Mixture estimation with uniform components

The main characteristic of a uniform distribution is:

1. Modeling complete uncertainty within given bounds.
2. The sharp boundary of the range of permissible values of the random variable.

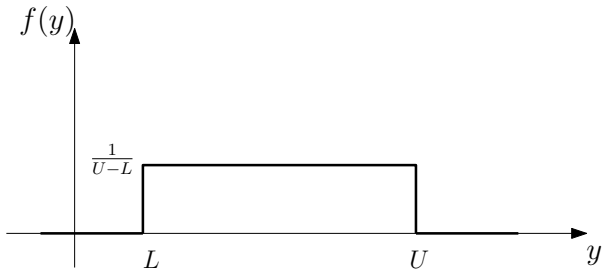
These exceptional properties of the uniform distribution have the consequence that it does not belong to the exponential class of distributions. This means that its estimation is not straightforward and should be combined with some heuristic procedures.

Next, we demonstrate two possible approaches to estimation of single uniform model. Then we extend it to mixture estimation.

Single uniform model

Maximum likelihood approach

Scalar uniform distribution can be defined through its probability density function (pdf) which is constant on a fixed interval (U, L) .The constant is $\frac{1}{U-L}$. It is drawn in the following picture



For estimation, the variables U and L are the unknown parameters. A standard way of their estimation is the ML method.

Model

$$f(y) = \frac{1}{U-L}, \text{ for } L \leq y \leq U$$

Likelihood

$$\mathcal{L} = \prod_{t=1}^{N} f(y_t) = \frac{1}{(U-L)^N}, \text{ for } L \leq \min(y), \ U \geq \max(y)$$

Maximum, under the restrictions is for

$$L \leq \min(y) \text{ and } U \geq \max(y)$$

These estimates suffer from the following drawbacks:

- The on-line estimation fails is the initial initial positions of the estimated borders L and U do not lie within the domain of the data generator.

- The variance of the estimates does not fall with time as it is with standard estimates where $\sigma_{est}^2 = \sigma^2/t$ (so called tightening of the estimates).
- Any wrong value of data shifts irrevocably the corresponding border to its position and this wrong shifting is not repaired by the following correct values.

Remark

These properties are especially disagreeable with mixture estimation because the data from other components are just those wrong values.

Method of moments

Model

$$f(y) = \frac{1}{2r},\ S - r \leq y \leq U$$

First moment

$$\bar{y} = \frac{1}{2r}\int_{S-r}^{S+r} y\,dy = \frac{1}{2r}\int_{-r}^{r} (y+S)\,dy = \frac{1}{2r}\left[\frac{y^2}{2} + Sy\right]_{-r}^{r} = S$$

Second central moment

$$\mathrm{var}(y) = \frac{1}{2r}\int_{S-r}^{S+r} (y-S)^2\,dy = \frac{1}{2r}\int_{-r}^{r} y^2\,dy = \frac{1}{2r}\left[\frac{1}{3}y^3\right]_{-r}^{r} =$$

$$= \frac{1}{6r}\left(r^3 - (-r)^3\right) = \frac{r^2}{3}$$

Estimates

$$S = \bar{y}$$

$$\mathrm{var}(y) = \frac{r^2}{3} \rightarrow r = \sqrt{3\mathrm{var}(y)} = \sqrt{3\left(m_2 - m_1^2\right)}$$

where m_1 and m_2 are first and second general moments $\bar{y}$ and $\bar{y^2}$.

This method of estimation gives results that are much better for mixture estimation. The parameters are naturally tightened (their variances decrease with the number of used data) and the exponential forgetting can be used in a standard way.

Mixture estimation with uniform components

Bayesian mixture estimation is based on weighting the data entering the estimation. The weights can be derived from proximities, i.e. "distances" of the data records to the individual components. They can be applied as the probabilities that the individual components are active with respect to the current data record or in the form of point estimates, i.e. one for the most probable components and zero for others.The latter form emphasizes the individual components and prevents from using the parasite data which have only small weight for a specific component.

The proximity is introduced as the value of the corresponding component model substituted with the data record and the existing point estimate of the parameters. This principle here

fails. The uniform distribution provides classification, not distance. The data record either belongs to the component or not. What we need is a kind of distance.

So, some other distribution must be introduced. We use normal distribution with the expectation equal to the center of the window and diagonal covariance matrix whose entries on the diagonal are proportional to the sides of the window.

