Intuition Report 2

Ramtin Mojtahedi Saffari - 20307293

Assessed link 1: ConvNetJS Trainer demo on MNIST

Self-reflection

- **Short Summary**

The provided link assesses the six well-known gradient descent optimization algorithms, including Stochastic gradient descent(SGD), SGD with momentum, Adagrad, Windowgrad, Adadelta, and Nesterov. The link is a browser-based Javascript library for training deep learning models (neural networks). The results of the trained model are being compared with the mentioned algorithms through loss function, testing, and training accuracies. The default network is a fully connected two layers ReLU network with a batch size of 8, the learning rate of 0.01, and an L2 regularization weight decay of 0.001 to predict ten classes in the output layer.

- **Hypothesis and Expectation**

Among the provided optimization algorithms, I hypothesize that it works better compared to the other algorithms. This is because Adadelta decreases the amount by which the learning rate is adaptive to coordinates. It is not set with a default

learning rate as it uses the amount of change itself as calibration for future change (solve the monotonic decrease of learning rate).

- **What I Achieved**

The Adadelta worked better than the other algorithms when I trained and ran the simulations, and Windowgrad showed the lowest performance. These performances were demonstrated based on the test and training dataset (highest loss values for Windowgrad and lowest values for the Adadelta).

- **What I Learned**

From the provided optimization algorithms, SGD has the simplest algorithm. SGD with momentum is an extension to the SGD algorithm, which helps to accelerate the converging. Adagrad is another algorithm that works better than SGD and SGD with momentum. This algorithm is simpler than SGD with momentum and has an adaptive gradient algorithm, and each of its parameters has its learning rate. These specifications help it escape saddle point much better than the SGD and SGD with momentum and faster convergence. One of the disadvantages of the Adagrad is that it accumulates the squared of the gradients, and it causes the learning rate to be shrunk and finitely becomes very small [1]. Windowgrad shows the worst performance compared to all algorithms. However, Nesterov and Adadelta work better than the others. The implementation of these two algorithms is closed to each other. However, Adadelta shows a bit better performance. Nesterov is a method that gives foresight to the momentum term. One of the disadvantages of Nesterov is that it doesn’t update each parameter to perform larger or smaller updates depending on their importance, and it is not an adaptive algorithm. On the other hand, Adadelta requires no manual tuning of a learning rate, and it seems robust to noisy gradient information, different model architecture choices, various data modalities, and selection of hyperparameters [2]. Overall, it has the benefits

of keeping the learning rate optimally high (instead of storing all of the past squared gradients, it recursively decaying the average of all past squared gradients) and is an adaptive algorithm.

Suggestion and Filling the Gaps

As one of the gaps in the link, it should add Adam as one of the well-known and promising algorithms to the analysis. This is important because Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. Also, Adam is relatively easy to configure, where the default configuration parameters do well on most difficulties [3].

Assessed link 2: ff-Net

Self-reflection

- **Short Summary**

The provided link depicts the effects of the learning rate and regularization on the loss values. This interactive link can change the learning rate and regularization values to see how these changes affect the loss values in a feed-forward neural network (initialized by random network parameters).

- **Hypothesis**

I hypothesize that selecting the learning rate and regularization parameter very high or too tiny causes the loss function to be increased as it causes underfitting or overfitting problems.

- **Testing the hypothesize and what I have learned**

In both cases, I selected learning rate and regularization near zero or the highest values, which causes the loss values to increase too fast. The high values for the learning rate cause the model to perform well on the training dataset. When it comes to the test data, it can't perform well on it, and this causes overfitting of the model (can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck) [4]. On the other hand, if the regularization parameter becomes too high, the model will be simple. This increases the risk of underfitting, and the model won't learn enough about the training data to make useful predictions. Also, if the regularization parameter becomes too low, the model will be more complex, and you run the risk of overfitting our data [5].

Based on testing different conditions, I found that the minimum loss value happens when the learning is about the middle, and regularization values are on the bar's one-fourth. In this case, the loss value becomes about 0.11. This experiment is aligned with my hypothesis to select the values of learning rate and regularization. Overall, large learning rates result in unstable training, and tiny rates fail to train. That's why the concept of adaptive learning rates can accelerate training and alleviate some of the related issues. In addition, selecting the appropriate learning rate and regularization values rather than extreme values can train a generalized model and prevent the overfitting or underfitting of the trained model.

Self-evaluation:

In this intuition report, I have gone through an in-depth analysis of two of the provided links. In my assessment, I have completely considered the required expectation for deep exploration, including proposing a hypothesis and what I expected, reporting on what I achieved and explored, in-depth discussion of what I have learned, and providing gaps and recommendations to fill them. Considering the quality and assessment level, I deserve to get the full mark (4 points) for this intuition report.

In advance, thank you very much for your time and consideration of this report.

Best regards,

Ramtin

References

[1] Kapoor, N. (2020, November 19). *A dive into Optimizers - Towards Data Science*. Medium. https://towardsdatascience.com/optimizers-88694509311c

[2] Beiranvand, V., Hare, W., & Lucet, Y. (2017). Best practices for comparing optimization algorithms. *Optimization and Engineering*, *18*(4), 815–848. https://doi.org/10.1007/s11081-017-9366-1

[3] Battini, D. (2018, November 8). *Adam Optimization algorithms in Deep Learning*. Tech-Quantum. https://www.tech-quantum.com/adam-optimization-algorithms-in-deep-learning/

[4] Palachy, S. (2019, September 16). *Understanding the scaling of L^2 regularization in the context of neural networks*. Medium. https://towardsdatascience.com/understanding-the-scaling-of-l%C2%B2-regularization-in-the-context-of-neural-networks-e3d25f8b50db

[5] Brownlee, J. (2020, September 11). *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/