Simulace s regresním modelem

Úloha: Simulace dat pro další použití.

- jednorozměrný vstup i výstup
- obecný řád modelu

Simulují se data z normálního, jednorozměrného regresního modelu

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + b_0 u_t + b_1 u_{t-1} + \cdots + b_n u_{t-n} + k + e_t$$

kde y, u, e jsou výstup, vstup a porucha, a, b, k jsou parametry.

Předpoklady: $e \sim N(0, r)$, r konstantní; vstup je generován předem.

Poznámka

Např. pro testování nových algoritmů je simulace dat velice důležitá. Data je totiž možno simulovat podle vlastních představ tak, aby byly otestovány všechny možné varianty použití nového algoritmu.

Závěrečné testování je ale dobré provést na reálných datech, aby algoritmus byl co nejlépe připraven k použití v reálném prostředí.

Značení

- y - yt,
- u - ut,
- a, b, k - a, b, k, (použití viz níže),
- r - cv (použití viz níže).

Volitelné parametry

- nd - počet simulovaných dat
- Sim.Cy.ord - řád modelu
- I_typU - volba řízení
- Sim.Cy.tha1
  Sim.Cy.thb1
  Sim.Cy.thk
  Sim.Cy.cv - parametry modelu
  (značení: Sim - simulace, Cy - komponenta pro spojitý výstup)

Doporučené experimenty

1. Měňte parametry soustavy tak, aby byla (i) pomalá, (2) rychlá.
   - Rychlostí se myslí délka odezvy na jednotkový vstup.
   - Tato rychlost je dána velikostí vlastních čísel charakteristické rovnice.
   - U prvního řádu je to velmi jednoduché: pro tha blízké nule dostaneme soustavu rychlou, pro tha blízké jedné je soustava pomalá.
   - Abychom rychlost soustavy mohli pozorovat, volíme vstup roven jedné a hodně malý rozptyl šumu.
2. Zkuste měnit řád modelu. Pozor: předvolby parametrů jsou nastaveny jen do řádu 5. Potom je potřeba další koeficienty dodat. Nicméně, viditelné rozdíly jsou patrné tak do řádu 3.
3. Zajímavé je poexperimentovat s řádem 2. Nastavit soustavu s reálnými kořeny, jedním dvojnásobným kořenem a s komplexními kořeny. Vlastnosti jsou dány řešením charakteristické rovnice.

Program

```
// Simulation of scalar regression model of order n
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion
deff('ps=genpsi(t,n,y,u)','ps=[y(t-(1:n)),u(t-(0:n)) 1]','c')

nd=100;                                  // number of data
I_typU=1;                                // type of input

// model
Sim.Cy.ord=2;                            // model order
Sim.Cy.tha=[.6 -.2 .1 -.5 .2];           // parameters at y
Sim.Cy.thb=[1 .5 -.3 -.1 .1 -.3];        // parameters at u
Sim.Cy.thk=-1;                           // model constant
Sim.Cy.cv=.01;                            // variance of noise

// model of input signal
select I_typU                            // choice of input
  case 1, ut=rand(1,nd,'u');             // random, uniform
  case 2, ut=ones(1,nd);                  // ones
  case 3                              // two steps
    ut=[ones(1,fix(nd/2)), -ones(1,fix(nd/2)+1)];
  case 4                                 // random jumps
    ut(1)=1; j=1;
    for i=2:nd
      if rand(1,1,'u')>.85, j=rand(1,1,'n'); end
      ut=[ut j];
```

```
    end
  case 5, ut=abs(sin(10*(1:nd)/nd));    // sinus
end
yt=zeros(1,nd);
ord=Sim.Cy.ord;
cv=Sim.Cy.cv;
th=[Sim.Cy.tha(1:ord) Sim.Cy.thb(1:(ord+1)) Sim.Cy.thk];
                                        // regression coefficients
Sim.Cy.th=th;

// time loop
for t=(ord+1):nd                        // time loop
  ps=genpsi(t,ord,yt,ut);               // regression vector
  yt(t)=ps*th'+cv*rand(1,1,'n');        // model simulation
end
Sim.Cy.yt=yt;
Sim.Cy.ut=ut;

// results
s=1:nd;
plot(s,yt(s),s,ut(s))
legend('output','input');
title('Simulation with regression model')
set(gca(),'data_bounds',[1 nd min([yt,ut])-.1 max([yt,ut])+.1])

save _data/dataT11.dat Sim
```

Simulace s vícerozměrným regresním modelem

Úloha: Simulace dat pro další použití, zacházení s vícerozměrnými daty.

- dvourozměrný vstup i výstup (lze měnit)
- první řád modelu (lze měnit)

Simulují se data z normálního, dvourozměrného regresního modelu

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

kde y, u, e jsou výstup, vstup a porucha, a, b, k jsou (maticové) parametry.

Předpoklady: $e \sim N(0, r)$, r konstantní; vstup je generován předem.

Poznámka

Obecně - y_t má dimenzi ny *a u_t má dimenzi* nu*. Pak a_i jsou matice* ny×ny*, b_i matice* ny×nu*, k a e_t jsou sloupcové vektory o dimenzi* ny*.*

Regresní vektor je

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

je to sloupcový vektor s dimenzí nf=(n+1)*nu+n*ny+1 *a*

vektor parametrů je

$$\theta = [b_0, a_1, b_1 \cdots a_n, b_n, k]$$

což je matice s dimenzí ny×nf*.*

Pozor: *Ve vícerozměrném případě má součin regresního vektoru a parametru tvar*

$$\theta\psi_t$$

a výsledkem je sloupec s dimenzí ny

Značení

- y - yt,
- u - ut,
- a, b, k - a, b, k, (použití viz níže),
- r - cv (použití viz níže).

Volitelné parametry

- nd - počet simulovaných dat
- Sim.Cy.ord - řád modelu
- I_typU - volba řízení
- Sim.Cy.thb0
  Sim.Cy.tha1
  Sim.Cy.thb1
  Sim.Cy.thk - parametry modelu
  (značení: Sim - simulace, Cy - komponenta pro spojitý výstup)

Doporučené experimenty

1. Měňte parametry simulované soustavy a sledujte vstup a výstup soustavy na grafech.
2. Měňte počet vstupů a výstupů soustavy.
   Pozor: parametry soustavy jsou nastaveny pro dva vstupy a dva výstupy. Pokud chcete tyto rozměry měnit, musíte dodat do programu vlastní parametry. Ty pak určují počet vstupů a výstupů. Parametry u mnohorozměrné soustavy jsou matice.

Program

```
// Simulation of a two-dimensional first order regression model
// ------------------------------------------------
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=100;                             // length of data
I_typU=2;                           // type of control (see below)

// model for simulation
Sim.Cy.ord=1;                       // model order
Sim.Cy.thb0=[1 .2; -.5 1];          // parameters at ut
Sim.Cy.tha1=[.4 .2; .1 .5];         // parameters at yt
Sim.Cy.thb1=[.1 -2; -.2 .1];
Sim.Cy.thk=[0; 0];                  // constant (model absolute term)
Sim.Cy.th=[Sim.Cy.thb0 Sim.Cy.tha1 Sim.Cy.thb1 Sim.Cy.thk];
Sim.Cy.cv=.1;                       // noise variance

[my mu]=size(Sim.Cy.thb0);
yt=zeros(my,nd);
yt(1)=1; yt(2)=3;                   // initial conditions for output

// definition of the control variable
select I_typU
```

```
case 1, ut=rand(mu,nd);          // random
case 2
  z=1;
  for i=1:mu
    ut(i,1:nd)=z*ones(1,nd);
    z=-z;
  end
end

// TIME LOOP OF THE SIMULATION
ord=Sim.Cy.ord;                    // model order
sd=sqrt(Sim.Cy.cv);
th=Sim.Cy.th;                      // regression coefficients
for t=3:nd
  ps=genps(ord,t,yt,ut);           // regression vector
  yt(:,t)=th*ps+sd*eye(my)*rand(my,1); // simulation
end
Sim.Cy.yt=yt;
Sim.Cy.ut=ut;

save _data/dataT11N.dat Sim         // save data

// RESULTS OF THE SIMULATION
set(scf(1),'position',[600 100 600 600])
subplot(211),plot(ut','.'),title('Input')
subplot(212),plot(yt'),title('Output')
```

Simulace s kategorickým modelem

Úloha: Simulace dat pro další použití.

- dvouhodnotové veličiny
- model s pamětí a řízením

Simulují se data z kategorického modelu, definovaného tabulkou

$[u_t,\, z_{t-1}]$	$z_t = 1$	$z_t = 2$
1, 1	0.9	0.1
1, 2	0.8	0.2
2, 1	0.3	0.7
2, 2	0.1	0.9

kde z, u jsou výstup a vstup, parametry jsou pravděpodobnosti, tedy nezáporná čísla se součtem jedna v každém řádku.

Předpoklady: parametry jsou konstantní; vstup je generován předem.

Poznámka

1. *Neurčitost modelu je dána volbou parametrů. Jsou-li jejich hodnoty blízko 0 a 1, má v sobě model málo neurčitosti. Pro parametry blízké 0.5 je model velmi neurčitý.*

2. *Pro generování diskrétního výstupu podle pravděpodobnostní tabulky* th *použijeme následující postup*

   Funkce j=psi2row(⋯) dává řádek j tabulky parametrů určený aktuálním řízením a minulým výstupem

   Příkaz

   $$zt(t)=\text{sum}(\text{rand}(1,1,'u')>\text{cumsum}(\text{th}(j,:)))+1;$$

   funguje takto (např. pro řádek 1 a generované náhodné číslo 0.941)

   $$\text{rand}(1,1,'u')>\text{cumsum}(\text{th}(j,:))$$

   $$0.941>[.9,\ 1] = [0,\ 1]$$

   dále

   $$\text{sum}([0,\ 1])+1=1+1=2$$

   Hodnota výstupu bude z(t)=2.

Značení

- z - yt,
- u - ut,
- θ - th

Volitelné parametry

- nd - počet simulovaných dat
- I_typU - volba řízení
- Sim.Cz.th - parametry modelu
  (značení: Sim - simulace, Cz - komponenta pro diskrétní výstup)

Doporučené experimenty

1. Zvolte parametry simulované soustavy tak, aby
   (a) v soustavě nebyla žádná náhoda - například tak, aby výstup neustále přeskakoval z jedničky na dvojku.
   (b) v soustavě bylo jen málo neurčitosti - výstup se chová tak, jak určuje vstup a minulý výstup a jen občas nastane porucha.
   (c) v soustavě bylo hodně neurčitosti - vliv vstupu a minulého výstupu se prakticky ztrácí
2. Měňte typ vstupu a určete, kdy jsou nejlépe viditelné vlastnosti simulovaného systému podle zadaných parametrů modelu.

Program

```
// Simulation of categorical model f(y(t)|u(t),y(t-1)) with y,u=1,2
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=50;                                   // number of data
I_typU=1;

Sim.Cz.th=[.9 .1                         // model parameters
           .8 .2
           .3 .7
           .1 .9];

select I_typU                            // choice of input
case 1 then
```

```
    ut=fix(2*rand(1,nd,'u'))+1;         // random
  case 2 then                           // two steps
    ut=[ones(1,fix(nd/2)), 2*ones(1,fix(nd/2))];
  case 3                                // regular 1 2 1 2 1 2 ...
    ut=ones(1,nd); ut(2:2:nd)=2;
end
zt=ones(1,nd);

// time loop
th=Sim.Cz.th;
for t=2:nd                              // time loop
  j=psi2row([ut(t),zt(t-1)],[2,2]);     // row in parameter table
  zt(t)=sum(rand(1,1,'u')>cumsum(th(j,:)))+1;
                                        // generation of y
end
Sim.Cz.zt=zt;
Sim.Cz.ut=ut;

// results
s=1:nd;
plot(s,zt(s),'.:',s,ut(s),'o')
legend('output','input');
title('Simulated data')
set(gca(),'data_bounds',[1 nd .9 2.1])

save _data/dataT12.dat Sim
```

Odhad s regresním modelem

Úloha: Odhad parametrů modelu pro další použití.

- jednorozměrný vstup i výstup
- obecný řád modelu

Jako data se využívá výsledku simulace ze souboru T11SimReg1 který se ukládá jako datový soubor dataT11.dat v adresáři _data a odkud je natažen. Nic ale nebrání tomu, vyrobit si vlastní data - buď přímo v souboru T11SimReg1 nebo si data uložit na disk a natáhnout si je tak, jak je to připravené.

Vlastní odhad se provádí postupným přepočtem statistik V a κ v časové smyčce a teprve po jejím ukončená jsou spočteny bodové odhady parametrů modelu ve tvaru

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + b_0 u_t + b_1 u_{t-1} + \cdots + b_n u_{t-n} + k + e_t$$

kde y, u, e jsou výstup, vstup a porucha, a, b, k jsou parametry.

Předpoklady: $e \sim N(0, r)$, r konstantní; vstup je generován předem.

Poznámka

Teorie bayesovského odhadování je založena na úplném popisu parametrů pomocí aposteriorní hustoty pravděpodobnosti. Využití této informace v podobě celé hustoty pravděpodobnosti je ale značně komplikované. Z této hustoty (tedy přímo z napočítaných statistik odhadu) je možno spočítat bodové odhady parametrů. Jejich využití je jednoduché a velmi často postačující.

Značení

1. y - yt
2. u - ut,
3. a, b, k - a,b,k (parametry)
4. r - cv (rozptyl šumu)
5. thE - odhadnuté parametry
6. ordE - řád odhadovaného modelu

Volitelné parametry

- I_dataSim - data: simulovaná, reálná,
- !!! ostatní parametry se přebírají ze simulace.

Doporučené experimenty

1. Jeden ze základních předpokladů dobrého odhadu jsou data, která skutečně vypovídají o vlastnostech soustavy.
   Zkuste měnit počet dat, vstupující do odhadu a porovnejte kvalitu odhadu pomocí připravené jednokrokové predikce (simulace z odhadnutého modelu).

2. Do odhadu je možno zavést tzv. exponenciální zapomínání. To způsobí, že vliv starších měřených dat se potlačuje a tím se zdůrazní význam aktuálních dat. To má význam, jestliže se vlastnosti soustavy pomalu mění nebo jestliže v minulosti byla změřena některá data chybová. Historie se pomalu zapomíná. Faktor zapomínání φ se volí blízký jedné (0.99, 0.995, 0.999) se do rovnic pro přepočet statistik zavede takto

$$\begin{aligned} V_t &= \varphi V_{t-1} + \Psi\Psi' \\ \kappa_t &= \varphi\kappa_{t-1} + 1 \end{aligned}$$

   Zkuste použít různí hodnoty zapomínání při odhadu a porovnejte kvalitu odhadu v jednotlivých případech.

3. Zkuste odhadovat s modelem různého řadu a sledujte vliv na kvalitu odhadu.

Program

```
// Estimation of scalar regression model of order n
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

deff('ps=genpsi(t,n,y,u)','ps=[y(t-(1:n)),u(t-(0:n)) 1]','c')

I_dataSim=2;                        // get data: 1=simulated, 2=real
ordE=2;                             // model order (valid for real data)

select I_dataSim
case 1
  load _data/dataT11.dat Sim        // load reults of task T11
  yt=Sim.Cy.yt;                     // or use your own data yt, ut
  ut=Sim.Cy.ut;                     //  and define order of the model
  ord=Sim.Cy.ord;
case 2
  load _data/dataReg1.dat yt ut     // from file _data/dataReg1.dat
end

nd=length(yt);
if I_dataSim==1,
  ordE=Sim.Cy.ord;
  th=Sim.Cy.th;
end
```

```
nPsi=2*ordE+3;                      // length of regression vector

V=1e-8*eye(nPsi,nPsi);              // comput. of information matrix
for t=(ordE+1):nd
  Ps=[yt(t) genpsi(t,ordE,yt,ut)]';
  V=V+Ps*Ps';
end
thE=inv(V(2:$,2:$))*V(2:$,1);       // point estimates of parameters
Est.Cy.ord=ordE;
Est.Cy.V=V;
Est.Cy.th=thE;

// results
if I_dataSim==1
  disp('Simulated parmeters')
  disp(th)
end
disp('Estimated parmeters')
disp(thE')

if I_dataSim==1
  if length(th)==length(thE)
    bar([th' thE])
    title('Simulated and estimated parameters')
  end
else
  bar(thE)
  title('Estimated parameters')
end

save _data/dataT21.dat Est
```

Odhad s kategorickým modelem

Úloha: Odhad parametrů modelu pro jeho další použití.

