BME646 and ECE60146: Homework 8

Spring 2023
Due Date: 11:59pm, Apr 17, 2023
TA: Fangda Li (li1208@purdue.edu)

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace. Late submissions will be accepted with penalty: **-10 points per-late-day, up to 5 days**.

1 Introduction

This homework has the following goals:

1. To gain insights into the workings of Recurrent Neural Networks. These are neural networks with feedback. You need such networks for language modeling, sequence-to-sequence learning (as in automatic translation systems), time-series data prediction, etc.

2. To understand how the gating mechanisms in GRU (Gated Recurrent Unit) helps combat the vanishing gradients problem in RNNs.

3. To use GRU-based RNNs for the modeling of variable-length product reviews provided by Amazon for automatic classification of such reviews.

2 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. Carefully review Slide 46 through 59 of the Week 12 slides on "Recurrent Neural Networks for Text Classification and Data Prediction" [2]. Make sure you understand how gating is done in GRU to address the problem of vanishing gradients that are caused by the long chains of feedback in a neural network.

2. Review the Week 13 slides on "Word Embeddings and Sequence-to-Sequence Learning" [3]. In particular, pay attention to Slide 39 through 49 on word2vec and fastText. Make yourself familiar with their use and advantages over one-hot vector encoding.

3. Download and install the newly released version 2.2.5 of DLStudio from the main documentation page [1]. Note that if you `pip install` it directly, you will not get the changes in the `Examples` directory of the distribution.

4. Download the text datasets from the same page into the `Examples` directory of the distribution and execute the following command on the downloaded gzipped archive:

   ```
   tar xvf text_datasets_for_DLStudio.tar.gz
   ```

   This will create a subdirectory 'data' in the `Examples` directory and deposit all the datasets in it. If you followed the previous instructions, you will find the following datasets in the `Examples/data/` directory:

   ```
   # sentiment400 (vocab_size = 64,350)
   sentiment_dataset_train_400.tar.gz
   sentiment_dataset_test_400.tar.gz

   # sentiment200 (vocab_size = 43,285)
   sentiment_dataset_train_200.tar.gz
   sentiment_dataset_test_200.tar.gz

   # sentiment40 (vocab_size = 17,001)
   sentiment_dataset_train_40.tar.gz
   sentiment_dataset_test_40.tar.gz

   # sentiment3 (vocab_size = 3,402)
   sentiment_dataset_train_3.tar.gz
   sentiment_dataset_test_3.tar.gz
   ```

   Refer to Slide 10 through 14 of the Week 12 slides to familiarize yourself with how the datasets are organized.

5. After you have installed the datasets, it's time to become familiar with some actual code for text classification. Do this by executing the following script in the `Examples` directory of the distribution:

   ```
   text_classification_with_GRU_word2vec.py
   ```

The script uses DLStudio's `GRUnetWithEmbeddings` as the RNN for word sequence processing. As the name suggests, it uses the word2vec embedding to represent each input word as opposed to the one-hot vector encoding.

6. Finally, you should go through Slide 94 to 96 of the Week 12 slide deck. This introduces you to Prof. Kak's implementation of the minimally gated GRU or pmGRU. Moreover, you should execute the following script in the `ExamplesDataPrediction` directory of the DLStudio module:

   `power_load_prediction_with_pmGRU.py`

   where the data can be downloaded from:

   https://engineering.purdue.edu/kak/distDLS/dataset_for_DataPrediction.tar.gz

With these steps, you are ready to start working on the homework described in what follows.

3 Programming Tasks

3.1 Sentiment Analysis with Your Own GRU

Your first task in this HW would be to write your own implementation of GRU. The steps are:

1. Before starting, make sure you fully understand the materials on GRU presented on Slide 46 through 59 of the Week 12 slides. You should gain intuitions on how the gating mechanisms in LSTM and GRU can combat the issue of vanishing gradients caused by the long chains of feedback. Additionally, you should be very comfortable with the equations on Slide 59 that summarize the interactions between all the variables within a GRU.

2. Based on that set of equations, implement your own GRU logic (without using `torch.nn.GRU`). Wrap your GRU module into a RNN that can be used for predicting sentiment. That is, your RNN should be able to take in a review, which is represented as a sequence of word embeddings produced by word2vec, and output a sentiment label (*i.e.* negative or positive).

3. Train your GRU-based RNN on the `sentiment400` training set for at least 4 epochs. Again, you should use the word embeddings produced by word2vec as the input to your RNN, as opposed to the one-hot vector representations. For better convergence, you should use the Adam optimizer and experiment with different learning rates. For quantitatively evaluating your RNN, record the confusion matrix as well as the overall prediction accuracy on the `sentiment400` test set.

4. In your report, designate a code block for your own implementation of GRU. Accompany that block with a paragraph explaining the GRU logic in detail and how you think the problem of vanishing gradients is mitigated with the gating mechanism. Include a plot of the training losses over iterations. Finally, report the confusion matrix and the overall accuracy achieved by your GRU-RNN on the `sentiment400` test set.

3.2 Sentiment Analysis Using `torch.nn.GRU`

In this task, we ask you to carry out the same sentiment prediction task but with PyTorch's GRU implementation. The steps are:

1. First, familiarize yourself with the documentation of the `torch.nn.GRU` module [4]. Also, you should go through Slide 67 through 87 of the Week 12 slides to understand how you may feed in an entire sequence of embeddings at once when using `torch.nn.GRU`. Using the module in such manner may help you speed up training dramatically.

2. Carry out the same sentiment prediction experiment that you did with your own GRU-RNN but this time use `torch.nn.GRU` instead. Perform the same quantitative evaluation on the `sentiment400` test set with your new RNN that is based on `torch.nn.GRU`.

3. Now, repeat the step above with PyTorch's bidirectional GRU (*i.e.* with `bidirectional=True`). Note that you'll also need to adjust several other places in your RNN to accommodate the change of shape for the hidden state. Does using a bidirectional scan make a difference in terms of test performance?

4. In your report, report the overall accuracy and the confusion matrix produced by your RNN that is based on `torch.nn.GRU` as well as its bidirectional variant. Also you should include plots of the training losses for the two RNNs. Write a paragraph comparing the test performances of all three RNN implementations that you have done.

4 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks.

1. **Make sure your submission zip file is under 10MB.** Compress your figures if needed. **Do NOT submit your network weights nor dataset.**

2. Your pdf must include a description of

   - The figures and descriptions as mentioned in Sec. 3.
   - Your source code. Make sure that your source code files are adequately commented and cleaned up.

3. Turn in a zipped file, it should include (a) a typed self-contained pdf report with source code and results and (b) source code files (only .py files are accepted). Rename your .zip file as hw8_<First Name><Last Name>.zip and follow the same file naming convention for your pdf report too.

4. For all homeworks, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert it to `.py` and submit that as source code.

5. You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once on BrightSpace.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

6. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**

7. To help better provide feedbacks to you, make sure to **number your figures**.

References

[1] DLStudio. URL https://engineering.purdue.edu/kak/distDLS/.

[2] Recurrent Neural Networks for Text Classification and Data Prediction, . URL https://engineering.purdue.edu/DeepLearn/pdf-kak/RNN.pdf.

[3] Word Embeddings and Sequence-to-Sequence Learning, . URL https://engineering.purdue.edu/DeepLearn/pdf-kak/Seq2Seq.pdf.

[4] GRU. URL https://pytorch.org/docs/stable/generated/torch.nn.GRU.html.