The Stata Journal (2012)
12, Number 2, pp. 284–298

Fitting nonparametric mixed logit models via expectation-maximization algorithm

Daniele Pacifico
Italian Department of the Treasury
Rome, Italy
daniele.pacifico@tesoro.it

Abstract. In this article, I provide an illustrative, step-by-step implementation of the expectation-maximization algorithm for the nonparametric estimation of mixed logit models. In particular, the proposed routine allows users to fit straightforwardly latent-class logit models with an increasing number of mass points so as to approximate the unobserved structure of the mixing distribution.

Keywords: st0258, latent classes, expectation-maximization algorithm, nonparametric mixed logit

1 Introduction

In this article, I develop a nonparametric method suggested by Train (2008) for the estimation of mixing distributions in conditional logit models, and I show how to implement it in practice. Specifically, the proposed code allows users to estimate a discrete mixing distribution with points and shares as parameters. This constitutes a typical latent-class logit model, where the underlying mixing distribution can be fit nonparametrically by setting a sufficiently large number of mass points.

Clearly, latent-class models can be fit through standard optimization methods, such as Newton–Raphson or Berndt–Hall–Hall–Hausman. However, this becomes difficult when the number of classes increases, because the higher the number of parameters the more difficult the inversion of the empirical Hessian matrix, with the possibility of singularity at some iterations. In such situations, the proposed expectation-maximization (EM) recursion could help, because it implies the repeated evaluation of a function that is easier to maximize. Moreover, EM algorithms have proved to be particularly stable, and—under conditions given by Boyles (1983) and Wu (1983)—they always climb uphill until convergence to a local maximum.

The article is structured as follows: section 2 presents the mixed logit model with both continuous and discrete mixing distributions; section 3 shows how a mixed logit model with discrete mixing distributions can be fit via EM algorithm; section 4 presents an illustrative, step-by-step implementation of the EM recursion; section 5 contains an empirical application based on accessible data; and section 6 concludes.

2 The mixed logit model

Assume there are N agents who face J alternatives on T choice occasions. Agents choose the alternative that maximizes their utility in each choice occasion. The random utility of agent i from choosing alternative j in period t is defined as follows

$$U_{ijt=}\boldsymbol{\beta}_i\boldsymbol{x}_{ijt} + \epsilon_{ijt}$$

where $\boldsymbol{x}_{ijt}$ is a vector of alternative-specific attributes and ϵ_{ijt} is a stochastic term that is assumed to be an independent and identically distributed extreme value. Importantly, each β_i is assumed to be random with the unconditional density $f(\beta \,|\, \boldsymbol{\vartheta})$, where $\boldsymbol{\vartheta}$ collects the parameters that characterize the distribution.

Conditional on knowing $\boldsymbol{\beta}_i$, the probability of the observed sequence of choices for agent i is given by the traditional McFadden's (1974) choice model

$$\Pr_i(\boldsymbol{\beta}) = \prod_{t=1}^{T}\prod_{j=1}^{J}\left\{\frac{\exp(\boldsymbol{\beta}_i\boldsymbol{x}_{ijt})}{\sum_{k=1}^{J}\exp(\boldsymbol{\beta}_i\boldsymbol{x}_{ikt})}\right\}^{d_{ijt}}$$

where d_{ijt} is a dummy that selects the chosen alternative in each choice occasion. However, because $\boldsymbol{\beta}_i$ is unknown, the conditional probability of the sequence of observed choices has to be evaluated for any possible value of $\boldsymbol{\beta}_i$. Hence, assuming that $f(\beta \,|\, \boldsymbol{\vartheta})$ has a continuous distribution, the unconditional probability becomes

$$\Pr_i(\boldsymbol{\vartheta}) = \int \prod_{t=1}^{T}\prod_{j=1}^{J}\left\{\frac{\exp(\boldsymbol{\beta}\boldsymbol{x}_{ijt})}{\sum_{k=1}^{J}\exp(\boldsymbol{\beta}\boldsymbol{x}_{ikt})}\right\}^{d_{ijt}} f(\boldsymbol{\beta}|\boldsymbol{\vartheta}) \tag{1}$$

