SNM: Stochastic Newton Method for Optimization of Discrete Choice Models

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

Virginie Lurkin
Transport and Mobility Laboratory
École Polytechnique Fédérale de Lausanne
Station 18, CH-1015 Lausanne
virginie.lurkin@epfl.ch

Michel Bierlaire
Transport and Mobility Laboratory
École Polytechnique Fédérale de Lausanne
Station 18, CH-1015 Lausanne
michel.bierlaire@epfl.ch

***Abstract*—Discrete Choice Models' software often used standard optimization algorithms. However, with the advent of Big Data, these algorithms will soon become a drawback for training such models. The field of Machine Learning has seen the emergence of many powerful first-order methods. Yet, they do not make use of second-order methods often due to the difficulty of computing the Hessian when having a massive amount of features. Discrete Choice Models are often much smaller than Machine Learning models. Therefore, it is possible to use second-order methods for optimizing such models.**

In this article, we present a Stochastic Newton Method (SNM) for training Multinomial Logit models. We benchmarked this algorithm against standard first-order methods and quasi-newton methods. Also, we highlight one problem when using stochastic methods and propose a future fix for SNM.

***Index Terms*—Discrete Choice Models, Optimization**

I. Introduction

In the "optimization world", there are three main categories of iterative algorithms: first-order methods, quasi-newton methods, and second-order methods. Of course, many other types of algorithms for optimizing objective function in a constrained or unconstrained environment. However, recently much work has been done on iterative optimization algorithms, see Section II, thanks to the Machine Learning community. Indeed, with the advent of Machine Learning since the last twenty years, many researchers have been working on new ways to optimize models. Even if computers have become more powerful years after years, Machine Learning models have become more and more complex. We often see Deep learning models with more than a million features. Thus, optimization's research for Machine Learning focusses on first-order methods. Indeed, computing the Hessian is quite tedious with a massive number of features.

Currently, most of the Discrete Choice models' packages, such as Larch [1] or Biogeme [2], make use of standard optimization algorithms. Within the era of Big Data, we want to use more and more data to get more accurate models. However, most of those standard algorithms will struggle with huge datasets. Therefore, the good practice coming from Machine Learning is to use stochastic algorithms. Stochastic first-order methods are known to be very efficient. However, they have some drawback such as the loss of information on the curvature of the function, contain in the Hessian. For odd objective function, this information is sometimes required. Therefore, thanks to the reasonable size of Discrete Choice models, we argue that it is possible to use Second-order stochastic methods for training Discrete Choice models and that it outperforms the standard first-order methods and quasi-newton methods.

II. Related Work

~~As stated earlier, optimization algorithms play an important role in Machine Learning. Indeed, if we want to use complex models, we need to use powerful optimization methods. First-order methods have been studied a lot, and many variants of the standard Stochastic Gradient Descent (SGD) exists. It is well known that standard first-order methods tend~~ to struggle when the curvature of the objective function is not homogeneous [3]. Therefore, Quian describes a method using momentum to help the gradient in these particular situations [4]. Other first-order methods adapt the step size to the parameters, such as Adagrad [5]. Then, there is an iterative process between researchers trying to improve previous algorithms. Ruder [6] gives a good overview of first-order methods, from SGD up to complex and recent first-order algorithms such as Nadam [7] or AMSGrad [8].

More recently, with the help of more powerful computers, researchers have been looking at quasi-newton methods. Indeed, such method may be beneficial when first-order methods are struggling due to the lack of information from the curvature. The idea behind quasi-newton methods is to use the gradient to approximate the Hessian iteratively. BFGS algorithm [9] is a standard quasi-newton method. Stochastic BFGS algorithms such as RES-BFGS [10], a regularized stochastic BFGS, are used nowadays. Many researchers are trying to make use of the structure of the problem to find alternative versions of a given algorithm to perform better on this specific problem. For example, Gower *et al.* [11] have implemented an alternative version of BFGS for Matrix Inversion. Keskar *et al.* [12] have implemented adaQN, an adaptative quasi-newton method specifically designed for training Recurrent Neural Networks. Some researchers, such as Ye and Zhang [13], are inspired by the progress on first-order methods to improve

