Homework 3: Recurrent Neural Networks

Deep Learning (84100342-0)
Spring 2018
Tsinghua University

1 Introduction

In this homework, we will try to solve a language modeling problem by designing and implementing recurrent neural networks (RNNs). A good understanding of basic architecture of RNN and some widely-used RNN units is required. Some optimization techniques might also be leveraged to facilitate the training process.

2 Language Modeling

Language modeling [3, 2, 1, 4] is a central task in NLP and language models can be found at the heart of speech recognition, machine translation, and many other systems. A language model is a probability distribution over sequences of words. Given such a sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$ with length m, it assigns a probability $P(\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$ to the whole sequence. In detail, you have a vocabulary dictionary of words $(\boldsymbol{v}_1, ..., \boldsymbol{v}_n)$. And given a sequence of words $(\boldsymbol{x}_1, ..., \boldsymbol{x}_t)$ a language model predicts the following word $\boldsymbol{x}_{t+1}$ by modeling: $P(\boldsymbol{x}_{t+1} = \boldsymbol{v}_j | \boldsymbol{x}_t, ..., \boldsymbol{x}_1)$ where $\boldsymbol{v}_j$ is a word in the vocabulary dictionary.

Conventionally, we evaluate our language model in terms of perplexity (PP), which is the inverse probability of the correct word, according to the model distribution P. It is defined as:

$$PP^t = \frac{1}{P(\boldsymbol{x}_{t+1}^{pred} = \boldsymbol{x}_{t+1} | \boldsymbol{x}_t, ..., \boldsymbol{x}_1)} \tag{1}$$

where $\boldsymbol{x}_{t+1}^{pred}$ is the prediction of our language model at time step $t + 1$.

3 Dataset

Please train your models on one text dataset provided by us:

- The dataset have three parts: train set, valid set, test set and the test set is not provided.
- Directory structure: "$./src/$" contains the start code and "$./data/$" contains the train set and the valid set.

4 Requirements and Evaluations

4.1 Programming Language

Python only.

4.2 Deep Learning Framework

We recommend PyTorch and TensorFlow. If using other frameworks, please contact TA.

4.3 Scoring

- Construct your own RNNs and train your model from scratch (fine-tuning is forbidden) using the recommended deep learning framework. (40%)
- For constructing RNN, you may use well-established libraries, such as torch.nn. Otherwise, it is a bonus (10%) if you implement the RNN network with basic arithmetic operators (e.g. torch., torch.mm, torch.cat).
- Train a model on the training set and cross-validate your model on the "valid_set.txt" to achieve a good performance on the "test_set" which is not provided. (30%)
- Use extra techniques you find in other materials to further improve your model. Please explain why you choose it and how it works. (30%)
- It is a bonus to write your own code with some better ideas such as pre-processing and evaluation. (10%)

4.4 Notification

- We have provided the start code to concentrate your attention on the construction and training of RNN itself. However, if you are able to write your own code with some better ideas such as pre-processing and evaluation, you can get a bonus with regard to what you implement. For example, **BLEU**, **METEOR**, **ROUGE** are also good evaluation methods.
- Please submit your code, document and trained models as an Archive (zip or tar). The document is supposed to cover your insights of the proposed model, the technical details, the experimental results (including training and cross-validation), and the necessary references. If you use additional data sources, please be sure to specify.
- We will focus on your code and document to decide your score. Still, under equal conditions (novelty, code quality, document quality), a higher accuracy along with a reasonable computation efficiency indicates a higher score.

References

[1] H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462, 2016.

[2] Y. Lv and C. Zhai. Positional language models for information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 299–306, New York, NY, USA, 2009. ACM.

[3] O. Press and L. Wolf. Using the output embedding to improve language models. *CoRR*, abs/1608.05859, 2016.

[4] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.