Programming Assignment 2

Recurrent Neural Network

CSE303: Introduction to Deep Learning

Objective

- You can do either (main) or (substitute)

- (Main)
  1. Recurrent Neural Network for sentiment analysis generation without deep learning framework (only python)

- (Substitute: 20% of total credit)

  If you cannot do main, you can do this:
  1. Recurrent Neural Network for sentiment analysis generation using a deep learning framework (e.g. pytorch, tensorflow)

[Andrew Ng, Sequential Models Course, Deep Learning Specialization]

Overall steps

1. **Prepare the training/test datasets and word embedding**
    1. Prepare data (Sentiments, Glove)
    2. Implement data loader and word embedding
2. **Design Recurrent Neural Network (using only python)**
    1. Design RNN (Vanilla RNN and LSTM)
    2. Design output layer with Cross-Entropy loss
    3. Design Dropout
3. **Implement whole training & test pipeline with optimizer**
    1. Design optimizer (SGD and ADAM)
    2. Training procedure
4. **Evaluate the performance**

Overall steps

1. **Prepare the training/test datasets and word embedding**
    1. Prepare data (Sentiments, Glove)
    2. Implement data loader and word embedding
2. **Design Recurrent Neural Network (using only python)**
    1. Design RNN (Vanilla RNN and LSTM)
    2. Design output layer with Cross-Entropy loss
    3. Design Dropout
3. **Implement whole training & test pipeline with optimizer**
    1. Design optimizer (SGD and ADAM)
    2. Training procedure
4. **Evaluate the performance**

1. Prepare training/test dataset & word embedding

1. **Prepare the dataset & load data (using emo_utils.py)**
   1. *Sentiment analysis* dataset [1]: 'test_emoji.csv', 'train_emoji.csv' (Check the uploaded file: **data_train_test.zip**)
      1. Train: 132 sentences, Test: 56 sentences with 5 emoji
   2. *Glove*: word embedding
      1. Prepare '**glove.6B.50d.txt**' and '**glove.6B.100d.txt**' that read 50d features and 100d features glove file (download link: [2])

Sentiments example

X (sentences)	Y (labels)
I love you	0
Congrats on the new job	2
I think I will end up alone	3
I want to have sushi for dinner!	4
It was funny lol	2
she did not answer my text	3
Happy new year	2
my algorithm performs poorly	3
he can pitch really well	1
you are failing this exercise	3
you did well on your exam.	2
What you did was awesome	2
I am frustrated	3

code	emoji	label
:heart:	❤️	0
:baseball:	⚾	1
:smile:	😀	2
:disappointed:	😔	3
:fork_and_knife:	🍴	4

[1] https://github.com/omerbsezer/LSTM_RNN_Tutorials_with_Demo
[2] http://nlp.stanford.edu/data/glove.6B.zip (Reference: https://nlp.stanford.edu/projects/glove/)

What is Glove?

- **Glove, EMNLP'14**
  - An unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word–word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
    - Project site: https://nlp.stanford.edu/projects/glove/
    - Paper: https://nlp.stanford.edu/pubs/glove.pdf
    - **Download:** http://nlp.stanford.edu/data/glove.6B.zip

    - word_to_index: dictionary mapping from words to their indices in the vocabulary (400,001 words, with the valid indices ranging from 0 to 400,000)
    - index_to_word: dictionary mapping from indices to their corresponding words in the vocabulary
    - word_to_vec_map: dictionary mapping words to their GloVe vector representation.

[1] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." *EMNLP* 2014.

Overall steps

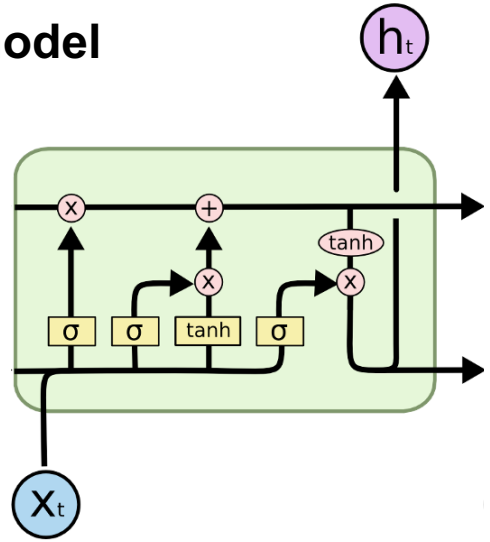
1. Prepare the training/test datasets and word embedding
    1. Prepare data (Sentiments, Glove)
    2. Implement data loader and word embedding
2. Design Recurrent Neural Network (using only python)
    1. Design RNN (Vanilla RNN and LSTM)
    2. Design output layer with Cross-Entropy loss
    3. Design Dropout
3. Implement whole training & test pipeline with optimizer
    1. Design optimizer (SGD and ADAM)
    2. Training procedure
4. Evaluate the performance

