

1 Lecture: Intro, models

Repetition of probability notions

Random variable

- discrete: finite number of realizations (coin, dice, level of service)
- continuous: infinite amount of realizations (car speed, intensity of traffic)

Probability function (discrete variable)

$$f(x) = P(X = x)$$

Density function (continuous variable)

$$P(X \in (a, b)) = \int_a^b f(x) dx$$

Examples

- discrete categorical rv

$$f(x_1, x_2)$$

$x_1 \backslash x_2$	1	2	$f(x_2)$
1	0.4	0.2	0.6
2	0.3	0.1	0.4
$f(x_1)$	0.7	0.3	= 1

- continuous rv

$$f(x) = ae^{-ax}, x \geq 0, a > 0$$

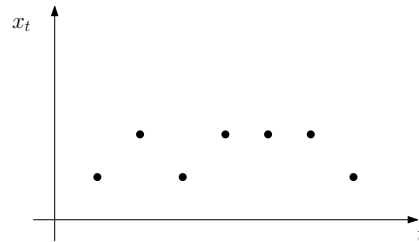
$$P(X > 1) = \int_1^{\infty} ae^{-ax} dx = a \left[\frac{-1}{a} e^{-ax} \right]_1^{\infty} = - [e^{-ax}]_1^{\infty} = e^{-a}$$

Random process

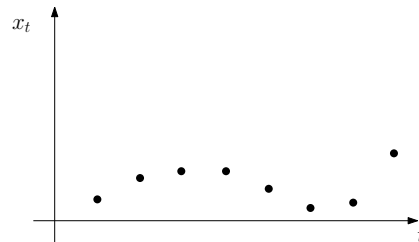
Random variable with time index x_t . Discrete time \rightarrow random sequence.

$$x_1, x_2, x_3, \dots$$

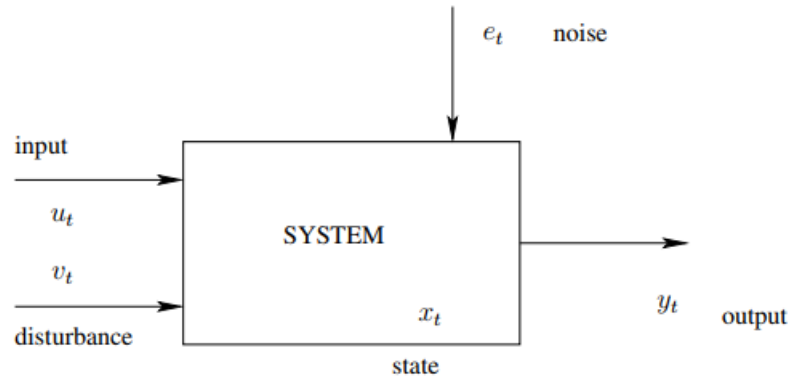
- discrete random process (with discrete time)



- continuous random process (with continuous time)



System and its variables



Bayesian models

Stochastic description of the output y_t in dependence on variables in regression vector ψ_t . The specific relation is given by the parameter Θ . It is in the form of conditional distribution

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

Discrete categorical model

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

Example for $\psi_t = [u_t, y_{t-1}]$, all binary

u_t	1	1	2	2	u_t	1	1	2	2
y_{t-1}	1	2	1	2	y_{t-1}	1	2	1	2
$y_t = 1$	$\Theta_{1 11}$	$\Theta_{1 12}$	$\Theta_{1 21}$	$\Theta_{1 22}$	$y_t = 1$	0.2	0.9	0.5	1
$y_t = 2$	$\Theta_{2 11}$	$\Theta_{2 12}$	$\Theta_{2 21}$	$\Theta_{2 22}$	$y_t = 2$	0.8	0.1	0.5	0

For $[u_t, y_{t-1}] = [1, 2]$ the prob. of $y_t = 1$ is 0.9 and $y_t = 2$ is 0.1.

Continuous regression model

$$\begin{aligned}y_t &= b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + \cdots + a_n y_{t-n} + b_n u_{t-n} + k + e_t = \\ &= \psi'_t \theta + e_t\end{aligned}$$

where **noise** $e \sim N(0, r)$, y_i is output, u_t is input and

$$\psi_t = [u_t, y_{t-1}, u_{t-1}, \cdots, y_{t-n}, u_{t-n}, 1]'$$

$$\theta = [b_0, a_1, b_1, \cdots, a_n, b_n, k]'; \Theta = \{\theta, r\}$$

n is **model order**.

If $a_1 = a_2 = \cdots = a_n = 0$ the model is **static**. Otherwise, it is **dynamic**.

Distribution

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

State-space model

The state model is

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

$$y_t = Ax_t + Bu_t + v_t$$

Transformation of 2nd order model regression model to state-space form

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

The state model is

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

$$y_t = [1, 0, 0, 0, 0] x_t$$

Programs

1. **T11simCont.sce**

simulation of the second order regression model

2. **T13simDisc.sce**

simulation of discrete model (controlled coin with memory)

$f(y_t(t) \mid u_t(t), y_t(t-1))$, $y_t, u_t=1,2$

3. **T15simState.sce**

simulation with regression model in a state-space form

2 Lecture: Estimation

Description of model parameters is given by parameter distribution

$$f(\Theta|d(t))$$

where $d_t = \{y_t, u_t\}$ and $d(t) = \{d_0, d_1, d_2, \dots, d_t\}$; $d_0 \equiv d(0)$ prior data.

Evolution of this distribution is based on the Bayes rule.

Bayes rule

$$f(A|B, C) = \frac{f(B|A, C) f(A|C)}{f(B|C)} \propto f(B|A, C) f(A|C)$$

where

A is what we estimate - parameter Θ

B is what we monitor - new data y_t, u_t

C is old data $d(t-1)$

$$\underbrace{f(\Theta|d(t))}_{\text{posterior pdf}} \propto f(y_t|\psi_t, \Theta) \underbrace{f(\Theta|d(t-1))}_{\text{prior pdf}}$$

under natural conditions of control

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

Comments

1. Recursion: prior \rightarrow posterior starts with the very prior $f(\Theta|d(0))$
2. The computations are recursive - the complexity of parameter distribution must not increase - conjugate distribution (Gauss-Wishart, Dirichlet)

3. Recursion on functions - unfeasible. For specific model (categorical, regression) the recursion can be converted to that on statistics, which gives algebraic recursion.
4. Batch estimation (for $t = 1, 2, \dots, N$)

$$f(\Theta|d(N)) \propto \underbrace{f(\Theta|d(0))}_{\text{prior pdf}} \underbrace{\prod_{t=1}^N f(y_t|\psi_t, \Theta)}_{\text{likelihood}} \quad (1)$$

5. Results of estimation

(a) posterior distribution $f(\Theta|d(t))$

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

Estimation of discrete model

Model (for binary $f(y_t|u_t, y_{t-1})$)

u_t	1	1	2	2
y_{t-1}	1	2	1	2
$y_t = 1$	0.2	0.9	0.5	1
$y_t = 2$	0.8	0.1	0.5	0

Statistics S_t

u_t	1	1	2	2
y_{t-1}	1	2	1	2
$y_t = 1$	$S_{1 11}$	$S_{1 12}$	$S_{1 21}$	$S_{1 22}$
$y_t = 2$	$S_{2 11}$	$S_{2 12}$	$S_{2 21}$	$S_{2 22}$

Update - for measured y_t, u_t, y_{t-1} recompute

$$S_{y_t|u_t, y_{t-1}; t} = S_{y_t|u_t, y_{t-1}; t-1} + 1$$

which means: the combination $[y_t, u_t, y_{t-1}]$ has been once more measured.

It is similar to the coin.