- vstup i výstup jsou binární (mají hodnoty 1 a 2)
- model s pamětí a řízením $f\left(z_t|u_t, z_{t-1}\right)$

Jako data se využívá buď simulace z datového souboru dataT12.dat (generovaného souborem T12SimCat) nebo reálných dat dataCat.dat - diskretizované hodnoty plyn a moment motoru. Reálná rada se generují souborem generDatCat.sce, kde lze získat podrobnější informace o datech. Oba datové soubory jsou uloženy v adresáři _data. Nic ale nebrání tomu, vyrobit si vlastní data - buď přímo v souboru T12SimCat nebo si data uložit na disk a natáhnout si je tak, jak je to připravené.

Vlastní odhad se provádí postupným přepočtem statistiky V (matice 4x2), na začátku nulová nebo s apriorními statistikami. Přepočet je

j=psi2row([u(t),z(t-1)],[2, 2]) - číslo řádku tabulky

V(j,z(t))=V(j,z(t))+1 - přepočet aktuálního prvku

Předpoklady: diskrétní vstup i výstup, matice V má obecně počet řádků roven součinu počtu hodnot veličin z regresního vektoru a počet sloupců roven počtu hodnot výstupu.

Poznámka

Apriorní hodnoty statistiky lze určit předběžným stanovením poměru koeficientů v jednotlivých řádcích statistiky, sílu počáteční informace zadáme absolutní velikosti čísel v apriorní statistice.

Značení

1. z - zt (diskrétní výstup)
2. u - ut,
3. θ - th
4. thE - odhadnuté parametry

Volitelné parametry

- I_dataSim - data: simulovaná, reálná,
- (ostatní parametry se přebírají ze simulace)

Doporučené experimenty

1. Testujte simulovaná data pro různé simulované soustavy. Zkuste porovnat odhad soustavy s hodně/málo neurčitosti.
2. Z reálných dat jsou k dispozici ještě "spotřeba" a "rychlost". Zkoušejte odhadovat model pro různé kombinace těchto veličin v roli výstupu a vstupu. Porovnejte výsledky.

Program

```
// Estimation of categorical model f(y(t)|u(t),y(t-1)) with y,u=1,2
// --- on-line update of statistics, off-line point estimates
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

I_dataSim=1;                             // get data: 1=simulated (in T12SimCat.sce)
                                         //           2=real (from file dataCat.dat)
if I_dataSim==1
  load _data/dataT12.dat Sim             // load of simulated data
  zt=Sim.Cz.zt;
  ut=Sim.Cz.ut;
  th=Sim.Cz.th;
else
  load _data/dataCat.dat sp co ga mo
  zt=mo;                                 // volba diskrétních
  ut=ga;                                 //   veličin (tady mo, ga)
end

nd=length(zt);                           // number of data
V=zeros(4,2);                            // statistics

for t=2:nd                               // time loop
  j=psi2row([ut(t),zt(t-1)],[2,2]);      // row in parameter table
  V(j,zt(t))=V(j,zt(t))+1;               // generation of y
end
thE=fnorm(V,2);                          // point estimates
Est.Cz.V=V;
Est.Cz.th=thE;

// results
if I_dataSim==1
  bar([th(:,1) thE(:,1)])
  title('Simulated and estimated parameters (first column of the model table)')
  legend('simulated','estimated');
else
  bar(thE(:,1))
  title('Estimated parameters - real data (first column)')
end

save _data/dataT22.dat Est
```

Předpověď výstupu s regresním modelem

Úloha: Předpověď výstupu soustavy jako informace o soustavě nebo pro další použití.

- jednorozměrný vstup i výstup
- obecný řád modelu
- obecný počet kroků predikce

Jako data se využívá výsledku simulace ze souboru T12SimCat1 který se ukládá jako datový soubor dataT11.dat v adresáři _data a odkud je natažen. Nic ale nebrání tomu, vyrobit si vlastní data - buď přímo v souboru T11SimReg1 nebo si data uložit na disk a natáhnout si je tak, jak je to připravené. Odhad parametrů modelu se provádí v souboru T21EstReg1, a to v batchové (jednorázové) formě. Pro tyto bodové odhady parametrů se provádí predikce na celém časovém intervalu.

Použitý model má tvar

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + b_0 u_t + b_1 u_{t-1} + \cdots + b_n u_{t-n} + k + e_t$$

kde y, u, e jsou výstup, vstup a porucha, a, b, k jsou bodové odhady parametrů.

Predikce se provádí postupným dosazováním dat nebo predikcí z minulých kroků do modelu. Tedy např. pro 2 řád modelu a predikci na 2 kroky platí

$$\begin{aligned} \hat{y}_t &= a_1 y_{t-1} + a_2 y_{t-2} \\ \hat{y}_{t+1} &= a_1 \hat{y}_t + a_2 y_{y-1} \\ \hat{y}_{t+2} &= a_1 \hat{y}_{t+1} + a_2 \hat{y}_{t+2} \end{aligned}$$

kde stříška označuje bodovou predikci výstupu.

Předpoklady: $e \sim N(0, r)$, r konstantní; vstup je generován předem.

Poznámka

Právě v jednoduchosti uvažované predikce se nejlépe ukazuje, jak je výhodné použití bodových odhadů parametrů modelu soustavy. Použití celé aposteriorní hustoty pravděpodobnosti by vedlo na tak složitý algoritmus, že bychom se nevyhnuli pracnému numerickému řešení.

Značení

- y - yt,
- u - ut,
- a, b, k - a, b, k,
- r - cv (parametry ze simulace),
- thE - odhadnuté parametry,
- ordE - řád odhadovaného modelu,

Volitelné parametry

- np - počet kroků predikce

Doporučené experimenty

1. Měňte délku predikce a kontrolujte její kvalitu v grafu.
2. Přidejte další kriterium kvality, kterým je průměrná kvadratická chyba predikce. Chybu predikce určíte jako rozdíl mezi hodnotami výstupu a jejich predikcí

   $$ep_i = y_i - yp_i, \;\; i = 1, 2, \cdots, nd$$

   Kriterium potom je

   $$SE = \frac{1}{nd} \sum_{i=1}^{nd} ep_i^2$$

   Průměr chyby predikce by měl být blízko nuly, kvadratická chyba SE by měla být co nejmenší.
3. Měňte parametry simulované soustavy (zejména rozptyl šumu) a sledujte kvalitu predikce. Soustavu lze měnit v simulaci, tj. ve souboru T11SimReg1.sce.
4. Podstatný vliv na kvalitu odhadu a tedy i predikce má volba vstupního signálu. Jestliže vstup je příliš hladký bez skoků, soustava není pro odhad dostatečně vybuzena, parametry se určí nedostatečně přesně a predikce není dobrá.

Program

```
// Prediction np-steps with scalar regression model of order n
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

// definition of regression vector
deff('ps=genpsi(t,n,y,u)','ps=[y(t-(1:n)),u(t-(0:n)) 1]','c')

load _data/dataT11.dat Sim    // results from simulation T11SimReg1.sce
yt=Sim.Cy.yt;
ut=Sim.Cy.ut;
th=Sim.Cy.th;
cv=Sim.Cy.cv;
ord=Sim.Cy.ord;
load _data/dataT21.dat Est    // results from estimation T21EstReg1.sce
thE=Est.Cy.th;
ordE=Est.Cy.ord;

np=5;     // no of prediction steps (1=prediction from the model)
```

```
nd=length(yt);
yp=zeros(1,nd);
for t=(ordE+1):(nd-np+1)      // time loop
  yi=yt(1:(t-1));             // old data (at time t)
  for j=0:(np-1)              // loop of prediction
    ps=genpsi(t+j,ordE,yi,ut);// construction of regression vector
    yi(t+j)=ps*thE;           // auxiliary prediction
  end
  yp(t+np-1)=yi(t+np-1);      // final prediction at t
end
Pre.Cy.np=np;
Pre.Cy.yp=yp;

// Results
s=(ordE+1):(nd-np);
plot(s,yt(s),s,yp(s))
title('Output and its prediction')
legend('output','prediction');

save _data/dataT31.dat Sim Est Pre
```

Předpověď výstupu s kategorickým modelem

Úloha: Předpověď výstupu jako informace o soustavě nebo pro další použití.

- vstup i výstup s hodnotami 1, 2
- model s pamětí a řízením
- jednokroková a n-kroková predikce

Jako data se využívá výsledku simulace ze souboru T12SimCat který se ukládá jako datový soubor dataT12.dat v adresáři _data a odkud je natažen. Nic ale nebrání tomu, vyrobit si vlastní data - buď přímo v souboru T12SimCat nebo si data uložit na disk a natáhnout si je tak, jak je to připravené. Odhad parametrů modelu se provádí v souboru T22EstCatX v první variantě, tj. pro I_dataSim=1a jeho výsledky jsou uloženy v datovém souboru dataT22.dat.

Použitý model má tvar tabulky

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
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	$\Theta_{1\|22}$	$\Theta_{2\|22}$

kde y, u jsou výstup a vstup, $\Theta_{i|jk}$ jsou pravděpodobnosti.

K dispozici jsou 2 programy:

1. T32PreCatN.sce - jednokroková predikce (simulace z odhadnutého modelu).
2. T32PreCatN.sce - vícekroková predikce pro np kroků dopředu.

Predikce v T32CatN se provádí postupným dosazováním dat nebo bodových predikcí z minulých kroků do modelu.

Předpoklady: Dvouhodnotová data

Značení

- y - yt,
- u - ut,
- $\hat{\Theta}_t$ - thE - odhadnuté parametry.

Volitelné parametry

- np - počet kroků v predikci.

Doporučené experimenty

1. Volte různé druhy soustavy od málo neurčitých až po deterministické. Volbu je třeba provést v simulaci, tj v souboru T12SimCat.
2. Zkuste různý počet kroků predikce np a porovnejte výsledky. Najděte hranici pro délku predikce, kdy ještě dostáváme přijatelné výsledky. Tato hranice je samozřejmě pro každý typ simulované soustavy jiná.
3. Upravte program tak, aby predikce pracovala se skutečnými parametry simulace (známé parametry soustavy). Porovnejte výsledky predikce se známými a odhadnutými parametry.

Program

Jednokroková predikce s bodovými odhady

```
// Prediction one-step ahead with scalar categorical model
// - model is binary + memmory + control   f(z(t)|u(t),z(t-1))
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

load _data/dataT12.dat Sim          // results from task T12SimCat.sce
zt=Sim.Cz.zt;                       //   the data zt, ut
ut=Sim.Cz.ut;                       //   and parameters th are
th=Sim.Cz.th;                       //   extracted from simulation
load _data/dataT22.dat Est          // results from estimation are taken
thE=Est.Cz.th;                      //   from task T22EstCat.sce


nd=min(50,length(zt));              // number of data (minimum is 50)
yp=ones(1,nd);
for t=2:nd                          // time loop
  j=psi2row([ut(t),zt(t-1)],[2,2]);               // row in the table
  yp(t)=sum(rand(1,1,'u')>cumsum(thE(j,:)))+1;    // prediction at t
end
Pre.Cz.np=1;                        // only 1 step prediction
Pre.Cz.yp=yp;

// Results
s=1:nd;
printf('There is %d wrong predictions from %d\n',sum(zt(s)~=yp(s)),length(s))

plot(s,zt(s),'b:o',s,yp(s),'r:.')
legend('output','prediction');
title 'Output and its prediction'

save _data/dataT32.dat Sim Est Pre
```

Vícekroková predikce s bodovými odhady

```
// Prediction np-step with categorical model
// - the old outputs are substituted by point estimates
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

load _data/dataT12.dat Sim          // results from task T12SimCat.sce
zt=Sim.Cz.zt;                       //   from simulation, the data
ut=Sim.Cz.ut;                       //   zt, ut and parameters th
th=Sim.Cz.th;                       //   are extracted
load _data/dataT22.dat Est          // results from task T22EstCat.sce
thE=Est.Cz.th;

np=3;     // numb. of prediction steps (1=prediction from the model)

nd=length(zt);
yp=zeros(1,nd);
for t=2:(nd-np+1)                   // time loop
  yi=zt(1:(t-1));                   // old data (at time t)
  for k=0:(np-1)                    // loop of prediction
    j=psi2row([ut(t+k),yi(t+k-1)],[2,2]);          // row of the table
    yi(t+k)=sum(rand(1,1,'u')>cumsum(thE(j,:)))+1; // point prediction
  end
  yp(t+np-1)=yi(t+np-1);            // final prediction at time t
end
Pre.Cz.np=np;
Pre.Cz.yp=yp;

// results
s=(fix(nd/2)):(nd-np);
printf(' %d wrong predictions from %d\n',sum(zt(s)~=yp(s)),length(s))

plot(s,zt(s),'b:o',s,yp(s),'r:.')
legend('output','prediction');

save _data/dataT32.dat Sim Est Pre
```

Předpověď výstupu s kategorickým modelem

Úloha: Předpověď výstupu jako informace o soustavě nebo pro další použití.

- čtvercová tabulka modelu, tj. model $f\left(y_t|y_{t-1}\Theta\right)$
- obecná n-kroková predikce
- predikce s bodovými odhady ve tvaru maximum pravděpodobnosti nebo střední hodnota
- volba, zda predikce běží se známými nebo odhadovanými parametry

Data jsou v programu simulována.

Použitý model má tvar tabulky

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
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	$\Theta_{1\|22}$	$\Theta_{2\|22}$

kde y, u jsou výstup a vstup, $\Theta_{i|jk}$ jsou pravděpodobnosti.

Predikce se provádí mocněním tabulky modelu (jako čtvercové matice) a násobením počáteční hodnotou.

Předpoklady: Model s čtvercovou tabulkou.

Značení

- y - yt,
- u - ut,
- $\hat{\Theta}_t$ - thE - odhadnuté parametry.

Volitelné parametry

- np - počet kroků v predikci
- I_est - volba predikce se známými nebo odhadovanými parametry
- I_pred - volba bodových odhadů (maximum pravděpodobnosti nebo střední hodnota)

Doporučené experimenty

1. Volte různé druhy soustavy od málo neurčitých až po deterministické.
2. Zkuste různý počet kroků predikce np a porovnejte výsledky. Najděte hranici pro délku predikce, kdy ještě dostáváme přijatelné výsledky. Tato hranice je samozřejmě pro každý typ simulované soustavy jiná.
3. Porovnejte výsledky predikce se známými a odhadovanými parametry.
4. Upravte program tak, aby pracoval s off-line odhadnutými parametry. Porovnejte výsledky se známými, odhadovanými a odhadnutými parametry. Navrhněte kriterium pro porovnání.

