REPRODUCTION: ONLINE LEARNING RATE ADAPTATION WITH HYPERGRADIENT DESCENT

Nikolaos Chrysovalantis Tsagkopoulos, Dimitri Diomaiuta & Konstantinos Mertzanis
[redacted], [redacted] & [redacted]
Department of Electronics and Computer Science
University of Southampton
{ncdt1u18,dd2u18,km2n18}@soton.ac.uk

ABSTRACT

A method improving the convergence rate of gradient-based optimisers was introduced by Baydin et al. (2017), reducing the importance of tuning. This report, as part of the *COMP6248 Reproducibility Challenge*, attempts to assess the reproducibility of that paper and determine the validity of its conclusions. Additionally, extensions examine possible limitations and produce higher order methods.

1 INTRODUCTION

In most gradient descent algorithms, the process of hyperparameter tuning is essential to efficiency, but it is also laborious and computationally expensive. To reduce its importance, while improving the convergence rate of gradient-based optimisers, Baydin et al. (2017) have introduced Hypergradient Decent (HD), a method applying gradient descent on the learning rate of an underlying gradient descent algorithm. HD, is an independent rediscovery of a preexisting algorithm (Almeida et al., 1999), and consequently lacks in originality. Baydin et al. (2017) have designed a method that is easy to implement and works well in practice, providing extensive demonstration and evaluation, but limited theoretical convergence guarantees. In this vein, HD was applied on Stochastic Gradient Descent (SGD), SGD with Nesterov momentum and Adam. The resulting algorithms, were incorporated into an API, which is publicly available.

This report, is part of the *COMP6248 Reproducibility Challenge*, attempting to reproduce the experimental results of Baydin et al. (2017). Having established the paper's aims and methodologies; the implementation and reimplementation details and their associated costs, are identified and discussed. Based on these findings, broader analysis and assessment of the paper are undertaken, highlighting its unique contributions and limitations, ultimately confirming its conclusions. In parallel to the reimplementation, grid search is employed, probing into possible limitations of HD-based algorithms for non-convex functions. Subsequently, an extension of a second order HD is undertaken. The reimplementation and extension code is publicly available on Github [1].

2 TUNING EXPERIMENT: LOGISTIC REGRESSION

A choice is made to use 10% of the datasets, facilitating extensions, while producing qualitatively similar results. Reducing the number of epochs or raising the batch size were rejected, hindering the investigation of the behavior of HD-based algorithms and having limited effect, respectively. The code shared by Baydin et al. (2017), is examined, adapted and used. Compatibility issues with newer PyTorch versions are resolved, a parser option is added, enabling dataset reduction, and new bash script utilities are developed and used to run batch tests on the models and to postprocess data. Featuring 16 plots, the first figure of the original is not reproduced here, in the interest of space.

The experiment on a Logistic Regression (LogReg) classifier uses the MNIST dataset. Figure 1 shows the the negative log-likelihood loss for training and validation along with the evolution of the first and second order learning rates α_t and β. In spite of discrepancies from the originals, due to

[1] https://github.com/COMP6248-Reproducability-Challenge/hypergradient-descent-reproduction

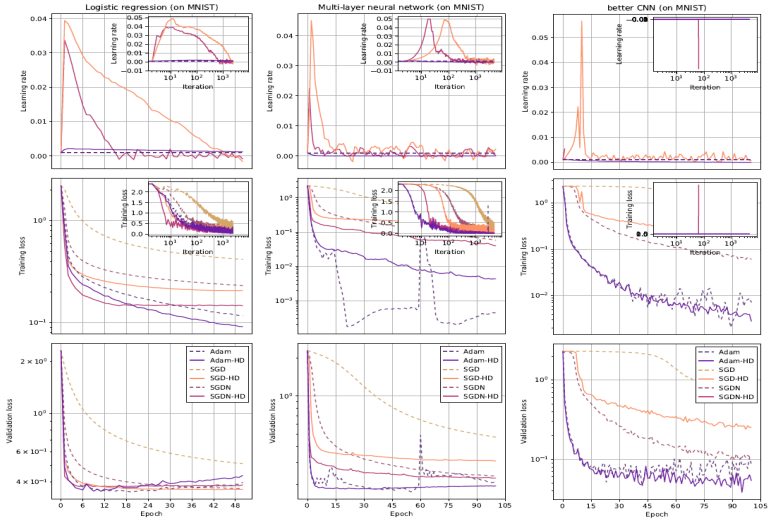

Figure 1: Comparing the behavior of HD and regular variants. Columns: *Left*: LogReg; *middle*: MLNN; *right*: BetterCNN. Rows: *top*: evolution of the learning rate α_t; *middle*: training loss; *bottom*: validation loss. One can see that qualitatively even the convergence rate follows the original.

dataset reduction and the stochasticity of the algorithms, since aggregation is infeasible, the results are qualitatively similar. In Figure 2 the results of grid search for different combinations are shown, produced in approximately 9 hours, on a digital ocean droplet with 1GB RAM, 1 vCPU and 25GB SSD, are shown in Figure 2. The scaling in the baseline versions of the optimizers follows the same patterns as in Baydin et al. (2017), while HD variants show minima for the same selection of hyperparameters as in the original paper, thus proving the reproducibility of the results.

