Sentence-level Sentiment Classification with RNN

Background

In this homework, we focus on fine-grained sentence-level sentiment classification problem. After implementing the details about MLP and CNN in python with Numpy and Pytorch framework, you have learned basic skills in deep learning. In this homework, you can apply all the skills in sentence-level sentiment classification problem to improve the performance of your network.

Requirements

pytorch==1.0, torchtext==0.3.1, nltk==3.3

Dataset

We use Stanford Sentiment Treebank (SST) dataset for experiments. This dataset contains 11,855 sentences, and has been splitted into the training / validation / test parts, containing 8,544 / 1,101 / 2,210 sentences respectively. Each sentence is categorized into one type of 5 sentiments.

To preprocess the dataset, we use `torchtext` package. Before training the model, we need to tokenize words, building vocabulary, construct word embedding and intialize data iterator. `torchtext` has good wrappers for these operations and we can directly use dataloader in a similar way as before.

Note: since we need to utilize pre-computed wording embedding, `torchtext` will download `glove` word embedding for the first running, and it may take a while. Also prepare at least 4G space for saving files. The downloaded SST data and embedding vectors are in `.data` and `.vector_cache` directories. Remember **DO NOT** include them in your submission file.

Python Files Descriptions

- `main.py`: the main script for running the whole program
- `model.py`: one RNN layer model construction. It handles how to project word tokenizers into embedding vectors, feed into the RNN cell, run multiple recurrent loops and get final prediction results.
- `cell.py`: basic RNN cells, including
    - `RNNCell`: An Elman RNN cell with tanh non-linearity

$$h' = \tanh(W_{ih}x + b_{ih} + W_{hh}x + b_{hh})$$

    - `GRUCell`: A gated recurrent unit cell

$$
\begin{aligned}
r &= \sigma(W_{ir}x + b_{ir} + W_{hr}h + b_{hr}) \\
z &= \sigma(W_{iz}x + b_{iz} + W_{hz}h + b_{hz}) \\
n &= \tanh(W_{in}x + b_{in} + r * (W_{hn}h + b_{hn})) \\
h' &= (1 - z) * n + z * h
\end{aligned}
$$

- `LSTMCell`: A long short-term memory cell

$$
\begin{aligned}
i &= \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \\
f &= \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \\
g &= \tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg}) \\
o &= \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \\
c' &= f * c + i * g \\
h' &= o * \tanh(c')
\end{aligned}
$$

Note: These three cells have a common forward API: `def forward(self, input, state)`. For `RNNCell` and `GRUCell`, the state is simply hidden state vector `h`. For `LSTMCell`, the state is the tuple of `(h, c)`.

Report

First you need to finish the missing parts of `cell.py` and `model.py`. Then in the experiment report, you need to answer the following questions:

1. Plot the loss value and accuracy value of one-layer RNN with 3 rnncells against to every epoch during training
2. Compare and analyze the performance of the 3 rnncells.
3. Currently we are using 300 dimensional embedding vector. Try to use 100 dimensional embedding vector by changing `vectors="glove.6B.100d"` with the best rnncell model you achieve above and compare the performance
4. Currently for the unknown words, we intialize their embedding vectors with normal distribution. Try to set `unk_init=None` with the best rnncell model you achieve in (2) and compare the performance

Submission Guideline:

You need to submit both report and codes, which are:

- **report**: well formatted and readable summary including your results, discussions and ideas. Source codes should *not* be included in report writing. Only some essential lines of codes are permitted for explaining complicated thoughts.

- **codes**: organized source code files with README for extra modifications or specific usage. Ensure that others can successfully *reproduce* your results following your instructions. **DO NOT include model weights/raw data/compiled objects/unrelated stuff over 50MB**

Deadline: May. 22th

TA contact info: Yulong Wang (王宇龙）, wang-yl15@mails.tsinghua.edu.cn