Program

```
// Prediction with a "square" discrete model f(z(t)|z(t-1),th)
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

np=5;                                // prediction length
nd=500;                              // data sample length
I_est=2;                             // estimation: 1=yes, 2=no
I_pred=2;                            // prediction: 1=arg max, 2=expect.

T=[.99 .005 .005;                    // simulation model parameter
   .01  .98  .01
   .02 .01  .97];

// SIMULATION
z(1)=1;                              // initial output
for t=2:nd                           // data generation
  z(t)=(sum(rand(1,1,'unif')>(cumsum(T(z(t-1),:)))))+1;
end

// ESTIMATION AND PREDICTION
S=ones(T);                           // initial model statistics
vals=(1:max(size(T)))';              // values of output
for t=2:nd
  S(z(t-1),z(t))=S(z(t-1),z(t))+1; // statistics update
  // known or estimated parameters
  select I_est                       // estimation  1 = no, 2 = yes
  case 1 then
    Te=T;
  case 2 then
    Te=fnorm(S,2);                   // parameter point estimates
  end
  Tp=Te^np;                          // model for prediction
  // type of prediction point estimates
  select I_pred
  case 1 then
    [xxx yy]=max(Tp(z(t),:));
    yp(t+np)=yy;
  case 2 then
    yp(t+np)=Tp(z(t),:)*vals;
  end
end

// RESULTS
s=2:nd;
```

```
set(scf(1),'position',[100 100 500 400])
plot(s,z(s),'o',s,yp(s),'.')
set(gca(),'data_bounds',[1 nd .9 3.1]);
title 'Discrete data and their prediction'
legend('data','prediction');
```

Odhad stavu se stavovým modelem

- stavový lineární model se známými parametry
- simulovaná data

V programu se provádí simulace se stavovým modelem a g generovaného vstupu a výstupu se pomocí Kalmanova filtru průběžně odhaduje stav. Kalmanův filtr je realizován procedurou

[x, xf, rx, yp]=Kalman(x, yt, ut, M, N, F, A, B, G, rw, rv, rx)

kde x je přepočítávaný stav, xf je výsledný odhad stavu v daném čase, rx je přepočítávaná kovariance stavu, yp je predikce výstupu, yt a ut jsou data, M, N, F, A, B, G jsou parametry stavového modelu, rw a rv jsou kovariance stavu a šumu ve stavovém modelu.

Použitý model má tvar

$$\begin{aligned} x_{t+1} &= Mx_t + Nu_t + F + w_t \\ y_t &= Ax_t + Bu_t + G + v_t \end{aligned}$$

kde x_t je v programu označen jako xf

Předpoklady: Známe parametry modelu, linearita modelu

Sci značení: $x_{t-1}/x_t/x_{t+1}$ - x, x_t - xf, (ostatní viz Kalmanův filtr nahoře)

Úloha: Odhad neznámé veličiny, filtrace zašuměného signálu.

Poznámka

Pro správný start odhadování je důležité správné nastavení kovariančních matic. Matice rw se nastavuje většinou jako diagonální s velkými čísly na diagonále ($10^3 - 10^5$). Kovarianční matice rw a rv by měly odpovídat kovariančním maticím šumů w_t a v_t z modelu. POZOR: nejsou to kovarianční matice stavu a výstupu ale jejich poruch. Na správném nastavení těchto matic velmi záleží celý odhad.

Doporučené experimenty

1. Měňte parametry stavového modelu pro simulaci a sledujte, kdy je odhad stavu obtížný.
   Poznámka
   Jestliže matice $\left[A,\, AM,\, AM^2,\, \cdots,\, AM^{n-1}\right]$, kde n je dimenze stavu, má hodnost menší než n, pak systém je tzv. nepozorovatelný a stav nelze odhadnout.
2. Měňte rozptyly šumů rw a rv. Pro velké rozptyly bude odhad méně přesný.
3. Měňte počáteční kovarianční matici rx. Nastavená hodnota 1000 na diagonále znamená, že Kalmanovu filtru necháváme volnost v jeho odhadu z dat. Pro menší diagonálu odhad tzv. "utahujeme", tj. dáváme větší váhu počátečním podmínkám. Tím se ale odhad stabilizuje - snižuje se nebezpečí, že nějaká hodně chybná data na začátku celý odhad rozhodí natolik, že nekonverguje.

Program

```
// Estimation of state variable with KF
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=100;                  // number of data

// State-space model
Fil.Cy.rw=.1*eye(2);     // state covariance
Fil.Cy.rv=.01;           // output covariance
Fil.Cy.rx=1000*eye(2);   // state estimate covariance
Fil.Cy.M=[.2 .1;         // state-space
          .2 .4];        //   model
Fil.Cy.N=[1; -1];        //   parameters
Fil.Cy.F=[1; -1];        //      "
Fil.Cy.A=[1 0];          //      "
Fil.Cy.B=1;              //      "
Fil.Cy.G=2;              //   parameters

// declarations
xt=zeros(2,nd);     // state declaration (initial state)
x=zeros(2,nd);      // initial state estimate
ut=zeros(1,nd);     // input
yt=zeros(1,nd);     // output
yp=zeros(1,nd);     // output prediction
rw=Fil.Cy.rw; rv=Fil.Cy.rv; rx=Fil.Cy.rx;    // KF covariances
M=Fil.Cy.M; N=Fil.Cy.N; F=Fil.Cy.F;          // state model parameters
A=Fil.Cy.A; B=Fil.Cy.B; G=Fil.Cy.G;          // output model parameters

// TIME LOOP
for t=2:nd
  // simulation
  xt(:,t) = M*xt(:,t-1)+N*ut(t)+F+rw*rand(2,1,'norm');
  yt(t)   = A*xt(:,t) + B*ut(t)+G+rv*rand(1,1,'norm');
  // estimation
  [x(:,t),rx,yp(t)]=...
    Kalman(x(:,t-1),yt(t),ut(t),M,N,F,A,B,G,rw,rv,rx);
end
Fil.Cy.ut=ut;
Fil.Cy.yt=yt;
Fil.Cy.xt=x;

// RESULTS
s=1:nd;
subplot(311),plot(s,xt(1,s),s,x(1,s))
set(gca(),"data_bounds",[1,nd,0 2]);
```

```
title('First state and its estimate')
subplot(312),plot(s,xt(2,s),s,x(2,s))
set(gca(),"data_bounds",[1,nd,-2 1]);
title('Second state and its estimate')
subplot(313),plot(s,yp(s),s,yt(s))
set(gca(),"data_bounds",[1,nd,2 4]);
title('Output and its prediction')
set(gcf(),"position",[500 50 600 500])
// Note: xf is the result of state estimation (not x)
//  at time t, xf(t) is the result. x(t+1) aims at the next time

save _data/dataT44.dat Fil
```

Odhad stavu s nelineárním stavovým modelem

- stavový model se známými parametry
- nelineární model
- simulovaná data

V programu se provádí simulace s nelineárním stavovým modelem.

Pro odhad stavu je tento model linearizován pomocí prvých dvou členů Taylorova rozvoje. Linearizovaný model je využit pro odhad stavu z generovaného vstupu a výstupu. Odhad se provádí pomocí Kalmanova filtru. Kalmanův filtr je realizován procedurou

[x, xf, rx, yp]=Kalman(x, yt, ut, M, N, F, A, B, G, rw, rv, rx)

kde x je přepočítávaný stav, xf je výsledný odhad stavu v daném čase, rx je přepočítávaná kovariance stavu, yp je predikce výstupu, yt a ut jsou data, M, N, F, A, B, G jsou parametry stavového modelu, rw a rv jsou kovariance stavu a šumu ve stavovém modelu.

Použitý model má tvar

$$\begin{aligned} x_{1;t+1} &= x_{1;t}x_{2;t} - u_t + w_{1;t} \\ x_{1;t+1} &= 0.5x_{1;t} + 0.8x_{2;t} + u_t + w_{2;t} \\ y_t &= x_{1;t} \end{aligned}$$

kde x_t je v programu označen jako xf

Předpoklady: Známe parametry modelu, linearizace modelu pomocí Taylorova rozvoje

Sci značení: $x_{t-1}/x_t/x_{t+1}$ - x, x_t - xf, (ostatní viz Kalmanův filtr nahoře)

Úloha: Odhad neznámé veličiny, filtrace zašuměného signálu.

Poznámka

Pro správný start odhadování je důležité správné nastavení kovariančních matic. Matice rw se nastavuje většinou jako diagonální s velkými čísly na diagonále ($10^3 - 10^5$). Kovarianční matice rw a rv by měly odpovídat kovariančním maticím šumů w_t a v_t z modelu. POZOR: nejsou to kovarianční matice stavu a výstupu ale jejich poruch. Na správném nastavení těchto matic velmi záleží celý odhad.

Doporučené experimenty

1. Měňte rozptyly šumů rw a rv a sledujte jejich efekt na kvalitu odhadu.
2. Zkuste změnit simulovanou soustavu. Ale pozor! Pro novou soustavu musíte přepočítat matice M, N, A, B, F a G které plynou z linearizace nelineárního modelu.

Program

```
// State estimation (nonlinear model)
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=200;                                   // length of data

ut=.1*(1-rand(1,nd,'norm'));             // control
x1=0; x2=-1;                             // initial state

xs=zeros(2,nd); xs(:,1)=[x1;x2];
yt=zeros(1,nd);

// SIMULATION
for t=2:nd
  x1=x1*x2+.1*x2-ut(t)+.01*rand(1,1,'norm');  // state
  x2=.5*x1+.8*x2+ut(t)+.01*rand(1,1,'norm');  //     equations
  yt(t)=x2;                              // output equation
  xs(:,t)=[x1;x2];                     // store the actual sate variable
end

xx=[0;0];                                // initial state estimate
rw=.1*eye(2,2); rv=.1; rx=1e3*eye(2,2); // model and initial covariances
xt=zeros(2,nd); rr=zeros(2,nd); rr(:,1)=diag(rx);
// ESTIMATION
for t=2:nd
  // parameters of the linearized model
  M=[xx(2) xx(1)+.1; .5 .8];
  N=[-1;1];
  F=[-.1*xx(1)+(.1-xx(2))*xx(2);0];
  A=[0 1];
  B=0;
  G=0;
  // Kalman filter
  [xx,rx,yp]=Kalman(xx,yt(t),ut(t),M,N,F,A,B,G,rw,rv,rx);
  xt(:,t)=xx;                            // stor actual state estimate
  rr(:,t)=diag(rx);                      // stor variance of noise estimate
end

// RESULTS
// figure 1
plot(xs')
plot(xt',':')
legend('state 1','state 2','estimate 1','estimate 2',4);
title('State and its estimate','fontsize',5,'FontName','Times')
xlabel('time','fontsize',4,'FontName','Times')
```

```
ylabel('values','fontsize',4,'FontName','Times')
set(gcf(),'position',[300 100 500 400])
```

Odhad stavu s neznámými parametry

- stavový model
- neznámé parametry
- linearizace vzniklého nelineárního modelu

V programu se provádí simulace s nelineárním stavovým modelem.

Pro odhad stavu se stav modelu se rozšíří o neznámé parametry a tím se stane nelineárním. Model je dále linearizován pomocí prvých dvou členů Taylorova rozvoje. Linearizovaný model je využit pro odhad stavu z generovaného vstupu a výstupu. Odhad se provádí pomocí Kalmanova filtru. Kalmanův filtr je realizován procedurou

[x, xf, rx, yp]=Kalman(x, yt, ut, M, N, F, A, B, G, rw, rv, rx)

kde x je přepočítávaný stav, xf je výsledný odhad stavu v daném čase, rx je přepočítávaná kovariance stavu, yp je predikce výstupu, yt a ut jsou data, M, N, F, A, B, G jsou parametry stavového modelu, rw a rv jsou kovariance stavu a šumu ve stavovém modelu.

Použitý model má tvar

```
a=.6; // unknown model parameter
sM=[a .5 // simulation parametrs
.1 .8];
sN=[.5; .4];
sA=[1 0];
yt(:,t)=sA*x+sqrt(rv)*rand(1,1,'norm'); x=sM*x+sN*ut(:,t)+sqrt(rw)*rand(2,1,'norm');
```

$$\begin{aligned} x_{t+1} &= Mx_t + Nu + w_t \\ y_t &= Ax_t + v_t \end{aligned}$$

kde $M = \begin{bmatrix} a & 0.5 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$, $A = [1, 0]$. V simulaci je $a = 0.6$ v odhadu je a neznámý parametr.

Předpoklady: Neznáme parametry modelu, linearizace modelu pomocí Taylorova rozvoje

Sci značení: $x_{t-1}/x_t/x_{t+1}$ - x, x_t - xf, (ostatní viz Kalmanův filtr nahoře) Parametry modelu pro simulaci jsou sM, sN a sA.

Úloha: Odhad neznámé veličiny, filtrace zašuměného signálu.

Poznámka

Pro správný start odhadování je důležité správné nastavení kovariančních matic. Matice rw se nastavuje většinou jako diagonální s velkými čísly na diagonále ($10^3 - 10^5$). Kovarianční matice rw a rv by měly odpovídat kovariančním maticím šumů w_t a v_t z modelu. POZOR: nejsou to kovarianční matice stavu a výstupu ale jejich poruch. Na správném nastavení těchto matic velmi záleží celý odhad.

Doporučené experimenty

1. Stav je v tomto případě složen z původních stavových proměnných (které se přirozeně v čase mění) a neznámých parametrů (které jsou konstantní). Vzhledem k tomu je třeba volit rozptyly původních stavů větší, zatímco rozptyly neznámých parametrů by měly být velmi malé. Zkuste experimentálně.

2. Nejzajímavější je měnit simulovanou soustavu (včetně dimenze stavu a výstupu). Tato změna ale vyžaduje novou konstrukci rozšířeného stavového modelu (stav je rozšířen o neznámé parametry) a jeho linearizace.

Program

```
// State estimation (model with unknown parameter)
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=100;                         // length of data
a=.6;                           // unknown model parameter
sM=[a .5                        // simulation parametrs
   .1 .8];
sN=[.5; .4];
sA=[1 0];
rv=.01; rw=.01*eye(2,2);        // noise variances

ut=rand(1,nd,'norm');
yt=zeros(1,nd);
xx=zeros(2,1);                   // initial state
xt=xx;

// SIMULATION
for t=1:nd
  xx     =sM*xx+sN*ut(:,t)+sqrt(rw)*rand(2,1,'norm');
  yt(:,t)=sA*xx+sqrt(rv)*rand(1,1,'norm');
  xt(:,t)=xx;
end
xt(3,:)=a;                      // true parameter

Rv=.01; Rw=.01;                 // model covariance matrices
Rx=1e6*eye(3,3);                // covariance matrix of state estimate
N=[.5; .4; 0];                  // parameters of model extended by unknown
A=[1 0 0];                      //     parmeter
G=0;
B=0;
x=zeros(3,1);
xe=zeros(3,nd);
```

```
// ESTIMATION
for t=1:nd
  M=[x(3) .5 0                // parameters of thr linearized model
     .1   .8 0
     0     0 1];
  dM=[x(3) .5 x(1)
      .1   .8 0
      0     0 1];
  F=(M-dM)*x;

  // KALMAN FILTER
  [x,Rx,yp]=Kalman(x,yt(t),ut(:,t),dM,N,F,A,B,G,Rw,Rv,Rx);
  xe(:,t)=x;
end

// RESULTS
set(scf(1),'position',[100 50 1200 400])
// figure 1
subplot(131)
plot(xt(1,:),'b','linewidth',2)
plot(xe(1,:),'r','linewidth',2)
legend('state','estimate');
title('First state and its estimate','fontsize',5,'FontName','Times')
xlabel('time','fontsize',4,'FontName','Times')
ylabel('values','fontsize',4,'FontName','Times')

// figure 2
subplot(132)
plot(xt(2,:),'b','linewidth',2)
plot(xe(2,:),'r','linewidth',2)
legend('state','estimate');
title('Second state and its estimate','fontsize',5,'FontName','Times')
xlabel('time','fontsize',4,'FontName','Times')
ylabel('values','fontsize',4,'FontName','Times')

// figure 3
subplot(133)
plot(xt(3,:),'b','linewidth',2)
plot(xe(3,:),'r','linewidth',2)
legend('true parameter','estimate');
title('Estimate of the parameter','fontsize',5,'FontName','Times')
xlabel('time','fontsize',4,'FontName','Times')
ylabel('values','fontsize',4,'FontName','Times')
```

Klasifikace s regresním modelem

- simulovaná data
- klasifikace - třídění dat do tříd
- logistický model a jeho ML odhad

V programu se provádí odhad modelu logistické regrese (fáze učení) a pro odhadnutý model, tj. model s pevnými parametry, se provádí třídění dat (fáze testování). Data se zařadí do třídy, která je indikována hodnotou odhadnutého výstupu (blíže viz poznámka).

