Ústav teorie informace a automatizace AV ČR, v.v.i.

The Czech Academy of Sciences,
Institute of Information Theory and Automation

RESEARCH REPORT

I. Nagy, E. Suzdaleva, P. Pecherková

Software documentation

Clustering and Classification Using Recursive Mixture Estimation

2018

Project GAČR GA15-03564S

ÚTIA AV ČR, P.O.Box 18, 182 08 Prague, Czech Republic
Tel: (+420)266052422, Fax: (+420)286890378, Url: http://www.utia.cas.cz, E-mail: utia@utia.cas.cz

Any opinions and conclusions expressed in this report are those of the authors and do not necessarily represent the views of the involved institutions.

1 Introduction

The report represents a documentation for the software packages, which were developed within the project GAČR GA15-03564S "Clustering and Classification Using Recursive Mixture Estimation". The main contribution of the project was to propose and implement the algorithms for mixture-based clustering and classification for various combinations of components as well as pointer models [1] in the programming free and open source environment Scilab (www.scilab.org). The discussed algorithms are generally based on the recursive Bayesian mixture estimation theory [2, 1, 3, 4], which was further extended and presented in papers mentioned below throughout this documentation.

The programs are implemented in a user-friendly way. The open source codes with the testing examples load the simulated data and can be utilized twofold:

- firstly, for the aim of the verification of the algorithm implementation
- and secondly, as the example to be used by potential users for their specific tasks,

which means that the users can load their own data, choose the component type and the suitable pointer model as well as modify the code if necessary.

The software package consists of two parts according to the specification of the mixture model used for the data clustering. Their summarized description is given below.

2 Dynamic pointer and various types of components

This part of the project software provides programs for the mixture-based clustering and classification using the dynamic pointer model [4] and the following types of the mixture components:

- normal regression models [2, 1, 3];
- categorical models [3];
- normal state-space model [5].

The programs, which are implemented in the Scilab version 5.5.2 are described in details in [6] along with the explanation of the algorithms developed. This package has the main program `ExAll.sce`, where the users can choose the necessary settings concerning a type of components and load their own data sets or use the available simulations. The main program is available online as the software attached to the book [6] as well as in the text. All the subroutines for

- the mixture simulation,
- the mixture initialization [7],
- the proximity computation [8]
- the statistics update [1, 3, 4, 5];

for all of the mentioned components are also given both online and as the text in the book with the detailed explanation. The unified approach to the used component types is one of the main contributions of the book.

3 Data-dependent pointer and various types of components

This part of the software package includes the programs for the mixture-based clustering and classification for two types of the data-dependent pointer model and various types of components. The implementation of the software is prepared in Scilab 6. Here, two main programs can be used. They are described below.

3.1 Dynamic data-dependent pointer and various types of components

The main program titled `ExAllDyn.sce` performs the mixture-based cluster analysis with the dynamic data-dependent pointer model [9]. The data variables are described by the following distributions standing for the mixture components:

- normal regression models [9];
- categorical models [10];
- normal state-space models [10];
- exponential models [11];
- uniform models [10, 12].

