Adopting Linear Model to Accelerate Neural Network Training

Yin Liu [*1] **Junyi Wei** [*1] **Zijun Ma** [1] **Hanying Jiang** [1] **Yihan Zhang** [1]

Abstract

Neural network (NN) models have achieved great success in many applications. However, in large neural networks, iterative model parameter updating via gradient descent requires tremendous computing time and resources. Recent study shows that the gradient-based training dynamics of NNs can be approximated by linear models. Although the approximation strictly holds with assumption of infinite NN width, our proof-of-concept evaluation indicates that NN training with small data set size and number of hidden units could also benefit from linear models. Compared to traditional propagation with gradient descent, we obtain better training efficiency by solving initial value problems (IVPs) derived from linear models with no harm in model accuracy.

1. Introduction

Recent work (Lee et al., 2019) has revealed the underlying mathematical principle of NN training dynamics. If we use $f_t(\chi)$ to denote the output of the neural network at time t related to training set χ, considering the output as a function of model parameter collection θ, we can get its Taylor expanded form, $f_t(\chi) = f_t^{lin}(\chi) + O(\theta_t^2)$, where $f_t^{lin}(\chi) = f_0(\chi) + \nabla_\theta f_0(\chi)(\theta_t - \theta_0)$ is used as an approximation of $f_t(\chi)$. As a result, the gradient descent learning dynamics can be approximated by the following ordinary differential equations (ODEs).

$$\dot{\theta}_t = -\eta \nabla_\theta f_0(\chi)^T \nabla_{f_t^{lin}(\chi)} L \tag{1}$$

$$\dot{f}_t^{lin}(\chi) = -\eta \hat{\Theta}_0(\chi, \chi) \nabla_{f_t^{lin}(\chi)} L \tag{2}$$

Since $\nabla_\theta f_0(\chi)$ and $\hat{\Theta}_0(\chi, \chi)$ could be computed and η is deterministic learning rate at initialization, we could actually solve the IVP described by the above ODEs and obtain good models without iterative propagation. **Therefore, it is promising to explore many off-the-shelf numerical solvers which have the potential to generate good models efficiently.**

[*]Equal contribution [1]University of Wisconsin-Madison. Correspondence to: Yin Liu <yinl@cs.wisc.edu>.

Table 1. Training time taken to reach certain accuracy threshold on (UCI, 2013) using 1 Intel Xeon Gold 6140 CPU and 1GB memory.

ACCURACY	TIME FOR NN/S	TIME FOR ODES/S	SPEEDUP
0.80	60.10	38.66	1.55×
0.85	75.52	39.02	1.94×
0.90	102.19	40.85	2.50×
0.95	147.73	69.83	2.12×

Table 2. Training time taken to reach certain accuracy threshold on (UCI, 2007) using 1 Intel Xeon Gold 6140 CPU and 1GB memory.

ACCURACY	TIME FOR NN/S	TIME FOR ODES/S	SPEEDUP
0.80	118.50	59.42	1.99×
0.85	140.88	72.52	1.94×
0.90	189.30	76.19	2.48×
0.95	201.45	90.97	2.21×

2. Experiment Evaluation

We conduct experiments on some multivariate and labeled data sets (over 1000 training instances) (UCI, 2013), (UCI, 2007), etc. Two-layer fully-connected NN models with various numbers of hidden units respectively on different data sets are built to do classification and cross entropy is used as the loss function. For simplicity, the learning rate is consistent throughout any single experiment. Moreover, we do both the traditional training and the IVP solving on the whole data set (i.e., without mini-batch). To solve the IVP, we choose the Backward Differentiation Formula (BDF) (Byrne & Hindmarsh, 1975) solver provided by Scipy since it scales well and handles the possible stiffness.

Table 1 and Table 2 show the requisite training time for reaching specific accuracy threshold. Unlike epochs which strictly stick to the iterative manner, BDF solver applies adaptive stepping, which outperforms the iterative propagation in terms of efficiency (e.g., over 2 times speedup to reach 90% training accuracy in experiment). In addition, applying the weights and biases obtained from solver to test data set, the accuracy is often comparably high (i.e., both around 93% using the parameters from traditional training and IVP solving, given appropriate hidden unit numbers).

References

Byrne, G. D. and Hindmarsh, A. C. A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 1 (1):71–96, 1975.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

UCI. Magic gamma telescope dataset. Website, 2007. https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope.

UCI. Banknote authentication dataset. Website, 2013. https://archive.ics.uci.edu/ml/datasets/banknote+authentication.