CS57700 Homework 1

Due date: 23 February 2023 11:59 pm

This homework will involve conceptual exercises and coding assignments. Instructions below detail how to turn in the conceptual part on Gradescope and codes via turnin.

1 Coding Assignment

In this section you will implement logistic regression and multi-layer neural network architecture for a multi-class classification problem. You must implement both of the classifiers from scratch in *Python*3, no built in functions from libraries like *sklearn* can be used. However, you can use libraries like *pandas*, *numpy* etc for pre-processing and implementation purpose

1.1 Problem

Emotion detection from text is an interesting and important problem in Natural Language Processing. Humans have a variety of emotions and it is difficult to collect enough records for each emotion. In this assignment you will have a labeled data for emotion detection and the objective is to build model to detect emotion.

1.2 Dataset Description

You are provided with 1200 Tweets where each Tweet is annotated with one of the 6 emotions *joy, love, sadness, anger, fear, surprise*. The training set can be found in **train.csv**. You are given the test set of 800 Tweets in the file **test.csv**. Note that, the gold labels are not provided for the test set. The description of each column of the csv files are given below.

1. **id** : Unique identifier for each tweet.
2. **text** : Text of the tweet.
3. **emotions** : Emotion label of the tweet (One of the 6 emotions). This field is kept empty in test.csv.

You have to build classifiers using the training set and predict the labels of the test set. **You must not use the test set while training the models**. You are provided with a file **main.py** where you will find the starter codes. Feel free to define any additional functions needed in this file.

1.3 Task 1: Logistic Regression (20 Points)

Build a Logistic Regression classifier to identify the emotions present in the Tweets. Fill in the function **LR()** in **main.py**. This function should learn a Logistic Regression classifier using the training set and predict the emotions in the Tweets contained in **test.csv**. It will output a csv file named **test_lr.csv** with the predicted emotions in the tweets in **test.csv**. **test_lr.csv** must have all the columns in the given order - id, text, emotions. Make sure you preserve the mapping id : text : emotions, as tweet ids are the identifiers which will be used to match your predicted labels with gold labels. (Hint: Follow the starter code provided). **Make sure to train your model using cross validation.**

1.4 Task 2: Multi-layer Neural Network (20 Points)

Build a Multi-layer Neural Network classifier to identify the emotions present in the Tweets. Fill in the function **NN()** in **main.py**. This function should learn a Multi-layer Neural Network classifier using the training set and predict the emotions in the Tweets contained in **test.csv**. It will output a csv file named **test_nn.csv** with the predicted emotions in the tweets in **test.csv**. **test_nn.csv** must have all the columns in the given order - id, text, emotions. Make sure you preserve the mapping id : text : emotions, as tweet ids are the identifiers which will be used to match your predicted labels with gold labels. (Hint: Follow the starter code provided). **Make sure to train your model using cross validation.**

2 Conceptual Questions (20 Points)

Now answer the following questions based on your implementation of Logistic Regression and Neural Network classifiers.
1. Write down the logistic function and loss function you used in Logistic Regression and derive the gradient. (4 Points)
2. Describe the architecture of the neural network you implemented using a computation graph **(Must be drawn using any software. Handwritten/drawn ones will be discarded from grading)**. Your answer should include the activation functions used in each layer (2 Points), the number of layers and number of neurons in each layer (2 Points). State the loss function used in your neural network implementation (2 Points). Derive the partial derivative associated with each node in the computation graph. (4 Points)
3. State one hyper-parameter you tuned in case of Logistic Regression and one in case of Neural Network. How did tuning these hyper-parameters change

the result? Explain with learning curves. You may have tuned more than one hyper-parameter for each model. In this section explain only one from each model. (6 Points)

3 Submission Instructions

3.1 Conceptual Part

Upload your answers to the conceptual questions as a **typed pdf** in gradescope.

- For your pdf file, use the naming convention **username_hw1.pdf**. For example, your TA with username *islam32* would name her pdf file for HW1 as **islam32_hw1.pdf**.
- To make grading easier, please start a new page in your pdf file for each question. Hint: use a *newpage* command in LaTeX after every question ends. For example, for HW1, use a *newpage* command after each of the questions 1-3.
- After uploading to gradescope, mark each page to identify which question is answered on the page. (Gradescope will facilitate this.)

3.2 Coding Part

You need to submit your codes via Turnin. Log into data.cs.purdue.edu (physically go to the lab or use ssh remotely) and follow these steps:

- Place only the files main.py, test_lr.csv, test_nn.csv in a folder named **username_hw1**. For example, your TA with username *islam32* would name her folder for HW1 as **islam32_hw1**. **This naming convention is important. If the folder is not named correctly, there's no way to identify whose submission is that. Hence, may result in no grading.**
- Change directory to outside of username_hw1 folder (run cd .. from inside username_hw1 folder)
- Execute the following command to turnin your code:

  turnin -c cs577 -p hw1Spring2023 username_hw1
- To overwrite an old submission, simply execute this command again.
- To verify the contents of your submission, execute this command:

  turnin -v -c cs577 -p hw1Spring2023

  Do not forget the **-v** option, else your submission will be overwritten with an empty submission.

3.3 Access to Turnin

If you don't have the access to data.cs.purdue.edu, please send an email to sciencehelp@purdue.edu to get access. Please do it at your earlier convenience because it's mandatory for your HW1 coding part submission.

3.4 Text Preprocessing + Input Features

- You are allowed to use libraries like NLTK for text preprocessing, but still do not use functions from ML libraries like Sklearn.
- Simple text preprocessing baseline: bag of words features. Create the vocabulary from the training data and use it on the test set to avoid cheating.
- More advanced text preprocessing: Word embeddings.
- Any other options are allowed, be creative!

3.5 Word Embedding Text Preprocessing

- If you are using word embeddings as your text feature representation, computing these representations will take a while and should not be done in the submitted code. There is a time limit of 10 minutes.
- Instead, you must submit the feature representation of each Tweet with your submission, and load it in your code. Ex: take pre-trained word embeddings and average them to create a feature representation for each tweet. Save the feature representation of each tweet (train and test) in a file and read from there when running the code. Reminder: test_lr.csv and test_nn.csv need to have the same format as the original.

3.6 Saving Model Parameters + Code Submission

- The models must be trained using cross-validation, which will allow you to tune your model and determine your best set of hyper-parameters.
- After this tuning is complete, save them in your code (set the values).
- When submitting **do not** perform cross-validation (**comment the code out**).
- The submitted code should learn a model from **scratch** (do not use loaded weights) using the best parameters you determined using cross-validation experimentation.

3.7 Grading Details

- Time Limit: Code must finish running within 10 minutes.
- **main.py** file should read the train set, train on it, label the test data, and output the respective files for each model (as described in the starter code - do not change the output prediction file format!)
- Grading Evaluation: Test Set Accuracy. We cannot provide details on what the test set accuracy should be at this time.