This probability defines a mixed logit model with continuous mixing distributions, and typically, the related log-likelihood function is fit with simulation methods.[1] If the distribution of each $\boldsymbol{\beta}_i$ is discrete, the probability in (1) becomes

$$\Pr_i(\boldsymbol{\vartheta}) = \sum_{c=1}^{C}\pi_c\prod_{t=1}^{T}\prod_{j=1}^{J}\left\{\frac{\exp(\boldsymbol{\beta_c}\boldsymbol{x}_{ijt})}{\sum_{k=1}^{J}\exp(\boldsymbol{\beta_c}\boldsymbol{x}_{ikt})}\right\}^{d_{ijt}} \tag{2}$$

where $\pi_c = f(\boldsymbol{\beta_c}|\boldsymbol{\vartheta})$ represents the share of the population with coefficients $\boldsymbol{\beta_c}$.

Equation (2) is a typical latent-class logit model. Nevertheless, here we follow McFadden and Train (2000) and define it as a mixed logit model with discrete mixing distributions, to emphasize the similarities with the continuous-mixture logit model of (1).

The estimation of latent-class models is usually based on standard gradient-based methods. However, these methods often fail to achieve convergence when the number of latent classes becomes high. In this case, an EM algorithm could help, because it requires the repeated maximization of a function that is far simpler with respect to the log likelihood derived from (2).

1. See Train (2003). In Stata, continuous mixed logit models can be fit with simulated maximum likelihood through the command `mixlogit`, written by Hole (2007).

3 An EM algorithm for the estimation of mixed logit models with discrete mixing distributions

EM algorithms were initially proposed in the literature to deal with missing-data problems. Nevertheless, they turned out to be an efficient method to fit latent-class models, where the missing information consists of the class share probabilities. Nowadays, they are widely used in many economic fields where the assumption that people can be grouped in classes with different unobserved preference heterogeneity is reasonable.

The recursion is known as "E-M" because it consists of two steps, namely, an "expectation" and a "maximization". As explained in Train (2008), the term to be maximized is the expectation of the missing-data log likelihood—that is, the joint density of the observed choice and the missing data—whilst the expectation is over the distribution of the missing information, conditional on the density of the data and the previous parameter estimates.

Consider the conditional logit model with discrete mixing distributions that we outlined in the previous section. Following (2), the log likelihood is defined as

$$\text{LL} = \sum_{i=1}^{N} \ln \sum_{c=1}^{C} \pi_c \prod_{t=1}^{T} \prod_{j=1}^{J} \left\{ \frac{\exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt})}{\sum_{k=1}^{J} \exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ikt})} \right\}^{d_{ijt}}$$

which can be maximized by means of standard, gradient-based optimization methods. However, the same log likelihood can be also maximized by repeatedly updating the following recursion

$$\begin{aligned} \boldsymbol{\eta}^{s+1} &= \text{argmax}_{\eta} \textstyle\sum_i \sum_c C_i(\boldsymbol{\eta}^s) \ln \pi_c \prod_t \prod_j \left\{ \frac{\exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt})}{\sum_{k=1}^{J} \exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ikt})} \right\}^{d_{ijt}} \\ &= \text{argmax}_{\eta} \textstyle\sum_i \sum_c C_i(\boldsymbol{\eta}^s) \ln(L_i \,|\, \text{class}_i = c) \end{aligned} \tag{3}$$

where $\boldsymbol{\eta}$ is a vector that contains the whole set of parameters to be estimated—that is, those that enter the probability of the observed choice plus those that may define the class shares—L_i is the missing-data likelihood function, and $C_i(\boldsymbol{\eta}^s)$ is the conditional probability that household i belongs to class c, which depends on the density of the data and the previous value of the parameters.

