Nathan Nguyen, Steve Aigbe and Christoph F. Eick

COSC 4368: Fundamentals of Artificial Intelligence Spring 2022

ProblemSet2 (Individual Tasks)

First Draft

Submission deadline: Task 3: Thursday, March 31; Task 4: Fr., April 8; all end of the day

Last updated: March 5, 5:30a

Allocated points to ProblemSet2: Task3: 29 points(?); Task 4: 39 points (?)

Allocated points to ProblemSet2 are tentative and subject to change.

**3. Using SVM and NN Tools Nathan**

The goal of this task is to apply different classification approaches to a challenging dataset. Compare the results and enhance the accuracy of the learnt models via selecting better parameters/preprocessing/kernels/background knowledge to summarize your findings in a report. For this problem, we will use the [UCI Machine Learning Repository](http://archive.ics.uci.edu/ml/index.php) and particularly the [Higher Education Students Performance Evaluation](http://archive.ics.uci.edu/ml/datasets/Higher+Education+Students+Performance+Evaluation+Dataset) dataset. This is a multiclass classification problem, which is to create classification models that distinguish the relation between a student’s life attributes and their grades. Please carefully read the dataset description on the UCI website as it gives information about the different attributes.

As far as classification algorithms are concerned, we will use:

1. Neural Networks (Multi-Layer Perceptron – MLP)

2. Support Vector Machines.

You will use 2 “variations” of each approach:

* For the SVM, you should use 2 different kernels (any kernel is fine, you can use the linear kernel as one kernel)
* For the MLP, you should use two of the following activation functions: 1. Logistic/sigmoid 2. Tanh and 3. Relu

Accuracy of the four classification algorithms for training and testing should be measured using 10-fold cross validation. Moreover, compute the average MAE (mean absolute error) for the learnt models training and testing.

An extra credit of up to 15% of this task will be given for exploring approaches which minimize the MAE for the dependent variable instead of the classification accuracy. Combination of parameters.

**In your report:**

1. Report the Accuracy of the four classification algorithms and the average Mean Absolute Error[[1]](#footnote-1) (MAE) for training and testing sets.
2. After comparing the experimental results, write a paragraph which summarizes the experimental results and also tries to explain/speculate why, in your opinion one classification algorithm outperformed the other.
3. If you conducted extra credit activities concerning minimizing the MAE of models, write 2-3 paragraphs which summarize these activities
4. Finally, at the end of your report write a short paragraph which summarize the most important findings of this task.

**Deliverables:**

Please submit both the report and the source code file.

**Suggestions:**

You can use built-in functions in python and R.

For python, it’s preferable to use Scikit-Learn for both SVM and MLP (see the scikit-learn documentation)

For R, we suggest the ‘mlp’ and ‘svm’ functions.

**4. Sentiment Classification on a Movie Review Dataset Steve**

A sentiment classification problem for movie reviews usually consists of taking a piece of text and predicting if the author likes or dislikes what she read: the input *X* of this task is a text and the output *Y* is the sentiment we want to predict, such as the rating of a movie review.

The dataset we are using contains movie reviews along with their assigned binary sentiment polarity labels.

* The dataset contains 50,000 reviews split evenly into 25k train and 25k test sets.
* The overall distribution of labels is balanced
* The train and test dataset (Numpy file) can be downloaded [here](https://drive.google.com/drive/folders/14Twk0nUI1DA4Ace53EHsRzmrb_te77eH?usp=sharing)

### Learning Goals

The learning goals for this assignment are:

* An introduction to **BERT**, which is a powerful architecture that can be used to solve many NLP tasks. BERT is a large, pre-trained neural language model based on the Transformer architecture that can adapted to many classification tasks.
* Learn how to solve sentiment **classification tasks by “fine-tuning” BERT** to the task. The process of fine-tuning adapts a general pre-trained model like BERT to a specific task that you’re interested in.
* Learn how to preprocess texts for deep learning tasks
* We’re going to fine-tune BERT to do the classification task of predicting the sentiment of movie reviews

## **Steps**

Following the below steps will help achieve the implementation:

1. Load and explore the dataset to see what it looks like
2. Load the required pretrained BERT model that can be downloaded from the link given below
3. Convert dataset examples into BERT input features (You will need to (1) tokenize your input sentences (2) Have a vocabulary/dictionary and convert each token to a vector/tensor)
4. Choose your hyperparameters such as learning rates, epoch etc
5. Train the pre-trained BERT model with the preprocessed train dataset. This is known as fine-tuning
6. Evaluate the fine-tuned model on the test dataset (the test dataset also needs to be converted into BERT feature vectors/tensors)
7. Report the accuracies (train & test), F1-score, precision, and recall values for training and test sets.

## **Result Report**

## You will perform several experiments with different BERT models fine-tuned for the sentiment classification task and report results in a table as shown below. The different BERT models are available in the google BERT storage [page](https://storage.googleapis.com/bert_models/2020_02_20/all_bert_models.zip).

For each model, specify the hyper-parameters you chose for training

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BERT Model**  **Learning\_rate: 3e-5**  **Epoch=1** | **Train Accuracy** | **Test Accuracy** | **Precision** | **Recall** | **F1-Score** |
| BERT-Tiny (L-2\_H-128\_A-2) | 0.8138 | 0.80496 | 0.806486 | 0.80496 | 0.804717 |
| BERT-Mini (L-4\_H-256\_A-4) | 0.8382 | 0.838122 | 0.838122 | 0.83812 | 0.838120 |
| BERT-Small (L-4\_H-512\_A-4) | 0.8524 | 0.8529 | 0.854810 | 0.85288 | 0.852680 |
| BERT-Medium (L-8\_H-512\_A-8) | 0.8602 | 0.8549 | 0.860484 | 0.85492 | 0.854358 |

## **Evaluations**

Include the followings in your report in addition to the table:

* Explain how you did the dataset preprocessing
* Report the different hyperparameters and their values used for each BERT model configuration and explain why you chose them. The hyperparameters include learning rates, batch sizes, number of epoch, warmup steps, optimizer etc.
* Report which BERT model configuration gave the best values for the accuracies (train & test), F1-score, precision, recall. Explain why you think this BERT model configuration gave the best results

## **Deliverables**

Here are the deliverables that you will need to submit:

* Your source code (notebook preferably)
* PDF Report

## **Improving your classifiers (Extra credit up to 20%)**

Extra credits will be awarded to students who report enhanced performances than those reported in the requested table. Here are some ideas for improvements:

1. **Increase the number of transformer layers for BERT**. The base version of BERT has 12 layers, which will give way better performance, at the cost of longer training/evaluation time.
2. Use **better negative sampling strategies[[2]](#endnote-1)** to enhance classification accuracy
3. **Hyperparameter Tuning**: (e.g., different learning rates, batch sizes, number of epoch). You are welcomed to try different values for each hyperparameter and report your findings!

Remark: There will be a lab taught by Steve on Monday, March 28, 2:30-3:50 to help you with Task4.

1. <https://machinelearning101.readthedocs.io/en/latest/_pages/05_model_metrics.html> [↑](#footnote-ref-1)
2. Negative sampling is a way to subsample frequently occurring words to reduce the number of training examples. Take a look at this [paper](https://arxiv.org/pdf/1310.4546.pdf) [↑](#endnote-ref-1)