Optimization of Discrete Choice Models using first-order methods

Gael Lederrey, Virginie Lurkin, and Michel Bierlaire

Transport and Mobility Laboratory
École Polytechnique Fédérale de Lausanne,
Station 18, CH-1015 Lausanne
gael.lederrey@epfl.ch

1 Introduction

Discrete Choice Models (DCM) are powerful operational tools designed to capture in detail the underlying behavioral mechanisms at the foundation of demand. These disaggregate models have received a great amount of attention in the last years. However, very little work has been done on estimating those models, that is computing the maximum log-likelihood. Many of the recent advances in Machine Learning, and especially since the emergence of Neural Networks, come from incredibly powerful methods for function optimization. Indeed, due to the power of computers and new optimization techniques, we can train Deep Neural Networks on huge datasets efficiently by employing good practice as given by Livni et al. (2014).

With the exponential increase of data availability nowadays, see Wu et al. (2014), the discrete choice modelers have now access to big datasets and it becomes crucial for them to learns how to take advantage of Machine Learning researchers. The majority of DCM software used in practice relies on second-order methods such as the BFGS algorithm. Whilst these methods are powerful, the computation of the Hessian can be prohibitely complex, particularly for large datasets. First-order methods work particularly well with Neural Networks (see Bottou et al. (2016) for an overview of optimization methods for Machine Learning).

In this work, we apply first-order optimization methods to a Multinomial Logit Model. The goal is to understand if it is possible to optimize these convex models without the curvature information from the Hessian. This is the first step towards designing more efficient optimization algorithms for Discrete Choice models.

2 Methodology

We use the *Swissmetro* dataset (Bierlaire et al., 2001) and build a multinomial logit model denoted by $\mathcal{M}$:

$$\begin{aligned} V_{\text{Car}} &= \text{ASC}_{\text{Car}} + \beta_{\text{TT,Car}}\text{TT}_{\text{Car}} + \beta_{\text{C,Car}}\text{C}_{\text{Car}} + \beta_{\text{Senior}}\mathbb{1}_{\text{Senior}} \\ V_{\text{SM}} &= \text{ASC}_{\text{SM}} + \beta_{\text{TT,SM}}\text{TT}_{\text{SM}} + \beta_{\text{C,SM}}\text{C}_{\text{SM}} + \beta_{\text{HE}}\text{HE}_{\text{SM}} + \beta_{\text{Senior}}\mathbb{1}_{\text{Senior}} \\ V_{\text{Train}} &= \text{ASC}_{\text{Train}} + \beta_{\text{TT,Train}}\text{TT}_{\text{Train}} + \beta_{\text{C,Train}}\text{C}_{\text{Train}} + \beta_{\text{HE}}\text{HE}_{\text{Train}} \end{aligned} \tag{1}$$

where $\mathbb{1}_{\text{Senior}}$ is a boolean variable equal to one if the age of the respondent is over 65 years olds, 0 otherwise, C denotes the cost, TT the travel time, and HE the headway for the train and Swissmetro. On this model, we remove all observations with unknown choice, unkown age and non-positive travel time. This gives a total of 9,036 observations.

This model is first estimated with Biogeme (Bierlaire, 2003) to obtain the optimal parameter values and verify that all parameters are significant. The final log-likelihood is -7145.721 and the parameters are given in Table 1.

All parameters are shown to be significant. We replicate the model in Python using the function `minimize` with the method `BFGS` from the package `scipy.optimize`. We obtain similar optimal parameter values to the Biogeme model. These results form our benchmark to test against other optimization methods.

Name	Value	Std err	t-test	p-value
$\mathrm{ASC}_{\mathrm{Car}}$	0	-	-	-
$\mathrm{ASC}_{\mathrm{SM}}$	$7.86 \cdot 10^{-1}$	$6.93 \cdot 10^{-2}$	11.35	0.00
$\mathrm{ASC}_{\mathrm{Train}}$	$9.83 \cdot 10^{-1}$	$1.31 \cdot 10^{-1}$	7.48	0.00
$\beta_{\mathrm{TT,Car}}$	$-1.05 \cdot 10^{-2}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\mathrm{TT,SM}}$	$-1.44 \cdot 10^{-2}$	$6.36 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\mathrm{TT,Train}}$	$-1.80 \cdot 10^{-2}$	$8.65 \cdot 10^{-4}$	-20.78	0.00
$\beta_{\mathrm{C,Car}}$	$-6.56 \cdot 10^{-3}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\mathrm{C,SM}}$	$-8.00 \cdot 10^{-3}$	$3.76 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\mathrm{C,Train}}$	$-1.46 \cdot 10^{-2}$	$9.65 \cdot 10^{-4}$	-15.09	0.00
β_{Senior}	-1.06	$1.16 \cdot 10^{-1}$	-9.11	0.00
β_{HE}	$-6.88 \cdot 10^{-3}$	$1.03 \cdot 10^{-3}$	-6.69	0.00

