Problem Set - ONLINE LEARNING RATE ADAPTATION WITH HYPERGRADIENT DESCENT

Mohbat, Momin, Huzaifa, Tobias, Xinyan

November 2021

Alternative to the hypergraident descent

We have seen how hypergradient descent can achieve better convergence and accuracy by utilizing a changing learning rate. In this problem we will compare hypergradient descent to a similar algorithm that has the same goal but utilizes a different approach to illustrate the following learning outcomes: (1) that there are many ways to improve a model using the idea of an adaptive learning rate, and (2) that the hypergradient descent method can be improved upon to achieve even better results.

For this problem, we will be comparing hypergradient descent to the MARTHE algorithm proposed by Donini et. al. [1]. MARTHE stands for Moving Average Real-Time Hyperparameter Estimation. It works similar to Hypergradient Descent, but instead of merely using information from the past and current iterate as Hypergradient Descent does, MARTHE uses a moving average of gradient data from the past k iterates (where k is a constant). Using this extra data can improve the prediction of the learning rate. Since, using too much past data could make the algorithm resistant to sudden changes in the loss surface, MARTHE forgets old gradient information, hence it takes "moving average".

Overall, when compared to Hypergradient Descent, MARTHE can help prevent us from being too "shortsighted" in estimating the learning rate (which can lead to an underestimation in the learning rate, as the result from HD will be too small if we are in a flat region of the graph). However, it is more complicated to implement and introduces additional hyperparameters and computational expenses compared to Hypergradient Descent.

You should refer to the referenced paper for more details on the algorithm's implementation to complete these problems (the paper is very short, so this should not be unreasonable).

1 Part 1

As instructed by Donini et. al. [1], use pytorch to implement a VGG network on the CIFAR10 dataset using SGD as the optimizer. Using batch size 128, and learning rate 0.0001, train the model for at least 20 epochs. Split the data into training and validation sets and plot validation accuracy vs epochs and validation loss vs epochs using matplotlib.

2 Part 2

Using the algorithm described by Donini et. al. [1], *Online Learning Rate Adaption with Hypergradient Descent* reproduce SGD with Hypergradient Descent. Run the optimizer SGD-HD on CIFAR-10, with batch size 128, and learning rate 0.0001. Set Hyper-parameter learning rate β at 0.0001. Train for at least 20 epochs. Split the data into training and validation sets and plot validation accuracy vs epochs and validation loss vs epochs using matplotlib.

3 Part 3

Refer to the paper *MARTHE: Scheduling the Learning Rate Via Online Hypergradients*.

Using the pseudo-code for the MARTHE algorithm given in the paper [1], again create a python function with PyTorch that implements a VGG network using SGD on CIFAR10, but this time, implement the MARTHE algorithm to improve the accuracy. Use $\mu = 0.99999$, Hyper-parameter learning rate β at 0.0001, and momentum to be 0.9. Again, refer to the paper for details on these hyper-parameters. Plot the validation accuracy and validation loss with respect to epochs along with the validation accuracy/loss from Part 2 on the same graph.

4 Part 4: Graduate Students

4.1 ResNet18 on CIFAR100

Repeat parts 1 and 2, but this time use RESNET-18 as the model instead of VGG, and use CIFAR100 instead of CIFAR 10. The optimizer is still SGD. How does the RESNET result graph compare to the VGG result graph?

4.2 Your observation

Discuss: which optimizer performs better with VGG and RESNET? What distinguishes MARTHE from HD, give your comments.

Solutions

Please refer to the Jupyter Notebook for specific results.

Summary of solution

The final graphs for parts 1, 2, and 3 should resemble the left and center graphs from Figure 3 of the paper for full credit. Partial credit may be awarded if the graph is nearly correct; 0 credit will be awarded if the graphs do not show that the MARTHE implementations have better accuracy than the normal implementations. The contents of the code does not matter (as long as the student is actually implementing these methods themselves); the student should be graded solely on the quality of the graphs generated by the program and not by the code. However, if a solution key for implementations is needed for reference, the code at https://github.com/awslabs/adatune/tree/master/adatune may be used. The student may not copy or use this code in their solution. Part 4 is correct as long as a better value of μ is selected and the student demonstrates via experiments that their value produces better results than before.

References

[1] Michele Donini, Luca Franceschi, Orchid Majumder, Massimiliano Pontil, and Paolo Frasconi. Marthe: Scheduling the learning rate via online hypergradients. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*. AAAI Press, 2020.