0.1 Introduction

0.2 Mathematical Knowledge

0.3 Word Embedding

Four methods are very popular: PPMI, SVD on PPMI, SGNS and Glove

PPMI and SVD on PPMI: "count-based" representations
SGNS and Glove: "neural" or "prediction-based" embeddings

These four methods perform better or as good as other similar but more complex models

0.3.1 Word-Context Pairs

D is the set of all possible word-context pairs in curpus

$\#(w,c)$: times of (w,c) in D

$$\#(w) = \sum_{c' \in V_c} \#(w,c'), \quad \#(c) = \sum_{w' \in V_w} \#(w',c)$$

$w \in V_w$, its vector $\vec{w} \in \mathbb{R}^d$
$c \in V_c$, its vector $\vec{c} \in \mathbb{R}^d$

each vector $\vec{w}$ is a raw in matrix $W : |V_w| * d$
each vector $\vec{c}$ is a raw in matrix $C : |V_c| * d$

W^x and C^x means being produced by a specific method x (e.g. W^{SGNS} or C^{SVD})

0.3.2 PMI and PPMI

PMI: pointwise mutual information

$$PMI(w,c) = \log \frac{\widehat{p}(w,c)}{\widehat{p}(w) \cdot \widehat{p}(c)} = \log \frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)}$$

M^{PMI}: The PMI matrix, $\quad M^{PMI}(w,c) = PMI(w,c)$

Sometimes, let $PMI(w,c) = 0$ if $\#(w,c) = 0$. (originally, $PMI(w,c) = -\infty$)
M_0^{PMI}:

$$M_0^{PMI}(w,c) = \begin{cases} M^{PMI}(w,c), & \#(w,c) > 0 \\ 0, & \#(w,c) = 0 \end{cases}$$

PPMI: positive mutual information

$$PPMI(w,c) = \max(PMI(w,c), 0)$$

M^{PPMI}: The PPMI matrix, $M^{PPMI}(w,c) = PPMI(w,c)$

M^{PPMI} outperforms M_0^{PMI} on semantic similarity tasks

0.3.3 SVD on PPMI

SVD: Singular Value Decomposition

$$M_d = U_d \cdot \Sigma_d \cdot U_d^{\mathrm{T}}$$

$$W^{SVD} = U_d \cdot \Sigma_d, \quad C^{SVD} = V_d$$

respect to L_2 loss ??????

0.3.4 SGNS

0.3.5 Comparison

0.3.6 Details of SGNS

0.4 Word2Vec

This section will introduce two important model in word2vec: CBOW model (Continuous Bag-of-Words Model) and Skip-gram model (Continuous Skip-gram Model).

From the figure, two models both include three layers: **Input Layer**, **Projection Layer**, **Output Layer**. The former is to predict the current word w_t giving its context w_{t-2},w_{t-1},w_{t+1},w_{t+2}

With the foregoing preparation, this section describes word2vec officially used in two important models –CBOW model (Coutinuous Bag-of-Words Model) and Skip-gram model (Continuous Skip-gram Model). About two models, author Tomas Mikolov in [] shows the schematic diagram shown in Figures 8 and 9. Be seen by the two models contain three layers: **Input layer**, **projection layer** and **output layer**. The former is known in the current word w_t context w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} premise predictive current word w_t (see Figure 8); and the latter on the contrary, it is known in the current word w_t premise predict its context w_{t-2}, w_{t-1} , w_{t+1}, w_{t+2} (see Figure 9). For two CBOW and Skip-gram model, word2vec given two frameworks, which are based on Hierarchical Softmax and Negative Sampling to design. This section describes the Hierarchical Softmax CBOW and Skip-gram model. In the previous section, we mentioned that the objective function neural network based language model is generally taken as follows **log-likelihood function**

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(w|Context(w)),$$

The key is the conditional probability function $p(w|Context(w))$ configuration, text [] in this model is given a construction method function (see (3.6) formula). For the objective function Hierarchical Softmax CBOW word2vec model based on optimized also the form (4.1); and for the objective function based on Hierarchical Softmax of Skip-gram model, the optimization of the form

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(Context(w)|w),$$

Therefore, the discussion process, we should focus on the $p(w|Context(w))$ or $p(Context(w)|w)$ on the structure, realize that this is very important because it allows us to targeted, distractions, and will not fall into some of the tedious details were to go. Next, from a mathematical point of these two models in detail.

