Understanding Word2Vec and Paragraph2Vec

August 13, 2016

Abstract

1 Introduction

In these notes we compute the update steps for Para2Vec algorithm [**?**]. These notes focus on the Distributed Memory (dm) model with mean taken at hidden layer (DM-mean). The original paper describes several other adaptations. As Para2Vec is an adaptation of the original word2vec algorithm, the update steps are an easy extension.

2 Word2Vec Architecture

We concentrate on the word2vec continuous bag of words model, with negative sampling and mean taken at hidden layer. This is a single hiddden layer neural network.

2.1 Notation

Let $W_I = \{w_I^0, w_I^1, \ldots w_I^{n_i}\}$ and $W_O = \{w_O^0, w_O^1, \ldots w_O^{n_o}\}$ denote the set of input and output layer nodes respectively. Let h denote the vector at the hidden layer. $n_i = |W_I|$, r, $n_o = |W_O|$ represent the size of input, hidden and output

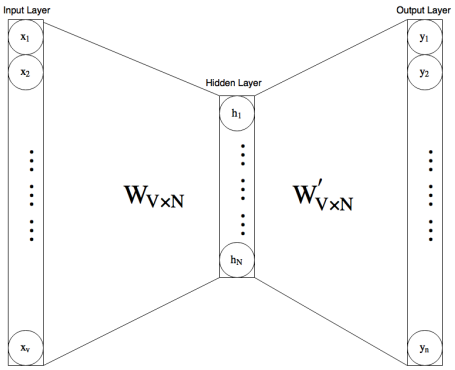

Figure 1:

layer respectively. Let $p_i \in \mathbb{R}^r$ and $q_j \in \mathbb{R}^r$ be the vector representation of the i^{th} input and j^{th} output node respectively. Let P represent a matrix of input layer weights with i^{th} row vector as p_i^T and similarly matrix Q is also defined.

2.2 Training

The training set consists of set of *context* entities (words in word2vec, documents and words in doc2vec) used to predict a target entity at output layer. Let $\mathcal{C}$ and $\mathcal{T}$ represent the set of context and target entities respectively, then $\mathcal{C} \subset W_I$ and $\mathcal{T} \subset W_O$.

For a given training sample, let $C = \{w_I^{c_1}, \ldots w_I^{c_{|C|}}\}$ represent the sequence of context entities and $W_O^{j^*}$ be the target entity. Word2Vec training maximizes the conditional probability of target given context. This probability is modeled using softmax function as:

$$P(W_O^{j^*}|C) = \frac{\exp({q_{j^*}}^T h)}{\sum_j^{n_o} \exp({q_j}^T h)} \tag{1}$$

where h is constructed in a feed-forward manner as

$$h = \frac{1}{|C|} \sum_{i \in C} p_i \tag{2}$$

To compute the update, we take a step in direction of gradient of log of $P(W_O^{j^*}|C)$. Let$f(q_0, \ldots, q_{n_i}, h) = \log(P(W_O^{j^*}|C))$.

$$\frac{\partial f}{\partial q_i} = (I(i = j^*) - \frac{\exp({q_i}^T h)}{\sum_j^{n_O} \exp({q_j}^T h)})h \tag{3}$$

$$= (I(i = j^*) - P(w_O^i|C))h \tag{4}$$

$$= e_i h \tag{5}$$

where $e_i = (I(i = j^*) - P(q_i|C))$ denotes the error incurred at i^{th} output node.

At $(t+1)^{th}$ step:

$$q_i^{t+1} \leftarrow q_i^t + \eta e_i h \tag{6}$$

Further, note that

$$\frac{\partial f(q_0, \ldots, q_{n_o}, h)}{\partial p_i} = \sum_j^{n_o} \frac{\partial f}{\partial q_j} \frac{\partial q_j}{\partial p_i} + \frac{\partial f}{\partial h} \frac{\partial h}{\partial p_i} \tag{7}$$

Note, $\frac{\partial q_j}{\partial p_i} = 0$ as Q and P are independent. Thus,

$$\frac{\partial f}{\partial p_i} = \frac{\partial f}{\partial h} \frac{\partial h}{\partial p_i} \tag{8}$$

$$= \frac{1}{|C|} \frac{\partial f}{\partial h} \, for \, w_I^i \in \mathcal{C} \tag{9}$$

$$= 0 \, otherwise \tag{10}$$

where,

$$\frac{\partial f}{\partial h} = \sum_{j}^{n_o} (I(j = j^*) - P(w_O^j|C))q_j \tag{11}$$

$$= \sum_{j}^{n_o} e_j q_j \tag{12}$$

Thus, update step for p_i where $w_I^i \in \mathcal{C}$ is:

$$p_i^{t+1} \leftarrow p_i^t + \frac{\eta}{|C|} \sum_{j}^{n_o} e_j q_j \tag{13}$$

