CSC413PA1

Che Liu, 1002246839

February 4, 2020

Part 1: Linear Embedding - GLoVE (4pts)

$$L\left(\{\mathbf{w}_i, b_i\}_{i=1}^{V}\right) = \sum_{i,j=1}^{V} \left(\mathbf{w}_i^\top \mathbf{w}_j + b_i + b_j - \log X_{ij}\right)^2$$

1. Given the vocabulary size *V* and embedding dimensionality *d*, how many trainable parameters does the GLoVE model have?

$\mathbf{w}_i$ and b_i are trainable parameters for each distinct word.
# of trainable parameters = (d + 1)*V

2. Write the gradient of the loss function with respect to one parameter vector $\mathbf{w_i}$

$$\overline{\mathbf{w}}_i = 4\sum_{j=1}^{V}((\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j - \log X_{ij})\mathbf{w}_j)$$

3. Implement the gradient update of GLoVE in language_model.ipynb.

$$\overline{\mathbf{W}} = 4(\mathbf{W}\mathbf{W}^T\mathbf{W} + \mathbf{b}\mathbf{1}^T\mathbf{W} + \mathbf{1}\mathbf{b}^T\mathbf{W} - \log \mathbf{X}\mathbf{W})$$
$$\overline{\mathbf{b}} = 4(\mathbf{W}\mathbf{W}^T\mathbf{1} + \mathbf{b}\mathbf{1}^T\mathbf{1} + \mathbf{1}\mathbf{b}^T\mathbf{1} - \log \mathbf{X}\mathbf{1})$$

4. Train the model with varying dimensionality *d*. Which *d* leads to optimal validation performance? Why does / doesn't larger *d* always lead to better validation error?

The learning rate was adjusted to $0.01/n$ to prevent numerical overflow.

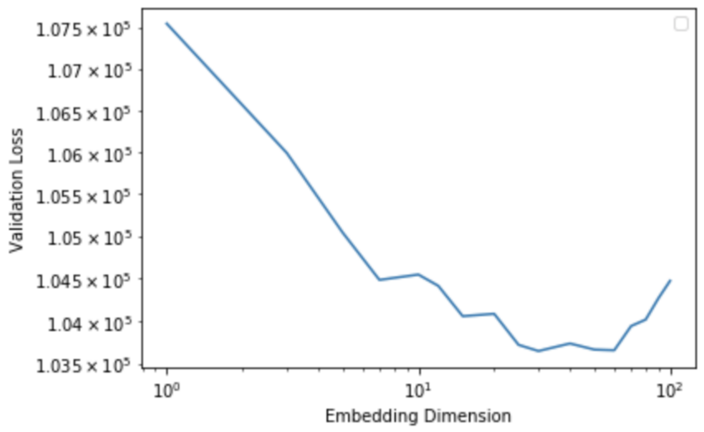

Figure 1: Plot of validation loss vs. embedding dimension

The model was trained with embedding dimension d equals 1, 3, 5, 7, 10, 12, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, and 100. As can be seen in the plot above, an embedding dimension of around 60 produced the optimal validation loss. A larger d doesn't always lead to better validation loss because of the potential of overfitting. In the case of overfitting, the neural network simply remembers the next word rather than trying to predict using the underlying patterns in the training text.

Part 2: Network architecture (2pts)

1. As above, assume we have 250 words in the dictionary and use the previous 3 words as inputs. Suppose we use a 16-dimensional word embedding and a hidden layer with 128 units. The trainable parameters of the model consist of 3 weight matrices and 2 sets of biases. What is the total number of trainable parameters in the model? Which part of the model has the largest number of trainable parameters?

A = # of embedding matrix weights = 250*16 = 4000
B = # of embedding to hidden weights = (3*16)*128 = 6144
C = # of embedding to hidden biases = 128
D = # of hidden to output weights = 128*250 = 32000
E = # of hidden to output biases = 250
# of trainable parameters = A + B + C + D + E = 42522
The hidden layer to output layer part has the most trainable parameters.

2. Another method for predicting the next word is an *n-gram model*, which was mentioned in Lecture 7. If we wanted to use an n-gram model with the same context length as our network, we'd need to store the counts of all possible 4-grams. If we stored all the counts explicitly, how many entries would this table have?