0.4.1 Skip-gram model with Hierarchical Softmax

This section describes word2vec another model – Skip-gram model, since the derivation and CBOW similar, and therefore will inherit the measure introduced mark.

network

Figure 12 shows the network structure of Skip-gram model, with network structure CBOW model, it also includes three layers: an input layer, a projection layer and output layer. The following sample $(w, Context(w))$, for example, three layers are described briefly.

1. **input layer**: the center of the current sample containing only the word w word vector $\mathbf{v}(w) \in \mathbb{R}^m$.
2. **projection layer**: This projection is identical to $\mathbf{v}(w)$ projection to $\mathbf{v}(w)$. Therefore, **this projection layer is actually superfluous** reason here mainly to facilitate retention projection layer and network structure CBOW models do comparison.
3. **Output layer**: and CBOW model, the output layer is also a lesson Huffman tree.

gradient calculation

For Skip-gram model, it is known that the current word w, need to predict its context $Context(w)$ of the words, the objective function should therefore form (4.2), and the key is the conditional probability function $p(Context(w)|w)$ configuration, in the Skip-gram model which is defined as

$$p(Context(w)|w) = \prod_{u \in Context(w)}^{p(u|w} ,$$

In the above formula $p(u|w)$ in accordance with section describes the Hierarchical Softmax thought, similar to (4.3) written as

$$p(u|w) = \prod_{j=2}^{l^u} p(d_j^u|\mathrm{v}(w), \theta_{j-1}^u),$$

among them

$$p(d_j^u|\mathbf{v}(w), \theta_{j-1}^u) = [\theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)]^{1-d_j^u} \tag{1}$$

The (4.6) followed by generations back, you can get the log-likelihood function (4.2) of the specific expression

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log \prod_{u \in Context(w)} \prod_{j=2}^{l^u} \{[\theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)]^{\ d_j^u}\} = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{j=2}^{l^u} \{(1 - d_j^u) \cdot \log[\tag{2}$$

Similarly, as in the following gradients of convenience, under the triple summation symbol braces contents of abbreviated as $\mathcal{L}(w, u, j)$, ie

$$\mathcal{L}(w, u, j) = (1 - d_j^u) \cdot \log[\theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)] + d_j^u \cdot \log[1 - \theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u].$$

So far, it has been deduced logarithmic likelihood function of expressions like (4.7), which is the objective function Skip-gram model. Then also use **stochastic gradient ascent** method to optimize the key is to give two types of gradients. First consider $\mathcal{L}(w, u, j)$ on θ_{j-1}^u gradient calculation (with the corresponding portion of the model is derived CBOW completely analogous).

$$\partial \frac{\mathcal{L}(w, u, j)}{\partial \theta_{j-1}^u} = \frac{\partial}{\partial \theta_{j-1}^u} \{(1 - d_j^u) \cdot \log[\theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)] + d_j^u \cdot \log[\theta(\mathbf{v}(w)^{\mathrm{T}}\theta_{j-1}^u)]\}$$

0.4.2 Skip-gram model with Negative Sampling

This section will introduce Skip-gram model with Negative Sampling. **Negative Sampling** (**NEG**) is proposed by Tomas Mikolov et al. in [], which is the simplified version of **NCE**(Noise Contrastive Estimation), the purpose is to improve the training and the quality of word vectors. Comparison with Hierarchical Softmax, NEG do not use the (complex) Huffman tree. Instead, it use (relatively simple) **Random Negative Sampling**, which can improve the performance much.

Note 5.1 The details of NCE is a little complex, the essence is to use a known probability density function to estimate an unknown probability density function. In short, assume there is an unknown probability density function Y and a known probability density function X, if we get the relationship between X and Y, we can obtain X as well.The detail of method reference to [].