	Value	Std err	t-test	p-value
ASC_{Car}	0	-	-	-
ASC_{SM}	$7.86 \cdot 10^{-1}$	$6.93 \cdot 10^{-2}$	11.35	0.00
$\text{ASC}_{\text{Train}}$	$9.83 \cdot 10^{-1}$	$1.31 \cdot 10^{-1}$	7.48	0.00
$\beta_{\text{TT,Car}}$	$-1.05 \cdot 10^{-2}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\text{TT,SM}}$	$-1.44 \cdot 10^{-2}$	$6.36 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\text{TT,Train}}$	$-1.80 \cdot 10^{-2}$	$8.65 \cdot 10^{-4}$	-20.78	0.00
$\beta_{\text{C,Car}}$	$-6.56 \cdot 10^{-3}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\text{C,SM}}$	$-8.00 \cdot 10^{-3}$	$3.76 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\text{C,Train}}$	$-1.46 \cdot 10^{-2}$	$9.65 \cdot 10^{-4}$	-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 I
PARAMETERS OF THE OPTIMIZED MODEL $\mathcal{M}$ BY BIOGEME.

second-order methods. We also see some algorithms making use of Conjugate Gradient and stochasticity to create better algorithms as done by Byrd *et al.* [14]. In the end, the most advanced and recent methods make use of quasi-newton methods, see [15–18]. However, very little work on second-order method, making use of the analytical Hessian, can be found in the literature.

III. METHODOLOGY

In this section, we present the model used in this article, several optimization algorithms as well as the Stochastic Newton Method[1].

A. Model

We use the *Swissmetro* dataset [19] and build a multinomial logit model denoted by $\mathcal{M}$:

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

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. For this model, we removed all observations with unknown choice, unkown age and non-positive travel time. This gives a total of 9,036 observations.

We first estimate the model with Biogeme [2] to obtain the optimal parameter values and verify that all parameters are significant. However, we do not use the usual log-likelihood. Instead, we are using a normalized log-likelihood which simply corresponds to the log-likelihood divided by the number of observations. Therefore, the final normalized log-likelihood is -0.7908 and the parameters are given in Table I.

We also provide a normalized model $\bar{\mathcal{M}}$ where the values of travel time, cost, and headway have been divided by 100. The parameters for this normalized model are the same as model $\mathcal{M}$ except that the values of parameters associated to the features normalized are multiplied by 100. The reason behind this normalization is to have parameters close to each other, *i.e.* in the same order of magnitude, as opposed to the values in Table I where the parameter values are in four orders of magnitude.

B. Stochastic Newton Algorithm (SNM)

In this article, we present an algorithm called Stochastic Newton Method. Within Neural Networks, the number of features K can easily exceed one million. Thus, this is leading to huge Hessian since it will have K^2 elements. Discrete Choice Models, on the other hand, tend to have a reasonable number of features. Indeed, since the primary purpose of Discrete Choice Models is explaining the behavioral aspect of the samples, the models cannot contain too many parameters. Therefore, the primary limitation of Newton methods encountered in Neural Networks is not valid for Discrete Choice models. Yet, one problem remains: the exponential growth of data. Indeed, computing the Hessian on many data can be as tedious as computing it for many features. Thus the need for a Stochastic Newton Method (SNM).

The main point of this algorithm is to compute a stochastic Hessian. We show here that computing a stochastic Hessian is possible for a Logit Model. The generalization can be applied to any finite-sum function as such as the log-likelihood of a Logit Model. Let N denote the number of samples, $\mathcal{C}$ denote the choice set and $\mathcal{C}_n$ denote the choice set available for observation n and define

$$y_{in} = \begin{cases} 1 & \text{if observation } n \text{ chose alternative } i, \\ 0 & \text{otherwise.} \end{cases}$$