Simulace se provádí tak, ze se generují náhodné vektory x_1 a x_2. Vytvoří se jejich lineární kombinace $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ (ϵ je šum) a pro výsledek větší než 0.5 se přiřadí $y = 1$ jinak $y = 0$.

Model v úloze učení testování má standardní tvar logistické regrese

$$z_t = x_t b$$

$$P\left(y_t = 1 | x_t\right) = \frac{\exp\left\{z_t\right\}}{1 + \exp\left\{z_t\right\}}$$

$$\hat{y}_t = \begin{cases} 1 & \text{pro } P\left(y_t = 1 | x_t\right) > 0.5 \\ 0 & \text{pro } P\left(y_t = 1 | x_t\right) < 0.5 \end{cases}$$

Předpoklady: Dvouhodnotové $y \in \{0, 1\}$

Sci značení: yL, xL - data pro učení; xT - data pro testování; c - odhad výstupu (třídy); yT - správná třída ze simulace.

Úloha: Odhad modelu logistické regrese, odhad třídy dat pro měřené datové záznamy (klasifikace).

Poznámka

V logistické regresi se používají veličiny y - výstup a x - regresní vektor. V klasifikaci mají jinou interpretaci: x jsou datové záznamy, které chceme třídit, y je označení třídy, do které data patří.

Doporučené experimenty

1. Zkuste měnit velikost datového vzorku pro učení (pomocí parametru nL). Sledujte vliv na kvalitu klasifikace. Ta se posuzuje jako počet špatných klasifikací vzhledem k počtu provedených klasifikací.
2. Simulace je v programu provedena heuristicky. Nejdříve se generují hodnoty nezávislých proměnných x. Potom se provede jejich lineární kombinace a výstup y se určí jako 1 pro její kladné hodnoty, v opačném případě je 0. Zkuste navrhnout svou vlastní simulaci.
3. Zkuste provést simulaci generovanou logistickým modelem, pro který si zadáte své vlastní parametry. Porovnejte simulované a odhadnuté parametry.

Program

```
// Classification with logistic regression
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nL=100;                        // number of data for learning
nT=100;                        // number of data for testing
sd=1;                          // magnitude of disturbance

x1=rand(1,nL+nT,'norm');       // 1. regression vector
x2=rand(1,nL+nT,'unif')<.3;    // 2. regression vector
x=[x1;x2]';                    // full regression vector
sL=1:nL;                       // inteval for learning
sT=1:nT;                       // interval for testing
yy=2*x1-3*x2+1+sd*rand(1,nL+nT,'norm');
y=double(yy>0);                // output (true classes)
yL=y(sL);
xL=x(sL,:);

// Learning
// py     probabilities of y=0 and y=1
// yp     prediction of y  yp=py(:,2)
// yr     point prediction of y  yr=round(yp)
// b      model parameters
[py,yp,yr,b]=lrEst([],yL,xL); // log. reg. estimation

// Testing
ct=zeros(sT);
yT=y(1,nL+sT);                 // true classes
xT=[ones(nT,1) x(nL+sT,:)];    // data for testing
for t=sT
  z=xT(t,:)*b;                 // regression
  yp=exp(z)/(1+exp(z));        // logistic function
  ct(t)=round(yp);             // class point estimates
end

// RESULTS
s=1:nT;
set(scf(1),'position',[600 50 600 600])
plot(s,yT(s),'bo',s,ct(s),'r.')
set(gca(),"data_bounds",[min(s)-.1 max(s)+.1 -.1 1.1])
title('Logistic regression')
legend('output','point estimate');

disp('Logistic regression:')
printf(' Wrong %d from %d\n',sum(yT~=ct),nT)
```

Adaptivní řízení s regresním modelem

- regresní model s vektorovým vstupem a výstupem
- model obecného řádu s neznámými parametry
- separace identifikace a řízení
- realizace adaptivního řízení při ustupujícím horizontu

V programu se nejprve provede předběžná identifikace modelu z apriorních dat a spočtou bodové odhady parametrů modelu.

potom, v časové smyčce, se regresní model převede do stavového tvaru. Provádí optimalizace, tj. výpočet řídícího předpisu na celém intervalu řízení, a to proti směru času. Nakonec se realizuje aplikace posledního (tedy aktuálního) předpisu pro řízení. Po jeho aplikaci se generuje (změří) nový výstup a s novými daty se přepočtou bodové odhady parametrů.

Použitý regresní model

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

Model se určí zadáním parametrů u y, u a konstanty.

Předpoklady: Konstantní parametry modelu pro simulaci.

Sci značení: M, N - parametry stavového modelu, Om - penalizace.

Úloha: Řízení s mnohorozměrným regresním modelem.

Poznámka

Realizuje se kvadraticky optimální řízení které má sledovat předem daný průběh. Tomuto průběhu se říká žádaná hodnota nebo set-point.

Při odhadu během řízení mohou nastat potíže s vybuzením soustavy.

Doporučené experimenty

1. Adaptivní řízení ve velice složitá úloha, která vyžaduje určitou inicializaci (předběžný odhad parametrů) tak, aby řízení již od začátku fungovalo alespoň trochu "rozumně". Pokud tomu tak není, může se celá uzavřená smyčka tak vybudit (většinou se generují obrovská čísla, ta se v počítači zaokrouhlí a tím se naruší konzistence systému), že se řízení rozpadne a nekonverguje.

   (a) Zkuste měnit délku počátečního odhadu ni a sledujte kvalitu řídícího procesu (zejména na začátku řízení).

   (b) Dále lze experimentovat s počátečním nastavením informační matice V (tady nastavena jako diagonální s diagonálou 0.1) a statistikou ka (čím je větší, tím více se prosazují počáteční hodnoty parametrů).
   Pozor: Matice V musí být symetrická (viz její přepočet).

2. Pro implementaci algoritmu adaptivního řízení se využívá metoda ustupujícího horizontu. Délka horizontu se zadává jako$Con.Cy.nh$ a je předvolena na hodnotu 3. Měla by být tak velká, aby se počítaný řídící zákon stačil ustálit.
   Zkuste měnit její hodnotu a sledujte kvalitu řízení.

3. V kriteriu řízení se penalizuje kvadrát výstupu a kvadrát přírůstku řídící veličiny, a to s vahou la (určuje významnost penalizace řízení proti penalizaci výstupu).

   (a) Zkuste měnit hodnotu $la \in (0, 1)$ a udělejte závěry o vlivu na kvalitu řízení.

   (b) Zkuste rozmyslet, jak je třeba změnit penalizační matici Om, aby se penalizovaly celé hodnoty řízení, nikoliv přírůstky. Porovnejte řízení s penalizací celé řídící veličiny a jejích přírůstků.

4. Pomocí parametru $Iadapt$ je možno přepínat řízení adaptivní a řízení se známými parametry. Testujte a porovnejte.

Program

```
// ADAPTIVE CONTROL WITH DYNAMIC REG. MODEL WITH CONSTANT
//   multivariate input and output
//   control with setpoint
//   penalization of input increments
//   preliminary (prior) estimation
// ------------------------------------------------
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

ni=50;              // preliminary estimation
nd=150;             // control

I_adapt=0;          // adaptive control  1=yes, 0=no,

Sim.Cy.ord=2;       // model order,    else fix prior est.
Sim.Cy.th(1).a=[.3 -.2; -.05 -.3];
Sim.Cy.th(2).a=[.1 -.1; .02 -.1];
Sim.Cy.thb0=[1 -.4; 1.3 -.4];
Sim.Cy.th(1).b=[.1 0;0 .2];
Sim.Cy.th(2).b=[.1 0;0 .1];
Sim.Cy.thk=[1; -1];
Sim.Cy.sd=.5;              // noise stdev

ord=Sim.Cy.ord;
ny=size(Sim.Cy.th(1).a,1);  // dimension of y
nu=size(Sim.Cy.th(1).b,2)/(1+ord); // and u

th=Sim.Cy.thb0;
for i=1:ord         // parameters -> vector th
```

```
  th=[th Sim.Cy.th(i).a Sim.Cy.th(i).b];
end
th=[th Sim.Cy.thk];
k=Sim.Cy.thk;
sd=Sim.Cy.sd;

Con.Cy.nh=3;        // length of control interval
Con.Cy.om=1;        // penalty:  y(t)^2
Con.Cy.la=.01;      // penalty: (u(t)-u(t-1))^2
nh=Con.Cy.nh;

[XX,XX,XX,nx,ny,nu]=reg2st(th,ord);
                    // penalization matrices
Om=zeros(nx,nx);
Om(1:ny,1:ny)=Con.Cy.om*eye(ny,ny);     // output
nn=ny+nu;
for i=1:nu                              // input increment
  Om((ny+i),(ny+i))=Con.Cy.la;          //  (only for ord>1 !!)
  Om((ny+i+nn),(ny+i+nn))=Con.Cy.la;
  Om((ny+i),(ny+i+nn))=-Con.Cy.la;
  Om((ny+i+nn),(ny+i))=-Con.Cy.la;
end

// setpoint
g=genstp(ny,nd,[3;5],[.95;.92],[-5;10]);

// Prior estimation
yi=zeros(ny,ni);
ui=rand(nu,ni,'norm');
V=.01*eye(nx+ny+nu,nx+ny+nu);
for i=(ord+1):ni
  ps=genps(ord,i,yi,ui);
  yi(:,i)=th*ps + .1*rand(ny,1,'norm');
  Ps=[yi(:,i); ps];
  V=V+Ps*Ps';
end

// TIME LOOP
S=list();
y=zeros(ny,nd); y(:,1:2)=[10 10;-10 -10];
u=zeros(nu,nd);
R=.00001*eye(nx,nx);    // regularizace
for t=(ord+1):(nd-nh)
  // Estimation
  if I_adapt==1
    ps=genps(ord,t,y,u);
    Ps=[y(:,t); ps];
    V=V+Ps*Ps';
    theta=v2thN(V/t,ny);
```

```
  else
    theta=th';
  end

  [M,N,A,nx,ny,nu]=reg2st(theta',ord);

  // Generation of control law
  for i=nh:-1:1
    M(1:ny,$)=-g(:,t+i)+k;
    T=R+Om;
    A=N'*T*N;
    B=N'*T*M;
    C=M'*T*M;
    S(t)=inv(A)*B;
    R=C-S(t)'*A*S(t);
  end
  // Generation of optimal control
  x=genph(ord,t,y,u);
  u(:,t)=-S(t)*x;
  y(:,t)=th*[u(:,t); x]+.01*rand(ny,1,'norm');
end

// Results
s=1:(nd-nh);
subplot(211)
plot(s,y(1,s),s,u(1,s),'--',s,g(1,s),'m.','markersize',2)
legend('output','input','setpoint',4);
subplot(212)
plot(s,y(2,s),s,u(2,s),'--',s,g(2,s),'c.','markersize',2)
set(gcf(),'position',[100 50 600 500])
legend('output','input','setpoint',1);
```

Řízení s kategorickým modelem

- model řízené mince s pamětí
- řízení na jednom intervalu

Použitý kategorický model

$[u_t,\, y_{t-1}]$	$y_t = 1$	$y_t = 2$
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	$\Theta_{1\|22}$	$\Theta_{2\|22}$

Předpoklady: Pevné a známé parametry modelu, řízení na jednom intervalu.

Sci značení: om penalizace, f $= E[fp|d(t)]$ - expectation over $y(t+1)$ fs $= \min f \rightarrow u(t+1)$ - optimal u

Úloha: Řízení s kategorickým modelem se známými parametry.

Poznámka

Realizuje se optimální s obecnou penalizací jednotlivých konfigurací veličin vystupujících v modelu (rozšířený regresní vektor).

Doporučené experimenty

1. Penalizace s v případě řízení s diskrétním modelem volí ve tvaru matice (stejných rozměrů jako u modelu) a penalizuje se každý stav (tj. kombinace veličin y_t, u_t, y_{t-1}) zvlášť. Určete si svou vlastní preferenci pro řízení, realizujte ji penalizační tabulkou *Con.Cz.om* a sledujte, jak je ve výsledku realizována.
2. Zkuste měnit simulovanou soustavu tak, aby v ní bylo více/méně neurčitosti. Sledujte kvalitu řízení.

Program

```
// Dynamic programming with discrete model (single horizon)
//  fp = omega + fs  (penalty function, old criterion)
//  f  = E[fp|d(t)] - expectation over zt(t+1)
//  fs = min f  =>  ut(t+1) - optimal ut
[u,t,n]=file();                         // find working directory
chdir(dirname(n(1)));                   // set working directory
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion

// VARIABLES TO BE SET
```

```
Con.Cz.nh=30;                          // length of control interval
I_detMod=0;                          // 1 for detrministic model: round(th)
y0=1;                          // initial condition for output

//        zt 1 2          ut y1  =  criterion
Con.Cz.om=[0 5        // 1 1
           0 5        // 1 2
           5 0        // 2 1
           5 0];      // 2 2

//       zt  1  2       ut y1  =  system model
Sim.Cz.th=[.2 .8     // 1 1
           .9 .1     // 1 2
           .1 .9     // 2 1
           .8 .2];   // 2 2
if I_detMod==1
  th=round(th);               // deterministický model
end

// computed variables and initializations
fs=zeros(1,2);
om=Con.Cz.om;
nh=Con.Cz.nh;
th=Sim.Cz.th;

// CONTROL LAW COMPUTATION
for t=nh:-1:1
  fp=om+ones(4,1)*fs;         // penalty + reminder from last step
  // expectation
  f=sum((fp.*th),'c');          // expectation over zt
  // minimization
  if f(1)<f(3),               // for zt(t-1)=1
    us(t,1)=1; fs(1)=f(1);    // optimal control, minimum of criterion
  else
    us(t,1)=2; fs(1)=f(3);    // optimal control, minimum of criterion
  end
  if f(2)<f(4),               // for zt(t-1)=2
    us(t,2)=1; fs(2)=f(2);    // optimal control, minimum of criterion
  else
    us(t,2)=2; fs(2)=f(4);    // optimal control, minimum of criterion
  end
end

// CONTROL APPLICATION
zt(1)=y0;
for t=1:nh
  ut(t+1)=us(t,zt(t));          // optimal control
  i=2*(ut(t+1)-1)+zt(t);        // row in the model table
  zt(t+1)=(rand(1,1,'unif')>th(i,1))+1;    // simulation
```

```
end
Con.Cz.zt=zt;
Con.Cz.ut=ut;

// RESULTS
plot(1:nh+1,zt,'ro')
plot(1:nh+1,ut,'g+','markersize',14)
set(gcf(),'position',[200 40 1000 600])
set(gca(),"data_bounds",[.8 nh+.2, .8 2.2])
title('Optimal control with discrete model')
legend('output','input');
```

Odhad směsi se statickým ukazovátkem i komponentami[1]

- dvourozměrný výstup, bez řízení
- simulovaná data
- inicializace odhadu - zašuměné parametry ze simulace
- standardní odhad / odhad s pevnými kovariancemi šumů komponent
- model ukazovátka $f(c_t|\alpha) = \alpha_{c_t}$

Simulují se data ze směsi normálních, dvourozměrných regresních komponent a statického ukazovátka

$$y_t = \theta_i + e_t, \;\; i = 1, 2, \cdots, n_c$$

kde y_t je výstup v čase t,

$\theta_i = \begin{bmatrix} (\theta_1)_i \\ (\theta_2)_i \end{bmatrix}$ jsou parametry statických komponent s dvourozměrným výstupem,

n_c je počet komponent.

