CS 224n Assignment #2: word2vec (43 Points)

1 Written: Understanding word2vec (23 points)

Let's have a quick refresher on the `word2vec` algorithm. The key insight behind `word2vec` is that *'a word is known by the company it keeps'*. Concretely, suppose we have a 'center' word c and a contextual window surrounding c. We shall refer to words that lie in this contextual window as 'outside words'. For example, in Figure 1 we see that the center word c is 'banking'. Since the context window size is 2, the outside words are 'turning', 'into', 'crises', and 'as'.

The goal of the skip-gram `word2vec` algorithm is to accurately learn the probability distribution $P(O|C)$. Given a specific word o and a specific word c, we want to calculate $P(O = o|C = c)$, which is the probability that word o is an 'outside' word for c, i.e., the probability that o falls within the contextual window of c.

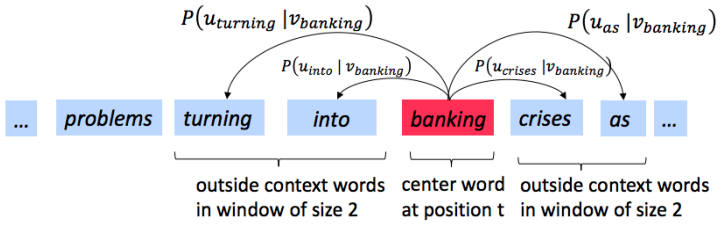


Figure 1: The word2vec skip-gram prediction model with window size 2

In `word2vec`, the conditional probability distribution is given by taking vector dot-products and applying the softmax function:

$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{w \in \text{Vocab}} \exp(\boldsymbol{u}_w^\top \boldsymbol{v}_c)} \tag{1}$$

Here, $\boldsymbol{u}_o$ is the 'outside' vector representing outside word o, and $\boldsymbol{v}_c$ is the 'center' vector representing center word c. To contain these parameters, we have two matrices, $\boldsymbol{U}$ and $\boldsymbol{V}$. The columns of $\boldsymbol{U}$ are all the 'outside' vectors $\boldsymbol{u}_w$. The columns of $\boldsymbol{V}$ are all of the 'center' vectors $\boldsymbol{v}_w$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ contain a vector for every $w \in$ Vocabulary.[1]

Recall from lectures that, for a single pair of words c and o, the loss is given by:

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o|C = c). \tag{2}$$

Another way to view this loss is as the cross-entropy[2] between the true distribution $\boldsymbol{y}$ and the predicted distribution $\hat{\boldsymbol{y}}$. Here, both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are vectors with length equal to the number of words in the vocabulary. Furthermore, the k^{th} entry in these vectors indicates the conditional probability of the k^{th} word being an 'outside word' for the given c. The true empirical distribution $\boldsymbol{y}$ is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else. The predicted distribution $\hat{\boldsymbol{y}}$ is the probability distribution $P(O|C = c)$ given by our model in equation (1).

(a) (3 points) Show that the naive-softmax loss given in Equation (2) is the same as the cross-entropy loss between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$; i.e., show that

[1] Assume that every word in our vocabulary is matched to an integer number k. $\boldsymbol{u}_k$ is both the k^{th} column of $\boldsymbol{U}$ and the 'outside' word vector for the word indexed by k. $\boldsymbol{v}_k$ is both the k^{th} column of $\boldsymbol{V}$ and the 'center' word vector for the word indexed by k. **In order to simplify notation we shall interchangeably use k to refer to the word and the index-of-the-word.**

[2] The Cross Entropy Loss between the true (discrete) probability distribution p and another distribution q is $-\sum_i p_i \log(q_i)$.

$$-\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -\log(\hat{y}_o). \tag{3}$$

Your answer should be one line.

(b) (5 points) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$.

(c) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to each of the 'outside' word vectors, $\boldsymbol{u}_w$'s. There will be two cases: when $w = o$, the true 'outside' word vector, and $w \neq o$, for all other words. Please write you answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$.

