IE643 Deep Learning : Theory and Practice **Aug-Dec 2020**

Assignment 1: Due On 29^{th} September 2020 (11:59 PM IST)

1 Instructions

Answer all questions. Write your answers clearly. You can score a maximum of 53 marks in this assignment. Use the code in file `Feed_forward_Net(Assignment 1).ipynb`.

Please make sure that all your answers are present in a single pdf document. If you use python notebook (.ipynb) files, make sure that your answers and plots are visible in .ipynb file. Upload on moodle, the python code, plots and pdf document as a single zip file named as "IE643_rollno_assignment1.zip". All your files within the zip file should follow similar naming convention. There will be no extensions to the submission deadline.

2 Questions

1. **[Use only Python]** Take the feed-forward network code file `Feed_forward_Net(Assignment 1).ipynb` posted in Moodle. Check that the code uses a simple squared error (SE) as loss function. Also note that the code currently has support for only the logistic sigmoid activation function. Answer the following:
   (a) The tanh activation function is given by $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. Implement the python functions to compute $\tanh(z)$ and its gradient. [**2 marks**]
   (b) The linear activation function is given by $\text{linear}(z) = z$. Implement the python functions to compute $\text{linear}(z)$ and its gradient. [**2 marks**]
   (c) The ReLU activation function is given by $\text{ReLU}(z) = \max\{z, 0\}$. Implement the python functions to compute $\text{ReLU}(z)$ and its (sub-)gradient. [**2 marks**]
   (d) Consider the simple data set in the code. Now construct a feed forward neural network with 5 layers, where the first and second hidden layer has linear activation, third layer has tanh activation, fourth hidden layer has ReLU activation and the output layer has logistic sigmoid activation. Train using the simple data for 1000 epochs using learning rate 0.001 and plot the loss against epochs. [**4 marks**]
   (e) Now complete the training code for minibatch setting and consider the minibatch size to be 20. Note that the code for creating mini-batches is already provided in the file. Consider the same neural network architecture in Question 1d. Train using the simple data with the minibatch size as 20, for 1000 epochs with learning rate 0.001 and plot the loss against epochs. [**4 marks**]
   (f) Compare and contrast the behavior of the training with and without minibatches. Use the loss obtained in the epochs to justify your answers. [**2 marks**]
2. **[Use only Python]** Take the feed-forward network code file `Feed_forward_Net(Assignment 1).ipynb` posted in Moodle. Illustrate the exploding gradient and vanishing gradient problems using appropriate neural network architecture and activation functions. Justify the architecture you used, indicate how you checked the exploding gradient and vanishing gradient problems, and explain your observations. [**5 marks**]

3. **[Use only Python]** Take the feed-forward network code file `Feed_forward_Net(Assignment 1).ipynb` posted in Moodle. Add to the code the implementation of one-hot encoding of the labels and cross-entropy (CE) loss function. Use the digits data set from `scikit-learn` package (please refer to `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html` for loading the digits data set). Construct a neural network with the input layer consisting of 64 neurons, two hidden layers H_1 and H_2 where H_1 consists of 256 and H_2 consists of 128 neurons. Assume logistic sigmoid activation function for H_1 and H_2. Construct the output layer appropriately for squared error (SE) loss and for cross-entropy (CE) loss. Now, record the following observations:

   (a) Explain how many neurons you used in the output layer for squared error (SE) and cross-entropy (CE) loss functions and indicate the type of activations used in the output layer for each loss function. [**4 marks**]

   (b) Consider the simple stochastic gradient descent algorithm with 200 epochs and mini-batch size 1 for both CE and squared error (SE) loss functions. Fix the learning rate to be 0.0001 for both CE and SE loss functions. Plot the loss obtained on train data against the epoch for SE and CE loss functions. Distinguish using different colors. Explain your observations. [**4 marks**]

   (c) Fix the loss function to be SE. Run the stochastic gradient descent algorithm with 200 epochs and mini-batch size 1 for different learning rates $\{0.1, 0.01, 0.001, 10^{-5}, 10^{-6}\}$. For each learning rate, plot the training loss against epoch. Explain your observations.[**4 marks**]

   (d) Fix the loss function to be CE. Run the stochastic gradient descent algorithm with 200 epochs and mini-batch size 1 for different learning rates $\{0.1, 0.01, 0.001, 10^{-5}, 10^{-6}\}$. For each learning rate, plot the training loss against epoch. Explain your observations.[**4 marks**]

   (e) Note down the learning rates for SE and CE loss functions which achieve the best training loss in each case. Now for the best learning rates noted, perform the stochastic gradient descent algorithm with 200 epochs with mini-batch size from the set $\{50, 100, 200, 300, 500\}$. Plot the train loss against epoch for each mini-batch size. Explain your observations. [**4 marks**]

4. **[Use only Python]** Take the feed-forward network code file, along with the neural network architecture you used in answering Question 3. Consider the digits data set. Shuffle and split the data set into two sets S_1 and S_2 such that S_1 contains 80% of the data and S_2 contains 20% of the data.

   (a) Write the required code which can create the S_1 and S_2 partitions of digits data set. [**2 marks**]

   (b) Implement python functions to compute the error (or loss) for a partition of the data set for SE and CE loss functions. [**2 marks**]

   (c) Fix the loss function to be SE. Choose the learning rates from the set $\{0.1, 0.01, 0.001, 10^{-4}, 10^{-5}, 10^{-6}\}$. For each learning rate, run the stochastic gradient descent algorithm on S_1, with 200 epochs and a mini-batch size of 50. For every 5 epochs, record the error achieved on the sets S_1 and S_2. Now plot these errors for every 5 epochs for each learning rate. Can you come up with a suitable selection procedure for the best learning rate using the experiments conducted? Explain your selection procedure and justify. [**3 marks**]

   (d) Fix the loss function to be CE. Choose the learning rates from the set $\{0.1, 0.01, 0.001, 10^{-4}, 10^{-5}, 10^{-6}\}$. For each learning rate, run the stochastic gradient descent algorithm on S_1, with 50 epochs and a mini-batch size of 50. For every 5 epochs, record the error achieved on the sets S_1 and S_2. Now plot these errors for every 5 epochs for each learning rate. Can you come up with a suitable selection procedure for the best learning rate using the experiments conducted? Explain your selection procedure and justify. [**3 marks**]

   (e) Compare the best learning rate achieved in the answers to Questions 4c and 4d with those obtained using the procedure illustrated in Question 3. Are they same? If not, can you provide appropriate reasons? [**2 marks**]