For $w_I^i \notin \mathcal{C}$:

$$p_i^{t+1} \leftarrow p_i^t \tag{14}$$

2.3 Time Complexity of updates for every training sample

$P(w_O^i|C)$ computation at each output node requires $O(n_o)$ operations. The update equation 6 and 13 requires $O(n_o r)$ computations. As, n_o can be in millions, this is prohibitive as we may have millions of words in the vocabulary.Negative sampling has been proposed as a potential solution to these scalability challenges.

3 Negative Sampling

The bottleneck step in previous computations is the softmax normalization. We can replace softmax by some other model, a commonly used one being logit function, defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

Logit has range in $(0, 1)$. It has some interesting properties:

$$\frac{d\sigma(x)}{dx} = \sigma(x)\sigma(-x) \tag{16}$$

$$\frac{d \log \sigma(x)}{dx} = \sigma(-x) \tag{17}$$

$$1 - \sigma(x) = \sigma(-x) \tag{18}$$

In softmax based approach we maximized the probability of target word conditioned on Context words. Note that as softmax is normalized, this also incorporates minimizing probability of non-target words. For negative sampling instead of considering all non-target words (also known as negative words), few examples are randomly sampled to constitute a negative sample set $\mathcal{N}$. The objective function for negative sampling becomes:

$$\max P(W_O^{j^*}|\mathcal{C}) \left(\Pi_{j \in \mathcal{N}} (1 - P(W_O^j|\mathcal{C})) \right) \tag{19}$$

where, $P(W_O^j|\mathcal{C}) = \frac{1}{1+e^{-q_j^T h}}$. Although We model the conditional probability as logit, it is not a well defined probability distribution as it isn't normalized i.e.

$$\sum_j^{n_o} P(W_O^j|\mathcal{C}) \neq 1$$

. Goldberg-levy explain a way to see the above objective function in terms of a well defined distribution.

Note that we could have also maximized a slightly different objective function:

$$\frac{P(W_O^{j^*}|\mathcal{C})}{\Pi_{j\in\mathcal{N}} P(W_O^j|\mathcal{C})}$$

We will later come back to this.

Derivation of update steps is similar to the softmax case:

$$f(q_0, \ldots, q_{n_i}, h) = \log\left(P(W_O^{j^*}|\mathcal{C})\left(\Pi_{j\in\mathcal{N}}(1 - P(W_O^j|\mathcal{C}))\right)\right) \tag{20}$$

$$= \log\sigma(q_{j^*}^T h) + \sum_{j\in\mathcal{N}} \log\sigma(-q_j^T h) \tag{21}$$

$$\begin{aligned}
\frac{\partial f}{\partial q_i} &= (I(i = j^*) - \sigma(q_j^T h))h \\
&= (I(i = j^*) - P(w_O^i|C))h \\
&= e_i h && \forall i \in \mathcal{N} \cup \{j^*\} \\
&= 0 && otherwise
\end{aligned}$$

where $e_i = (I(i = j^*) - P(w_O^i|C))$ denotes the error incurred at i^{th} output node. Update for output node vector thus is:

$$q_i^{t+1} \leftarrow q_i^t + \eta e_i h \qquad i \in \mathcal{N} \cup \{j^*\} \tag{22}$$

$$q_i^{t+1} \leftarrow q_i^t \qquad otherwise \tag{23}$$

For p_i:

$$\begin{aligned}
\frac{\partial f}{\partial p_i} &= \frac{\partial f}{\partial h}\frac{\partial h}{\partial p_i} \\
&= \frac{1}{|C|}\frac{\partial f}{\partial h} && for\, w_I^i \in \mathcal{C} \\
&= 0 && otherwise
\end{aligned}$$

where,

$$\frac{\partial f}{\partial h} = \sum_j^{\mathcal{N}} \left(I(j = j^*) - \sigma(q_j^T h)\right) q_j \tag{24}$$

