Odhad statické směsi s log-normálními komponentami

Nezáporná data se v praxi vyskytují velice často. V dopravě jsou to např. hodnoty rychlosti automobilů, intenzity dopravního proudu, obsazenosti na detektorech, jakýkoli druh podílu apod. Nezápornost těchto dat je třeba při práci s nimi respektovat. Pokud se nerespektuje, můžeme při výpočtech dostávat nesmysly a tak zcela znehodnotit výsledky.

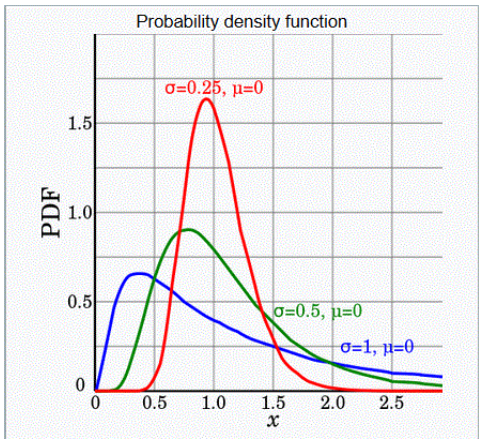
Velice důležitým rozdělením, které respektuje nezápornost dat je **log-normální rozdělení**, které má hustotu pravděpodobnosti

$$f(x) = \frac{1}{\sqrt{2\pi r}} \frac{1}{x} \exp\left\{-\frac{1}{2r}(\ln x - \mu)^2\right\}$$

se střední hodnotou $E[x] = \exp\left\{\mu + \frac{r}{2}\right\}$

a rozptylem $D[x] = [\exp\{r\} - 1]\exp\{2\mu + r\}$.

Jeho hustota pravděpodobnosti pro stejné μ a různé r je na obrázku



Co je ale pro nás nejdůležitější, je to, že toto rozdělení dostaneme jako transformaci normálního rozdělení $N_y(\mu, r)$ když položíme $y = \exp\{x\}$. A naopak, vezmeme-li log-normální rozdělení $LN_x(\mu, r)$ a položíme $x = \ln(y)$, dostaneme x jako normální rozdělení $N(\mu, r)$.

To nám umožní následující práci s nezápornými daty:

1. Předpokládáme, že data x mají log-normální rozděleni.
2. Vytvoříme nová (transformovaná) data $y = \ln\{x\}$
   Poznámka: *Všimněme si, že i když byla původní data nezáporná, transformovaná data jsou rozložena na celé reálné ose.*
3. Dále pracujeme s transformovanými daty, u kterých teď předpokládáme normální rozdělení.
   Poznámka: *Pojem pracujeme je zcela obecný. Tady máme na mysli odhad modelu směsi distribucí.*

4. Výsledky, které jsou nyní pro transformovaná (normální) data, převedeme zpět do původní podoby (nezáporná data, pomocí transformace $x = \exp\{y\}$.
   Poznámka: To platí pro lineární charakteristiky, jako je třeba střední hodnota, tj. centra komponent.