The objective function is:

$$G = \prod_{w \in \mathcal{C}} \prod_{u \in Context(w)} g(u), \tag{3}$$

Here, we want to maximize $\prod_{u \in Context(w)} g(u)$ giving $(w, Context(w)))$, and $g(u)$ is defined as

$$g(u) = \prod_{z \in u \cup NEG(u)} p(z|w),$$

where $NEG(u)$ represents the negative samples generated by u, the conditional probability

$$p(z|w) = \begin{cases} \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z), & L^u(z) = 1; \\ 1 - \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z), & L^u(z) = 0; \end{cases}$$

where

$$L^u(z) = \begin{cases} 1, & u = z; \\ 0, & u \neq z, \end{cases}$$

It can also be written as one expression

$$p(z|w) = [\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]^{L^u(z)} \cdot [1 - \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]^{1-L^u(z)} \tag{4}$$

And then we use the log of G, so the final objective function is

$$\begin{aligned} L = \log G = \log \prod_{w \in \mathcal{C}} \prod_{u \in Context(w)} g(u) = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \log g(u) \\ = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \log \prod_{z \in \{u\} \cup NEG(u)} p(z|w) \\ = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{z \in \{u\} \cup NEG(u)} \log p(z|w) \\ = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{z \in \{u\} \cup NEG(u)} \log \{[\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]^{L^u(z)} \cdot [1 - \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]^{1-L^u(z)}\} \\ = \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{z \in \{u\} \cup NEG(u)} \{L^u(z) \cdot \log[\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)] + [1 - L^u(z)] \cdot \log[1 - \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\}. \end{aligned}$$

In order to calculate the gradient more conveniently, we use $L(w, u, z)$ to represent the contents of curly braces as

$$\mathcal{L}(w, u, z) = L^u(z) \cdot \log[\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)] + [1 - L^u(z)] \cdot \log[1 - \sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]$$

And next, let's use **Stochastic gradient ascent method** to optimize it. The point is to calculate two kinds of gradient. Let's consider the gradient θ^z firstly.

$$\begin{aligned}
&\frac{\partial \mathcal{L}(w,u,z)}{\partial \theta^z} \\
&= \frac{\partial}{\partial \theta^z}\{L^u(z)\cdot \log[\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)] + [1-L^u(z)]\cdot \log[1-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\} \\
&= L^u(z)[1-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\mathbf{v}(w) - [1-L^u(z)]\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)\mathbf{v}(w) \\
&= \{L^u(z)[1-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)] - [1-L^u(z)]\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)\}\mathbf{v}(w) \\
&= [L^u(z)-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\mathbf{v}(w).
\end{aligned}$$

Thus, the updating formula of θ^z can be written as

$$\theta^z := \theta^z + \eta[L^u(z)-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\mathbf{v}(w).$$

The next, let's consider the gradient of $\mathbf{v}(w)$. Using the **symmetry** of $\mathbf{v}$(w) and θ^z, we have

$$\frac{\partial \mathcal{L}(w,u,z)}{\partial \mathbf{v}(w)} = [L^u(z)-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\theta^z,$$

Thus, the updating formula of $\mathbf{v}(u)$ can be written as

$$\begin{aligned}
\mathbf{v}(w) &:= \mathbf{v}(w) + \eta \sum_{z\in\{u\}\cup NEG\{u\}} \frac{\partial \mathcal{L}(w,u,z)}{\partial \mathbf{v}(w)} \\
&= \mathbf{v}(w) + \eta \sum_{z\in\{u\}\cup NEG\{u\}} [L^u(z)-\sigma(\mathbf{v}(w)^{\mathrm{T}}\theta^z)]\theta^z.
\end{aligned}$$

The following takes the sample $(w, Context(w))$ as the example and gives

0.5 EM Algorithm based on SGHS

0.6 Sense Assignment based on SGNS

0.6.1 Introduction

Corpus is made up by M sentences, and each sentence is made up by several words. Each word in each sentence has one or multiple senses. In the beginning, in each word of each sentence, senses are assigned **randomly**. Every sense have both input embedding and output embedding.

The training algorithm is an iterating between **Assign** and **Learn**. The **Assign** is to use the **score function** (sum of log probability) to select the best sense of the center word. And it uses above process to adjust senses of whole sentence and repeats that until sense assignment of the sentence is stable (not changed). The **Learn** is to use the new sense assignment of each sentence and the gradient of the **loss function** to update the input embedding and output embedding of each sense (using stochastic gradient decent).