The likelihood function for a choice model is given by

$$\mathcal{L}^* = \prod_{n=1}^{N} \prod_{i \in \mathcal{C}_n} P_n(i)^{y_{in}} \quad (2)$$

where $P_n(i)$ denotes the probability that observation n choses alternative i. For a Logit model, we can define this probability as

$$P_n(i) = \frac{e^{V_{in}}}{\sum_{j \in \mathcal{C}_n} e^{V_{jn}}} \quad (3)$$

where V_{in} denotes the utility of alternative i for observation n. If we take the logarithm of Equation (3), we get the log-likelihood:

$$\begin{aligned} \mathcal{L} &= \sum_{n=1}^{N} \sum_{i \in \mathcal{C}_n} y_{in} \left(V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \\ &= \sum_{n=1}^{N} \left(\sum_{i \in \mathcal{C}_n} y_{in} V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \end{aligned} \quad (4)$$

The second equality is done using the fact that $\sum_{i \in \mathcal{C}_n} y_{in} = 1$. We then update the log-likehood of Equation (4) to create a normalized log-likelihood.

$$\bar{\mathcal{L}} = \frac{1}{N}\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{i \in \mathcal{C}_n} y_{in} V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \quad (5)$$

[1] The code is available on github: https://github.com/glederrey/IEEE2018_SNM

This is done such that the value of the log-likelihood stay in the same magnitude of order for any subset of observations $\mathcal{I}$. Indeed, if we denote $\mathcal{L}_\mathcal{I}$ as the log-likelihood computed on the observation from $\mathcal{I}$ and $\mathcal{N}$ the set of all observations, we see that

$$\begin{aligned}
\mathcal{L}_\mathcal{I} &= \sum_{n \in \mathcal{I}} \left(\sum_{i \in \mathcal{C}_n} y_{in} V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \\
&< \sum_{n \in \mathcal{I}} \left(\sum_{i \in \mathcal{C}_n} y_{in} V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \\
&\quad + \sum_{n \in \mathcal{N} \setminus \mathcal{I}} \left(\sum_{i \in \mathcal{C}_n} y_{in} V_{in} - \ln \sum_{j \in \mathcal{C}_n} e^{V_{jn}} \right) \\
&= \mathcal{L}
\end{aligned} \tag{6}$$

As shown in Equation (6), the standard log-likelihood cannot be compared on different set of data if they do not have the same number of data. Therefore, it can be shown that normalizing this log-likelihood as done in Equation (5) produces log-likelihood of same order of magnitude independently of the number of observations.

The first derivatives of $\bar{\mathcal{L}}$ with respect to the coefficient for $k = 1, \ldots, K$ are given by

$$\begin{aligned}
\frac{\partial \bar{\mathcal{L}}}{\partial \beta_k} &= \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{i \in \mathcal{C}_n} y_{in} \frac{\partial V_{in}}{\partial \beta_k} - \sum_{i \in \mathcal{C}_n} 1 P_n(i) \frac{\partial V_{in}}{\partial \beta_k} \right) \\
&= \frac{1}{N} \sum_{n=1}^{N} \sum_{i \in \mathcal{C}_n} \left(y_{in} - P_n(i) \right) \frac{\partial V_{in}}{\partial \beta_k}
\end{aligned} \tag{7}$$

The second derivatives for $k = 1, \ldots, K$ and $l = 1, \ldots, K$ are given by

$$\frac{\partial^2 \bar{L}}{\partial \beta_k \partial \beta_l} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i \in \mathcal{C}_n} P_n(i) W_{ink} W_{inl} \tag{8}$$

where

$$W_{ink} = \left(\frac{\partial V_{in}}{\partial \beta_k} - \sum_{j \in \mathcal{C}_n} \frac{\partial V_{jn}}{\partial \beta_k} P_n(j) \right)$$

From the definition of the second derivatives in Equation (8), it is easy to compute the second derivative for only one observation o.

$$\left. \frac{\partial^2 \bar{L}}{\partial \beta_k \partial \beta_l} \right|_m = - \sum_{i \in \mathcal{C}_o} P_o(i) W_{iok} W_{iol} \tag{9}$$

From the definitions in Equations (8) and (9), we can conclude that the Hessian on a subset of the observations $\mathcal{I}$ is simply the average of the Hessians for each of observation $i \in \mathcal{I}$.

