University of Nebraska Lincoln

Practical Machine Learning
ENGR 491/891

Programming Assignment 4

Spring 2022

Multi-Layer Perceptron

Assignment Goals

The goal of this assignment is to solve multi-class classification problems using the Multi-Layer Perceptron (MLP) model.

- Part A: An extensive study of the MLP model
- Part B: Design an MLP for optimal performance

Assignment Instructions

Note: You must use Keras with TensorFlow 2.0 backend to create the MLP models. You are allowed to use Python libraries such as Pandas, NumPy, Matplotlib.

i. The code should be written in a Jupyter notebook. Use the following naming convention.
   <lastname1>_ <lastname2>_ assignment4.ipynb
ii. The Jupyter notebook should be submitted via Canvas.

The programming code will be graded on **both implementation, correctness and quality of the results**.

Score Distribution

ENGR 891: 100 points

Part A: Extensive Study of the MLP Model

You will create Multi-Layer Perceptron (MLP) neural network models based on the specification. The MLP model will be trained using the Backpropagation algorithm that is implemented using the mini-batch Stochastic Gradient Descent (SGD) optimization algorithm.

Mini-batch Size & Training:
The size of mini-batches should be set to 64. A network should be trained for 50 epochs with ***early stopping*** on.

Dataset:
You will use the MNIST (Modified National Institute of Standards and Technology) dataset, which is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents.

There are 70,000 images. Each image is grayscale 28 x 28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black). The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.

You may directly load this dataset using the Keras API:
https://keras.io/api/datasets/mnist/

The train set contains 60,000 images, from which you should ***randomly*** select 5000 images as the validation set.

Experiments:
You will perform multi-class classification on the MNIST dataset using the following 10 models. After loading the dataset, create a separate validation subset with 5000 samples. Then, scale the data.

Create the models based on the architecture, SGD learning rate and layer configurations given below.

Report:
- Report following results in the last column of the "Layer Configuration" table (given below): train accuracy, test accuracy & no. of epochs at which the training stopped. Train and test accuracies should be reported up to **3 decimal places**.
- Report the learning curves (accuracy & loss) for each model.

[50 pts]

Note: grading will be based on the quality of the obtained results. Only the model implementation will not earn full credit.

Architecture		SGD Learning Rate
Experiment 1 to 7	2 hidden layers Hidden layer 1: neurons=300 Hidden layer 2: neurons=100	0.1
Experiment 8	2 hidden layers Hidden layer 1: neurons=300 Hidden layer 2: neurons=100	0.5
Experiment 9	10 hidden layers each with 100 neurons	0.1
Experiment 10	20 hidden layers each with 100 neurons	0.1

For all experiments the final (classification) layer should have 10 neurons (i.e., equal to the number of classes). Also, the final layer activation function must be "softmax".

Layer Configuration				Result
	kernel_initializer (all layers)	activation (hidden layers)	Dropout (hidden layers)	
Experiment 1	zeros	sigmoid	No	Train accuracy = Test accuracy = Epochs =
Experiment 2	ones	sigmoid	No	Train accuracy = Test accuracy = Epochs =
Experiment 3	random_normal	sigmoid	No	Train accuracy = Test accuracy = Epochs =
Experiment 4	random_normal	tanh	No	Train accuracy = Test accuracy = Epochs =
Experiment 5	random_normal	relu	No	Train accuracy = Test accuracy = Epochs =
Experiment 6	random_normal	relu	Yes Hidden layer 1: rate=0.1 Hidden layer 2: rate=0.1	Train accuracy = Test accuracy = Epochs =
Experiment 7	random_normal	relu	Yes Hidden layer 1: rate=0.5 Hidden layer 2: rate=0.1	Train accuracy = Test accuracy = Epochs =
Experiment 8	random_normal	relu	Yes Hidden layer 1: rate=0.5 Hidden layer 2: rate=0.1 SGD learning rate=0.5	Train accuracy = Test accuracy = Epochs =

Experiment 9	random_normal	relu	No	Train accuracy = Test accuracy = Epochs =
Experiment 10	random_normal	relu	No	Train accuracy = Test accuracy = Epochs =

Answer the following questions.