Table 1. Parameters of the optimized model $\mathcal{M}$ by Biogeme.

3 Preliminary Results

We start by opzimizing model $\mathcal{M}$ with four different algorithms: Gradient Descent, mini-batch Stochastic Gradient Descent (SGD), Adagrad, and SAGA. Gradient Descent is the base algorithm using all observations to perform a step (see Ruder (2016)). The mini-batch SGD uses a batch of observations to compute the gradient and perform a step. We use a batch size of 100 observations. Adagrad is a more advanced algorithm deriving from the mini-batch SGD, see Duchi et al. (2011). This algorithm has been widely used for training Neural Networks. The main contribution of this algorithm is the adaptation of the learning rate to the parameters. The fourth algorithm, SAGA, makes use of variance-reduction techniques and has been shown to perform particularly well on finite sum structure (see Defazio et al. (2014)).

Starting from a zero-vector, we run the four algorithms for 1,000 steps to optimize the log-likelihood of $\mathcal{M}$. The results are given in Fig. 1. All four algorithms perform well in the early steps. The algorithms all plateau in the later steps and do not converge to the optimal value. It is especially visible for the Gradient Descent and the mini-batch SGD algorithms. Adagrad and SAGA are still improving marginally after 1,000 steps. Adagrad performs better than the two basic algorithms due to its adaptive step size.

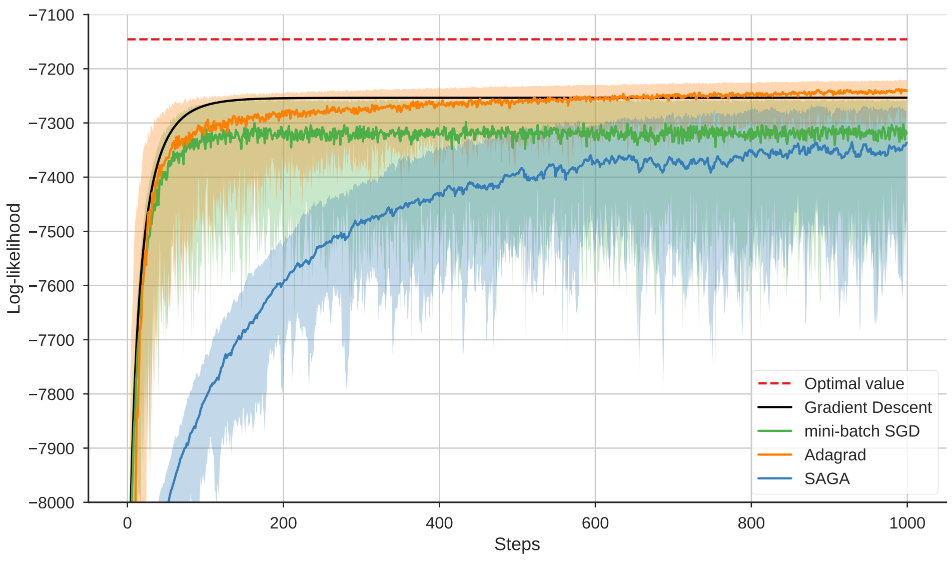

Fig. 1. Optimization of the model $\mathcal{M}$ for different first-order methods. The line corresponds to the average over 100 runs and the transparent part corresponds to the 95% confidence interval.

