STOCHASTIC OPTIMIZATION WITH ADAPTIVE BATCH SIZE: DISCRETE CHOICE MODELS AS A CASE STUDY

hEART 2019: 8th Symposium of the European Association for Research in Transportation

Gael Lederrey
Transport and Mobility Laboratory,
École Polytechnique Fédérale de Lausanne,
Lausanne, Switzerland
gael.lederrey@epfl.ch

Virginie Lurkin
Department of Industrial Engineering
Innovation Sciences,
Eindhoven University of Technology,
Eindhoven, The Netherlands
v.j.c.lurkin@tue.nl

Tim Hillel
Transport and Mobility Laboratory,
École Polytechnique Fédérale de Lausanne,
Lausanne, Switzerland
tim.hillel@epfl.ch

Michel Bierlaire
Transport and Mobility Laboratory,
École Polytechnique Fédérale de Lausanne,
Lausanne, Switzerland
michel.bierlaire@epfl.ch

February 28, 2019

Introduction

The 2.5 quintillion bytes of data created each day brings new opportunities, but also new stimulating challenges for the discrete choice community. Opportunities because more and more new and larger data sets will undoubtedly become available in the future. In addition, with these new data comes the possibility to uncover new insights into customers' behaviors. Challenging because insights can only be discovered if models can be estimated, which is not simple on these large datasets. State-of-the art algorithms, but also standard practices regarding model specification might no longer be adapted to these large data sets. Indeed, three state-of-the-art discrete choice softwares (Pandas Biogeme (Bierlaire, 2018), PyLogit (Brathwaite et al., 2017), and Larch (Newman et al., 2018)) are using the standard optimization algorithm available in the Python package `scipy`, *i.e.* the BFGS algorithm.

In contrast, extracting useful information from big data sets is at the core of Machine Learning (ML). Primarily interested in achieving high prediction accuracy, ML algorithms (and especially Neural Networks) have proved to be successful on models involving a huge number of parameters. Thus, large-scale machine learning models often involve both large volumes of parameters and large datasets. As such, first-order stochastic methods are a natural choice for large-scale machine learning optimization. Due to the high cost of computing the full-Hessian, the second-order methods have been much less explored. And yet, algorithms exploiting second-order information can provide faster convergence.

For the sake of interpretability, discrete choice models usually have a more restricted set of parameters than models typically investigated in the ML community. We, therefore, argue that it is possible to use second-order information to estimate these models. In this paper, inspired by the good practices and the intensive use of stochastic gradient methods in the ML field, we introduce the algorithm called Window Moving Average - Adaptive Batch Size (WMA-ABS) which is used to improve the efficiency of stochastic second-order methods. The objective of this paper is to investigate the convergence of our algorithms by benchmarking it against against standard second-order methods and quasi-newton methods using simple logit models on different datasets. We present preliminary results that indicate that our algorithms outperform the standard second-order methods, especially for large datasets. It constitutes a first step to show that stochastic algorithms can finally find their place in the optimization of Discrete Choice Models (DCMs).

Related Work

While the principle of optimization is pretty easy to understand[1], there are many ways to achieve the optimum. Some of the major subfields are Convex Programming, Integer Programming, Combinatorial Optimization, and Stochastic Optimization. For this paper, we are interested in Iterative Optimization Algorithms (IOA) with the addition of stochasticity. Thus, restrict our to discussion to IOAs. We refer the reader to the book named "Optimization: Principles and Algorithms" of Bierlaire (2015) for a good mathematical introduction on the principle of optimization and some algorithms.

The common ancestor of all IOAs is most likely the Gradien Descent (Cauchy, 1847). Its principle is straightforward. We define a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ such that it is defined and derivable in a neighborhood of a point a. We know that f will decrease the fast along its negative gradient. Thus, if

$$x_{n+1} = x_n - \gamma \nabla f(x_n) \tag{1}$$

with γ the step size small enough, then $f(x_n) \geq f(x_{n+1})$. A large variety of algorithms have been then developed based on gradient descent. One of the branches deriving from this algorithm is the Stochastic Algorithms. One of the first algorithm to appear is Stochastic Gradient Descent[2] (SGD). The principle is almost the same as in Equation 1. Let us define a function F such that it is a finite sum of other functions.

$$F(x) = \frac{1}{n} \sum_{=1}^{n} f_i(x) \tag{2}$$

Each function f_i is generally associated with the i-th observation in the dataset. Then, we can simply replace $f(x_n)$ in Equation 1 by $f_j(x_n)$ where j is a random index. This definition corresponds to the SGD. This algorithm inspired many researchers. For a good (but non-exhaustive) list of first-order stochastic IOA, we recommend the reader to have a look at the paper of Ruder (2016).

