Assignment 1 of MATP6960: programming

(Due together with theoretical questions at 11:59PM on Oct-08-2021)

Instruction: Each student needs to submit the source code file and a report by Latex. Include your solutions to the theoretical questions in the report file. 40% of the total credit will be for the theoretical questions and 60% for the programming. **You will be evaluated based on whether you have all the results in the requirements below, and also the report**. Compress your source files (**DO NOT send the data.**) and PDF report file into a single .zip file, name it as "MATP6960_Assignment1_YourLastNameInitial", and send it to `optimization.rpi@gmail.com`

Problem description

In the class, we introduced the vanilla stochastic gradient method (vanilla-SGD) and its proximal version. Also, we introduced three momentum-accelerated proximal stochastic gradient methods, namely, Hybrid-SGD [1], SpiderBoost [2], and PStorm [3]. In this project assignment, you are required to implement these methods to train a neural network. Let $f_{\boldsymbol{\theta}}$ denote a neural network parameterized by $\boldsymbol{\theta}$. Then the training process will be to solve the following problem

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i) + \lambda \|\boldsymbol{\theta}\|_1, \tag{1}$$

where ℓ is a loss function such as the logarithmic softmax function in the provided code, and $\lambda \geq 0$ is a parameter to control the sparsity. One benefit of using a sparsity regularizer is to obtain a compressed model (i.e., with sparse weight matrices) so that storing the model will not take much memory.

Requirements

Read the provided code `SGD_FashionMNIST.py`. In the code, two neural network architectures are defined. One uses convolutional layers and another fully connected. Make sure you understand the data loading and model training process. The following lines in the code replace `optimizer.step()` and perform the stochastic gradient update.

```
alpha0 = 1
for p in model.parameters():
      # perform update: x_new = x_old - alpha0/sqrt(iter) * stoc_grad
      p.grad.data.mul_(-alpha0/pow(iter, 0.5))
      p.data.add_(p.grad.data)
```

You are required to do the follows.

1. Implement Hybrid-SGD, SpiderBoost, and PStorm by either replacing the codes shown in the above, or writing a custom optimizer. For these algorithms, you can refer to the class notes for their update formula or refer to the papers in the reference. Hybrid-SGD is in Algorithm 1 of [1], and its parameter setting is in Theorem 1 and in section 6.1. SpiderBoost in in Algorithm 2 of [2] with some parameters set in Theorem 2. PStorm is in Algorithm 1 of [3] with parameter settings in Theorem 2.

2. Solve (1) with $\lambda = 0$ and a small $\lambda > 0$ (e.g., $\lambda = 2 \times 10^{-4}$, or you can tune it) by the vanilla SGD, Hybrid-SGD, SpiderBoost, and PStorm. The vanilla SGD is included in the provided code, but you can tune its step size. You can use either LeNet5 or the fully-connected architecture, or you can define another network architecture. However, **your testing accuracy must be at least 89% when** $\lambda = 0$. You are required to report the architecture you use. In addition, you are required to report the testing accuracy, the density level of your model parameters and the violation of stationarity. Plot these results in terms of the epoch number. If $\lambda = 0$, the violation of stationary at a point $\boldsymbol{\theta}$ is measured by $\|\nabla\phi(\boldsymbol{\theta})\|$. If $\lambda > 0$, the violation of stationary at a point $\boldsymbol{\theta}$ is measured by $\big\|\mathbf{prox}_r\big(\boldsymbol{\theta} - \nabla\phi(\boldsymbol{\theta})\big) - \boldsymbol{\theta}\big\|$, where

$$\phi(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i), \quad r(\boldsymbol{\theta}) = \lambda\|\boldsymbol{\theta}\|_1.$$

3. Discuss what you observe in the test. For example, how do you tune the algorithm parameters, how these parameters affect the performance of these algorithms, which algorithm performs better, how does λ affect the density of the model parameter and the testing accuracy?

References

[1] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

[2] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. In *Advances in Neural Information Processing Systems*, pages 2406–2416, 2019.

[3] Y. Xu. Momentum-based variance-reduced proximal stochastic gradient method for composite nonconvex stochastic optimization. *arXiv preprint arXiv:2006.00425*, 2020.