- Q-1) Among experiments 1 to 5, which experiment performed the best (based on test accuracy & epochs)? Explain why.

- Q-2) Among experiments 1 to 5, which experiment performed the worst (based on test accuracy)? Explain why.

- Q-3) Compare experiment 6 with experiment 7 and determine which model experiences less overfitting. Explain why. To answer this question, use the train accuracy, test accuracy, and learning curves of these two experiments. In your answer show these measures (statistics and learning curves).

- Q-4) Compare experiment 7 with experiment 8. Explain the change in the number of epochs in experiment 8. If it has increased, explain why. If it has decreased, explain why.

- Q-5) Compare experiment 8 with experiment 9. Show the accuracy learning curves of these two models. Explain the difference. Which model performed poorly? Explain why.

- Q-6) Compare experiment 9 with experiment 10. Which model performed poorly? Explain why.

[**18 pts**]

Part B: Design an MLP for Optimal Performance

You will design an MLP for the optimal performance on the following dataset.

Dataset:
The CIFAR-10 dataset (Canadian Institute For Advanced Research) contains 60,000 32 x 32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. You may directly load this dataset using the Keras API:
https://keras.io/api/datasets/cifar10/

The train set contains 50,000 images, and the test set contains 10,000 images.

Pre-processing: **[5 pts]**

You will need to perform some pre-processing steps. First step is to reshape the data. The dimension of the training set is 50000 x 32 x 32 x 3 and test set is 10000 x 32 x 32 x 3. Before you use this data for the MLP model, you need to flatten each sample (i.e., 32 x 32 x 3 = 3072) such the dimension of training and test set becomes:

- 50000 x 3072
- 10000 x 3072

Use the NumPy "reshape" function for this purpose.
https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

Example: X_train = X_train.reshape((X_train.shape[0], 3072))

The train and test labels will be loaded as a 1D column vector. You need to convert the 1D vector into a 1D array (because the sklearn K-NN model requires the label data to be a 1D array).

You may convert the label data into a 1D array by using the NumPy "ravel" function:
https://numpy.org/doc/stable/reference/generated/numpy.ravel.html

Example: y_train = y_train.ravel()

Finally, scale the data using the min-max scaling technique, i.e., by dividing the training & test data matrix by 255.0.

Example: X_train = X_train/255.0

Experiments:

- Experiment 11) Design an MLP classifier and perform multi-class classification. You must have an optimal design by finding optimal values for the following attributes: hidden layers, neurons in each hidden layer, initializer for the weights, activation function for the hidden layers, regularization technique (weight decay, dropout, early stopping), size of mini batch, learning rate. You may optimize additional attributes.

- Report the following: a 2-column table that shows the above attributes in the 1st column and their values in the 2nd column, train accuracy, test accuracy, test confusion matrix, the test classification report, learning curves (accuracy and loss).

Note: Experiment 11 will be graded best on the quality of the test performance. No help will be provided on the optimal design. You are free to try anything you want. I won't tell you whether your results are optimal or not.

[25 pts]

Answer the following question.

- Q-7) Explain the quality of your test performance.

[**2 pts**]

Deliverables

You will submit two products.

- A single Jupyter notebook containing all experiments. For each experiment show the required performance measures. If it's convenient, you may split your experiments in multiple notebooks and submit. Your code must be clearly annotated. Also add header descriptions for each block.
- A PDF copy of the written report with the cover page (get it from Canvas). The report must include the following items.

a) Part A: Using the table (given in Part A) report training accuracy, test accuracy, and epochs at which the training stopped.
b) Part A: learning curves for experiments 1 to 10.
c) Part A: Answers to Q-1 to Q-6.
d) Part B: a 2-column table that shows the model attributes in the 1st column and their values in the 2nd column, train accuracy, test accuracy, test confusion matrix, the test classification report, learning curves (accuracy and loss).
e) Part B: Answer to Q-7.