CSCE 636: Deep Learning

HW 4: Fall 2021

1.

a.

Given,

Dimension of word embedding = d
Size of vocabulary = |V|
Number of hidden units = D.

We have,

- Dimension of L = |V| x d
- Dimension of H = D x D
- Dimension of I = d x D
- Dimension of b_1 = D
- Dimension of U = D x |V|
- Dimension of b_2 = |V|

b. We know that, $Cross\ Entropy\ (CE) = -\sum_{j=1}^{|V|} y_j{}^{(t)} log(y'_j{}^{(t)})$

where, $y_j{}^{(t)}$is the ground truth, and $y'_j{}^{(t)}$is the prediction for sample j. We know that $y_j{}^{(t)}$can be 1 only for one of the |V| samples, and for all other samples it would be 0. Hence, the cross-entropy equation reduces to:

$$CE = -y_j{}^{(t)}.log\left(y'_j{}^{(t)}\right) = -log\left(y'_j{}^{(t)}\right)$$, where j is the sample for which $y_j{}^{(t)}$is 1. (1)

$CE = -y_j{}^{(t)}.log\left(y'_j{}^{(t)}\right) = -log\left(y'_j{}^{(t)}\right)$, where j is the sample for which $y_j{}^{(t)}$is 1.

From the question, we have that perplexity is:

$$PP^{(t)}\left(y^T, y'_j{}^{(t)}\right) = \frac{1}{\sum_{j=1}^{|V|} y_j{}^{(t)} y'_j{}^{(t)}}$$

Now, here as well, $y_j{}^{(t)}$can be 1 only for one of the |V| samples, and for all other samples it would be 0. Therefore, the perplexity equation reduces to:

$$PP^{(t)} = \frac{1}{y_j{}^{(t)}.y'_j{}^{(t)}} = \frac{1}{y'_j{}^{(t)}}$$, where j is the sample for which $y_j{}^{(t)}$is 1. (2)

From (1), we have:

$$CE = -log\left(y'_j{}^{(t)}\right) = log\left(\frac{1}{y'_j{}^{(t)}}\right)$$

Raising both sides by exponential, we have:

From (2)

$$exp^{CE} = exp^{log\left(\frac{1}{y'_j{}^{(t)}}\right)} = \frac{1}{y'_j{}^{(t)}} = PP$$

Therefore, $PP = exp^{CE}$

This is the relation between Cross-entropy and Perplexity.

c. Hyperparameters:

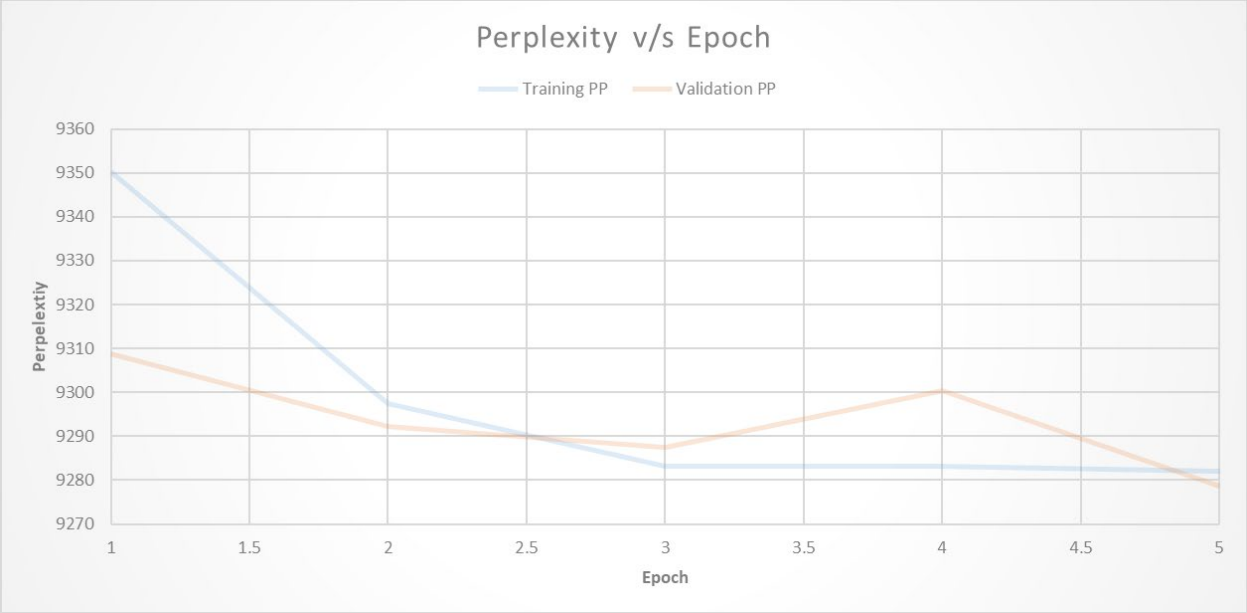
(You will find the completed code in the attached python files)