(d) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}} = \frac{e^{\boldsymbol{x}}}{e^{\boldsymbol{x}} + 1} \tag{4}$$

Please compute the derivative of $\boldsymbol{\sigma}(x)$ with respect to $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector.

(e) (4 points) Now we shall consider the Negative Sampling loss, which is an alternative to the Naive Softmax loss. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. Note that $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[3]

Please repeat parts (b) and (c), computing the partial derivatives of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. After you've done this, describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss. Note, you should be able to use your solution to part (d) to help compute the necessary gradients here.

(f) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{6}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{U}$

(ii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$

[3]Note: the loss function here is the negative of what Mikolov et al. had in their original paper, because we are doing a minimization instead of maximization in our assignment code. Ultimately, this is the same objective function.

(iii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})/\partial \boldsymbol{v}_w$ when $w \neq c$

Write your answers in terms of $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})/\partial \boldsymbol{U}$ and $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})/\partial \boldsymbol{v}_c$. This is very simple – each solution should be one line.

Once you're done: *Given that you computed the derivatives of $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ with respect to all the model parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ in parts (a) to (c), you have now computed the derivatives of the full loss function $\boldsymbol{J}_{\text{skip-gram}}$ with respect to all parameters. You're ready to implement* `word2vec`*!*

2 Coding: Implementing word2vec (20 points)

In this part you will implement the word2vec model and train your own word vectors with stochastic gradient descent (SGD). Before you begin, first run the following commands within the assignment directory in order to create the appropriate conda virtual environment. This guarantees that you have all the necessary packages to complete the assignment.

```
conda env create -f env.yml
conda activate a2
```

Once you are done with the assignment you can deactivate this environment by running:

```
conda deactivate
```

(a) (12 points) First, implement the `sigmoid` function in `word2vec.py` to apply the sigmoid function to an input vector. In the same file, fill in the implementation for the softmax and negative sampling loss and gradient functions. Then, fill in the implementation of the loss and gradient functions for the skip-gram model. When you are done, test your implementation by running `python word2vec.py`.

(b) (4 points) Complete the implementation for your SGD optimizer in `sgd.py`. Test your implementation by running `python sgd.py`.

(c) (4 points) Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run `sh get_datasets.sh`. There is no additional code to write for this part; just run `python run.py`.

Note: The training process may take a long time depending on the efficiency of your implementation ***(an efficient implementation takes approximately an hour)****. Plan accordingly!*

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as `word_vectors.png` in your project directory. **Include the plot in your homework write up.** Briefly explain in at most three sentences what you see in the plot.

3 Submission Instructions

You shall submit this assignment on GradeScope as two submissions – one for "Assignment 2 [coding]" and another for 'Assignment 2 [written]":

(a) Run the `collect_submission.sh` script to produce your `assignment2.zip` file.

(b) Upload your `assignment2.zip` file to GradeScope to "Assignment 2 [coding]".

(c) Upload your written solutions to GradeScope to "Assignment 2 [written]".

(a) $J_{\text{naive-softmax}} = -\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -\log(\hat{y}_o)$.

Your answer should be one line.

这里并不对计算一个0，会对window内的单词计算.

证明：y_w是一列，$y_w = \begin{cases} 1, & w = o \text{（真实上下文）} \\ 0, & w \neq o \text{（其他）} \end{cases}$

$\therefore -\sum_{w \in V} y_w \log(\hat{y}_w) = -\sum_{w \in V, w \neq 0} y_w \log(\hat{y}_w) - y_o \log(\hat{y}_o)$

$= 0 - 1 \cdot \log(\hat{y}_o) = -\log(\hat{y}_o)$

(b) (5 points) Compute the partial derivative of $J_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$.

$$\frac{\partial J}{\partial v_c} = -\frac{\partial}{\partial v_c} \log p(o|c) = -\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

$$= -\frac{\partial}{\partial v_c} \left[\log \exp(u_o^T v_c) - \log \left(\sum_{w \in V} \exp(u_w^T v_c) \right) \right]$$

