

# Odhad směsi s datově závislým dynamickým ukazovátkem a statickými komponentami<sup>1</sup>

- 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

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

- model diskrétní komponenty

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

- model ukazovátka

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

**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 spojitě 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:

---

<sup>1</sup>Tato úloha je zobecněním jak spojitých tak i diskrétních směsí. 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, do 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) Přepočítání 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 první části odhadování ponecháme kovarianční matice pevné, a v další části je již odhadujeme.

### 4. Jako výsledek se ukazují hodnoty odhadovaného ukazovátko 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

// pointer
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 coeffs
    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

```