CS 224n Assignment #2: word2vec (49 Points)

Due on Tuesday Jan. 23, 2024 by **4:30pm (before class)**
Please do not forget to tag questions on Gradescope

1 Understanding word2vec (31 points)

Recall that the key insight behind word2vec is that *'a word is known by the company it keeps'*. Concretely, consider a 'center' word c surrounded before and after by a context of a certain length. We term words in this contextual window 'outside words' (O). For example, in Figure 1, the context window length is 2, the center word c is 'banking', and the outside words are 'turning', 'into', 'crises', and 'as':
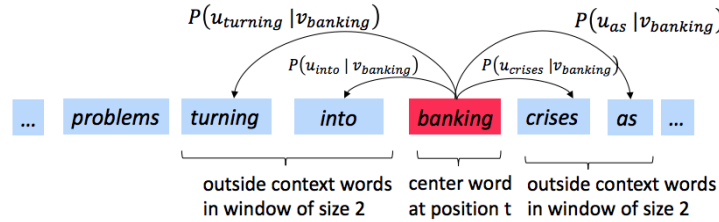

图 1: The word2vec skip-gram prediction model with window size 2

Skip-gram word2vec aims to learn the probability distribution $P(O|C)$. Specifically, given a specific word o and a specific word c, we want to predict $P(O = o|C = c)$: the probability that word o is an 'outside' word for c (i.e., that it falls within the contextual window of c). We model this probability by taking the softmax function over a series of vector dot-products:

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

For each word, we learn vectors u and v, where $\boldsymbol{u}_o$ is the 'outside' vector representing outside word o, and $\boldsymbol{v}_c$ is the 'center' vector representing center word c. We store these parameters in 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:

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

We can view this loss as the cross-entropy[2] between the true distribution $\boldsymbol{y}$ and the predicted distribution $\hat{\boldsymbol{y}}$, for a particular center word c and a particular outside word o. Here, both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are vectors with length equal to the number of words in the vocabulary. Furthermore, the kth entry in these vectors indicates the conditional probability of the kth 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, for this particular example of center word c and outside word o.[3] The predicted distribution $\hat{\boldsymbol{y}}$ is the probability distribution $P(O|C = c)$ given by our model in equation (1).

---

[1] Assume that every word in our vocabulary is matched to an integer number k. Bolded lowercase letters represent vectors. $\boldsymbol{u}_k$ is both the kth column of $\boldsymbol{U}$ and the 'outside' word vector for the word indexed by k. $\boldsymbol{v}_k$ is both the kth 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 word k and the index of word k.**

[2] The **cross-entropy loss** between the true (discrete) probability distribution p and another distribution q is $-\sum_i p_i \log(q_i)$.

[3] Note that the true conditional probability distribution of context words for the entire training dataset would not be one-hot.

Note: Throughout this homework, when computing derivatives, please use the method reviewed during the lecture (i.e. no Taylor Series Approximations).

(a) (2 points) Prove that the naive-softmax loss (Equation 5) is the same as the cross-entropy loss between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, i.e. (note that $\boldsymbol{y}$ (true distribution), $\hat{\boldsymbol{y}}$ (predicted distribution) are vectors and $\hat{\boldsymbol{y}}_o$ is a scalar):

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

Your answer should be one line. You may describe your answer in words.

Answer: 真实的经验分布 y 是一个 one-hot 向量，只有真实的 outside word o 为 1，其余地均为 0；预测的分布 $\hat{y}$ 是一个概率分布，通过等式26来给定：$P(O|C = c)$ 因此，

$$
\begin{aligned}
J_{cross-entropy} &= -\sum_{w \in \text{Vocab}} y_w log(\hat{y}_w) \\
&= -(y_1 log(\hat{y}_1) + y_2 log(\hat{y}_2) + \cdots + y_o log(\hat{y}_o) + \cdots + y_{|V|} log(\hat{y}_{|V|})) \\
&= -(0 * log(\hat{y}_1) + 0 * log(\hat{y}_2) + \cdots + 1 \cdot log(\hat{y}_o) + \cdots + 0 log(\hat{y}_{|V|})) \\
&= -log(\hat{y}_o) = J_{naive-softmax}(v_c, o, U)
\end{aligned}
\tag{4}
$$