This conditional probability—$C_i(\boldsymbol{\eta}^s)$—is the key future of the EM recursion and can be computed by means of the Bayes' theorem:

$$C_i(\boldsymbol{\eta}^s) = \frac{L_i|\text{class}_i = c}{\sum_{c=1}^{C} L_i|\text{class}_i = c} \tag{4}$$

Now, given that

$$\ln(L_i \,|\, \text{class}_i = c) = \ln \pi_c + \ln \prod_{t=1}^{T} \prod_{j=1}^{J} \left\{ \frac{\exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt})}{\sum_{k=1}^{J} \exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ikt})} \right\}^{d_{ijt}} \tag{5}$$

the recursion in (3) can be split into the following steps:

1. Form the contribution to the likelihood $(L_i \,|\, \text{class}_i = c)$ as defined in (3) for each class;[2]

2. Form the individual-specific conditional probabilities of class membership as in (4);

3. Following (5), update the sets of $\boldsymbol{\beta_c}$, $c = 1, 2, \ldots, C$, by maximizing—for each class—a conditional logit model with weighted observations, with weights given by the conditional probabilities of class membership:

$$\boldsymbol{\beta_c^{s+1}} = \text{argmax}_{\boldsymbol{\beta}} \sum_{i=1}^{N} C(\boldsymbol{\eta}^s) \ln \prod_{t=1}^{T} \prod_{j=1}^{J} \left\{ \frac{\exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt})}{\sum_{k=1}^{J} \exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ikt})} \right\}^{d_{ijt}}$$

4. Maximize the other part of the log likelihood in (3), and get the updated vector of class shares:

$$\boldsymbol{\pi^{s+1}} = \text{argmax}_{\boldsymbol{\pi}} \sum_{i=1}^{N} \sum_{c=1}^{C} C_i(\boldsymbol{\eta}^s) \ln \pi_c$$

   - If the class share probabilities depend on a vector of demographics—$\boldsymbol{z_i}$—the relative parameters are updated as

$$\boldsymbol{\alpha^{s+1}} = \text{argmax}_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \sum_{c=1}^{C} C_i(\boldsymbol{\eta}^s) \ln \frac{\exp(\boldsymbol{\alpha_c} \boldsymbol{z_i})}{\sum_c \exp(\boldsymbol{\alpha_c} \boldsymbol{z_i})}, \ \boldsymbol{\alpha_C = 0} \tag{6}$$

     This is a grouped-data log likelihood, where we have used a logit specification to constrain the estimated class share probabilities into the right range.[3] The updated class share probabilities—π_c, $c = 1, 2, \ldots, C$—are then computed as

$$\pi_c^{s+1} = \frac{\exp(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{c}}^{\boldsymbol{s+1}} \boldsymbol{z_i})}{\sum_c \exp(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{c}}^{\boldsymbol{s+1}} \boldsymbol{z_i})}, \ c = 1, 2, \ldots, C$$

     which in turn allows updating the conditional probabilities of class membership by means of the new vectors $\boldsymbol{\beta_c^{s+1}}$ and π_c^{s+1}, $c = 1, 2, \ldots, C$.

   - If the class share probabilities do not depend on demographics, the empirical maximization of the function in (6) can be avoided, because its analytical solution would be given by

$$\pi_c^{s+1} = \frac{\sum_i C_i(\boldsymbol{\eta}^s)}{\sum_i \sum_c C_i(\boldsymbol{\eta}^s)}, \ c = 1, \ldots, C \tag{7}$$

2. For the first iteration, starting values have to be used for the densities that enter the model. Note that these starting values must be different in every class; otherwise, the recursion estimates the same set of parameters for all the classes.
3. Differently from the $\boldsymbol{\beta_c}$s, the vectors $\boldsymbol{\alpha_c}$, $c = 1, 2, \ldots, C$, are jointly estimated. This is necessary to ensure that $\sum_c \pi_c = 1$.

where the updated conditional probabilities—$C_i(\boldsymbol{\eta}^{s+1})$—are computed by means of the updated values of $\boldsymbol{\beta}_c$, $c = 1, 2, \ldots, C$, and the previous values of the class shares.

5. Once the conditional probabilities of class membership have been updated—either in models with or without covariates on $\boldsymbol{z_i}$—the recursion can start again from step 3 until convergence.