Table 2 shows the values of the parameters for the steps 250, 500, 750, and 1000 for the Gradient Descent algorithm compared to the optimized values. As we can see, the parameters $\mathrm{ASC}_{\mathrm{SM}}$, $\mathrm{ASC}_{\mathrm{Train}}$, and

β_{Senior} are still far away from their optimal value with Gradient Descent. This is due to the difference in orders of magnitude for the optimal parameter values as shown in Table 1. This presents a maximum value constraint on the step-size to ensure that the algorithm does not diverge for the parameters with small optimal value. This constraint limits the gradient for the optimization and causes the algorithm to plateau. This issue can be addressed by normalizing the values fo the data such that the optimal parameter values are all in the same order of magnitude.

	steps					
Parameters	**250**	**500**	**750**	**1000**	**optimized**	**%Error**
ASC_{Car}	–	–	–	–	0	–
ASC_{SM}	$4.29 \cdot 10^{-4}$	$7.90 \cdot 10^{-4}$	$1.15 \cdot 10^{-3}$	$1.51 \cdot 10^{-3}$	$7.86 \cdot 10^{-1}$	99.81%
$\text{ASC}_{\text{Train}}$	$-9.75 \cdot 10^{-6}$	$6.36 \cdot 10^{-5}$	$1.37 \cdot 10^{-4}$	$2.10 \cdot 10^{-4}$	$9.83 \cdot 10^{-1}$	99.98%
$\beta_{\text{TT,Car}}$	$-1.21 \cdot 10^{-2}$	$-1.21 \cdot 10^{-2}$	$-1.21 \cdot 10^{-2}$	$-1.21 \cdot 10^{-2}$	$-1.05 \cdot 10^{-2}$	15.64%
$\beta_{\text{TT,SM}}$	$-1.23 \cdot 10^{-2}$	$-1.26 \cdot 10^{-2}$	$-1.26 \cdot 10^{-2}$	$-1.26 \cdot 10^{-2}$	$-1.44 \cdot 10^{-2}$	12.92%
$\beta_{\text{TT,Train}}$	$-1.59 \cdot 10^{-2}$	$-1.60 \cdot 10^{-2}$	$-1.60 \cdot 10^{-2}$	$-1.60 \cdot 10^{-2}$	$-1.80 \cdot 10^{-2}$	11.02%
$\beta_{\text{C,Car}}$	$-7.85 \cdot 10^{-3}$	$-8.02 \cdot 10^{-3}$	$-8.03 \cdot 10^{-3}$	$-8.03 \cdot 10^{-3}$	$-6.56 \cdot 10^{-3}$	22.40%
$\beta_{\text{C,SM}}$	$-7.35 \cdot 10^{-3}$	$-7.45 \cdot 10^{-3}$	$-7.46 \cdot 10^{-3}$	$-7.46 \cdot 10^{-3}$	$-8.00 \cdot 10^{-3}$	6.79%
$\beta_{\text{C,Train}}$	$-1.26 \cdot 10^{-2}$	$-1.40 \cdot 10^{-2}$	$-1.40 \cdot 10^{-2}$	$-1.40 \cdot 10^{-2}$	$-1.46 \cdot 10^{-2}$	3.75%
β_{Senior}	$-1.61 \cdot 10^{-4}$	$-3.26 \cdot 10^{-4}$	$-4.92 \cdot 10^{-4}$	$-6.57 \cdot 10^{-4}$	-1.06	99.94%
β_{HE}	$-2.25 \cdot 10^{-3}$	$-2.01 \cdot 10^{-3}$	$-2.00 \cdot 10^{-3}$	$-2.00 \cdot 10^{-3}$	$-6.88 \cdot 10^{-3}$	70.87%
Log-likelihood	-7253.83	-7253.55	-7253.49	-7253.42	-7145.72	1.51%

Table 2. Values of the parameters for different steps of the Gradient Descent algorithm compared to the optimized value found by Biogeme. The last column shows the relative percentage error between the iteration 1000 and the optimized value. Likelihood for zero-vector: -9927.06.

We divide the values of the data for *Cost*, *Time*, and *Headway* by 100 so that all associated β's will be multiplied by 100, therefore normalizing the order of magnitude. The results of the optimization of the normalized model are given in Fig. 2.