The unified approach to the mixture estimation algorithms is kept similarly as in the previous case. The program is demonstrated below.

```
//The research was supported by the project  GACR GA15-03564S

// Mixture estimation for all model types
//  - components: tMix=1   normal regression model
//                tMix=2   categorical model
//                tMix=3   normal state-space model
//                tMix=4   exponential model
//                tMix=5   uniform model
//  - dynamic data-dependent pointer model
// ---------------------------------------------------------------------
exec("Intro.sce",-1), mode(0);

// CHOICE OF COMPONENT TYPE
tMix=5;       // 1=regr, 2=categ, 3=state, 4=exp, 5=uniform

// TASK SPECIFICATION
nd=600;                                 // number of data
fi=.99999;                              // exponential forgetting
np=0;                                   // length of prediction (can be 0)
//////////////////////////////////

// DATA LOAD  (simulators are E1Sim, E2Sim, E3Sim)
select tMix
  case 1 then                           // - REGRESSION MIXTURE
    load('_data/datRegEx','yt','zt','ct','yi','ths','cvs','als','ncs');
  case 2 then                           // - DISCRETE MIXTURE
    load('_data/datDisEx','yt','zt','ct','yi','als','ps','ncs','kys');
  case 3 then                           // - STATE-SPACE MIXTURE
    load('_data/datStaEx','yt','zt','ct','als','ncs','M','A','F','Rw','Rv');
  case 4 then                           // - EXPONENTIAL MIXTURE
    load('_data/datExpEx','yt','zt','ct','yi','ths','as','als','ncs');
  case 5 then                           // - UNIFORM MIXTURE
    load('_data/datUniEx','yt','zt','ct','yi','ths','als','ncs');
end
nc=ncs; // no. of components is taken from simulation

// INITIALIZATION ------------------------------------------------------
// initialization of the pointer statistics
nz=max(zt);
w=ones(nc,1)/nc; wy=w*ones(1,np+1);                             // initial weights
nu=list(); al=list();
for i=1:nz
```

```
  nu(i)=5*eye(nc,nc)+1*rand(nc,nc,'unif');        // statistics
  nu(i)=1e-5*ones(nc,nc);     // XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  al(i)=fnorm(nu(i),2);                           // point estimates
end

// initialization of the component statistics
select tMix
  case 1 then [Est,ka]=initReg(yi);               // ini.com.stats
  case 2 then mu=initDis(yi,max(yt,'c'));         // ini.com.stats
  case 3 then Rx=eye(2,2)*1e6; xx=zeros(2,1);     // ini. KF stats
  case 4 then [Est,ka]=initExp(yi,as);            // ini. Exp stats
  case 5 then [Est,ka]=initUni(yi);               // ini. Unif stats
end
if tMix==1, EstI=Est; end

// TIME LOOP -----------------------------------------------------------
printf('running .....|\n '),itime=0;
for t=2:nd                                                        //  1
  itime=itime+1; if itime>(nd-1)/20, mprintf('.'), itime=0; end
  // Computation of data-component "proximities"
  select tMix                                                     //  2
    case 1 then mp=predReg(nc,yt(:,t),Est);                       //  3
    case 2 then mp=predDis(nc,yt(:,t),mu);                        //  4
    case 3 then [mp,KF]=predSta(nc,xx,yt(:,t),M,A,F,Rw,Rv,Rx);    //  5
    case 4 then mp=predExp(nc,yt(:,t),Est);
    case 5 then mp=predUni(nc,yt(:,t),Est);
  end                                                             //  6
  if t<1 // number of steps without al
    Wp=(w*mp'); W=Wp/sum(Wp);                 // without al        //  7
  else
    Wp=(w*mp').*al(zt(t)); W=Wp/sum(Wp);      // with al
  end
  wp=sum(W,'r')'; w=wp/sum(wp);

  // Update of statitics and point estimates of parameters
  // - statistics of the pointer
  nu(zt(t))=nu(zt(t))+W;            // update of statistics       //  9
  al(zt(t))=fnorm(nu(zt(t)),2);     // point estimate of alpha    // 10
  wy=(al(zt(t))')^np*w;             // weights for prediction     // 11

  // - statistics of components
  select tMix                                                     // 12
    case 1 then [Est,yp,ka]=updtReg(w,yt(:,t),Est,ka,fi,wy);      // 13
    case 2 then [Est,yp,mu]=updtDis(w,yt(:,t),mu);                // 14
    case 3 then [xx,Rx,yp]=updtSta(w,KF);                         // 15
    case 4 then [Est,yp,ka]=updtExp(w,yt(:,t),Est,ka,fi,wy,as);
    case 5 then [Est,yp,ka]=updtUni(w,yt(:,t),Est,ka,wy,.2,.1,3);
                        // catch point; forget point; lag before forg.
  end                                                             // 17
  ypt(:,t+np)=yp;                   // store                      // 18
  wt(:,t)=w;                        // store (components)         // 19
  wyt(:,t)=wy;                      // store (prediction)         // 20

  // storing current results
  if tMix~=3                                                      // 21
  for i=1:nc                                                      // 22
    Res(i).th(:,t)=Est(i).th(:);    // i-th component center      // 23
  end                               // only for regression model  // 24
```

```
end                                                                    // 25
end                                                                    // 26

// end of TIME LOOP   ------------------------------------------------

[xxx Ect]=max(wt,'r');              // pt.est. - active comp.      // 27
[xxx Ecp]=max(wyt,'r');             // pt.est. - predic. comp.     // 28

if (max(ct)==max(Ect))
[q,T]=c2c(ct,Ect);
Tct=q(Ect);
if prod(sum(T,1))==0
  disp ' !!! At least one component is not found !!!'
end
end

select tMix
case 1 then exec('Results1.sce',-1)     // regression mixture
case 2 then exec('Results2.sce',-1)     // categ. mixture
case 3 then exec('Results3.sce',-1)     // state-space mixture
case 4 then exec('Results4.sce',-1)     // exp. mixture
case 5 then exec('Results5.sce',-1)     // unif. mixture
end
```

