ECE 657 Spring 2020

Assignment 3

Guidelines:

- Upload your assignment as **ONE** compressed folder in **.zip** format named "**FirstName_YOURID.zip".** Please do not use any other format for compressing your files.
- The compressed folder should contain the codes for all problems, your report as a single PDF file, and two directories named 'data' and 'models'. Please refer to the uploaded folder and keep the same format.
- Allowed libraries besides Python3 standard libraries:
    - Tensorflow
    - Keras
    - PyTorch
    - Scikit-learn
    - pickle
    - numpy
    - pandas
    - matplotlib
    - random
    - glob
    - nltk
- You are free to create an extra file called 'utils.py' which contains any codes you plan to use in several parts of your code instead of copying it in each file. Just make sure you submit that as well!

Important Notes:

1. We expect your files to run smoothly without any errors, little to no effort will be made to correct errors.
2. Start early as some parts of the assignment may take a while to train. You can use Google Colab if your computational resources are low.
3. Make sure you use *relative paths* in your implementations. If you used Google Colab, fix the directories to be local directories before you submit.
4. For problems 2 and 3, your accuracy/loss will be compared to other students and part of the grade is assigned to how you compare to the rest of the class.

Problem 1: CNN for Image Classification

Using your preferred machine learning library, train a small convolutional network (CNN) to classify images from the *CIFAR10* dataset. Note that most libraries have utility functions to download and load this dataset (TensorFlow, PyTorch, keras).

Using the API for loading the dataset will readily divide it into training and testing sets. Randomly sample 20% of the training set and use that as your new training set for the purposes of this problem. Use the test set for validation. Implement the following in ***one*** *.ipynb* with all the output shown (already run).

1- **MLP:** Build a multi-layer perceptron with the following layers:

- Fully connected layer with 512 units and a sigmoid activation function
- Fully connected layer with 512 units and a sigmoid activation function
- Output layer with the suitable activation function and number of neurons for the classification task

2- **CNN1:** Build a Convolutional neural network with the following architecture:

- 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
- 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
- Fully connected (Dense) layer with 512 units and a sigmoid activation function
- Fully connected layer with 512 units and a sigmoid activation function
- Output layer with the suitable activation function and number of neurons for the classification task

3- **CNN2:** Build a Convolutional Neural network with the following architecture:

- 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
- 2x2 Max pooling layer
- 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
- 2x2 Max pooling layer
- Fully connected layer with 512 units and a sigmoid activation function
- Dropout layer with 0.2 dropout rate
- Fully connected layer with 512 units and a sigmoid activation function
- Dropout layer with 0.2 dropout rate
- Output layer with the suitable activation function and number of neurons for the classification task

Use a batch size of 32, utilize Adam as the optimizer and choose an appropriate loss function while monitoring the accuracy in both networks. Train each network for 5 epochs.

What you will submit

a) Well commented code and a description of what each part does.

b) Report:

- Any preprocessing steps you made
- Description of the output layer used and the loss function (use 1 setting for all 3 networks) and why you made these choices.
- Change the number of layers and the number of neurons per layer in the MLP, plot/tabulate the training and validation accuracies and comment on the results.
- Train and test accuracy for all three networks and comment (what happens and why) on the performance of the MLP vs CNNs.
- Plot the training and validation curves for the two CNNs and comment on the output. How does the training time compare for each of the CNNs? How does the different architectures influence these results? What do you expect the accuracies to be if the networks were trained for more epochs?
- Recommendations to improve the network. What changes would you make to the architecture and why?

c) Do not upload the dataset but make sure your code is able to download it if we need to run your code.

Problem 2: Recurrent Neural Networks for Regression

You are provided with a dataset for stock price prediction for 5 years with one sample per day (q2_dataset.py). Create a Recurrent Neural Network using the machine learning platform of your choice (PyTorch, Tensorflow, or Keras) to predict the next day opening price using the past 3 days *Open*, *High*, and *Low* prices and *volume*. Therefore, each sample will have (4*3 =) 12 features.