Program (est_catg.sce)

Estimation of regression model

Model

$$f(y_t|\psi_t\Theta) = \frac{1}{\sqrt{2\pi r}} \exp\left\{-\frac{1}{2r}(y_t - \psi_t'\theta)^2\right\} \propto$$
$$\propto r^{-0.5} \exp\left\{-\frac{1}{2}[-1, \theta'] \underbrace{\begin{bmatrix} y_t \\ \psi_t \end{bmatrix}}_{D_t \text{ data matrix}} \begin{bmatrix} -1 \\ \theta \end{bmatrix}\right\}$$

Statistics

$$V_t, \kappa_t$$

where V_t is a square positive definite matrix with the dimension of D_t , **information matrix** and κ_t is a scalar **counter** of data samples.

Statistics update

$$V_t = V_{t-1} + D_t$$

$$\kappa_t = \kappa_{t-1} + 1$$

Point estimates

$$\hat{\theta} = (V_{\psi})^{-1} V_{y\psi}$$

$$\hat{r} = \frac{V_y - \hat{\theta} V_{y\psi}}{\kappa}$$

where $V_y = V(1, 1)$, $V_{y\psi} = V(2 : \text{end}, 1)$, $V_{\psi} = V(2 : \text{end}, 2 : \text{end})$.

Program (est_regr.sce)

Batch estimation

According to (1), the estimation can be performed in an off-line mode for the whole measured dataset at once.

Example with

$$y_t = b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + k + e_t$$

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

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

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

Program (est_regrBatch.sce)

```
// Batch estimation of 2ne order regression model
```

```

// -----
clc, clear, close, mode(0)

nd=200;                                // number of data
r=.1; b0=1, a1=.3, b1=-.6, a2=.3, b2=.1, k=1 // parameters
y=zeros(1,nd);                          // output
u=rand(1,nd,'n');                        // input
// simulation
for t=3:nd
    y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+a2*y(t-2)+b2*u(t-2)+k+sqrt(r)*rand(1,1,'n');
end
// estimation

Y=y(3:$)';
X=[u(3:$)' y(2:$-1)' u(2:$-1)' y(1:$-2)' u(1:$-2)' ones(nd-2,1)];

th=inv(X'*X)*X'*Y;                       // point estimates
b0E=th(1), a1E=th(2), b1E=th(3), a2E=th(4), b2E=th(5), kE=th(6)

```


Prior information

Example (coin)

$x = 1, 1, 2, 1, 2, 2, \dots$

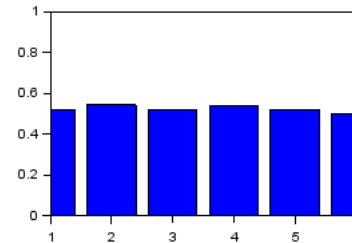
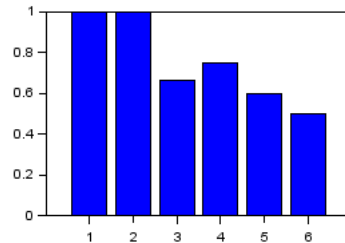
1. $S = [0, 0]$

x	1	1	2	1	2	2
S	[1, 0]	[2, 0]	[2, 1]	[3, 1]	[3, 2]	[3, 3]
θ	[1, 0]	[1, 0]	$[\frac{2}{3}, \frac{1}{3}]$	$[\frac{3}{4}, \frac{1}{4}]$	$[\frac{3}{5}, \frac{2}{5}]$	$[\frac{1}{2}, \frac{1}{2}]$

2. $S = [10, 10]$

x	1	1	2	1	2	2
S	[11, 10]	[12, 10]	[12, 11]	[13, 11]	[13, 12]	[13, 13]
θ	$[\frac{11}{21}, \frac{10}{21}]$	$[\frac{12}{22}, \frac{10}{22}]$	$[\frac{12}{23}, \frac{11}{23}]$	$[\frac{13}{24}, \frac{11}{24}]$	$[\frac{13}{25}, \frac{12}{25}]$	$[\frac{1}{2}, \frac{1}{2}]$

Comparison of estimation without and with prior information



Program (est_init.sce) - try various setting.0

Generally to initialization

Let us have a statistics: $S_t = S_{t-1} + y_t$ (sum) and $\kappa_t = \kappa_{t-1} + 1$ (count). Let the estimate is $\hat{\theta}_t = S_t / \kappa_t$. Let our prior knowledge is $\hat{\theta}_0 = \theta_0$. Then we set:

$$\kappa_0 = N, S_0 = \kappa_0 \theta_0,$$

where N expresses the strength of the prior information.

Then: $\hat{\theta} = (\kappa_0 \theta_0) / \kappa_0 = \theta_0$ and the prior information is obtained as if from N data records. This is why the several first measured records cannot change it so easy.

and for static regression model

To introduce θ_0 , we set

$$\kappa_0 = N, V_0 = \kappa_0$$

1	θ'_0		
θ_0	1	0	0
	0	1	0
	0	0	1

Program (est_init2.sce)

Programs

1. **T21estCont_LS.sce**

estimation of 2nd order regression model
– least squares estimation (off-line)

2. **T22estCont_B.sce**

estimation of 2nd order regression model
– Bayesian on-line estimation with statistic update

3. **T22estCont_B2.sce**

estimation of 2nd order regression model
– the model for simulation differs from that for estimation
-- Bayesian on-line estimation with statistic update

4. **T22estCont_B3.sce**

– like the previous one but model order **ord** can be set

5. **T22estCont_B4.sce**

estimation of 2nd order regression model
Estimation with REAL DATA (intensities of traffic in Strahov tunnel)

6. T23estDisc.sce

estimation of discrete model $f(y(t)|u(t),y(t-1))$ with y,u from $\{0,1\}$

3 Lecture: Prediction

Estimation of the value of future output.

– predictive pdf

$$f(y_{t+k}|y(t-1)), k = 0, 1, 2, \dots$$

– point prediction

$$\hat{y}_{t+k} = E[y_{t+k}|y(t-1)] = \int_{y^*} y_{t+k} f(y_{t+k}|y(t-1)) dy_{t+k}$$

Case 1 $k = 0$ - output estimation

We are at time t , y_t is not measured, yet and we estimate it on the base of past data.

- model with known parameters

$$f(y_t|y(t-1)) = \text{model}$$
$$\hat{y}_t = \int_{y^*} y_t f(y_t|y(t-1)) dy_t$$

- model with unknown parameters

$$f(y_t|y(t-1)) = \int_{\Theta^*} f(y_t, \Theta|y(t-1)) d\Theta =$$
$$= \int_{\Theta^*} \underbrace{f(y_t|y(t-1), \Theta)}_{\text{model}} \underbrace{f(\Theta|y(t-1))}_{\text{parameter estimate}} d\Theta$$

Case 2 $k > 0$ - time prediction (for $k = 1$)

$$\begin{aligned} f(y_{t+1}|y(t-1)) &= \int_{y^*} \int_{\Theta^*} f(y_{t+1}, y_t, \Theta|y(t-1)) d\Theta dy_t = \\ &= \int_{y^*} \int_{\Theta^*} f(y_{t+1}|y(t), \Theta) f(y_t|y(t-1), \Theta) f(\Theta|y(t-1)) d\Theta dy_t = \end{aligned}$$

Point prediction of Θ : $f(\Theta|y(t-1)) = \delta(\Theta, \hat{\Theta}_{t-1})$

$$= \int_{y^*} f(y_{t+1}|y(t), \hat{\Theta}_{t-1}) f(y_t|y(t-1), \hat{\Theta}_{t-1}) dy_t =$$

and for y_t : $f(y_t|y(t-1), \hat{\Theta}_{t-1}) = \delta(y_t, \hat{y}_t)$

$$= f(y_{t+1} | [\hat{y}_t, y(t-1)], \hat{\Theta}_{t-1})$$

It holds

$$\int \delta(x, a) f(x) dx = f(a)$$

Point prediction with regression model

The 1st order regression model $y_t = a_1 y_{t-1} + a_2 y_{t-2} + b u_t + e_t$ with known parameters a_1, a_2, b .

