This is, how to realize the common method for marginal mixture estimation. Here, the components are common to all variables x_i. This method produces less components but it does not go to too much to details in the analysis of the space of explanatory variables x.

The algorithm of marginal mixture estimation with **common** components is here

Measure $x_t = [x_1, x_2, \cdots, x_n]_t$

For all components j

For variables i compute

$$q_j = \prod_{i=1}^{n} \underbrace{f_j\left(x_{i;t}|\hat{\theta}_{t-1}\right)}_{q_{ij}}$$

end

end

Compute weights $w = \frac{q}{\sum q_j}$

For all components j

For all variables i

update $S_{ij;t} = S_{ij;t-1} + w_j x_{i;t}$

compute $\hat{\theta}_{ij;t} = \cdots$

end

end

Program

```
// T71mixMargC.sces
// Estimation of marginal mixtures with common components
// - f(y|x1,x2) categorical, f(x1,x2) modified binomial
// kx(i)          nuber of values of var. (i)
// pI(i)(j)       initial parameters (var. i and comp. j)
// kI             initial counter (common to all)
// N(j).x(i).s    statistics (var. i and comp. j)
// N(j).x(i).k    counter (var. i and comp. j)
// C(j).x(i).p    parameters (var. i and comp. j)
// f(j).y         maginal model for y
// f(j).x(i).c    local model f_j(x_i|y)
// pj             product_i( local models_i )
// fyp            predictive pf of y|x  - sum_j( w_j*pj_j)
```

```
// ----------------------------------------
clc, clear, close(winsid()), mode(0)
getd functions

nd=1000;                          // total length of dataset

// Data                                                          // D1
nL=800;                           // number of samples for learnig  // D2
tH=[.2 .3 .3 .2];                 // pointer parameters for x       // D3
pS=list();                                                       // D4
// for x1                                                        // D5
pS(1)=[.05 .95 .9 .1];            // binomial p in components       // D6
nS(1)=4+1;                        // binomial N1 =(bin N)+1         // D7
// for x2                                                        // D8
pS(2)=[.1 .5 .01 .99];            // binomial p in components       // D9
nS(2)=5+1;                        // binomial N1 =(bin N)+1         // D10
nv=length(nS);                    // numb. of x-variables           // D11
                                                                  // D12
nn=prod(nS+1);                    // numb. of combinations of x     // D13
thY=fnorm(eye(nn,nn)+.001*randu(nn,nn),2); // pars of model f(y|x)  // D14
thY=mixData(thY);                                                 // D15

// Simulation                                                     // S1
for t=1:nd                                                        // S2
  c(t)=sampCat(tH);               // pointers                     // S3
  for i=1:nv                                                      // S4
    x(t,i)=sampBin1(pS(i)(c(t)),nS(i));                           // S5
  end                                                             // S6
  select 1   //  y can be derived either from c (1) or from x (2) // S7
  case 1, z(t)=xt2col(c(t,:),nS);  // coded c                    // S8
  case 2, z(t)=xt2col(x(t,:),nS);  // coded x                    // S9
  end                                                             // S10
  y(t)=sampCat(thY(z(t),:));                                      // S11
end                                                               // S12
                                                                  // S13
                                                                  // S14
// data to Learning and Testing part                              // S15
iL=samwr(nL,1,1:nd); iT=setdiff(1:nd,iL);                         // S16
xL=x(iL,:);          xT=x(iT,:);                                  // S17
yL=y(iL);            yT=y(iT);                                    // S18
```

```
dL=[xL yL];          dT=[xT yT];                                          // S19
cT=c(iT,:);                                                               // S20
csvWrite(dL,'dLComm.csv',';');                                            // S21
csvWrite(dT,'dTComm.csv',';');                                            // S22
disp('Data written to disk'), printf('\n')                                // S23
                                                                          // S24
nv=size(xL,2);                    // number of x-variables               // S25
ky=max(dL(:,$));                  // number of values in y               // S26
kx=max(dL(:,1:nv),'r');           // number of values in xi              // S27

// Initialization                                                         // I1
tic();                                                                    // I2
nd=length(yL);                    // data length for learning            // I3
kI=10;                            // number of initial data              // I4
pI=list();                        // initial parameters                  // I5
pI(1)=[.1 .3 .6 .9];              // comp. pars in x1                    // I6
pI(2)=[.2 .4 .7 .8];              // comp. pars in x2                    // I7
nc=length(pI(1));                                                         // I8
                                                                          // I9
for i=1:nv, for j=1:nc, N(j).x(i).k=kI; end, end // ini.kappa             // I10
for i=1:nv                                                                // I11
  for j=1:nc                                                              // I12
    N(j).x(i).s=pI(i)(j)*N(j).x(i).k*kx(i);  // ini.stats for Binom.      // I13
  end                                                                     // I14
end                                                                       // I15
                                                                          // I16
for i=1:nv                                                                // I17
  for j=1:nc                                                              // I18
    C(j).x(i).p=N(j).x(i).s/N(j).x(i).k/kx(i); // prior pt. est.          // I19
    P_INI(i,j)=pI(i)(j);          // initial pars                        // I20
  end                                                                     // I21
end                                                                       // I22

// Estimation of components                                               // E1
for t=1:nd                                                                // E2
  for j=1:nc        // for components (j)                                 // E3
    q(j)=1;                                                               // E4
    for i=1:nv      // for variables (i)                                  // E5
      p=C(j).x(i).p;                                                      // E6
      q(j)=q(j)*binom1(xL(t,i),p,kx(i));                                  // E7
```

```
    end                              // prox. for clus. j in var. i     // E8
  end                                                                   // E9
  W=q/sum(q);                        // weights w for cl. j in vars     // E10
  Wt(:,t)=W;                         // remember weights w              // E11
  for j=1:nc                                                            // E12
    for i=1:nv                                                          // E13
      N(j).x(i).s=N(j).x(i).s +W(j)*xL(t,i);  // update statistics      // E14
      N(j).x(i).k=N(j).x(i).k+W(j);                                     // E15
                                                                        // E16
      C(j).x(i).p=(N(j).x(i).s-N(j).x(i).k)/N(j).x(i).k/(kx(i)-1);     // E17
                                     // parameter estimates             // E18
      if C(j).x(i).p<0, C(j).x(i).p=0; end    // check for             // E19
      if C(j).x(i).p>1, C(j).x(i).p=1; end    //   probability         // E20
      P_EST(i,j)=C(j).x(i).p;        // remember parameters            // E21
    end,                                                                // E22
  end,                                                                  // E23
  PT(:,t)=P_EST(:);                  // pars. for plot                  // E24
end                                                                     // E25
cp=amax(Wt','c');                    // estimated pointers              // E26

// Estimation of local models                                           // L1
for j=1:nc                                                              // L2
  k=find(cp==j);                     // k = set of inexes t in clus. cp // L3
  f(j).y=pfY(yL(k),ky);              // fj(y) marginal                  // L4
  for i=1:nv                                                            // L5
    f(j).x(i).c=pfXY(xL(k,i),yL(k),kx(i),ky);                           // L6
                                     // fj(xi|y) conditional            // L7
  end                                                                   // L8
end                                                                     // L9

// Prediction of y                                                      // P1
nd=length(yT);                       // data length for testing         // P2
for t=1:nd                                                              // P3
  for j=1:nc                                                            // P4
    q(j)=1;                                                             // P5
    for i=1:nv                                                          // P6
      p=C(j).x(i).p;                                                    // P7
      q(j)=binom1(xT(t,i),p,kx(i)); // proximity for xi in clus. j      // P8
    end                                                                 // P9
  end                                                                   // P10
```

```
  W=q/sum(q);                        // vweights for clus. j in xi      // P11
  WT(:,t)=W; // remember                                                // P12
                                                                        // P13
  Py=0;                              // formal start for summation      // P14
  for j=1:nc                                                            // P15
    pj=f(j).y;                       // start for NB                    // P16
    for i=1:nv                                                          // P17
      pj=pj.*f(j).x(i).c(xT(t,i)+1,:); // NB                            // P18
    end                                                                 // P19
    Py=Py+W(j)*pj;                   // weighted sum of fj(x|y)         // P20
  end                                                                   // P21
  fyp=fnorm(Py);                     // normalization                   // P22
  Fyp(:,t)=fyp;                      // remember y-predictions          // P23
end                                                                     // P24
[nic,yp]=max(Fyp,'r');               // point estimates of y based on x // P25

// RESULTS
r=c2c(yT,yp);
disp 'Classification of y'
Acc=acc(yT(1:nd),r(yp))
disp 'Overall time'
toc(), printf('\n');
Vals_y=vals(yT)
Vals_yp=vals(r(yp))
```

