Deep Learning Homework 1

İpek Erdoğan,2019700174

May 3, 2021

Mathematical Expressions

Forward Propagation

Our inputs are one hot representations of our input words which have dimensions of [1x250]. Then, we multiply each of our 3 input words with weight matrix W1, to get the 16-dimensioned embeddings. After getting [1x16] dimensioned embeddings e1, e2 and e3; we concatenate them in one embedding vector e, which has dimension of [1x48]. Then, to convert this embedding into a 128-dimensioned hidden layer, we multiply this embedding with weight matrix W2 [48x128] (which is the concatenation of 3 weight matrices $W2_1$,$W2_2$ and $W2_3$) and add bias b1 [1x128]. Then, we pass the result through sigmoid fuction and multiply it with matrix W3, which has dimension of [128x250], and finally add bias b2 [1x250] to it. At the end, we get a [1x250] dimensioned output vector.

$$x = 1, 2, 3$$

$$word_x = \begin{bmatrix} 1 & \dots & 0 \end{bmatrix}_{1\times 250}$$

$$W1 = \begin{bmatrix} w1_{1,1} & \dots & w1_{1,16} \\ w1_{2,1} & \ddots & w1_{2,16} \\ w1_{250,1} & \dots & w1_{250,16} \end{bmatrix}_{250\times 16}$$

$$e_x = word_x \times W1 = \begin{bmatrix} e_{x1} & \dots & e_{x16} \end{bmatrix}_{1\times 16}$$

$$e = concat(e_1, e_2, e_3) = \begin{bmatrix} e_1 & \dots & e_{48} \end{bmatrix}_{1\times 48}$$

$$W2_x = \begin{bmatrix} w2_{x1,1} & \dots & w2_{x1,128} \\ w2_{x2,1} & \ddots & w2_{x2,128} \\ w2_{x16,1} & \dots & w2_{x16,128} \end{bmatrix}_{16\times 128}$$

$$W2 = concat(w2_1, w2_2, w2_3) = \begin{bmatrix} x_{1,1} & \dots & x_{1,128} \\ x_{2,1} & \ddots & x_{2,128} \\ x_{48,1} & \dots & x_{48,128} \end{bmatrix}_{48\times 128}$$

$$b_1 = \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

$$h = e \times W2 + b_1 = \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

$$h_{activated} = \sigma(h) = \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

$$W3 = \begin{bmatrix} x_{1,1} & \dots & x_{1,250} \\ x_{2,1} & \ddots & x_{2,250} \\ x_{128,1} & \dots & x_{128,250} \end{bmatrix}_{128\times 250}$$

$$b_2 = \begin{bmatrix} x_1 & \dots & x_{250} \end{bmatrix}_{1\times 250}$$

$$output = h_{activated} \times W3 + b_2 = \begin{bmatrix} x_1 & \dots & x_{250} \end{bmatrix}_{1\times 250}$$

Backward Propagation

Derivatives of elements

$$\frac{\partial Loss}{\partial output} = prediction - target$$

$$\frac{\partial output}{\partial h_a ctivated} = W_3^T$$

$$\frac{\partial h_a ctivated}{\partial h} = \sigma(h) \odot (1 - \sigma(h))$$

$$\frac{\partial h}{\partial e_x} = W_{2x}^T$$

$$\frac{\partial e_x}{\partial W_1} = word_x^T$$

$$\frac{\partial h}{\partial W_2} = e^T$$

$$\frac{\partial output}{\partial W_3} = h_{activated}^T$$

$$\frac{\partial h}{\partial b_1} = 1$$

$$\frac{\partial output}{\partial b_2} = 1$$

Gradients

$$\nabla(W_1) = \sum_{x=1}^{3} \frac{\partial Loss}{\partial output} \frac{\partial output}{\partial h_a ctivated} \frac{\partial h_a ctivated}{\partial h} \frac{\partial h}{\partial e_x} \frac{\partial e_x}{\partial W_1}$$

$$\nabla(W_1(part1)) = \begin{bmatrix} x_1 & \dots & x_{250} \end{bmatrix}_{1\times 250} \times \begin{bmatrix} x_{1,1} & \dots & x_{1,128} \\ x_{2,1} & \ddots & x_{2,128} \\ x_{250,1} & \dots & x_{250,128} \end{bmatrix}_{250\times 128} \odot \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128} \times \begin{bmatrix} w2T_{x1,1} & \dots & w2T_{x1,16} \\ w2T_{x2,1} & \ddots & w2T_{x2,16} \\ w2T_{x128,1} & \dots & w2T_{x128,16} \end{bmatrix}_{128\times 16}$$