于是上述的命题得证。

(b) (6 points)

   (i) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. <u>Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, $\boldsymbol{U}$, and show your work to receive full credit.</u>

   - **Note**: Your final answers for the partial derivative should follow the shape convention: the partial derivative of any function $f(x)$ with respect to x should have the **same shape** as x.[4]

   - Please provide your answers for the partial derivative in vectorized form. For example, when we ask you to write your answers in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$, you may not refer to specific elements of these terms in your final answer (such as $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...).

   (ii) When is the gradient you computed equal to zero?
   Hint: You may wish to review and use some introductory linear algebra concepts.

   (iii) The gradient you found is the difference between the two terms. Provide an interpretation of how each of these terms improves the word vector when this gradient is subtracted from the word vector v_c.

Answer: 首先给出原等式：

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o | C = c). \tag{5}$$

1) 首先根据上式可得：

$$\hat{y}_o = P(O = o | C = c) = \frac{exp(u_o^T v_c)}{\sum_{w \in \text{Vocab}} exp(u_w^T v_c)} \tag{6}$$

2) 因此有：$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -log(\hat{y}_o)$

---

[4]This allows us to efficiently minimize a function using gradient descent without worrying about reshaping or dimension mismatching. While following the shape convention, we're guaranteed that $\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$ is a well-defined update rule.

3) 求导:

$$\begin{aligned}
\frac{\partial J}{\partial v_c} &= -\frac{\partial(u_o^T v_c - log(\sum_w exp(u_w^T v_c)))}{\partial v_c} \\
&= -\frac{\partial u_o^T v_c}{\partial v_c} + \frac{\partial log(\sum_w exp(u_w^T v_c))}{\partial v_c} \\
&= -u_o + \frac{1}{\sum_w exp(u_w^T v_c)} \frac{\partial \sum_w exp(u_w^T v_c)}{\partial v_c} \\
&= -u_o + \frac{1}{\sum_w exp(u_w^T v_c)} \sum_w exp(u_w^T v_c) u_w \\
&= -u_o + \sum_w (\hat{y}_w u_w)
\end{aligned} \quad (7)$$

4) 令:

$$y_w = \begin{cases} 1, & w = o \\ 0, & w \neq o. \end{cases} \quad (8)$$

于是有 $u_o = \sum_{w \in \mathsf{Vocab}} y_w u_w$:

$$\begin{aligned}
&-u_o + \sum_w (\hat{y}_w u_w) \\
&= \sum_{w \in \mathsf{Vocab}} (-y_w u_w + \hat{y}_w u_w) \\
&= \sum_{w \in \mathsf{Vocab}} (-y_w + \hat{y}_w) u_w
\end{aligned} \quad (9)$$

5) 接下来对上式进行向量化，需要有 y 是 one-hot 向量，只在 word o 时取 1，其余为 0

$$\frac{\partial J}{\partial v_c} = \sum_{w \in \mathsf{Vocab}} U(\hat{y} - y) \quad (10)$$

(c) (1 point) In many downstream applications using word embeddings, L2 normalized vectors (e.g. $\mathbf{u}/||\mathbf{u}||_2$ where $||\mathbf{u}||_2 = \sqrt{\sum_i u_i^2}$) are used instead of their raw forms (e.g. $\mathbf{u}$). Let's consider a hypothetical downstream task of binary classification of phrases as being positive or negative, where you decide the sign based on the sum of individual embeddings of the words. When would L2 normalization take away useful information for the downstream task? When would it not?

Hint: Consider the case where $\mathbf{u}_x = \alpha \mathbf{u}_y$ for some words $x \neq y$ and some scalar α. Give examples of words x and y which satisfy the given relation and why would they be affected/not affected due to the normalization?

