csc421 a1

Mingren Chen

January 2019

1 Q1

The total trainable parameters are
1) word embedding weights is a 250×16 matrix
2) embed to hid weights is a $128 \times 16 \times 3$ matrix
3) hid to output weights is a 250×128 matrix
4) output bias is a 250×1 vector
5) hid bias is a 128×1 vector
in total 42522 parameters. And hid to output weights has most params

By using 4 gram model, we use previous 3 words to predict the fourth. So we use 250^4 entries since we have 250 words.

2 Q2

```
  ############################   YOUR CODE HERE   ##############################
  return output_activations - expanded_target_batch

  ########################################################################
hid_to_output_weights_grad = np.dot(loss_derivative.T, activations.hidden_layer)
output_bias_grad = np.sum(loss_derivative, 0)
embed_to_hid_weights_grad = np.dot(hid_deriv.T, activations.embedding_layer)
hid_bias_grad = np.sum(hid_deriv, 0)
```

```
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] -0.2546730202374652
param_gradient.word_embedding_weights[2, 5] 0.0

param_gradient.embed_to_hid_weights[10, 2] -0.6526990313918256
param_gradient.embed_to_hid_weights[15, 3] -0.13106433000472612
param_gradient.embed_to_hid_weights[30, 9] 0.11846774618169402
param_gradient.embed_to_hid_weights[35, 21] -0.10004526104604389

param_gradient.hid_bias[10] 0.2537663873815642
param_gradient.hid_bias[20] -0.03326739163635368

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
```

3 Q3

1. I use "city" "of" "new" as input and output is york with possibility of 98%. This is reasonable since new york appears in the sentences for several times. When predict words "good" "year" "for", the probability of "him" is 8% where good year for him is not in the raw sentence.
2. Words in the same cluster has the common of they are highly possible to replace each other in a sentence and the new sentence is usually Grammatically correct.
3. No. they are not close since they can not replace each other. One of them is adj. and another is Noun. But the combination appear in the sentence for 8 times so we can predict them easily.
4.I think the pair ("government", "university") are closer. My training result shows the distance of ("government", "university") is 1.05, the other one is 1.22. The reason is basicly university and government are both noun and there are same adj describing "government" and "university" in raw sentence so our algorithm think government and university are closer.