2. Design neural network

1. Overall steps

1. Design both RNN & LSTM layers
2. Design Dropout layer
3. Design FC layer and SoftMax (*If you've designed already, use them*)
4. Design Cross-entropy loss (*If you've designed already, use them*)

LSTM model



LSTM computation

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

2. Design neural network

- Overall network

Bias should be added

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 10)	0
embedding_1 (Embedding)	(None, 10, 50)	20000050
lstm_1 (LSTM)	(None, 10, 128)	91648
dropout_1 (Dropout)	(None, 10, 128)	0
lstm_2 (LSTM)	(None, 128)	131584
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 5)	645
activation_1 (Activation)	(None, 5)	0

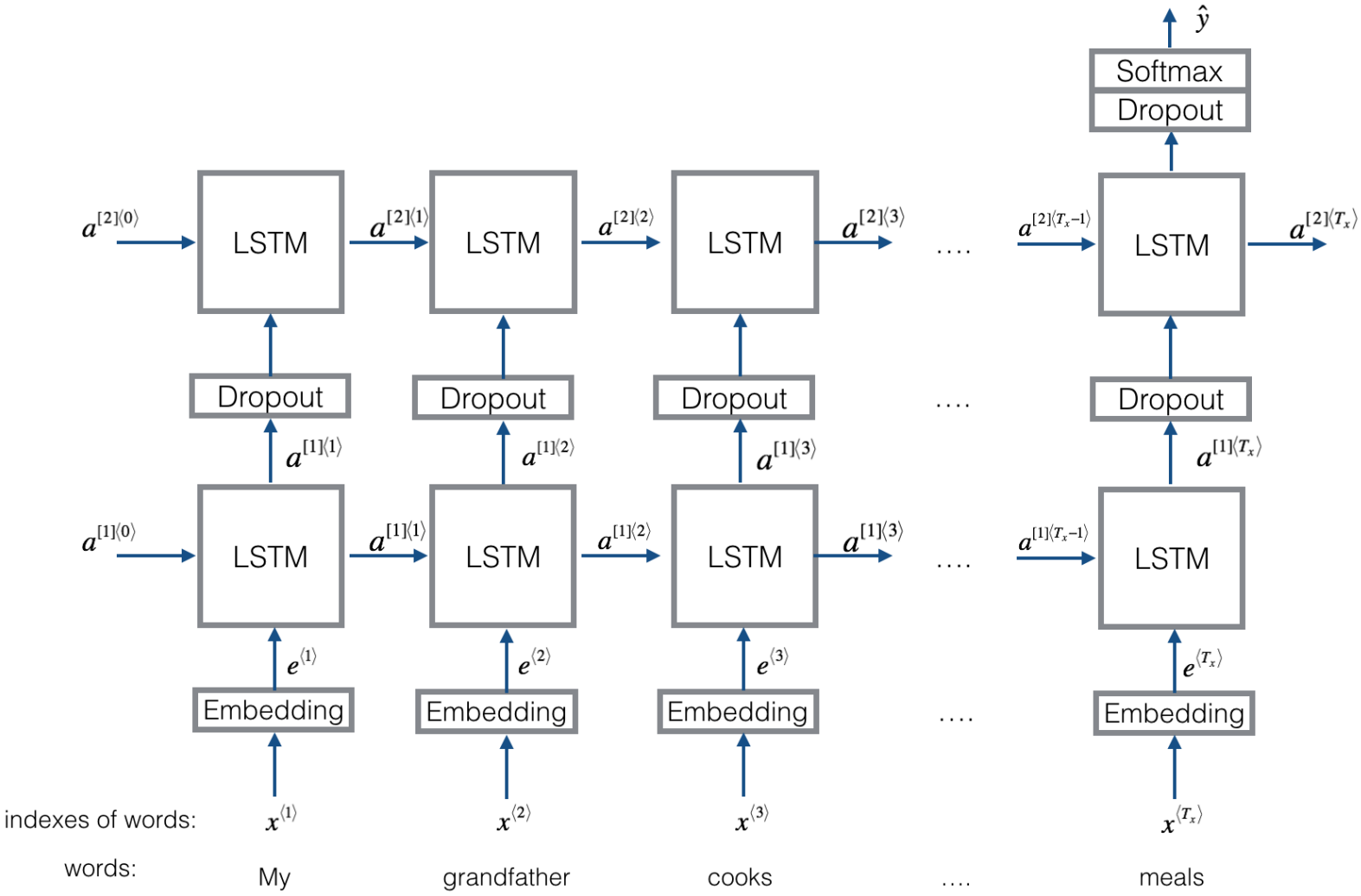
Total params: 20,223,927
Trainable params: 20,223,927
Non-trainable params: 0