$u_o^T v_c$ 是 f，$\sum_{w \in V} \exp(u_w^T v_c)$ 是 $z(v_c)$

$$= -\left[u_o - \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot \sum_{w \in V} u_w \cdot \exp(u_w^T v_c) \right]$$

$$= -u_o + \frac{\sum_{w \in V} u_w \cdot \exp(u_w^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} = -u_o + \sum_{w \in V} u_w p(w|c)$$

$p(w|c)$

$\therefore \frac{\partial J}{\partial v_c} = -u_o + \sum_{w \in V} u_w p(w|c)$

注：y是输入训练集（一列中中心词$y_o=1$，$y_{w\neq o}=0$）

$\hat{y}$是输出值/预测值

U代表一个单词作为上下文的向量.

V代表一个单词作为中心词的向量.

$\therefore u_o = U^T y$表示取第o个单词作为上下文向量（因为y是one-hot）

$\hat{y}_w = p(w|c)$，所以$\sum_{w \in V} u_w \hat{y}_w = U^T \hat{y}$

$\therefore \frac{\partial J}{\partial v_c} = -U^T y + U^T \hat{y} = U^T(\hat{y} - y)$ 对一个o-hot向量再sum.

(c) (5 points) Compute the partial derivatives of $J_{\text{naive-softmax}}(v_c, o, U)$ with respect to each of the 'outside' word vectors, u_w's. There will be two cases: when $w = o$, the true 'outside' word vector, and $w \neq o$, for all other words. Please write you answer in terms of y, $\hat{y}$, and v_c.

① 当$w=o$时.

$$\frac{\partial J}{\partial u_o} = -\frac{\partial}{\partial u_o}\left(u_o^T v_c - \log \sum_{w \in V} \exp(u_w^T v_c)\right)$$

$$= -\left(v_c - \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot \exp(u_o^T v_c) \cdot v_c\right)$$

$$= -v_c + v_c p(o|c)$$

这里对应了$\sum_{w \in V, w=o}$

② 当 $w \neq 0$ 时

$$\frac{\partial}{\partial u_w} = -\frac{\partial}{\partial u_w}\left(\exp(u_o^T v_c) - \log \sum_{w \in V} \exp(u_w^T v_c)\right)$$

$$= \frac{\partial}{\partial u_w} \log \sum_{w \in V} \exp(u_w^T v_c)$$

这里求的不再有 $\sum_{w \in V}$ ∴ 对 u_w 求导 若对 v_c 求导也要 $\sum_{w \in V}$

$$= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot v_c \exp(u_w^T v_c)$$

$\cdot p(w|c)$

$$= v_c p(w|c)$$

$\because \frac{\partial J}{\partial v} = (\hat{y} - y) \cdot U$

$$\therefore \frac{\partial J}{\partial u_w} = \begin{cases} -v_c + v_c p(o|c), & w = o \\ v_c p(w|c), & w \neq o \end{cases}$$

(d) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}}} = \frac{e^{\boldsymbol{x}}}{e^{\boldsymbol{x}}+1} \tag{4}$$

Please compute the derivative of $\boldsymbol{\sigma}(x)$ with respect to $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector.

$$\frac{\partial \sigma(x)}{\partial x} = \frac{e^x(e^x+1) - e^x \cdot e^x}{(e^x+1)^2} = \frac{e^x}{(e^x+1)^2}$$

$$= \sigma(x)\left[1 - \sigma(x)\right]$$

$\because \frac{\partial \sigma(-x)}{\partial x} = \left(\frac{1}{1+e^x}\right)' = -\sigma(-x)\left[1-\sigma(x)\right]$

$= -\left\{\sigma(x)\left[1-\sigma(x)\right]\right\}$

(e) (4 points) Now we shall consider the Negative Sampling loss, which is an alternative to the Naive Softmax loss. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. Note that $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

skip-gram: 这里涉及中心词c，选择K个O(与正采样不重复)

$$J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[3]

