This method with different components defines components to each variable x_i individually. It means, each variable can have different number of components. It provides very detailed analysis of the space x, however, the number of components can be rather high. As the component for all variables can be arbitrary combination of the local components.

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

```
Measure $x_t = [x_1, x_2, \cdots, x_n]_t$
For all variables $i$
    For all components $j$ within $x_i$
        $q_{ij} = f_j\left(x_{i;t}|\hat{\theta}_{t-1}\right)$
    end
    weights in $x_i$   $w_i = \frac{\left[q_1, q_2, \cdots q_{m(i)}\right]_i}{\sum_j q_{ij}}$
    For all components $j$ within $x_i$
        update $S_{ij;t} = S_{ij;t-1} + w_j x_{i;t}$
        compute $\hat{\theta}_{ij;t} = \cdots$
    end
end
```

Program

```
// T72mixMargD.sce
// Estimation of f(y,x1,x2) with different components
// - y generated from product of fy1*fy2
// 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)
// fy             maginal model for y
// f(j).x(i).c    local model f_j(x_i|y)
// fpi            weighted sum of local models
// Fyp            product of fpi
// ------------------------------------------
clc, clear, close(winsid()), mode(0)
```

```
getd functions

// DATA                                                               // D1
nd=1000;                           // total number of dataset         // D2
nL=800;                            // number of samples for learnig   // D3
pS=list(); tH=list(); thY=list();                                     // D4
// for x1        for x2                                               // D5
tH(1)=[.2 .8];   tH(2)=[.3 .7];    // pointer parameters              // D6
pS(1)=[.002 .994]; pS(2)=[.005  .991]; // binomial p                  // D7
nS(1)=5+1;       nS(2)=5+1;        // binomial n                      // D8
nX=length(nS);                     // numb. of x-variables            // D9
ny=6;                                                                 // D10
thY(1)=genThs(ny,nS(1),.1);                                           // D11
thY(2)=genThs(ny,nS(2),.1);                                           // D12

// SIMULATION of the whole dataset (L + T)                            // S1
for t=1:nd                                                            // S2
  fY=ones(ny,1);                                                      // S3
  for i=1:nX                                                          // S4
    c(t,i)=sampCat(tH(i));                                            // S5
    x(t,i)=sampBin1(pS(i)(c(t,i)),nS(i));                             // S6
    fY=fY.*thY(i)(:,x(t,i));                                          // S7
  end                                                                 // S8
  fY=fnorm(fY);                                                       // S9
  y(t)=sampCat(fY);                                                   // S10
end                                                                   // S11
                                                                      // S12
// Learning and Testing data                                          // S13
iL=samwr(nL,1,1:nd); iT=setdiff(1:nd,iL);                             // S14
xL=x(iL,:);          xT=x(iT,:);                                      // S15
yL=y(iL);            yT=y(iT);                                        // S16
dL=[xL yL];          dT=[xT yT];                                      // S17
cT=c(iT,:);                                                           // S18
//zT=z(iT);                                                           // S19
csvWrite(dL,’dLDiff.csv’,’;’);     // write data to disk              // S20
csvWrite(dT,’dTDiff.csv’,’;’);                                        // S21
disp(’Data written to disk’), printf(’\n’)                            // S22
                                                                      // S23
nx=size(xL,2);                     // number of x-variables           // S24
ky=max(dL(:,$));                   // number of values in y           // S25
```

```
kx=max(dL(:,1:nx),'r');            // number of values in xi          // S26

// 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)=pS(1); pI(2)=pS(2);          // exact values (for validation)   // I6
for i=1:nx, nc(i)=length(pI(i)); end                                  // I7
                                                                      // I8
for i=1:nx, for j=1:nc(i), N(j).x(i).k=kI; end, end // init.kappa     // I9
for i=1:nx                                                            // I10
  for j=1:nc(i)                                                       // I11
    N(j).x(i).s=pI(i)(j)*N(j).x(i).k*(kx(i)-1)+N(j).x(i).k;           // I12
                                   // statistics for Binom.           // I13
  end                                                                 // I14
end                                                                   // I15
                                                                      // I16
for i=1:nx                                                            // I17
  for j=1:nc(i)                                                       // I18
    C(j).x(i).p=(N(j).x(i).s-N(j).x(i).k)/N(j).x(i).k/(kx(i)-1);      // I19
                                   // prior est. of parameters        // I20
