Estimating a latent class model

Jasper Knockaert

DCA Workshop
23 June 2017

- Classical multinomial logit choice model:

$$U_{in} = \beta \cdot X_{in} + \epsilon_{in}$$

- Mixed logit model: β distributed

$$U_{in} = \beta_n \cdot X_{in} + \epsilon_{in}$$

- Latent class model: special case of distribution
  - A vector β_c for each class c
  - Class membership is *latent*
  - Class membership model defined by probability mass function $P_5(c|X_n; \gamma, \Sigma_\mu)$
  - Choice probability for alternative y:

$$\sum_c \left[P_5(c|X_n^*; \gamma, \Sigma_\mu) \cdot P_4(y_n|U_{1n}, \ldots, U_{jn}; \beta_c, \Sigma_\epsilon) \right]$$

  - Main advantage of latent class: closed-form expression for choice likelihood

Main issue with latent class model is abundance of local optima:

var	description	1-class	2-class	3-class	4-class
	# estimations	11	132	1048	3038
s	# converged estimations	11	132	1034	2858
p	# unique optima	1	16	91	614
H	stopping criterion	0.00049	0.00033	0.000012	0.0096

for each latent class 10 utility function coefficients and 1 class constant (with the constant for one class arbitrarily fixed to 0)

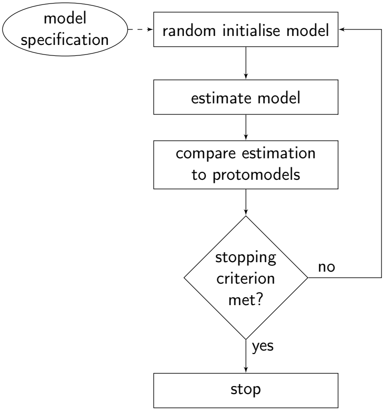
22174 revealed preference observations by 544 respondents (work with Stefanie Peer)

Estimation with local optima:

- Heuristic optimisation:
  - Calculate LL in more or less randomly sampled starting points ("informed" grid search)
  - Identify most promising starting point
  - Run classical optimisation (i.e. biogeme)
  - Hole and Yoo (2017) The use of heuristic optimization algorithms to facilitate maximum simulated likelihood estimation of random parameter logit models, Appl. Statist.
    - population-based heuristic optimization algorithms: DE and PSO algorithms

Estimation with local optima:

- Heuristic optimisation:
  - Calculate LL in more or less randomly sampled starting points ("informed" grid search)
  - Identify most promising starting point
  - Run classical optimisation (i.e. biogeme)
  - Hole and Yoo (2017) The use of heuristic optimization algorithms to facilitate maximum simulated likelihood estimation of random parameter logit models, Appl. Statist.
    - population-based heuristic optimization algorithms: DE and PSO algorithms
- My approach: use random initialisation and run optimisation from each starting point.



- Generic methodology for random initialisation
- Challenge is in clustering the estimation results around optima
- Heuristic as stopping criterium

- To cluster we define a similarity indicator:

$$S(\beta,\gamma) = \prod_{k=1\ldots K}\left[1 - \mathrm{erf}\frac{|\beta_k - \gamma_k|}{\sqrt{2\left(t_{\beta_k}^2 + t_{\gamma_k}^2\right)}}\right]$$

- Stopping criterium:

$$H = \left[\frac{p}{p+1}\right]^s$$

Illustration of optima found 2-class version of the RP model in previous slide (132 properly converged estimations)

optimum	LL	#estimations
1	−52599.378	13
2	−52601.516	25
3	−52628.548	21
4	−52677.798	1
5	−52909.474	19
6	−52910.33	12
7	−52915.372	14
8	−52932.691	1
9	−52934.631	3
10	−52935.529	1
11	−52936.441	12
12	−52936.482	1
13	−52938.413	2
14	−52973.077	1
15	−53038.479	4
16	−53042.439	2

- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible

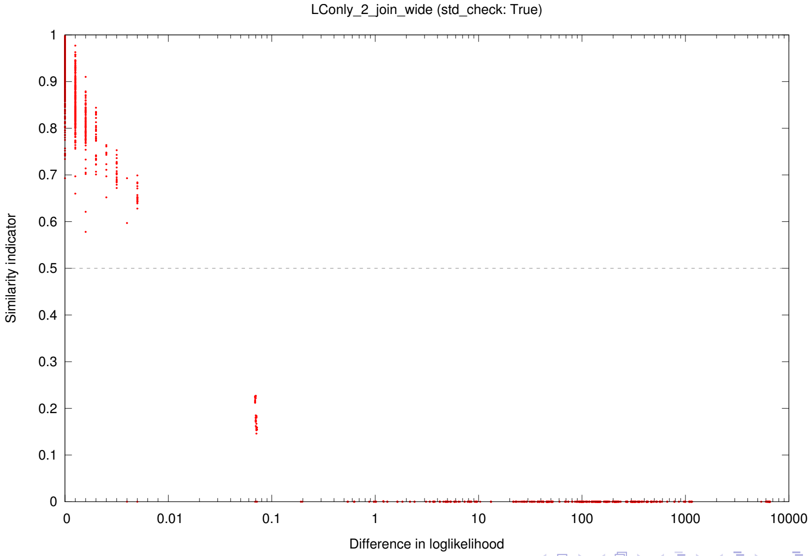
- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible
- Estimation: pythonbiogeme

- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible
- Estimation: pythonbiogeme
- Comparison: a python script patches the estimation html output, and compares it to "protomodels" in sqlite database; after comparison the sqlite database is updated; support for behaviourally identical permutations of models (e.g. unordered latent classes)

- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible
- Estimation: pythonbiogeme
- Comparison: a python script patches the estimation html output, and compares it to "protomodels" in sqlite database; after comparison the sqlite database is updated; support for behaviourally identical permutations of models (e.g. unordered latent classes)
- Evaluation of stopping criterium: python script reads parameters from sqlite database

- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible
- Estimation: pythonbiogeme
- Comparison: a python script patches the estimation html output, and compares it to "protomodels" in sqlite database; after comparison the sqlite database is updated; support for behaviourally identical permutations of models (e.g. unordered latent classes)
- Evaluation of stopping criterium: python script reads parameters from sqlite database
- Parallelisation is possible using a network share for the files (sqlite database supports concurrency); I implemented PBS support for managing parallel estimations on a cluster

- Random initialisation: bash script substitutes random numbers in model template; results are saved as genuine pythonbiogeme model file so reproducible
- Estimation: pythonbiogeme
- Comparison: a python script patches the estimation html output, and compares it to "protomodels" in sqlite database; after comparison the sqlite database is updated; support for behaviourally identical permutations of models (e.g. unordered latent classes)
- Evaluation of stopping criterium: python script reads parameters from sqlite database
- Parallelisation is possible using a network share for the files (sqlite database supports concurrency); I implemented PBS support for managing parallel estimations on a cluster
- Supporting tools, e.g. to rebuild sqlite database but also to visualise clustering (with gnuplot)


LConly_2_join_wide (std_check: True)
Similarity indicator
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.01
0.1
1
10
100
1000
10000
Difference in loglikelihood

Does it actually matter?

Does it actually matter?

optimum	LL	#estimations	value of traveltime	
			high	low
1	-52599.378	13	47.2	10.5
2	-52601.516	25	42.9	10.8
3	-52628.548	21	40.3	11.1
4	-52677.798	1	40.6	16.3
5	-52909.474	19	30.6	10.5

Does it actually matter?

optimum	LL	#estimations	value of traveltime	
			high	low
1	−52599.378	13	47.2	10.5
2	−52601.516	25	42.9	10.8
3	−52628.548	21	40.3	11.1
4	−52677.798	1	40.6	16.3
5	−52909.474	19	30.6	10.5

optimum	LL	#estimations	value of morning SDE	
			high	low
1	−52599.378	13	43.4	3.0
2	−52601.516	25	37.8	3.2
3	−52628.548	21	32.7	3.4
4	−52677.798	1	46.7	2.8
5	−52909.474	19	11.9	5.0

Some observations (and caveats)

- The presence of local optima is wildly variable; some latent class models only have one optimum even with three of four classes

Some observations (and caveats)

- The presence of local optima is wildly variable; some latent class models only have one optimum even with three of four classes
- Sampling matters (interval, distribution)

Some observations (and caveats)

- The presence of local optima is wildly variable; some latent class models only have one optimum even with three of four classes
- Sampling matters (interval, distribution)
- The cutoff points (for the algorithm heuristics) is also variable; (e.g. $H = 0.01$ seems a fine value but with very non-linear "Utility" functions a much lower value is needed)

Some observations (and caveats)

- The presence of local optima is wildly variable; some latent class models only have one optimum even with three of four classes
- Sampling matters (interval, distribution)
- The cutoff points (for the algorithm heuristics) is also variable; (e.g. $H = 0.01$ seems a fine value but with very non-linear "Utility" functions a much lower value is needed)
- Why suboptima?
    - Analytical feature of LC? Conditions?
    - Numerical feature? (I use high-precision version of pythonbiogeme)
    - Misspecification?

Some observations (and caveats)

- The presence of local optima is wildly variable; some latent class models only have one optimum even with three of four classes
- Sampling matters (interval, distribution)
- The cutoff points (for the algorithm heuristics) is also variable; (e.g. $H = 0.01$ seems a fine value but with very non-linear "Utility" functions a much lower value is needed)
- Why suboptima?
  - Analytical feature of LC? Conditions?
  - Numerical feature? (I use high-precision version of pythonbiogeme)
  - Misspecification?
- Report estimation statistics!