4 A Stata routine for the EM estimation of mixed logit models with discrete mixing distributions

In this section, I show how the EM algorithm outlined above can be coded into Stata. I propose a detailed step-by-step procedure that can be easily implemented in a simple do-file and that works with accessible data from Huber and Train (2001) on household's choice of electricity supplier. Note that this is the same dataset used by Hole (2007) for an application of his Stata command `mixlogit`, which performs the estimation of parametric mixed logit models via simulated maximum likelihood.

The data contain information on 100 residential electricity customers who were asked up to 12 choice experiments.[4] In each experiment, the customer was asked which of the four suppliers he or she would prefer among four hypothetical electricity suppliers, and the following characteristics of each offer were stated:

- The price of the contract (in cents per kWh) whenever the supplier offers a contract with a fixed rate (`price`)
- The length of contract that the supplier offered, expressed in years (`contract`)
- Whether the supplier is a local company (`local`)
- Whether the supplier is a well-known company (`wknown`)
- Whether the supplier offers a time-of-day rate instead of a fixed rate (`tod`)
- Whether the supplier offers a seasonal rate instead of a fixed rate (`seasonal`)

Each customer is identified by the variable `pid`. For each customer, the variable `gid` identifies a given choice occasion, while the dummy variable `y` identifies the stated choice in each choice occasion.

Because no individual-level variables are included in this database, I also simulate a customer-specific variable—`_x1`—to show how to fit a model with demographics in the probability of class membership. Hence the data setup required for fitting the model is as follows:

4. Because some customers stopped before answering all 12 experiments, there is a total of 1,195 choice occasions in the sample.

```
. use http://fmwww.bc.edu/repec/bocode/t/traindata.dta
. set seed 1234567890
. by pid, sort: egen _x1=sum(round(rnormal(0.5),1))
. list in 1/12, sepby(gid)
```

	y	price	contract	local	wknown	tod	seasonal	gid	pid	_x1
1.	0	7	5	0	1	0	0	1	1	16
2.	0	9	1	1	0	0	0	1	1	16
3.	0	0	0	0	0	0	1	1	1	16
4.	1	0	5	0	1	1	0	1	1	16
5.	0	7	0	0	1	0	0	2	1	16
6.	0	9	5	0	1	0	0	2	1	16
7.	1	0	1	1	0	1	0	2	1	16
8.	0	0	5	0	0	0	1	2	1	16
9.	0	9	5	0	0	0	0	3	1	16
10.	0	7	1	0	1	0	0	3	1	16
11.	0	0	0	0	1	1	0	3	1	16
12.	1	0	0	1	0	0	1	3	1	16

Section 4.1 presents the steps for fitting a model with covariates on the class share probabilities. Alternatively, the optimization of the algorithm when only a constant term is included among the class probabilities is presented in section 4.2.

4.1 A model with covariates on the class share probabilities

To present a flexible routine, we will work with global variables so that the code can be easily adapted to other databases. The dependent variable is called `$depvar`; the list of covariates that enter the probability of the observed choice is called `$varlist`; the list of variables that enter the grouped-data log likelihood is called `$varlist2`; the variable that identifies the panel dimension—that is, the choice makers—is called `$id`; and the variable that defines the choice situations for each choice maker is called `$group`. We also define the number of latent classes, `$nclasses`, and the number of maximum iterations, `$niter`.

```
. **(1) Set the estimation framework**
. global depvar "y"
. global varlist "price contract local wknown tod seasonal"
. global varlist2 "_x1"
. global id "pid"
. global group "gid"
. global nclasses "2"
. global niter "50"
```

Starting values are computed by randomly splitting the sample into C different subsamples—one for each class—and fitting a separate `clogit` for each of them.[5] After each `clogit` estimation, we use the command `predict` to obtain the probability of every alternative in each class, and we store them in the variables `_l1`, `_l2`, ..., `_l`C. As for the starting values for the probability of class membership, we simply define equal shares, that is, $1/C$:

```
. **(2) Split the sample**
. by $id, sort: generate double _p=runiform() if _n==_N
. by $id, sort: egen double _pr=sum(_p)
. local prop 1/$nclasses
. generate double _ss=1 if _pr<=`prop´
. forvalues s=2/$nclasses {
  2. replace _ss=`s´ if _pr>(`s´-1)*`prop´ & _pr<=`s´*`prop´
  3. }
. **(3) Get starting values for both the beta coefficients and the class shares**
. forvalues s=1/$nclasses {
  2. generate double _prob`s´=1/$nclasses
  3. clogit $depvar $varlist if _ss==`s´, group($group) technique(nr dfp)
  4. predict double _l`s´
  5. }
```

In what follows, the steps to calculate the conditional probabilities of (4) from these starting values are presented.

