Deep Learning 2020 **(Due: [Dec 2])**

Submission Assignment #[4]

Instructor: Jakub Tomczak *Name:* [Ho, Yu-Feng], *Netid:* [2689918]

1 Problem statement

- Batch processing
- Model constructing (MLP, Elman RNN, LSTM)
- Performance among three models(MLP, Elman RNN, LSTM)

2 Methodology

- Slice out batches Pad tokens

  IMDb dataset is a large movie review dataset for binary sentiment classification Maas et al. (2011). It contains 50,000 reviews of movies. I divided it into a training set that contains 20,000 reviews and a validation split that contains 5,000 reviews. Classification is [0,1] depending on positive and negative sentiment. First, the datasets were sorted from short to long words. Second, I sliced out the datasets by striding 40. Therefore I got 500 batches for the training set and 125 batches for the validation split. Furthermore, I also did a **variable batch size**. Third, for each batch I padded 0 behind to make it contains the same size of tokens. **However, the amount of padding does increase with the batch size because the larger batch contains more diverse length of sentences.**

- Models structure

  There are three models totally in this report, No-recurrent model, Elman Recurrent model and long short-term memory (LSTM) model. The first layer is embedding layer, and **the number of tokens as an indicator of embedding vectors can be found by len(i2w), the length of the list of strings**. The main difference is the hidden layer. No-recurrent model has a embedding layer and two fully connected layers. Elman Recurrent model switched the first fully connected layer to recurrent hidden layer. Likewise, LSTM switched to LSTM hidden layer. However, the weights and biases are initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$,where $k = \frac{1}{\text{hidden\_size}}$

- Hyperparameters set

  Models have different hyperparameters usage. No-recurrent model contains 300 embedding size and hidden size, so do Elman Recurrent model and LSTM. Activation function is used $ReLU$ in all layers, and apply $softmax$ as calculating cross entropy for loss. Learning rates are set as 0.0005 and 0.001 by introducing ADAM optimizer $(\beta_1, \beta_2) = (0.9, 0.999)$. Finally, every model ran for epochs in order to analyze performance.

3 Results and discussion

- Accuracy test

  As set learning rate as 0.0005, **the validation accuracy after first epoch is 87% in Non-recurrent model, and for variable batch size is 84%**. Back to accuracy test regarding 35 epochs, the final accuracy of MLP converges at 0.9958, of RNN converges at 0.9974, and of LSTM converges at 0.9994. It can be declared that LSTM contains the best performance among models. To evaluate the performance, the accuracy plot is introduced (Figure 1). It shows that after 15 epochs all models almost converge, LSTM reach to the highest accuracy, and RNN gradually surpasses MLP to the second place. Furthermore, in the process of hyperparameters tuning, I introduced learning rate = 0.001 for 10 epochs as a trial. The result demonstrates in Figure 2, and it indicates that RNN is more stable in high learning rates but LSTM performs well in the proper hyperparameters set.

- Improvement from MLP, to RNN, to LSTM

  In sentiment analysis, it is important to remember the former information, because the words in one sentence are all relevant and construct the context. Recurrent model has the ability to receive the former information in one batch Dang et al. (2020). Furthermore, Long short-term memory model is a modification of RNN, avoiding the practical difficulties of sequence span long interval Bengio et al. (1994). The problem is derived from remembering the words irrelevant to the context but influence the understanding. LSTM can forget irrelevant words to enhance accuracy. It is common in natural language and rational that people sometimes make statements with ambiguity.

- Result plot

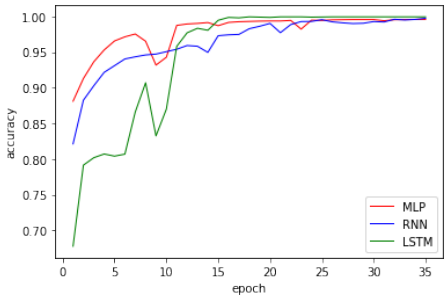

Figure 1: Accuracy curve of MLP, RNN, and LSTM with learning rate = 0.0005 and 35 epochs

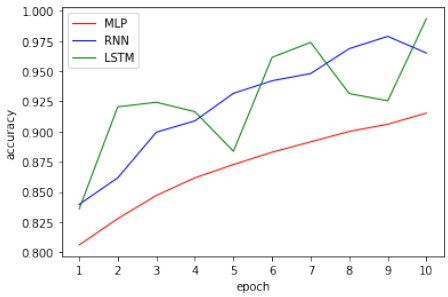

Figure 2: Accuracy curve of MLP, RNN, and LSTM with a high learning rate and fewer epochs(learning rate = 0.001)

References

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Dang, C., Moreno García, M., and De La Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9:483.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.