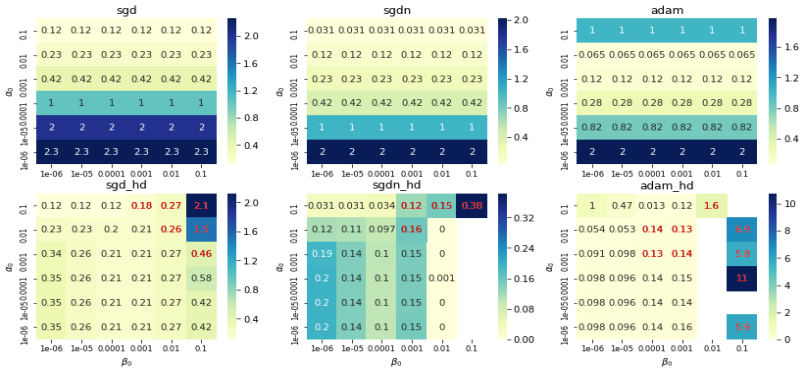

Figure 2: Grid search for different α_0 and β, with LogReg on MNIST. Results marked in red perform worse than the baseline variants at the top and blank ones indicate `NaN` values.

3 Tuning Experiment: Multi-Layer Neural Network

The experiment on a Multi-Layered Neural Network (MLNN) with two hidden layers, on the MNIST dataset, was originally carried out for arbitrary hyperparameters. It was hinted by Baydin et al. (2017) that HD might not be suitable for non-convex functions, however in the original experiment on MLNN, it appeared to be successful. It is hypothesized, here, that other values may indicate problematic performance for HD. To test this hypothersis, an extension is undertaken, in the form of grid search. To gain in speed, the new experiment is broken into two parts, running in parallel. One third of the results are produced within 6 hours using a University of Southampton GPU-server with 128GB RAM, $2\times$ Intel Xeon E5 $-$ 2623 v4 2.6GHz 8 thread CPUs, $4\times$ Nvidia

GTX 1080TI GPU cards each with 3584 cores and 11GB RAM. Two thirds are produced within 8 hours, using a Google cloud droplet with 8 vCPUs and 52GB RAM. Figure 3, confirms the results of the original experiment. For many other combinations, however, it is evident that the HD-based algorithms are problematic, particularly for Adam-HD.

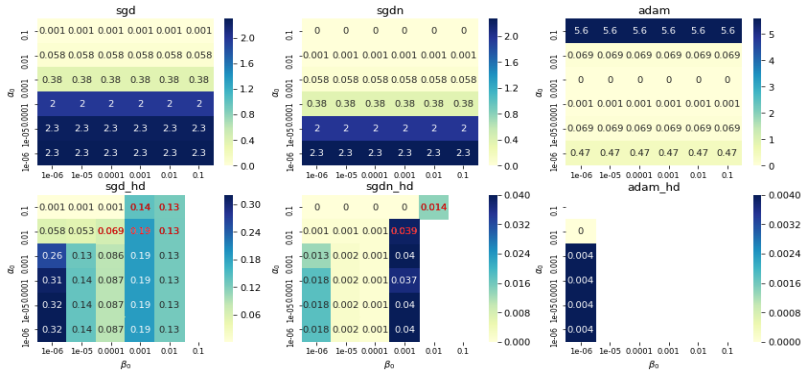

Figure 3: Grid search for different α_0 and β, with a two-layer MLNN on MNIST. Results marked in red perform worse than the baseline variants at the top and blank ones indicate `NaN` values.

4 Tuning Experiment: Convolutional Neural Network

The experiment on Convolutional Neural Networks (CNN), was originally undertaken on the VGG Net on the CIFAR-10 dataset, for arbitrary hyperparameters. Carrying on testing the hypothesis that more combinations may be provide valuable insight in the limitations of HD, the computationally costly VGG Net is dropped in favor of a simpler CNN, BetterCNN, enabling the application of grid search. BetterCNN uses a topology of two convolutional layers of size 5×5 and 3×3 kernels and a drop-out layer, followed by 2 dense layers and an output layer. Two pooling layers were included and ReLu were used as activation functions. The dataset was subject to the same transformations as in the original paper.

Its results, trained on 10% of the MNIST dataset, are produced within 14 hours, using a Google cloud droplet with 8 vCPUs and 52GB RAM, and are shown in Figure 4. For the original hyperparameters, the results are similar, proving their reproducibility. For $\beta > 10^{-3}$, however, the HD-based algorithms do not converge or cause precision errors, and out of the converging ones, 39% performs worse than the baseline. These results, illustrate that an approach with the full dataset or more difficult problems would be more appropriate, but infeasible lacking resources. Also, in addition to the results of the MLNN, the indications that HD algorithms may not be appropriate for non-convex functions are strengthened, unless it worked for very smalls betas because the learning rate was less sensitive.