We present now the Stochastic Newton Method (SNM), see Algorithm 1. The computation of both the stochastic gradient and the stochastic Hessian are done on lines 9 and 10. One particular feature of this algorithm is the computation of the direction for the next step. Indeed, with small batches, the Hessian may be singular. For example, it is possible that a variable associated with a parameter β_k is always equal to 0 for a small batch, *e.g.* binary variables. Therefore, the derivative of V_{in} by β_k will always be zero. Therefore, the row and column of the Hessian will both be zero for this particular parameter, thus making it singular. The countermeasure to this possibility is to test if the Hessian is singular or not. If it is not the case, then the algorithm performs a standard Newton step with the stochastic Hessian and gradient. However, if the Hessian is singular, the algorithm performs a Stochastic Gradient Descent (SGD) step. Concerning the choice of the step size, for a given objective function, it often differs between SGD and Newton Method. Therefore, we have two possibilities: the algorithm should use two different step sizes, or we can perform a line search, as explained at the end of Section III-C.

C. Benchmark algorithms

We use several algorithms to train models $\mathcal{M}$ and $\bar{\mathcal{M}}$. These algorithms fall into three different categories: first-order methods, second-order methods, and quasi-newton methods. For first-order methods, we use mini-batch SGD [6] and Adagrad [5]. For the quasi-newton methods, we use BFGS algorithm [9] and RES-BFGS [10], a regularized stochastic version of BFGS. The main second-order algorithm is the Newton method [20]. We run all the algorithms presented above with a backtracking Line Search method using the Armijo-Goldstein condition [21] to avoid the long and tedious search of a good step size.

IV. Results

In this section, we show why first-order moethods tend to struggle to optimize Multinomial Logit Model. Then, we do a benchmark on multiple algorithms. Finally, we present a weakness of SNM and a future way to fix it.

A. Raw data vs Normalized data

Most of the data we can obtain are not normalized. This is often a preprocessing step required for some optimization algorithm to work. As explained in Section III-A, the optimization of the model leads to optimized parameters ranging over four orders of magnitude. Since the step size is the same for all parameters, it is difficult to find an optimal step size. Figure 1(a) and 1(b) show the optimization process of the log-likelihood for SGD and Adagrad, respectively, for both the raw model $\mathcal{M}$ and the normalized model $\bar{\mathcal{M}}$. For both algorithms, the optimization was done a thousand times for two epochs with a batch size of 100 observations. The lines correspond to the average while the colored part corresponds to the 95% confidence interval. The results show that these algorithms perform better on the normalized model $\bar{\mathcal{M}}$. Table II show the average value of the log-likelihood after two epochs for these two algorithms on both models.

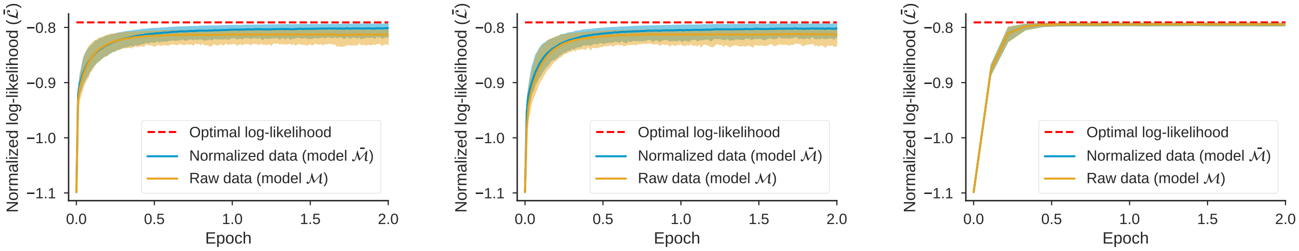
Figure 1(c) shows the results of the training on both models with SNM. We ran this algorithm with batches of 1,000