While much work has been done on first-order stochastic IOA, researchers have neglected second-order and quasi-Newton methods. Indeed, due to the high number of parameters, the computation of the Hessian can be tricky, even impossible in some cases. Thus, researchers have been exploring and developing these techniques for a specific purpose. For example, Gower et al. (2018) improved the BFGS algorithm specifically for solving Matrix Inversion problems. They are using a stochastic version of the BFGS algorithm in their case. Since our computers are much more powerful nowadays, researchers have started to work with second-order methods. However, they try to avoid to compute the Hessian since it is quite a heavy work. Martens (2010), for example, developed a Hessian-free optimization technique specifically for Deep Learning. Other researchers have worked on Hessian-free optimization such as Kiros (2013), and Wang et al. (2014). Some researchers have been working on modifying the BFGS algorithms in a stochastic way which corresponds to Hessian-free optimization (Mokhtari and Ribeiro, 2014, Keskar and Berahas, 2016, Bordes et al., 2009, 2010). The literature on second-order stochastic methods is sparser compared to first-order and quasi-Newton methods. Nevertheless, we can cite the article of Byrd et al. (2011) who are trying to get some information from the Hessian to improve the optimization process. More recently, Agarwal et al. (2016) developed a second-order stochastic method specifically for Machine Learning, *i.e.* using the finite-sum objective function.

Methodology

To demonstrate the utility and the development of the Window Moving Average - Adaptive Batch Size (WMA-ABS) algorithm, we decided to use Discrete Choice Models (DCMs) as our case study. Indeed, while much work has been done on stochastic first-order IOA due to the high number of parameters in Neural Network, in DCMs, the computation of the Hessian can be computed analytically. It is thus convenient to use stochastic second-order IOA. Figure 1 shows the optimization of the model MNL-SM[3] with the Newton Method, the Trust-Region algorithm, and the BFGS optimization algorithm. We can see that all of these algorithms are able to converge to a final solution in 14 epochs at the maximum. However, we can clearly see that Trust-Region is the fastest algorithm among the three. Thus, the first obvious step to speed up these softwares would be to use the "`trust-ncg`" (Newton conjugate gradient trust-region algorithm) algorithm in `scipy`.

[1] Finding a maximum or a minimum value that can be subject to some constraints

[2] We do not have a precise date for its emergence. Many sources are dating its origin back to the 1940's.

[3] The models are defined in the section Models at the end of this paper.

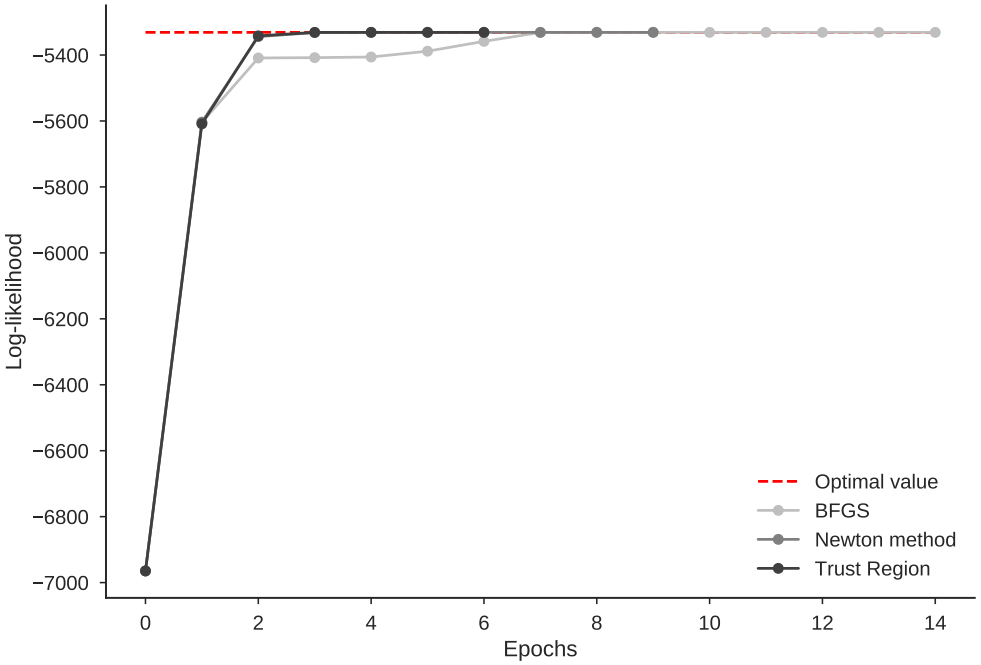


Figure 1: Optimization of the model MNL-SM with the Newton method, the BFGS, and the Trust-Region algorithms.

However, using standard second-order IOA may not be good enough with the arrival of bigger datasets. Indeed, we could potentially get faster optimization time by using stochastic algorithms. They have already shown their efficiency in Machine Learning. Thus, this second step is shown in Figure 2 with the optimization of the model MNL-SM with a Stochastic Newton Method (SNM). The SNM algorithm is defined thoroughly in Lederrey et al. (2018).

