18.4}$$

19 Normal distribution

We have normal regression model with regression vector ψ_t, regression coefficients θ and noise variance r, We denote $\Theta=\{\theta,r\}$. The model equation is

$$y_t=\psi_t^{'}\theta+e_t,\ \ e_t\sim N(0,r)\,.$$

The conditional pdf of the model is

$$f(y_t|\psi_t,\Theta)=\frac{1}{\sqrt{2\pi}}r^{-0.5}\exp\left\{-\frac{1}{2r}\left(y_t-\psi_t^{'}\theta\right)^2\right\}. \tag{19.1}$$

Expectation

$$E[y_t|\psi_t,\Theta]=\psi_t^{'}\theta,$$

variance

$$D[y_t|\psi_t,\Theta]=r.$$

For the purpose of estimation it is advantageous to modify the model in the following way:

- exponent is divided as follows

$$y_t-\psi_t^{'}\theta=-\left[-1\,\theta'\right]\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]=-\left[y_t\,\psi_t\right]\left[\begin{array}{c}-1\\ \theta\end{array}\right]$$

  (the sign minus is formal).

- the square in the exponent is written as row times column

$$\left(y_t-\psi_t^{'}\theta\right)^2=\left(y_t-\psi_t^{'}\theta\right)\left(y_t-\psi_t^{'}\theta\right)=$$
$$=\left[-1\,\theta'\right]\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]\left[y_t\,\psi_t\right]\left[\begin{array}{c}-1\\ \theta\end{array}\right]=\left[-1\,\theta'\right]D_t\left[\begin{array}{c}-1\\ \theta\end{array}\right],$$

  where $D_t=\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]\left[y_t\,\psi_t\right]$ is so called data matrix.

Model (19.1) now has the form

$$f(y_t|\psi_t,\Theta)=\frac{1}{\sqrt{2\pi}}r^{-0.5}\exp\left\{-\frac{1}{2r}\left[-1\,\theta'\right]D_t\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}. \tag{19.2}$$

20 Inverse Gauss-Wishart distribution

Its abbreviation is GiW

The distribution has the form

$$f\left(\Theta|d\left(t\right)\right)\propto r^{-0.5\kappa_t}\exp\left\{-\frac{1}{2r}\left[-1\,\theta'\right]V_t\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}, \tag{20.1}$$

where κ_t and V_t are statistics (κ_t is the counter, V_t is the information matrix).

Matrix V_t is symmetric and positive definite and for computation of parameter point estimates it can be decomposed to sub-matrices

$$V_t=\left[\begin{array}{cc}V_y & V_{y\psi}^{'}\\ V_{y\psi} & V_{\psi}\end{array}\right], \tag{20.2}$$

where (for y_t scalar) V_y is a number, $V_{y\psi}$ is a column vector and V_ψ is a rectangle matrix.

21 Point estimate with quadratic criterion

The optimal point estimates must minimize the posted criterion. Here it is quadratic one

$$\min_{\hat{\Theta}_t}E\left[\left(\Theta-\hat{\Theta}_t\right)^2|d\left(t\right)\right]. \tag{21.1}$$

We perform the square and than apply the expectation. Then we are going to use completion to square in $\hat{\Theta}$

$$\min_{\hat{\Theta}_t}E\left[\Theta^2-2\hat{\Theta}_t\Theta+\hat{\Theta}_t^2|d\left(t\right)\right]=$$

$$=\min_{\hat{\Theta}_t}\left\{E\left[\Theta^2|d\left(t\right)\right]-2\hat{\Theta}_tE\left[\Theta|d\left(t\right)\right]+\hat{\Theta}_t^2\right\}=*1*$$

