CSC413 Programming Assignment One

ZhenDi Pan 1003241823

2020-02-03

Part One

Question 1

The number of trainable parameters of the matrix W with V distinct words and d dimension is $V \times d$, since there are V words and each word has d parameters. For every word, there is bias scalar b_i, so the total number of trainable parameters are $V(d+1)$.

Question 2

The scanned hand-written steps are as follow:

$$\nabla_{w_i} J(w_i, b_i) = \frac{\partial \left(2 \sum_{\substack{j \\ j \neq i}}^{V} (w_i^T w_j + b_i + b_j - \log X_{ij})^2 + (w_i^T w_i + b_i + b_i - \log X_{ii})^2 \right)}{\partial w_i}$$

$$= 4 \sum_{\substack{j=1 \\ j \neq i}}^{V} (w_i^T w_j + b_i + b_j - \log X_{ij}) w_j + 4 w_i (w_i^T w_i + b_i + b_i - \log X_{ii})$$

$$= 4 \sum_{j=1}^{V} (w_i^T w_j + b_i + b_j - \log X_{ij}) w_j$$

So the graient of the loss function w.r.t to w_i is: $4 \sum_{j=1}^{V} (w_i^T w_j + b_i + b_j - log X_{ij}) w_j$

Question 4

$d = 10$ leads to the optimal validation performance. When d gets too large, there is the problem of over-fitting. Too many parameters are used, this result in low training error however the model captures the training set too well and thus it has high variance. In other words, it is excessively complicated, so it leads to bad performance on validation error.

Part Two

Question 1

The number of trainable parameters in word_embedding_weights: $250 \times 16 = 4000$.

The number of trainable parameters in embed_to_hid_weights: $128 \times 48 = 6144$.

The number of trainable parameters in hid_bias: 128

The number of trainable parameters in hid_to_output_weights: $250 \times 128 = 32000$

The number of trainable parameters in output_bias: 250

The total number of trainable parameters is 42522, hid_to_output_weights has the largest number of parameters.

Question 2

The table would have to store all possible permutations of the words of 4-gram. So it is $250^4 = 3906250000$ entries in total.

Part Three

The output of print_gradients() is shown below:

```
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] 0.25376638738156426
param_gradient.hid_bias[20] -0.03326739163635369
param_gradient.output_bias[0] -2.062759603217304
param_gradient.output_bias[1] 0.03902008573921689
param_gradient.output_bias[2] -0.7561537928318482
param_gradient.output_bias[3] 0.21235172051123635
```

Part Four

Question 1

The words I picked was 'city', 'of', 'new'. The prediction results from the model is as below:

city of new york Prob: 0.99311

city of new . Prob: 0.00066

city of new life Prob: 0.00050

city of new home Prob: 0.00040

city of new ? Prob: 0.00032

city of new world Prob: 0.00027

city of new year Prob: 0.00026

city of new music Prob: 0.00022

city of new to Prob: 0.00018

city of new one Prob: 0.00017

By using find occurrence function, we see that the only word in the training set followed by this 3-

gram is 'york', which makes 'city of new york'. The rest of the results with low probabilities seem somewhat reasonable. Therefore, the prediction results are quite sensible. An example where the model makes plausible predictions that did not appear in the training set is when we predict the 3-gram 'government', 'of', 'united', which did not appear in the training set, the model gives the following predictions:

government of united ? Prob: 0.27924

government of united . Prob: 0.14816

government of united people Prob: 0.04556

government of united states Prob: 0.04189

government of united , Prob: 0.03168

government of united have Prob: 0.02985

government of united are Prob: 0.02145

government of united time Prob: 0.01810

government of united one Prob: 0.01432

government of united children Prob: 0.01293

The tri-gram "government of united" did not occur in the training set.

The results are quite plausible, particularly the fourth prediction: 'government of united states'.

Question 2

We can see a cluster of words: 'she', 'he', 'I', 'we', 'they' and 'you'. These words are all pronouns. Their usage are quite similar. The tsne_plot_representation plot and the tsne_plot_GLoVE_representation plot have similar shape and they both make sensible clusters. Their clusters of words are somewhat different but the words are scattered out quite evenly, which is good. When we look at the 2 dimensional glove plot, the clusters seem to have a pattern, and the clusters aren't very sensible. Because 2 dimensional glove embedding does not capture the relationships of our data well enough, so the prediction is somewhat off compared to the neural model. (The 256 dimensional plot is done in the code file, with an added 256 dimension from part 1)

Question 3

The word distance between new and york is 3.731(rounded). As we can also see from the plot, the words 'new' and 'york' are not really close to each other. Although together the phrase 'new york' do show up in our data, the word 'new' can be used to describe a lot of things whereas 'york' is a single noun which isn't used very often, at least not in the same way as 'new'

Question 4

The word distance between government and university is 0.982(rounded) and the word distance between government and political is 1.262(rounded). Although government is usually linked with the word politics, our model learned the representation as government and university are both organizations, whereas political is an adjective word, the words are used in different context. Therefore the distance between government and university is closer.