Skip-Gram with Negative Sampling Algorithm Explanation

Arthur Claude, Antoine Guiot, Armand Margerin

February 2020

1 Introduction

This document provides an explanation of the mathematical concepts used for the implementation of our Skip-Gram algorithm.
The main idea behind the Skip-Gram model is that we take every word (focus word) in an input text and also take one-by-one the words that surround it within a window (context words). We then use a model of the probability for each word to actually appear in the window around the focus word in order to build an embedding for each word.

2 Notations

As in our code, **U** is the word representation matrix. Each word is represented by one line of this matrix: the word w_i is represented by the line vector u_i. **V** is the matrix of contextual representations of words. Each word as a contextual word is represented by a line of this matrix: the word w_j is represented by the line vector u_j.

3 Model

The conditional probability of observing the word w_j in the context of the word w_i is modeled by:

$$p(w_j|w_i;U,V) = \frac{1}{1+e^{-u_i^T v_j}} = \sigma(u_i^T v_j)$$

4 Problem

We define a maximum likelihood problem. We are looking for U and V such as:

$$\arg\max_{U,V} \prod_{(w_i,w_j)\in D} \sigma(u_i^T v_j) = \arg\max_{U,V} \sum_{(w_i,w_j)\in D} \log \sigma(u_i^T v_j)$$

In order to avoid a trivial solution (where U and V are matrices containing coefficients all equal to a constant), we introduce the very important concept of **negative sampling**.

For each pair of a focus word and one of its context words, we randomly select **k** words in our vacabulary (different of the focus word and the context word). **k** is the **negativeRate** parameter in our code.
This random selection is realized is made according to the probability $q(select w_j) = f(w_j)^{3/4}$ with $f(w_j)$ the occurrence frequency of the word w_j in the input text. The selected words are noted w'_j, and the pairs (w_i, w'_j) are called **negative pairs**. This step is included in our code when the function **sample** is called in the function **train**.

Now, we are looking for **U** and **V** such as:

$$\arg\max_{U,V} \sum_{(w_i,w_j)\in D} \left(\log\sigma(u_i^T v_j) + \sum_k \log\left(1 - \sigma(u_i^T v'_j)\right) \right)$$

$$= \arg\max_{U,V} \sum_{(w_i,w_j)\in D} \left(\log\sigma(u_i^T v_j) + \sum_k \log\left(\sigma(-u_i^T v'_j)\right) \right)$$

5 Resolution thanks to the stochastic gradient descent method

Because we want to maximize a sum, we use the stochastic gradient descent to search **U** and **V**.

In this way, we are going through the sum, and for each element of the sum the vectors u_i and v_j are updated in the the direction of the gradient of this element calculated in u_i and v_j. Updates are performed with the amplitude η, the learning rate.

The gradients are calculated thanks to the next formulas:

- Partial derivative of $\log\sigma(u_i^T v_j)$ with respect to a component of u_i

$$\frac{\partial \log\sigma(u_i^T v_j)}{\partial u_{i,\lambda}} = v_{j,\lambda}\sigma(-u_i^T v_j)$$

- Partial derivative of $\log\sigma(u_i^T v_j)$ with respect to a component of v_j

$$\frac{\partial \log\sigma(u_i^T v_j)}{\partial v_{j,\lambda}} = u_{i,\lambda}\sigma(-u_i^T v_j)$$

- Partial derivative of $\log \sigma(-u_i^T v_j')$ with respect to a component of u_i

$$\frac{\partial \log \sigma(-u_i^T v_j')}{\partial u_{i,\lambda}} = -v_{j,\lambda}' \sigma(u_i^T v_j')$$

- Partial derivative of $\log \sigma(-u_i^T v_j')$ with respect to a component of v_j'

$$\frac{\partial \log \sigma(-u_i^T v_j')}{\partial v_{j,\lambda}'} = -u_{i,\lambda} \sigma(u_i^T v_j')$$

All the elements of this section are implemented in our **trainWord** function.