Homework 3

COMS 4705 Fall 2017
Prof. Kathleen McKeown

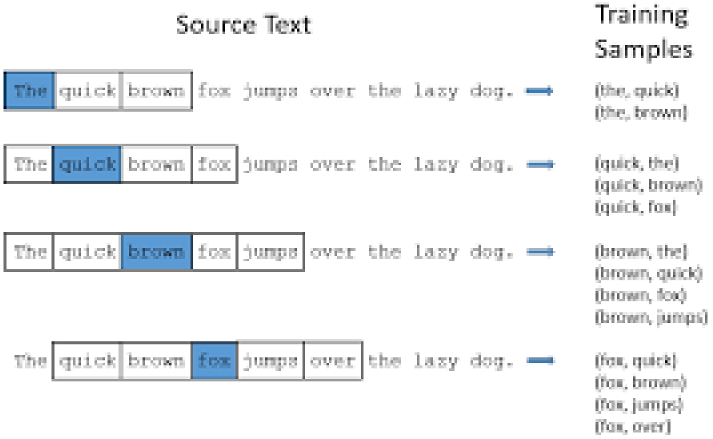
The assignment consists of a programming part and a written part. For the programming part, make sure you have set up the development environment as mentioned in the Prerequisites file. Then download the source code and extract it in the same directory as the embedding file(as mentioned in the Prerequisites) . The rest of the details for the programming assigment can be found inside the 'hw3.ipynb' file.

The written portion of the assignment will discuss a neural network model that produces word embeddings. For understanding word embeddings we suggest the following readings : class slides and textbook chapter 10. The appendix in this document also has a high level overview. You may notice different resources using different notation. The concepts remain the same. For this assignment we will be using the notation as discussed in the section 'Skip-gram Neural Network architecture'. We use upper case letters to denote matrices and lower case letters to denote row/-column vectors.

1 Skip-gram models

The skip-gram neural network model produces word embeddings by reducing the task of the neural network to predicting the context words for a given input word.

Consider the sentence 'The quick brown fox jumped over the lazy dog'. For training data we define a window, let's say 2, and produce pairs of words looking within the window size on both sides of the current word.



Given the input word, we would like the probability of the context words to be higher than the rest of the words in the vocabulary. For every sample pair noted in the diagram above the input is the first word in the sample. The output label is the second word in the sample pair. The input and output words are converted to a one hot vector of size vocab_sizeX1.

Now that we have our training data let's look at the neural network architecture. It will be a simple feed forward network with one hidden layer. The hidden layer weights at the end of training will become our word embeddings!

1.1 Skip-gram Neural Network architecture

1. Input layer - The training samples are broken down into smaller batches each with *batch_size* number of samples. We will be passing one batch of samples for a single iteration to our

neural network. Once we complete the forward propogation and backward propogation, we pass in the next batch. Each one hot vector is $1Xvocab\_size$. We will stack the input one hot vectors of all the samples. We will feed into the network the transpose of the horizontally stacked one-hot vectors. The dimensions of the input, A_{in} to our neural network will be therefore be $vocab\_sizeXbatch\_size$.

1. Fully connected feed forward layer - The feed forward layer (or hidden layer) will have hd_1 number of nodes. The layer does a affine transformation and activation of the input values. A_{in} is the input to the layer and A_1 is the output of the layer. W_1 and b_1 are the weights associated with hidden layer.

$$Z_1 = W_1 A_{in} + b_1$$

$$A_1 = f_1(Z_1)$$

2. Output layer - The output layer will have vocab_size nodes. We do a final output affine transformation and softmax (f_{out}) activation to get the probabilities for the vocabulary vector.

   For each sample, we calculate our loss on the predicted vocabulary vector to our label vector.

$$Z_{out} = W_{out} A_1 + b_{out}$$

$$A_{out} = f_{out}(Z_{out})$$

$$Loss = crossentropyloss(y, A_{out})$$

   We predict the label by looking at the index of highest probability for each sample.

$$predicted\_label[i] = argmax(A_{out}.T[i])$$

Questions

Weights are usually normalized values between [-1,1]. For easy calculations we will be using integers in the assignment.

Question 1. You are given the following values $vocab = 5,000$, $batch\_size = 100$, $hd_1 = 128$. Write the dimensions of $A_{in}, W_1, b_1, A_1, W_{out}, b_{out}, A_{out}$. (7 points)

