NLP Assignment 5 : Neural Language Modelling

180128022

April 2019

1 Introduction

In this assignment, a neural language model has been implemented.

2 Implementation

The new implementation has been done on the provided code, while making the following changes:

1. **forward**: In the class, ***NGramLanguageModeler***, the function, **forward**, has been tweaked a little to return **embeddings** along with **log_probs**, which is used in the subsequent stages.

2. Along with the above, there have been many changes done in the code to suit the requirements. New code modules have been written for:

- Model Description
- Sanity Check
- Test

3 Model Description

The neural network model is almost like a multilayer perceptron with one input layer, one hidden layer and one output layer, where the output of the first layer is the input for the next and so on. The following table summarises the data (although the shapes have been printed for each layer, they have been commented out in the code to make the output a little precise and stop them from being printed multiple times as they go through loops).

Layers	Dimension
Input Layer	(1 x 20)
Hidden Layer	(1 x 128)
Output Layer	(1 x 17)

In the code, EMBEDDING_DIM = 10 and CONTEXT_SIZE = 2. Multiplying them and passing them as the *in_features* argument of TORCH.NN.MODULES.LINEAR, we get the dimension for the input layer. Similarly, there are 128 neurons in the hidden layer and 17 ω vectors in the output layer (there are 17 unique words in the 5 given sentences, and they constitute the word_to_ix dictionary) which give them the dimensions mentioned. The mathematical equations are given by:

1. $h_j(\mathbf{x}, \theta, b_j) = R(\sum(x_i\theta_i) + \mathrm{b}_j)$

R = Relu Function (Relu Function is used in the code as the activation function),
i = 1, 2, ..., 19, 20 (i gives the number of neurons in the input layer) and j = 1, 2, ..., 127, 128 (j denotes the number of neurons in the hidden layer). Finally, these 128 neurons are mapped to 17 neurons in the output layer using Softmax Activation function.

2. $f_k(\mathbf{x}, \theta, \omega_k, \mathbf{b}, c_k) = S(\sum(\omega_{kj}h_j(\mathbf{x}, \theta, b_j) + c_k))$

S = Softmax Activation Function, j = 1, 2, ..., 127, 128 and k = 1, 2, ..., 16, 17 (it gives the number of neurons in the output layer, so ω_{kj} gives a measure of each of the 17 ω(s) having the dimension of 1 x 128).

4 Sanity Check

For the sanity check, the sentence, "The mathematician ran to the store ." has been used as the target sentence. Having done the pre-processing, that is, creating a list of the 5 sentences, where each sentence is represented as a list of tokens, and tweaking the hyper-parameters like learning-rate and epochs, the model has been found to predict accurately 5 consecutive times. The model predicts for the context "START The" the word "mathematician" instead of "physicist" because in the bigram, "The mathematician" occurs more number of times (3 times) than "The physicist" (1 time) in the training data sentences.

5 Test

The test to predict the correct word between "physicist" and "philosopher" for the sentence, "The ------ solved the open problem. ", gives the result as "physicist". This is observed by changing learning rate to 0.01 and epochs to 20.

This would **not** be possible with the Lab 2 Bigram ML model from Lab 2 because the probabilities for bigrams "the physicist" and "the philosopher" would be 1, whereas, the probabilities for bigrams "physicist solved" and "philosopher solved" would be 0. Hence, the results will not be accurate since both the probabilities are the same.

The model is predicting correctly for the right reasons - the embeddings for "physicist" and "mathematician" are closer together than the embeddings for "philosopher" and "mathematician". Using **nn.CosineSimilarity** it is seen that the cosine similarity value between "physicist" and "mathematician" is much higher (0.28) than that of "philosopher" and "mathematician" (-0.14).