$\hat{\Theta}_t$ is a deterministic number

$$*1*=\min_{\hat{\Theta}_t}\left\{E\left[\Theta^2|d\left(t\right)\right]-E\left[\Theta|d\left(t\right)\right]^2+E\left[\Theta|d\left(t\right)\right]^2-2\hat{\Theta}_tE\left[\Theta|d\left(t\right)\right]+\hat{\Theta}_t^2\right\}=*2*$$

we used the formula $D\left[\Theta\right]=E\left[\Theta^2\right]-E\left[\Theta\right]^2$ valid for the variance

$$*2*=\min_{\hat{\Theta}_t}\left\{D\left[\Theta|d\left(t\right)\right]+\left(\hat{\Theta}_t-E\left[\Theta|d\left(t\right)\right]\right)^2\right\}=D\left[\Theta|d\left(t\right)\right]$$

the minimum is

$$\hat{\Theta}_t=E\left[\Theta|d\left(t\right)\right]$$

as $D\left[\Theta|d\left(t\right)\right]$ is a constant with respect to $\hat{\Theta}_t$.

22 Point estimates of regression model parameters

MAP (Maximum Aposteriori Probability) estimation for normal regression model practically corresponds to minimization of quadratic criterion.

We look for maximum posterior pdf (which is a result of Bayesian estimation) (20.1)

$$f\left(\Theta|d\left(t\right)\right)\propto r^{-0.5\kappa}\exp\left\{-\frac{1}{2r}\left[-1\,\theta'\right]V\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}=$$

$$=r^{-0.5\kappa}\exp\left\{-\frac{1}{2r}\left(V_y-2\theta'V_{y\psi}+\theta'V_\psi\theta\right)\right\},$$

where we used the division of information vector V according to (20.2).

First we age going to estimate θ, i.e. to differentiate with respect to θ and lay the result equal to zero. It is a derivation of vectors according to vectors.

$$\frac{\partial f\left(\left\{\theta,r\right\}|d\left(t\right),r\right)}{\partial\theta}\propto r^{-0.5\kappa}\exp\left\{-\frac{1}{2r}\left[-1\,\theta'\right]V\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}\left(\frac{-1}{2r}\right)\left(-2V_{y\psi}+2V_\psi\theta\right)=0.$$

From it he get

$$\hat{\theta}=V_\psi^{-1}V_{y\psi}. \tag{22.1}$$

We substitute the result into the posterior pdf and we obtain

$$\Lambda=V_y-2\hat{\theta}'V_{y\psi}+\hat{\theta}'V_\psi\hat{\theta}=$$

$$=V_y-2V_{y\psi}^{'}V_\psi^{-1}V_{y\psi}+V_{y\psi}^{'}V_\psi^{-1}V_\psi V_\psi^{-1}V_{y\psi},$$

and

$$\Lambda=V_y-V_{y\psi}^{'}V_\psi^{-1}V_{y\psi}. \tag{22.2}$$

The posterior pdf with the optimal point estimate of regression coefficient is(22.1)

$$f\left(r|d\left(t\right)\right)\propto r^{-0.5\kappa}\exp\left\{-\frac{\Lambda}{2r}\right\}.$$

We differentiate it and lay equal to zero

$$-\kappa\frac{1}{2r}+\Lambda\frac{1}{r^2}=0,$$

From it we have

$$\hat{r}=\frac{\Lambda}{\kappa}. \tag{22.3}$$

$\hat{\theta}$ a $\hat{r}$ are point estimates which we are seeking for.

23 Point estimates of categorical model parameters

Here, the point estimates of parameters are given by a mere normalization of rows of or the statistics matrix ν_t

$$\hat{\Theta}_{y|\psi;t} = \frac{\nu_{y|\psi;t}}{\sum_{i\in y^*} \nu_{i|\psi;t}}, \ \forall y \in y^* \, \mathrm{a} \, \psi \in \psi^*. \tag{23.1}$$

