0.1 Skipgram-model with negative sampling

0.1.1 Introduction

Negative sample words are sampled according to a smoothed unigram distribution.

0.1.2 Definition

Corpus C: $(S_1, S_2, \ldots, S_M)$
M: the total number of sentences

S_i: the ith sentence, $S_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,L_i})$
L_i: the length of sentence S_i
$w_{i,j}$: the word in the position j of sentence S_i

V: lookup table of sense input embedding
U: lookup table of sense output embedding
V_w: the input embedding of w, $w \in D$, $V_w \in \mathbb{R}^d$
U_w: the output embedding of w, $w \in D$, $U_w \in \mathbb{R}^d$
D: Vocabulary
d: the size of embedding vector (both input and output)

K: the number of negative samples

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

c: the size of context

$P_n(w)$: smoothed unigram distribution , $P_n(w) = \frac{count(w)^{\frac{3}{4}}}{(\sum_{i=1}^{M} L_i)^{\frac{3}{4}}}, \quad w \in D$
$count(w)$: the number of times w occurred in C
$\sum_{i=1}^{M} L_i$: the number of total words in C

0.1.3 Objective Function

$$G = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \le j \le c \\ j \neq 0 \\ 1 \le j+t \le L_i}} \left\{ \log\, p(w_{i,t+j}|w_{i,t}) + \sum_{k=1}^{K} \mathbb{E}_{z_k \sim P_n(w)} \log\, [1 - p(z_k|w_{i,t})] \right\} \quad (1)$$

where $p(w'|w) = \sigma({U_{w'}}^{\mathrm{T}} V_w)$ and $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$.

$p(w_{i,t+j}|w_{i,t})$ is the probability of using center word $w_{i,t}$ to predict one surrounding word $w_{i,t+j}$, which needs to be **maximized**. $z_1, \ldots, z_K$ are the negative sample words to

replace word $w_{i,t+j}$, and $p(z_k|w_{i,t})$ $(1 \leq k \leq K)$ is the probability of using center word $w_{i,t}$ to predict one negative sample word z_k, which needs to be **minimized**. Equivalently, the whole objective function needs to be **maximized**.

Take $(w_{i,t}, w_{i,t+j})$ as a training sample, and define **loss function** $loss$ for each sample

$$loss(w_{i,t}, w_{i,t+j})$$

$$= -\log\, p(w_{i,t+j}|w_{i,t}) - \sum_{k=1}^{K} \mathbb{E}_{z_k \sim P_n(w)} \log\, [1 - p(z_k|w_{i,t})]$$

Here the loss is defined as the negative log probability.

And the loss function of whole corpus is

$$loss(C) = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq j+t \leq L_i}} loss(w_{i,t}, w_{i,t+j})$$

To maximize the objective function is equivalently to minimize the loss function. So the objective of learning algorithm is

$$\arg\min_{\{V,U\}} \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq j+t \leq L_i}} loss(w_{i,t}, w_{i,t+j})$$

Use

$$N = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq j+t \leq L_i}} 1$$

to represent the number of total training samples in one epoch. (An epoch is a measure of the number of times all of the training samples are used once.) And the number of epochs is T. So the total iterations is $N * T$.

Use stochastic gradient descent:

- Initialize $\{V, U\}$
- For $N * T$ Iterations:
  - For each training sample $(w_{i,t}, w_{i,t+j})$
    - Generate negative sample words to replace $w_{i,t+j}$: $(w_1, \ldots, z_k)$
    - Calculate the gradient $\Delta = -\nabla_{\{V,U\}} loss(w_{i,t}, w_{i,t+j})$

* Δ is only made up by $\{\Delta_{V_{w_{i,t}}}, \Delta_{U_{w_{i,t+j}}}, [\Delta_{U_{w_1}}, \ldots, \Delta_{U_{z_k}}]\}$
* Update Embeddings:
  - $V_{w_{i,t}} = V_{w_{i,t}} + \alpha\Delta_{V_{w_{i,t}}}$
  - $U_{w_{i,t+j}} = U_{w_{i,t+j}} + \alpha\Delta_{U_{w_{i,t+j}}}$
  - $U_{z_k} = U_{z_k} + \alpha\Delta_{U_{z_k}}, 1 \leq k \leq K$

  (α is the learning rate and will be updated every several iterations)

The detail of gradient calculation of $loss(w_{i,t}, w_{i,t+j})$ is

$$\Delta_{V_{w_{i,t}}} = -\frac{\partial loss(w_{i,t}, w_{i,t+j})}{\partial V_{w_{i,t}}} = [1 - \log\ \sigma(U_{w_{i,t+j}}^{\mathrm{T}} V_{w_{i,t}})]U_{w_{i,t+j}} + \sum_{i=1}^{k}[-\log\ \sigma(U_{z_k}^{\mathrm{T}} V_{w_{i,t}}))]U_{z_k}$$

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

$$\Delta_{U_{z_k}} = -\frac{\partial loss(w_{i,t}, w_{i,t+j})}{\partial U_{z_k}} = [-\log\ \sigma(U_{z_k}^{\mathrm{T}} V_{w_{i,t}}))]V_{w_{i,t}}, 1 \leq k \leq K$$

0.2 Sense Assignment based on SGNS

0.2.1 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).

