Robustness of SGD and Newton Methods

CS 269: Optimization Methods for Deep Learning, Project 5

Zhiping Xiao (Patricia), Haoran Wang *

1 Introduction

The goal of this project is to check how sensitive stochastic gradient and Newton methods are to the parameters, considering the implementation of our Matlab code. [1]

We've learned that there are many variants of Newton methods for NN, proposed for handling complex cases such as large data or deep networks. Working on small sets from project 4, we made a reasonable decision of using the standard Newton to verify robustness.

2 Experiment Settings

All settings and data sets are the same as project 4, except that we add the regularization term back in *sgd.m* with parameter $C = 0.01$, kept the same with the original code's defaults, momentum of SGD is 0.9. We run 500 epochs for SGD, 50 iterations for Newton. The number 50 is chosen for:

- One iteration of Newton is typically $10 \sim 20$ times slower than one epoch of SGD. In terms of time-consumption they are reasonably comparable.
- We observed that Newton method converges fast, and the test accuracy is typically converging before 50 iterations.

The settings we're going to test for the standard Newton method are as listed in Table 1. Percentage of data for subsampled Hessian, in the code implementation, is controlled by parameter **SR**; and **lambda**, the detailed updating procedure is influenced by **drop**, **boost**, ρ, [2] etc., when λ is not zero.

PARAM	VALUES
SR	0.2, 0.5 (0.1, 0.25)
lambda	0, 1

Table 1: Settings to test with Newton

We have the Newton linear system:

$$Gd = -\nabla f(\theta)$$

Comparing it to the Levenberg-Marquardt method's modification:

$$(G^S + \lambda I)d = -\nabla f(\theta)$$

and knowing that λ is always updated by multiplying something with the previous λ, by having $\lambda = 0$, Levenberg-Marquardt method is disabled.

*Contact us at *{patriciaxiao, hwan252}@g.ucla.edu* for more details on this project. We are group #1. Special thanks to Xin Jiang and Kewei (Vivian) Cheng for inter-group discussions with us.

[1] `https://github.com/cjlin1/simpleNN`

[2] ρ is given as fixed constant instead of parameterized, in the implementation.

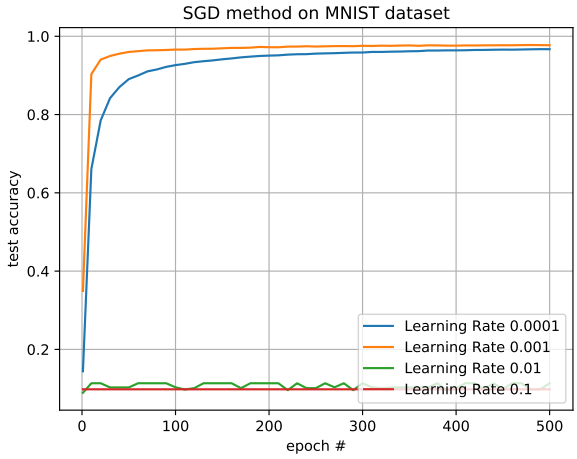

Figure 1: *Test Accuracy on MNIST, using SGD optimizer, recorded every 10 epoch.*

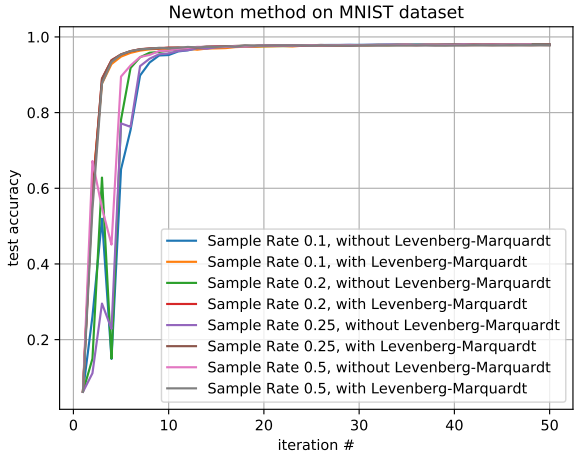

Figure 2: *Test Accuracy on MNIST, using Newton method, recorded each iteration.*

As for the stochastic gradient method part, everything remains the same as before except that this time we test different learning rates: 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}.

The random seed that was supposed to be changed 10 times in project 4 was not fixed to be 111, the same as default.

Datasets we use are the same as before, the mini-sized 5000 training sets for MNIST and CIFAR10, and the original test sets.

3 Results and Discussion

The time complexity is mostly influenced by the dataset and the type of optimizer. SGD on MNIST typically costs $12 \sim 15$ s/epoch, on CIFAR10 usually around 20 s/epoch; Newton with 20% sample rate on MNIST typically costs around $220 \sim 240$ s/iteration, and on CIFAR10 typically $460 \sim 500$ s/iteration; Newton with 50% sample rate around 30 minutes to an hour each iteration, and suffers from the hazard of exceeding the memory limit.