The equation above includes the first four elements of the formula and it results with dimension of [1x16]. Last element of the formula has the dimension of [250x1], so I included it to multiplication from the left side, to be able to make the matrix multiplication and get a [250,16] dimensioned result. (To update W1, 3 different gradients for 3 different words have been calculated and summed)

$$\nabla(W_1) = \sum_{x=1}^{3} \begin{bmatrix} 1 \\ \dots \\ 0 \end{bmatrix}_{250\times 1} \times \begin{bmatrix} x_1 & \dots & x_{16} \end{bmatrix}_{1\times 16}$$

$$\nabla(W_2) = \frac{\partial Loss}{\partial output} \frac{\partial output}{\partial h_a ctivated} \frac{\partial h_a ctivated}{\partial h} \frac{\partial h}{\partial W_2}$$

$$\nabla(W_2(part1)) = \begin{bmatrix} x_1 & \dots & x_{250} \end{bmatrix}_{1\times 250} \times \begin{bmatrix} x_{1,1} & \dots & x_{1,128} \\ x_{2,1} & \ddots & x_{2,128} \\ x_{250,1} & \dots & x_{250,128} \end{bmatrix}_{250\times 128} \odot \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

The equation above includes the first three elements of the formula and it results with dimension of [1x128]. The fourth(last) element of the formula has the dimension of [48x1], so I included it to multiplication from the left side, to be able to make the matrix multiplication and get a [48,128] dimensioned result. (This result have been splitted into 3 parts to update $W2_1, W2_2, W2_3$ respectively.)

$$\nabla(W_2) = \begin{bmatrix} e_1 \\ \dots \\ e_{48} \end{bmatrix}_{48\times 1} \times \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

$$\nabla(W_3) = \frac{\partial Loss}{\partial output} \frac{\partial output}{\partial W_3}$$

In the equation above, first element of the formula has the dimension of [1x250] and second element of the formula has the dimension of [128x1]. So I included the second one to multiplication from the left side, to be able to make the matrix multiplication and get a [128,250] dimensioned result.

$$\nabla(W_3) = \begin{bmatrix} x_1 \\ \dots \\ x_{128} \end{bmatrix}_{128\times 1} \times \begin{bmatrix} x_1 \dots x_{250} \end{bmatrix}_{1\times 250}$$

$$\nabla(b_1) = \frac{\partial Loss}{\partial output}\frac{\partial output}{\partial h_activated}\frac{\partial h_activated}{\partial h}\frac{\partial h}{\partial b_1}$$

$$\nabla(b_1) = \begin{bmatrix} x_1 & \dots & x_{250} \end{bmatrix}_{1\times 250} \times \begin{bmatrix} x_{1,1} & \dots & x_{1,128} \\ x_{2,1} & \ddots & x_{2,128} \\ x_{250,1} & \dots & x_{250,128} \end{bmatrix}_{250\times 128} \odot \begin{bmatrix} x_1 & \dots & x_{128} \end{bmatrix}_{1\times 128}$$

$$\nabla(b_2) = \frac{\partial Loss}{\partial output}\frac{\partial output}{\partial b_2}$$

For the gradients of biases, last elements' derivatives are 1 since biases are adding to the hidden layer and output.

Results

I have initialized all of the weights and biases with random sampling from a normal distribution with mean=0 and variance =0.5. You can see the training and validation results from epoch=0 and epoch=200 (with learning rate=0.001).

EPOCH 0
Epoch training loss: 9.665354304072332
Epoch training accuracy: 16.4048322147651
Epoch validation loss: 9.654757931201589
Epoch validation accuracy: 17.363440860215054

EPOCH 200
Epoch training loss: 9.607568527766173
Epoch training accuracy: 23.440805369127517
Epoch validation loss: 9.604091038294564
Epoch validation accuracy: 23.230107526881721

After the training phase, I have loaded the weights from 200.epoch's model and tested it. You can see the results down below.