0.2.2 Definition

Corpus C: $(S_1, S_2, \ldots, S_M)$
M: the total number of sentences

S_i: the ith sentence , $S_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,L_i})$
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, $w \in D$

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, $w \in D$, $1 \leq s \leq N_w$, $V_{w,s} \in \mathbb{R}^d$
$U_{w,s}$: the output embedding of sense s of word w, $w \in D$, $1 \leq s \leq N_w$, $U_{w,s} \in \mathbb{R}^d$

D: Vocabulary
d: the size of embedding vector (both input and output)

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

c: the size of context

$P_n(w)$: smoothed unigram distribution , $P_n(w) = \frac{count(w)^{\frac{3}{4}}}{(\sum_{i=1}^{M} L_i)^{\frac{3}{4}}}$, $w \in D$
$count(w)$: the number of times w occurred in C
$\sum_{i=1}^{M} L_i$: the number of total words in C

0.2.3 Objective Function

$$G = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{L_i}\sum_{t=1}^{L_i}\sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq j+t \leq L_i}}\left(\log p\Big[(w_{i,t+j}, h_{i,t+j})|(w_{i,t}, h_{i,t})\Big] + \sum_{k=1}^{K}\mathbb{E}_{z_k \sim P_n(w)}\log\Big\{1 - p\Big[[z_k, R(N_{z_k})]|(w_{i,t}, h_{i,t})\Big]\Big\}\right) \tag{2}$$

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

$p\Big[(w_{i,t+j}, h_{i,t+j})|(w_{i,t}, h_{i,t})\Big]$ is the probability of using center word $w_{i,t}$ with sense $h_{i,t}$ to predict one surrounding word $w_{i,t+j}$ with sense $h_{i,t+j}$, which needs to be **maximized**. $[z_1, R(N_{z_1})],\ldots,[(z_K, R(N_{z_K})]$ are the negative sample words with random assigned senses to replace $(w_{i,t+j}, h_{i,t+j})$, and $p\Big[[z_k, R(N_{z_k})]|(w_{i,t}, h_{i,t})\Big]$ $(1 \leq k \leq K)$ is the probability of using center word $w_{i,t}$ with sense $h_{i,t}$ to predict one negative sample word z_k with sense $R(N_{z_k})$, 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 z_k's sense

s_k 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** $f_{i,t}$ with fixed negative samples $\bigcup_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le j+t \le L_i}} [(z_{j,1}, s_{j,1}), \ldots, (z_{j,K}, s_{j,K})]$ (senses are assigned randomly already)

$$f_{i,t}(s) = \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le t+j \le L_i}} \left(\log p[(w_{i,t+j}, h_{i,t+j})|(w_{i,t}, s)] + \sum_{k=1}^{K} \log \left\{ 1 - p[(z_{j,k}, s_{j,k})|(w_{i,t}, s)] \right\} \right)$$

to select the "best" sense (with the max value) of each center word in the **Assign**.

Taking $[(w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})]$ as a training sample, we define **loss function** $loss$ for each sample as

$$loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)$$

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

Here the loss is defined as the negative log probability.

And the loss function of whole corpus is

$$loss(C) = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le j+t \le L_i}} loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)$$

After **Assign**, h is fixed. So we the same method in the normal Skip-gram with negative sampling model (stochastic gradient decent) to minimize G in the **Learn**. So the objective of **Learn** is

$$\arg\min_{\{V,U\}} \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le j+t \le L_i}} loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)$$

Use

$$N = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L_i} \sum_{t=1}^{L_i} \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le j+t \le L_i}} 1$$

to represent the number of total training samples in one epoch. (An epoch is a measure of the number of times all of the training samples are used once.) .

Use stochastic gradient descent:

- For N Iterations:
  - For each training sample $(w_{i,t}, w_{i,t+j})$
    - Generate negative sample words to replace $w_{i,t+j}$: $(w_1, \ldots, z_k)$
    - Calculate the gradient $\Delta = -\nabla_{\{V,U\}} loss(w_{i,t}, w_{i,t+j})$
    - Δ is only made up by $\{\Delta_{V_{w_{i,t}}}, \Delta_{U_{w_{i,t+j}}}, [\Delta_{U_{w_1}}, \ldots, \Delta_{U_{z_k}}]\}$
    - Update Embeddings:
      - $V_{w_{i,t}} = V_{w_{i,t}} + \alpha \Delta_{V_{w_{i,t}}}$
      - $U_{w_{i,t+j}} = U_{w_{i,t+j}} + \alpha \Delta_{U_{w_{i,t+j}}}$
      - $U_{z_k} = U_{z_k} + \alpha \Delta_{U_{z_k}}, 1 \leq k \leq K$

      (α is the learning rate and will be updated every several iterations)

0.2.4 Algorithm Description

Initialization:

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

Assign:

FOR i:= 1 TO M
  DO
    FOR t:= 1 TO L_i
      $h_{i,t} = \max_{1 \leq s \leq N_{w_{i,t}}} f_{i,t}(s)$
    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 $\big[(z_1, s_1), \ldots, (z_K, s_K)\big]$

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

        $V_{w_{i,t},h_{i,t}} = V_{w_{i,t},h_{i,t}} + \alpha\Delta_{V_{w_{i,t},h_{i,t}}}$
        $U_{w_{i,t+j},h_{i,t+j}} = U_{w_{i,t+j},h_{i,t+j}} + \alpha\Delta_{U_{w_{i,t+j},h_{i,t+j}}}$
        $U_{z_k,s_k} = U_{z_k,s_k} + \alpha\Delta_{U_{z_k,s_k}}, 1 \leq k \leq K$

      END
    END
  END
END

The detail of gradient calculation of $loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)$ is

$$\Delta_{V_{w_{i,t},h_{i,t}}} = -\frac{\partial loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)}{\partial V_{w_{i,t},h_{i,t}}}$$

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

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

$$\Delta_{U_{z_k,s_k}} = -\frac{\partial loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)}{\partial U_{z_k,s_k}}$$
$$= [-\log\, \sigma({U_{z_k,s_k}}^{\mathrm{T}} V_{w_{i,t},h_{i,t}}))]V_{w_{i,t},h_{i,t}}$$

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

0.3 Implementation