Use both
- LSTM & RNN
- 50D & 100D
- SGD & ADAM

2. Design neural network

- Overall structure

Bias should be added



Overall steps

1. Prepare the training/test datasets and word embedding
    1. Prepare data (Sentiments, Glove)
    2. Implement data loader and word embedding
2. Design Recurrent Neural Network (using only python)
    1. Design RNN (Vanilla RNN and LSTM)
    2. Design output layer with Cross-Entropy loss
    3. Design Dropout
3. Implement whole training & test pipeline with optimizer
    1. Design optimizer (SGD and ADAM)
    2. Training procedure
4. Evaluate the performance

3. Implement whole training & test pipeline

1. Design optimizer (SGD and ADAM)

2. Training procedure
    1. Initialize the model parameters
    2. Implement and do forward propagation
    3. Implement and compute the cross-entropy loss
    4. Implement and do backward propagation
    5. Implement and update model parameter using optimizer
    6. Draw the plot of the loss

Overall steps

1. Prepare the training/test datasets and word embedding
    1. Prepare data (Sentiments, Glove)
    2. Implement data loader and word embedding
2. Design Recurrent Neural Network (using only python)
    1. Design RNN (Vanilla RNN and LSTM)
    2. Design output layer with Cross-Entropy loss
    3. Design Dropout
3. Implement whole training & test pipeline with optimizer
    1. Design optimizer (SGD and ADAM)
    2. Training procedure
4. Evaluate the performance

4. Evaluate the performance (1)

- **Measure all accuracy of four cases using test sets & report results (emoji), accuracies, and training loss graph for all cases**
  - Vanilla RNN vs LSTM
    - Compare the results between RNN and LSTM
  - SGD vs ADAM
    - Implement dropout technique
    - Train with/without dropout
    - Report the accuracy comparison dropout
  - Glove50d vs Glove100d (The length of the Glove vector)

Submission

Due : Nov 24, 11:59PM
To : lms.dgist.ac.kr

1. **Submit zip file including (1) Source code, (2) PDF file(report)**
   1. File name: PA2_studentID_name.zip (PA2_202312234_김종민.zip)
   2. Submission exclude training & Test dataset
2. **Report should include the results and results comparisons:**
   You can submit either (main) or (substitute)
   - (**main**) Show the results of Five Different Settings: (a) RNN+SGD+50d, (b) LSTM+SGD+50d, (c) LSTM+ADAM+50d, (d) LSTM+SGD+100d, (e) LSTM+SGD+50d+dropout followings:
     (1) accuracy comparison for test set
     (2) all emojis for test set
     (3) Training Loss graph
     & Describe the results comparison of optimizers (SGD, ADAM), RNN structures (RNN, LSTM), and the length of glove vectors (50d, 100d) & Describe the difference between Word2Vec and Glove in terms of vector generation
   - (**substitute**) Submit the same results above using deep learning framework
3. **Final credit: max(main, substitute*20%)**

Notice

Due : Nov 24, 11:59PM
To : lms.dgist.ac.kr

1. **Library**
    1. You cannot use a deep learning framework(Tensorflow, Pytorch, etc), but you can use Numpy or other libraries.
2. **Delayed submission**
    1. 25% score will be degraded every 1-day delay & after 3 days delayed, you will get 10% of the total score
    (e.g., 100% → 75% (1day) → 50% (2day) → 25% (3day) → 10% (〉3day)
3. **Plagiarism**
    1. No grade for copied codes (from friends and the internet)
    2. You can refer to sources from the internet, but do not copy and paste.
4. **Partial credit**
    1. Even though you are not successfully designing the network and obtained reasonable results, please send your code.
    2. There will be partial credit for each module implementation.