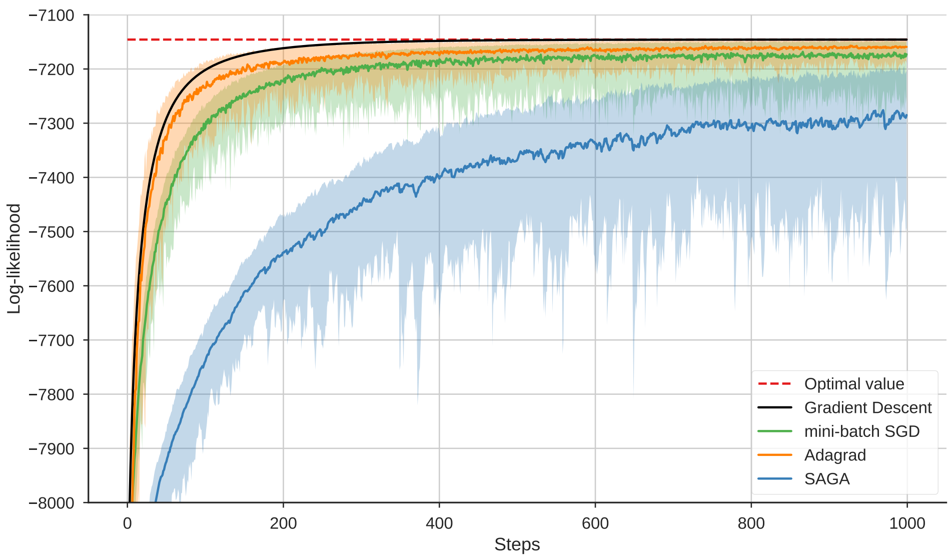

Fig. 2. Optimization of the normalized model $\mathcal{M}$ for different first-order methods. The line corresponds to the average over 100 runs and the transparent part corresponds to the 95% confidence interval.

The four algorithms still encounter a plateau, but they obtain a solution much closer to the optimal solution within 1,000 iterations. However, none of these algorithms converge fully to the optimal solution, as the gradient never decreases enough to reach termination. We show the parameters' values for different steps in Table 3.

	steps					
Parameters	**250**	**500**	**750**	**1000**	**optimized**	**%Error**
$\mathrm{ASC}_{\mathrm{Car}}$	–	–	–	–	0	–
$\mathrm{ASC}_{\mathrm{SM}}$	$6.39 \cdot 10^{-1}$	$7.34 \cdot 10^{-1}$	$7.68 \cdot 10^{-1}$	$7.80 \cdot 10^{-1}$	$7.86 \cdot 10^{-1}$	0.85%
$\mathrm{ASC}_{\mathrm{Train}}$	$4.69 \cdot 10^{-1}$	$8.02 \cdot 10^{-1}$	$9.17 \cdot 10^{-1}$	$9.59 \cdot 10^{-1}$	$9.83 \cdot 10^{-1}$	2.41%
$\beta_{\mathrm{TT,Car}}$	-1.05	-1.05	-1.05	-1.05	-1.05	0.04%
$\beta_{\mathrm{TT,SM}}$	-1.38	-1.43	-1.44	-1.44	-1.44	0.15%
$\beta_{\mathrm{TT,Train}}$	-1.58	-1.72	-1.77	-1.79	-1.80	0.53%
$\beta_{\mathrm{C,Car}}$	$-6.51 \cdot 10^{-1}$	$-6.59 \cdot 10^{-1}$	$-6.57 \cdot 10^{-1}$	$-6.56 \cdot 10^{-1}$	$-6.56 \cdot 10^{-1}$	0.08%
$\beta_{\mathrm{C,SM}}$	$-7.77 \cdot 10^{-1}$	$-7.95 \cdot 10^{-1}$	$-7.99 \cdot 10^{-1}$	$-8.00 \cdot 10^{-1}$	$-8.00 \cdot 10^{-1}$	0.07%
$\beta_{\mathrm{C,Train}}$	-1.40	-1.45	-1.46	-1.46	-1.46	0.00%
β_{Senior}	$-8.94 \cdot 10^{-1}$	-1.04	-1.06	-1.06	-1.06	0.88%
β_{HE}	$-4.41 - 1.05 \cdot 10^{-1}$	$-5.94 \cdot 10^{-1}$	$-6.54 \cdot 10^{-1}$	$-6.76 \cdot 10^{-1}$	$-6.88 \cdot 10^{-1}$	1.76%
Log-likelihood	-7155.32	-7146.76	-7145.85	-7145.74	-7145.72	0.00%

Table 3. Values of the parameters for different steps of the Gradient Descent algorithm compared to the optimized value found by Biogeme on the normalized version of $\mathcal{M}$. The last column shows the relative percentage error between the iteration 1000 and the optimized value. Likelihood for zero-vector: -9927.06.