Předpoklady: $e \sim N(0, r)$, r konstantní.

Sci značení: y - yt, θ - th, r - cv.

Úloha: Simulace a odhad s modelem směsi statických komponent - základní konfigurace pro odhad směsi.

Poznámka

Inicializace odhadu směsi komponent je velice důležitá. Pokud nejsou počáteční centra komponent dostatečně blízko dat, jsou váhy w_t prakticky nulové (hodnota distribuce normálního rozdělení velmi rychle klesá se vzdáleností od střední hodnoty) a odhad je velmi nepřesný nebo dokonce selhává. Proto je vždy třeba počátečnímu nastavení center (středních hodnot) a šířce (kovarianční matici) odhadovaných komponent věnovat patřičnou péči.

Zde pro inicializaci využijeme znalost "skutečných" parametrů ze simulace. Jako počáteční hodnoty parametrů v odhadu použijeme skutečné hodnoty parametrů, a ty více nebo méně zašumíme.

Tento trik v inicializaci, který je z praktického hlediska "nefér" použijeme proto, že nám zde jde spíše o testování odhadu směsi (při kterém zkoušíme, co tento odhad dokáže), než o skutečný odhad, např. pro reálná data, kde samozřejmě "skutečné" parametry neznáme.

Doporučené experimenty

1. Komponenty modelu jsou statické. Tedy každá komponenta je gaussovský kopeček se středem daným regresními koeficienty *th* a šířkou úměrnou rozptylu *cv*. Protože komponenty jsou dvourozměrné, pohybujeme se v rovině a středy jsou body. Volbou středů a rozptylů můžeme volit klastry, které se překrývají nebo naopak jsou prakticky disjunktní. Podle

[1] Tato úloha je paralelní s úlohou T71MixDyn a liší se jen modelem ukazovátka, který je zde statický.

toho je možno nastavit situaci, kdy klasifikace je triviální a čekáme, že všechny rozhodnutí budou správné nebo naopak, kdy správní klasifikace je velmi obtížná a jsme spokojeni i s určitým procentem správných rozhodnutí.
Zkuste nastavit své vlastní soustavy a ověřte procento správných klasifikací.

2. Odhad modelu směsi je dosti obtížný a lze konstatovat, že bez nějaké inicializace (nalezení středů komponent poblíže center skutečných klastrů) nebude úspěšný. V této úloze se používají počáteční parametry odvozené od skutečných parametrů soustavy (parametry ze simulace).
Zkuste tyto parametry nastavit ručně a sledujte jejich vliv na odhad (zejména jeho počáteční fázi).

3. Jedna z velice úspěšných možností jak přimět odhad, aby se na začátku "chytil", je buď neodhadovat rozptyly komponent a ponechat je počáteční malé nebo s jejich odhadem začít až v průběhu odhadování. Odhad/neodhad rozptylů je možno zvolit pomocí parametru $IstCov$. Zpožděné odhadování je třeba doprogramovat: if t>100, $\cdots$, end
Zkuste odhadovat různými způsoby a porovnejte výsledky.

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné komponenty.

2. Odhad se provádí podle standardních vzorců:

   (a) výpočet vah w_t pro změřený výstup y_t.

   (b) Přepočet statistik komponent a ukazovátka.

   (c) Konstrukce bodových odhadů parametrů. Tady je možnost volby:

      i. průběžné odhadování kovariančních matic komponent,

      ii. použití počátečních kovariančních matic bez průběžného přepočtu.
      Tato varianta je bezpečnější. Při průběžném odhadu se může stát, že jedna komponenta překryje ostatní a výsledek je daný právě jen touto komponentou.
      Možná je také varianta, kdy v prvé části odhadování ponecháme kovarianční matice pevné, a v další části je již odhadujeme.

3. Jako výsledek se ukazují hodnoty odhadovaného ukazovátka ve srovnání se simulovaným. Tím úloha získává charakter klasifikace.

Kód programu
Program

```
// Mixture estimation - static components and pointer model
// - simulated data two-dimensional data
// - initialization by parameters from simulation + noise
// - with or withou estimation of covariances of components
//   FOR NC>5 IT IS NECESSARY TO SET PARAMETERS FOR SIMULATION
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
```

```
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion
rand('seed',0)

nd=100;             // number of data
nc=3;               // number of componentd
I_estCov=0;         // estimation of noise covariances ? 0|1 no|yes

// simulated reg.coef.
Sim.nc=nc;
Sim.Cy(1).th=[0.9 -.5]';
Sim.Cy(2).th=[0.4 0.6]';
Sim.Cy(3).th=[-.7 -.5]';
Sim.Cy(4).th=[-.1  .8]';
Sim.Cy(5).th=[ .5 -.1]';
// simulated noise covariances
Sim.Cy(1).sd=0.1*[1 0;0,1];
Sim.Cy(2).sd=0.1*[1 0;0,1];
Sim.Cy(3).sd=0.1*[1 0;0,1];
Sim.Cy(4).sd=0.1*[1 0;0,1];
Sim.Cy(5).sd=0.1*[1 0;0,1];
// simulated pointer parameters
Sim.Cp.th=fnorm(ones(1,nc)+.1,2);

Sim.ct(1)=1;                // initial poiner

// initial parameters
a=.8;                       // std of scattering initial parametrs
for j=1:nc                  //  from those used in simulation
  [mr,mc]=size(Sim.Cy(j).th);
  Ps=[Sim.Cy(j).th;1]+[a*rand(mr,mc,'n');0]; // initial parameters
  Est.Cy(j).V=Ps*Ps';                        // statistics
  Est.Cy(j).th=Sim.Cy(j).th+a*rand(2,1,'n'); // pt.est. of reg.coef
  Est.Cy(j).sd=.1*eye(2,2);                  // standard deviation
end
Est.ka=ones(1,nc);          // counter
Est.Cp.V=ones(1,nc);        // pointer statistics
Est.Cp.th=fnorm(ones(1,nc)); // pointer parameter
w=fnorm(ones(1,nc));        // weights

// SIMULATION ============================================================
for t=2:nd
  Sim.ct(t)=sum(rand(1,1,'u')>cumsum(Sim.Cp.th))+1;    // pointer
  j=Sim.ct(t); // active component
  y=Sim.Cy(j).th+Sim.Cy(j).sd*rand(2,1,'norm');        // output
  Sim.yt(:,t)=y;
end

// ESTIMATION ============================================================
```

```
printf(' ')
for t=2:nd
  if t/10==fix(t/10), printf('.'); end
  for j=1:nc
    [xxx,G(j)]=GaussN(Sim.yt(:,t),Est.Cy(j).th,Est.Cy(j).sd);
                                          // proximity
  end
  Lq=G-max(G);
  q=exp(Lq);

  ww=q'.*Est.Cp.th; w=ww/sum(ww);          // generation of weights
  wt(:,t)=w';

  // Update of statistic
  Ps=[Sim.yt(:,t)' 1];                     // extended reg.vec.
  for i=1:nc
    Est.Cy(i).V=Est.Cy(i).V+w(i)*Ps'*Ps;   // information matrix
    Est.ka(i)=Est.ka(i)+w(i);              // counter
    Est.Cp.V(i)=Est.Cp.V(i)+w(i);          // pointer statistics

    //nove rozdeleni informacni matice V
    Vyy=Est.Cy(i).V(1:2,1:2);              // part Vyy - psi.psi'
    Vy=Est.Cy(i).V($,1:2);                 // part Vy  - psi.y
    V1=Est.Cy(i).V($,$);                   // part V1  - y.y
    Est.Cy(i).th=inv(V1+1e-8*eye(V1))*Vy;  // pt.est. - reg.coef.
    Est.Cy(i).tht(:,t)=Est.Cy(i).th';      // pt.est. - covar.
    if I_estCov~=0
      // pt.est. of noise covariance - used or not
      Est.Cy(i).cv=(Vyy-Vy'*inv(V1+1e-8*eye(V1))*Vy)/Est.ka(i);
    end
  end
  Est.Cp.th=fnorm(Est.Cp.V,2);     // pt.est. of pointer parameter
  [ss,Est.ct(1,t)]=max(w);         // store
end

// Results
disp(Sim.Cp.th,'pt.pars_sim')
disp(Est.Cp.th,'pt.pars_est')

s=2:nd;
wr=sum(Sim.ct(s)~=Est.ct(s)');
printf('\n Wrong classifications %d from %d\n',wr,length(s))

set(scf(1),'position',[50 50 400 400])
plot(s,Sim.ct(s),'bo',s,Est.ct(s),'rx')
legend('simulated','estimated');
title 'Values of the pointer'

set(scf(2),'position',[550 50 400 600])
```

```
for i=1:nc
  subplot(nc,1,i)
  plot(Est.Cy(i).tht')
  title('Parameters estimate evolution '+string(i))
end
```

```
// Mixture estimation - static components and pointer model
// - real two-dimensional data
// - initialization from expert knowledge
//   initial parameters are set through simulation structures + noise
// - with or without noise covariances estimation
[u,t,n]=file();                         // find working directory
chdir(dirname(n(1)));                   // set working directory
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion
rand('seed',0)

nd=100;             // number of data
nc=3;               // number of components
I_estCov=0;         // estimation of noise covariances ? 0|1 no|yes

// MODEL INITIALIZATION
// Here expertly based initial parameters are set
// (they are copied to estimation structure Est)
// initial regression coefficients
Sim.nc=nc;
Sim.Cy(1).th=[0.9 -.5]'; // HERE SET THE INITIAL CENTERS OF COMPONENTS
Sim.Cy(2).th=[0.4 0.6]'; //  -"-
Sim.Cy(3).th=[-.7 -.5]'; //  -"-
// initial (or fixed) covariances
Sim.Cy(1).sd=0.1*[1 0;0,1];   // covariances of components
Sim.Cy(2).sd=0.1*[1 0;0,1];   //  for I_estCov=0; stay fixed
Sim.Cy(3).sd=0.1*[1 0;0,1];   //  (they are not estimated)
// initial pointer probabilities
Sim.Cp.th=fnorm([1 1 1]+.1,2);

Sim.ct(1)=1;                  // initial poiner value
```

```
// initial parameters
a=.8;                          // std of scattering initial parametrs
for j=1:nc                     //  from those used in simulation
  Ps=[Sim.Cy(j).th;1];         // initial parameters
  Est.Cy(j).V=Ps*Ps';                          // statistics
  Est.Cy(j).th=Sim.Cy(j).th+a*rand(2,1,'n'); // pt.est. of reg.coef
  Est.Cy(j).sd=.1*eye(2,2);                  // standard deviation
end
Est.ka=ones(1,nc);             // counter
Est.Cp.V=ones(1,nc);           // pointer statistics
Est.Cp.th=fnorm(ones(1,nc));   // pointer parameter
w=fnorm(ones(1,nc));           // weights

// DATA ========================================================
load _data/dataAuto.dat         // HERE, THE DATA ARE LOADED
Sim.yt=dt([1 2],1:nd);   // DATA ASSIGNMENT (time runs in rows)

// ESTIMATION ==================================================
printf(' ')
for t=2:nd
  if t/10==fix(t/10), printf('.'); end
  for j=1:nc
    [xxx,G(j)]=GaussN(Sim.yt(:,t),Est.Cy(j).th,Est.Cy(j).sd); // likelihood
  end
  Lq=G-max(G);
  q=exp(Lq);

  ww=q'.*Est.Cp.th; w=ww/sum(ww);            // generation of weights
  wt(:,t)=w';

  // Update of statistic
  Ps=[Sim.yt(:,t)' 1];                       // extended reg.vec.
  for i=1:nc
    Est.Cy(i).V=Est.Cy(i).V+w(i)*Ps'*Ps;     // information matrix
    Est.ka(i)=Est.ka(i)+w(i);                // counter
    Est.Cp.V(i)=Est.Cp.V(i)+w(i);            // pointer statistics

    //nove rozdeleni informacni matice V
    Vyy=Est.Cy(i).V(1:2,1:2);                // part Vyy - psi.psi'
    Vy=Est.Cy(i).V($,1:2);                   // part Vy  - psi.y
    V1=Est.Cy(i).V($,$);                     // part V1  - y.y
    Est.Cy(i).th=inv(V1+1e-8*eye(V1))*Vy;    // pt.est. - reg.coef.
    Est.Cy(i).tht(:,t)=Est.Cy(i).th';        // pt.est. - covar.
    if I_estCov~=0
      // pt.est. of noise covariance - used or not
      Est.Cy(i).cv=(Vyy-Vy'*inv(V1+1e-8*eye(V1))*Vy)/Est.ka(i);
    end
  end
```

```
  Est.Cp.th=fnorm(Est.Cp.V,2);     // pt.est. of pointer parameter
  [ss,Est.ct(1,t)]=max(w);         // store
end

// Results
s=2:nd;
set(scf(2),'position',[550 50 400 600])
for i=1:nc
  subplot(nc,1,i)
  plot(Est.Cy(i).tht')
  title('Evolution of parameter estimatess '+string(i))
end
```

Odhad směsi s dynamickým ukazovátkem a statickými komponentami[2]

- dvourozměrný výstup, bez řízení
- simulovaná data
- inicializace odhadu - zašuměné parametry ze simulace
- standardní odhad / odhad s pevnými kovariancemi šumů komponent
- model ukazovátka $f\left(c_t|c_{t-1},\alpha\right)=\alpha_{c_t|c_{t-1}}$

Simulují se data ze směsi normálních, dvourozměrných regresních komponent a statického ukazovátka

$$y_t=\theta_i+e_t,\ \ i=1,2,\cdots,n_c$$

kde y_t je výstup v čase t,

$\theta_i=\begin{bmatrix}(\theta_1)_i\\(\theta_2)_i\end{bmatrix}$ jsou parametry statických komponent s dvourozměrným výstupem,

n_c je počet komponent.

Předpoklady: $e\sim N\left(0,r\right),\ \ r$ konstantní.

Sci značení: y - yt, θ - th, r - cv.

Úloha: Simulace, odhad a predikce s dynamickým modelem směsi statických komponent - základní konfigurace pro odhad směsi.

Poznámky

1. *Dynamické ukazovátko umožňuje predikci aktivní komponenty, protože dává do souvislosti po sobě jdoucí aktivity komponent.*

2. *Inicializace odhadu směsi komponent je velice důležitá. Pokud nejsou počáteční centra komponent dostatečně blízko dat, jsou váhy w_t prakticky nulové (hodnota distribuce normálního rozdělení velmi rychle klesá se vzdáleností od střední hodnoty) a odhad je velmi nepřesný nebo dokonce selhává. Proto je vždy třeba počátečnímu nastavení center (středních hodnot) a šířce (kovarianční matici) odhadovaných komponent věnovat patřičnou péči.*

   Zde pro inicializaci využijeme znalost "skutečných" parametrů ze simulace. Jako počáteční hodnoty parametrů v odhadu použijeme skutečné hodnoty parametrů, a ty více nebo méně zašumíme.