First, for each latent class, we multiply the variables `_l1`, `_l2`, ..., `_l`C by the dummy variable that identifies the observed choice in each choice situation, that is, `$depvar`. Note that this allows storing the probabilities of the observed choice for each class.

Second, for each latent class, we multiply the probabilities of the observed choices in each choice situation to obtain the probability of the agent's sequence of choices:[6]

```
. **(4) Compute the probability of the agent´s sequence of choices**
. forvalues s=1/$nclasses {
  2. generate double _kbb`s´=_l`s´*$depvar
  3. recode _kbb`s´ 0=.
  4. by $id, sort: egen double _kbbb`s´=prod(_kbb`s´)
  5. by $id, sort: replace _kbbb`s´=. if _n!=_N
  6. }
```

Third, we construct the denominator of (4), that is, the unconditional choice probability for each choice maker. This is done by computing a weighted average of the probabilities of the agent's sequence of choices in each class, with weights given by the class shares, that is, the variables `_prob1`, `_prob2`, ..., `_prob`C:

5. If the same starting values were used for all the classes, the EM algorithm would perform the same computations for each class and return the same results at each iteration.
6. This last step is done through the user-written command `_gprod`. Type `findit gprod` and install the package `dm71`.

```
. **(5) Compute the choice probability**
. generate double _den=_prob1*_kbbb1
. forvalues s=2/$nclasses {
  2. replace _den=_den+_prob`s´*_kbbb`s´
  3. }
```

Finally, we compute the ratios defined in (4) and rearrange them to create individual-level variables. These are the conditional probabilities of class membership as defined in the previous section:

```
. **(6) Compute the conditional probabilities of class membership**
. forvalues s=1/$nclasses {
  2. generate double _h`s´=(_prob`s´*_kbbb`s´)/_den
  3. by $id, sort: egen double _H`s´=sum(_h`s´)
  4. }
```

Before starting the loop that iterates the EM recursion until convergence, we need to specify a Stata `ml` command that performs the estimation of the grouped-data model defined in (6):[7]

```
. **(7) Provide Stata with the ML command for the grouped-data model**
. by $group, sort: generate _alt=sum(1)
. summarize _alt
. by $id, sort: generate double _id1=1 if _n<=r(mean)
. generate _con=1
. program logit_lf
.   args lnf a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18 a19 a20
.   tempvar denom
.   generate double `denom´=1
.   forvalues c=2/$nclasses {
    2. replace `denom´=`denom´+exp(`a`c´´)
    3. }
.   replace `lnf´=_H1*ln(1/`denom´) if $depvar==1 & _id1==1
.   forvalues c=2/$nclasses  {
    2. replace `lnf´=`lnf´+_H`c´*ln(exp(`a`c´´)/`denom´) if $depvar==1 & _id1==1
    3. }
.   replace `lnf´=0 if `lnf´==.
. end
```

Here it is worth noting that following (6), we set to 0 one vector of parameters for identification purposes.

We now present the loop that repeats the steps above until convergence:

```
. local i=1
. while `i´<= $niter {
```

7. The `ml` command presented in the following lines allows for up to 20 latent classes. However, the routine can be easily modified to account for a different number of classes.