As we can see in Table 3, whilst Gradient Descent approaches the optimal solution closely, it is not terminated within 1,000 iterations. To improve the convergence rate, the next step is to use a method to find optimal step size. We implemented this using a line search. The Adagrad algorithm cannot be used with a line search, so is not investigated here. Fig. 3 shows the results with the three remaining algorithms.

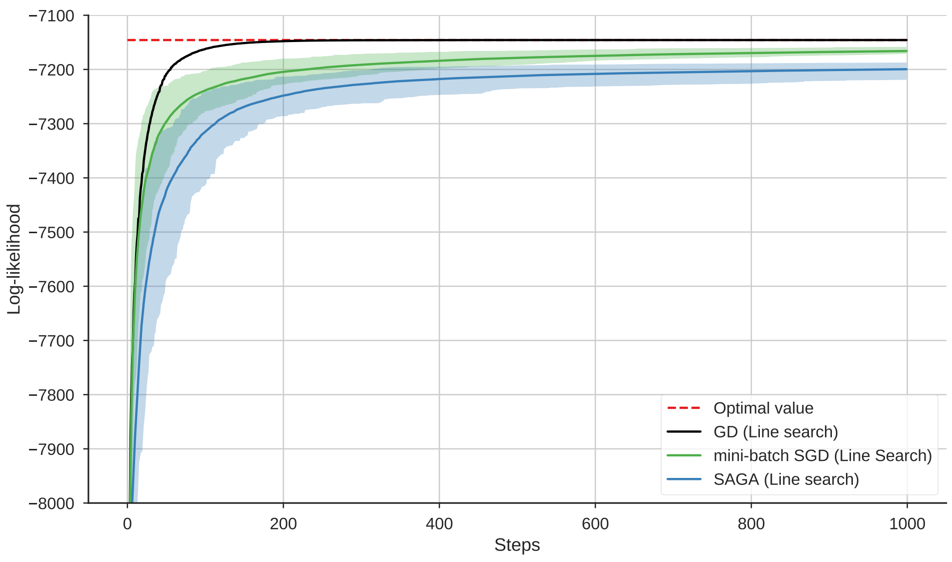

Fig. 3. Optimization of the normalized model $\mathcal{M}$ for different first-order methods using Line Search method for the step-size. The line corresponds to the average over 100 runs and the transparent part corresponds to the 95% confidence interval.

Another advantage of using Line Search with Stochastic algorithms is that it dampens the oscillations within optimization. Both SAGA and mini-batch SGD have much smoother convergence, as seen by comparing Fig. 3 with Fig. 2. We also see that the algorithms converge faster in the initial steps. However, all three algorithms are still not able to reach termination within 1,000 steps. We are currently investigating the possible reasons behind this behavior.

We are also currently implementing and testing second-order methods. However, in classical second-order methods, the Hessian is computed on all observations, which is impracticable. Therefore, we have to use methods such as Hessian-free optimization, as done by Byrd et al. (2011) and Bordes et al. (2010), on our model before testing these methods on non-convex models. The main idea behind Hessian-free optimization is to use insights from Newton's method but to come up with a better way to minimize the quadratic function we get.

Bibliography

Bierlaire, M. (2003). BIOGEME: a free package for the estimation of discrete choice models. *Swiss Transport Research Conference 2003.*

Bierlaire, M., Axhausen, K., and Abay, G. (2001). The acceptance of modal innovation: The case of Swissmetro. *Swiss Transport Research Conference 2001.*

Bordes, A., Bottou, L., Gallinari, P., Chang, J., and Smith, S. A. (2010). Erratum: SGDQN is Less Careful than Expected. *Journal of Machine Learning Research*, 11(Aug):2229–2240.

Bottou, L., Curtis, F. E., and Nocedal, J. (2016). Optimization Methods for Large-Scale Machine Learning. *arXiv:1606.04838 [cs, math, stat].* arXiv: 1606.04838.

Byrd, R., Chin, G., Neveitt, W., and Nocedal, J. (2011). On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning. *SIAM Journal on Optimization*, 21(3):977–995.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Livni, R., Shalev-Shwartz, S., and Shamir, O. (2014). On the Computational Efficiency of Training Neural Networks. *arXiv:1410.1141 [cs, stat].* arXiv: 1410.1141.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs].* arXiv: 1609.04747.

Wu, X., Zhu, X., Wu, G. Q., and Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107.