Answer: L2 正则化保留了向量的方向信息，但是丢失了向量的大小信息。

如题中所描述的若单词 x 是 good, y 是 better, 向量的长度可能影响着词向量的描述"good"的程度，此时就不适宜使用 L2 正则化

而如果单词 x 是 good, y 是 bad，两个向量的长度并不重要，而其代表的"good"的方向相反，此时就适宜使用 L2 正则化。

(d) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\mathrm{naive\text{-}softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ 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 your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$. In this subpart, you may use specific elements within these terms as well (such as $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...). Note that $\boldsymbol{u}_w$ is a vector while $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ are scalars. Show your work to receive full credit.

Answer:

$$
\begin{aligned}
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_w} &= \frac{-log(\hat{y}_o)}{u_w} \\
&= \frac{\partial -log \frac{exp(u_o^T v_c)}{\sum_{w \in \text{Vocab}} exp(u_w^T v_c)}}{\partial u_w} \\
&= -\frac{\partial (u_o^T v_c - log(\sum_{w \in \text{Vocab}} exp(u_w^T v_c)))}{\partial u_w} \\
&= -\frac{\partial u_o^T v_c}{\partial u_w} + \frac{\partial log(\sum_w exp(u_w^T v)_c))}{\partial u_w} \\
&= -\frac{\partial u_o^T v_c}{\partial u_w} + \frac{1}{\sum_w exp(u_w^T v_c)} \cdot \sum_w exp(u_w^T v_c) v_c
\end{aligned}
\tag{11}
$$

对于 $-\frac{\partial u_o^T v_c}{\partial u_w}$ 来讲，其可写为：$-\frac{\partial u_o^T}{\partial u_w} v_c$，其中，$\frac{\partial u_o^T}{\partial u_w} = \begin{cases} 1, w = o \\ 0, w \neq o \end{cases}$ ，而对于 y_w 而言，有 $y_w = \begin{cases} 1, w = o \\ 0, w \neq o \end{cases}$ ，二者相等，所以可以将 $\frac{\partial u_o^T}{\partial u_w} = y_w$，按照 1-b 的样式，可以将其写为 $\sum_w y_w$

故上式最终可化为：

$$
\begin{aligned}
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_w} &= -\frac{\partial u_o^T v_c}{\partial u_w} + \frac{1}{\sum_w exp(u_w^T v_c)} \cdot \sum_w exp(u_w^T v_c) v_c \\
&= -\sum_w y_w v_c + + \frac{1}{\sum_w exp(u_w^T v_c)} \cdot \sum_w exp(u_w^T v_c) v_c \\
&= \sum_w (-y_w v_c + \frac{exp(u_w^T v_c)}{\sum_w exp(u_w^T v_c)} v_c)
\end{aligned}
\tag{12}
$$

而我们有：

$$
\frac{exp(u_w^T v_c)}{\sum_w exp(u_w^T v_c)} = \hat{y}_w
\tag{13}
$$

故最终得到：

$$
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_w} = \sum_w (-y_w v_c + \hat{y}_w v_c) = \sum_w (\hat{y}_w - y_w) v_c
\tag{14}
$$

对于 y 来讲，它是一个 one-hot 向量且仅在 word o 处为 1，将上式写为 $y, \hat{y}, v_c$ 的项，则为：

$$
\frac{\partial J_{native-softmax}(v_c,o,U)}{U} = \sum_w v_c (\hat{y} - y)^T
\tag{15}
$$

case 1 我们先考虑 $w = o$ 的情况，

$$
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_{w=o}} = v_c (\hat{y}_{w=o} - y_{w=o})
\tag{16}
$$

因为 $y_{w=o} = 1$，则上式可写为：

$$
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_{w=o}} = v_c (\hat{y}_{w=o} - 1)
\tag{17}
$$

case 2 然后再考虑 $w \neq o$ 的情况，此时 $y_{w \neq o} = 0$，则有：

$$
\frac{\partial J_{native-softmax}(v_c,o,U)}{\partial u_{w \neq o}} = v_c \hat{y}_{w \neq o}
\tag{18}
$$

