Hands-on 3

Optimization

Instructor: Dr. Souvik Chakraborty

APL 745: Deep Learning for Mechanics

February 7, 2024

Submission due on February 21, 2024

Instructions:

(i) You are allowed to discuss in a group; however, you must submit your own handwritten homework copy (no computer-typed submission will be accepted). Further, copying homework from your friends is forbidden. If found copied, the homework submission will be considered invalid for all the students involved and will be graded zero.

(ii) Write all the steps, including your reasoning and the formulae you referred to, if any. If sufficient reasoning is not provided and steps are unclear, step marks will not be given.

(iii) For practical submissions, the codes are accepted in *.ipynb* or *.py* format.

(iv) Unless mentioned otherwise, only *numpy*, *pytorch* and *matplotlib* libraries may be used. Direct commands for algorithms to be implemented are not allowed.

(v) The *.rar* file containing all submission related files shall be named in the format, **Name_Entrynumber.rar**

Question 1. Optimizer implementation for a feed-forward neural network

Prerequisites:

1) Download MNIST dataset using *torchvision*. The documentation for the same can be found in this link.
2) Transform the labels provided in the dataset using one hot encoding.
3) Create three datasets with following specifications,
   - Training dataset: 5000 samples
   - Validation dataset: 1000 samples
   - Testing dataset: 2000 samples

Network:

1) Network: Train a network with 2 hidden dense layers. Each hidden layer must have 200 nodes and ReLU activation. The output layer should have softmax activation. Choose the number of neurons in the input and out layers based on the dataset provided.

Tasks:

1) **Optimizer implementation:** Write code to implement the following optimizers from scratch:
   (Note: Utilize automatic differentiation in PyTorch and do not use any optimizer modules available in the PyTorch)
   - Stochastic Gradient Descent (SGD)
   - Momentum Optimizer
   - Nesterov Accelerated Gradient (NAG) Optimizer
   - AdaGrad Optimizer
   - RMSprop Optimizer
   - Adam Optimizer
   - Adam Optimizer with Bias Correction
2) **Parameter Tuning:**

- Experiment with different values for optimizer parameters, such as learning rate, damping parameter for momentum update, and coefficients for computing running averages of the gradients. Show at least three set of values and comment on the observations.

3) **Performance Metrics:**
   - Calculate and compare the following metrics:
     - Training accuracy: $\frac{\text{Number of correctly classified training samples}}{\text{Total number of training samples}}$
     - Validation accuracy: $\frac{\text{Number of correctly classified validation samples}}{\text{Total number of validation samples}}$
     - Loss convergence: Observe the loss variation (cross-entropy) with the number of epochs.
4) **Plots and Analysis:**
   - Create plots showing the trends of training and validation accuracy with number of epochs as training progresses.
   - Compare the performance of your implemented optimizer with PyTorch's Adam optimizer in the same plot.
   - Analyze how changes in optimizer parameters affect convergence and accuracy.

Submission:

Submit your Python code, along with a report describing your observations and insights from the experiments. Include visualizations and comparisons of the training and validation accuracy trends for each optimizer under different parameter settings. Additionally, showcase the impact of the implemented learning rate schedulers on model training.