Assignment 4: Sentiment Analysis using Feedforward Neural Networks

Purposes

We are going to use Feedforward Neural Networks (FNNs) to perform sentiment analysis on the IMDb movie reviews.

- Case study of sentiment analysis (textbook Page 247 - 260).
- Processing text using `sklearn.feature_extraction.text`.
- Building and training basic FNNs using PyTorch.
- Training an FNN using k-Fold Cross Validation.
- Dropout Regularization on FNNs.
- Ensemble Learning.

Implementation Tasks

1. (2 points) **Data Preparation.** You are required to download and transform the text reviews to *term frequency-inverse document frequency (tf-idf)* vectors. You may use the approach and the programs in the textbook to do so. Make sure that all text reviews are cleaned and correctly transformed. You may use 70% data as training data and the rest as test data.

2. (3 points) **Building an FNN.** Define an FNN classifier which takes a tf-idf vector, i.e., the vector representation of a review, and predicts the writer's attitude: 1 for positive and 0 for negative. This task should be solved together with the next one. You need to tune your model, i.e., changing the number of layers or neurons, to see if a good accuracy can be obtained by training.

3. (3 points) **Training a Baseline Model and Tuning the Hyperparameters.** You are required to tune the FNN's structure (the previous task) and try a few values for learning rate and weight decay (L2-regularization parameter), to obtain an accuracy as high as possible or $\geq 90\%$. Compare the accuracy and time cost to the logistic regression model in the textbook. Please feel free to add more layers and neurons to make the FNN as powerful as possible. If the training performance is bad, please use `torch.optim.Adam` which often has a better performance than `torch.optim.SGD` in deep learning.

4. (5 points) **Training using k-Fold Cross Validation.** We learned how to use k-fold cross validation in scikit-learn. But the PyTorch training functions does not directly support k-fold cross validation. In order to use this technique, you are required to implement a basic k-fold cross validation in training an FNN. You may first read the tutorial and then adapt the code for your program. Comparing the performance (time cost and accuracy) of the training with and without k-fold cross validation. You may tune the value of k for a good performance.

5. (7 points) **Training using Dropout Regularization.** The dropout regularization on a baseline FNN is to create a simplified version by randomly dropping out a subset of the edges. You may use the FNN in Task 2 and train it without regularization. It can be performed by the PyTorch function `dropout`. You may first read the detailed tutorial, to see how a dropout model can be created, and then complete the following two tasks:

    - (2 points) Create a single dropout model and compare its performance to the baseline model. You may tune the dropout probability for each layer.
    - (5 points) Create a set (≥ 5) of different dropout models, train them using bagging and compare the performance to the baseline model. You may tune the dropout probability for each layer.

Report

You are required to write a report for this assignment. It should include the following parts:

- For Task 2, a brief description of the FNN you used in your experiments.
- For Task 3, give the final parameters you used after tuning, and show the comparison (time cost and accuracy) to the logistic regression model in the textbook.
- For Task 4, show the time cost and accuracy comparison.
- For Task 5.1 and 5.2, show the time cost and accuracy comparisons.