Project: Robustness of Newton Methods and Running Time Analysis

Last updated: June 10, 2021

Outline

1. Robustness of Newton methods
   - Part 1: On CNN_4layers
   - Part 2: On VGG11
2. Running time of two implementations of Gauss-Newton Matrix-vector products

Outline

Goal I

- From past projects we have known that stochastic gradient is sensitive to the learning rate
- We would like to check the situation of Newton methods
- We want to know whether or not Newton is more robust than commonly used optimizers.

Outline

Settings I

- In part 1, let's run experiments on CNN_4layers
- However, to avoid lengthy training time, let's consider a 5000-instance subset at this directory
- Let's use MNIST-5000 and CIFAR10-5000 training data.
- We still use the full test data

Parameters Considered (Newton Method) I

- The Newton implementation is available in simpleNN
- Both Python and MATLAB implementations are available but you are suggested to use Python as Python can be run directly without modifying the config file or driver file.
- Check README for their use

Parameters Considered (Newton Method) II

- For the maximal number of CG iterations, to save time, let's only consider

$$\texttt{CGmax} = 80$$

- For calculating the whole gradient, in the Newton method we split all data to several batches. The batch size doesn't affect the accuracy. Instead, it affects the efficiency. To strike a balance between time and memory space, we suggest you to consider

Parameters Considered (Newton Method) III

`bsize`= 3000

- We check the percentage of data for subsampled Hessian:
  - Percentage of data for subsampled Hessian: 5%, 20%
  - With/without

    Levenberg-Marquardt method

Parameters Considered (Stochastic Gradient Methods) I

- We would like to run
  - simple stochastic gradient + momentum
  - Adam
- We check the following initial learning rates

$$10^{-5},\ 10^{-4},\ 10^{-3},\ \text{and}\ 10^{-2}$$

- And we only consider the following batch size:

$$\texttt{bsize} = 512$$

Parameters Considered for All Optimizers

- For regularization parameters C, let's consider

$$\texttt{C} = 0.1$$

  for all optimizers (including Newton, SG with momentum, and Adam).
- Remember to set correct input dimension on each data set, e.g., set `--dim 28 28 1` for MNIST-5000 data.

Checking the Convergence I

- Because we didn't provide the validation set, we cannot run a standard training, validation, and prediction procedure to compare optimization method.
- Let's simply check the relation between epochs and test accuracy (or you can say validation accuracy if you treat test set as the validation set)
- You may also check, for example, the best accuracy in the entire procedure or the final accuracy
- Running time may not be important as you now run jobs on different machines with various loads.

Checking the Convergence II

- You now have

  4 settings for Newton

  and

  4 settings for SG with momentum

  and

  4 settings for Adam

- You may design figures to show the comparison results. For example,

  # epochs or training time vs. test accuracy

Checking the Convergence III

- Visualization is always a concern. For example, you may not want to draw eight curves in the same figure

Outline

Accuraccy Comparison on VGG11 I

- We would like to study Newton method on deeper networks.
- In this part, let's run experiments on VGG11.
- However, it takes long if run on CPU. Even CIFAR10-5000 data, Newton may take over three days on some workstation machines (linux5 to linux14), while take one or two days on linux1 to linux4.

Accuraccy Comparison on VGG11 II

- The jobs will be reniced by workstations if we run too long. So if you want to use workstations, you may need to run jobs in multiple machines and start the project earlier.
- Here you can see which workstation machine has lower CPU usage. Notice that linux15 may run very slowly due to the old CPU architecture. linux1 to linux4 are the fastest while linux5 to linux14 are fine. Here is hardware information for workstation machines.

Accuraccy Comparison on VGG11 III

- Because each job takes a long time and it may be interupted for various reasons, you can consider running the same task on multiple machines simultaneously.
- We have committed the VGG network, please git pull the latest code.
- You only need to run CIFAR10-5000 because MNIST image sizes is too small to be run on VGG11.

Parameters Considered on VGG11 I

- All settings are the same as part 1.
- Notice that for regularization parameter C, we need to specify $C = 0.1$ for all optimizer. If you directly use the default regularization parameter, then Newton on VGG11 will encounter numerical error (so the process will not terminate).

Outline

Gauss-Newton Matrix-vector Products I

- We talked about two ways
- One is solely by back propagation. The complexity is the lowest, but the memory consumption is huge
- The other is

$$\text{forward} + \text{backward}$$

- We want to analyze the running time of matrix-matrix products by using the MATLAB code.
- An option `-Jacobian` decides which one is used.
- For other parameters let's use the default ones (e.g., 5% subsampled Gauss Newton)

Gauss-Newton Matrix-vector Products II

- For this part, we need to run at least one data set (MNIST or CIFAR10). Notice that we should use full data. Otherwise each iteration takes too little time and the results may not be accurate
- We only consider CNN_4layers.
- No need to run many iterations. We suggest you can run 5 iterations.
- By profiling we may check if for the matrix-matrix products, the result is consistent with our complexity analysis

Gauss-Newton Matrix-vector Products III

- How about the situation of the total time? Any operations with lower complexity but take more time? Why did that happen? From the experiences of project 4, you may see some slow MATLAB operations.

Presentations and Reports I

- We've announced students who are selected to present on NTU COOL.
- Please do a 10-minute presentation (9-minute the contents and 1-minute Q&A)