总结如下：

$$
\frac{\partial J}{\partial u_w} = \begin{cases} v_c(\hat{y}_w - 1), & if w = o \\ v_c \hat{y}_w, & if w \neq o \end{cases}
\tag{19}
$$

(e) (1 point) Write down the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{U}$. Please break down your answer in terms of the column vectors $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_1}$, $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_2}$, ..., $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_{|\text{Vocab}|}}$. No derivations are necessary, just an answer in the form of a matrix.

Answer: 首先写出标量对向量求导的结果：

$$\frac{\partial J}{\partial U} = \left[\frac{\partial J}{\partial u_1} \ \frac{\partial J}{\partial u_2} \ \cdots \frac{\partial J}{\partial u_{Vocab}} \right] \tag{20}$$

通过上一问的答案我们可知：

$$\frac{\partial J}{\partial u_w} = \begin{cases} v_c(\hat{y}_w - 1), & if w = o \\ v_c \hat{y}_w, & if w \neq o \end{cases} \tag{21}$$

(f) (2 points) The Leaky ReLU (Leaky Rectified Linear Unit) activation function is given by Equation 22 and Figure 2:
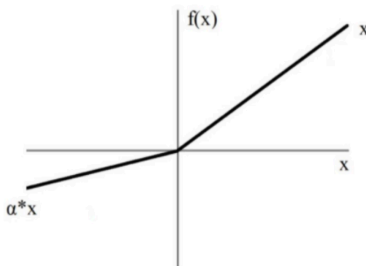
$$f(x) = \max(\alpha x, x) \tag{22}$$

图 2: Leaky ReLU

Where x is a scalar and $0 < \alpha < 1$, please compute the derivative of $f(x)$ with respect to x. You may ignore the case where the derivative is not defined at 0.[5]

Answer: 当 x>0 时，

$$\frac{\partial f(x)}{\partial x} = \frac{\partial x}{x} = 1 \tag{23}$$

当 x<0 时，

$$\frac{\partial f(x)}{\partial x} = \frac{\partial \alpha x}{x} = \alpha \tag{24}$$

综上，

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 1, x > 0 \\ \alpha, x < 0 \end{cases} \tag{25}$$

(g) (3 points) The sigmoid function is given by Equation 26:

---

[5]If you're interested in how to handle the derivative at this point, you can read more about the notion of subderivatives.

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{26}$$

Please compute the derivative of $\sigma(x)$ with respect to x, where x is a scalar. Please write your answer in terms of $\sigma(x)$. Show your work to receive full credit.

Answer: 对函数求导：

$$\begin{aligned}
\frac{d\sigma(x)}{dx} &= \frac{e^x}{e^x+1} \\
&= \frac{e^x(e^x+1) - e^x \cdot e^x}{(e^x+1)^2} \\
&= \frac{e^x}{e^x+1}\left(\frac{1+e^x-e^x}{e^x+1}\right) \\
&= \frac{e^x}{e^x+1}\left(1 - \frac{e^x}{e^x+1}\right) \\
&= \sigma(x)(1 - \sigma(x))
\end{aligned} \tag{27}$$

(h) (6 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 $\boldsymbol{u}_{w_1}, \boldsymbol{u}_{w_2}, \ldots, \boldsymbol{u}_{w_K}$.[6] For this question, assume that the K negative samples are distinct. In other words, $i \neq j$ implies $w_i \neq w_j$ for $i, j \in \{1, \ldots, K\}$. 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:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)) \tag{28}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[7]

(i) Please repeat parts (b) and (d), 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 the sth negative sample $\boldsymbol{u}_{w_s}$. Please write your answers in terms of the vectors $\boldsymbol{v}_c$, $\boldsymbol{u}_o$, and $\boldsymbol{u}_{w_s}$, where $s \in [1, K]$. Show your work to receive full credit. **Note:** you should be able to use your solution to part (g) to help compute the necessary gradients here.