We are at time t and know all u_t , and $y(t-1)$.

The prediction is expectation and unknown values are replaced by their predictions (expectations)

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + b u_t + e_t$$

$$\hat{y}_t = a_1 y_{t-1} + a_2 y_{t-2} + b u_t$$

$$\hat{y}_{t+1} = a_1 \hat{y}_t + a_2 y_{t-1} + b u_{t+1}$$

$$\hat{y}_{t+2} = a_1 \hat{y}_{t+1} + a_2 \hat{y}_t + b u_{t+2}$$

Full prediction under condition of normality

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

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

→

$$E[y_{t+2}|y(t-1)] = a^3y_{t-1} + a^2bu_t + abu_{t+1} + bu_{t+2}$$

$$D[y_{t+2}|y(t-1)] = D[a^2e_t + ae_{t+1} + e_{t+2}] = (a^4 + a^2 + 1)r$$

Predictive pdf

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

Programs

1. **T31preCont.sce**
np-step prediction with continuous model (known parameters)
2. **T32preCont_Adapt.sce**
n-step prediction with continuous model (with estimation)
3. **T32preCont_Adapt2.sce**
n-step prediction with continuous model (with estimation)
– the model for simulation differs from that for estimation
4. **T32preCont_Adapt3.sce**
np-step prediction with continuous model (with estimation)
– real data (intensity) from Strahov tunnel are used
5. **T33preCat_Off.sce**
prediction with discrete model (off-line), known parameters
6. **T34preCat_OffEst.sce**
prediction with discrete model (off-line), unknown parameters
7. **T35preCat_OnEst.sce**
prediction with discrete model (on-line)

4 Lecture: State-space model

Model

$f(x_t|x_{t-1}, u_t)$ model of the state

$f(y_t|x_t, u_t)$ model of the output

is generated by the equations

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

$$y_t = Ax_t + Bu_t + v_t$$

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

Estimation

State description

$$f(x_{t-1}|d(t-1)) \xrightarrow{\text{prediction}} f(x_t|d(t-1)) \xrightarrow{\text{filtration}} f(x_t|d(t))$$

Evolution

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

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

Kalman filter

For normal model and normal prior

Notation

$$f(x_t|x_{t-1}, u_t) = N_{x_t}(Mx_{t-1} + Nu_t, r_w)$$

$$f(y_t|x_t, u_t) = N_{y_t}(Ax_t + Bu_t, r_v)$$

and

$$f(x_{t-1}|d(t-1)) = N_{x_{t-1}}(x_{t-1|t-1}, R_{t-1|t-1})$$

$$f(x_t|d(t-1)) = N_{x_t}(x_{t|t}, R_{t|t})$$

$$f(x_t|d(t)) = N_{x_t}(x_{t|t}, R_{t|t})$$

Substitution into the evolution equations gives Kalman filter (KF)

Kalman filter

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

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

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

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

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

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

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

Nonlinear model

$$x_t = g(x_{t-1}, u_t) + w_t$$

$$y_t = h(x_t, u_t) + v_t$$

EXAMPLE

For

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

the model is

$$x_{1;t} = \exp\{-x_{1;t-1} - x_{2;t-1}\} + u_t + w_t$$

$$x_{2;t} = x_{1;t-1} - 0.3u_t + w_{2;t}$$

$$y_t = x_{2;t} + v_t$$

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

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

$$h(x, u_t) \doteq h(\hat{x}_t, u_t) + h'(\hat{x}_t, u_t)(x - \hat{x}_t)$$

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

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

After substitution the linearization into the model, we have (for $x = x_{t-1}$ in the case of the state

equation and $x = x_t$ 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

$$\begin{aligned}\bar{M} &= g'(\hat{x}_{t-1}, u_t), & F &= g(\hat{x}_{t-1}, u_t) - \underbrace{g'(\hat{x}_{t-1}, u_t)}_{\bar{M}} \hat{x}_{t-1}, \\ \bar{A} &= h'(\hat{x}_t, u_t), & G &= h(\hat{x}_t, u_t) - \underbrace{h'(\hat{x}_t, u_t)}_{\bar{A}} \hat{x}_t.\end{aligned}$$

EXAMPLE (continuation) - ... only first equation is nonlinear

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

$$g_2(x, u_t) = x_1 - 0.3u_t$$

$$g'_1(x, u_t) = \left[\frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_2} \right] = [-\exp\{-x_1 - x_2\}, -\exp\{-x_1 - x_2\}]$$

$$g'_2(x, u_t) = \left[\frac{\partial g_2}{\partial x_1}, \frac{\partial g_2}{\partial x_2} \right] = [1, 0]$$

$$\bar{M} = \begin{bmatrix} -\exp\{-x_1 - x_2\}, & -\exp\{-x_1 - x_2\} \\ 1 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} \exp\{-x_1 - x_2\} + u_t \\ x_1 - 0.3u_t \end{bmatrix} - \bar{M}x_{t-1}$$

The output equation is linear with $\bar{A} = [0, 1]$

Fully linearized model is

$$x_t = \bar{M}x_{t-1} + F + w_t$$

$$y_t = \bar{A}x_t + v_t$$

With

$$N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad G = 0, \quad B = 0.$$

we can use subroutine Kalman

$$[x_t, R_x, y_t] = \text{Kalman}(x_t, y_t, u_t, \bar{M}, N, F, \bar{A}, B, G, R_w, R_v, R_x)$$

Programs

1. **T46statEst_KF.sce**
state estimation (Kalman filter)
2. **T47statEst_Noise.sce**
Kalman as a noise filter
3. **T48statEst_NL.sce**
nonlinear model estimation (T48statEst_L.sce - linear version)
4. **T48statEst_Par.sce**
unknown parameters

5 Lecture: Control

Minimum variance control - in each step t minimizes $E[y^2]$.

Model (e.g. first order)

$$y_t = b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + k + e_t$$

$$E[y_t^2] = (b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + k)^2 + r \rightarrow \min$$

$$\rightarrow b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + k = 0$$

$$u_t = -\frac{1}{b_0} (a_1 y_{t-1} + b_1 u_{t-1} + k)$$

Often unstable !!!

Derivation of optimal control

Model

$$y_t = \psi_t' \theta + e_t$$

Criterion

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

where $J_t = y_t^2 + \omega u_t^2$.

Bellman equations

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

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

for $t = N, N-1, N-2, \dots, 1$.

Control with regression model

Regression model in state-space form (2nd order)

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

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

The penalty can be written as

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

where Ω is a diagonal matrix

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

Bellman equations, where we guess the form of $\varphi_{t+1}^* = x_t' R_{t+1} x_t$

$$\begin{aligned}
 E \left[x_t' R_{t+1} x_t + x_t' \Omega x_t | u_t, d(t-1) \right] &= E \left[x_t' U x_t | u_t, d(t-1) \right] = \\
 &= (M x_{t-1} + N u_t)' U (M x_{t-1} + N u_t) + \rho = \\
 &= x_{t-1}' \underbrace{M' U M}_C x_{t-1} + 2 u_t' \underbrace{N' U M}_B x_{t-1} + u_t' \underbrace{N' U N}_A u_t + \rho = \\
 &= u_t' A u_t + 2 u_t' A \underbrace{A^{-1} B}_{S_t} x_{t-1} + x_{t-1}' S_t' A S_t x_{t-1} + \\
 &\quad + \underbrace{x_{t-1}' C x_{t-1} - x_{t-1}' S_t' A S_t x_{t-1}}_{x_{t-1}' R_t x_{t-1}} + \rho = \\
 &= (u_t + S_t x_{t-1})' A (u_t + S_t x_{t-1}) + x_{t-1}' R_t x_{t-1} + \rho
 \end{aligned}$$

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

Recursion

Optimization

$$R_{N+1} = 0$$

for $t = N, N - 1, \dots, 1$

$$U = R_{t+1} + \Omega$$

$$A = N'UN$$

$$B = N'UM$$

$$C = M'UM$$

$$S_t = A^{-1}B$$

$$R_t = C - S_t'QS_t$$

end

Application

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

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

$$y_t \cdots \text{funct}(u_t)$$

end

Extended criterion

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

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

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.

$$\Omega = \begin{bmatrix} 1 & & & & & -s_t \\ & \omega + \lambda & -\lambda & & & \\ & & 0 & & & \\ & -\lambda & \lambda & & & \\ & & & \dots & & \\ & & & & 0 & \\ -s_t & & & & & s_t^2 \end{bmatrix}$$

with $x_t = [y_t, u_t, y_{t-1}, u_{t-1}, \dots, 1]$ the expression $x_t' \Omega x_t$ gives the extended criterion.

