Text multi-classification based on SoftMax Regression

Tianyi Xiong

In this project, I aim to do some other semantic classification methods since we have used lots of semantic classification method in our programming assignments this semester. So, I choose to use Bag-of-Word model and N-gram model. Bag-of-Word is a kind of non-sequence model. In this model, we do not focus on the sequence of the sentence, it only cares about the frequency of each word. But N-gram model cares about the sequence of the sentence, it uses one-hot encoding to record the position of each word in the sentence. In this program, the second task is to compare which one is better or how much better.

For the database documents, I download the movie review document from the Rotten Tomatoes database, basically, it is about sentiment Analysis on movie review, each sentence has a label which represents the sentiment of this sentence (use number 0 to 5, and 0 represents negative and 5 represents positive). Here is the database link: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/overview/description

In 'feature_extraction.py', I designed Bag-of-Word method and N-gram method. In the Bag-of-Word method, the matrix of bag-of-word-feature consists with the frequency of the words, which means, in the Bag of word method I only record the word frequency. But in the N-gram method, I used one-hot encoding to record the position of each word. For example: 'I Love you' and 'Do you love yourself' will be parsed into two vectors and a dictionary together.:

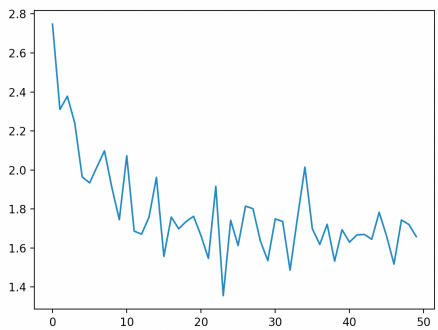
{'I': 0, 'love': 1, 'you': 2, 'do': 3, 'yourself': 4, 'I_love': 5, 'love_you': 6, 'do_you': 7, 'you_love': 8, 'love_yourself': 9}
[[1. 1. 1. 0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 1. 0. 0. 1. 1. 1.]]

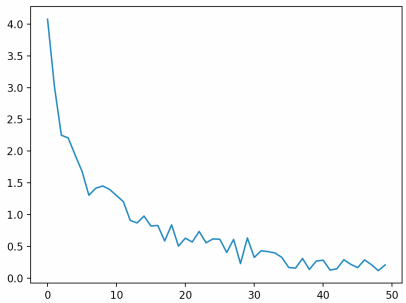
For the sentence 'I love you': I, love, you appear in the first, second, third position of the dictionary. And 'I love' is in the sixth of the dictionary, by parity of reasoning.

So, N-gram model will record each unique word and put all these words from different sentence into one dictionary, then it will also record the sequence of the sentence, put the combination of words into this dictionary. Then, using one-hot encoding to represent each sentence.

Activation function which I used is SoftMax function. The weights in the matrix would be compressed from 0 to 1. Here is the strategy of SoftMax: $P(y=j)=\frac{e^{x^T W_j}}{\sum_{k=1}^{K} e^{x^T W_k}}$, and probability is equal to softmax (X dot weight). Loss -=np.log(y(index)).

The results are:





The first diagram is the loss of Bag of word model, and the second is the loss of N-gram model. And the diagram below is the accuracy of these two models, we can find that N-gram model has extremely high accuracy in the train function, and it is also better than Bag-of-word model in the test function. In my opinion, the reason of N-gram model is not as good as I consider is that the matrix is overfitting. If there is no such combination of a sentence in the test function, it will not predict successfully. But finally we can easily observe that N-gram model is significantly better than BOW model.

```
Bow shape (1000, 363)
Gram shape (1000, 968)
epoch 0 loss [3.13019358]
epoch 5 loss [2.14278416]
epoch 10 loss [1.91390893]
epoch 15 loss [1.74432826]
epoch 20 loss [1.77289886]
epoch 25 loss [1.64352788]
epoch 30 loss [1.75078924]
epoch 35 loss [1.67570604]
epoch 40 loss [1.6531951]
epoch 45 loss [1.67610332]
Bow train 0.8 test 0.42
epoch 0 loss [4.23303218]
epoch 5 loss [1.95820423]
epoch 10 loss [1.19474836]
epoch 15 loss [0.83094318]
epoch 20 loss [0.81764636]
epoch 25 loss [0.56287381]
epoch 30 loss [0.26977637]
epoch 35 loss [0.25467642]
epoch 40 loss [0.18955982]
epoch 45 loss [0.04168279]
Gram train 0.9933333333333333 test 0.48
```