3.1.1 Comments

The structure of the presented program is similar to that given in [6] to keep the unified approach as much as possible. Therefore, there is no necessity to describe the program in details. The functions used inside the main program performing the mixture initialization, the proximity computation, the statistics update, setting a Scilab start and plotting the results are either identical to those given in [6] or developed according to the papers mentioned above in dependence on the component type.

To summarize the information for the users, the key points of the code are as follows.

The variable `tMix` serves for choosing the component type. The initialization (before the time loop) and the proximity computation (rows 2–6) are running for the chosen type of the components automatically. Rows 7–11 concern with updating the pointer statistics and obtaining the weighting vector necessary for the data clustering. Rows 12–15 perform the component statistics update with the obtained weights. The active component and its prediction are given in rows 27–28.

The rest of the program deals with storing the results and their plotting according to the component type. The extensive description of results of experiments conducted with the help of the presented programs along with the graphical representation of clusters are given in the mentioned sources.

The users can either use the simulations or load their own data. The users are free to modify any part of the code to tailor it for their specific tasks.

3.2 Static data-dependent pointer and various types of components

Here, the main program `ExAllStat.sce` performs the clustering and classification using the static data-dependent pointer model [10] based on [1]. The same five types of the components can be chosen within the program, i.e.,

- normal regression models;
- categorical models;
- normal state-space models;
- exponential models;
- uniform models.