# of entries = # of possible permutations * # of words = 15438000 * 250 = 3859500000

Part 3: Training the Neural Network (4pts)

- **compute_loss_derivative**
  This function computes $\frac{\partial C}{\partial \mathbf{z}}$, whose dimension is $B \times N_V$
  B: batch size
  N_V: dictionary size

$$\bar{\mathbf{z}} = \frac{\partial C}{\partial \mathbf{z}} = \mathbf{Y} - \mathbf{T} \in \mathcal{R}^{B \times N_V}$$

  where

$$\mathbf{Y} = \begin{bmatrix} -\mathbf{y}_1- \\ -\mathbf{y}_2- \\ \cdots \\ -\mathbf{y}_B- \end{bmatrix}, \mathbf{y}_i \in \mathcal{R}^{1 \times N_V}$$

$$\mathbf{T} = \begin{bmatrix} -\mathbf{t}_1- \\ -\mathbf{t}_2- \\ \cdots \\ -\mathbf{t}_B- \end{bmatrix}, \mathbf{t}_i \in \mathcal{R}^{1 \times N_V}$$

- **embed_to_hid_weights_grad**
  This function computes $\frac{\partial C}{\partial \mathbf{W}_{he}}$, whose dimension is $N_H \times 3D$
  N_H: number of hidden units
  D: embedding dimension

$$\overline{\mathbf{W}}_{he} = \frac{\partial C}{\partial \mathbf{W}_{he}} = \bar{\mathbf{h}}^T \mathbf{E}$$

- **hid_bias_grad**
  This function computes $\frac{\partial C}{\partial \mathbf{b}_h}$, whose dimension is N_H
  N_H: number of hidden units

$$\bar{\mathbf{b}}_h = \frac{\partial C}{\partial \mathbf{b}_h} = \bar{\mathbf{h}}^T \mathbf{1}$$

- **hid_to_output_weights_grad**
  This function computes $\frac{\partial C}{\partial \mathbf{W}_{oh}}$, whose dimension is $N_V \times N_H$
  N_V: number of output units, or number of words
  N_H: number of hidden units

$$\overline{\mathbf{W}}_{oh} = \frac{\partial C}{\partial \mathbf{W}_{oh}} = \overline{\mathbf{z}}^T \mathbf{h}$$

- **output_bias_grad**
  This function computes $\frac{\partial C}{\partial \mathbf{b}_o}$, whose dimension is N_V
  N_V: number of output units, or number of words

$$\overline{\mathbf{b}}_o = \frac{\partial C}{\partial \mathbf{b}_o} = \overline{\mathbf{z}}^T \mathbf{1}$$

Output of the function *print_gradients*:

```
loss_derivative[2, 5] 0.001112231773782498
loss_derivative[2, 121] -0.9991004720395987
loss_derivative[5, 33] 0.0001903237803173703
loss_derivative[5, 31] -0.7999757709589483

param_gradient.word_embedding_weights[27, 2] -0.27199539981936866
param_gradient.word_embedding_weights[43, 3] 0.8641722267354154
param_gradient.word_embedding_weights[22, 4] -0.2546730202374648
param_gradient.word_embedding_weights[2, 5] 0.0

param_gradient.embed_to_hid_weights[10, 2] -0.6526990313918257
param_gradient.embed_to_hid_weights[15, 3] -0.13106433000472612
param_gradient.embed_to_hid_weights[30, 9] 0.11846774618169399
param_gradient.embed_to_hid_weights[35, 21] -0.10004526104604386

param_gradient.hid_bias[10] 0.2537663873815642
param_gradient.hid_bias[20] -0.03326739163635357

param_gradient.output_bias[0] -2.0627596032173052
param_gradient.output_bias[1] 0.0390200857392169
param_gradient.output_bias[2] -0.7561537928318482
param_gradient.output_bias[3] 0.21235172051123635
```

Part 4: Analysis (4pts)

1. Pick three words from the vocabulary that go well together (for example, 'government of united', 'city of new', 'life in the', 'he is the' etc.). Use the model to predict the next word. Does the model give sensible predictions? Try to find an example where it makes a plausible prediction even though the 4-gram wasn't present in the dataset (raw_sentences.txt). To help you out, the function find_occurrences lists the words that appear after a given 3-gram in the training set.