Algorithm 1 Stochastic Newton Method (SNM)

Input: Starting parameter value (θ_0), data ($\mathcal{D}$), function (f), gradient (∇f), Hessian ($\nabla^2 f$), number of epochs (n_{ep}), batch size (n_{batch})

Output: Epochs (e), parameters (θ), function values (f_v)

```
function SNM
    $(n_\mathcal{D}, m) = |\mathcal{D}|$                                    ▷ Number of samples and parameters
    $n_{iter} \leftarrow \lceil n_{ep} n_\mathcal{D} / n_{batch} \rceil$     ▷ Number of iterations
    Initialize $e$, $\theta$ and $f_v$. Set $\theta[0] \leftarrow \theta_0$
    for $i = 0 \ldots n_{iter}$ do
        $e[i] \leftarrow i \cdot n_{batch} / n_\mathcal{D}$                  ▷ Store the epoch
        $f_v[i] \leftarrow f(\theta[i])$                                     ▷ Store the function value
        idx $\leftarrow n_{batch}$ values from $\mathcal{U}(0, n_\mathcal{D})$ without replacement
        grad $\leftarrow \nabla f_{\texttt{idx}}(\theta[i])$                ▷ Gradient on the samples from idx
        hess $\leftarrow \nabla^2 f_{\texttt{idx}}(\theta[i])$              ▷ Hessian on the samples from idx
        if hess is non singular then
            inv_hess $\leftarrow$ hess$^{-1}$
            step $\leftarrow -$grad $\cdot$ inv_hess
        else
            step $\leftarrow$ grad
        $\alpha \leftarrow$ Backtracking Line Search with step on the subset of data with indices from idx
        $\theta[i+1] \leftarrow \theta[i] + \alpha \cdot$ step
    $e[n_{iter}] \leftarrow n_{iter} \cdot n_{batch} / n_\mathcal{D}$
    $f_v[n_{iter}] \leftarrow f(\theta[n_{iter}])$
    return $e$, $\theta$ and $f_v$
```



(a) SGD

(b) Adagrad

(c) SNM

Fig. 1. Evaluation of the algorithms on raw data (model $\mathcal{M}$) and normalized data (model $\bar{\mathcal{M}}$). The vertical axis corresponds to the normalized log-likelihood presented in Equation (5). Each time, a thousand run have been executed. The lines correspond to the average value over all the runs and the colored part correspond to the 95% confidence interval. SGD and Adagrad are run with a batch size of 100 observations, SNM is run with a batch size of 1,000 observations.

	SGD	**Adagrad**	**SNM**
on $\mathcal{M}$	-0.813107	-0.812080	-0.794219
on $\bar{\mathcal{M}}$	-0.801739	-0.801646	-0.794219
rel. diff.	1.42%	1.30%	0.00%

TABLE II

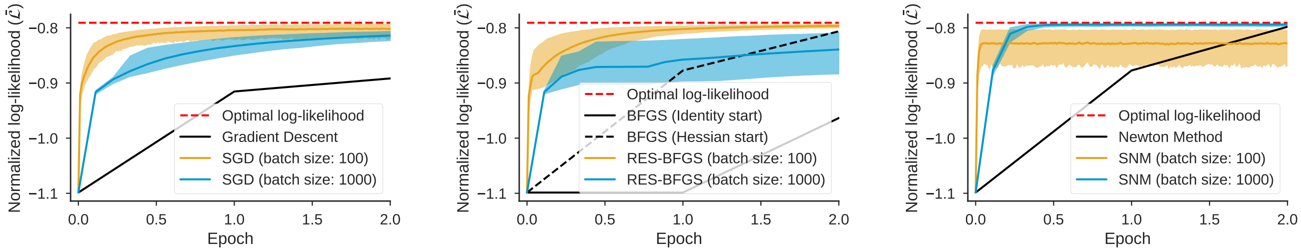
AVERAGE NORMALIZED LOG-LIKELIHOOD OVER A THOUSAND RUNS AT THE SECOND EPOCH FOR SGD, ADAGRAD AND SNM.

observations. In Sections IV-B and IV-C, we explain why we had to use a batch size with more observations. The qualitative results as well as the quantitative results in Table II show that second-order methods have less problem with badly conditioned optimization problem. Thus, it indicates that the information contained in the Hessian is important when the problem is ill-conditioned.