   Tento trik v inicializaci, který je z praktického hlediska "nefér" použijeme proto, že nám zde jde spíše o testování odhadu směsi (při kterém zkoušíme, co tento odhad dokáže), než o skutečný odhad, např. pro reálná data, kde samozřejmě "skutečné" parametry neznáme.

[2] Tato úloha je paralelní s úlohou T71MixStat a liší se jen modelem ukazovátka, který je zde dynamický.

Doporučené experimenty

(Jsou stejné, jako pro statickou směs. Zde je uvedeme jen zkráceně. Podrobněji jsou v T71Mix1Stat)

1. Zkuste nastavit své vlastní soustavy a ověřte procento správných klasifikací.
2. Zkuste nastavit parametry ručně a sledujte jejich vliv na odhad (zejména jeho počáteční fázi).
3. Jedna z velice úspěšných možností jak přimět odhad, aby se na začátku "chytil", je buď neodhadovat rozptyly komponent a ponechat je počáteční malé nebo s jejich odhadem začít až v průběhu odhadování. Odhad/neodhad rozptylů je možno zvolit pomocí parametru *IstCov*. Zpožděné odhadování je třeba doprogramovat: if t>100, $\cdots$, end
   Zkuste odhadovat různými způsoby a porovnejte výsledky.
4. (nové) Dynamické ukazovátko znamená, že výběr aktivní komponenty závisí na minulé aktivní komponentě. Tedy v přepínání aktivních komponent je určitý řád, daný modelem ukazovátka (kategorický dynamický model). Předvolený model ukazovátka je dán rovnoměrnou tabulkou *Sim.Cp.th*.
   Zvolte svůj vlastní model ukazovátka (např. deterministický) a sledujte efekt, který do odhadu takové směsi přináší.

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné komponenty.
2. Odhad se provádí podle standardních vzorců:
   (a) výpočet vah W_t a w_t pro změřený výstup y_t.
   (b) Přepočet statistik komponent a ukazovátka.
   (c) Konstrukce bodových odhadů parametrů. Tady je možnost volby:
      i. průběžné odhadování kovariančních matic komponent,
      ii. použití počátečních kovariančních matic bez průběžného přepočtu.

      Tato varianta je bezpečnější. Při průběžném odhadu se může stát, že jedna komponenta překryje ostatní a výsledek je daný právě jen touto komponentou.

      Možná je také varianta, kdy v prvé části odhadování ponecháme kovarianční matice pevné, a v další části je již odhadujeme.
3. Jako výsledek se ukazují hodnoty odhadovaného ukazovátka ve srovnání se simulovaným. Tím úloha získává charakter klasifikace.

Kód programu

```
// Mixture estimation - static components and dynamic pointer model
// - simulated two-dimensional data
// - initialization by parameters from simulation + noise
// - dynamic components
[u,t,n]=file();                         // find working directory
chdir(dirname(n(1)));                   // set working directory
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion
rand('seed',0)

nd=100;             // number of data
nc=3;               // number of componentd
I_estCov=0;         // estimation of noise covariances ? 0|1 no|yes

// simulated reg.coef.
Sim.nc=nc;
Sim.Cy(1).th=[0.9 -.5]';
Sim.Cy(2).th=[0.4 0.6]';
Sim.Cy(3).th=[-.7 -.5]';
// noise covariances
r=.1;                               // amplitude of noise covariances
Sim.Cy(1).sd=r*[1 0;0,1];
Sim.Cy(2).sd=r*[1 0;0,1];
Sim.Cy(3).sd=r*[1 0;0,1];
// simulated noise covariances
Sim.Cp.th=fnorm(rand(nc,nc,'u')+.1,2);

Sim.ct(1)=1;                        // initial poiner

// initial parameters
a=.5;   // std of scattering init.params from simulated ones
for j=1:nc                          //  from those used in simulation
  [mr,mc]=size(Sim.Cy(j).th);
  Ps=[Sim.Cy(j).th;1]+[a*rand(mr,mc,'n');0]; // initial parameters
  Est.Cy(j).V=Ps*Ps';                       // statistics
  Est.Cy(j).th=Sim.Cy(j).th+a*rand(2,1,'n'); // pt.est. of reg.coef
  Est.Cy(j).sd=.1*eye(2,2);                 // standard deviation
end
Est.ka=ones(1,nc);                          // counter
Est.Cp.V=.1*ones(nc,nc);                    // pointer statistics
Est.Cp.th=fnorm(rand(nc,nc,'u')+.1);        // pointer parameter
w=fnorm(ones(1,nc));                        // weights

// SIMULATION ==========================================================
for t=2:nd
  i=Sim.ct(t-1);    // last active component
  Sim.ct(t)=sum(rand(1,1,'u')>cumsum(Sim.Cp.th(i,:)))+1;    // pointer
  j=Sim.ct(t);      // active component
  y=Sim.Cy(j).th+Sim.Cy(j).sd*rand(2,1,'norm');             // output
```

```
  Sim.yt(:,t)=y;                                          // stor
end

// ESTIMATION ============================================================
printf(' ')
for t=2:nd
  if t/10==fix(t/10), printf('.'); end
  for j=1:nc
    [xxx,G(j)]=GaussN(Sim.yt(:,t),Est.Cy(j).th,Est.Cy(j).sd); // likelihood
  end
  Lq=G-max(G);                // rough normlization
  q=exp(Lq);                  // exponent

  ww=(q*w)'.*Est.Cp.th;       // matrix weights
  W=ww/sum(ww);               // normalization    - f(c(t),c(t-1)|d(t))
  w=sum(W,1);                 // marginalization  - f(c(t)|d(t))
  wt(:,t)=w';                 // stor

  // Update of statistic
  Est.ka=Est.ka+w;                          // counter
  Est.Cp.V=Est.Cp.V+W;                      // ptr.stat. update
  Ps=[Sim.yt(:,t)' 1];                      // extended reg.vec.
  for i=1:nc
    Est.Cy(i).V=Est.Cy(i).V+w(i)*Ps'*Ps;    // information matrix

    //nove rozdeleni informacni matice V
    Vyy=Est.Cy(i).V(1:2,1:2);               // part Vyy - psi.psi'
    Vy=Est.Cy(i).V($,1:2);                  // part Vy  - psi.y
    V1=Est.Cy(i).V($,$);                    // part V1  - y.y
    Est.Cy(i).th=inv(V1+1e-8*eye(V1))*Vy;   // pt.est. - reg.coef.
    Est.Cy(i).tht(:,t)=Est.Cy(i).th(:);     // pt.est. - covar.
    if I_estCov~=0
      // pt.est. of noise covariance - used or not
      Est.Cy(i).cv=(Vyy-Vy'*inv(V1+1e-8*eye(V1))*Vy)/Est.ka(i);
    end
  end
  Est.Cp.th=fnorm(Est.Cp.V,2);     // pt.est. of pointer parameter
  [ss,Est.ct(1,t)]=max(w);         // store pointer values
end

// Results
disp(Sim.Cp.th,'pt.pars_sim')
disp(Est.Cp.th,'pt.pars_est')

s=2:nd;
wr=sum(Sim.ct(s)~=Est.ct(s)');
printf('\n Wrong classifications %d from %d\n',wr,length(s))

set(scf(1),'position',[50 50 400 400])
```

```
plot(s,Sim.ct(s),'bo',s,Est.ct(s),'rx')
title 'Pointer values and their predictions'

set(scf(2),'position',[550 50 400 600])
for i=1:nc
  subplot(nc,1,i)
  plot(Est.Cy(i).tht')
  title('Evolution of parameter estimates '+string(i))
end
```

Předpověď s modelem směsi komponent

- dvourozměrný výstup, bez řízení
- simulovaná data
- model se známými parametry
- statické komponenty
- dynamický model ukazovátka $f\left(c_t|c_{t-1},\alpha\right)=\alpha_{c_t|c_{t-1}}$

Simulují se data ze směsi normálních, dvourozměrných regresních komponent a statického ukazovátka

$$y_t=\theta_i+e_t,\ \ i=1,2,\cdots,n_c$$

kde y_t je výstup v čase t,

$\theta_i=\begin{bmatrix}(\theta_1)_i\\(\theta_2)_i\end{bmatrix}$ jsou parametry statických komponent s dvourozměrným výstupem,

n_c je počet komponent.

Délka predikce je n_p. Jsme v čase t, známe $y\left(t-1\right)$ a právě jsme změřili y_t.

Předpověď konstruujeme podle následujícího algoritmu:

- **Jeden krok**
  1. $\forall$ komponenty $c=1,2\cdots n_c$ spočtěte $m_c=f\left(y_t|\hat{\theta}_{t-1}\right)$
     tj, vektor "vzdáleností" y_t od jednotlivých komponent; jeho prvky jsou jsou hodnoty hp komponent s dosazenou hodnotou y_t a bodovými odhady parametrů z času $t-1$.
  2. Určete bodový odhad parametru modelu ukazovátka $\hat{\alpha}_{t-1}$
  3. Vezměte vektor vah w_{t-1} z minulého kroku.
  4. Určete matici vah W_t, reprezentující sdruženou pravděpodobnost $f\left(c_t,c_{t-1}|d\left(t\right)\right)$ takto
     $$W_t=\left(w_{t-1}m'\right).*\hat{\alpha}_{t-1}$$
     kde vektory w a m jsou sloupce - tedy násobíme sloupec krát řádek, což dá matici, a násobení $.*$ je násobení dvou matic prvek po prvku.
  5. Váhový vektor w_t, který určuje aktivity komponent v aktuálním čase a s veškerou dostupnou informaci (včetně y_t) spočteme marginalizací, tedy vysčítáme přes minulé ukazovátko (tj. sečteme W_t ve sloupcích).
  6. Predikce výstupu $\hat{y}_t$ je
     $$\hat{y}_t=\sum_{c=1}^{n_c}w_c\hat{y}_{c;t}$$
     kde $\hat{y}_{c;t}$ jsou "obyčejné" predikce z jednotlivých komponent.

- **Více kroků**

  Provádí se bez další měřené informace (budoucí hodnoty y neznáme). Predikce ukazovátka je dána násobením váhy w_t aktuálním odhadem parametru modelu ukazovátka $\hat{\alpha}_{t-1}$.Tedy

  $$\hat{w}_{t+k} = (\hat{\alpha}_{t-1})^k \, w_t$$

  a opět

  $$\hat{y}_{t+k} = \sum_{c=1}^{n_c} \hat{w}_{c;t+k} \hat{y}_{c;t}$$

  kde $\hat{y}_{c;t}$ jsou predikce z komponent (protože komponenty jsou statické, časový index nehraje roli).

Předpoklady: $e \sim N(0, r)$, r konstantní.

Sci značení: y - yt, θ - th, r - cv.

Úloha: Simulace a predikce s dynamickým modelem směsi statických komponent - základní konfigurace pro predikci se směsí.

Poznámky

Uvedený program je vlastně "vytažen" z programu pro odhad parametrů směsi a předpověď budoucího výstupu. Zde se vynechá přepočet statistik a výpočet bodových odhadů parametrů. Co zůstává je výpočet vah komponent. S nimi se pak kombinují predikce z jednotlivých komponent a dají předpověď celé směsi.

Doporučené experimenty

1. Nastavte svoji simulovanou soustavu (s různým překryvem simulovaných klastrů) a sledujte, jak jsou jednotlivé soustavy predikovatelné.
   Pozor. V rámci statických komponent nemá smysl mluvit o predikci. Predikovatelnost tedy má smysl sledovat jako pokrytí simulovaných klastrů predikovanými. Jde tedy spíše o predikci komponent než jednotlivých bodů, generovaných komponentami.

2. Program umožňuje nastavit délku predikce (pomocí parametru np).
   Testujte, do jaké vzdálenosti má ještě cenu predikovat (samozřejmě také v závislosti na typu simulované soustavy).

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné komponenty. K dispozici jsou 3 varianty modelu ukazovátka od deterministického, přes téměř deterministického až po model s více neurčitostí. Model ale můžeme volit zcela libovolný, zadám matice alS s rozměry $nc \times nc$.

2. Odhad se provádí nestandardně - počítají se jen váhy w_t vždy vzhledem k naměřeným datům

(a) výpočet vah w_t pro změřený výstup y_t.

3. Jako hlavní výsledek se ukazují simulované a predikované klastry.

Kód programu

```
// Prediction with known mixture (without estimation)
// - static components
// - dynamic pointer model
// - simulated one-dimensional data
// - known model parameters
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

np=5;                                // number of steps of prediction
nc=3;                                // number of components
nd=np+1+1000;                        // number of data

// INITIALIZATION of simulation ----------------------------------------
thS=[-2 1 5];                        // centres of components
cvS=[.1 .1 .1]*1;
select 1       // these pointr models are pre-set, you can either
               // chose one or write your own model. Rows must sum to 1
case 1 then, alS=[0 1 0; 0 0 1; 1 0 0];
case 2 then, alS=[.07 .87 .06; .06 .07 .87; .92 .04 .04];
case 3 then, alS=[.8 .1 .1;.1 .2 .7;.1 .1 .8];  // parameters of the pointer
end
c(1)=1;                              // initial values

// SIMULATION
for t=2:nd
  rnd=rand(1,1,'unif');
  cus=cumsum(alS(c(t-1),:));
  c(t)=sum(rnd>cus)+1;               // pointer generation
  th=thS(c(t));
  sd=sqrt(cvS(c(t)));
  y(t)=th+sd*rand(1,1,'norm');       // output generation
end

w1=ones(nc,1)/nc;                    // initial pointer distribution

// PREDICTION
for t=2:nd // TIME LOOP ---------------------------------------------
  yt=y(t);                           // measured output

  // proximity of yt to components
  for i=1:nc
```

```
    m(i)=GaussN(yt,thS(i),cvS(i)); // density of normal component
  end

  // weighting vector
  wP=fnorm(alS'*w1);               // weights for prediction
  wp=(alS')^np*wP;                 // np-steps prediction
  w=fnorm(wP.*m);                  // normalization

  // update of statistics
  Ps=[yt;1];                       // ext. regression vector
  yp=0;
  for i=1:nc
    yp=yp+wp(i)*(thS(i)+sqrt(cvS(i))*rand(1,1,'norm'));
  end
  ytp(t+np)=yp;

  w1=w;                            // old weighting vector
  wt(:,t)=w;                       // stor weighting vector
  wtp(:,t+np)=wp;
end  // END OF TIME LOOP ----------------------------------------------

// RESULTS
[xxx ce]=max(wtp,'r');              // estimated active components
s=(np+2):nd;
wr=sum(c(s)~=ce(s)');
tx='\nWrong class. %d from %d, i.e. %d percent\n';
printf(tx,wr,length(s),100*wr/length(s))
s=(nd-100+1):nd;
plot(s,c(s),'bo','markersize',8)
plot(s,ce(s),'r.','markersize',3)
title 'Pointer and its estimate'
legend('pointer','estimate');
set(gca(),'data_bounds',[min(s) max(s) .9 nc+.1])
set(gcf(),'figure_position',[400,100])

scf(1);
set(gcf(),'figure_position',[600,150])
title 'Histograms of data and predicted data'
subplot(211)
[f1,n1]=nan_hist(y,150,'plot');
ylabel 'data'
subplot(212)
[f2,n2]=nan_hist(ytp,150,'plot');
ylabel 'predictions'

s=551:600;
scf(2);
set(gcf(),'figure_position',[200,150])
title 'time course of data and predictions'
```

```
plot(s,y(s),'ob')
plot(s,ytp(s),'xr')
legend('data','predictions');
```

Klasifikace s modelem směsi komponent

- statický model $f\left(y|x_1, x_2 \cdots, x_n\right)$
- diskrétní výstup
- x_i spojité nebo diskrétní
- hodnoty y interpretovány jako indexy tříd v klasifikaci
- 2 fáze: (i) učení s učitelem, (ii) testování

Simulují se regresní vektory, provede se jejich lineární kombinace a ta se diskretizuje na intervalech. Pořadí intervalů definuje hodnoty y.

