Předpověď s modelem směsi komponent

- dvourozměrný výstup, bez řízení
- simulovaná data
- model se známými parametry
- 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,

$\theta_i=\left[\begin{array}{c}(\theta_1)_i\\(\theta_2)_i\end{array}\right]$ 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, prvky vektoru "vzdáleností" y_t od jednotlivých komponent - vektoru m 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.

další kroky

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

Program

Popis programu

Kód programu

```
// Test predikce dat se směsí se STATICKÝMI komponentami
[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;
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
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 (polo-update a do-update)
  wP=fnorm(alS'*w1);          // váhy odpovídající času t bez využití y(t)
  wp=(alS')^np*wP;            // predikce vah np kroků dopředu
  w=fnorm(wP.*m);             // do-update w s aktuálním y(t)

  // 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)
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])
subplot(211)
[f1,n1]=nan_hist(y,150,'plot');
subplot(212)
[f2,n2]=nan_hist(ytp,150,'plot');

s=551:600;
scf(2);
set(gcf(),'figure_position',[200,150])
plot(s,y(s),'ob')
plot(s,ytp(s),'xr')
```