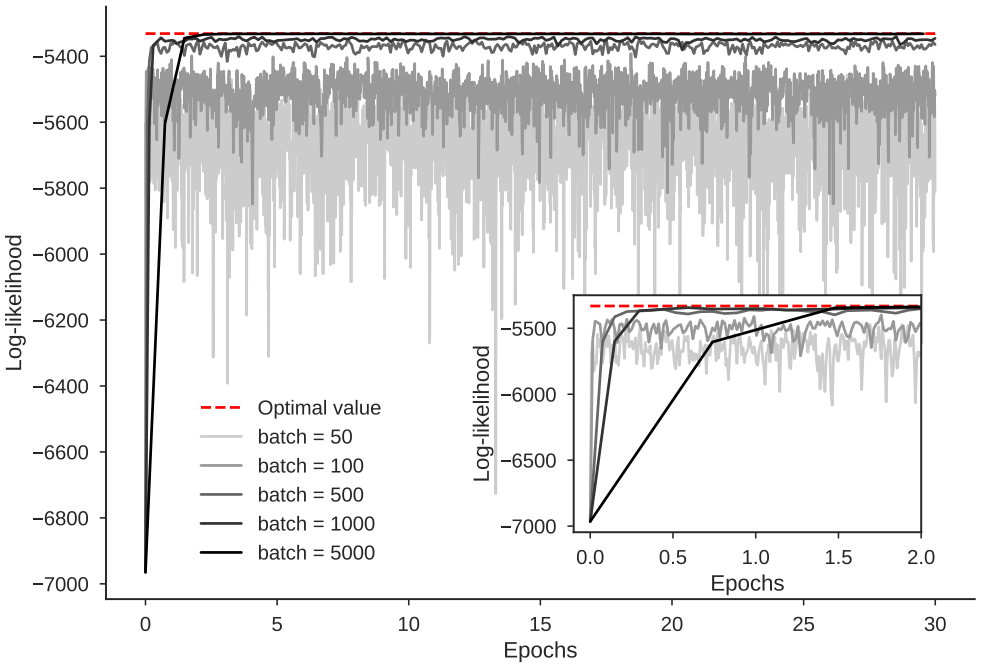


Figure 2: Optimization of the model MNL-SM with the Stochastic Newton Method.

As we can see in Figure 2, the SNM is not able to optimize the model MNL-SM to convergence, even with batches bigger than 50% of the dataset. Meanwhile, the NM is able to optimize it in 9 epochs. The two main flaws of this algorithm are:

- The value of the objective function is especially noisy with small batch size.
- The algorithm seems to reach a plateau after a few epochs for all batch sizes.

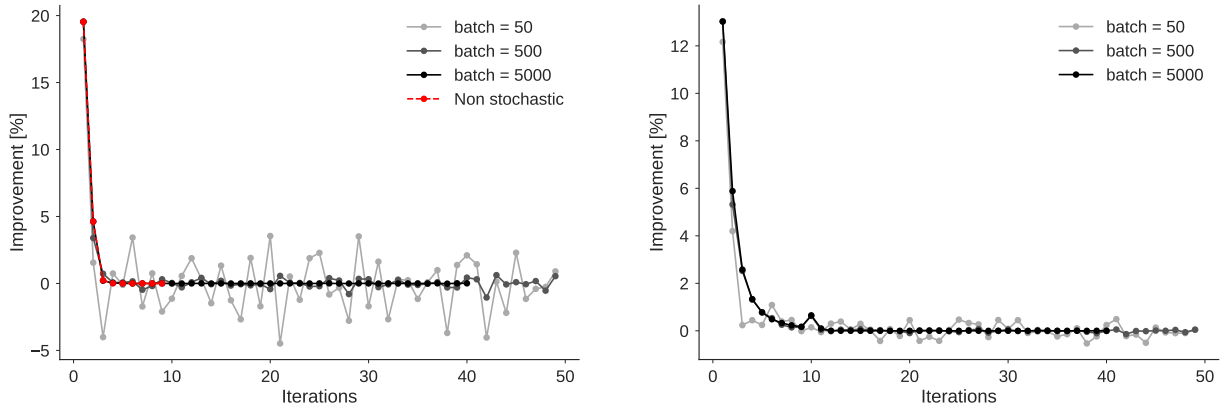
We refer the reader to our previous work (Lederrey et al., 2018) for a more advanced discussion of these flaws. Nevertheless, the SNM has one important strength. We can see that with small batches, the SNM is able to quickly reach a plateau close to the optimal solution. It is thus able to optimize quicker at the beginning of the optimization process with smaller batch size than with larger batch size. To create a new stochastic second-order IOA, it is important to deal with these two flaws and to take advantage of the quick optimization using small batch size. We should start understanding the flaws by looking at the improvement of the objective function. We define it as:

$$\Delta_i = \frac{\mathcal{L}_{i-1} - \mathcal{L}_i}{\mathcal{L}_{i-1}} \tag{3}$$

