CIS 472/572, Winter 2021
Homework 4 (Programming): Deep Neural Networks

1 Overview

Sentiment Analysis (SA) is a task in Natural Language Processing (NLP). SA involves determining whether an input text is positive, negative, or neutral. Sentiment analysis is often used to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

In this project, you are going to train deep neural networks and evaluate it on a sentiment analysis dataset, namely **IMDB**. This dataset contains 50,000 data points, each data point consists of an IMDB comment and a label. The comment is a paragraph of several sentences derived from the IMDB website. A label is either **positive** or **negative**.

In this assignment, you are going to practice in a real deep learning project. Your activities include:

- Tuning hyper-parameters to reach the highest development accuracy.
- Visualize the training and finetuning processes.

2 Data and starter code

You are provided:

- GLoVe embeddings (50D, 100D, 200D, 300D).
- A training data file of 25,000 samples.
- A development data file of 25,000 samples.
- A starter Jupiter notebook with a complete algorithm.

All these files are available at the following link:
`http://nlp.uoregon.edu/download/cis472/hw4/`

The provided code is written using PyTorch library. The document of PyTorch is available here: `https://pytorch.org/docs/stable/index.html`. The provided model is a simplified version of the convolutional neural network for sentence classification by Yoon Kim. The paper is accessible at `https://arxiv.org/pdf/1408.5882.pdf`. Although the training algorithm is already provided, customizing for your own catching/saving logs and visualization is needed.

3 Requirements

3.1 Finetuning

Hyperparameter tuning is a very important skill in machine learning. The starter code provides a working implementation with a set of initial hyperparameters. However, this set does not give the

best development accuracy. You need to adjust these values and train the model to achieve your highest accuracy. You should finetune at least 3 hyper-parameters. Here are some hyper-parameters that you can choose.

- Number of training epochs
- Batch size
- Optimizer type
- Learning rate
- Maximum sentence length
- Dimension of hidden layers
- Activation function
- Regularization

Once you are done with tuning, for each hyper-parameter that you have tuned, you should present the values of the hyper-parameter that you have tried and the corresponding performance of the model on the development data (i.e., using a table or a graph). Please report the performance in accuracy percentage (2 digits precision).

Tip: Calibrating these parameters is not an easy work. Blindly adjusting might not lead to a good performance. You should have a strategy of which values you are going to try. Two most common strategies are grid search and hill-climbing. For simplicity, we suggest fine-tuning each hyper-parameter at a time. Once you finish the tuning of one hyper-parameter, you can fix that hyper-parameter with its best value (based on development data) and continue with the tuning of another hyper-parameter.

Note: Batch size and sentence length is relative to the amount of consumed GPU memory. You might run into an "Out of memory error" if these hyper-parameters are too high.

Also, you can use any version of the GloVe embeddings (i.e., 50D, 100D, 200D, or 300D); however, 300D is often used in practice and might lead to best performance. In your report, please specify which version of GloVe you use in the experiments.

3.2 Training progress visualization

Visualization is a very handy tool to examine the training progress. In some cases, visualization can help identifying severe faults in your code. Visualization is often done for typical performance mesaures, including training loss, training accuracy, development loss, development accuracy over the time (i.e.,. the number of parameter updates or epochs).

From the training session with the best hyper-parameter setting you found, you should:

- Provide a graph/table to capture training and development loss over time (i.e., learning curves for loss function over the number of parameter updates or epochs).
- Provide a graph/table to capture training and development accuracy over time (i.e., learning curves for accuracy over the number of parameter updates or epochs).
- Provide any findings/conclusions you have learnt in the tuning and visualization process.

4 Turn in

A single pdf file with the following information (in order):

- Your name and your UO ID.
- A link to your Google Colab file. Make sure that the instructor and GE have access to your Google Colab by enabling document sharing.
- Credit: Full names of UO students who helped you (if any).
- Reference: Paper titles and URL(s) to the sources that you consult for this assignment (if any).
- Reports required in sections 3.1 and 3.2.

5 Rubric

Task	Requirement	Points
Finetuning	Development performance evaluation	30 points
	$0 < acc \leq 70\%$	4
	$70\% < acc \leq 75\%$	8
	$75\% < acc \leq 77\%$	12
	$77\% < acc \leq 79\%$	20
	$79\% < acc \leq 81\%$	25
	$81\% < acc < 100\%$	30
Visualization & Report	Attempted values and corresponding development performance for three different hyperparameters	30 points
	Learning curves for loss function on training and development data	20 points
	Learning curves for accuracy on training and development data	20 points

Any error (intentionally or not) that leads to a mixture of training-development data/predictions/logs will result in 0 points for the finetuning part and partial credit for other related parts.