Program description

For simple reading, the variables are distinguished manually (in the name of variables - e.g. $p1$, $p2$ or $C(j).x1$, $C(j).x2$) while the components are denoted by an index (mostly j). That means: the program is written just for two explanatory variables, the number of components can be changed arbitrarily (by specifying more parameters for simulation and initialization).

The parameters for simulation (as well as for estimation) are ordered so that the columns correspond to variables - must be 2 columns) and the rows correspond to the number of values in variables - they can be also changed and are determined by the number of rows. The sums in columns must always be equal to one.

Rows 1-29 perform simulation of the data. First the pointers for x-variables are generated. Then the variables $x1$ and $x2$ conditioned each on its own pointer (loop 16-21). In the end, the variable y is generated, conditioned on both $x1$ and $x2$. For its generation the probability function $f(y|x1,x2) \propto f(y|x1)\, f(y|x2)$ is used (loop 26-29). This part of the program can be

substituted by calling a dataset of real measurements.

Rows 33-60 perform a standard mixture estimation of a system with two independent explanatory variables and scalar target variable - all discrete variables modeled by categorical distribution. First the weights $w1$ and $w2$ are constructed then the statistics are recomputed using the weights and finally the point estimates of parameters are computed. All this runs in the time loop 37-60.

Second part of the algorithm is computed off-line. Here, the data clusters for each variable and its components are collected and denoted by C. Each data cluster contains values of the variable x and the corresponding values of y. From them, the frequency tables are constructed (T) and normalized in columns (i.e. over the variable y) - they are the local models $f_j\left(y|x_i\right)$ where j denotes components. They are denoted by f.

The third part of the algorithm is classification itself. After measuring the variable x, the weights $w1$ and $w2$ are computed exactly in the same way as in estimation. Then the predictive probability function $f\left(y|x\right)$ is constructed in the following way:

1. First construct the models within each component using the naive Bayes principle as the weighted sums of the variable components

$$f_j\left(y|x\right) = \prod_i f_j\left(y|x_i\right)$$

2. Then construct the predictive pf over all variables combining the component model as the weighted sum

$$f\left(y|x\right) \propto \sum_j w_j f_j\left(y|x\right)$$

The last command computes accuracy of out classification (relative number of good classifications).