Homework 3: Pytorch and CNNs

Due February 10, 2021 (11:59 PM)

1 Introduction and Task

For this homework, you will be training a convolutional neural network (CNN) classification model to predict the topical category from a paper's title and abstract. 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 created from HW1.

The colab notebook for this homework is located here: `https://github.com/dbamman/nlp21/blob/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 in pytorch, to show an example of a complete pytorch model. The $\_\_init\_\_()$ method defines the layers which will be used in the model. 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 example instantiations of the logistic regression model. The first one takes as input the average GLoVE embedding for all words in a paper's title and abstract. 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 of the paper's title and abstract. You can add a bag-of-words implementation similar to your function for HW2 into $read\_bow\_data()$ to experiment with this model. Observe how the bag-of-words featurization's dev accuracy compares to that of the average embedding representation model. **You are not required to add anything to this function or to the logistic regression classifier for this homework.**

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

3 Deliverable 1: CNN Model

The first 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 the paper's title and abstract separated by 2 SEP tokens. 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 $forward()$.

3.1 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.
- $self.pool\_1$, $self.pool\_2$, $self.pool\_3$: These should be the $nn.MaxPool1d$ layers corresponding to $self.conv\_1$, $self.conv\_2$, and $self.conv\_3$.
- $self.fc$: This should be a linear layer which takes in the concatenated 1-, 2-, and 3-word CNN representations and outputs a prediction over the topic category classes.

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

3.3 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.

4 Deliverable 2: ACL Data Exploration

Explore your model's predictions on the full set of ACL publications from 2013-2020. Answer the following question in 200 words or less in a PDF file:

> Should we trust these results as reflecting trends in the attention the ACL community gives to these topics? Think about the potential biases that might exist in this method and the results, especially given your experience in creating this dataset. Explore this model and data with whatever methods you think would help your argument – e.g., try plotting a confusion matrix over the development data to see which classes are being confused, examine the data points that have the highest confidence wrong predictions, etc.). How would you go about interrogating this method to know whether to trust these findings?

5 How to Submit

There are 2 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 Homework3 code
  - 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 Homework3 writeup as a PDF file.