Follow the following steps and check the comments in the provided files and follow them:

1. In your train_RNN.py file: Before any preprocessing, create the dataset by using the latest 3 days as the features and the next day's opening price as the target. ***Randomize*** the created data and split it into 70% training and 30% testing and save it to '**train_data_RNN.csv**' and '**test_data_RNN.csv**' in the **data** directory respectively. Keep this code in your file but *comment it out*!
2. Populate the file **train_RNN.py** so that it reads your train_data_RNN.csv, preprocesses the data, and trains your RNN network. After training, the file should save your model with the name **'YOUR_ID_RNN_model'** in the **models** directory. You can use any extension you want as you will load your own model. Note that we will check that the test data was not used at ANY point in this file after saving it.
3. Populate the file **test_RNN.py** so that it reads your test_data_RNN.csv and runs the prediction model. Print the loss on your test data and show a plot of the true and predicted values (use appropriate labels for the axes and legend).
4. Make sure both your train_RNN.py and test_RNN.py files run by calling them from the command line without any extra inputs/arguments.

What to add in your report:

- Explanation of how you created your dataset.
- Any preprocessing steps you followed
- All design steps you went through in finding the best network in your report and how you chose your final design.
- Architecture of your final network, number of epochs, batch size (if needed), loss function, training algorithm, etc.
- Output of the training loop with comments on your output
- Output from testing, including the final plot and your comment on it
- What would happen if you used more days for features (feel free to actually try it – but do not upload the datasets).

Problem 3: Sentiment Analysis

The IMDB large movie review dataset has many positive and negative reviews on movies. Download the dataset here (http://ai.stanford.edu/~amaas/data/sentiment/). Check the README file as it provides a description of the dataset. Use the provided training and testing data for your network. You would need to go through data preprocessing to prepare the data for NLP. Then, create a network to classify the review as positive or negative.

All choices are up to you and many resources exist online to help you make good design choices. While you can check online resources, you should not copy other's code! Using a pretrained network or training a network that already exists is not allowed. Any act of plagiarism will be escalated. Make sure you understand and support any preprocessing steps and design decisions in your report.

Provide the following:

- **train_NLP.py** and **test_NLP.py** files (fill in the ones provided) that train and test the network with the datasets provided and that run directly from the command line to train and test the network.
  - Assume the data resides in the **data** directory in a folder called **aclImdb**, exactly like that you downloaded (please do not upload it!), refer to Figure 1Figure 1 and Figure 2.
- Save the model as **'YOUR_ID_NLP_model'** in your **models** directory**.** Again, save it in any format you want since you will be loading it yourself in your test file.
- A report explaining all the preprocessing steps you made, the design choices for your network, the training and testing accuracies, and any comments you have on the output.

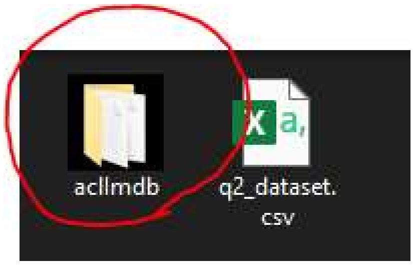

Figure 1: aclImdb folder in data directory

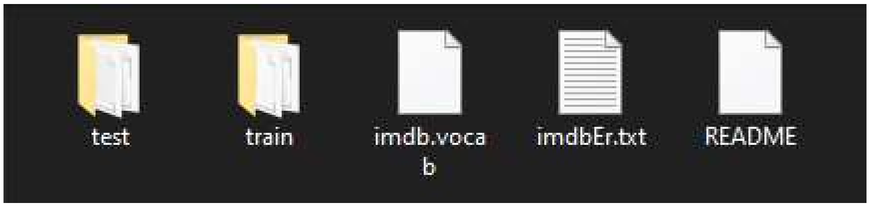

Figure 2: Contents of aclImdb (that we have)