Control with categorical model

Model

$$f(y_t|u_t, y_{t-1}) = \Theta_{y_t|u_t, y_{t-1}}$$

$$J = J_{y_t|u_t, y_{t-1}}$$

model (Θ)					penalty (J)						
u_t		1	1	2	2	u_t		1	1	2	2
y_{t-1}		1	2	1	2	y_{t-1}		1	2	1	2
$y_t = 1$		0.7	0.2	0.9	0.4	$y_t = 1$		0	1	1	2
$y_t = 2$		0.3	0.8	0.1	0.6	$y_t = 2$		1	2	2	3

where each state is penalized individually. (Above - we do not want big values)

Direct use Bellman equations. Only manipulation is a bit awkward.

Programs

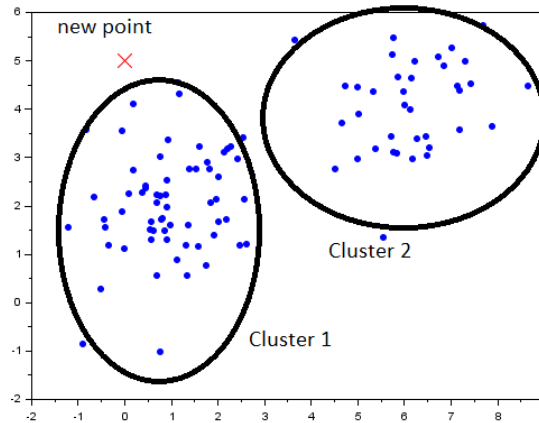
1. **T50ctrlMinVar.sce**
minimum variance control
2. **T52ctrlDisc.sce**
control with categorical model
3. **T53ctrlX.sce**
control with regression model
4. **T54ctrlXEst.sce**
adaptive control with regression model

6 Lecture: Model based classification I

Clustering: detecting groups (classes) of similar objects creating clusters.

Classification: assigning a new object to one of the existing classes.

Example



Two normal clusters with expectations $[1, 2]$ and $[6, 4]$.

Red cross is a new measurement. It evidently belongs to Cluster 1.

Generating multimodal data

- components = models of individual clusters $f_j(x_t|\theta_j)$, $j = 1, 2, \dots, \nu$
- pointer = discrete random process c_t whose values point at the active component

Each cluster has its own model - component.

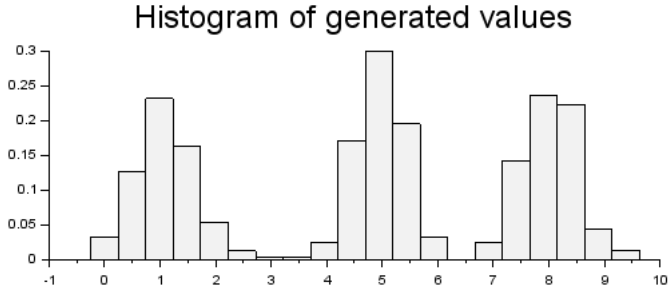
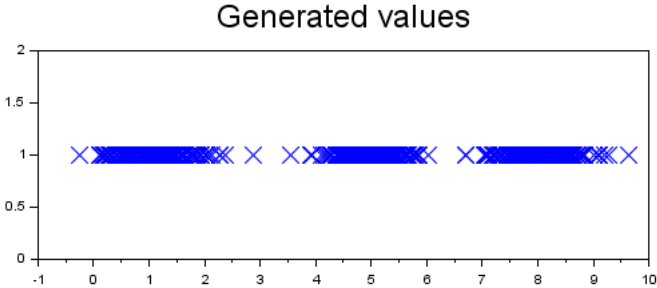
Example

```
// Simulation of a mixture with regression components
// -----
clc, clear, close, mode(0)

nd=500; // number of steps
th=[1 5 8]; // component expectations
sd=[1 1 1]*.5; // component standard deviations
al=[.3 .4 .3]; // switching probabilities
for t=1:nd
    c(t)=sum( cumsum(al)<rand(1,1,'u') )+1;
    x(t)=th(c(t))+sd(c(t))*rand(1,1,'n');
end
```

```
// results
scf();
subplot(211)
plot(x,ones(x),'x','markersize',10)
title('Generated values','fontsize',5)
subplot(212)
histplot(20,x);
title('Histogram of generated values','fontsize',5)
```

with the result



Classification 1 - known components

Given components $f(x|c=i)$, $i=1, 2, \dots, \nu$, switching probabilities $f(c)$, $i=1, 2, \dots, \nu$ and one data record $x = \xi$, estimate the most probable value of c .

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

Example (for $\nu = 3$)

Components

$$f(x|c=1) = N_x(1, 0.5)$$

$$f(x|c=2) = N_x(5, 0.5)$$

$$f(x|c=3) = N_x(8, 0.5)$$

Model of switching

$$f(c) = \alpha_c \begin{array}{c|ccc} c & 1 & 2 & 3 \\ \hline \alpha & 0.3 & 0.4 & 0.3 \end{array}$$

Measurement

$$x = \xi = 2.1$$

Classification – weights

$$w_1 \propto f(c = 1|\xi) \propto f(x = 2.1|c = 1) f(c = 1) = N_x(1, 0.5)|_{x=2.1} \alpha_1 = 0.168 \cdot 0.3 = 0.05$$

$$w_2 \propto 0.00013 \cdot 0.4 = 0.00005$$

$$w_3 \propto 4 \cdot 10^{-16} \cdot 0.3 \doteq 0$$

– normalization

$$w = [0.999, 0.001, 0]$$

⋯ and we classify to the first class.

Classification 2 - known pointer for learning

= Learning with a teacher

Component and pointer models are unknown, values of the pointer are known for learning. → At each step of estimation we update only the component indicated by the pointer.

In practice:

Learning

We divide the data sample x_1, x_2, \dots, x_N into groups C_c with respect to the pointer values $c = c_1, c_2, \dots, c_N$ and learn the parameters for all components individually.

Testing

Runs as in the previous case.

Classification 3 - EM-like algorithm

The expectation-maximization (EM) algorithm is an approach for performing maximum likelihood estimation in the presence of unknown (pointer) variables.

It starts with prior component parameters. Then it repeats the following two steps:

- 1. determine the values for the pointer variables,*
- 2. estimate the component parameters,*

until steady state is reached.