(ii) In the lecture, we learned that an efficient implementation of backpropagation leverages the reuse of previously computed partial derivatives. Which quantity could you reuse (we store the common terms in a single matrix/vector) amongst the three partial derivatives calculated above to minimize duplicate computation?

Write your answer in terms of
$\boldsymbol{U}_{o,\{w_1,\ldots,w_K\}} = \left[\boldsymbol{u}_o, -\boldsymbol{u}_{w_1}, \ldots, -\boldsymbol{u}_{w_K}\right]$, a matrix with the outside vectors stacked as columns, and $\mathbf{1}$, a $(K+1) \times 1$ vector of 1's.[8] Additional terms and functions (other than $\boldsymbol{U}_{o,\{w_1,\ldots,w_K\}}$ and $\mathbf{1}$) can be used in your solution.

(iii) Describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss.

---

[6]Note: In the notation for parts (g) and (h), we are using words, not word indices, as subscripts for the outside word vectors.

[7]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.

[8]Note: NumPy will automatically broadcast 1 to a vector of 1's if the computation requires it, so you generally don't have to construct $\mathbf{1}$ on your own during implementation.

Caveat: So far we have looked at re-using quantities and approximating softmax with sampling for faster gradient descent. Do note that some of these optimizations might not be necessary on modern GPUs and are, to some extent, artifacts of the limited compute resources available at the time when these algorithms were developed.

Answer: (i)

$$J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)) \tag{29}$$

对上式子 v_c, u_o 和第 s^{th} 个负样本 u_{w_s}。

首先考虑 v_c:

$$
\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial v_c} &= -\frac{\partial \log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) + \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c))}{\partial v_c} \\
&= -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))u_o \\
&\quad - \sum_{s=1}^{k} \frac{1}{\sigma(-u_{w_s}^T v_c)} \sigma(-u_{w_s}^T v_c)(1 - \sigma(-u_{w_s}^T v_c))(-u_{w_s}^T) \\
&= -(1 - \sigma(u_o^T v_c))u_o + \sum_{s=1}^{k}(1 - \sigma(-u_{w_s}^T v_c))u_{w_s} \\
&= u_o(\sigma(u_o^T v_c) - 1) + \sum_{s=1}^{k} u_{w_s}(1 - \sigma(-u_{w_s}^T v_c))
\end{aligned}
\tag{30}
$$

然后考虑 J 对 u_o 的导数:

$$
\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_o} &= -\frac{\partial \log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) + \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c))}{\partial u_o} \\
&= -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))v_c \\
&\quad - \frac{\partial \sum_{s=1}^{K} log(\sigma(-u_{w_s}^T v_c))}{\partial u_o}
\end{aligned}
\tag{31}
$$

考虑等式的最后一部分 $\frac{\partial \sum_{s=1}^{K} log(\sigma(-u_{w_s}^T v_c))}{\partial u_o}$,我们首先知道 $o \notin \{w_1, \cdots, w_K\}$,也就是说 $\frac{\partial \sum_{s=1}^{K} log(\sigma(-u_{w_s}^T v_c))}{\partial u_o} = 0$,则原式可化为:

$$
\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_o} &= -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))v_c - 0 \\
&= v_c(\sigma(u_o^T v_c) - 1)
\end{aligned}
\tag{32}
$$

然后考虑 J 对 u_{w_s} 的导数:

$$
\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_{w_s}} &= -\frac{\partial \log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) + \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c))}{\partial u_{w_s}} \\
&= \frac{\partial log(\sigma(u_o^T v_c))}{\partial u_{w_s}} - \frac{\sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c))}{\partial u_{w_s}}
\end{aligned}
\tag{33}
$$

对于等式的第一部分 $\frac{\partial log(\sigma(u_o^T v_c))}{\partial u_{w_s}}$ 来讲,我们有 $u_o \notin u_{w_s}$,所以此部分导数为 0,于是上式可化为:

$$
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_{w_s}} = 0 - \frac{\sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c))}{\partial u_{w_s}} \tag{34}
$$

