# Iterativní odhad směsi (jako EM algoritmus)

Odhad parametrů modelu směsi distribucí, v jeho on-line formě, tak, jak jej zde prezentujeme, běží v čase a jen jednou prochází datový vzorek, který je buď celý k dispozici a postupně se z něj odebírají data nebo se jeho položky teprve postupně měří. Jeho struktura je následující:

1. Změříme (vezmeme) nová data, "napasujeme" je na komponenty a získáme proximity. Po úpravě dostaneme váhy komponent.
2. S vahami přepočteme statistiky komponent a modelu ukazovátka.
3. Ze statistik určíme nové bodové odhady parametrů komponent a ukazovátka.

Celý odhad je komplikovaný, protože váhy komponent závisí na odhadech parametrů a naopak, odhady parametrů závisí na vahách komponent. Takže celý odhad není nepodobný tomu, když se pes honí za vlastním ocasem.

Zřejmě se nabízí i jiný postup odhadu (který používá slavný EM algoritmus). Vyjde se z apriorních odhadů parametrů komponent. Na tomto základě se určí váhy komponent pro všechna data spočtou se bodové odhady ukazovátka. Ty ale vedou na nové odhady parametrů a ty zase. určují jiné rozdělení aktivit komponent, tj. jiné váhy. Toto se opakuje, dokud se výpočty neustálí, tj. dokud dva po sobě jdoucí odhady v iteracích nedají stejné aktivní komponenty (stejné bodové odhady ukazovátka).

Iterativní odhad tedy běží podle následujícího schématu:

1. Provede se inicializace statistik modelů a jejich bodových odhadů, včetně apriorního nastavení ukazovátka pro všechna data.
2. Smyčka iterací
   (a) Pro existující bodové odhady parametrů komponent (a případně i modelu ukazovátka) se určí váhy komponent a jejich bodové odhady.
   (b) Pro nové bodové odhady ukazovátka (tj. aktivity komponent) se přepočtou odhady parametrů
   (c) Testuje se, zda se pohnuly bodové odhady ukazovátka. Jestliže ano, jde se na (a); jestliže ne, je konec.

Program pro tuto úlohu se simulovanými daty je následující (popis je za programem)

```
// Pokus o odhad směsi iterativní off-line metodou
// - pro dané hodnoty parametrů se spočtou všechny váhy
// - z vah se určí celé ukazovátko
// - pro něj se udělá nový odhad parametrů
// -- data se rozdělí podle komponent
// -- pro každou komponentu se spočte střední hiodnota
// - vypočte se kriterium (součet vah aktivních komponent) - není třeba
// - a to se opakuje
exec SCIHOME/ScIntro.sce, mode(0), //rand('seed',0);
```

```
nd=550;
// Simulace ----------------------------
thS=list();
thS(1)=[1 1]';
thS(2)=[2 5]';
thS(3)=[5 0]';
thS(4)=[8 3]';
thS(5)=[6 6]';
thS(6)=[4 3]';
nc=length(thS);
sdS=.6; cvS=[sdS**2 0;0 sdS**2];
alS=fnorm(1+ones(1,nc));
for t=1:nd
  cS(t)=sum(rand(1,1,'u')>cumsum(alS))+1;
  yt(:,t)=thS(cS(t))+sdS*rand(2,1,'n');
end // --------------------------------

// Inicializace
thI=list(); thE=thI;
for i=1:nc
  thI(i)=10*rand(2,1,'u');
end
thE=thI;
cv=.1*eye(2,2);

// Iterativní odhad
tht=[]; ct=[];
for x=1:50     // iterace
  for t=1:nd   // odhad pro všechna data
    for i=1:nc                          // proximity
      [xxx,G(i)]=GaussN(yt(:,t),thE(i),cv);
    end
    Lq=G-max(G);                        // rough normlization
    w=exp(Lq);                          // exponent
    w=w/sum(w);                         // neuvažuje se alfa

    wt(:,t)=w;                          // schovej váhy
  end
  [xxx,c]=max(wt,'r');                  // NOVÝ ODHAD c
  if prod(c==ct($,:)), break, end       // test na knec (změnu céček)
  for i=1:nc
    j(i).c=find(c==i);                  // komponenty
  end
  ct=[ct; c];                           // stará + nová  céčka

  // chci, aby v komponentách byly maximální pravděpodobnosti
  // tedy ne: .3 .3 .4  ale: 0 0 1
  kr=0;
  for i=1:nc
```

```
    kr=kr+sum(wt(i,j(i).c));            // kriterium
  end
  printf('iter = %d\n',x)                     // iter
  printf('krit = %g\n\n',kr)                  // krit

  for i=1:nc
    thE(i)=mean(yt(:,j(i).c),2);        // NOVÝ ODHAD KOMPONENT
  end                                   // pro stat. komponenty je střední
  clear wt;                             // hodnota průměr
end

// Výsledky
[q T]=c2c(cS,c);                        // přetočení komponent
wrong=sum(q(c)(:)~=cS(:))               // počet špatných proti simulaci
from=nd

set(scf(1),'position',[500 100 600 500]);    // data a centra komponent
plot(yt(1,:),yt(2,:),'.','markersize',3)
for i=1:nc
  plot(thE(i)(1),thE(i)(2),'.','markersize',8)
end
```