Using the introduced theory, the procedure is like this:

1. Take a dataset $X = [x_1, x_2, \dots, x_N]$ for estimation
2. Set initial components $f(x|c)$ and their stationary probabilities α_c , $c = 1, 2, \dots, n_c$
3. Determine weights $w = f(c|X) \propto \alpha_c f(x_t|\theta_c)$ and pointer estimate

For $t = 1 : N$

$$w_1 = f(c = 1|X) \propto \alpha_1 f(x_t|\theta_1)$$

$$w_2 = f(c = 2|X) \propto \alpha_2 f(x_t|\theta_2)$$

...

$$w_{n_c} = f(c = n_c|X) \propto \alpha_{n_c} f(x_t|\theta_{n_c})$$

$$c_t = \arg \max(w_1, w_2, \dots, w_{n_c})$$

4. Recompute component parameters θ and switching probabilities α

For $j = 1 : n_c$ do

- (a) select subset of dataset whose records correspond to pointer value j
- (b) use this subset for estimation of parameters of the j -th component $f_j(x|\theta_j)$
for normal components - average and variance
- (c) switching probabilities α are relative frequencies of the pointer values

5. If the pointer changes go to 3

Remark: It uses learning with a teacher.

Classification 4 - mixture estimation

Neither model parameters nor pointer values are known. Classification is to be performed with on-line measured data. The procedure is as follows:

1. For each data record x_t determine the weights with respect to currently estimated components $w_j = f(c_t = j|x_t)$.
2. Data record is added to the statistics with its weight $S_{j;t} = S_{j;t-1} + w_j$, $\kappa_{j;t} = \kappa_{j;t-1} + w_j$ and point estimates $\hat{\theta}_t$ are computed in a standard way for each component (pointer model can be skipped).

Algorithm

Initial setting: Set initial parameters of components (θ, r) and corresponding statistics S, κ .

for $t = 1 : nd$

1. measure data record x_t
2. determine weights w
for $j = 1 : n_c$
 - (a) $q_j = f(x_t|\theta_j)$ - proximity

(b) $w_j = \aleph(q_j \alpha_j)$ - where \aleph means normalization to sum equal to 1

end

3. recompute statistics and parameters (e.g. for static normal components)

for $j = 1 : n_c$

(a) $S_{j;t} = S_{j;t-1} + w_j x_t$

(b) $\kappa_{j;t} = \kappa_{j;t-1} + w_j$

(c) $\gamma_{j;t} = \gamma_{j;t-1} = w_j$

(d) $\theta_j = \frac{S_{j;t}}{\kappa_{j;t}}$

(e) $\alpha_j = \aleph(\gamma)$

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 $f(c|x)$

(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.

Programs

1. **T61classKn.sce**
classification with known models of components
2. **T62classUnKn.sce**
classification with unknown models of components
3. **T63EM_C.sce**
iterative estimation of pointer and components (like EM algorithm)
4. **T64MixReg.sce**
Bayesian mixture estimation

7 Lecture: Model based classification II

Naive Bayes

Estimation of multivariate model can be considerably simplified by the assumption of conditional independence of explanatory variables.

Conditional independence

$$f(x_1, x_2, \dots, x_n | c) = \prod_{i=1}^n f(x_i | c)$$

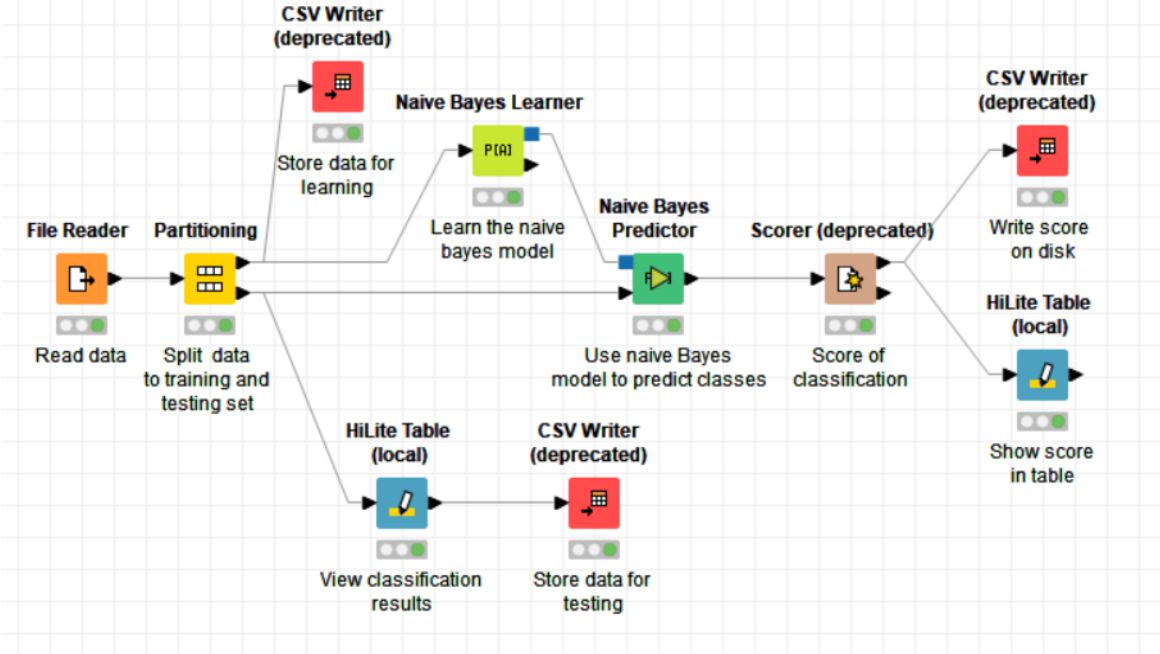
Principle of naive Bayes

$$\begin{aligned} f(c|x) &\propto f(x|c) f(c) = f(x_1, x_2, \dots, x_n | c) f(c) \\ &= f(c) \prod_{j=1}^{n_x} f(x_j | c) \end{aligned}$$

!! USES ONLY MODELS OF SINGLE VARIABLE !!

KNIME: Task00_NaiveBayes

Naive Bayes
Naive Bayes learner and predictor to classify shuttle data.



Logistic regression

Used for discrete target and continuous explanatory variables.

Starts with Bernoulli model

$$f(c_t|p) = p^{c_t} (1 - p)^{1-c_t}, \quad c_t = 0, 1$$

$$p = P(c_t = 1).$$

Expectation $E[c_t] = p$ is extended by regression $b'x_t = b_0 + b_1x_{1;t} + \dots, b_{m;t}x_m$

To ensure borders of $p \in (0, 1)$ we model $\text{logit}(p) = \ln \frac{p}{1-p}$

$$\text{logit}(p) = b'x_t$$

from which the **model** is

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

Usage

$$z_t = b'x_t \in (-\infty, \infty)$$

for estimated b and measured x_t compute z_t

$$p = P(c_t = 1) = \frac{\exp(z_t)}{1 + \exp(z_t)}$$

for $p > 0.5$ set $c_t = 1$ else $c_t = 0$

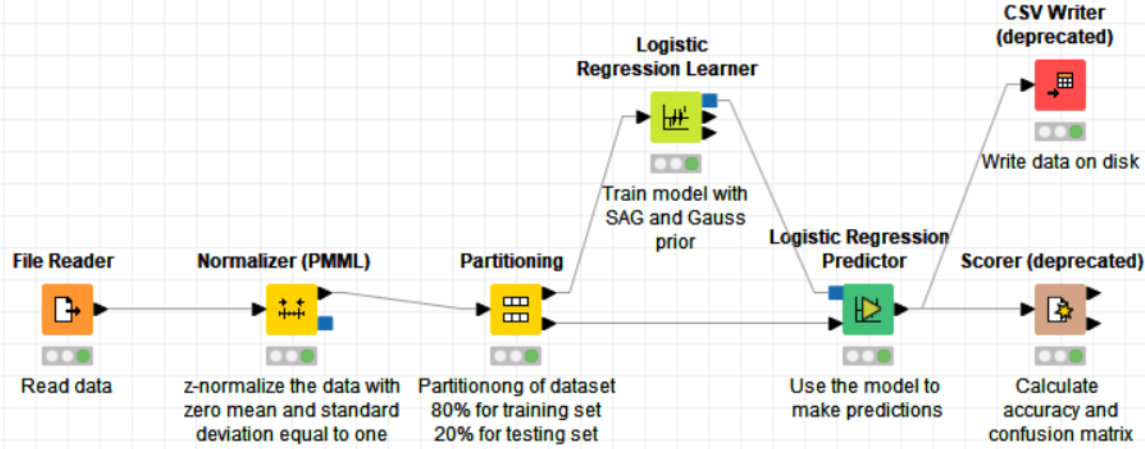
Estimation by ML

$$L_N(b) = \prod_{t=1}^N \frac{\exp\{c_t x_t b\}}{1 + \exp\{x_t b\}} \rightarrow \ln L_N(b) = \sum_{t=1}^N [c_t x_t b - \ln(1 + \exp\{x_t b\})]$$

and maximize numerically.