The point estimate is an expectation of parameter with posterior pdf (18.1) - for lucidity we skip the time index t

$$\hat{\Theta}_{y|\psi} = E\left[\Theta_{y|\psi}|d\left(t\right)\right] = \int_0^\infty \Theta_{y|\psi} f\left(\Theta|d\left(t\right)\right) d\Theta =$$

$$= \frac{1}{B\left(\nu\right)} \int_0^\infty \Theta_{y|\psi} \prod_{i\in y^*} \prod_{\varphi\in\psi^*} \Theta_{i|\varphi}^{\nu_{i|\varphi}} d\Theta = *1*,$$

where beta function B is given in (18.2). Formally we express the model in a product form (17.1)

$$\Theta_{y|\psi} = \prod_{i\in y^*} \prod_{\varphi\in\psi^*} \Theta_{i|\varphi}^{\delta(i|\varphi,y|\psi)}$$

and substitute. We continue

$$*1* = \frac{1}{B\left(\nu_t\right)} \int_0^\infty \prod_{i\in y^*} \prod_{\varphi\in\psi^*} \Theta_{i|\varphi}^{\nu_{i|\varphi}+\delta(i|\varphi,y|\psi)} d\Theta =$$

$$= \frac{1}{\prod_{\varphi\in\psi^*} B\left(\nu_\varphi\right)} \prod_{\varphi\in\psi^*} \int_0^\infty \prod_{i\in y^*} \Theta_{i|\varphi}^{\nu_{i|\varphi}+\delta(i|\varphi,y|\psi)} d\Theta_{y|\psi} = *2*,$$

where

$B\left(\nu_\varphi\right) = \frac{\prod_{i\in y^*} \Gamma\left(\nu_{i|\varphi}\right)}{\Gamma\left(\sum_{i\in y^*} \nu_{i|\varphi}\right)}$ according to (18.2)

we use the assumption of independence between parameters from different components.

For individual components it holds

$$\int_0^\infty \prod_{i\in y^*} \Theta_{i|\varphi}^{\nu_{i|\varphi}+\delta(i|\varphi,y|\psi)} d\Theta_{y|\psi} = \begin{cases} B\left(\nu_\varphi\right) & \text{pro } \delta = 0, \\ B\left(\nu_\psi + 1\right) & \text{pro } \delta = 1. \end{cases}$$

The terms with $\delta = 0$ are canceled

$$*2* = \frac{B\left(\nu_\psi + \delta\left(i,y\right)\right)}{B\left(\nu_\psi\right)} = \frac{\frac{\prod_{i\in y^*} \Gamma\left(\nu_{i|\psi}+\delta(i,y)\right)}{\Gamma\left(\sum_{i\in y^*} \nu_{i|\psi}+1\right)}}{\frac{\prod_{i\in y^*} \Gamma\left(\nu_{i|\psi}\right)}{\Gamma\left(\sum_{i\in y^*} \nu_{i|\psi}\right)}} = *3*\,.$$

and again the terms for which $y \neq i$ are canceled, too, and we get

$$*3* = \frac{\frac{\Gamma\left(\nu_{y|\psi}+1\right)}{\Gamma\left(\sum_{i\in y^*}\nu_{i|\psi}+1\right)}}{\frac{\Gamma\left(\nu_{y|\psi}\right)}{\Gamma\left(\sum_{i\in y^*}\nu_{i|\psi}\right)}} = \frac{\frac{\nu_{y|\psi}}{\sum_{i\in y^*}\nu_{i|\psi}}\frac{\Gamma\left(\nu_{y|\psi}\right)}{\Gamma\left(\sum_{i\in y^*}\nu_{i|\psi}\right)}}{\frac{\Gamma\left(\nu_{y|\psi}\right)}{\Gamma\left(\sum_{i\in y^*}\nu_{i|\psi}\right)}} = \frac{\nu_{y|\psi}}{\sum_{i\in y^*}\nu_{i|\psi}}.$$

In the above derivation we also have used the properties of the gamma function (18.4).

This completes the proof of (23.1).

