Assignment 0010: Sentiment Analysis Using Deep Learning

Dr. Ankur Mali

03/08/2024

1 Introduction

Sentiment analysis is a crucial task in natural language processing (NLP) that involves determining the sentiment behind a piece of text. This document outlines a comprehensive approach using a deep multi-layer perceptron (MLP) that you must implement using TensorFlow. The starter code is provided, you need to modify/extend/adapt the script shared with you.

2 Preprocessing and Tokenization - 1 point

As you have learned from the first assignment, it's important to pre-process your dataset; thus, the first phase is focused on preprocessing input text.

2.1 Preprocessing

Preprocessing involves transforming raw text into a cleaner format. The steps include:

- Lowercasing: $T_{lower} = \text{lowercase}(T)$
- Removing Noise: $T_{clean} = \text{remove\_noise}(T_{lower})$
- Stopwords Removal: $T_{stop} = \text{remove\_stopwords}(T_{clean})$
- Stemming/Lemmatization: $T_{stem} = \text{stem\_or\_lemmatize}(T_{stop})$

Now, the second stage focuses on tokenizing your input; now, you can use character-level tokenization or word-level tokenization.

2.2 Tokenization

Tokenization converts text into tokens with sequence padding or truncation:

$$S = \text{tokenize}(T_{stem})$$
$$S_{fixed} = \text{pad\_or\_truncate}(S, n)$$

3 Creating the Embedding Layer and Deep MLP - 5 points

This is an important phase of your assignment, where you develop a deep learning model to perform classification. The first step is to construct an embedding layer, it's important to divide your dataset into Train/Validation/Test splits, where 80% can be used for train, 10% for validation, and the remaining 10% for test. Your embedding layer will only contain words from your train set.

3.1 Embedding Layer

The embedding layer maps tokens to vectors:

$$V \in \mathbb{R}^{n \times d}$$

where d = dimension of embedding layer and n = number of unique tokens in corpus.

3.2 Deep MLP Architecture

Next, the above layer will be passed through a series of hidden layers to transform features to learn the syntactic and temporal relationships from the input data. The MLP architecture is defined as follows:

- Input Layer: Accepts V.
- Hidden Layers: $y^{(l)} = \sigma^{(l)}(W^{(l)} z^{(l-1)} + b^{(l)})$ where $sigma^{(l)}$ is non-linear activation function (ReLU, TeLU, Sigmoid, Tanh, etc).
- Output Layer: $y^{(L)} = \delta(W^{(L)} z^{(L-1)} + b^{(L)})$, where δ is sigmoid activation function (Note: If number of classes >2, then use softmax as your activation function).

Where W and b are weights and biases for your model, L is the total number of layers.

Important: Now you will develop 2 different models, MLP-1. This model will have one embedding layer, 4 hidden layers (hidden size >= 64), and one output layer, trained for minimum >= 25 epochs. The second model is known as MLP-2; this model will also have one embedding layer, followed by 5 hidden layers (hidden size >= 64) and one output layer. Similarly, this model should be trained for >= 25 epochs. Remember number of hidden units, learning rate, optimizer, batch sizes are set of hyper-parameters that should be optimized.

Very Important: Random Forest and Random Classifier can get 78.6% and 77.5% accuracy respectively on test split, your MLP should achieve accuracy beyond this.

4 L1/L2 Regularization - 2 points

Another crucial part of this assignment is to avoid overfitting and underfitting, to do so you have to build an regularization and tune it ensure your model achieves do not exhibit overfitting. The standard cross entropy loss is calculated as follows:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

Regularization adds a penalty to the loss function (L). For instance L1 penalty is represented as follows:

$$L_{reg} = L + \lambda_1 \sum_{l} \|W^{(l)}\|_1$$

where λ_1 is coefficient controlling your regularization term, whereas L2 penalty is represented as follows

$$L_{reg} = L + \lambda_2 \sum_{l} \|W^{(l)}\|_2^2$$

where λ_2 is another coefficient that controls L2 regularization term.

Important: If difference between train and validation accuracy is greater than >= 4.5, you will lose your points

Report your findings, first you will train MLP-1 and see if it exhibits overfitting or not, if it does then you will add regularization. Similarly you will follow the similar procedure for MLP-2

5 Experimental Evaluation - 2 points

As discussed in the class neural networks are stochastic models, hence its important to provide statistical significance for any NN.

5.1 Setup

The experiment involves $k = 10$ trials. To do this, you will find the best model and only change the initialization seed for the model, all other hyperparameters will be same. This should be done for both models MLP1 and MLP2 respectively. Finally you will report (average accuracy $\pm$ standard deviation) across 10 trials for MLP1 and MLP2 respectively.

6 Things to Include in Report

We have provided the data and basic script that builds MLP class.

1. Explain tokenization strategy.

2. Explain hyper-parameter optimization steps – Explain difficulties you might have faced, what you have learned from optimization.
3. Explain how your loss landscape changes as regularization is introduced – do you observe any over-fitting prior regularization, if yes explain your observation.
4. Provide loss curve for train and validation loss (This should be reported for both model MLP-1 and MLP-2).
5. Provide confusion matrix for both model (MLP-1 and MLP-2).
6. Provide your detailed report with modified python script.