KNIME: Task01_Logistic_Regression

Logistic Regression
Example to building a basic prediction / classification model using logistic regression.



Poisson regression

Starts with Poisson model

$$f(c_t|\lambda) = \exp\{-\lambda\} \frac{\lambda^{c_t}}{c_t!}, c_t = 0, 1, 2, \dots$$

$\lambda \geq 0$ is intensity of occurring events.

To ensure nonnegativity of λ , we extend $\ln(\lambda) = b'x_t = b_0 + b_1x_{1;t} + \dots + b_mx_{m;t}$

$$\rightarrow \lambda = \exp(b'x_t)$$

Model in logarithm

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

Estimation by LN

Log-likelihood is

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

and it is maximized numerically.

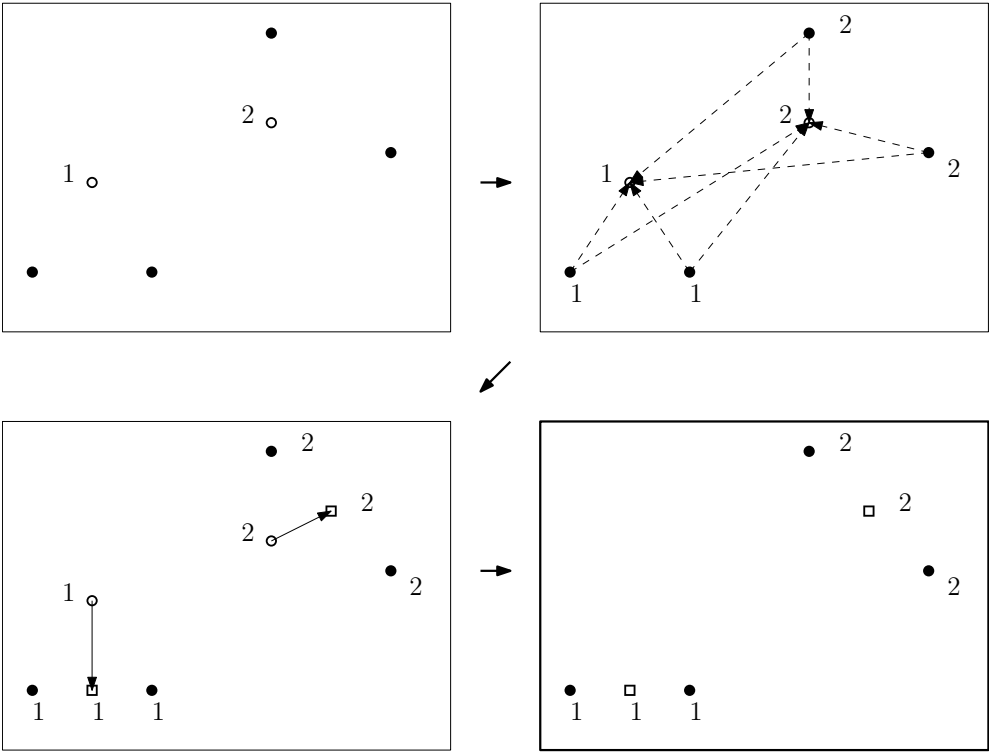
8 Lecture: Clustering

We have multimodal data x and want to capture density clusters.

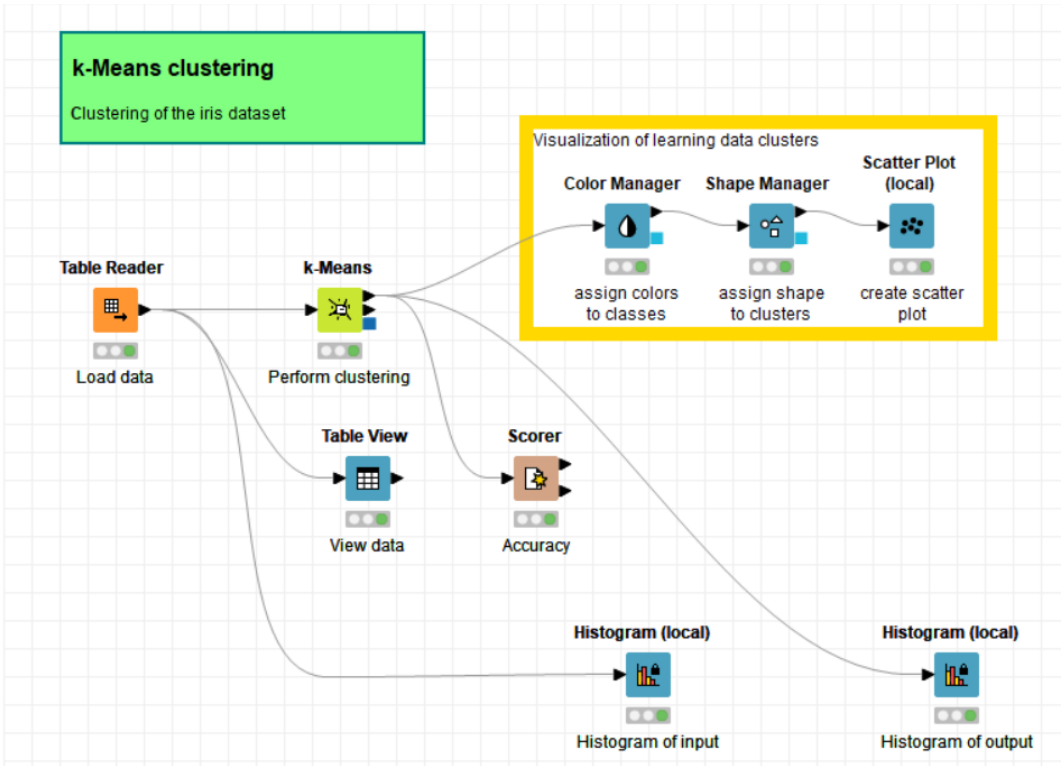
K-means clustering

0. Set n initial cluster centers (n fixed)
1. To each data point x_i assign the nearest center.
The assigned points to a center form the cluster.
2. For each cluster compute its centroid (point average)
3. Shift the centers to the centroids
4. Repeat from 1 if changes occur

Example



KNIME: Task02_k-Means_Clustering



K-medoids clustering

Similar to k -means.

0. Determine md as the desired number of clusters. Randomly select md data points as initial centers of medoids.
0. To each medoid find the points that are closest to it. They will be initial clusters.
0. Determine overall distance of points from their medians.
1. Randomly select one medoid and one non-medoid (data point that is not a medoid).
2. Swap them and again determine overall distance of points from their medians.
3. If the distance is smaller, continue by 1. If not, algorithm ends.