Where i is the current iteration, and $\mathcal{L}_i$ is the value of the log likelihood at iteration i. The improvement for different batch sizes for the SNM is given in Figure 3a. The general trend corresponds to the results in Figure 2. Indeed, we see that there is a plateau for all batch size. However, we also see much noise for the small batch size, as seen for the objective function values in Figure 2. Therefore, we need to use a smoothing technique to reduce this noise. An appropriate algorithm used in Computer Vision as well as Economics is the Weighted Moving Average (WMA). For a window of n values at the current iteration M, the WMA at M is defined by:

$$\text{WMA}_{M,n} = \frac{n\mathcal{L}_M + (n-1)\mathcal{L}_{M-1} + \cdots + 2\mathcal{L}_{M-n+2} + \mathcal{L}_{M-n+1}}{n + (n-1) + \cdots + 2 + 1} \tag{4}$$

The improvement using WMA with a window of 10 values is given in Figure 3b. As we can see, the improvement is smoother. However, we see a small spike in the tenth iteration. It is simply because at this point we are discarding the very first value. This first value ($\sim$12%) being very high compared to the second ($\sim$5%), the WMA is creating a small artifact at this point.



(a) Raw log likelihood

(b) Using WMA

Figure 3: Improvement of the log likelihood with different batch size.

The final step is to define an algorithm that will create new spikes of improvements. One way to achieve this is to update the batch size automatically. The Window Moving Average - Adaptive Batch Size (WMA-ABS) algorithm has been developed using the advantages of both small batch size and full batch size algorithms. Indeed, the idea is to speed up the process by starting with a small batch size and augment the batch size when the improvement is under a certain threshold. It should lead to an increase in the objective function value. The theoretical/expected results are shown in Figure 4. For example, we used the improvement curve of SNM with a batch size of 500 observations, in grey. We set a threshold of 3%. Thus, every time the improvement goes under this value, we multiply the batch size by a specific number. We do this until the batch size corresponds to the full size. At this point, we cannot use the WMA-ABS anymore, and we have to let the algorithm converge as shown in Figure 1.

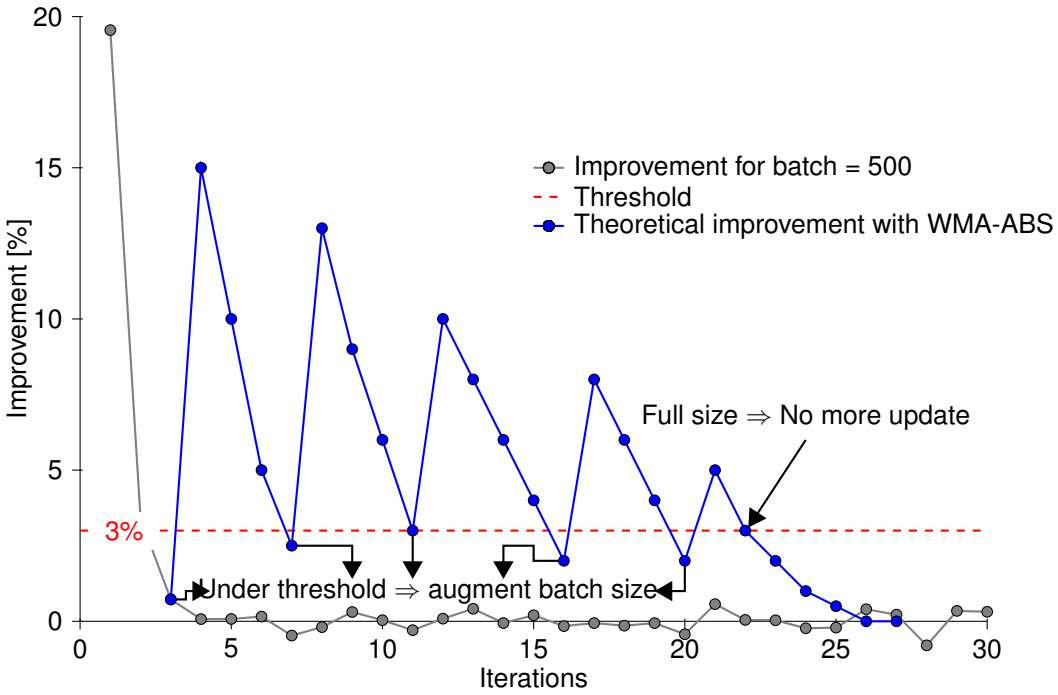


Figure 4: Theoretical improvement with ABS algorithm.

Before presenting the WMA-ABS algorithm, we need to introduce its four parameters:

- $\mathbf{W}$ corresponds to the size of the window. It is the same as n in the explanation of the WMA, see Equation 4.
- $\mathbf{\Delta}$ corresponds to the threshold under which the batch size will be augmented.
- $\mathbf{C}$ corresponds to the number of times the improvement has been under the threshold Δ. If the number of time is bigger than C, the batch size will be augmented.
- $\boldsymbol{\tau}$ corresponds to the factor multiplying the current batch size. For example, if $\boldsymbol{n}$ corresponds to the current batch size, the next one will be equal to $\boldsymbol{\tau} \cdot \boldsymbol{n}$.