B. Comparison of the algorithms

In this section, we want to compare the three main categories of algorithms: first-order methods, quasi-newton methods, and second-order methods. Figure 2(a) shows the results for SGD with batch size of 100 and 1,000 as well as gradient descent. Figure 2(b) shows the results for RES-BFGS with batch sizes of 100 and 1,000 as well as standard BFGS. Finally, Figure Figure 2(c) shows the results for SNM with batch sizes of 100 and 1,000 as well as Newton method. For these three figures, we executed a thousand runs. The lines give the average value for the normalized log-likelihood, and the colored parts show the 95% confidence interval.

From Figure 2, we already see that first-order methods are the furthest from the optimal value. Then, we see that stochastic quasi-newton methods tend to struggle to reach the optimal value, especially with the first approximation of the Hessian being an identity matrix. Interestingly, we see that the RES-BFGS works better with smaller batch size while it



(a) First-order methods (b) Quasi-Newton methods (c) Second-order methods

Fig. 2. Comparison of the different algorithms presented in Section III-C and III-B. The vertical axis corresponds to the normalized log-likelihood presented in Equation (5). Each time, a thousand run have been executed. The lines correspond to the average value over all the runs and the colored part correspond to the 95% confidence interval.

	batch	**first-order**	**quasi-newton**	**second-order**
Stochastic	100	-0.801452	-0.796492	-0.828409
	1000	-0.812886	-0.837937	-0.794219
Full batch size		-0.891737	-0.963458/-0.806245	-0.798112

TABLE III

AVERAGE NORMALIZED LOG-LIKELIHOOD OVER A THOUSAND RUNS AT THE SECOND EPOCH FOR FIRST-ORDER METHODS, QUASI-NEWTON METHODS AND SECOND-ORDER METHODS.

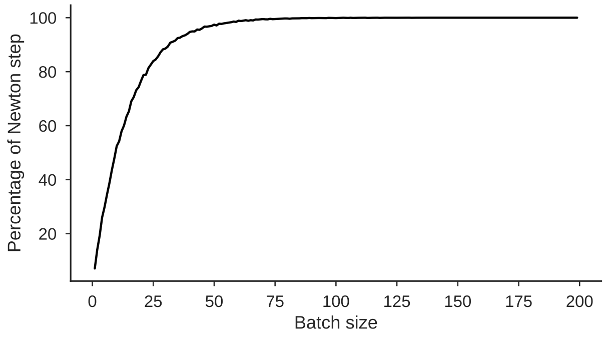


Fig. 3. Theoretical percentage of Newton step in function of the batch size for model $\bar{\mathcal{M}}$. The percentage was computed on a thousand draws.

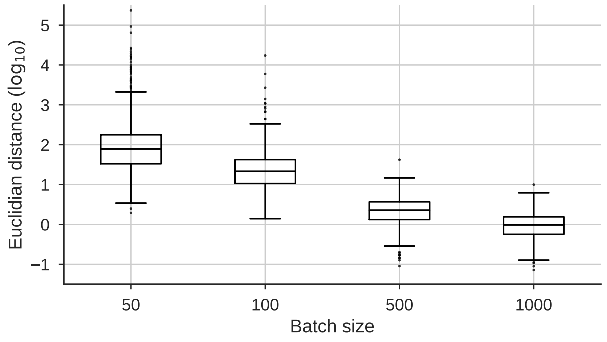


Fig. 4. Euclidian distance between the optimal parameters obtained on the full dataset and optimal parameters found on batches of the data for different batch sizes. The line in the middle represents the median. A thousand draws were coomputed for each batch size.

tends to struggle and plateau with big batch size. Nevertheless, it can get closer to the optimal solution than SGD. However, SNM is the best algorithm out of the three. Table III gives the average value of the normalized log-likelihood for the second epoch. In this table, we reported two values for the quasi-newton method and the full batch size: the first value reported is with the first approximation of the Hessian being an identity matrix, the second value corresponds to the real hessian. The numbers confirm that SNM is the best algorithm. However, it is interesting to note that contrary to the other two algorithms, SNM runs better with bigger batch size. In the next section, see Section IV-C, we study the reason behind this behavior.

