Homework 3: Pytorch and CNNs

Due February 9, 2022 (11:59 PM)

1 Introduction and Task

For this homework, you will be training a bag-of-words logistic regression and a convolutional neural network (CNN) classification model to predict the positive/negative sentiment of movie reviews. The model you will implement for this homework is similar to the CNN described in Zhang and Wallace.[1] This model is trained on the dataset you used in HW2.

The colab notebook for this homework is located here: `https://github.com/dbamman/nlp22/tree/main/HW3/HW_3.ipynb`

2 Pytorch Basics and Logistic Regression Model

Pytorch is a python library used to create neural networks, which we will be using for multiple homeworks throughout the semester. In pytorch, you can define a model class, where multiple operations can be performed on input data to produce some output prediction. The pytorch library defines several pre-set layers (such as a linear, convolutional, or pooling layer), which perform computations on an input. These layers are often chained together, where the output of one layer is fed into the next. Documentation for pytorch can be found here: https://pytorch.org/docs/stable/index.html.

The Colab notebook includes a logistic regression model (with averaged embedding document representation) to show an example of a complete pytorch model. The $\_\_init\_\_()$ method defines the layers which will be used in the model (i.e., the parameters to be learned). For this example, there is a single linear layer. The $forward()$ method takes the input data and defines the operations which should be applied to it, using the layers defined in the $\_\_init\_\_()$ method. For this example, the $self.linear$ layer is called on the original output and returned. The $evaluate()$ method is used to evaluate how the model performs on the dev dataset.

The Colab notebook includes two instantiations of the logistic regression model. The first one is an example that takes as input the average GLoVE embedding for all words in the document. This model is fully implemented and can be run without modifying any code. Secondly, there is an additional logistic regression model which takes in a bag-of-words featurization. You will be completing the bag-of-words implementation, similar to your function for HW2, in $read\_bow\_data()$ for your first deliverable. Observe how the bag-of-words featurization's dev accuracy compares to that of the average embedding representation model.

[1] https://arxiv.org/pdf/1510.03820.pdf

3 Deliverable 1: BoW Representation

The first deliverable for this homework is the creation of a bag of words feature for a logistic classifier model. Represent each **lowercase** word tokenized with the $NLTK.word\_tokenize$ function through its binary value; the feature value for a word that shows up in a review should be 1, no matter how many times it is mentioned.

4 Deliverable 2: CNN Model

The second deliverable for this homework is the creation of a CNN model inspired by Zhang and Wallace. This model performs 3 different convolutions on the input data, of window sizes 1, 2, and 3. The input data to this model is a sequence of tokens consisting of a movie's review. All examples are padded to the maximum length of 549 tokens. These tokens will be mapped to corresponding pre-trained 50-dimensional GLoVE word embeddings. You can read more about GloVe here: https://nlp.stanford.edu/projects/glove/

You will add code to two functions in the CNNClassifier class: $\_\_init\_\_()$ and $forward()$.

4.1 Model Initialization

A model's initialization method creates and initializes the layers with their correct sizes. You will initialize the following parameters:

- $self.embeddings$: This should initialize embeddings from the $pretrained\_embeddings$ parameter. These embeddings should be initialized to the GLoVE embedding values but should update during training.
- $self.conv\_1$, $self.conv\_2$, $self.conv\_3$: Each of these layers should be a $nn.Conv1d$ layer with 50 filters which convolves over the glove embeddings. $self.conv\_1$ should have a 1-word window, $self.conv\_2$ should have a 2-word window, and $self.conv\_3$ should have a 3-word window. All should have a stride of 1.
- $self.fc$: This should be a linear layer which takes in the concatenated 1-, 2-, and 3-word CNN representations and outputs a prediction over the output classes.

4.2 Forward Method

A model's forward method establishes how input data should be passed through the different model layers which you defined in $\_\_init\_\_()$. You will create the following intermediate/hidden representations:

- $x1$: should contain the representation learned from the 1-word convolutional layer, i.e. the 1-word convolution over the GLoVE embeddings, a tanh activation function, with max pooling applied (cnn $\rightarrow$ tanh $\rightarrow$ max)
- $x2$: should contain the representation learned from the 2-word convolutional layer, i.e. the 2-word convolution over the GLoVE embeddings, a tanh activation function, with max pooling applied (cnn $\rightarrow$ tanh $\rightarrow$ max)
- $x3$: should contain the representation learned from the 3-word convolutional layer, i.e. the 3-word convolution over the GLoVE embeddings, a tanh activation function, with max pooling applied (cnn $\rightarrow$ tanh $\rightarrow$ max)
- $combined$: should contain the concatenated representation from $x1$, $x2$, and $x3$.
- out: should contain the output from passing $combined$ through the linear layer.

4.3 Pytorch Debugging Tips

For this homework, you might find it helpful to use these tips to help debug your code.

- The pytorch function $.shape$ can be called on torch variables to view their dimensions. This might be helpful in ensuring the layer dimensions are set correctly.
- Python debugger is a valuable tool which you can use to step through your program's execution. The line of code "import pdb; pdb.set_trace()" will add a breakpoint to your code, where you can view dimensions of pytorch layers and experiment with function calls.
- If you find your model's dev accuracy is not increasing, it may be helpful to print or plot your model's training loss. Code to do this is included in the Model Exploration section. Oftentimes, if your training loss is not decreasing, the way you are connecting your layers is incorrect. You may find it useful to visualize how data should be flowing through your model's layers on paper and ensure this matches your $forward()$ function.

5 Deliverable 3: BoW vs. CNN

Answer the following question in 200 words or less in a PDF file:

> Compare and contrast the performance of your BoW representation and CNN. Did one model demonstrate a higher dev performance than the other? What do you see as the advantages of one model over the other that might lead to this performance difference on this data? Submit your <200 word answer to this question as a PDF on gradescope.

6 How to Submit

There are 3 deliverables which will be submitted to Gradescope. **Note: there are 2 different Gradescope submission links, one for code and one for the PDF writeup.**

- Submit to Gradescope HW3 code (ipynb)
  - Download your Colab notebook as an .ipynb file (File → Download .ipynb)
  - Submit **HW_3.ipynb**. The file must be named in this way for the Gradescope autograder.
- Submit to Gradescope HW3 writeup
  - Submit your answer for deliverable 3 as a PDF file.