Učení

Regresní vektory x rozdělíme do tříd podle hodnot y. Regresní vektory ze třídy i označíme X_i. Pro každou třídu spočteme průměr

$$\theta_i = \frac{1}{N_i} \sum_{x \in X_i} x$$

kde N_i je počet regresních vektorů ve třídě i a dále kovarianční matici

$$r_i = \frac{1}{N_i} \sum_{x \in X_i} \left(x - \theta_i\right) \left(x - \theta_i\right)^{'}.$$

Všechny vektory jsou sloupce. Tyto odhady odpovídají odhadu směsi normálních komponent, kde aktivity komponent jsou známy (učení s učitelem).

Testování

Pro testování předpokládáme odhadnutý (naučený) model. Pro měřené (klasifikované) regresní vektory provádíme "jakoby odhad", parametry modelu už ale neměníme[3]

Pro změřený regresní vektor x spočteme aproximované likelihoody jednotlivých komponent - do komponent dosadíme x (odhady parametrů máme z fáze učení).

Vektor likelihoodů násobíme prvek po prvku vektorem stacionárních pravděpodobností jednotlivých komponent (tedy vektorem normovaných četností jednotlivých tříd tak, jak se objevily ve fázi učení).

Součin normujeme a získáme vektor aktuálních pravděpodobností komponent vzhledem je změřenému x. Tento vektor nazýváme váhovým vektorem.

Vektor x klasifikujeme do třídy, určené pořadím (indexem) maximálního prvku váhového vektoru.

Předpoklady: Postup odpovídá odhadu se směsí normálních komponent.

Sci značení:

xL, yL - data pro učení

xT - regresní vektory pro testování

Úloha: Klasifikace dat - základní konfigurace pro směsovou klasifikaci.

[3] To, že klasifikaci provádíme s pevným modelem je jen záležitost zvyku nebo napodobení klasické klasifikace. Ve skutečnosti nám nic nebrání pokračovat v odhadu modelu a tak jej dále upřesňovat. Toto další odhadování je již odhadem bez učitele. Správné hodnoty tříd již neznáme.

Poznámka: Počáteční odhad modelu s učitelem je velmi významná metoda pro inicializaci standardního odhadu modelu směsi komponent. Její motivace je zakotvena právě ve směsové logistické regresi.

Poznámky

Program je realizován pomocí funkcí lrLearn(yL,xL) - učení a lrTest(xT,Est) - testování. Pokud bychom chtěli vidět skutečné algoritmy, které úlohu realizují, museli bychom se podívat do těchto funkcí. Ty jsou v adresáři _func.

Program

Popis programu

Kód programu

```
// Mixture used as Logistic Regression
// - learning (estimation with known pointer)
// - testing (only pointer estimation = classification)
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nL=1000;                      // number of data for learning
nT=150;                       // number of data for testing

// Simulation --------------------------
nd=nT+nL;                     // length of learning + testing
x=rand(nd,2,'n');             // regression vectors
x1=[x ones(nd,1)];            //  -"- extended by 1
ys=x1*[1,1,2]'+.001*rand(nd,1,'n'); // continuous output
y=zeros(nd,1);                // discretization
s1=find(ys<0);                y(s1)=1;
s2=find((ys>=0) & (ys<3));    y(s2)=2;
s3=find(ys>=3);               y(s3)=3;

yL=y(1:nL); xL=x(1:nL,:);     // data for learning
// Learning ----------------------------
Est=lrLearn(yL,xL);

tT=(1:nT)+nL;                 // times for testing
xT=x(tT,:);                   // reg.vecs. for testing
// Testing -----------------------------
yT=lrTest(xT,Est);            // testing
yR=y(tT)';                    // estimates of tested output

// Results -----------------------------
s=1:nT;
plot(s,yR,'bo',s,yT,'rx','markersize',8)
set(gcf(),'position',[700 100 600 400])
title 'True and estimated pointer values'
```

```
printf('Wrong %d from %d\n',sum(yR~=yT),nT)
```

Odhad směsi s dynamickým ukazovátkem a diskrétními komponentami

- diskrétní výstup - řízená koruna s pamětí $f\left(y_t|u_t, y_{t-1}\right)$
- simulovaná data
- inicializace odhadu - zašuměné parametry ze simulace
- model ukazovátka $\quad f\left(c_t|c_{t-1}, \alpha\right) = \alpha_{c_t|c_{t-1}}$

Simulují se data ze směsi kategorických komponent a dynamického ukazovátka:

- komponenta i pro $i = 1, 2, \cdots, n_c$

u_t, y_{t-1}	$y_t = 1$	$y_t = 2$
1, 1	$\Theta^i_{1\|11}$	$\Theta^i_{2\|11}$
1, 2	$\Theta^i_{1\|12}$	$\Theta^i_{2\|12}$
2, 1	$\Theta^i_{1\|21}$	$\Theta^i_{2\|21}$
2, 2	$\Theta^i_{1\|22}$	$\Theta^i_{2\|22}$

(0.1)

kde y_t a u_t jsou výstup a vstup v čase t,

- ukazovátko

	$c_t = 1$	$c_t = 2$	$\cdots$	$c_t = n_c$
$c_{t-1} = 1$	$\alpha_{1\|1}$	$\alpha_{2\|1}$		$\alpha_{n_c 1}$
$c_{t-1} = 2$	$\alpha_{1\|2}$	$\alpha_{2\|2}$		$\alpha_{n_c 2}$
$\cdots$			$\cdots$	
$c_{t-1} = n_c$	$\alpha_{1\|n_c}$	$\alpha_{2\|n_c}$		$\alpha_{n_c n_c}$

n_c je počet komponent.

Předpoklady:

Sci značení:

Úloha: Simulace a odhad dynamickým modelem směsi diskrétních komponent - základní konfigurace pro odhad směsi s diskrétními komponentami.

Poznámky

1. *Dynamické ukazovátko umožňuje predikci aktivní komponenty, protože dává do souvislosti po sobě jdoucí aktivity komponent.*

2. *Inicializace odhadu směsi komponent je velice důležitá. Bez dobré inicializace může být konvergence v odhadu pomalá nebo odhad selže.*

   Vzorec pro váhy komponent, které přiřazují komponentám pravděpodobnosti aktivity, se sestává ze tří částí. Jsou to (i) vektor "vzdáleností" aktuálně změřeného datového vzorku od jednotlivých komponent, (ii) model ukazovátka a (iii) minulý váhový vektor w_{t-1}. Druhá a

třetí položka jsou stejné, jako u směsi spojitých komponent. Vzdálenost změřeného datového vzorku od komponenty je dán hodnotou modelu s dosazeným aktuálním odhadem parametrů a hodnotami změřených dat. Model v diskrétním případě je dán tabulkou (0.1). Prvky této tabulky jsou bodové odhady parametrů a změřená data y_t , u_t a y_{t-1} určují jeden prvek této tabulky. A hodnota tohoto prvku určuje zjišťovanou "vzdálenost".

Z uvedeného plyne, že komponenty , které mají některý prvek stejný, mají od příslušného datového vzorku stejnou vzdálenost.

Zde jako počáteční hodnoty parametrů pro odhad použijeme skutečné hodnoty parametrů, a ty více nebo méně zašumíme.

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné komponenty.
2. Odhad se provádí podle standardních vzorců:
   (a) výpočet vah W_t a w_t pro změřený výstup y_t.
   (b) Přepočet statistik komponent a ukazovátka.
   (c) Konstrukce bodových odhadů parametrů.
3. Jako výsledek se ukazují hodnoty odhadovaného ukazovátka ve srovnání se simulovaným. Tím úloha získává také charakter klasifikace.

Kód programu

```
// Mixture estimation - static components and dynamic pointer model
// - simulated two-dimensional data
// - initialization by parameters from simulation + noise
// - dynamic components
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion
rand('seed',0)

nd=100;             // number of data
nc=3;               // number of componentd
I_estCov=0;         // estimation of noise covariances ? 0|1 no|yes

// simulated reg.coef.
Sim.nc=nc;
Sim.Cy(1).th=[0.9 -.5]';
Sim.Cy(2).th=[0.4 0.6]';
Sim.Cy(3).th=[-.7 -.5]';
// noise covariances
```

```
r=.1;                              // amplitude of noise covariances
Sim.Cy(1).sd=r*[1 0;0,1];
Sim.Cy(2).sd=r*[1 0;0,1];
Sim.Cy(3).sd=r*[1 0;0,1];
// simulated noise covariances
Sim.Cp.th=fnorm(rand(nc,nc,'u')+.1,2);

Sim.ct(1)=1;                       // initial poiner

// initial parameters
a=.5;   // std of scattering init.params from simulated ones
for j=1:nc                         //  from those used in simulation
  [mr,mc]=size(Sim.Cy(j).th);
  Ps=[Sim.Cy(j).th;1]+[a*rand(mr,mc,'n');0]; // initial parameters
  Est.Cy(j).V=Ps*Ps';                       // statistics
  Est.Cy(j).th=Sim.Cy(j).th+a*rand(2,1,'n'); // pt.est. of reg.coef
  Est.Cy(j).sd=.1*eye(2,2);                 // standard deviation
end
Est.ka=ones(1,nc);                          // counter
Est.Cp.V=.1*ones(nc,nc);                    // pointer statistics
Est.Cp.th=fnorm(rand(nc,nc,'u')+.1);        // pointer parameter
w=fnorm(ones(1,nc));                        // weights

// SIMULATION ============================================================
for t=2:nd
  i=Sim.ct(t-1);    // last active component
  Sim.ct(t)=sum(rand(1,1,'u')>cumsum(Sim.Cp.th(i,:)))+1;    // pointer
  j=Sim.ct(t);      // active component
  y=Sim.Cy(j).th+Sim.Cy(j).sd*rand(2,1,'norm');             // output
  Sim.yt(:,t)=y;                                            // stor
end

// ESTIMATION ============================================================
printf(' ')
for t=2:nd
  if t/10==fix(t/10), printf('.'); end
  for j=1:nc
    [xxx,G(j)]=GaussN(Sim.yt(:,t),Est.Cy(j).th,Est.Cy(j).sd); // likelihood
  end
  Lq=G-max(G);                // rough normlization
  q=exp(Lq);                  // exponent

  ww=(q*w)'.*Est.Cp.th;       // matrix weights
  W=ww/sum(ww);               // normalization    - f(c(t),c(t-1)|d(t))
  w=sum(W,1);                 // marginalization  - f(c(t)|d(t))
  wt(:,t)=w';                 // stor

  // Update of statistic
  Est.ka=Est.ka+w;                          // counter
```

```
  Est.Cp.V=Est.Cp.V+W;                        // ptr.stat. update
  Ps=[Sim.yt(:,t)' 1];                        // extended reg.vec.
  for i=1:nc
    Est.Cy(i).V=Est.Cy(i).V+w(i)*Ps'*Ps;     // information matrix

    //nove rozdeleni informacni matice V
    Vyy=Est.Cy(i).V(1:2,1:2);                // part Vyy - psi.psi'
    Vy=Est.Cy(i).V($,1:2);                   // part Vy  - psi.y
    V1=Est.Cy(i).V($,$);                     // part V1  - y.y
    Est.Cy(i).th=inv(V1+1e-8*eye(V1))*Vy;    // pt.est. - reg.coef.
    Est.Cy(i).tht(:,t)=Est.Cy(i).th(:);      // pt.est. - covar.
    if I_estCov~=0
      // pt.est. of noise covariance - used or not
      Est.Cy(i).cv=(Vyy-Vy'*inv(V1+1e-8*eye(V1))*Vy)/Est.ka(i);
    end
  end
  Est.Cp.th=fnorm(Est.Cp.V,2);     // pt.est. of pointer parameter
  [ss,Est.ct(1,t)]=max(w);         // store pointer values
end

// Results
disp(Sim.Cp.th,'pt.pars_sim')
disp(Est.Cp.th,'pt.pars_est')

s=2:nd;
wr=sum(Sim.ct(s)~=Est.ct(s)');
printf('\n Wrong classifications %d from %d\n',wr,length(s))

set(scf(1),'position',[50 50 400 400])
plot(s,Sim.ct(s),'bo',s,Est.ct(s),'rx')
title 'Pointer values and their predictions'

set(scf(2),'position',[550 50 400 600])
for i=1:nc
  subplot(nc,1,i)
  plot(Est.Cy(i).tht')
  title('Evolution of parameter estimates '+string(i))
end
```

Odhad směsi s dynamickým ukazovátkem a stavovými komponentami

- skalární výstup, dvourozměrný vstup a stav
- simulovaná data
- kovarianční matice stavového modelu nastaveny podle simulace
- dynamický model ukazovátka

Simulují se data ze směsi stavových komponent a dynamického ukazovátka:
- komponenta i pro $i = 1, 2, \cdots, n_c$

$$\begin{aligned} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t &= M^i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{t-1} + N^i \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + F^i + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t \\ y_t &= A^i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + B^i \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + G^i + v_t \end{aligned}$$

kde M^i, N^i, F^i, A^i, B^i a G^i jsou matice stavového modelu i-té komponenty
- ukazovátko

	$c_t = 1$	$c_t = 2$	$\cdots$	$c_t = n_c$
$c_{t-1} = 1$	$\alpha_{1\|1}$	$\alpha_{2\|1}$		$\alpha_{n_c 1}$
$c_{t-1} = 2$	$\alpha_{1\|2}$	$\alpha_{2\|2}$		$\alpha_{n_c 2}$
$\cdots$			$\cdots$	
$c_{t-1} = n_c$	$\alpha_{1\|n_c}$	$\alpha_{2\|n_c}$		$\alpha_{n_c n_c}$

n_c je počet komponent.

Předpoklady:

Sci značení:

Úloha: Simulace a odhad dynamickým modelem směsi stavových komponent - základní konfigurace pro odhad směsi se stavovými komponentami.

Poznámky

1. *Dynamické ukazovátko umožňuje predikci aktivní komponenty, protože dává do souvislosti po sobě jdoucí aktivity komponent.*
2. *Vzorec pro váhy komponent, které přiřazují komponentám pravděpodobnosti aktivity, se předchozích typech směsí sestával ze tří částí. Jsou to (i) vektor "vzdáleností" aktuálně změřeného datového vzorku od jednotlivých komponent, (ii) model ukazovátka a (iii) minulý váhový vektor w_{t-1}. Druhá a třetí položka jsou stejné, jako u směsi spojitých komponent. Vzdálenost změřeného datového vzorku od komponenty se počítá opět jako hodnota datové predikce s touto komponentou s dosazenými aktuálními daty a odhadem stavu. Tato predikce je*

$$f\left(y_t|x_{t-1}\right) = \int_{x_t^*} f\left(y_t, x_t|x_{t-1}\right) dx_t = \int_{x_t^*} f\left(y_t|x_t\right) f\left(x_t|x_{t-1}\right) dx_t.$$

Tuto hodnotu pro každou komponentu dostaneme přímo z Kalmanova filtru.

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné komponenty.
2. Odhad se provádí podle standardních vzorců:
   (a) přepočet Kalmanových filtrů pro každou komponentu.
   (b) výpočet vah W_t a w_t pro změřený výstup y_t.
   (c) spojení výsledků pro jednotlivé komponenty do jednoho modelu.
3. Jako výsledek se ukazují hodnoty odhadovaného ukazovátka ve srovnání se simulovaným, simulované a odhadnuté datová klastry a vývoj datové predikce.