For mixture estimation with uniform components we use the method of moments.

```
// Unif est - sim (Method of moments)
// - dDel  no
// - forg  no
// ----------------------------------------
[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

// ==========================================================
frg=1;   // forgetting   yL=yL-frg/t
ptw=0;    // point w      ptw=1 - use point w
// ==========================================================

nd=500;                                  // number of data
// SIMULATION
thS=list();
alS=[.4 .3 .3];                          // alS
thS(1)=[1 4
        1 7];                            // thS
thS(2)=[3 11
        6 11];
thS(3)=[10 15
        10 15];
nc=length(alS);                          // numb. of components
ny=size(thS(1),1);                       // dimension of y
for t=1:nd
  c(t)=sampCat(alS);                     // pointer generation
  for i=1:ny
    y(i,t)=randu()*(thS(c(t))(i,2)-thS(c(t))(i,1))+thS(c(t))(i,1);
  end                                    // generation of y
end

// INITIALIZATION
S=list(); T=list(); C=list(); r=list();
thR=list(); thC=list(); D=list();
S(1)=[5; 2];                             // initial statistics (sums)
S(2)=[9; 4];                             // - centers of initial windows
S(3)=[12; 9];
for j=1:nc, C(j)=S(j); r(j)=.1*ones(ny,1); D(j)=diag(r(j)); end
                                         // ceneters, radiuses, covar.
ka=ones(1:nc);                           // counter
```

```
nc=length(S);                       // number of components
nu=ones(1,nc);                      // pointer statistics
al=fnorm(nu);                       // pointer parameter
Lm=zeros(1,nc);
for j=1:nc
  thR(j)=zeros(ny,nd); thC(j)=zeros(ny,nd);
  T(j)=S(j)^2;
end
thI=thCons(C,r);                    // initial windows
ir=0;                               // switch for estimation D

for t=2:nd   // ----------- TIME LOOP --------------
  // weights
  if t>15, ir=1; end                // D is estimated from t=15
  for j=1:nc
    V=ir*1e-2*D(j)+(1-ir)*.1*eye(ny,ny);    // cov. matrix
    [xxx,Lm(j)]=GaussN(y(:,t),C(j),V);
  end                               // Gaussian proximity
  Lm=Lm-max(Lm);
  m=exp(Lm);
  if sum(m)<1E-10, m=ones(1,nc); end // component is too far
  w=(m/sum(m));                     // weights
  if ptw==1, w=dDel(w); end         // point est. of pointer
  wt(:,t)=w';

  // on-line estimation - statistics
  for j=1:nc
    ka(j)=frg*ka(j)+w(j);           // counter
    S(j)=frg*S(j)+w(j)*y(:,t);      // sum
    T(j)=frg*T(j)+w(j)*y(:,t)^2;    // sum of squares
    C(j)=S(j)/ka(j);                // centers
    r(j)=sqrt(3*(T(j)/ka(j)-C(j)^2)); // radiuses
    for i=1:ny
      if r(j)(i)<1e-8, r(j)(i)=.01; end // correct zero length
    end
    D(j)=diag(r(j));                // cov. matrix for proximity
  end

  // on-line estimation - parameters
  for j=1:nc
    thR(j)(:,t)=r(j);               // stor radiuses
    thC(j)(:,t)=C(j);               // stor centers
  end
  nu=nu+w;                          // pointer statistics
  al=fnorm(nu);                     // pointer parameter
end
th=thCons(C,r);                     //final windows

// RESULTS
```

```
cp=amax(wt,1);                    // estimates of components
Acc=acc(c,cp)                     // accuracy of classification

for j=1:nc
set(scf(),'position',[100+50*j 200+50*j 800 600])
  plot([thC(j)'])                 // evolution of centers
  title('Evolution of parameters - component '+string(j))
end

set(scf(),'position',[900 300 800 600]) // final windows
for j=1:nc
  plot(y(1,:),y(2,:),'c.')    // data
end
unifplot(thI,'.:r')           // initial parameters
unifplot(th)                  // estimated parameters
ellips(D,C)                   // cuts of Gaussians for proximity
title 'Uniform clusters (red: initial support, blue: estimated support)'
```

Program description

For the program, the library `functions` is necessary.

At the beginning of the program we have a possibility to choose the value of forgetting frg (1 means no forgetting, the recommended values for forgetting are around the value 0.95 - in dependence of the data). Another option if ptw. For its value 0 we use probability weights, for its value 1 we use one for the most probable component and zero for the others.

Then, the simulation is performed, $c\,(t)$ is the pointer variable and $y\,(t)$ is two-dimensional target variable (output).

In the section Initialization we set initial centers and radiuses and the statistics and parameters for the pointer model (nu, al).

The Time loop performs estimation:

– proximities and weights,

– recomputation of the statistics,

– construction of the point estimates of parameters.

The section Results demonstrates the evolution of the estimated parameters and the data clusters with the final estimated windows of the uniform components.