改写O-c的和

Please repeat parts (b) and (c), computing the partial derivatives of $J_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. After you've done this, describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss. Note, you should be able to use your solution to part (d) to help compute the necessary gradients here.

$$\frac{\partial J}{\partial v_c} = -\frac{\sigma(u_o^T v_c)(1-\sigma(u_o^T v_c))}{\sigma(u_o^T v_c)} \cdot u_o - \sum_{k=1}^{K} \frac{\sigma(1-\sigma)(-u_k)}{\sigma(-u_k^T v_c)}$$

(已知 $\sigma(x)+\sigma(-x)=1$)

$$= -\left[1-\sigma(u_o^T v_c)\right] u_o + \sum_{k=1}^{K}\left[1-\sigma(-u_k^T v_c)\right] u_k$$

$$\frac{\partial J}{\partial u_o} = -\frac{\sigma(u_o^T v_c)(1-\sigma(u_o^T v_c))}{\sigma(u_o^T v_c)} \cdot v_c - \sum_{k=1}^{K} \frac{\sigma(-u_k^T v_c)(1-\sigma(-u_k^T v_c))}{\sigma(-u_k^T v_c)} \cdot 0 \quad \left(\because \frac{\partial(-u_k^T v_c)}{u_o} = 0 \ (o \neq k)\right)$$

$$= -\left[1-\sigma(u_o^T v_c)\right] v_c$$

$$\frac{\partial J}{\partial u_k} = 0 - \frac{\sigma(-u_k^T v_c)\left[1-\sigma(-u_k^T v_c)\right]\cdot(-v_c)}{\sigma(-u_k^T v_c)} = \left[1-\sigma(-u_k^T v_c)\right] v_c$$

综上：

大量计算与U有关

For naive-softmax: $\frac{\partial J}{\partial v_c} = U(\hat{y}-y)$、$\frac{\partial J}{\partial U} = (\hat{y}-y)v_c$

For neg_sample: $\frac{\partial J}{\partial v_c} = \sigma(-u_o^T v_c)\cdot(-u_o) + \sum_{k=1}^{K}\sigma(u_k^T v_c)\cdot u_k$

少量计算，与V有关

$\frac{\partial J}{\partial u_o} = \sigma(-u_o^T v_c)(-v_c)$.　　$\frac{\partial J}{\partial u_k} = \sigma(u_k^T v_c)\cdot v_c$

(f) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \le j \le m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{6}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{U}$

(ii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$

(iii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_w$ when $w \neq c$

Write your answers in terms of $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{U}$ and $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$. This is very simple – each solution should be one line.

Once you're done: *Given that you computed the derivatives of $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ with respect to all the model parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ in parts (a) to (c), you have now computed the derivatives of the full loss function $\boldsymbol{J}_{\text{skip-gram}}$ with respect to all parameters. You're ready to implement* `word2vec`*!*

这里用 $J_{neg\text{-}sample}$ 来求偏导.

(i) $\frac{\partial J(v_c, w_{t+j}, U)}{\partial U}$ 代入(e)中的 u_o. ∂u_k

(ii) 代入(e)中 $\frac{\partial J}{\partial v_c}$.

(iii) $w \neq c$ 时, (ii)中的 v_c 是不包括 v_w 的

(中心词是确定的, 负采样不换中心词)

$\therefore \frac{\partial J}{\partial v_w} = 0$

Window内的每一个outside words.