We can now introduce the WMA-ABS algorithm. It has to be used at the end of each iteration, after each performed step. The method will simply decide when is the right time to update the batch size to gain new improvements in the loss function. It is given in Algorithm 1. The four parameters above have to be defined before using ABS. Thus, they are not listed in the Input of this algorithm. We also describe the following parameters that are inherent to the model: N the number of observations in the model, c the counter for the number of times under the threshold (it is set at 0 at the initialization)

Before presenting and discussing the results in details, we would like to suggest some values for the four parameters to the reader. Indeed, while it is possible to optimize the parameters of the WMA-ABS, this is not the goal since it will take a lot of time to do it. Thus, we recommend to use the parameters given in Table 1.

Table 1: Suggested parameters for the WMA-ABS algorithm

	Small	Big
$\mathbf{W}$	10	10
$\mathbf{\Delta}$	1%	1%
$\mathbf{C}$	1	2
$\boldsymbol{\tau}$	2	2

The only parameter that changes is the count. This is due to the fact that on big models/datasets, the probability to draw a small batch with data containing almost no useful information is higher than for a smaller model/dataset. We thus

Algorithm 1 Window Moving Average - Adaptive Batch Size (WMA-ABS)

Input: Current iteration index (M), function value at iteration M (f_M), and batch size (n)
Output: New batch size (n')

function ABS
  Store f_M in a list $\mathcal{F}$
  Compute $WMA_{M,W}$ using $\mathcal{F}$ and store it in a list $\mathcal{A}$
  if $M > 0$ **then** ▷ We need at least two values to compute the improvement.
    Compute i the improvement as in Equation 3 using the list $\mathcal{A}$ and store it in a list $\mathcal{I}$
    if $n < N$ **then**
      if $\mathcal{I}_M < \Delta$ **then** ▷ Improvement under the threshold
        $c = c + 1$
      else
        $c = 0$ ▷ We restart the counter
      if $c == C$ **then** ▷ We will update the batch size
        $c = 0$ ▷ We restart the counter
        $n' = \tau \cdot n$
        if $n' >= N$ **then** ▷ The batch size is too big now
          $n' = N$
      else
        $n' = n$
  return n'

recommend using the set of parameters for small models/datasets if the model of the reader has less than 10 parameters and/or less than 10'000 observations in the dataset.

Results

Before we discuss the benefits of the WMA-ABS algorithm, we want to make sure that the theoretical results presented in Figure 4 are working in practice. Figure 5 shows the values of the improvements for the SNM and the WMA-ABS on the model MNL-SM. The set of parameters used is the one given in Table 1 for "small" models.

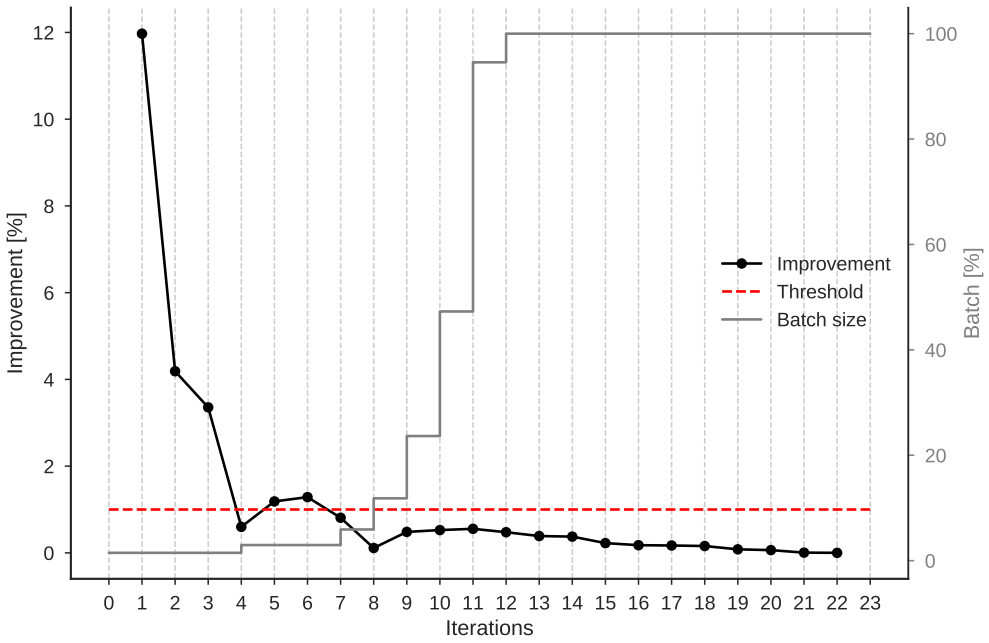


Figure 5: Improvement of the log likelihood and update of the batch size using the ABS algorithm.

