Assignment 1

Introduction to Artificial Intelligence and Its Basic Deep Learning Applications (GEV6130)

March 25, 2024 (due on April 1, 2024 by 11:59pm)

Please provide detailed comments (i.e., annotations) for every statement. If there is any collaboration with your colleague, then please mention it. **Turn in your assignment via email at wy.shin@yonsei.ac.kr by attaching a *pdf* file as well as an *.ipynb* file, which should contain the source code, comments, and output for each problem.**

1. (100 points) [**Neural Network Design**] Implement a fully-connected neural network consisting of one input layer, one hidden layer, and one output layer. Use 'sigmoid' as an activation function. Do not use any bias term in the network. Use the following *default settings* unless otherwise specified.

- The input layer size, hidden layer size, and output layer size are given by 2, 8, and 1, respectively.
- The stochastic gradient descent (SGD) optimizer is used for the weight updates by assuming that the batch size is 1.
- The learning rate is fixed to 0.01.

Do **NOT** use any machine learning package except for some libraries such as Numpy and matplotlib.

(a) (30 points) Plot the **training loss** (with respect to the mean squared error per epoch) versus the number of epochs using the file "training.txt" for the training dataset. At the same time, plot the **validation loss** (with respect to the mean squared error per epoch) versus the number of epochs using another file "test.txt" for the validation dataset. That is, validate your trained model for every epoch. Make discussions based on your experimental results by comparing the training loss with the validation loss. You may shuffle data samples in the training and validation sets to impose more randomness.

(b) (30 points) Plot the training loss (with respect to the mean squared error per epoch) according to **different hidden layer sizes**, ranging from 2 to 16, after training your model sufficiently for each case. Make discussions based on your experimental results. You may shuffle data samples in the training set to impose more randomness.

(c) (40 points) Instead of the SGD, use the **adaptive gradient (Adagrad) optimizer**, where the (initial) learning rate is set to 0.01. Plot the training loss (with respect to the mean squared error per epoch) versus the number of epochs. Make discussions based on your experimental results. You may shuffle data samples in the training set to impose more randomness.

Refer to the link for the detailed description of Adagrad: https://ruder.io/optimizing-gradient-descent/.