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*—BLA BLA BLA**

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

I. Introduction

☞ **Note:**

- Not a lot of work on optimization of DCMs
- ML are doing this a lot
- Expecting a lot more data. ⇒ Standard quasi Newton may have troubles
- Becomes interesting to search for new algorithms exploiting the structure of DCMs ⇒ Show structure DCM with obs, draws and panel.

II. Related Work

☞ **Note:**

- Lot of research has been done on first-order
- Some recent research on stochastic Hessian-free optimization
- Very few research on stochastic Newton method

III. Methodology

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

A. Model

We use the *Swissmetro* dataset [1] and build a multinomial logit model denoted by $\mathcal{M}$:

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

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.

	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.

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. Algorithms

We use several algorithms to train models $\mathcal{M}$ and $\mathcal{M}_N$. These algorithms fall into three different categories: first-order methods, second-order methods, and quasi-newton methods. As first-order methods, we use mini-batch SGD [3] and Adagrad [4]. For the quasi-newton methods, we use BFGS algorithm [5] and RES-BFGS [6], a regularized stochastic version of BFGS. The main second-order algorithm is the Newton method [7]. We run all the algorithms presented above with a backtracking Line Search method using the Armijo-Goldstein condition [8] to avoid the long and tedious search of a good learning rate.

C. Stochastic Newton Algorithm

In this article, we present an algorithm called Stochastic Newton Method. Within Neural Networks, the number of features K can easily exceed one million

☞ **NOTE:** REFERENCE

. 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

☞ **NOTE:** REFERENCE?

. 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 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

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

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

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

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:

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

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.

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

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 m.

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

Algorithm 1 Stochastic Newton Method

```
Input: Starting parameter value ($\theta_0$), data ($\mathcal{D}$), function ($f$), gradient ($\nabla f$), Hessian ($\nabla^2 f$), number of epochs ($n_{ep}$), batch size ($n_{batch}$)
Output: Epochs ($e$), parameters ($\theta$), function values ($f_v$)
 1: function SNM
 2:     $(n_{\mathcal{D}}, m) = |\mathcal{D}|$                                ▷ Number of samples and parameters
 3:     $n_{iter} \leftarrow \lceil n_{ep} n_{\mathcal{D}} / n_{batch} \rceil$                    ▷ Number of iterations
 4:     Initialize $e$, $\theta$ and $f_v$. Set $\theta[0] \leftarrow \theta_0$
 5:     for $i = 0 \dots n_{iter}$ do
 6:         $e[i] \leftarrow i \cdot n_{batch} / n_{\mathcal{D}}$                           ▷ Store the epoch
 7:         $f_v[i] \leftarrow f(\theta[i])$                                   ▷ Store the function value
 8:         idx $\leftarrow n_{batch}$ values from $\mathcal{U}(0, n_{\mathcal{D}})$ without replacement
 9:         grad $\leftarrow \nabla f_{\texttt{idx}}(\theta[i])$                     ▷ Gradient on the samples from idx
10:         hess $\leftarrow \nabla^2 f_{\texttt{idx}}(\theta[i])$                    ▷ Hessian on the samples from idx
11:         if hess is non singular then
12:             inv_hess $\leftarrow$ hess$^{-1}$
13:             step $\leftarrow$ $-$grad $\cdot$ inv_hess
14:         else
15:             step $\leftarrow$ grad
16:         $\alpha \leftarrow$ Backtracking Line Search with step on the subset of data with indices from idx
17:         $\theta[i+1] \leftarrow \theta[i] + \alpha \cdot$ step
18:     $e[n_{iter}] \leftarrow n_{iter} \cdot n_{batch} / n_{\mathcal{D}}$
19:     $f_v[n_{iter}] \leftarrow f(\theta[n_{iter}])$
20:     return $e$, $\theta$ and $f_v$
```

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 learning rate, for a given objective function, it often differs between SGD and Newton Method. Therefore, we have two possibilities: the algorithm should use two different learning rates, or we can perform a line search, as explained at the end of Section III-B.

IV. Results

In this section, we will show why a second-order stochastic algorithm is needed in comparison with first-order stochastic methods and quasi-newton stochastic methods. We will also demonstrate the utility of the particular direction for SNM.

