Neural Language Model

Vijeta Agrawal

29^{th} *April* 2019

1 Aim

The main focus of this report is to show the results of the implementation of a neural language Model. Modify word_embeddings_tutorial.py to run a sanity check and test the model to report finding.

2 Introduction and Background

Language Modeling (LM) is one of the most important parts of modern Natural Language Processing (NLP). There are many sorts of applications for Language Modeling, like: Machine Translation, Spell Correction Speech Recognition, Summarization, Question Answering, Sentiment analysis etc. Each of those tasks require use of language model. Language model is required to represent the text to a form understandable from the machine point of view **chromiak˙2017**

3 Neural Network Language Model

The goal of this lab is to give description of a neural network language model. We use the language model to make predictions using some toy data and tuning the hyper parameters.

The below table shows the dimensionality of Input Layer, Output Layer and Hidden Layer respectively. The model has one hidden layer.

Layer	Dimension
Input Layer	1 x 20
Output Layer	1 x 128
Hidden Layer	1 x 17

Table 1:

The mathematical equations to describe the model are given below:

Number of Embeddings = 10
Number of Context words = 2
Input Layer Dimensionality = 10*2 = 20
For training, the inputs are denoted in the form of vector **x** of length 20.

Neurons in Hidden Layer = 128
Because of this, vector **x** is passed to the activation function, called Relu, denoted by R, along with hyper-parameters. The equation for i^{th} neuron's output value is given as:

$$Hid_i(\mathbf{x}, \theta_i, b_i) = R_i(x_1 * \theta_{i1} + x_2 * \theta_{i2} + \cdots + x_j * \theta_{ij} + \cdots + x_{20} * \theta_{i20} + b_i)(1)$$

where parameter θ_{ij} represents the jth input, ith neuron and i = 1,2,..,128 and j = 1,2,..,20. So, each θ vector has the dimension 1 x 20.
Number of θ vectors is 128.
The 128 hidden neurons is mapped to 17 output neurons with softmax activation function, which is denoted by S.

$$Out_k(\mathbf{x}, \Theta, \omega_k, \mathbf{b}, c_k) = S_k(\omega_{k1} * Hid_1(\mathbf{x}, \theta_1, b_1) + \omega_{k2} * Hid_2(\mathbf{x}, \theta_2, b_2) + \cdots + \omega_{ki} * Hid_i(\mathbf{x}, \theta_i, b_i)(2)$$

$$+ \cdots + \omega_{k1}7 * Hid_1(\mathbf{x}, \theta_1 7, b_1 7 + c_k)(3)$$

where ω_{ki} represents the i^th hidden neuron, k^th output neuron and i = 1,2,..,128 and k = 1,2,..,17. Each ω vector has the dimension 1 x 128.
The number of ω vectors is 17.
The resulting layer can be interpreted as log probabilities and so, can be used to make predictions, because each neuron represents a word in the vocabulary (17 words).

4 Sanity check

After preprocessing the toy data, we modify the given python script to run on the given 5 sentences. To get the correct prediction 5 consecutive times, I am using the following hyperparametres: Target Sentence: "The mathematician ran to the store."

1. epochs = 25
2. Learning rate $\lambda = 0.02$

Since, the training set contains more occurances of the bigram "The mathematician" than the bigram "The physicist" for the context "start the", it predicts "mathematician" correctly 5 consecutive times.

5 Sentence completion

Target Sentence = The _____ solved the open problem.
Candidate Words = "physicist" or "philosopher".
The task is to predict which of the candidate words is the most likely to complete the target sentence.
After setting the hyperparameters as:

1. epochs = 20
2. Learning rate $\lambda = 0.01$

The model is predicting the correct candidate word "physicist".
Yes, the result makes sense because after calculating the cosine similarity we can see in the results section that the embeddings for "physicist" and "mathematician" are closer together than the embeddings for "philosopher" and "mathematician".

6 Results

```
********** Sanity Check **********

********** 5 Consecutive Run Training Data Prediction **********
(['START', 'The'], 'mathematician')
(['The', 'mathematician'], 'ran')
(['mathematician', 'ran'], 'to')
(['ran', 'to'], 'the')
(['to', 'the'], 'store')
(['the', 'store'], '.')
(['store', '.'], 'STOP')
Epoch 1 : 1

(['START', 'The'], 'mathematician')
(['The', 'mathematician'], 'ran')
(['mathematician', 'ran'], 'to')
(['ran', 'to'], 'the')
(['to', 'the'], 'store')
(['the', 'store'], '.')
(['store', '.'], 'STOP')
Epoch 2 : 1

(['START', 'The'], 'mathematician')
(['The', 'mathematician'], 'ran')
(['mathematician', 'ran'], 'to')
(['ran', 'to'], 'the')
(['to', 'the'], 'store')
(['the', 'store'], '.')
(['store', '.'], 'STOP')
Epoch 3 : 1

(['START', 'The'], 'mathematician')
(['The', 'mathematician'], 'ran')
(['mathematician', 'ran'], 'to')
(['ran', 'to'], 'the')
(['to', 'the'], 'store')
(['the', 'store'], '.')
(['store', '.'], 'STOP')
Epoch 4 : 1

(['START', 'The'], 'mathematician')
(['The', 'mathematician'], 'ran')
(['mathematician', 'ran'], 'to')
(['ran', 'to'], 'the')
(['to', 'the'], 'store')
(['the', 'store'], '.')
(['store', '.'], 'STOP')
Epoch 5 : 1


********** Prediction **********

('The', 'physicist', 'solved the open problem.')

********** Calculating Cosine Similarity **********

physicist and mathematician : 0.195
philosopher and mathematician : 0.166
```

Figure 1: Code Output