The unified approach to the components is kept. The program is given below.

```
//The research was supported by the project  GACR GA15-03564S

// Mixture estimation for all model types
//  - components: tMix=1   normal regression model
//                tMix=2   categorical model
//                tMix=3   normal state-space model
//                tMix=4   exponential model
//                tMix=5   uniform model
//  - static data-dependent pointer model
// ---------------------------------------------------------------
exec("Intro.sce",-1), mode(0);

// CHOICE OF COMPONENT TYPE
tMix=1;       // 1=regr, 2=categ, 3=state, 4=exp, 5=uniform

// TASK SPECIFICATION
nd=600;                                 // number of data
fi=.99999;                              // exponential forgetting
////////////////////////////////////

// DATA LOAD  (simulators are E1Sim, E2Sim, E3Sim)
select tMix
  case 1 then                           // - REGRESSION MIXTURE
    load('_data/datRegEx','yt','zt','ct','yi','ths','cvs','als','ncs');
  case 2 then                           // - DISCRETE MIXTURE
    load('_data/datDisEx','yt','zt','ct','yi','als','ps','ncs','kys');
  case 3 then                           // - STATE-SPACE MIXTURE
    load('_data/datStaEx','yt','zt','ct','als','ncs','M','A','F','Rw','Rv');
  case 4 then                           // - EXPONENTIAL MIXTURE
    load('_data/datExpEx','yt','zt','ct','yi','ths','as','als','ncs');
  case 5 then                           // - UNIFORM MIXTURE
    load('_data/datUniEx','yt','zt','ct','yi','ths','als','ncs');
end
nc=ncs; // no. of components is taken from simulation

// INITIALIZATION ----------------------------------------------------
// initialization of the pointer statistics
np=0;
nz=max(zt);
w=ones(nc,1)/nc; wy=w*ones(1,np+1);               // initial weights
nu=list(); al=list();
for i=1:nz
  nu(i)=5*eye(1,nc)+1*rand(1,nc,'unif');        // statistics
  nu(i)=1e-5*ones(1,nc);     // XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  al(i)=fnorm(nu(i));                           // point estimates
end

// initialization of the component statistics
select tMix
  case 1 then [Est,ka]=initReg(yi);               // ini.com.stats
  case 2 then mu=initDis(yi,max(yt,'c'));         // ini.com.stats
  case 3 then Rx=eye(2,2)*1e6; xx=zeros(2,1);     // ini. KF stats
  case 4 then [Est,ka]=initExp(yi,as);            // ini. Exp stats
  case 5 then [Est,ka]=initUni(yi);               // ini. Unif stats
end
if tMix==1, EstI=Est; end

// TIME LOOP ---------------------------------------------------------
```

```
printf('running ............|\n '),itime=0;
for t=2:nd                                                          //  1
  itime=itime+1; if itime>(nd-1)/20, mprintf('.'), itime=0; end
  // Computation of data-component "proximities"
  select tMix                                                       //  2
    case 1 then mp=predReg(nc,yt(:,t),Est);                         //  3
    case 2 then mp=predDis(nc,yt(:,t),mu);                          //  4
    case 3 then [mp,KF]=predSta(nc,xx,yt(:,t),M,A,F,Rw,Rv,Rx);      //  5
    case 4 then mp=predExp(nc,yt(:,t),Est);
    case 5 then mp=predUni(nc,yt(:,t),Est);
  end                                                               //  6
  if t<1 // number of steps without al
    wp=mp'; w=wp/sum(wp);               // without al               //  7
  else
    wp=mp'.*al(zt(t)); w=wp/sum(wp);    // with al                 //  8
  end


  // Update of statitics and point estimates of parameters
  // - statistics of the pointer
  nu(zt(t))=nu(zt(t))+w;            // update of statistics        //  9
  al(zt(t))=fnorm(nu(zt(t)),2);     // point estimate of alpha     // 10

  wy=ones(1,nc)/nc;
  // - statistics of components
  select tMix                                                       // 12
    case 1 then [Est,yp,ka]=updtReg(w,yt(:,t),Est,ka,fi,wy);        // 13
    case 2 then [Est,yp,mu]=updtDis(w,yt(:,t),mu);                  // 14
    case 3 then [xx,Rx,yp]=updtSta(w,KF);                           // 15
    case 4 then [Est,yp,ka]=updtExp(w,yt(:,t),Est,ka,fi,wy,as);
    case 5 then [Est,yp,ka]=updtUni(w,yt(:,t),Est,ka,wy,.2,.1,3);
                        // catch point; forget point; lag before forg.
  end                                                               // 17
  ypt(:,t+np)=yp;                   // store                      // 18
  wt(:,t)=w;                        // store (components)         // 19
  wyt(:,t)=wy;                      // store (prediction)         // 20

  // storing current results
  if tMix~=3                                                        // 21
  for i=1:nc                                                        // 22
    Res(i).th(:,t)=Est(i).th(:);    // i-th component center      // 23
  end                               // only for regression model  // 24
end                                                                 // 25
end                                                                 // 26

// end of TIME LOOP   ---------------------------------------------

[xxx Ect]=max(wt,'r');              // pt.est. - active comp.      // 27
[xxx Ecp]=max(wyt,'r');             // pt.est. - predic. comp.     // 28

if (max(ct)==max(Ect))
[q,T]=c2c(ct,Ect);
Tct=q(Ect);
if prod(sum(T,1))==0
  disp ' !!! At least one component is not found !!!'
end
end
```

```
select tMix
case 1 then exec('Results1.sce',-1)     // regression mixture
case 2 then exec('Results2.sce',-1)     // categ. mixture
case 3 then exec('Results3.sce',-1)     // state-space mixture
case 4 then exec('Results4.sce',-1)     // exp. mixture
case 5 then exec('Results5.sce',-1)     // unif. mixture
end
```

