1. Introduction

This code estimates, nonparametrically the CDF of the random coefficient logit distribution. The true distribution is a mixture of two, correlated, bivariate normals.

The approximating grid of points comes from a Halton sequence and is stored in gridR. The estimated weights are saved in the vector theta.

The estimated values of the CDF of the random coefficients are evaluated over (another, evenly spaced) grid of points. The estimated CDF is saved in the vector estimatedCDFValues.

The confidence intervals for the estimated weights and the estimated CDF are constructed using either Tikhonov regularization (using generalized cross validation to choose the perturbation) or subsampling. Subsampling may not have correct coverage for this application. The confidence intervals from regularization tend to be conservative in simulation studies.

Endpoints for the 95% confidence intervals for theta are stored in leftEndPoints and rightEndPoints. EndPoints for the 95% confidence intervals for estimatedCDFValues are stored in leftEndPointsCDF and rightEndPointsCDF.

Comments should be sent to Jeremy Fox at fox@uchicago.edu.
An Qi contributed to this code.

2. The random coefficients logit model

In the logit, agents $i = 1,\ldots, N$ can choose between $j = 1,\ldots J$ mutually exclusive alternatives. For each agent i, the exogenous variables for choice j is stored in a $K \times 1$ vector $x_{i,j}$ which could include the product characteristics, the price of good j, and the demographics of agent i.
Let $x_i = (x'_{i,1},...,x'_{i,J})$

In the model, there are $r = 1\ldots, R$ types of consumers. The fixed preferences of type r are stored in a $1 \times K$ vector β^r. The proportion of type r in the population is θ^r, which sum to 1.
Let $\theta = (\theta^1,...,\theta^r)$. θ will be estimated using constrained OLS in the code.

For agent i of type r, the utility derived from choosing good j is equal to

$$u_{i,j} = x'_{i,j}\,\beta^r + \varepsilon_{i,j}$$

There is also an outside good with utility $\varepsilon_{i,0}$. Assume that the errors are distributed as Type I extreme value.

Define choice data as

$$y_{i,j} = \begin{cases} 1 \text{ if } u_{i,j} > u_{i,j'} \text{ for all } j' \neq j \\ 0 \quad \text{otherwise} \end{cases}$$

Because the errors are extreme value, it follows that

$$\Pr(y_{i,j} = 1 \mid x_i) = \sum_{r=1}^{R} \theta^r \frac{\exp(x'_{i,j}\,\beta^r)}{1 + \sum_{j'=1}^{J} \exp(x'_{i,j}\,\beta^r)}$$

Given $x_{i,j}$, $y_{i,j}$, and β^r, the code will estimate the parameters θ^r using constrained OLS.

3. Parameters

N represents the number of agents. *J* represents the number of alternatives. *K* represents the number of product characteristics, which is hard-coded in the choice of distribution and should not be changed. *R* is the number of grid points in the approximation. standardErrorMethod selects the method used to construct confidence intervals: 0 for Tikhonov regularization, 1 for subsampling.

4. Generate fake data

obj is a mixture of two, correlated, bivariate normals, with distribution as specified in the code. It is the distribution that is to be estimated by the model. *betas* is a N × K array, with values drawn randomly from *obj*. *xdata* is a N × J × K array, with values drawn randomly from a uniform distribution over the interval (0,1).

In the model, the probability that an agent *i* will choose alternative *j* is given by

$$\frac{\exp(xdata(i, j, :) \cdot betas(i, :))}{1 + \sum_{j'=1}^{J} \exp(xdata(i, j', :) \cdot betas(i, :))}$$

For each *i* and *j*, the choice probabilities are calculated and stored in *choiceProbs*, an N × J array. Using *choiceProbs* and a number drawn randomly from a uniform distribution over the interval (0,1), a choice for each *i* and *j* is generated. These choices are stored in *choiceData*, an N × J array. These *choiceData* will be the dependent data in the regression.

5. Estimation

gridR is an R × K array of points generated using a Halton Sequence that covers [-10,10] × [-10,10]. *gridR* will be the approximating grid of points for the estimated weights *theta*. For each *i* and *j* and for each point *r* on the grid, choice probabilities are calculated using the equation

$$\frac{\exp(xdata(i, j, :) \cdot gridR(r, :))}{1 + \sum_{j'=1}^{J} \exp(xdata(i, j', :) \cdot gridR(r, :))}.$$

These choice probabilities are stored in *choiceProbsR*, an N × R × J array.

The data in *choiceProbsR* is then rearranged into an NJ × R array, with each consumer *i* forming a cluster of size J. The rearranged *choiceProbsR* is stored in *regData*. Likewise, *choiceData* is rearranged into an NJ × 1 array *dependentData*.

The regression equation is $dependentData = regData \cdot theta + residuals$,
where *theta* contains the estimated weights, constrained such that
$\sum_{r=1}^{R} theta(r) = 1$

Using lsqlin, a *theta* is generated.

This *theta* is now used to create an estimated CDF. The grid of points used to evaluate the CDF is an evenly spaced grid of points covering [-10,10] × [-10,10]. This grid is stored in *combinedIndex*, a D × K array, where D is the number of grid points.