C. Effect of the batch size

As shown in Figure 2, SNM is the only algorithm for which a more significant batch size works better. This behavior is quite odd, and the explanation may come from the direction. Indeed, as explained in Section III-B, the direction is either a gradient step or a Newton step depending on the singularity of the Hessian. Therefore, we are interested to know if this direction, *i.e.* the choice between a gradient step and a Newton step, depends on the batch size. In Figure 3, we show the percentage of Newton step that the algorithm is capable of performing in function of the batch size. This percentage is computed on a thousand draws.

The algorithm is only capable of performing a Newton step if the Hessian is non-singular. If we take a look at model $\mathcal{M}$, Equation 1, we see that we have one binary variable: $\mathbb{1}_{Senior}$. On our 9,036 observations, 8,406 observations have a 0 value for $\mathbb{1}_{Senior}$. In the particular case where all observations from a given batch have a 0 for $\mathbb{1}_{Senior}$, the hessian will be singular. However, as shown in Figure 3, this percentage goes quickly to 100%. With a batch size of 100 observations, the algorithm will perform a Newton step 99.86% of the time. Thus, we see that having binary variables that are often equal to 0 is not a problem for the algorithm.

However, small batch sizes create other problems. Indeed, when computing the Hessian with small batch size, the only information we get is from a small subset of the data. Therefore, for a given batch, the optimum can be different from the optimum on the whole dataset. Using the well optimized function `minimize` from the package `scipy.optimize`, we compute the optimum for different batch size. Then, we compare the euclidian distance between the optimum on the full dataset and the optimum from the different batches. Figure 4 shows the results of this experiment.

In Figure 4, we see that when taking small batches, the optimal solution is pretty far from the optimal solution on all data point. Therefore, this creates a problem in the computation of the step for SNM. Indeed, since we do not take into account previous Hessian, as opposed to RES-BFGS, the algorithm will often change direction with small batches. Indeed, every time we change the batch, the algorithm is chasing a different optimum. Thus, it makes it difficult for it to achieve the real optimum. One way to fix this problem is to keep the information about previous Hessian and use this information to correct the direction of the algorithm.

V. Conclusion

In this article, we tested several algorithms to optimize a Multinomial Logit Model. We showed that first-order methods have many limitations on these particular models. In the comparison process, we added a stochastic quasi-newton method and introduced a new stochastic second-order method called SNM. To be able to use such an algorithm, we showed that computing the Hessian on only one observation (or a batch of observations) is legit thanks to the finite-sum shape of the log-likelihood. We then showed that this new method works particularly well to optimize our model. Also, we presented the main weakness of this method and the beginning of the solution to fix..

Obviously, the next step is to fix the weakness of SNM. However, it is not a straightforward step. Indeed, we can merge the previous Hessian in may different ways and they will have to be tested. Nevertheless, with this new methodology in mind, many questions arise. As shown, we only tested our method on simple multinomial logit models. However, Discrete Choice theory contains many complex models such as Nested Logit Models, Cross-Nested Logit Models, and even models with Panel data. All of these models, on the contrary of Multinomial Logit Models, are not convex. Therefore, the optimization will be much more complicated. However, before experimenting with more complex models, it would be interesting to study the theoretical aspects of this model. Indeed, for the moment, we do not know the theoretical convergence rate of this algorithm. Thus, some theoretical work is required on this method. Finally, we also saw that making use of the model's structure can lead to good performance. For example, Latent variables require many draws from their probability distribution to be correctly estimated. Therefore, future algorithms should optimize while taking into account this structure. The same goes for models with Panel data. For this kind of models, the addition of a dimension is tricky when optimizing with standard methods.