对于等式的第二部分来讲,$o \notin \{w_1, \ldots, w_K\}$,将求和式子展开,只有 $u_{w_s} = u_{w_s}$ 时,第二项导数不为零,求和的其余项对 u_{w_s} 求导均为零,于是能够得到:

$$
\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_{w_s}} &= 0 - (1 - \sigma(-u_{w_s}^T v_c))(-v_c) \\
&= v_c(1 - \sigma(-u_{w_s}^T v_c)), where s \in [1, K]
\end{aligned}
\tag{35}
$$

(ii)

在上述运算的过程中，可以观察到 $\sigma(u_o^T v_c) - 1$ 和 $(1 - \sigma(-u_{w_s}^T v_c))$ 形式类似，可以通过矩阵运算来获得一次之后重复使用，结合矩阵 $U_{o,\{w_1,\dots,w_K\}} = [u_o, -u_{w_1}, \cdots, -u_{w_K}]$ 的定义，于是得到：

$$\sigma(U^T v_c) - 1 = \begin{bmatrix} \sigma(u_o^T v_c) - 1 \\ \sigma(u_{w_1}^T v_c) - 1 \\ \vdots \\ \sigma(u_{w_K}^T v_c) - 1 \end{bmatrix} \tag{36}$$

(iii) negative sampling loss 只使用了 K 个负样本进行计算

native softmax loss 是在整个词汇表 $|V|$ 上来进行计算的。

所以 negative sampling loss 的计算效率要更高一些。

(i) (2 points) Now we will repeat the previous exercise, but without the assumption that the K sampled words are distinct. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \dots, w_K$ and their outside vectors as $\boldsymbol{u}_{w_1}, \dots, \boldsymbol{u}_{w_K}$. In this question, you may not assume that the words are distinct. In other words, $w_i = w_j$ may be true when $i \neq j$ is true. Note that $o \notin \{w_1, \dots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)) \tag{37}$$

for a sample $w_1, \dots w_K$, where $\sigma(\cdot)$ is the sigmoid function.

Compute the partial derivative of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to a negative sample $\boldsymbol{u}_{w_s}$. Please write your answers in terms of the vectors $\boldsymbol{v}_c$ and $\boldsymbol{u}_{w_s}$, where $s \in [1, K]$. Show your work to receive full credit. Hint: break up the sum in the loss function into two sums: a sum over all sampled words equal to w_s and a sum over all sampled words not equal to w_s. Notation-wise, you may write 'equal' and 'not equal' conditions below the summation symbols, such as in Equation 44.

Answer: 这部分内容不再假设抽样的样本中词向量是不同的，也就是说有可能出现词向量相同的情况。首先是对 v_c 的导数

$$\begin{aligned} \frac{\partial \boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_{w_s}} &= \frac{\partial(-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)))}{\partial u_{w_s}} \\ &= \frac{\partial(-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c))}{\partial u_{w_s}} - \frac{\partial \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)))}{\partial u_{w_s}} \end{aligned} \tag{38}$$

在上述公式中，由于 $o \notin \{w_1, \dots, w_K\}$，故第一项的导数为：

$$\frac{\partial(-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c))}{\partial u_{w_s}} = 0 \tag{39}$$

将第二项的导数中求和部分分为两部分，第一部分为 $w_i = w_s$ 的部分，第二部分为 $w_i \neq w_s$ 的部分，则求和可以分为两部分 $\sum_{i=1:K且w_i=w_s}$ 和 $\sum_{i=1:K且w_i\neq w_s}$：

$$-\frac{\partial \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)))}{\partial u_{w_s}} = -\frac{\sum_{i=1:K且w_i=w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} - \frac{\sum_{i=1:K且w_i\neq w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} \tag{40}$$