We can clearly see that the first time the threshold goes under 1% happens at the fourth iteration. At the fifth iteration, we see a bump concerning the improvement, and it stays above 1% of improvements until the seventh iteration. After this iteration, the improvement stays under the 1% threshold due to getting closer to the optimal solution. It is thus better to quickly use the full batch in order to perform the last few steps. In this particular case, the convergence criterion was set to an especially small value, *i.e.* 10^{-10}, to better understand the behavior of the algorithm close to the optimal solution. For the rest of the results, we will use a convergence criterion of 10^{-6}.

We can now test the other two algorithms presented in Figure 1, *i.e.* the BFGS and the Trust-Region algorithms. Figure 6 shows the results of the optimization of the model MNL-SM with the three algorithms aforementioned with the set of parameters for a "big" model as suggested in Table 1.

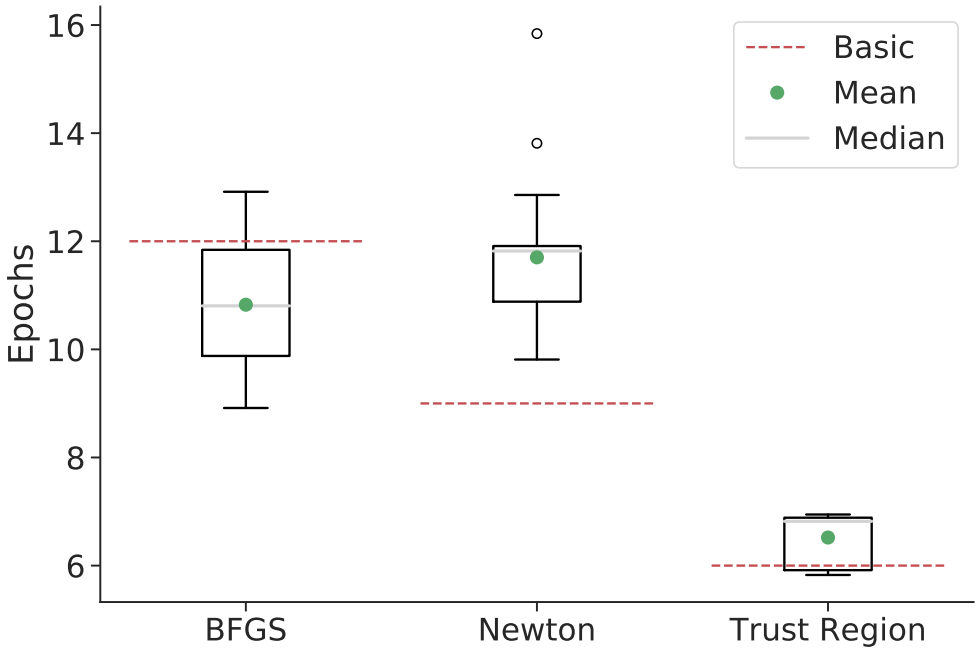


Figure 6: Comparison of WMA-ABS on the three optimization methods for the model MNL-SM.

We can clearly see that BFGS is beaten except for the "bad" cases. On the other hand, Trust-Region and Newton Method are still the best algorithms. We thus decided to optimize the hyperparameters using the epochs as the objective function using the Python package `hyperopt`. The optimal parameters are given in Table 2.

Algorithm	W	Δ [%]	C	τ
S-NM-WMA-ABS	18	4.4	1	6.2
S-TR-WMA-ABS	9	3.9	1	5.1
S-BFGS-WMA-ABS	29	2.6	1	1.8

Table 2: Best parameters found for the model MNL-SM and each algorithm using the package `hyperopt`.

As we can see, the value for C is always 1, as suggested for small models. The optimal threshold is a bit higher than the one suggested, same for the factor τ. The values for the window are inconsistent and needs to be further looked into. Nevertheless, we decided to redo the optimization with the optimal set of parameters. The results are given in Figure 7.

