IE643 Deep Learning - Theory and Practice **July-Dec 2024**

Assignment 1: Due On 29^{th} September 2024 (11:59 PM IST)

1 Instructions

Answer all questions. Write your answers clearly. You can score a maximum of 65 marks in this assignment.

Make sure that your answers and plots are visible in the python notebook (.ipynb) file. Name the .ipynb file for Question 1 as "IE643_rollno_assignment1_Q1.ipynb" and similarly for other questions. Upload in moodle, the .ipynb files correspoding to the questions as a single zip file named as "IE643_rollno_assignment1.zip". There will be no extensions to the submission deadline.

The links for all related files used in Assignment 1 are provided in moodle.

2 Assignment Questions (Solutions to be submitted)

1. Answer the following.

   (a) Consider a data set $D = \{(x^i, y^i)\}_{i=1}^n$, where for each $i \in \{1, 2, \ldots, n\}$ we have $x^i \in \mathbb{R}^d$, $y^i \in \{+1, -1\}$. Recall that D is linearly separable if there exists a non-zero vector $w^* \in \mathbb{R}^d$, and some $b \in \mathbb{R}$ and a $\gamma > 0$ such that $y^i(\langle w^*, x^i \rangle - b) \geq \gamma$ for every $i \in \{1, 2, \ldots, n\}$. Using this and the discussions in class, formulate an optimization problem of the form $\min_z f(z)$ s.t. $Az \leq q$ which will help in checking whether a data set is linearly separable or not. Clearly discuss the structure, shape and content of your matrices A, q, and the decision variables z. Also describe a suitable choice of the objective function f. (**Note:** We do not aim to optimize anything in particular here, rather we just wish to find if a linear separator exists or not for the data set.) [**4 marks**]

   (b) Use an open source optimization solver available in Python to implement a solver for your optimization problem. (**Open source solver examples**: `https://www.cvxpy.org/`). Make sure that you install all relevant packages in your notebook and that your implementation can be readily checked during evaluation. [**4 marks**]

   (c) Consider the 3 data sets in moodle, provided in 3 files named `data1.csv`,`data2.csv`, `data3.csv`. In these data sets, the first **four** columns represent the features of the samples and the last column represents the class label. Use your implementation in part 1b to check if these three data sets are linearly separable or not. Make appropriate changes to the labels used in the data sets if needed. [**6 marks**]

   (d) Use the perceptron implementation code posted in moodle that you have completed as part of your homework. Now use this code to run perceptron algorithm on the three data sets provided in files `data1.csv`, `data2.csv`, `data3.csv` and check if each of these three data sets is linearly separable or not. Explain how you used the perceptron algorithm to check for linear separability. Justify the correctness of your approach. [**6 marks**]

   (e) Are your answers to the questions in parts 1c and 1d comparable? Explain with proper reasons. [**3 marks**]

2. **[Use only Python]** Use the code template for multi-layer perceptron (or feed-forward network) coded from scratch, posted in moodle. Answer the following:

   (a) 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**]

   (b) The ReLU activation function is given by $\text{ReLU}(z) = \max\{z, 0\}$. Implement the python functions to compute $\text{ReLU}(z)$ and its gradient. [**2 marks**]

   (c) Consider an appropriate neural network architecture where each hidden layer has only ReLU activation functions and the output layer has a linear activation function. Illustrate the exploding gradient and vanishing gradient problems in this network. Justify the architecture you used, indicate how you checked the exploding gradient and vanishing gradient problems, and explain your observations. [**3 marks**]

   (d) To handle vanishing gradients, one idea is to forward the output activations of every neuron in every hidden layer to every other neuron in **all** subsequent layers. Derive the forward pass and backpropagation expressions and implement this idea in your code. For the network architecture where you faced vanishing gradient problem, check if this idea prevents the vanishing gradient problem or not. Explain your observations. [**5 marks**]

3. **[Use only Python]** Use the code template for multi-layer perceptron (or feed-forward network) coded from scratch, posted in moodle. Consider the MultiLabelData data set posted in moodle. In this data set represented by $D = \{(x^i, y^i)\}_{i=1}^n$, where for every sample index i, $x^i \in \mathbb{R}^{294}$ denoting a 294 dimensional feature vector and y^i denotes a multi-label vector denoted by $y^i = (y_1^i, y_2^i, \ldots, y_6^i)$ where each $y_j^i \in \{0, 1\}$, 0/1 denoting the absence/presence of the label component y_j^i. Thus, in the file `MultiLabelData.csv`, the first 294 columns denote the feature/attribute values of x^i while the last 6 columns denote the presence or absence of the multi-label components $y_j^i, j \in \{1, 2, \ldots, 6\}$.

   (a) Write code to read the data into suitable `numpy` arrays for features and labels. [**1 mark**]

   (b) Write the required code to shuffle and split the data set into three sets S_1, S_2 and S_3 such that S_1 contains 70% of the data, S_2 contains 15% of the data and S_3 contains 15% of the data. [**2 marks**]

   (c) After the split is done, comment on the label distributions for each of the 6 labels in the splits S_1, S_2, S_3. [**2 marks**]

   (d) Design a suitable feed forward neural network with 3 or 4 hidden layers with appropriate activation functions, a suitable output layer with suitable activation functions and a corresponding loss function to perform training on the S_1 split of MultiLabelData data set. Justify the design choice of your neural network and loss function and implement the loss function in the code. [**3 marks**]

   (e) Illustrate how you will carry out backpropagation for the loss gradients in the layers. Include its implementation in code. [**3 marks**]

   (f) Use **mean average precision** (MAP) as performance metric to assess the performance of your neural network. Implement in code, the computation of MAP. (**Note:** Do not use any existing package to compute the mean average precision metric.) [**3 marks**]

   (g) For the chosen loss function, choose the learning rates from the set $\{0.1, 0.01, 0.001, 10^{-4}\}$ and mini-batch sizes from $\{8, 16, 32\}$. For each (learning rate, mini-batch size) pair, run the mini-batch stochastic gradient descent (SGD) algorithm on S_1, with 200 epochs (use more epochs if necessary). For every 5 epochs, record the loss and MAP metric achieved on the sets S_1 and S_2. Now plot the loss for every 5 epochs for each (learning rate, mini-batch size) pair on S_2 (use a single plot and different colors for different pairs). Plot the MAP metric for every 5 epochs for

each (learning rate, mini-batch size) pair on S_2 (use a single plot and different colors for different pairs). Can you come up with a suitable selection procedure for the best (learning rate, mini-batch size) pair using the experiments conducted? Explain your selection procedure and justify. [**6 marks**]

(h) Using the best (learning rate, mini-batch size) pair identified above, conduct training using mini-batch SGD on the set $S_1 \bigcup S_2$ with max epochs set to 500. For every 5 epochs, record the loss and MAP metric achieved on the sets $S_1 \bigcup S_2$ and S_3. Include a stopping condition such that you can stop the training when the MAP metric on the set $S_1 \bigcup S_2$ does not increase significantly for p epochs with a suitable choice for p. Plot the loss on $S_1 \bigcup S_2$ and S_3 in a single plot and comment on the observations. Plot the MAP metric on $S_1 \bigcup S_2$ and S_3 in a single plot and comment on the observations. [**5 marks**]

(i) Which of the 6 labels was/were easy to classify? Which of the 6 labels was/were the most difficult to classify? Explain your observations. Come up with an idea to improve the classification performance related to the most difficult label(s) without impacting the performance on other labels. Implement your idea and explain why it works. Compare and contrast the performances of your new idea with the previous implementation. [**5 marks**]