From the running time we conclude that the Newton method is more sensitive to the data size, and its time-efficiency might need further improvement. Another issue is that the Newton program takes so much computational resources, that sometimes it is not affordable to the computer.

Having Hessian subsample-percentage being 50% crushes the memory of a 16GB-RAM Desktop PC with i7 processor running Windows 10. Further more, it is reported by Xin Jiang that 8GB-RAM Desktop with latest Ubuntu will be crushed by 20%-sample-rate Newton. Desktop computer with Mac OS is not available to us thus not tested. We found the Newton method runs the fastest with Mac Pro (Laptop), and it would hardly crush the memory of a laptop computer with Windows operating systems either.

This phenomenon indicates that a shortcoming of the Newton method is its high memory-complexity when computing Hessian. From our observation, laptop is better than desktop in running it, mac is better than windows, and the larger RAM the better.

But still, we want to shed light on the relations between the subsample size and the performance. Therefore, we add two additional options of sample rate (**SR**), namely 0.1 and 0.25, so that we could gain a better understanding upon the influence this parameter. Although a notice from Prof. Lin came lately that we could reduce the number of

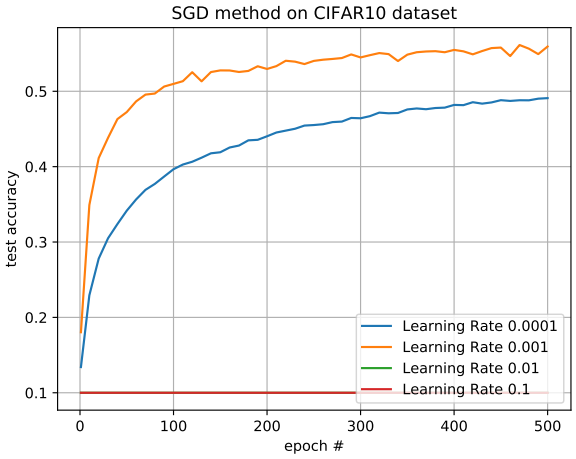

Figure 3: *Test Accuracy on CIFAR10, using SGD optimizer, recorded every 10 epoch. Green line is hidden behind the red one.*

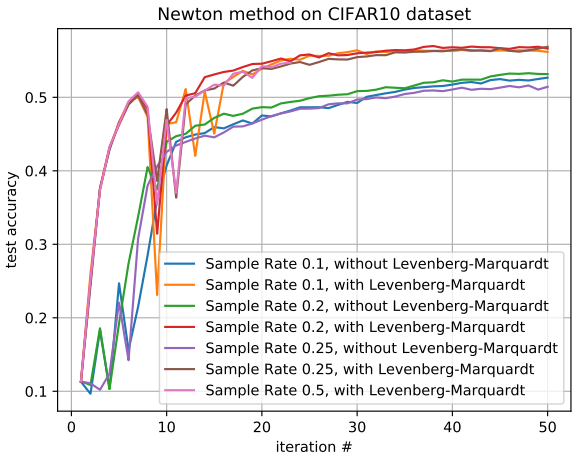

Figure 4: *Test Accuracy on CIFAR10, using Newton method, recorded each iteration. Sample rate 50% hasn't finished yet, thus we only report the results so far.*

categories to fit in the memory, at that stage we've finished running most of the experiments, and we've found a machine that is capable of running the 50% CIFAR10 Newton. The problem we face is that we don't have enough time to finish running the experiments on 50% subsample rate before the report-submission deadline. We'll include the results in the presentation.

As is shown by comparing Figure 1 and 2, or Figure 3 and 4, the Newton method with no more than 30 iterations is able to achieve similar test accuracy with SGD after running 500 epochs.

With SGD, the smaller the learning rate is, the more stable the curve appears, but the slower the algorithm converges. When the learning rate is too large, the loss will be exploded and reflected as loss being NaN and testing accuracy fluctuate around that of a random guess. We conclude that SGD is pretty sensitive to learning rate. The highest accuracy of SGD in the end is around 97.71% on MNIST and 55.94% on CIFAR10, achieved by learning rate 10^{-3}.

With Newton methods, it shows in Figure 2 and 4 that both sample rate and Levenberg-Marquardt (LM) method affect its performance. We've observed that introducing LM method makes the learning curve increase faster and more stably, and achieve higher test accuracy in the end. Larger sample rate leads to faster learning. Newton methods generally suffer from severe fluctuation at the beginning, and gradually reach stable accuracy. Without LM method, larger learning rate might end up in lower accuracy in the end. The highest test accuracy at iteration 50 is achieved only when Levenberg-Marquardt method applied. That is, 97.94% on MNIST dataset, with sample rate 20%, and on CIFAR10 56.87%, achieved when the sample rate is 25%.

From our observation, with LM method, the sample rate has little influence on the final test accuracy. This finding applies to both datasets, thus we assume that with sample rate 50%, the final test accuracy with LM will be around 57%. From the other curves, the final accuracy without LM might be about 50%.

Newton methods have better robustness in general, when given adequate amount of memory space, and allow enough time.