Popis programu

Simulují se dvourozměrná data obsahující 6 komponent (klastrů). Inicializace odhadu je dána náhodným přiřazením parametrů thl(i)=10*rand(2,1,'u'). V časové smyčce se nejdříve určí proximity a normalizují se na pravděpodobnostní váhy (vliv parametru alfa z modelu ukazovátka se zanedbává). Váhy se převedou na bodové odhady ukazovátka ve vektoru $c$ a do struktury $j(i).c$ se uloží časové indexy dat, kdy byla příslušná komponenta aktivní. Přepočet bodových odhadů parametrů komponent (jejich center) je velice jednoduchý. Pro každou komponentu je roven průměru dat, které komponentě patří (tedy byla s nim aktivní).

Tento odhad se opakuje ve smyčce iterací, s indexem cyklu $x$.

Abychom mohli sledovat vývoj odhadů v iteracích, zavedli jsme kriterium kvality odhadu. Lepší je odhad, kde pravděpodobnost aktivní komponenty je maximální (ideálně rovna jedné; ostatní váhy jsou pak rovny nule). Tedy kriteriem je součet vah aktivních komponent. V programu je kriterium označeno $kr$.

Stejný program, ale pro reálná data je tady

```
// Pokus o odhad směsi iterativní off-line metodou
// - pro dané hodnoty parametrů se spočtou všechny váhy
// - z vah se určí celé ukazovátko
// - pro něj se udělá nový odhad parametrů
// -- data se rozdělí podle komponent
// -- pro každou komponentu se spočte střední hiodnota
// - vypočte se kriterium (součet vah aktivních komponent) - není třeba
// - a to se opakuje
exec SCIHOME/ScIntro.sce, mode(0), rand('seed',0);
```

```
load _data/d_con.dat d_con;
yt=d_con';
[nv,nd]=size(yt);

nc=3;

// Inicializace
thI=list(); thE=thI;
for i=1:nc
  thI(i)=rand(nv,1,'u');
end
thE=thI;
cv=.1*eye(nv,nv);

tht=[]; ct=[];
for x=1:50     // iterace
  for t=1:nd   // off-line odhad
    for i=1:nc                      // proximity
      [xxx,G(i)]=GaussN(yt(:,t),thE(i),cv);
    end
    Lq=G-max(G);                    // rough normlization
    w=exp(Lq);                      // exponent
    w=w/sum(w);                               // není alfa

    wt(:,t)=w;
  end
  [xxx,c]=max(wt,'r');              // nové c
  if prod(c==ct($,:)), break, end   // test na změnu céček
  for i=1:nc
    j(i).c=find(c==i);              // komponenty
  end
  ct=[ct; c];                       // stará + nová   céčka

  kr=0;
  for i=1:nc
    kr=kr+sum(wt(i,j(i).c));        // kriterium (váhy -> 0 nebo 1)
  end
  printf('iter = %d\n',x)                     // iter
  printf('krit = %g\n\n',kr)                  // krit

  for i=1:nc
    thE(i)=mean(yt(:,j(i).c),2);    // nový odhad komponent
  end
  clear wt;
end

// Výsledky
// Dále je možno volit i1 a i2 (veličiny) a spouštět jako blok
```

```
cla
i1=2; i2=5;
scf(1);
plot(yt(i1,:),yt(i2,:),'.','markersize',3)
for i=1:nc
  plot(thE(i)(i1),thE(i)(i2),'.r','markersize',8)
end
```