24 Logistic regression in details

Derivative of likelihood for logistic regression

Derivative of logarithm for likelihood $\ln L$ with the model(**??**) with respect to Θ is

$$\frac{\partial}{\partial\Theta}\ln L\left(\Theta\right) = \sum_{\tau=1}^{t}\left[y_\tau\psi_\tau - \frac{\exp\left(z_\tau\right)}{1+\exp\left(z_\tau\right)}\psi_\tau\right] = \sum_{\tau=1}^{t}\left(y_\tau - p_\tau\right)\psi_\tau,$$

where according to (**??**) $z_\tau = \psi_\tau\Theta$ a and so$dz_\tau/d\Theta = \psi_\tau$. Further we denote

$$p_\tau = \frac{\exp\left(z_\tau\right)}{1+\exp\left(z_\tau\right)} = P\left(y_t = 1|\psi_\tau,\Theta\right).$$

The second derivative$\ln L$ with respect to Θ is

$$\frac{\partial^2}{\partial\Theta^2}\ln L\left(\Theta\right) = \frac{\partial}{\partial\Theta}\sum_{\tau=1}^{t}\left(y_\tau - p_\tau\right)\psi_\tau = \sum_{\tau=1}^{t}\frac{\partial}{\partial\Theta}p_\tau\psi_\tau = \sum_{\tau=1}^{t}p_\tau\left(1-p_\tau\right)\psi_\tau^{'}\psi_\tau,$$

as

$$\frac{\partial}{\partial\Theta}p_\tau = \frac{\partial}{\partial\Theta}\frac{\exp\left(z_\tau\right)}{1+\exp\left(z_\tau\right)} = \frac{\exp\left(z_\tau\right)\psi_\tau^{'}\left(1+\exp\left(z_\tau\right)\right) - \exp\left(z_\tau\right)\exp\left(z_\tau\right)\psi_\tau^{'}}{\left(1+\exp\left(z_\tau\right)\right)^2} =$$

$$= \frac{\exp\left(z_\tau\right)\psi_\tau^{'}}{\left(1+\exp\left(z_\tau\right)\right)^2} = \left(\frac{\exp\left(z_\tau\right)}{1+\exp\left(z_\tau\right)}\frac{1}{1+\exp\left(z_\tau\right)}\right)\psi_\tau^{'} = p_\tau\left(1-p_\tau\right)\psi_\tau^{'}.$$

For numerical maximization it is advantageous to use Newton algorithm (both the derivatives are analytical).

Newton algorithm

Let us denote $g\left(x\right)$ the function to be minimized; here $x = \left[x_1, x_2\cdots x_n\right]'$. The gradient g' and Hess matrix g'' are

$$g'\left(x\right) = \begin{bmatrix} \frac{\partial g}{\partial x_1} \\ \frac{\partial g}{\partial x_2} \\ \cdots \\ \frac{\partial g}{\partial x_n} \end{bmatrix}$$

$$g''(x) = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_n} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} & \cdots & \frac{\partial^2 g}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 g}{\partial x_n \partial x_1} & \frac{\partial^2 g}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_n^2} \end{bmatrix}.$$

The algorithm starts at the point $x^{(0)}$ and generates further points $x^{(1)}$, $x^{(2)}$, $\cdots$ as follows: We take Taylor expansion of g at $x^{(i)}$ and use its first three terms (quadratic function)

$$g(x) \doteq g\left(x^{(i)}\right) + g'\left(x^{(i)}\right)\left(x - x^{(i)}\right) + \frac{1}{2} g''\left(x^{(i)}\right)\left(x - x^{(i)}\right)^2.$$

For the next point $x^{(i+1)}$ we minimize this quadratic function

$$g'\left(x^{(i)}\right) + g''\left(x^{(i)}\right)\left(x^{(i+1)} - x^{(i)}\right) = 0$$

from which we have

$$x^{(i+1)} = x^{(i)} - \frac{g'\left(x^{(i)}\right)}{g''\left(x^{(i)}\right)}.$$

We repeat it till the estimates stabilize.