Programming Assignment 2

Building Feed-Forward Neural Networks (FFNNs) from Scratch and FFNNs for Sentiment Classification

In this assignment, you will build two Feed Forward Neural Networks (FFNN) from scratch and PyTorch or TensorFlow Neural Network (NN) modules, respectively. Then you will use two created FFNNs for sentiment classification. We are defining sentiment classification as two classes: positive and negative. You will build the models using training data and evaluate with test data.

1. Data Processing

Our data set consists of airline reviews. The zip directory for the data contains training and test datasets, where each file contains one airline review tweet. Each of the training data and test data contains **4182** reviews.

a. Create your Vocabulary: Read the complete training data word by word and create the vocabulary V for the corpus. You must not include the test set in this process. Remove any markup tags, e.g., HTML tags, from the data. Lower case capitalized words (i.e., starts with a capital letter) but not all capital words (e.g., USA). Remove all stopwords. You can use appropriate tools in nltk to stem. Stem at white space and also at each punctuation. In other words, “child’s” consists of two tokens “child and ‘s”, “home.” consists of two tokens “home” and “.”. Consider emoticons in this process. You can use an emoticon tokenizer, if you so choose. If yes, specify which one.

b. Extract Features: Convert documents to vectors using **tf-idf** representation. Do not use the existing tf-idf calculation library.

2. Building FFNNs from Scratch or NN modules and FFNNs for sentiment classification

a. You will build two FFNN systems using the **Sigmod** and/or **ReLu** activation functions:
   - **FFNN1**: build this system from **scratch** using only **numpy**. Do not use any existing libraries, such as scikit-learn or tensorflow/Kereas.
   - **FFNN2:** build this system using **PyTorch.nn** or TensorFlow modules. Refer to the provided code examples for guidance.

   For each system, set 2 layers with hidden vector size 20. Use sigmoid as the activation function.

b. Training: Initialize the weights with random numbers. Use Mean Squared Error (MSE) as your loss function You can start training with a learning rate of 0.0001. If you would like to tune any parameters, you must do so using cross-validation on the training data only. Once you have finalized your system, you are ready to evaluate the test data.

Note that if you want to experiment with different stemmers or other aspects of the input features, you must do so on the training set, either through cross-validation or by setting a development set from the get-go. You must **not** do such preliminary evaluations on test data. Once you have finalized your system, you are ready to evaluate the test data. You can ignore any words that appear in the test set but not the training set.

c. Evaluation: Compute the most likely class for each review in the test set using each of the combinations of stemming + **tf-idf**, no-stemming + **tf-idf**. Compute and report accuracy with confusion matrix on a .txt or .log file.

d. Comparison: compare the performance of the two systems. Provide explanations at the end of the code file.

e. **Bonus points**:

   i. Come up with your one or more activation functions and then compare them with Sigmod and/or ReLu.
   ii. Based on Programming Assignment 1, use **all** training and test data (**4182*2** in total) to generate embeddings. Replace tf-idf vectors with generated **embeddings** to compute and report accuracy with confusion matrix on a.txt or .log file.

Documentation and Deliverables:

a. Documentation: Use the same documentation format from Assignment 1. Start all your files with a description of your code. Write a short description of each function on top of it.
b. Deliverables: Submit a zip file named **AIT726_PA#_StudentNameInitials.zip** (e.g., AIT726_PA1_AM.zip).
c. Zip file should include: Your code(s), and .log file or .txt file that contains your output and another file that contains accuracy. You can choose whatever is convenient for you. Log can be created using logging library. Txt file can be created using a simple io library.

NOTE: If you use **Jupyter** Notebook, you can include comments and running outputs directly within the notebook files. Once you're done, save the notebooks as HTML files and compress all notebook files, HTML files, any relevant intermediate datasets, and other related files into a single zip file. Please submit only **one zip file**. If your datasets are too large to upload to Blackboard, please reach out to your instructor or GTA for assistance.