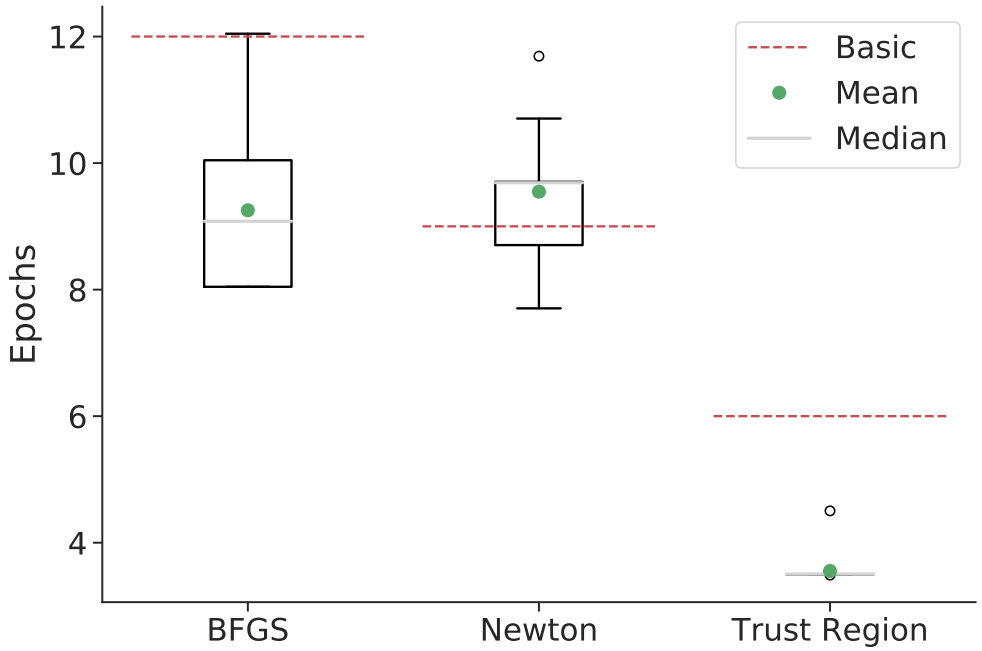


Figure 7: Comparison of WMA-ABS on the three optimization methods for the model MNL-SM. `hyperopt` has been used to find the optimal parameters.

This time, Trust-Region is beaten. Newton method is still better than its stochastic counterpart, but the gap has been reduced. The power of stochastic algorithms lies in the use of big datasets with some kind of correlation in the data. It is thus time to show results on a bigger model with more data. The results of the optimization of the model MNL-CLT are given in Figure 8. The set of parameters was the one for big models.

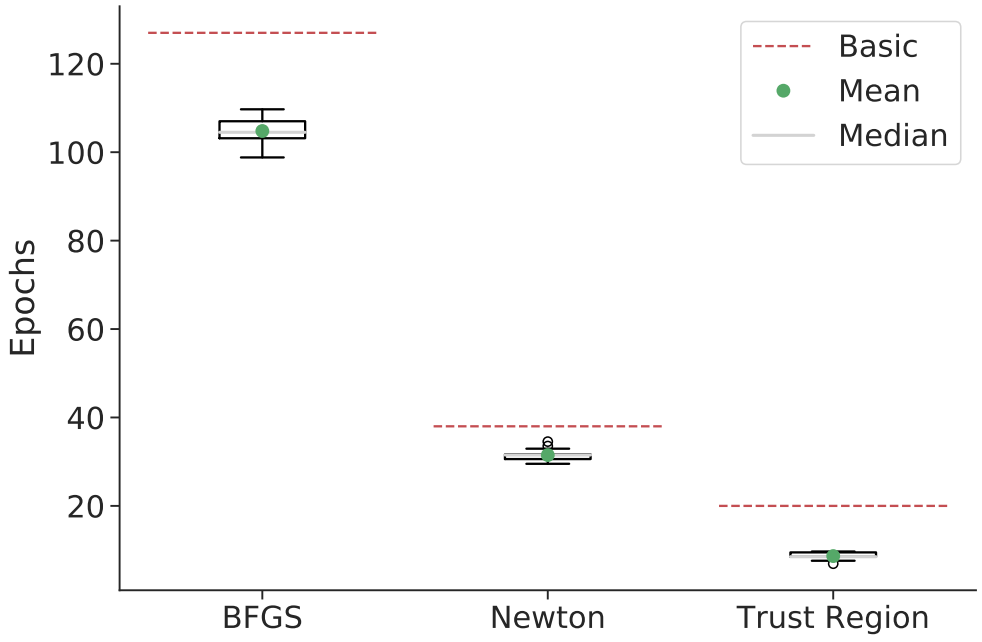


Figure 8: Comparison of WMA-ABS on the three optimization methods for the model MNL-CLT.

On this particular model and dataset, we were able to beat all three non-stochastic algorithms by quite a margin. Indeed, every attempt at optimizing the MNL-CLT with a stochastic algorithm lead to a smaller number of epochs compared to its optimization with its non-stochastic counterpart. Table 3 summarizes the results obtained for both models, all three sets of parameters, and all three optimization algorithms.

Algorithms	Param. used	MNL-SM	MNL-CLT
BFGS	Non-stochastic	12	127
	Small	9.81 ± 1.10 (-18.25%)	107.29 ± 3.55 (-15.52%)
	Big	10.83 ± 1.12 (-9.78%)	104.79 ± 2.90 (-17.49%)
	`hyperopt`	9.25 ± 1.21 (-22.88%)	/
Newton	Non-stochastic	7	38
	Small	9.97 ± 1.26 (+10.78%)	31.88 ± 1.69 (-16.11%)
	Big	11.70 ± 1.42 (+30.02%)	31.49 ± 1.34 (-17.12%)
	`hyperopt`	9.55 ± 0.96 (+6.09%)	/
Trust-Region	Non-stochastic	6	20
	Small	5.06 ± 0.39 (-15.68%)	10.29 ± 1.23 (-48.56%)
	Big	6.52 ± 0.48 (+8.66%)	8.65 ± 0.72 (-56.75%)
	`hyperopt`	3.55 ± 0.22 (-40.80%)	/

Table 3: Results of the optimization for both models and all optimization algorithms.

