Continuous Bag of Words

Nate Van

May 2023

Table of Contents

Motivation

Continouous Bag of Words (CBOW) catches my interest when I was introduced to Word2Vec using skip gram method. The Word2Vec is an natural language processing technique using neural network model to train word embedding. With the embedding vector, the Word2Vec can represent word semantics or the meaning of words. From a different point of view, human was able to recognize words meaning by the characters. However, words embedding is a phenomenon recognize words by numbers. Even though I have learned about skip gram, I want to further my understanding of Word2Vec. Thus, I am motivated to learn CBOW.

Dataset Description

The data that I'm using is from the Python library 'dataset' named 'CC-News.' The data includes news gathered all around the world, and the language is in English. In this work, I pull 150k documents from the data for training the words embedding.

System Design

Before training model, the documents need to be process into train token and word represent in the numbers. In the text processing phase, each document is tokenize into useful tokens, which remove stopword, lemmatization, and remove duplication. With all documents tokenized, the top most common words is selected as the training data. In this case, the most common words is the vocab size. Each word in the set of training data is represented in indices, which also the size of vocabulary. Because of limited resource, the CBOW vocab size $|V|$ is 10000. Now, the model is ready to train.
The CBOW uses a window method where the target word is the central word and the rest are the context. In another word, the context words and target word are the input data and output label correspondingly. The window slides one step at a time and get the context words and target word from the training data. For this design, the window size is 5.
The words embedding has dimension N of 50 for a small size of $|V|$. The CBOW neural network only consists of input layer, a hidden layer, and output layer and also two weights W and W' as shown on figure 1. In the input layer, it is the input of 4 words in one hot vector, $|V|$x4 vector. Next, the hidden contains the average of the input words embedding, Nx1 vector.

Lastly, the output layer transforms embedding into probability of each word, usually using softmax equation 1, in the vocabulary, $|V|$x1. Thus, feed forwarding the inputs will result in a single one hot vector that is the predicted word. The loss is calculated using the predicted word and the target word. Typically, the loss function is the cross entropy loss equation 2. To train the W and W', back propagation is used (this is beyond the scope of this project).

$$a(x_i) = \frac{e_i^x}{\sum_{j=0}^{|V|-1} e_j^x} \tag{1}$$

$$L = -\frac{1}{N} \sum_{i=0}^{N-1} (a(x_i) - \sum_{j=0}^{|V|-1} log(a(x_j))) \text{ *N is the batch size*} \tag{2}$$

One of the biggest concern about machine learning is the computation time to training the model. For CBOW, the complexity is costly as it's the summation through the whole vocabulary. To reduce the time, I used negative sampling method to calculate the loss. Instead, of the size of $|V|$, the summation reduce to number of the negative sample words, which I choose 5 words. The negative sample words are randomly selected words in the vocabulary. Also, sigmoid is used instead of exponential for the activation function. The final equation is shown on equation 3. To reduce time further, Adam optimizer is introduced, which is a faster optimizer for stochastic gradient descent, with 0.001 learning rate, 5 epochs, and batch size of 50.

$$L = -\frac{1}{N} \sum_{i=0}^{N-1} (log(\sigma(x_i)) - \sum_{j=0}^{4} log(\sigma(x_j))) \tag{3}$$

Evaluation

To evaluate the quality of the CBOW, I used cosine similar between words to determine the distance between words. The closer it is to 1 the smaller the distance between words, and it holds true for the opposite. A dataset containing pairs of similar questions is used to evaluate. For each words, if it is in the vocabulary, the score is calculated by averaging the embedding of the words in the questions then calculated the cosine similar. With the trained embedding, the score is is 0.5724, which is not a quality model.

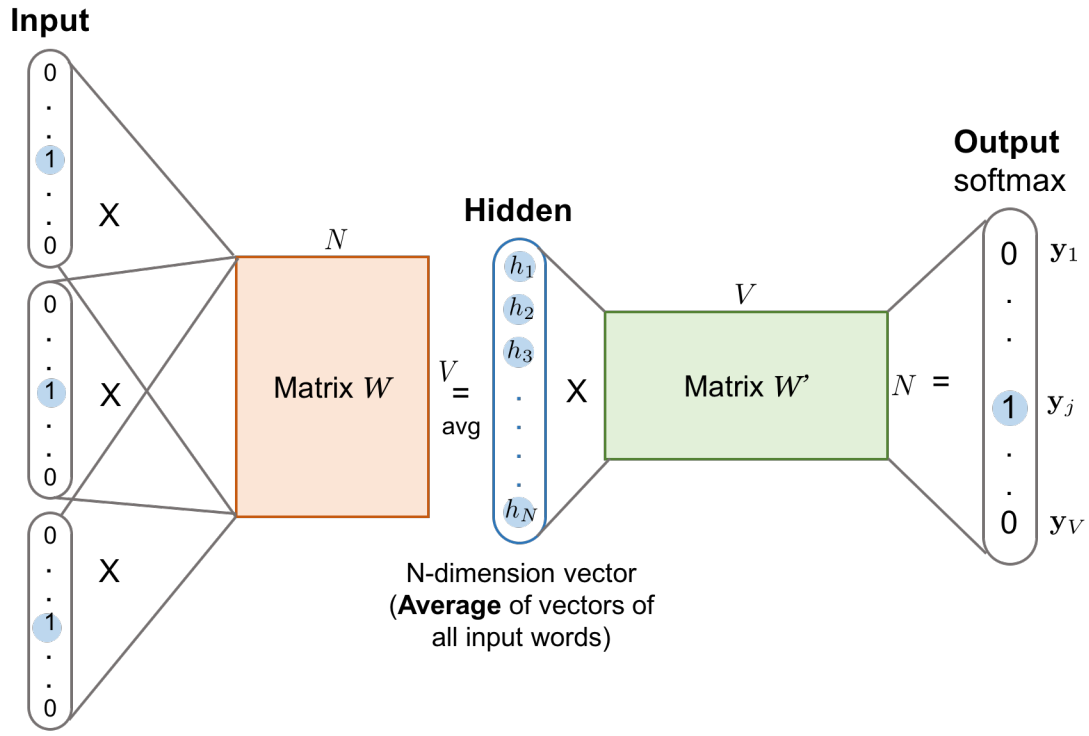

Figure 1: CBOW Model

For comparison, a words embedding that randomly choose using a uniform distribution has a score of 0.3518. The trained embedding does yield better result, however, not by much. The reason maybe because of small size of vocabulary.

Conclusion

Overall, the objective for this project was to further my understand of Word2Vec. I learn that each of the embedding vector is the parameters to represent a word. To train the word embedding, I learn the infrastructure of the Neural Network that is the input layer, hidden, and output layer. To calculate the loss, I learn the softmax function, cross entropy, and most importantly negative sampling.