5 Extension: Higher Order Hypergradients

Baydin et al. (2017), have proposed an extension toward higher order hypergradients to make the algorithms even more invariant to the hyperparameters. This requires a gradient of the objective function w.r.t β. Analytically, the 2^{nd} order HD would be, introducing a new hyperparameter γ:

$$\frac{\partial f(\theta_{t-1})}{\partial \beta} = \frac{\partial f(\theta_{t-1})}{\partial \alpha_{t-1}} \frac{\partial \alpha_{t-1}}{\partial \beta} = \frac{\partial f(\theta_{t-1})}{\partial \alpha_{t-1}} \frac{\partial(\alpha_{t-2} + \beta_{t-1} \nabla f(\theta_{t-2}) \nabla f(\theta_{t-3}))}{\partial \beta} = -\nabla f(\theta_{t-1}) \nabla f(\theta_{t-2}) \nabla f(\theta_{t-2}) \nabla f(\theta_{t-3})$$

requiring only an extra copy of the previous time step of the gradient. Thus, the hyperparameter β changes according to: $\beta_t = \beta_{t-1} + \gamma \frac{\partial f(\theta_{t-1})}{\partial \beta}$. Generally, the number of components follow a 2^n rule for the n^{th} gradient:

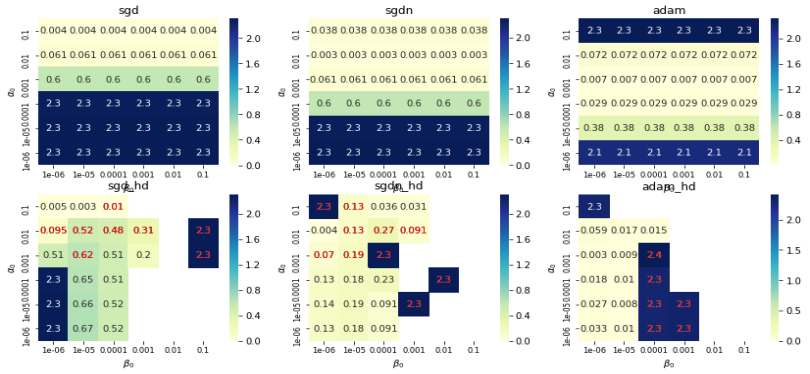

Figure 4: Grid search for different α_0 and β, with BetterCNN on MNIST. Results marked in red perform worse than the baseline variants at the top and blank ones indicate `NaN` values.

$$\frac{\partial f(\theta_{t-1})}{\partial k_n} = \frac{\partial f(\theta_{t-1})}{\partial k_{n-1}} \left(\frac{\partial f(\theta_{t-1})}{\partial k_{n-1}} \right)_{t'=t-1}, k_t^n = k_{t-1}^n + k_t^{n+1} \frac{\partial f(\theta_{t-1})}{\partial k^n}$$

where $\{k\}_{i=1}^N$ is the set of the hyperparameters, and $t' = t - 1$ indicates adjusting the gradient to the previous time step. Figure 5, suggests that HD lacks scale invariance with the problem, which is not mentioned in the original paper, especially for higher order HD, exploding in the early steps. Baydin et al. (2017) introduced the multiplicative rule which is a scaling variation and the extension was implemented based on that in order to be robust.

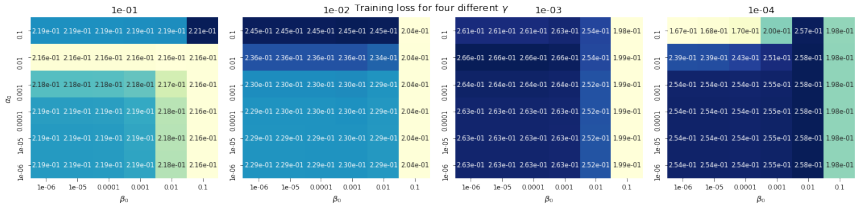

Figure 5: Grid search for four different γ, training loss converged around the optimal value for every γ while for $\gamma = 10^{-1} - 10^{-2}$ training loss had the best values.

6 Conclusion and Discussion

The experimental results presented by Baydin et al. (2017), are indeed reproducible. In spite of limited theoretical convergence guarantees and novelty, showcasing HD as a useful method, which achieves to minimize the need of hyperparameter tuning, has been accomplished. Additionally, through comprehensive implementations of the method and the development of a publicly available API, the facilitation of the method's adoption has also been successful. In the extension, possible limitations, regarding non-convex functions and higher order HD, have been suggested.

References

Luís B Almeida, Thibault Langlois, José D Amaral, and Alexander Plakhov. Parameter adaptation in stochastic optimization. In *On-line learning in neural networks*, pp. 111–134. Cambridge University Press, 1999.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.