Test loss: 9.604344681574869
Test accuracy: 23.116129032258065

t-SNE Plotting

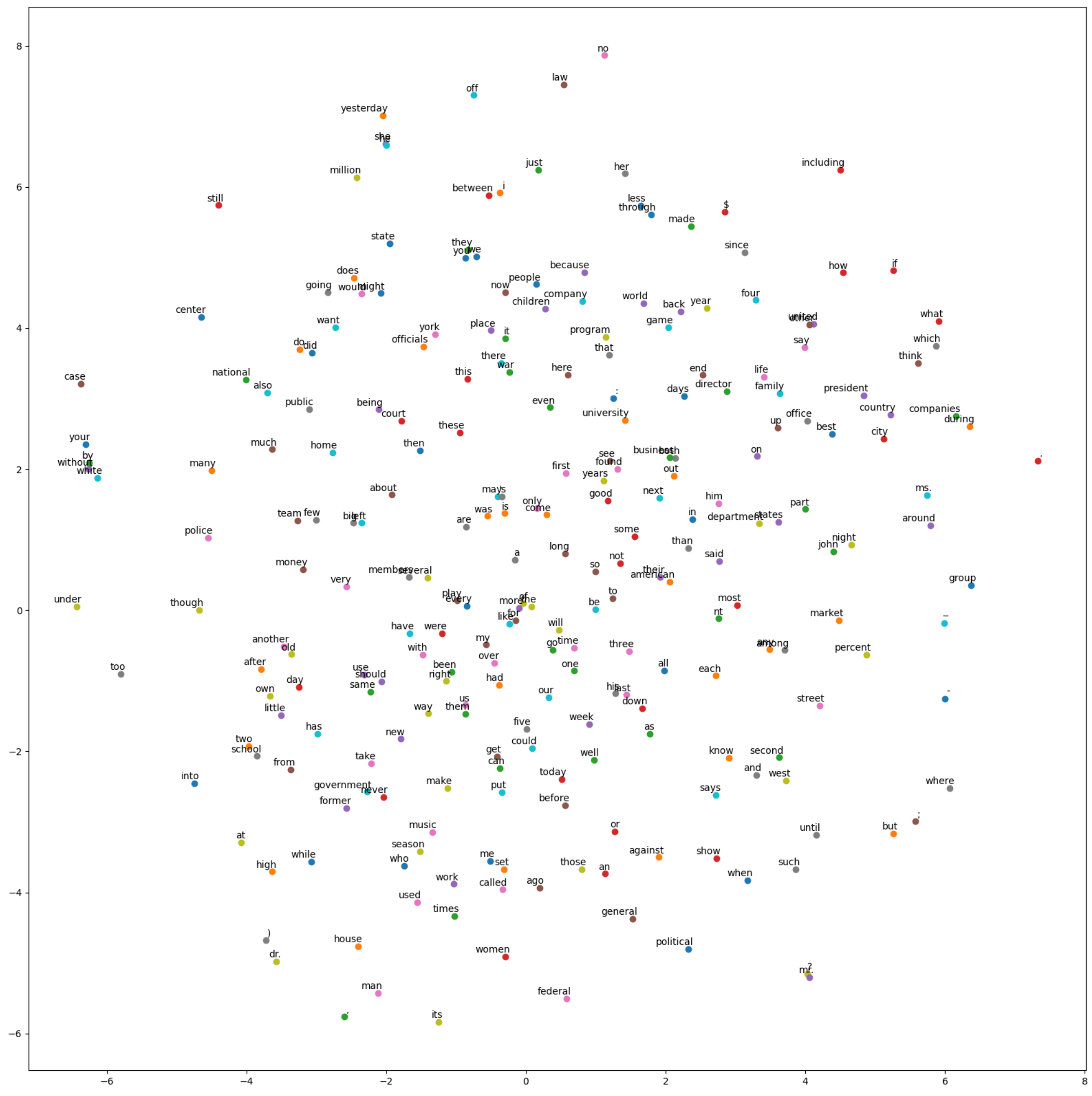

Figure 1: Map of the embeddings

Normally, if the encoder was successfull and well-trained, we would get successful representations and by clustering these representations, we would get meaningful (context-related) clusters. Yet, due to lack of my model's performance and low learning rate, my embeddings weren't that successful to result with good clusters. Still we can see ”what” and ”which”, ”do” and ”did”, ”these” and ”this” words are near to each other.

For the following data points ’city of new’, ’life in the’, ’he is the’; my model didn’t give that much sensible predictions. That might because of our dataset is unbalanced. There were lots of ”.”

Model predicted the next word for sample ”city of new” [133,17,84] as ”.” [143]
Model predicted the next word for sample ”life in the” [156,200,248] as ”.” [143]
Model predicted the next word for sample ”he is the” [169,191,248] as ”it” [192]

Additional Notes

Softmax have been implemented in the loss function and the derivative of the loss function and softmax have been calculated together. Stable softmax have been used to get rid of the softmax underflow and overflow.

Lecture slides, Stanford University CS231n lecture resource [1], ”Deep Learning” book [2] and some additional websites [3], [4] have been used during this study. For plotting the tSNE graph, a ready-to-go code has been used [5].

References

[1] “CS231n: Convolutional Neural Networks for Visual Recognition”, `https://cs231n.github.io/optimization-2/#mat`.

[2] Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[3] “Classification and Loss Evaluation - Softmax and Cross Entropy Loss”, `https://deepnotes.io/softmax-crossentropy`.

[4] “A Gentle Introduction to Cross-Entropy Loss Function”, `https://sefiks.com/2017/12/17/a-gentle-introduction-to-cross-entropy-loss-function/`.

[5] “Visualizing Word Vectors with t-SNE”, `https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne`.