Assignment 5 Introduction to Deep Learning – CAP 4613

Due: Sunday, April 3

Instructions [Failure to follow instructions will be penalized at least by 1 point]

- All the solutions must be based on the methods taught in class. Otherwise, it will not be graded. This is an important note to the students who do not study the lectures and learn from other online resources.

- The Colab link must be shared.

- Make sure that it is possible to copy the Colab link from your pdf.

- Make sure that your pdf is not cutting the text.

- Provide the full answers in your pdf file with all the outputs of your code.

- Only one file must be submitted electronically.

- Combine the handwritten and Python parts before submission.

- The paper-based problems must be with no coding/python etc.

- The paper-based problems/parts can be typed, but if handwritten, it must be clean and legible.

- Show your detailed work to each problem.

HINT: The in-class Python files are provided on Canvas.

Problem 1) [Python] Application of Keras to build, compile, and train a neural network to perform XOR operation:

a) Create an np array of shape 4x2 for the inputs and another 4x1 array for the labels of XOR.

input		desired
x_1	x_2	label
0	0	0
0	1	1
1	0	1
1	1	0

b) Plot the given data points with two different markers for each group.
c) Based on the plot from part (b), what is the minimum number of layers and nodes that is required to classify the training data points correctly? Explain.
d) Build the network that you proposed in part c using the Keras library.

e) Compile the network. Make sure to select a correct loss function for this classification problem. Use stochastic gradient descent learning (SGD, learning rate of 0.1). Explain your selection of the loss function.
f) Train the network for 200 epochs and a batch size of 1.
g) Use the trained weights and plot the final classifier lines in the plot of part (b).
h) Plot the training loss (i.e., the learning curve) for all the epochs.
i) Repeat steps (d) to (g) after adding 2 more nodes to the first layer and training for 400 epochs.
j) What behavior do you observe from the classifier lines after adding more nodes? Which number of nodes is more suitable in this problem? Explain.

Problem 2) [Python] Application of Keras to build, compile, and train a neural network as a three-class classifier for MNIST dataset (0 vs. 1 vs. 2):

a) Use *mnist* function in *keras.datasets* to load MNIST dataset and split it into training and testing sets. Then, randomly select 20% of the training images along with their corresponding labels to be the validation data.
b) Feature extraction: average the pixel values in the quadrants in each image to generate a feature vector of 4 values for each image.
c) Convert the label vectors for all the sets to binary class matrices using *to_categorical()* Keras function.
d) Build, compile, train, and then evaluate:
    i. Build a neural network with 1 layer that contains 10 nodes using the Keras library.
    ii. Compile the network. Make sure to select a correct loss function for this classification problem. Use stochastic gradient descent learning (SGD, learning rate of 0.0001). Explain your selection of the loss function.
    iii. Train the network for 50 epochs and a batch size of 16.
    iv. Plot the training loss (i.e., the learning curve) for all the epochs.
    v. Use the *evaluate()* Keras function to find the training and validation loss and accuracy.
e) Repeat step (d) for each of the following networks:

Model #	**Details**	**Training**		**Validation**	
		loss	accuracy	loss	accuracy
1	1 layer 10 nodes				
2	1 layer 50 nodes				
3	1 layer 100 nodes				

4	2 layers 100 nodes, 10 nodes				
5	2 layers 100 nodes, 50 nodes				

f) What behavior do you observe in the training loss and the validation loss when you increase the number layers and nodes in the previous table. Which model is more suitable in this problem? Explain.
g) Evaluate the selected model in part (e) on the testing set and report the testing loss and accuracy.

Problem 3) [Python] Application of Keras to build, compile, and train a neural network to classify songs from Spotify dataset.

The Spotify dataset is a publicly available dataset with information about songs that did and didn't make it in the weekly Hot-100 list issued by Billboard. The goal is to develop a model to predict if a song will make this list. The dataset contains a total of 6,398 tracks with 15 features extracted from the audio features of these tracks. The classes are 1 and 0 which describes whether that track as made it in the Hot-100 list or not respectively.

a) Import the data file (spotify_preprocessed.csv) to your code. The data is preprocessed and ready to use.
b) Shuffle the data then split it into training (90% of the data) and test set (10% of the data). Split the training set further into training and validation sets with 80% and 20% percentages respectively.
c) Build, compile, train, and then evaluate:
    i. Build a neural network with 2 hidden layers that contain 32 nodes each and an output layer that has 1 unit using the Keras library.
    ii. Compile the network. Select binary cross-entropy (`binary_crossentropy`) as the loss function. Use stochastic gradient descent learning (SGD, learning rate of 0.01).
    iii. Train the network for 50 epochs and a batch size of 16.
    iv. Plot the training loss and validation loss (i.e., the learning curve) for all the epochs.
    v. Use the `evaluate()` Keras function to find the training and validation loss and accuracy.
d) Try different design ideas with the model until you get the best training and validation performance. For example, changing the number of hidden layers and number of units in each, changing the loss function, the learning algorithm, the learning rate, number of epochs and the batch size. Repeat the scores in a table.
e) Repeat parts (c) and (d) and select the model with the best performance.
f) Evaluate the selected model on the test set and report the testing loss and accuracy.