Hyperparameters	Values	Test Perplexity
batch_size	64	9466.330078125
embed_size	500	
hidden_size	250	
num_steps	20	
max_epochs	5	
early_stopping	5	
dropout	0.3	
lr	0.01	
Optimizer	SGD	
batch_size	64	9437.9169921875

embed_size	500	
hidden_size	250	
num_steps	30	
max_epochs	5	
early_stopping	5	
dropout	0.3	
lr	0.01	
Optimizer	SGD	
batch_size	64	9467.8115234375
embed_size	1000	
hidden_size	500	
num_steps	30	
max_epochs	5	
early_stopping	5	
dropout	0.3	
lr	0.01	
Optimizer	SGD	
batch_size	64	9466.330078125
embed_size	1000	
hidden_size	500	

num_steps	20	
max_epochs	5	
early_stopping	5	
dropout	0.3	
lr	0.01	
optimizer	SGD	
batch_size	64	9225.990234375
embed_size	50	
hidden_size	100	
num_steps	10	
max_epochs	5	
early_stopping	5	
dropout	0.3	
lr	0.01	
optimizer	AdamW	

Training and Validation Perplexity v/s Epoch for the best combination



Text Generated:

in palo alto restricting provoked dealt blacks team roper mo. crusaders notify sunday expertise stuff kept primarily notorious miss terms minute owners diversified conviction secondary deposit lengthy preparation audio enterprise parent heights midst erbamont translate buyer strikes sierra segment passes quoted unidentified journalism terms with fully garratt oldest fired recruit competitors grenfell seller integration portfolio proof dd runaway change importing casual approve multiples wealthy roll cases turnaround problem sagging dire j.c. flavor eidsmo kia benefited farms hospital breed stress-related sim narrowed takeover-stock st. incredible question telerate reducing fujitsu yields predictable zealand retinoblastoma newspapers windsor tim forfeiture massachusetts hollywood evaluation him retreated father instrumental

Discussion: I understand that the perplexity I got is very high, and it is not ideal. I tried a lot of things to improve the score, like changing hyperparameters, increasing the number of epochs (the perplexity gets stagnant after a certain number of epochs). I had even tried defining the model in two ways: first using nn.Linear for all the matrix multiplications (*RNNLM.py*) and the second, defining empty matrices and using torch.matmul for matrix multiplication (*RNNLM_orig.py*). Both resulted in similar results. In the end, I didn;t have enough time to get to the root cause of the problem (it's the semester end). If I had more time, I would have been able to identify the problem.

2.

a. Considering two cases:
   - ***Case 1***: Q = K = V = X (Self Attention). In that case Q, K and V are not learnable. And only W^Q, W^K, W^V are learnable. Hence the number of parameters in either case would be:
     i. *Single-head attention*: (d x d) + (d x d) + (d x d) = **3 x d x d**
     ii. *Multi-head attention*: {(d x d/h) + (d x d/h) + (d x d/h)} x h = **3 x d x d**
   - ***Case 2***: Q, K and V are also learnable. Hence the number of parameters in either case would be:

i. *Single-head attention*: (n x d) + (d x d) + (n x d) + (d x d) + (n x d) + (d x d) = **3 x {(n x d) + (d x d)}**
ii. *Multi-head attention*: (n x d) + (n x d) + (n x d) + {(d x d/h) + (d x d/h) + (d x d/h)} x h = **3 x {(n x d) + (d x d)}**

b. Considering single-head and multi-head separately, we have:

- *Single-head attention*:
  i. For the 3 linear transformations: (n x d) matrix is multiplied with (d x d) matrix. So there would be three for loops (1 to n, 1 to d and 1 to d). Therefore, the complexity for that would be: $O(nd^2)$
  ii. For Softmax: Two matrices of sizes n x d are being multiplied (one being transposed to d x n). Therefore, again there would be three for loops (1 to n, 1 to d and 1 to n). Therefore, the complexity for that would be: $O(n^2d)$
  iii. For final multiplication (after Softmax): Two matrices of sizes n x d are being multiplied (one being transposed to d x n). Therefore, again there would be three for loops (1 to n, 1 to d and 1 to n). Therefore, the complexity for that would be: $O(n^2d)$

  Total complexity = 3* $O(nd^2)$ + $O(n^2d)$ + $O(n^2d)$ = **$O(nd^2)$ + $O(n^2d)$**
- *Multi-head attention*:
  i. For the 3 linear transformations: (n x d) matrix is multiplied with (d x d/h) matrix. So there would be three for loops (1 to n, 1 to d and 1 to d/h). Therefore, the complexity for that would be: O(nd.d/h)
  ii. For Softmax: Two matrices of sizes n x d/h are being multiplied (one being transposed to d/h x n). Therefore, again there would be three for loops (1 to n, 1 to d/h and 1 to n). Therefore, the complexity for that would be: $O(n^2d/h)$.
  iii. For final multiplication (after Softmax): Two matrices of sizes n x d/h are being multiplied (one being transposed to d/h x n). Therefore, again there would be three for loops (1 to n, 1 to d/h and 1 to n). Therefore, the complexity for that would be: $O(n^2d/h)$

  Therefore, total complexity = h * (3 * O(nd.d/h) + $O(n^2d/h)$ + $O(n^2d/h)$) = h * (O(nd.d/h) + $O(n^2d/h)$) = **$O(nd^2)$ + $O(n^2d)$**

3. a.

The limitation can be solved by adding an identity matrix (I) to A: A is substituted by A`, such **A` = A + I**.

b.

The limitation can be solved by substituting A with A``, such that **A`` = D`$^{(-1/2)}$A` D`$^{(-1/2)}$**, where **D` is the diagonal node degree matrix of A`, and A` = A + I**.