注: $J(v_c, 0, U)$ 的0只有一个, 每次计算一个, 多个加起来

Coding:

① naive-softmax-∂J/∂-

$\hat{y} = \text{softmax}(U \cdot V_c)$ 5×1

(U: 5×10, V_c: 10×1)

设U 5×10，V 10×5，5个10维

$loss = -\log(\hat{y}_o)$ 见(a)

— 只有中心词时对应的y_o为1，详见(a)

$\frac{\partial J}{\partial V_c} = U^T(\hat{y} - y)$ 见(b) 10×1

(10×5 · 5×1)

$\frac{\partial J}{\partial U} = (\hat{y} - y) V_c^T$ 见(c) 5×10

(5×1, 1×10)

② neg-sampling-∂J/∂- 见(e)

score $[\sigma(U_o^T V_c), \sigma(-U_{k1}^T V_c) \dots \sigma(-U_{kk}^T V_c)]$

- $loss = -\log(\sigma(U_o^T V_c)) - \sum_{k=1}^{K} \log(\sigma(-U_k^T V_c))$

先score[1:] *= -1

~~不从0取为了~~

再sum

- $\frac{\partial J}{\partial V_c} = [1 - \sigma(U_o^T V_c)] \cdot (-U_o) + \sum_{k=1}^{K} (1 - \sigma(-U_k^T V_c)) U_k$

后来score = [$(1 - \sigma(U_o^T V_c))$, $1 - \sigma(-U_k^T V_c)$ …]

score[0] *= -1. 为了这个.

再统一：score · U[0, k]

- $\frac{\partial J}{\partial U} =$ score · V_c 4×10

(4×1, 1×10)

将4×10的4个不同处的梯度值加到原来的梯度那些上

③ skip-gram_loss_∂J/∂ (见f)

loss是w窗口加起来所有的J(θ). naive Softmax.

SGD：注意学习率退火

Note: ☆☆☆☆☆

$$\frac{\partial \sum_{w\in V} \exp(U w^T v_c)}{\partial U_w} \quad \text{与} \quad \frac{\partial \sum_{w\in V} \exp(U w^T v_c)}{\partial v_c}$$

$$= \exp(U w^T v_c)\cdot v_c \qquad = \sum_{w\in V}\left[\exp(U w^T v_c)\cdot U_w\right]$$

$$\Downarrow \qquad \Downarrow$$

$$\frac{\partial\left(e^{5u_1}+e^{5u_2}+e^{5u_3}\right)}{\partial([u_1,u_2,u_3])} \qquad \frac{\partial\left(e^{u_1 x}+e^{u_2 x}+e^{u_3 x}\right)}{\partial x}$$

$$= \left(5e^{5u_1}, 5e^{5u_2}, 5e^{5u_3}\right) \qquad = u_1 e^{u_1 x}+u_2 e^{u_2 x}+e^{u_3 x}\cdot u_3.$$

run.py 代码思路整理

① 初始数据

dataset加载.

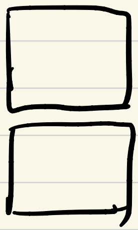
tokens 存储所有词—数字索引映射

nwords 是词汇总数.

dimVectors 词向量是10维的.

contextSize 上下文的界限5个. (窗口大小)

wordVectors.



U (40) nwords × dimvectors.

V (0)

② 训练数据 sgd, 返回训练好的wordvectors

f: word2vec_sgd_wrapper 得到 loss 和 gradient.

(word2vecModel, word2Ind, wordVectors, dataset,

skip-gram　tokens　vec　dataset.

windowsize, word2vecLossAndGradient)

C=5　neg sample.

x0: wordvectors.　③ 可视化.

step: 0.3.　iters: 40000　每10次打印, 5000次保存.

sgd.py 没什么好说的

word2vec.py

word2vec_sgd_wrapper

① 初始化

切开wordvectors U V

loss=0.
grad=0
N=nwordsx2

② batchsize=50

对batch中的每个样本:

随机一个window_size

找到其对应的中心词, 上下文

先随机挑个句子
再随机找个中心词
利用window找上下文,

利用skip-gram得到

loss & grad(Vc) & grad(U) 并更新.

要/batchsize.

☆ 所以说, 并没有按序找每个词为中心词, 而是随机挑句子, 随机挑中心词.

Amazing! ^^

<否则将会是一个庞大的工程, 计算量可能爆炸>