Question 2. Forward propogation - You are given the following $W_1, b_1, W_{out}, b_{out}, A_1$ and the activation function f_1. For the purpose of easier calculations we will assume f_{out} is the same as f_1. Calculate Z_1, A_1, Z_{out} and A_{out}. (4+2+4+2 points)

$$W_1 = \begin{bmatrix} 3 & 2 & -1 & 0 & -3 & 2 \\ 4 & -1 & -2 & 3 & 5 & -6 \\ 2 & -1 & 3 & 4 & -3 & 1 \\ 6 & -4 & 2 & -5 & 1 & 2 \end{bmatrix} \qquad b_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

$$W_{out} = \begin{bmatrix} 3 & -1 & 2 & -4 \\ 1 & -5 & -1 & 3 \end{bmatrix} \qquad b_{out} = \begin{bmatrix} 4 \\ -5 \end{bmatrix}$$

$$A_{in} = \begin{bmatrix} 1 & 2 \\ 4 & 5 \\ 1 & 3 \\ 3 & 1 \\ 1 & 4 \\ 2 & 1 \end{bmatrix}$$

$$f_1(x) = f_{out}(x) = relu(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \end{cases}$$

Question 3. Backward propogation - You are given the $\partial Loss/\partial A_{out}$.

$$\frac{\partial Loss}{\partial A_{out}} = \begin{bmatrix} 3 & 5 \\ 2 & -4 \end{bmatrix}$$

Use $W_1, b_1, Z_1, W_{out}, b_{out}$ from q2 to solve the questions below.

a. Write the formula for calculating $\partial Loss/\partial W_{out}$, $\partial Loss/\partial b_{out}$ and $\partial Loss/\partial A_{out}$. (2 point each)

b. Calculate the $\partial Loss$ wrt $W_1, b_1, W_{out}, b_{out}$ given the derivative of relu. (5+4+5+4 points each)

$$relu\_der(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \end{cases}$$

c. Update the $W_1, b_1, W_{out}, b_{out}$ using the gradients with a learning rate of 0.01. (1 points each)

Question 4. The GloVe and word2vec libraries provide pre-trained word embeddings for dimensions from 50 - 300. In an NLP application task what are the trade-offs of using word embeddings of smaller vs larger dimensions? (5 points)

APPENDIX

Word Embeddings

In very simplistic terms, Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text. A Word Embedding format generally tries to map a word using a dictionary to a vector.

A vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else. This is just a very simple method to represent a word in the vector form.

The different types of word embeddings can be broadly classified into two categories-

- Frequency based Embedding - The embeddings Count vectors, TF-IDF vectors that you have come across in this course till now broadly fall under this category. The dimensions of the such vectors are decided by the vocabulary size of the train data.

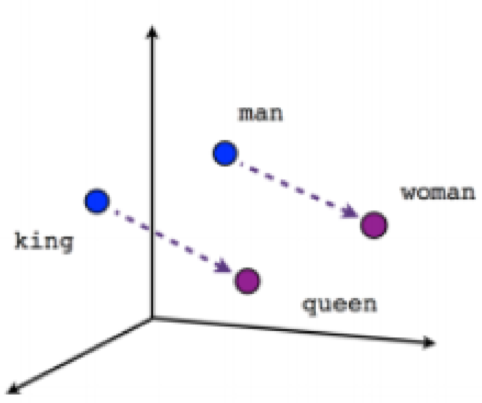
- Prediction based Embedding - The frequency methods limited each dimension in the vector to independent though words rarely are. It was not capable of representing semantic or syntactic information. For example, the word 'house' and the word 'houses' would be mapped to a different dimension in the vector. By looking at the vectors themselves you would not be able to conclude any relation between the 2 words.

  The vectors produced by these methods would also result in high computational cost due to the large matrix multiplications required in neural networks.

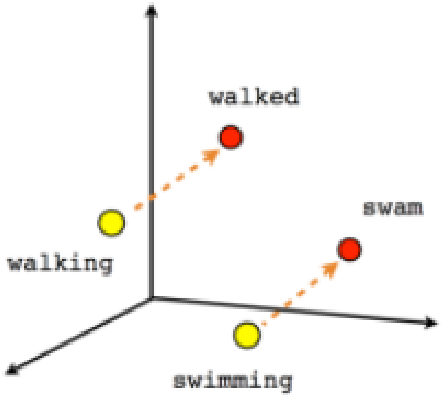
  Word vectors proved to be limited in their word representations until Mikolov et al. introduced word2vec.

  The word2vec methods were prediction based in the sense that they provided probabilities to the words in limited dimensions and proved to be state of the art for tasks like word analogies and word similarities. They were also able to achieve tasks like 'king'-'man'+'woman'='queen' , which was considered a result almost magical.

  We can visualize the learned vectors by projecting them down to 2 dimensions using dimensionality reduction techniques. When we inspect these visualizations it becomes apparent that the vectors capture some general, and in fact quite useful, semantic information about words and their relationships to one another. Some of the induced vector space specialize towards certain semantic relationships, e.g. male-female, verb tense and even country-capital relationships between words, as illustrated in the figure below -



Male-Female



Verb tense

Word2vec is not a single algorithm but a combination of two techniques CBOW(Continuous bag of words) and skip-gram model. Both of these are achieved through shallow neural network architecture. They learn weights which act as word vector representations. For first part of this assignment we will look at skip-gram architecture.