3.2.1 Comments

As it can be seen, the structure of the program is the same as the previous one with the exception of the pointer statistics update in rows 9–10 and the absence of the pointer prediction.

4 Conclusion

The research report presents the software documentation for the clustering and classification using recursive mixture estimation. Three types of the pointer models and five types of the mixture components can be chosen in the main programs prepared in a user-friendly way. The software is implemented in the free and open source programming environment Scilab. The program codes are editable and adjustable for users' tasks.

Acknowledgements

The research was supported by the project GAČR GA15-03564S.

References

[1] Kárný, M., Kadlec, J., Sutanto, E.L.,1998. Quasi-Bayes estimation applied to normal mixture. In: 3rd European IEEE Workshop on Computer-Intensive Methods in Control and Data Processing, Rojíček, J., Valečková, M., Kárný, M., Warwick, K. (Eds.), 1998, September, Prague, Czech Republic, pp. 77-82.

[2] Peterka, V., 1981. Bayesian system identification. In: Eykhoff, P. (Ed.), Trends and Progress in System Identification. Oxford, Pergamon Press, pp. 239-304.

[3] Kárný, M., Böhm, J., Guy, T. V., Jirsa, L., Nagy, I., Nedoma, P., Tesař, L., 2006. Optimized Bayesian Dynamic Advising: Theory and Algorithms. Springer-Verlag, London.

[4] Nagy, I., Suzdaleva, E., Kárný, M., Mlynářová, T., 2011. Bayesian estimation of dynamic finite mixtures. International Journal of Adaptive Control and Signal Processing. 25(9), 765-787.

[5] Nagy, I., Suzdaleva, E., 2013. Mixture estimation with state-space components and Markov model of switching. Applied Mathematical Modelling. 37(24), 9970-9984.

[6] Nagy, I., Suzdaleva, E., 2017. Algorithms and Programs of Dynamic Mixture Estimation. Unified Approach to Different Types of Components. Springer, (Cham 2017) SpringerBriefs in Statistics.

[7] Suzdaleva, E., Nagy, I., Mlynářová, T., 2016. Expert-based initialization of recursive mixture estimation. In: 8th IEEE International Conference on Intelligent Systems, 2016, September 4-6, Sofia, Bulgaria, pp. 308-315.

[8] Nagy, I., Suzdaleva, E., Pecherková, P., 2016. Comparison of various definitions of proximity in mixture estimation. In: Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Lisbon, Portugal, July, 29 – 31, 2016, pp. 527-534.

[9] Suzdaleva, E. and Nagy, I., 2018. An online estimation of driving style using data-dependent pointer model. Transportation Research Part C. 86C, 23-36.

[10] Likhonina, R., Suzdaleva, E., Nagy, I., 2016. Comparison of mixture-based classification with the data-dependent pointer model for various types of components. Research report 2355. ÚTIA AV ČR, Prague.

[11] Suzdaleva, E., Nagy, I., Petrouš, M., 2017. Recursive clustering hematological data using mixture of exponential components. In: Proceedings of International Conference on Intelligent Informatics and BioMedical Sciences ICIIBMS 2017, Okinawa, Japan, November, 24 – 26, 2017, pp. 63-70.

[12] Nagy, I., Suzdaleva, E., Mlynářová, T., 2016. Mixture-based clustering non-gaussian data with fixed bounds. In: 8th IEEE International Conference on Intelligent Systems, 2016, September 4-6, Sofia, Bulgaria, pp. 265-271.