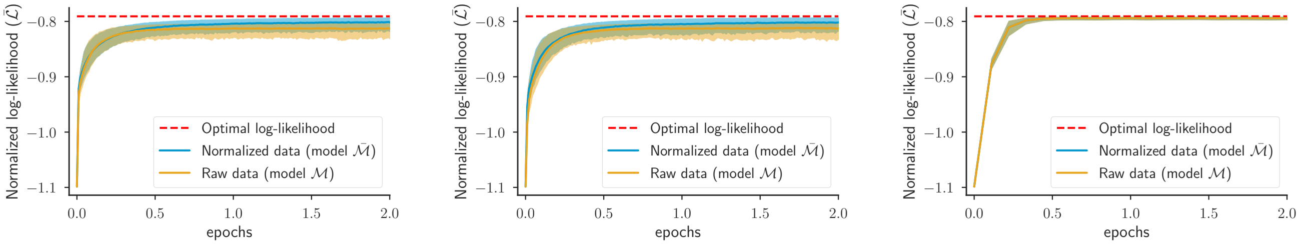
	SGD	**Adagrad**	**SNM**
on $\mathcal{M}$	-0.813107	-0.812080	-0.794219
on $\bar{\mathcal{M}}$	-0.801739	-0.801646	-0.794219
rel. diff.	1.42%	1.30%	0.00%

TABLE II

Average normalized log-likelihood over a thousand runs at the second epoch for SGD, Adagrad and SNM.

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 learning rate is the same for all parameters, it is difficult to find an optimal learning rate. Figure 1(a) and 1(b) show respectively 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 for two epochs with a batch size of 100 observations a thousand times. 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.



(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 batch size of 100 observations, SNM is run with a batch size of 1,000 observations.

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

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.

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

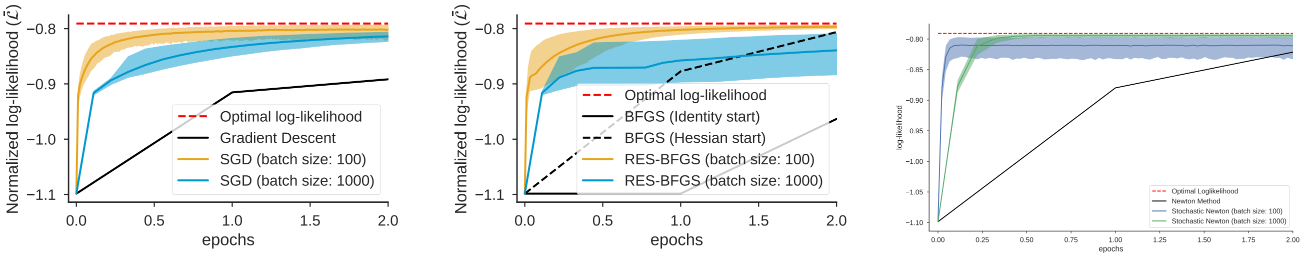
C. Direction of SNM

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-C, 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. Figure **??** shows the percentage of Newton step for different batch size. The values are computed on fifty runs, and SNM is optimized for two epochs.

As we can see, the percentage of Newton step tend to be quite small with small batch size while it becomes 100% for big batch size. As explained in Section III-C, the singularity of the Hessian depends highly on the data. 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 $\mathbb{1}_{Senior}$. Therefore, with small batches, it is highly possible to draw only observations with zeros as $\mathbb{1}_{Senior}$. One way to make this probability smaller is to create a variable y such that $y = 1 - \mathbb{1}_{Senior}$ and include it in the remaining alternatives instead. In the end, the only drawback of such binary variables is the fact that bigger batch size has to be used, explaining the surprising results in Figure 2(c).

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



(a) First-order methods

(b) Quasi-Newton methods

(c) Second-order methods

Fig. 2. Comparison of the different algorithms presented in Section III-B and III-C. 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.

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 one weakness of this method with a simple way to fix it.

With this new methodology in mind, many questions arise. Indeed, we only tested our method on simple multinomial logit models. However, The 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 have any clue about 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 the manuscript.

References

[1] 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

[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] 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

[4] 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

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

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

[7] J. Caswell, "A treatise of algebra, both historical and practical : with some additional treatises I. of the cono-cuneus; being a body representing in part a conus, an part a cuneus ; II. of angular sections; and other things relating there unto, and to Trigonometry ; III. of the angle of contact; with other things appertaining to the composition of magnitudes, the inceptive of magnitudes, and the composition of motions, with the results thereof ; IV. of combination, alternations, and aliquot parts," Tech. Rep., 1685.

[8] 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