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

```
// P73Mix1KF.sce
// Mixture estimation with state-space components
//
[u,t,n]=file();                          // find working directory
chdir(dirname(n(2)));                    // 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');
```