We can clearly see that Trust-Region is the best algorithm amongst the three we tested. In addition, we see that adding the WMA-ABS algorithm to this algorithm lead to the highest percentage of decrease in terms of epochs. It is also interesting to note that the model MNL-SM is sometimes faster to optimize with a non-stochastic algorithm. This is simply due to the simplicity and the size of its dataset.

Conclusion & Future Work

In this paper, we have presented an adaptive batch size algorithm called WMA-ABS. The central idea is to look at the improvement and augment the batch size when the improvement is too low using some smoothing technique to be more accurate. We show that it works well with different second-order IOAs and quasi-Newton method. We have confirmed that the use of stochastic algorithms is more interesting for models having a larger dataset and more parameters. While the WMA-ABS algorithm has some hyperparameters that can be optimized, it can also be used with the recommended sets of parameters given in Table 1 and give satisfying results. Concerning the DCMs softwares, we would like to advise them to use the Trust-Region algorithm at the very least. Indeed, this algorithm is available in the Python package `scipy` under the name "`trust-ncg`" and has been shown to outperform the other two algorithms by quite a margin.

For future research, we need to perform a full sensitivity analysis on the hyperparameters of the WMA-ABS algorithm. This will allow us to confirm that the suggested parameters are a good starting point despite not being the optimal ones. Secondly, we need to test We would also like to define a heuristic that will help the user to choose (or choose automatically for him) between stochastic and non-stochastic algorithms. Indeed, as it has been shown, for smaller models, the non-stochastic algorithms are generally better than their stochastic equivalents.

References

Agarwal, N., Bullins, B., and Hazan, E. (2016). Second-Order Stochastic Optimization for Machine Learning in Linear Time. *arXiv:1602.03943 [cs, stat]*. arXiv: 1602.03943.

Bierlaire, M. (2015). *Optimization: Principles and Algorithms*. EPFL Press.

Bierlaire, M. (2018). Pandasbiogeme: a short introduction. Technical report, Technical Report TRANSP-OR 181219, Transport and Mobility Laboratory, Ecole ....

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., and Gallinari, P. (2009). SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *J. Mach. Learn. Res.*, 10:1737–1754.

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.

Brathwaite, T., Vij, A., and Walker, J. L. (2017). Machine Learning Meets Microeconomics: The Case of Decision Trees and Discrete Choice. *arXiv:1711.04826 [stat]*. arXiv: 1711.04826.

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.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

Gower, R. M., Hanzely, F., Richtárik, P., and Stich, S. (2018). Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization. *arXiv:1802.04079 [cs, math]*. arXiv: 1802.04079.

Hillel, T. (2019). *Understanding travel mode choice: A new approach for city scale simulation*. PhD thesis, University of Cambridge, Cambridge.

Hillel, T., Elshafie, M. Z. E. B., and Jin, Y. (2018). Recreating passenger mode choice-sets for transport simulation: A case study of London, UK. *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction*, 171(1):29–42.

Keskar, N. S. and Berahas, A. S. (2016). adaQN: An Adaptive Quasi-Newton Algorithm for Training RNNs. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 1–16. Springer, Cham.

Kiros, R. (2013). Training Neural Networks with Stochastic Hessian-Free Optimization. *arXiv:1301.3641 [cs, stat]*. arXiv: 1301.3641.

Lederrey, G., Lurkin, V., and Bierlaire, M. (2018). SNM: Stochastic Newton Method for Optimization of Discrete Choice Models. *IEEE - ITSC'18*.

Martens, J. (2010). Deep learning via Hessian-free optimization. *ICML*, 27:735–742.

Mokhtari, A. and Ribeiro, A. (2014). RES: Regularized Stochastic BFGS Algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104.

Newman, J. P., Lurkin, V., and Garrow, L. A. (2018). Computational methods for estimating multinomial, nested, and cross-nested logit models that account for semi-aggregate data. *Journal of Choice Modelling*, 26:28–40.

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

Wang, X., Ma, S., and Liu, W. (2014). Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. *arXiv:1412.1196 [math]*. arXiv: 1412.1196.

Models

In this section, we want to present the two models and their respective dataset that have been used in this paper.

MNL-SM

The model MNL-SM is a simple Multinomial Logit Model using the Swissmetro dataset (Bierlaire et al., 2001). The Swissmetro dataset contains 10'728 observations. This model has 4 different parameters and can be found on the Biogeme website[4] under the name `logit01.py`.

MNL-CLT

The model MNL-CLT is a Multinomial Logit Model using the London Passenger Mode Choice dataset[5] Hillel et al. (2018). The model has 100 parameters and is available in Hillel (2019). We used only the years 2012-2013 to train the model for a total of 54'766 observations.

[4] http://biogeme.epfl.ch/examples_swissmetro.html

[5] Contact `tim.hillel@epfl.ch` for more information about this dataset.