Using the trained model, the predictions and probabilities of different 4-grams are as follows:

- 'government of united' $\longrightarrow$ '.' (0.17502), 'own' (0.12561), 'states' (0.08049)
- 'city of new' $\longrightarrow$ 'york' (0.96931), '.' (0.00834), ',' (0.00266)
- "life in the" $\longrightarrow$ 'world' (0.14029), 'game' (0.08300), 'street' (0.06545)
- 'he is the' $\longrightarrow$ 'best' (0.17128), 'same' (0.10897), 'only' (0.09627)
- 'as long as' $\longrightarrow$ 'he' (0.19866), 'i' (0.16571), 'it' (0.14390)

- 'not the first' $\longrightarrow$ '.' (0.32289), 'time' (0.14007), 'one' (0.07764)
- 'i have to' $\longrightarrow$ 'do' (0.17456), 'get' (0.09619), '.' (0.08752)

The model seems to give sensible predictions because the vast majority of them are grammarly correct. For instance, the model learned that the word following "life in the" should be a noun, and the word following "he is the" should be an adjective.

An example where the trained model makes a plausible prediction even through the 4-gram wasn't present in the dataset is 'if you were'. Two instances of 4-grams present in the dataset are 'if you were a city' and 'if you were me'. The top three words that the model predicts are 'going', 'nt' (for "weren't"), and 'not', with probabilities 0.13487, 0.06253, and 0.05236, respectively. None of the three predictions were present in the dataset, but all of them are obviously plausible.

2. Plot the 2-dimensional visualization using the method tsne_plot_representation. Look at the plot and find a few clusters of related words. What do the words in each cluster have in common? Plot the 2-dimensional visualization using the method tsne_plot_GLoVE_representation for a 256 dimensional embedding. How do the t-SNE embeddings for both models compare? Plot the 2-dimensional visualization using the method plot_2d_GLoVE_representation. How does this compare to the t-SNE embeddings? (You don't need to include the plots with your submission.)

For the plot generated by tsne_plot_representation, the words that belong to the same cluster belong (mostly) to the most part of speech. For instance, the words 'i', 'you', 'we', 'they' form a cluster near top-middle part of the plot, and they are all pronouns. Another example would be the words 'when', 'where', 'who', 'what', and 'how'. They are all interrogative word and form a cluster at the bottom-left part of the plot. Furthermore, 'say', 'says', and 'said' belong to the same cluster. They are essentially the same word but in third person or different tense.

As for tsne_plot_GLoVE_representation, it is harder to make sense of the clusters since the embedding size is 256D, which is much bigger than before. When visualize in 2D, the 2D distance between two words hardly reflect the actual distance between two words in the 256-dimensional space. Words that are very far apart in 256D may end up close together in 2D. However, some clusters still make sense, such as the one containing 'might', 'may' and 'should'.

Finally, for plot_2d_GLoVE_representation, the words seem to be more concentrated than the two t-SNE plots. Since the words are all on top of each other, it is hard to see what words are within each cluster.

3. Are the words 'new' and 'york' close together in the learned representation? Why or why not?

Whether the two words are close together can be determined by looking at words that are close to them. For 'new', the top 10 nearest words are 'old', 'white', 'big', 'political', 'united', 'american', 'its', 'several', 'another', and 'our'. Whereas for 'york', the words are 'states', 'school', 'ms.', 'general', 'public', 'music', 'state', '?', 'city', and 'former'. Since the word 'new' is an adjective, seven out of ten words are adjectives. On the other hand, the word 'york' is a noun, so six out of the ten closest words are nouns. Since their nearest words are so different, the words 'new' and 'york' are not close together in the learned representation.

4. Which pair of words is closer together in the learned representation: ('government', 'political'), or ('government', 'university')? Why do you think this is?

The distance between 'government' and 'university' are 1.132.
The distance between 'government' and 'political' are 1.222.

'government' and 'university' are closer together in the learned representation, because the distance between their vector representations are smaller than that between 'government' and 'political'. Intuitively, since 'government' and 'university' are both nouns but 'political' is an adjective, the first two are more likely to be used in similar context. An example would be 'I work at the government' and 'I work at the university'.