To start, fit again the C `clogit` models—one for each class—using the conditional probabilities of class membership as weights. Then recompute the probability of the chosen alternative in each choice occasion to update probabilities of the agent's sequence of choices:

```
. **(8) Update the probability of the agent´s sequence of choices**
. capture drop _l*  _kbbb*
. forvalues s=1/$nclasses {
  2. clogit $depvar $varlist [iw=_H`s´], group($group) technique(nr dfp)
  3. predict double _l`s´
  4. replace _kbb`s´=_l`s´*$depvar
  5. recode _kbb`s´ 0=.
  6. by $id, sort: egen double _kbbb`s´=prod(_kbb`s´)
  7. by $id, sort: replace _kbbb`s´=. if _n!=_N
  8. }
```

Now launch the `ml` model for the estimation of the grouped-data log likelihood, and use the active estimates to update the class share probabilities:

```
. **(9) Update the class share probabilities:
. global variables="($varlist2)"
. forvalues s=3/$nclasses  {
  2. global variables="$variables ($varlist2)"
  3. }
. ml model lf logit_lf $variables, max
. capture drop _denom _a*
. generate double _denom= 1
. forvalues c=2/$nclasses {
  2. local k = `c´ - 1
  3.   predict double _a`c´, eq(eq`k´)
  4.   replace _denom=_denom+exp(_a`c´)
  5. }
. replace _prob1=1/_denom
. forvalues c=2/$nclasses {
  2. replace _prob`c´=exp(_a`c´)/(_denom)
  3. }
```

Once the class share probabilities have been updated, the next step requires updating both the choice probability (that is, the variable `_den`) and the conditional probabilities of class membership (that is, the variables `_H*`):

```
. **(10) Update the choice probability**
. replace _den=_prob1*_kbbb1
. forvalues s=2/$nclasses {
  2. replace _den=_den+_prob`s´*_kbbb`s´
  3. }
. **(11) Update the conditional probabilities of class membership**
. drop _H*
. forvalues s=1/$nclasses {
  2. replace _h`s´=(_prob`s´*_kbbb`s´)/_den
  3. by $id, sort: egen double _H`s´=sum(_h`s´)
  4. }
```

Now that both the parameters and the conditional probabilities have been updated, the routine computes the new log likelihood and restarts the loop until convergence.

As Train (2008) points out, convergence of EM algorithms for nonparametric estimation is still controversial and constitutes an area of future research. Here we follow Train (2008) and Weeks and Lange (1989) and define convergence when the percentage variation in the log-likelihood function from one iteration to the other is sufficiently small. In what follows, for example, the routine automatically stops the internal loop before the selected number of iterations, provided that the log likelihood has not changed more than 0.0001% over the last five iterations:[8]

```
. **(12) Update the log likelihood**
. capture drop _sumll
. egen _sumll=sum(ln(_den))
. **(13) Check for convergence**
. sum _sumll
. global z=r(mean)
. local _sl`i´=$z
. if `i´>=6 {
.   local a=`i´-5
.   if (-(`_sl`i´´ - `_sl`a´´)/`_sl`a´´<= 0.0001) {
.      continue, break
. }
. }
. **(14) If not converged, display the log likelihood and restart the loop**
. display as green "Iteration " `i´ ": log likelihood = " as yellow $z
. local i=`i´ +1
. }
```

When convergence is achieved, results can be displayed by typing

```
. forvalues s=1/$nclasses {
  2. clogit $depvar $varlist [iw=_H`s´], group($group)
  3. }
```

4.2 A model without covariates on the class shares probabilities

If the model does not include demographics on the class share probabilities, the maximization of the grouped-data log likelihood can be avoided, because its solution can be provided analytically by following (7). This is important because the maximization of the grouped-data model slows down the overall estimation process, which could become time-consuming when the number of latent classes is high. In this case, the EM algorithm outlined in the previous section can be optimized by simply substituting the ninth step of the loop with the following lines:

8. Such a small value is advisable because—as discussed in Dempster, Laird, and Rubin (1977)—EM algorithms may move very slowly toward the maximum.

```
**(9-bis) Update class share probabilities by means of (13)**
. forvalues s=1/$nclasses {
  2. capture drop _nums`s´
  3. egen double _nums`s´=sum(_h`s´)
  4. }
. capture drop _dens
. generate double _dens=_nums1
. forvalues s=2/$nclasses {
  2. replace _dens=_dens+_nums`s´
  3. }
. forvalues s=1/$nclasses {
  2. replace _prob`s´=nums`s´/dens
  3. }
```

Subsequently, the loop continues as before from step 10 till the end.