Program

Popis následuje za programem

```
// P75dMixLogNorm.sce
// Mixture estimation - static components and pointer model
// Log data - estimate as normal - exponentialize results
// - real data
// - initialization ???
[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
//rand('seed',0)

nd=1350;            // number of data
nc=10;              // number of component
ni=10;              // number of initial data
ch=[1:12];          // variables (max 12)
I_estCov=0;         // estimation of noise covariances ? 0|1 no|yes


// DATA LOAD ============================================================
load _data/d_con.dat d_con;
//yt=scal(d_con(1:nd,ch)');
yt=d_con(1:nd,ch)';
nv=length(ch);

yn=log(yt+1e-15);                  // transformation to normal variables

// initial parameters
select 1
case 1
  a=ones(nv,1);                    // std of scattering initial parametrs
  for j=1:nc                       //  from those used in simulation
    Est.Cy(j).V=zeros(nv+1,nv+1);
    for i=1:ni
      Ps=[yn(:,i);1]+[a.*rand(nv,1,'n');0];  // initial parameters
      Est.Cy(j).V=Est.Cy(j).V+Ps*Ps';        // statistics
    end
    Vyy=Est.Cy(j).V(1:nv,1:nv);              // part Vyy - y.
    Vy=Est.Cy(j).V($,1:nv);                  // part Vy  - psi.y
    V1=Est.Cy(j).V($,$);                     // part V1  - psi.psi'
    Est.Cy(j).th=inv(V1+1e-8*eye(V1))*Vy;    // pt.est. - reg.coef.
    Est.Cy(j).sd=.1*eye(nv,nv);              // standard deviation
```

```
  end
case 2
  yi=yn(:,1:ni);
  my=mean(yi,2);                              // mean
  sy=stdev(yi,2);                             // standeard deviation
  for i=1:nc
    yy=my+1.4*sy.*rand(nv,1,'n');             // mean data point \pm stdev
    Ps=[yy;1];                                // regression vector
    V=Ps*Ps';                                 // inf. matrix
    Est.Cy(i).th=yy;                          // ini. parameters
    Est.Cy(i).V=V;                            // ini. inf. matrix
    Est.Cy(i).sd=.1*eye(nv,nv);               // ini. standard deviation
  end
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

if 0           // DISPLAY INITIAL CLUSTERS
  i1=1; i2=2;  // marginal to be displayed
  scf(100);
  tx=['b.';'r.';'g.';'m.';'k.';'y.';'b.';'r.';'g.';'m.';'k.';'y.']; // colors for graph
  plot(yn(i1,:),yn(i2,:),'c.','markersize',4)
  for i=1:nc
    plot(Est.Cy(i).th(1),Est.Cy(i).th(2),tx(i),'markersize',12)
  end
  pause
end

// ESTIMATION ===========================================================
printf(' ')
kk=ceil(nd/10); printf('\n  2 4 6 8 |\n ')
for t=1:nd     // TIME LOOP ------------------------------------------
if t/kk==fix(t/kk), printf('.'); end

  // Proximity
  for j=1:nc
    [xxx,G(j)]=GaussN(yn(:,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';

  Ps=[yn(:,t)' 1];                            // extended reg.vec.
  for i=1:nc
```

```
    // update of statistic
    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

    // point estimates of parameters
    Vyy=Est.Cy(i).V(1:nv,1:nv);              // part Vyy - y.y'
    Vy=Est.Cy(i).V($,1:nv);                  // part Vy  - psi.y
    V1=Est.Cy(i).V($,$);                     // part V1  - psi.psi'
    Est.Cy(i).th=inv(V1+1e-8*eye(V1))*Vy;    // pt.est. - reg.coef.
    Est.Cy(i).tht(:,t)=Est.Cy(i).th';        // store
    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);               // est. of pointer parameter
  [ss,cE(t)]=max(w);                         // store
end

// RESULTS
// histogram of poinzer
disp(vals(cE),' Frequencies of values of estimated pointer')

// weights of components
set(scf(1),'position',[50 450 750 200]);
plot(wt')
title 'Weights of the pointer'

// marginals of estimated clusters
i1=1; i2=2;    // VARIABLES OF DISPLAYED MARGINAL
tx=['b.';'rx';'g+';'mo';'kp';'ys';'bx';'r+';'go';'mp';'ks';'y.']; // colors for graph
C=list();
for i=1:nc
  j=find(cE==i);
  C(i)=yt(:,j);
end
set(scf(2),'figure_position',[850 200]);
for i=1:nc
  if ~isempty(C(i))
  plot(C(i)(1,:),C(i)(2,:),tx(i),'markersize',3)
  end
end
title('Estimated clusters for variables '+string(i1)+' and '+string(i2))
```

Program sleduje odhad ODHAD SMĚSI NORMÁLNÍCH KOMPONENT - odhad statické směsi (z hlavního menu). Liší se jen v transformaci veličin (vstup a výstup) a inicializaci odhadu.

Inicializace

Jsou k dispozici dvě inicializační procedury. Obě jsou založeny na existenci apriorních dat (která jsou zde reprezentována počáteční částí zpracovávaných dat). Počet dat použitých pro inicializaci je označen ni.

1. - z apriorních dat se berou jednotlivé datové záznamy a z nich se konstruují zašuměné regresní vektory,
   - konstruuje se apriorní statistika V,
   - ze statistiky se počítají apriorní odhady parametrů.
2. - z apriorních dat se počítá střední hodnota a rozptyl,
   - generují se centra komponent (parametry) se spočtenou střední hodnotou a rozptylem,
   - nim se zároveň počítají počáteční statistiky (pro jedinou hodnotu dat).

Tato část je v programu označena jako "- initial parameters:".

V další části je možno volbou "if 1" prohlížet marginály dat a v nich apriorní klastry. Podle výsledku je možno ještě upravovat počáteční nastavení. Po volbě "if 0" se tato prohlížecí část programu vypne a provádí se kompletní odhad směsi.

Zpětná transformace dat ani center se v programu přímo neprovádí. Původní data zůstávají zachována. Při odhadu se počítají bodové odhady aktivních komponent (tj. která komponenta byla v daném čase aktivní). Tato aktivita se pak uplatňuje přímo na původních (nezáporných datech). Jsou vykreslována tato data a aktivita komponent je vyznačena barvami. To se odehrává v samém konci programu.