VI. Acknowledgements

We would like to thank Tim Hillel for his valuable insight on the development of this method as well as his comments that significantly improved this article.

References

[1] J. Newman, V. Lurkin, and L. Garrow, "LARCH: A package for estimating multinomial, nested, and cross-nested logit models that account for semi-aggregate data," Aug. 2016. [Online]. Available: https://orbi.uliege.be/handle/2268/201287

[2] M. Bierlaire, "BIOGEME: a free package for the estimation of discrete choice models," *Swiss Transport Research Conference 2003*, Mar. 2003. [Online]. Available: https://infoscience.epfl.ch/record/117133

[3] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," *Proceedings of Eighth Annual Conference of the Cognitive Science Society, 1986*, 1986. [Online]. Available: https://ci.nii.ac.jp/naid/10022346408

[4] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, Jan. 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608098001166

[5] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[6] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv:1609.04747 [cs]*, Sep. 2016, arXiv: 1609.04747. [Online]. Available: http://arxiv.org/abs/1609.04747

[7] T. Dozat, "Incorporating Nesterov Momentum into Adam," Feb. 2016. [Online]. Available: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ

[8] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," Feb. 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[9] R. Fletcher, *Practical Methods of Optimization; (2Nd Ed.)*. New York, NY, USA: Wiley-Interscience, 1987.

[10] A. Mokhtari and A. Ribeiro, "RES: Regularized Stochastic BFGS Algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 23, pp. 6089–6104, Dec. 2014.

[11] R. M. Gower, F. Hanzely, P. Richtrik, and S. Stich, "Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization," *arXiv:1802.04079 [cs, math]*, Feb. 2018, arXiv: 1802.04079. [Online]. Available: http://arxiv.org/abs/1802.04079

[12] N. S. Keskar and A. S. Berahas, "adaQN: An Adaptive Quasi-Newton Algorithm for Training RNNs," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer, Cham, Sep. 2016, pp. 1–16. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46128-1_1

[13] H. Ye and Z. Zhang, "Nestrov's Acceleration For Second Order Method," *arXiv:1705.07171 [cs]*, May 2017, arXiv: 1705.07171. [Online]. Available: http://arxiv.org/abs/1705.07171

[14] R. Byrd, G. Chin, W. Neveitt, and J. Nocedal, "On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 977–995, Jul. 2011. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/10079923X

[15] R. Kiros, "Training Neural Networks with Stochastic Hessian-Free Optimization," *arXiv:1301.3641 [cs, stat]*, Jan. 2013, arXiv: 1301.3641. [Online]. Available: http://arxiv.org/abs/1301.3641

[16] A. Bordes, L. Bottou, and P. Gallinari, "SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent," *J. Mach. Learn. Res.*, vol. 10, pp. 1737–1754, Dec. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1577069.1755842

[17] A. Bordes, L. Bottou, P. Gallinari, J. Chang, and S. A. Smith, "Erratum: SGDQN is Less Careful than Expected," *Journal of Machine Learning Research*, vol. 11, no. Aug, pp. 2229–2240, 2010. [Online]. Available: http://www.jmlr.org/papers/v11/bordes10a.html

[18] N. Agarwal, B. Bullins, and E. Hazan, "Second-Order Stochastic Optimization for Machine Learning in Linear Time," *arXiv:1602.03943 [cs, stat]*, Feb. 2016, arXiv: 1602.03943. [Online]. Available: http://arxiv.org/abs/1602.03943

[19] M. Bierlaire, K. Axhausen, and G. Abay, "The acceptance of modal innovation: The case of Swissmetro," *Swiss Transport Research Conference 2001*, Mar. 2001. [Online]. Available: https://infoscience.epfl.ch/record/117140

[20] J. Caswell, "A treatise of algebra, both historical and practical," Tech. Rep., 1685.

[21] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, Jan. 1966. [Online]. Available: https://msp.org/pjm/1966/16-1/p01.xhtml