KNIME: Task03_k-Medoids_Clustering

k-Medoids Clustering

Clustering of the iris dataset



Fuzzy clustering (*c*-means)

In the *c*-means algorithm we minimize criterion

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

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)

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

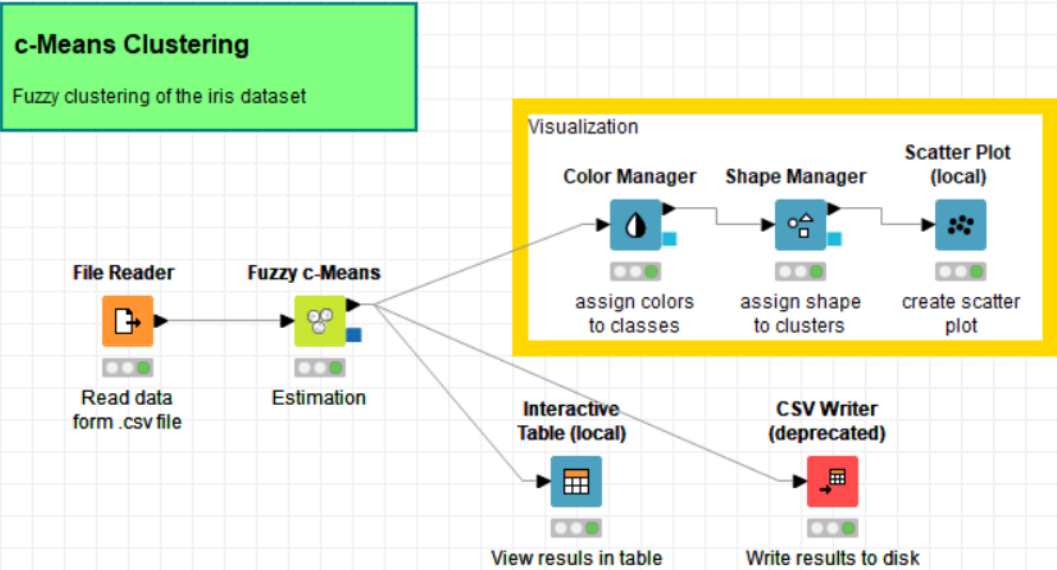
- weights (are given as membership functions)

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \quad (3)$$

Algorithm

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

KNIME: Task04_c-Means_Clustering



Density based clustering (dbscan)

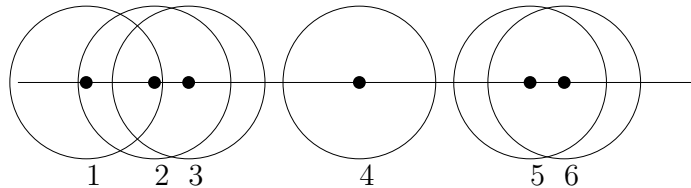
We have a set of data $X = \{x_1, x_2, \dots, x_N\}$, where $x_i \in R^m$

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 and y exists, if both these points are accessible from some inner point.



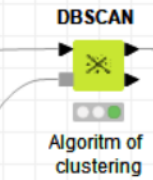
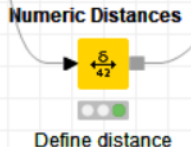
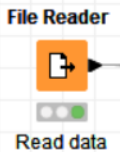
Algorithm of clustering

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 “clus” 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.

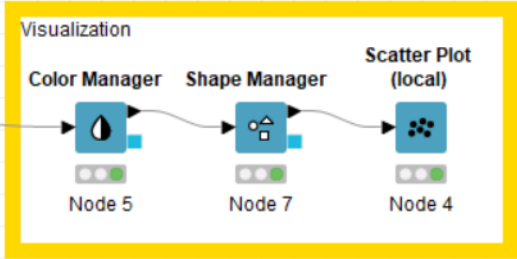
Clusters are formed by points that are connected.

KNIME: Task05_Density_Clustering

DBSCAN
Density based clustering of data



check Summary Table



We have 2 variables x and y and targ - pointing at classes
DBSCAN creates Class - estimate of targ.
The result can be seen in DBSCAN/Summary Table

Visualization: show x and y and for color (shape) choose
- either targ (to see the input data)
- or Class (to see the result of clustering)

The variable in Select one column defines the color (shape)

Hierarchical clustering (agglomerative)

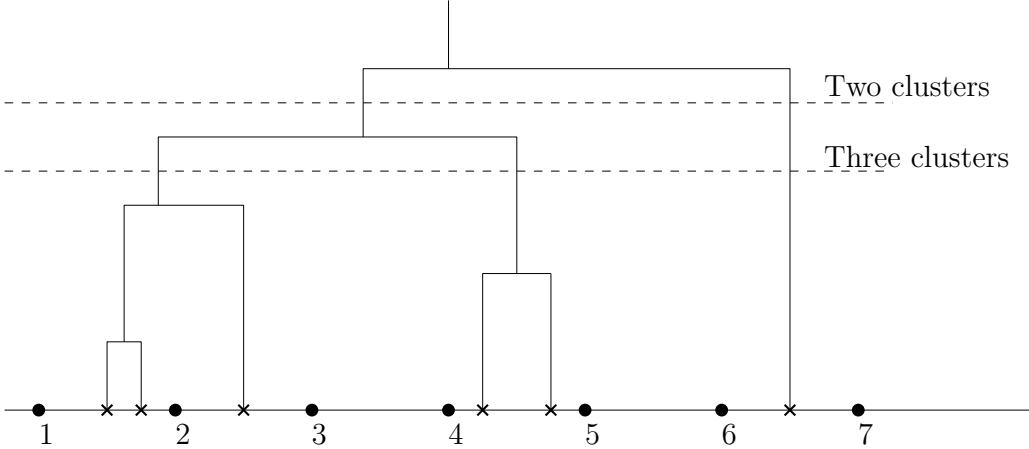
1. All data points are denoted as clusters on the level 1 (with only one point).
2. Find two nearest clusters and join them together in a new point. Its level is equal to the number of points in joined in this new point.
3. The coordinates of the cluster lie on a connecting line of the coordinates of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

For more information and the divisive version of the algorithm see the textbook.

Example

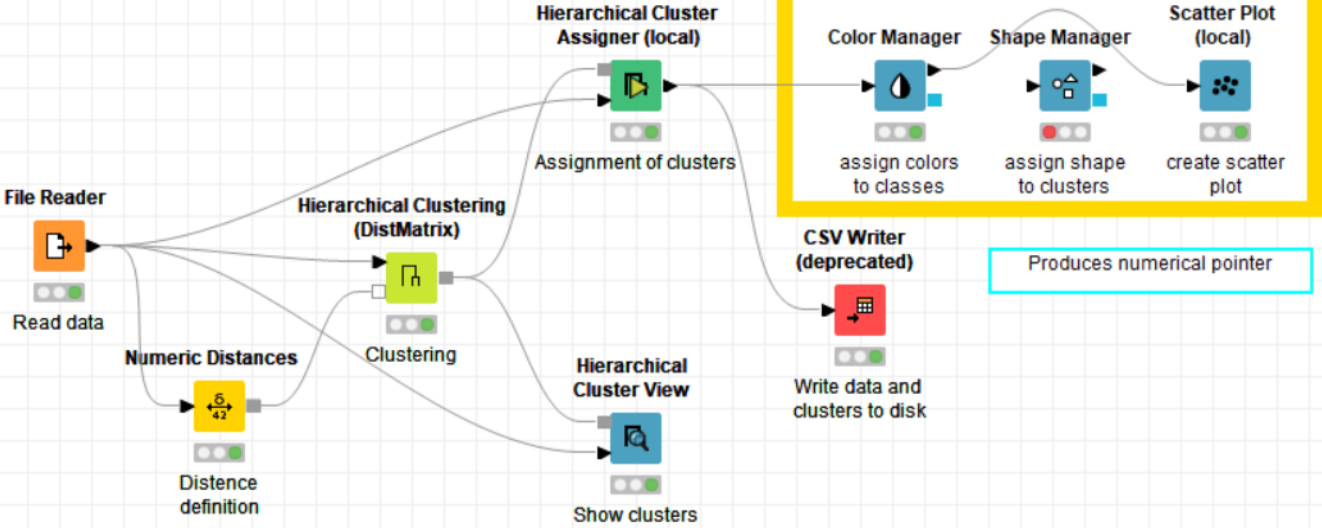
The data are $x = [1.4, 1.8, 2.5, 4.2, 4.7, 6.5]$.