$$= \sum_j^{\mathcal{N}} e_j q_j \tag{25}$$

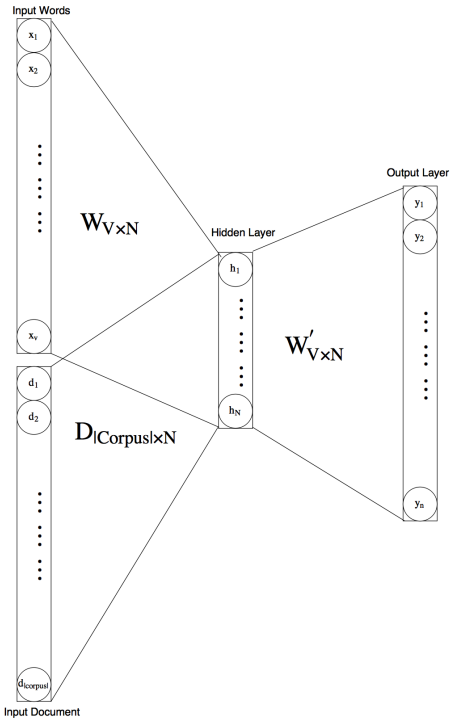

Figure 2: Paragraph2Vector

Update for p_i where $w_I^i \in \mathcal{C}$ is

$$p_i^{t+1} \leftarrow p_i^t + \frac{\eta}{|C|} \sum_j^{\mathcal{N}} e_j q_j \tag{26}$$

3.1 Time Complexity of Negative Sampling

Computing updates in equation 22 takes $O(|\mathcal{N}|r)$ time. Computing updates of equation 29 takes $O\left((|\mathcal{N}| + |\mathcal{C}|)\, r\right)$ time. As the number of negative samples and context words are small constants, this method is practically $O(r)$.

4 Pargraph2Vec

Paragraph2Vec technique includes several different algorithm. We discuss DM with average at hidden layer. The only difference from word2vec is inclusion of documents along with words as input nodes. P2V neural net has input nodes representing documents in the training data (see fig 2).

The rationale behind including documents as input nodes is based upon considering documents as another context. In this abstract sense of context there is no difference between a word and a document. At the time of training we consider (context set, target) pairs as in word2vec, however, for P2V document is also considered a member of the context set. The objective function and the training update steps are exactly the same as word2vec.

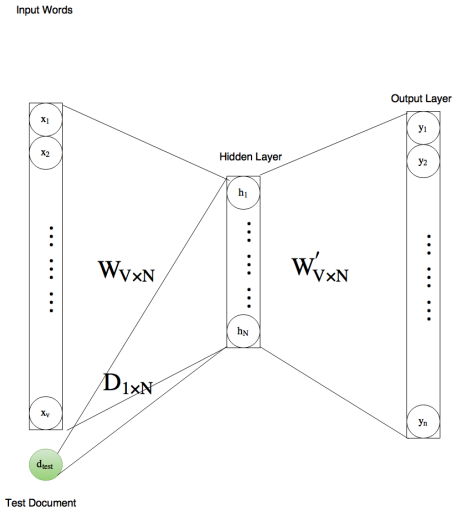

Figure 3: Paragraph2Vector Inference

4.1 Inference

How do we handle new documents in this model? How would one handle new words in word2vec? One solution is to add a new node for this word, find new training data where the new word is present and run some more iterations of W2V training. That is exactly how new documents are handled in P2V. We retrain the model with the words present in the new document (this is the inference step in P2V). Note that the training update equation 29 doesn't impact members not in the context set, $\mathcal{C}$. Thus, at inference time model can be seen as in figure 3. Also, note that the weights P and Q have already been learnt in the training step and one may choose to treat them as constants at the inference step.

The objective function is same as in equation 19, repeated here for clarity:

$$f(q_0, \ldots, q_{n_i}, h) = \log \sigma(q_{j^*}^T h) + \sum_{j \in \mathcal{N}} \log \sigma(-q_j^T h) \tag{27}$$

Update steps treating P and Q constant

P and Q have no updates. Let d_{test} represent the vector representing the new test document.

$$\frac{\partial f}{\partial d_{test}} = \frac{1}{|C|} \frac{\partial f}{\partial h} \tag{28}$$

$\frac{\partial f}{\partial h}$ remains same as in equation 24. Update step for this document vector is:

$$d_{test}^{t+1} \leftarrow d_{test}^t + \frac{\eta}{|C|} \sum_j^{\mathcal{N}} e_j q_j \tag{29}$$

Run time for inference is $O(|\mathcal{N}|r)$.

If P and Q are also updated, then the inference step become similar to training and takes $O((|\mathcal{N}| + |\mathcal{C}|)r)$ time.