5 Application

For our application, we use the data illustrated in the previous section. To begin with, a typical conditional logit model is fit with the `clogit` command:

```
. use http://fmwww.bc.edu/repec/bocode/t/traindata.dta, clear
. clogit y price contract local wknown tod seasonal, group(gid)
Iteration 0: log likelihood = -1379.3159
  (output omitted)
Iteration 4: log likelihood = -1356.3867
Conditional (fixed-effects) logistic regression   Number of obs   =       4780
                                                  LR chi2(6)      =     600.47
                                                  Prob > chi2     =     0.0000
Log likelihood = -1356.3867                       Pseudo R2       =     0.1812
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
price	-.6354853	.0439523	-14.46	0.000	-.7216302	-.5493403
contract	-.13964	.0161887	-8.63	0.000	-.1713693	-.1079107
local	1.430578	.0963826	14.84	0.000	1.241672	1.619485
wknown	1.054535	.086482	12.19	0.000	.8850338	1.224037
tod	-5.698954	.3494016	-16.31	0.000	-6.383769	-5.01414
seasonal	-5.899944	.35485	-16.63	0.000	-6.595437	-5.204451

From the results above, we can see that—on average—customers prefer lower prices, shorter contract lengths, fixed-rate plans, and a local and well-known company. Now we use the command `mixlogit` from Hole (2007) to fit a parametric mixed logit model with independent, normally distributed coefficients:[9]

9. The model is fit via simulated maximum likelihood with 300 Halton draws.

```
. mixlogit y, id(pid) group(gid) rand(price contract local wknown tod seasonal)
> nrep(300)
Iteration 0:   log likelihood = -1249.8219  (not concave)
  (output omitted)
Iteration 7:   log likelihood = -1101.6085
Mixed logit model                                 Number of obs   =       4780
                                                  LR chi2(6)      =     509.56
Log likelihood = -1101.6085                       Prob > chi2     =     0.0000
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Mean						
price	-1.004329	.0721185	-13.93	0.000	-1.145679	-.8629798
contract	-.2274985	.047386	-4.80	0.000	-.3203735	-.1346236
local	2.208746	.2439681	9.05	0.000	1.730578	2.686915
wknown	1.656329	.1707167	9.70	0.000	1.32173	1.990927
tod	-9.364151	.5858618	-15.98	0.000	-10.51242	-8.215883
seasonal	-9.496181	.5792009	-16.40	0.000	-10.63139	-8.360968
SD						
price	.2151655	.0311095	6.92	0.000	.154192	.2761389
contract	.384136	.044778	8.58	0.000	.2963728	.4718992
local	1.788806	.2370063	7.55	0.000	1.324282	2.25333
wknown	1.185838	.1731652	6.85	0.000	.8464401	1.525235
tod	1.6553	.2094545	7.90	0.000	1.244777	2.065824
seasonal	-1.119371	.2836182	-3.95	0.000	-1.675252	-.5634893

As can be seen from the maximized log likelihood, we can reject the conditional logit specification in favor of a random coefficient model. Moreover, the magnitude of the coefficients is significantly different when compared with the estimates from `clogit`.[10]

We now fit a nonparametric mixed logit model by means of the EM algorithm outlined in the previous section. Following Greene and Hensher (2003) and Train (2008), the choice of the appropriate number of classes is made by means of some information criteria. Here we opt for the Bayesian information criterion and the consistent Akaike information criterion (CAIC), which penalize more heavily those models with a large number of parameters.

10. This is an indication of the bias produced by the irrelevant alternatives property of standard conditional logit models. See Bhat (2000) for this point.