0.6.2 Definition

M: the total number of sentences , Dataset: $(S_1, S_2, \dots, S_M)$

S_i: the ith sentence , $S_i = (w_{i,1}, w_{i,2}, \dots, w_{i,L_i})$
L_i: the length of sentence S_i

$w_{i,j}$: the word in the position j of sentence S_i
h: lookup table of sense assignment
$h_{i,j}$: the sense index of word $w_{i,j}$, $1 \leq h_{i,j} \leq N_{w_{i,j}}$
N_w: the max number of senses of word w, $w \in D$
D: Vocabulary

V: lookup table of sense input embedding
U: lookup table of sense output embedding
$V_{w,s}$: the input embedding of sense s of word w
$U_{w,s}$: the output embedding of sense s of word w

K: the number of negative samples

$R(x)$: a random number (integer) from 1 to x
$R()$: a random number (real) from 0.0 to 1.0

0.6.3 Objective Function

$$G = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{L_i}\sum_{t=1}^{L_i}\sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq j+t \leq L_i}}\Bigg(\log p\Big[(w_{i,j+t}, h_{i,j+t})|(w_{i,t}, h_{i,t})\Big] + \sum_{k=1}^{K}\mathbb{E}_{w_k \sim P_n(w)}\log\Big\{1 - p\Big[[w_k, R(N_{w_k})]|(w_{i,t}, h_{i,t})\Big]\Big\}\Bigg) \tag{5}$$

where $p\Big[(w', s')|(w, s)\Big] = \sigma({U_{w',s'}}^{\mathrm{T}} V_{w,s})$ and $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$.

$p\Big[(w_{i,j+t}, h_{i,j+t})|(w_{i,t}, h_{i,t})\Big]$ is the probability of using center word $w_{i,t}$ with sense $h_{i,t}$ to predict one surrounding word $w_{i,j+t}$ with sense $h_{i,t+j}$, which needs to be **maximized**. And $p\Big[[w_k, R(N_{w_k})]|(w_{i,t}, h_{i,t})\Big]$ is the probability of using center word $w_{i,t}$ with sense $h_{i,t}$ to predict one negative sample word w_k with a **random sense** $R(N_{w_k})$, which needs to be **minimized**. It is noteworthy that, $h_{i,t}$ ($w_{i,t}$'s sense) and $h_{i,t+j}$ ($w_{i,t+j}$'s sense) are assigned advance and $h_{i,t}$ may be changed in the **Assign**. But w_k's sense (negative sample) is always assigned randomly.

The final objective is to find out optimized parameters $\theta = \{h, U, V\}$ to maximize the Objective Function G, where h is updated in the **Assign** and $\{U, V\}$ is updated in the **Learn**.

When the center word $w_{i,t}$ is giving, we use **score function** $f_{i,t}$

$$f_{i,t}(s) = \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le t+j \le L_i}} \Bigg(\log p\Big[(w_{i,t+j}, h_{i,t+j}) | (w_{i,t}, s)\Big]$$

$$+ \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n(w)} \log \Big\{1 - p\Big[[w_k, R(N_{w_k})] | (w_{i,t}, s)\Big]\Big\}\Bigg)$$

to select the "best" sense of each center word in the **Assign**. To be noted that, for each score function $f_{i,t}$, all negative samples maintain same as long as generated. So different score is only based on different sense (input value).

And taking $\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j}), \big[(w_1, s_1), \ldots, (w_K, s_K)\big]\big)$ as a training sample (assuming the negative samples and relative senses are generated already), we use **loss function** *loss* for each sample

$$loss\Bigg((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j}), \big[(w_1, s_1), \ldots, (w_K, s_K)\big]\Bigg)$$

$$= -\log p\Big[(w_{i,t+j}, h_{i,t+j}) | (w_{i,t}, h_{i,t})\Big] - \sum_{k=1}^{K} \log \Big\{1 - p\Big[[w_k, s_k] | (w_{i,t}, h_{i,t})\Big]\Big\}$$

to calculate the gradient and update embeddings (including embeddings of negative samples) in the **Learn**. Here the loss is defined as the negative log probability.