Kód programu

```
// Mixture estimation with state-space components
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(1)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nd=150;                        // number of data
nc=3;                          // number of components

// init of simulatiom
// component 1
Sim.Cy(1).M=[.3 -.3;           // reg.coef for simulation
             .5 .1];
Sim.Cy(1).N=[.2 -.3
             .3  .5];
Sim.Cy(1).A=[.6 .4];
Sim.Cy(1).B=[0.3 0.7];
Sim.Cy(1).F=[-3 -7]';
// component 2
Sim.Cy(2).M=[.05 -.3;
             .4 -.1];
Sim.Cy(2).N=[.2 -.3
             .3  .5];
Sim.Cy(2).A=[.4 0.8];
Sim.Cy(2).B=[-0.4 -0.5];
Sim.Cy(2).F=[0 0]';
// component 3
Sim.Cy(3).M=[.6 -.1;
             .8 -.5];
Sim.Cy(3).N=[.2 -.3
```

```
            .3  .5];
Sim.Cy(3).A=[1 -.4];
Sim.Cy(3).B=[0.4 0.5];
Sim.Cy(3).F=[3 .4]';

Sim.rv=1e-3;                  // common coves of output model
Sim.rw=1e-3;                  // common coves of state model

ct=zeros(1,nd);
ct(1)=1;                       // pointer initial condition

Sim.Cp.th=[1 0 0;
           0 1 0;
           0 0 1]+1;
Sim.Cp.th=fnorm(Sim.Cp.th,2); // pointer model parameter

nx=max(size(Sim.Cy(1).M));
[ny,nu]=size(Sim.Cy(1).B);
ut=rand(nu,nd,'n');
yt=zeros(ny,nd);
xt=zeros(nx,nd);
x=zeros(nx,1);


// SIMULATION ============================================================
for t=2:nd
  ct(t)=sum(rand(1,1,'u')>cumsum(Sim.Cp.th(ct(t-1),:)))+1;
  j=ct(t);    // index of active component
  u=ut(:,t);
  x=Sim.Cy(j).M*x+Sim.Cy(j).N*u+Sim.Cy(j).F+sqrt(Sim.rw)*rand(nx,1,'n');
  y=Sim.Cy(j).A*x+Sim.Cy(j).B*u+sqrt(Sim.rv)*rand(ny,1,'n');
  yt(:,t)=y; xt(:,t)=x;
end
Sim.xt=xt;
Sim.yt=yt;
Sim.ut=ut;

// init of estimation
Est.Cp.V=1e-1*ones(nc,nc);         // pointer statistics
Est.Cp.th=fnorm(Est.Cp.V,2);       // pointer parameter
w1=zeros(nc,1); w1(1)=1;           // initial ptr.value
sx=zeros(nx,1);                    // initial state
rx=1e3*eye(nx,nx);                 // initial covariance matrix
                                   //   of the state estimate
//ESTIMATION ============================================================
printf(' '), tt=fix(nd/10);
for t=(2:nd)
  if t/tt==fix(t/tt), printf('.'); end
  for j=1:nc
```

```
    [Est.Cy(j).x,Est.Cy(j).rx,yp,ry]=..      // Kalman filter
    KalmanXY(sx,yt(:,t),ut(:,t),Sim.Cy(j).M,Sim.Cy(j).N,..
             Sim.Cy(j).A,Sim.Cy(j).B,Sim.Cy(j).F,Sim.rw,Sim.rv,rx);
    [xxx,Lq(j)]=GaussN(yt(:,t),yp,ry);       // likelihood
    Est.Cy(j).yp(t)=yp;
  end
  Lqq=Lq-max(Lq);                            // rough normalization
  q=exp(Lqq);                                // exponent from logarithm

  Wp=(w1*q').*Est.Cp.th; W=Wp/sum(Wp);       // matrix weight for pointer
  wp=sum(W,'r')'; w=wp/sum(wp);              // weights for components

  // updated components are merged with the weights w
  sx=0; s1=0; s2=0;
  for i=1:nc              // KL approximation of expectation
    sx=sx+w(i)*Est.Cy(i).x;
  end
  xe(:,t)=sx;

  for i=1:nc              // KL approximation of covariance
    s1=s1+w(i)*(Est.Cy(i).x-sx)*(Est.Cy(i).x-sx)';
    s2=s2+w(i)*Est.Cy(i).rx;
  end
  rx=s1+s2;

  Est.Cp.V=Est.Cp.V+W;
  Est.Cp.th=fnorm(nu,2);
  w1=w;
  wt(:,t)=w;
end
Est.xt=xe;
Est.wt=wt;
Est.rx=rx;

// RESULTS
s=15:nd;
[xxx ce]=max(wt,'r'); wr=sum(ct(s)~=ce(s));
printf('\nWrong classifications %d from %d\n',wr,length(s))

set(scf(1),'position',[70 70 500 500])
plot(1:nd,ct,'o',1:nd,ce,'.')
title('Simulated and estimated pointer values')

set(scf(2),'position',[700 70 500 500])
plot(xt(1,s),xt(2,s),'bx','markersize',10)
plot(xe(1,s),xe(2,s),'ro')
title('Simulated and estimated clusters')

col=['b','r','g','k','m'];
```

```
set(scf(3),'position',[300 270 500 500])
title('Output predictions')
for i=1:nc
  plot(Est.Cy(i).yp,col(i))
end
legend('first','second','third');
```

Odhad směsi s dynamickým ukazovátkem a statickými komponentami[4]

- smíšený (spojitý i diskrétní) jednorozměrný výstup, bez řízení
- simulovaná data
- inicializace odhadu - datový vzorek klasifikovaný expertem
- standardní odhad / odhad s pevnými kovariancemi šumů komponent
- značení:
  - y_t spojitý výstup
  - z_t diskrétní výstup
  - c_t pointer
- modely:
  - model spojité komponenty

$$f\left(y_t|y_{t-1}, z_t, z_{t-1}, \Theta_c\right)$$

  - model diskrétní komponenty

$$f\left(z_t|z_{t-1}, \beta\right) = \beta_{z_t|z_{t-1}}$$

  - model ukazovátka

$$f\left(c_t|c_{t-1}, z_{t-1}, \alpha\right) = \alpha_{c_t|c_{t-1};z_{t-1}}$$

Předpoklady: diskrétní modely závisí jen na diskrétních veličinách.

Sci značení: standard podle dohody o značení.

Úloha: Simulace, odhad a predikce s dynamickým modelem směsi statických komponent. Úloha patří do nadstavby.

Poznámky

Program

Popis programu

1. Simulace hodnot ukazovátka a podle toho generování dat z příslušné spojité a diskrétní komponenty.
2. Inicializace odhadovacího algoritmu se provádí s množinou apriorních dat. Tato množina se sestává z datových záznamů (buď předem změřených nebo uměle zkonstruovaných tak, aby dobře reprezentovali datový prostor). K těmto datovým záznamům jsou expertně přiřazeny hodnoty ukazovátka (třídy, do kterých patří). Je několik možností, jak tuto před-klasifikaci získat:

[4] Tato úloha je zobecněním jak spojitých tak i diskrétních směsi. Patří už mezi nadstandardně složité záležitosti

(a) pro získání dat použijeme speciální měření (např. délky kolon jsou zaznamenávány studenty),

(b) expert sleduje měření a určuje, ko které kategorie patří (např. stupeň dopravy)

(c) expert sestavuje datové záznamy uměle, tak aby odpovídaly jeho představám o fungování systému (např. tato zatáčka (s parametry ...) patří do kategorie velmi nebezpečných, a tahle (s parametry ...) je bez nebezpečí - přitom zatáčky nemusí vůbec existovat)

Tato inicializace je velmi účinná a ve standardním případě stačí nějakých 5 dat do každé komponent a algoritmus je velmi dobře připraven

3. Odhad se provádí podle modifikovaných vzorců ze spojitého a diskrétního odhadu:

   (a) výpočet vah W_t a w_t pro změřený výstup y_t.

   (b) Přepočet statistik komponent a ukazovátka.

   (c) Konstrukce bodových odhadů parametrů. Tady je možnost volby:

      i. průběžné odhadování kovariančních matic komponent,

      ii. použití počátečních kovariančních matic bez průběžného přepočtu.
      Tato varianta je bezpečnější. Při průběžném odhadu se může stát, že jedna komponenta překryje ostatní a výsledek je daný právě jen touto komponentou.
      Možná je také varianta, kdy v prvé části odhadování ponecháme kovarianční matice pevné, a v další části je již odhadujeme.

4. Jako výsledek se ukazují hodnoty odhadovaného ukazovátka ve srovnání se simulovaným. Tím úloha získává charakter klasifikace.

Kód programu

```
// Estimation of mixture with DATA DEPENDENT DYNAMIC POINTER
// Models:
// Cy - continuous componen     f( yt(t) | yt(t-1),zt(t),zt(t-1),theta )
// Cz - discrete component      f( zt(t) | zt(t-1),be )
// Cp - discrete pointer model  f( ct(t) | ct(t-1),zt(t-1),als )
//      ct=1,2,3;  zt=1,2;
// - mixed (discrete and continuous) modeled data
// - pre-classified data sample for initialization
[u,t,n]=file();                         // find working directory
chdir(dirname(n(1)));                   // set working directory
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion

nd=150;                       // number of data
ni=20;                        // number of initial data
nc=3;                         // number of components

// parameters for continuous components
Sim.Cy(1).th=[.95 1 1 5];
```

```
Sim.Cy(2).th=[.6 -1 1 0];
Sim.Cy(3).th=[.81 1 -1 -5];
// common variance for continuous components
r=.1;

// parameters for discrete components
Sim.Cz(1).th=fnorm([1 0
                   0 1]+1,2);
Sim.Cz(2).th=fnorm([0 1
                   1 0]+2,2);
Sim.Cz(3).th=fnorm([1 0
                   0 1]+3,2);

// parameters for pointer model
Sim.Ca(1).th=fnorm([0 1 0
                   0 0 1
                   1 0 0]+5,2);
Sim.Ca(2).th=fnorm([0 0 1
                   1 0 0
                   0 1 0]+8,2);

// initial conditions
yt(1)=0; zt(1)=1; ct(1)=1;

// SIMULATION ----------------------------------------------------------
for t=2:nd
  als=Sim.Ca(zt(t-1)).th;          // parameter of Cp
  ct(t)=sum(rand(1,1,'unif')>cumsum(als(ct(t-1),:)))+1; // pointer value

  be=Sim.Cz(ct(t)).th;             // parameter of Cz
  zt(t)=sum(rand(1,1,'unif')>cumsum(be(zt(t-1),:)))+1;  // disc. output

  ps=[yt(t-1) zt(t) zt(t-1) 1];    // regression vector of Cy
  th=Sim.Cy(ct(t)).th;             // parameters of Cy
  yt(t)=th*ps'+sqrt(r)*rand(1,1,'norm');               // cont. output
end // ---------------------------------------------------------------

ky=min(size(yt));    // dimension of cont. output
mz=max(zt);          // number of values of disc. output

// Initial statistics and par. estimates
for j=1:nc
  // model Cy
  Est.Cy(j).V=zeros(5,5);
  Est.Cy(j).cv=r;
  // model Cz
  Est.Cz(j).V=rand(mz,mz,'u')+.1*ones(mz,mz);
  Est.Cz(j).th=fnorm(Est.Cz(j).V);
end
```

```
// initial estimates of th = estimation with known components !!!
// (for initiation, a sample of pre-classified data is used)
for t=2:ni
  Ps=[yt(t) yt(t-1) zt(t) zt(t-1) 1]';
  Est.Cy(ct(t)).V=Est.Cy(ct(t)).V+Ps*Ps';
  Vy=Est.Cy(ct(t)).V(1,1);               // part Vy      yt.yt
  Vyps=Est.Cy(ct(t)).V(2:$,1);           // part Vyps    psi.yt
  Vps=Est.Cy(ct(t)).V(2:$,2:$);          // part Vps     psi.psi'
  Est.Cy(ct(t)).th=inv(Vps+1e-5*eye(Vps))*Vyps;  // pt.est. of reg.coef.
end

// poiner
Est.ka=5*ones(1,nc);
for j=1:mz
  // statistics for pointer
  Est.Cp(j).V=rand(nc,nc,'u')+.1*ones(nc,nc);
  // parameters of pointer model
  Est.Cp(j).th=fnorm(Est.Cp(j).V,2);
end
// weighting vector
w=fnorm(rand(1,nc,'u')+ones(1,nc));

// TIME LOOP ============================================================
printf(' '), tt=fix(nd/10);
for t=(2:nd)
  if t/tt==fix(t/tt), printf('.'); end
  // computation of likelihoods
  ps=[yt(t-1) zt(t) zt(t-1) 1]';        // regression vector
  for j=1:nc                            // likelihood for yt
    yp=Est.Cy(j).th'*ps;
    rh=Est.Cy(j).cv;
    [xxx Lm(j)]=GaussN(yt(t),yp,rh);
    Zp(j)=Est.Cz(j).th(zt(t-1),zt(t));   // prediction for zt
  end
  Lm=Lm-max(Lm);
  Yp=exp(Lm);

  // computation of weights W and w
  al=Est.Cp(zt(t-1)).th;
  Wp=((Yp.*Zp)*w)'.*al;
  if sum(Wp)<1e-6, Wp=rand(nc,nc,'u'); end
  W=Wp/sum(Wp);                         // matrix W
  w=sum(W,'r');                         // weighting vector w
  wt(:,t)=w';
  [xxx ce(t)]=max(w);

  // update of statistics
  Ps=[yt(t) yt(t-1) zt(t) zt(t-1) 1]';      // extended regression vector
```

```
  for j=1:nc                              // update of Cy.V, w, Cz.V
    Est.Cy(j).V=Est.Cy(j).V+w(j)*Ps*Ps';
    Est.ka(j)=Est.ka(j)+w(j);
    Est.Cz(j).V(zt(t-1),zt(t))=Est.Cz(j).V(zt(t-1),zt(t))+w(j);
  end
  Est.Cp(zt(t-1)).V= Est.Cp(zt(t-1)).V+W; // update of Ca

  // computation of point estimates
  for j=1:nc
    Vy=Est.Cy(j).V(1,1);                  // part Vy
    Vyps=Est.Cy(j).V(2:$,1);              // part Vyps
    Vps=Est.Cy(j).V(2:$,2:$);             // part Vps
    Est.Cy(j).th=inv(Vps+1e-5*eye(Vps))*Vyps;      //regression coefs
    th(j).t(:,t)=Est.Cy(j).th;
    // how about estimation of coveriances:
    // for 0 - fix variance, for 1 - variances estimation
    if 1
      Est.Cy(j).r=(Vy-Vyps'*inv(Vps+1e-5*eye(Vps))*Vyps)/Est.ka(j);
     end
    Est.Cz(j).th=fnorm(Est.Cz(j).V,2);            // parameters for Cz
  end
  Est.Cp(zt(t-1)).th=fnorm(Est.Cp(zt(t-1)).V,2);  // pointer pt.est.
end // END OF TIME LOOP ============================================

// Results
s=2:nd;
wr=sum(ct(s)~=ce(s));
printf('\nWrong classifications %d from %d\n',wr,length(s))

set(scf(1),'position',[600 50 600 600])
plot(1:nd,ct','x',1:nd,ce','.r')
set(gca(),'data_bounds',[1 nd .8 nc+.2])
legend('simulation','estimation'); // 5 means place legend hy hand
xlabel('Time [periods]')
ylabel('Pointer values')
title('The pointer estimation')

set(scf(4),'position',[150 50 400 500])
for i=1:nc
  subplot(nc,1,i)
  plot(th(i).t')
  xlabel('Time [periods]')
  ylabel('Parameter values')
  title('Evolution of parameter estimation of the  '+..
          string(i)+'. normal component')
end
```