Construct dendrogram.



KNIME: Task06_Hierarchical_Clustering

Hierarchical clustering
Construction of decision tree and using it for clustering.



9 Lecture: Classification

K-nearest neighbour

We have data $X = \{x_i\}_{i=1}^N$ with detected clusters. The task is: assign a newly measured point y to some cluster.

Algorithm

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


KNIME: Task07_k-NearNeighb

K nearest neighbour
Application of the k nearest neighbour principle for classification.


File Reader
Read data from .csv file



Partitioning
Divide to learn and test parts



K Nearest Neighbor
Classification




CSV Writer
Write the results to .csv file




Use of new CSV Writer

Visualization of resulting clusters


Color Manager
assign colors to classes



Shape Manager
assign shape to clusters



Scatter Plot (local)
create scatter plot



Decision trees

We have discrete data records $x_t = [x_1, x_2, \dots, x_n]_t$, $t = 1, 2, \dots, N$ and a pointer variable $c_t \in \{1, 2, \dots, m\}$ which assigns the data records x_t to one of m classes.

Example

Let us have the following data

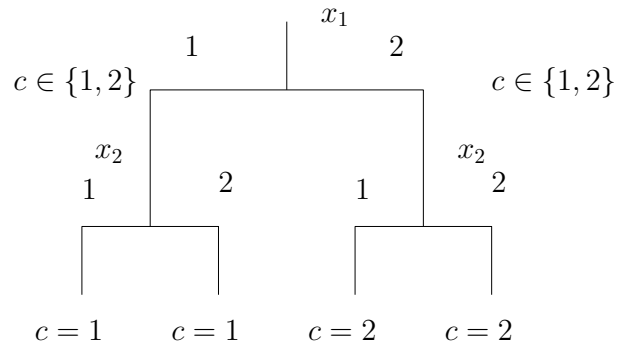
t	x_1	x_2	c
1	1	1	1
2	1	2	1
3	2	1	2
4	2	2	2

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

We chose the root cluster as x_1 with values $\{1, 2\}$. Then,

- if $x_1 = 1$ then $x_2 \in \{1, 2\}$
 - if $x_1 = 1$ and $x_2 = 1$ then $c = 1$

- if $x_1 = 1$ and $x_2 = 2$ then $c = 1$
- if $x_1 = 2$ then $x_2 \in \{1, 2\}$
 - if $x_1 = 2$ and $x_2 = 1$ then $c = 2$
 - if $x_1 = 2$ and $x_2 = 2$ then $c = 2$

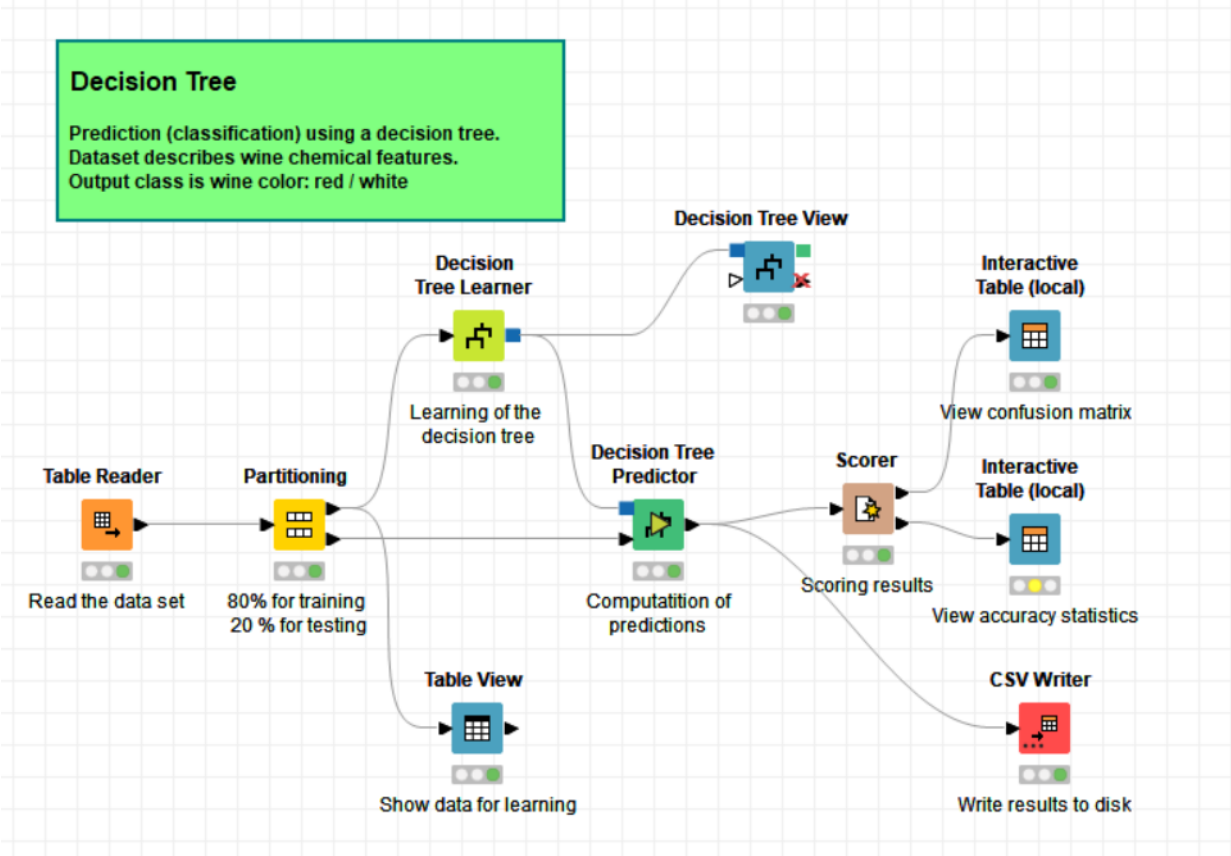


Now, we measure $x_t = [1, 2]$. Using the tree, we classify it to $c_t = 1$

Problem: What order of the variables in the tree is the best one.

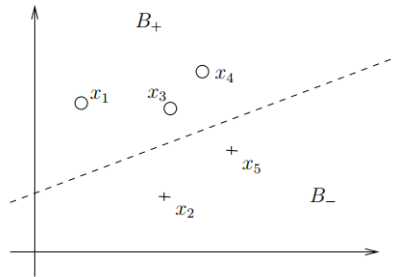
KNIME: Task08_Decision_Tree

Decision Tree
Prediction (classification) using a decision tree.
Dataset describes wine chemical features.
Output class is wine color: red / white



Support vector machines

We have a sequence of data points $x_i, i = 1, 2, \dots, n$. Some of them have the attribute $+$ and the rest $-$. We are to separate them so that the distance of the line (hyperplane) from the $+$ points and $-$ points would be maximal.



Let us denote the separating line as $y = \alpha y + \beta = 0$. Then, we look for maximal δ such that two parallel lines $y = \alpha y + \beta + \delta = 0$ and $y = \alpha y + \beta - \delta = 0$ also separate the points - i.e. the points are separated by a strip of the width 2δ .

The task leads to numerical optimization of nonlinear function.

KNIME: Task09_Support_Vec_Mach

