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.

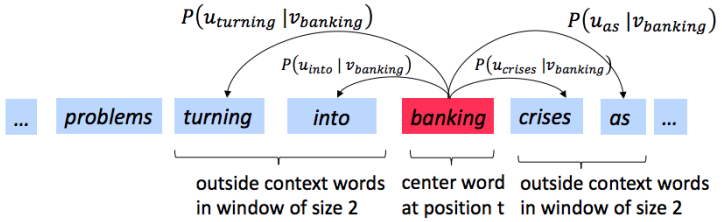


Figure 1: The word2vec skip-gram prediction model with window size 2

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.

(b)번 풀이과정 : 손실함수 v_c로 미분하기 { 출처 리포고불로그

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

cross entropy

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

$$-\log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)} = J \text{ 일때}$$

$$\frac{\partial}{\partial v_c} J = -\left[\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)}\right]$$

$$= -\frac{\partial}{\partial v_c}\left[u_o^T v_c - \log \sum_{w=1}^{W} \exp(u_w^T v_c)\right]$$

(y, $\hat{y}$)

왜 이렇게 되는지 잘 모르겠다

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

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

$$= -u_o^T + \sum_{w=1}^{W} p(w|c)\cdot u_w$$

$$= -u_o^T + \sum_{w=1}^{W} p(w|c)\cdot u_w$$

이때 이것을 $y, U, \hat{y}$ 으로 표현하면

u_o는 $U \times y$로 이는 y가 실제 주변단어인 애들만 1이고 나머지는 0인 원핫벡터이기 때문이다

또 $\sum_{w=1}^{W} p(w|c)\cdot u_w$ 는 c가 중심단어일때 모든 w에 대한 조건부확률을 구한값인 $\hat{y}$과 U의 가중합으로 $U\cdot\hat{y}$으로 표현할수있다

따라서 답은 $\frac{\partial J}{\partial v_c} = U(\hat{y}-y)$ 이다

(c)번 풀이과정 : 손실함수 u_w로 미분

$$-\log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)} = J \text{ 일때}$$

$$\frac{\partial}{\partial u_o} J = -\left[\frac{\partial}{\partial u_o} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)}\right]$$

$$= -\frac{\partial}{\partial u_o}\left[u_o^T v_c - \log \sum_{w=1}^{W} \exp(u_w^T v_c)\right]$$

트랜스포즈 붙여야하는데?

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

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

$$= -v_c + \sum_{w=1}^{W} p(w|c)\cdot v_c$$

$$= -v_c + \sum_{w=1}^{W} p(w|c)\cdot v_c$$

미분과정을 이해 제대로 못해서 대략해본. 맞나?

문맥단어 $w=0$인경우 위와 같은 논리로 $(\hat{y}-y)\, v_c$ 이며 $w\neq 0$인경우 원핫벡터로 표현된 y값은 0이기 때문에 $\hat{y}_w$로 표현 가능.

따라서 $\hat{y}_w v_c$가 된다

(d)번 : 시그모이드 미분하기

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

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

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

겉미분 속미분

$$= (-1)\cdot(1+e^{-x})^{-2}\, e^{-x}(-1)$$

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

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

$$= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) = \sigma(1-\sigma)$$

(e) Nagative Sampling

소프트맥스시 분모에서 중심단어와 나머지 모든 단어에 대한 내적을 하게 되는데 연산량이 어마어마함. 따라서 일부단어만 뽑아서 계산하는데 이것이 N·S임

⇒ 사용자가 지정한 window size에 등장하지 않는 단어를 5-20개 뽑고 이를 정답과 합쳐서 전체 단어인 것처럼 소프트맥스를 구하는것

뽑힐확률은 다음과 같이 정의됨 이에 따른 N·S Loss function은 다음과 같다

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}} \qquad J = -\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K}\log(\sigma(-u_k^T v_c))$$

K개의 단어 w에 대한 Outside vector가 $u_1 \ldots u_k$ 일때

아웃사이드 워드는 샘플링 X 오직 중심단어에 대해서만 진행

1번: v_c에대해미분

* $\sigma'(x) = \sigma(1-\sigma)$

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

$y = \ln|f(x)|$ 이면

$$y' = \frac{f'(x)}{f(x)} \qquad = \frac{\frac{\partial}{\partial v_c}\sigma(u_o^T v_c)}{\sigma(u_o^T v_c)} - \sum_{k=1}^{K}\frac{\frac{\partial}{\partial v_c}\sigma(-u_k^T v_c)}{\sigma(-u_k^T v_c)}$$

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

$$= \sigma(u_o^T v_c)(1-\sigma(u_o^T v_c))\cdot u_o$$

겉미분 속미분

2번: u_o에 대해 미분 / 3번: u_k에 대해 미분

위와 같은 전개로 표현될수있다

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

이때 문맥단어 O와 K는 절대 겹치는것이 없는 애들이기 때문에 각각 상수처리되어 미분시 사라지므로 다음과 같이 미분값을 구할수있다

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

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

이는 모든 단어에 대한값이 아닌 K에만 구하면 되기때문에 계산이 간편하다

(f)번: Skip-gram

$$J_{\text{skip-gram}}(v_c, w_{t-m}, \ldots w_{t+m}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} J(v_c, w_{t+j}, U)$$

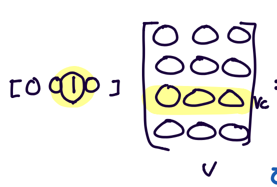
(i) $\partial J_{\text{skip-gram}}(v_c, w_{t-m}, \ldots w_{t+m}, U)/\partial U$

$$\sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial U} J(v_c, w_{t+j}, U)$$

(ii) $\partial J_{\text{skip-gram}}(v_c, w_{t-m}, \ldots w_{t+m}, U)/\partial v_c$

$$\sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial v_c} J(v_c, w_{t+j}, U)$$

(iii) $\partial J_{\text{skip-gram}}(v_c, w_{t-m}, \ldots w_{t+m}, U)/\partial v_w$ when $w \ne c$



세번째 단어를 입력으로 하는 은닉층은 다음과 같이 계산됨

가중치행렬에서 해당행벡터를 참조해오는 식으로 진행. 따라서 v_c가아닌 다른 주변 애들은 업데이트 되지 않는다.

(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]".