  end                                                                 // I21
end                                                                   // I22

// ESTIMATION  of f(x) clusters                                       // E1
for t=1:nd                                                            // E2
  for i=1:nx        // for variables (i)                              // E3
    q=0;                                                              // E4
    for j=1:nc(i)   // for components (j)                             // E5
      p=C(j).x(i).p;                                                  // E6
      q(j)=binom1(xL(t,i),p,kx(i)); // prox. for clus. j in var i     // E7
    end                                                               // E8
    w=q/sum(q);                    // weights w for cl. j in var i    // E9
    wt(i).w(:,t)=w;                // remember weights w              // E10
    for j=1:nc(i)                                                     // E11
      N(j).x(i).s=N(j).x(i).s +w(j)*xL(t,i); // update statistics     // E12
      N(j).x(i).k=N(j).x(i).k+w(j);                                   // E13
                                                                      // E14
      C(j).x(i).p=(N(j).x(i).s-N(j).x(i).k)/N(j).x(i).k/(kx(i)-1);    // E15
```

```
                                   // parameter estimates             // E16
      if C(j).x(i).p<0, C(j).x(i).p=0; end   // check for             // E17
      if C(j).x(i).p>1, C(j).x(i).p=1; end   //   probability         // E18
      pt(i,j)=C(j).x(i).p;         // remember estimates              // E19
    end,                                                              // E20
  end                                                                 // E21
  PT(:,t)=pt(:);                   // parameters for final display    // E22
end                                                                   // E23
                                                                      // E24
cp=list();                                                            // E25
for i=1:nx                                                            // E26
  cp(i)=amax(wt(i).w','c');        // estimated pointers              // E27
end                                                                   // E28
Acc1=acc(c(iL,1),cp(1))                                               // E29
Acc2=acc(c(iL,2),cp(2))                                               // E30

// LOCAL MODELS f(xi|y) in clusters                                   // L1
fy=pfY(yL,ky);                     // marginal f(y)                   // L2
for i=1:nx                                                            // L3
  for j=1:nc(i)                                                       // L4
    k=find(cp(i)==j);              // k = set of times t              // L5
                                   //   in cluster j and variable xi  // L6
    f(j).x(i).c=pfXY(xL(k,i),yL(k),kx(i),ky);                         // L7
                                   // conditional fj(xi|y)            // L8
  end                                                                 // L9
end                                                                   // L10

// PREDICTION                                                         // P1
nd=length(yT);                     // data length for testing         // P2
for t=1:nd                                                            // P3
  Fyp=fy;                          // beginning of NB  f(y)           // P4
  for i=1:nx                                                          // P5
    q=0;                                                              // P6
    for j=1:nc(i)                                                     // P7
      p=C(j).x(i).p;                                                  // P8
      q(j)=binom1(xT(t,i),p,kx(i)); // proximity pro xi v klastrech j // P9
    end                                                               // P10
    w=q/sum(q);                    // weights for clusters j in xi    // P11
    WT(i).w(:,t)=w;                // remember weights                // P12
                                                                      // P13
    fpi=0;                         // formal start for summation      // P14
    for j=1:nc(i)                                                     // P15
      fpi=fpi+w(j)*f(j).x(i).c(xT(t,i),:);                            // P16
                                   // weighted sum for f(xi|y)        // P17
    end                                                               // P18
    Fyp=Fyp.*fpi;                  // NB for  f(y|x)  over xi         // P19
  end                                                                 // P20
  yp(t)=amax(Fyp);                 // point estimate of y based on x  // P21
end

// 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 1-30 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 37-55.

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 for individual variables as the weighted sums of the variable components
$$f\left(y|x_i\right) \propto \sum_j w_j f_j\left(y|x_i\right)$$
2. Then, using the naive Bayes principle construct the predictive pf over all variables

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

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