对于 $\sum_{i=1:K\text{且}w_i \neq w_s}$ 此部分来讲，其求导得到的结果为 0，则上式子可继续推导为：

$$\frac{\sum_{i=1:K\text{且}w_i \neq w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} = 0 \tag{41}$$

和

$$
\begin{aligned}
\frac{\sum_{i=1:K\text{且}w_i = w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} &= \sum_{i=1:K\text{且}w_i = w_s} \frac{1}{\sigma(-u_{w_i}^T v_c)} \cdot \sigma(-u_{w_i}^T v_c)(1 - \sigma(-u_{w_i}^T v_c))(-v_c) \\
&= \sum_{i=1:K\text{且}w_i = w_s} (1 - \sigma(-u_{w_i}^T v_c))(-v_c)
\end{aligned} \tag{42}
$$

则原式为：

$$
\begin{aligned}
\frac{\partial \boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial u_{w_s}} &= \frac{\partial(-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^K \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)))}{\partial u_{w_s}} \\
&= \frac{\partial(-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)))}{\partial u_{w_s}} - \frac{\sum_{i=1:K\text{且}w_i = w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} - \frac{\sum_{i=1:K\text{且}w_i \neq w_s} \partial log(\sigma(-u_{w_i}^T v_c))}{\partial u_{w_s}} \\
&= 0 - 0 + \sum_{i=1:K\text{且}w_i = w_s} v_c(1 - \sigma(-u_{w_i}^T v_c)) \\
&= \sum_{i=1:K\text{且}w_i = w_s} v_c(1 - \sigma(-u_{w_i}^T v_c))
\end{aligned} \tag{43}
$$

(j) (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:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{44}$$

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

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

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

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

Answer: (i)

$$\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{U}} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial U} \tag{45}$$

(ii)

$$\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_c} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial v_c} \tag{46}$$

(iii)

$$\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_w}\Bigg|_{w \neq c} = 0 \tag{47}$$

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 $\boldsymbol{J}_{skip\text{-}gram}$ with respect to all parameters. You're ready to implement word2vec!*

2   Implementing word2vec (18 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. **Windows users** may wish to install the Linux Windows Subsystem[9]. Also note that you probably want to finish the previous math section before writing the code since you will be asked to implement the math functions in Python. You'll probably want to implement and test each part of this section in order, since the questions are cumulative.

    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

For each of the methods you need to implement, we included approximately how many lines of code our solution has in the code comments. These numbers are included to guide you. You don't have to stick to them, you can write shorter or longer code as you wish. If you think your implementation is significantly longer than ours, it is a signal that there are some numpy methods you could utilize to make your code both shorter and faster. for loops in Python take a long time to complete when used over large arrays, so we expect you to utilize numpy methods. We will be checking the efficiency of your code. You will be able to see the results of the autograder when you submit your code to Gradescope, we recommend submitting early and often.
Note: If you are using Windows and have trouble running the .sh scripts used in this part, we recommend trying Gow or manually running commands in the scripts.

(a) (12 points) We will start by implementing methods in word2vec.py. You can test a particular method by running python word2vec.py m where m is the method you would like to test. For example, you can test the sigmoid method by running python word2vec.py sigmoid.

  (i) Implement the sigmoid method, which takes in a vector and applies the sigmoid function to it.

  (ii) Implement the softmax loss and gradient in the naiveSoftmaxLossAndGradient method.

  (iii) Implement the negative sampling loss and gradient in the negSamplingLossAndGradient method.

  (iv) Implement the skip-gram model in the skipgram method.

---

[9]https://techcommunity.microsoft.com/t5/windows-11/how-to-install-the-linux-windows-subsystem-in-windows-11/m-p/2701207

When you are done, test your entire implementation by running python word2vec.py.

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

(c) (2 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 and the compute power of your machine **(an efficient implementation takes one to two hours)**. *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.** In at most three sentences, briefly explain what you see in the plot. This may include, but is not limited to, observations on clusters and words that you expect to cluster but do not.

Answer:

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]".