The next table shows—for an increasing number of latent classes—the maximized log likelihood, the number of parameters, the Bayesian information criterion, and the CAIC:

Classes	Log Likelihood	N.param.	CAIC	BIC
Cl.1	-1356.39	6	2746.40	2740.40
Cl.2	-1211.35	13	2495.57	2482.57
Cl.3	-1118.23	20	2348.57	2328.57
Cl.4	-1085.30	27	2321.95	2294.95
Cl.5*	-1040.49	34	2271.55*	2237.55
Cl.6	-1028.56	41	2286.93	2245.93
Cl.7	-1006.37	48	2281.79	2233.79
Cl.8*	-990.24	55	2288.76	2233.76*
Cl.9	-983.64	62	2314.80	2252.80
Cl.10	-978.10	69	2342.96	2273.96
Cl.11	-963.10	76	2352.19	2276.19

From these results, we can infer that the five-class model is optimal according to the CAIC, whilst the Bayesian criterion points to a model with eight latent classes. The next table summarizes the fit coefficients for a model with eight latent classes:

Variable	Class1	Class2	Class3	Class4	Class5
price	-0.910	-0.737	-0.488	-2.110	-0.642
contract	-0.438	0.218	-0.592	-0.662	0.096
local	0.370	2.416	0.782	0.717	2.186
wknown	0.369	2.840	0.710	0.241	1.207
tod	-8.257	-6.690	-4.132	-14.191	-3.836
seasonal	-6.440	-7.213	-6.560	-17.207	-4.052
Prob	0.120	0.097	0.091	0.070	0.096

Variable	Class6	Class7	Class8	W.Average
price	-1.208	-1.533	-0.082	-0.946
contract	-0.198	-0.409	-0.156	-0.269
local	6.578	0.621	4.937	2.369
wknown	5.103	0.930	3.444	1.918
tod	-14.847	-16.007	-1.088	-9.003
seasonal	-15.334	-14.818	-1.060	-9.058
Prob	0.111	0.236	0.178	-

Results show that the parameters are rather different from class to class, meaning that there is an extended unobserved heterogeneity in the choice behavior. Interestingly, the weighted averages of the parameters are close to the correspondent values obtained from the parametric estimation with `mixlogit`.

As for the EM recursion, the estimation of the eight-class model took 49 iterations before reaching convergence, that is, about one minute on our standard-issue PC with a dual-core processor and 4 GB RAM. However, the routine is already close to the maximum at the 11th iteration, that is, after less than 10 seconds.

This is a common feature of EM algorithms, and it actually suggests another application of the EM procedure proposed in this contribution. In fact, the routine could also be used to obtain good starting values for the estimation of latent-class models with gradient-based algorithms. This could be particularly useful either to speed up the estimation process or to avoid convergence problems when fitting models with a high number of latent classes.

6 Acknowledgments

I am grateful to Kenneth Train for his helpful comments on the implementation of the EM algorithm. Thanks also to an anonymous referee for relevant comments and suggestions on how to improve both this article and the estimation routine. Thanks to Hong il Yoo for his careful check on the Stata code.

7 References

Bhat, C. R. 2000. Automatic modeling methods for univariate series. In *Handbook of Transport Modelling*, ed. D. A. Hensher and K. J. Button, 71–90. Oxford: Elsevier.

Boyles, R. A. 1983. On the convergence of the EM algorithm. *Journal of the Royal Statistical Society, Series B* 45: 47–50.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39: 1–38.

Greene, W. H., and D. A. Hensher. 2003. A latent class model for discrete choice analysis: Contrasts with mixed logit. *Transportation Research, Part B* 37: 681–698.

Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.

Huber, J., and K. Train. 2001. On the similarity of classical and Bayesian estimates of individual mean partworths. *Marketing Letters* 12: 259–269.

McFadden, D. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zerembka, 105–142. New York: Academic Press.

McFadden, D., and K. Train. 2000. Mixed MNL models for discrete response. *Journal of Applied Econometrics* 15: 447–470.

Train, K. E. 2003. *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press.

———. 2008. EM algorithms for nonparametric estimation of mixing distributions. *Journal of Choice Modelling* 1: 40–69.

Weeks, D. E., and K. Lange. 1989. Trials, tribulations, and triumphs of the EM algorithm in pedigree analysis. *Journal of Mathematics Applied in Medicine & Biology* 6: 209–232.

Wu, C. F. J. 1983. On the convergence properties of the EM algorithm. *Annals of Statistics* 11: 95–103.

About the author

Daniele Pacifico works with the Italian Department of the Treasury in Rome, Italy, and is a member of the Centre for the Analysis of Public Policies in Modena, Italy.