The relative gradients calculation is

$$\Delta_{V_{w_{i,t}, h_{i,t}}} = -\frac{\partial loss\Bigg((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j}), \big[(w_1, s_1), \ldots, (w_K, s_K)\big]\Bigg)}{\partial V_{w_{i,t}, h_{i,t}}}$$

$$= [1 - \log \sigma(U_{w_{i,t+j}, h_{i,t+j}}{}^{\mathrm{T}} V_{w_{i,t}, h_{i,t}})] U_{w_{i,t+j}, h_{i,t+j}} + \sum_{k=1}^{K} [-\log \sigma(U_{w_k, s_k}{}^{\mathrm{T}} V_{w_{i,t}, h_{i,t}}))] U_{w_k, s_k}$$

$$\Delta_{U_{w_{i,t+j}, h_{i,t+j}}} = -\frac{\partial loss\Bigg((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j}), \big[(w_1, s_1), \ldots, (w_K, s_K)\big]\Bigg)}{\partial U_{w_{i,t+j}, h_{i,t+j}}}$$

$$= [1 - \log\,\sigma(U_{w_{i,t+j},h_{i,t+j}}{}^{\mathrm{T}} V_{w_{i,t},h_{i,t}})] V_{w_{i,t},h_{i,t}}$$

$$\Delta_{U_{w_k,s_k}} = -\frac{\partial loss\Big((w_{i,t},h_{i,t}),(w_{i,t+j},h_{i,t+j}),\big[(w_1,s_1),\ldots,(w_K,s_K)\big]\Big)}{\partial U_{w_k,s_k}}$$
$$= [-\log\,\sigma(U_{w_k,s_k}{}^{\mathrm{T}} V_{w_{i,t},h_{i,t}}))] V_{w_{i,t},h_{i,t}}$$

0.6.4 Algorithm Description

Initialization:

$$h_{i,j} = R(N_{w_{i,j}}),\ 1 \le i \le M,\ 1 \le j \le L_i$$
$$V_{w,s} = \Big[\underbrace{\frac{R()-0.5}{d},\ldots,\frac{R()-0.5}{d}}_{d}\Big]^{\mathrm{T}},\ w \in D,\ 1 \le s \le N_w$$
$$U_{w,s} = \Big[\underbrace{0,\ldots,0}_{d}\Big]^{\mathrm{T}},\ w \in D,\ 1 \le s \le N_w$$

Assign:

```
FOR $i$:= 1 TO $M$
   DO
      FOR $t$:= 1 TO $L_i$
         $h_{i,t} = \max_{1 \le s \le N_{w_{i,t}}} f_{i,t}(s)$
      END
   UNTIL no $h_{i,t}$ changed
END
```

Learn:

```
FOR $i$:= 1 TO $M$
   FOR $t$:= 1 TO $L_i$
      FOR $j$:= $-c$ TO $c$
         IF $j \neq 0$ AND $t + j \geq 1$ AND $t + j \leq L_i$ THEN

            generate negative samples $\big[(w_1,s_1),\ldots,(w_K,s_K)\big]$

            $\Delta = -\nabla_\theta loss\Big((w_{i,t},h_{i,t}),(w_{i,t+j},h_{i,t+j}),\big[(w_1,s_1),\ldots,(w_K,s_K)\big]\Big)$
            $\Delta$ is made up by $\{\Delta_{V_{w_{i,t},h_{i,t}}}, \Delta_{U_{w_{i,t+j},h_{i,t+j}}}, [\Delta_{U_{w_1,w_1}},\ldots,\Delta_{U_{w_K,w_K}}]\}$
```

```
                    $V_{w_{i,t},h_{i,t}} = V_{w_{i,t},h_{i,t}} + \alpha\Delta_{V_{w_{i,t},h_{i,t}}}$
                    $U_{w_{i,t+j},h_{i,t+j}} = U_{w_{i,t+j},h_{i,t+j}} + \alpha\Delta_{U_{w_{i,t+j},h_{i,t+j}}}$
                    FOR $k$:= 1 TO $K$
                        $U_{w_k,s_k} = U_{w_k,s_k} + \alpha\Delta_{U_{w_k,s_k}}$
                    END

                END
            END
        END
    END
```

Iterating between **Assign** and **Learn** till the convergence of the value of G makes the whole algorithm complete.

0.7 Implementation and Evaluation

0.8 Conclusion