For each grid point *combinedIndex(d,:)*, where $1 \le d \le$ D, the CDF evaluated at that point is
$\sum_{r=1}^{R} theta(r) \cdot 1[gridR(r,:) \le combinedIndex(d,:)]$, where
$1[gridR(r,:) \le combinedIndex(d,:)] = 1$ if $gridR(r,k) \le combinedIndex(d,k)$ for all k, $1 \le$ k $\le$ K.

The estimated CDF values are stored in *estimatedCDFValues*.

6. Constructs confidence intervals using Tikhonov regularization

This code executes if *standardErrorMethod*=0.

To calculate the covariance matrix for *theta*, Tikhonov regularization is needed to correct the near-singularity of *regData*. The Tikhonov parameter *alpha* is calculated using generalized cross validation.

Define the function

$$V(\lambda) = \frac{(1/NJ) \parallel (I - A(\lambda) \cdot dependentData \parallel^2}{[(1/NJ) \cdot trace(I - A(\lambda))]^2},$$

where *I* is the identity matrix and $A(\lambda) = regData \cdot (regData^T \cdot regData + NJ\lambda \cdot I)^{-1} \cdot regData^T$

The minimum of this function provides the optimum Tikhonov parameter, which is stored in *alpha*.

The residuals are calculated and are stored in *residuals*.

Let $sumMat = \sum_{i=1}^{N} X_i^T \cdot e_i \cdot e_i^T \cdot X_i$,

where for each *i*, X_i is the matrix composed of the ((i–1)·J+1)th to (i·J)th rows of *regData* and e_j is the column vector composed of the ((i–1)·J+1)th to (i·J)th terms of the residuals.

The covariance matrix adjusted for clustering and heteroskedasticity is

$$(regData^T \cdot regData + alpha \cdot I)^{-1} \cdot sumMat \cdot (regData^T \cdot regData + alpha \cdot I)^{-1}.$$

and is stored in *varianceMat*.

The standard errors are calculated by taking the square root of the diagonal of *varianceMat* and are stored in *stdErrors*.

For each point *r* of *theta*, the 95% confidence interval is

$$[0,1] \bigcap [theta(r) - 1.96 \cdot stdErrors(r), theta(r) + 1.96 \cdot stdErrors(r)].$$

Using the covariance matrix for *theta*, the variance for the estimated CDF can be constructed.

For each grid point *combinedIndex(d,:)*, $1 \le d \le D$, the variance of the CDF evaluated at that point is

$$\sum_{r=1}^{R} varianceMat(r,r) \cdot 1[gridR(r,:) \le combinedIndex(d,:)] +$$
$$2\sum_{r=1}^{R-1} \sum_{s=r+1}^{R} varianceMat(r,s) \cdot 1[gridR(r,:) \le combinedIndex(d,:)] \cdot 1[gridR(s,:) \le combinedIndex(d,:)]$$

The variances for the estimated CDF are stored in CDFVariances. The standard errors for the CDF are calculated by taking the square root of these values and are stored in CDFStdErrors.

For each point *d* of *estimatedCDFValues*, the 95% confidence interval is

$$[0,1] \bigcap [estimatedCDFValues(d) - 1.96 \cdot CDFStdErrors(d),$$
$$estimatedCDFValues(d) + 1.96 \cdot CDFStdErrors(d)]$$

7. Constructs confidence intervals using subsampling

This code executes if *standardErrorMethod*=1.

Each subsample contains regression data for *N/4* agents, with *J* regression points for each agent. If *N/4* is not a natural number, *N/4* will be rounded to the nearest natural number. Each subsample will contain *B=round(N/4)·J* regression points, with *M=N−B/J+1* subsamples total.

For each subsample *i*, a *regDatai* consisting of the ((i−1)·J+1)th to ((i−1)·J+B)th rows of *regData* and a *dependendDatai* consisting of the ((i−1)·J+1)th to ((i−1)·J+B)th values of *dependentDatai* are created. *regDatai* and *dependentDatai* are then used to run a regression,

generating estimated weights and an estimated CDF. This process is repeated *M* times, generating *M* sets of estimated weights and *M* estimated CDFs. The estimated weights are stored in *thetaMat,* an R × M array. The estimated CDFs are stored in *CDFMat,* a D × M array.

Confidence intervals are then constructed using the point estimates for the subsamples and the point estimate for the entire sample.

For each point *r* of *theta,* the 95% confidence interval is

$[0,1] \bigcap [theta(r) - (1/\sqrt{NJ}) \cdot c1, theta(r) - (1/\sqrt{NJ}) \cdot c2]$, where

c1 is the 97.5% quantile of $(\sqrt{B} \cdot (thetaMat(r,:) - theta(r)))$ for the M estimated *thetas,* and
c2 is the 2.5% quantile of $(\sqrt{B} \cdot (thetaMat(r,:) - theta(r)))$ for the M estimated *thetas.*

For each point *d* of *estimatedCDFValues,* the 95% confidence interval is

$[0,1] \bigcap [estimatedCDFValues(d) - (1/\sqrt{NJ}) \cdot c1, estimatedCDFValues(d) - (1/\sqrt{NJ}) \cdot c2]$, where

c1 is the 97.5% quantile of $(\sqrt{B} \cdot (CDFMat(d,:) - estimatedCDFValues(d)))$ for the M estimated CDFs, and
c2 is the 2.5% quantile of $(\sqrt{B} \cdot (CDFMat(d,:) - estimatedCDFValues(d)))$ for the M estimated CDFs.