Stanford CS224n Assignment 2

Aman Chadha

January 23, 2021

1 Written: Understanding word2Vec (23 points)

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

$$-\sum_{w \in Vocab} y_w \log \hat{y}_w = -\log \hat{y}_o$$

Your answer should be one line.

Answer: The true empirical distribution (i.e., the ground truth) y is a one-hot vector where $y_w = 1$ when $w = o$ and $y_w = 0$ when $w \neq o$. Mathematically,

$$y_w = \begin{cases} 1 \text{ if w = o} \\ 0 \text{ if w} \neq \text{ o} \end{cases}$$

As such, considering the cross-entropy loss $J_{cross-entropy}$,

$$\begin{aligned} J_{cross-entropy}(y, \hat{y}) &= -\sum_{w \in \text{Vocab}} y_w \log(\hat{y}_w) \\ &= -\left(y_1 \log(\hat{y}_1) + \cdots + y_o \log(\hat{y}_o) + \cdots + y_{|V|} \log(\hat{y}_{|V|})\right) \\ &= -y_o \log(\hat{y}_o) \\ &= -\log(\hat{y}_o) \\ &= -\log(P(O = o|C = c)) \\ &= \boxed{J_{naive-softmax}(v_c, o, U)} \end{aligned}$$

Thus, the naive-softmax loss $J_{naive-softmax}$ given in Equation (2) is the same as the cross-entropy loss $J_{cross-entropy}$.

(b) (5 points) Compute the partial derivative of $J_{naive-softmax}(v_c, o, U)$ with respect to v_c. Please write your answer in terms of y, $\hat{y}$, and U. Note that in this course, we expect your final answers to follow the shape convention.[1] This means that the partial derivative of any function $f(x)$ with respect to x should have the same shape as x. For this subpart, please present your answer in vectorized form. In particular, you may not refer to specific elements of $y, \hat{y}$, and U in your final answer (such as $y_1, y_2, \ldots$).

Answer:

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} &= \frac{\partial}{\partial v_c}\left(-\log(\hat{y}_o)\right) \\ &= \frac{\partial}{\partial v_c}\left(-\log\frac{exp(u_o^T \cdot v_c)}{\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)}\right)\end{aligned}$$

Applying the divisive property of logarithms, i.e., $\log(\frac{x}{y}) = \log(x) - \log(y)$,

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} &= \frac{\partial}{\partial v_c}\left[-\left(\log(exp(u_o^T \cdot v_c) - \log\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right)\right] \\ &= \frac{\partial}{\partial v_c}\left[-\log(exp(u_o^T \cdot v_c) + \log\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right] \\ &= \frac{\partial}{\partial v_c}\left[-u_o^T \cdot v_c + \log\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right] \\ &= -\frac{\partial}{\partial v_c}u_o^T \cdot v_c + \frac{\partial}{\partial v_c}\left[\log\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right]\end{aligned}$$

Since $\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a$, we have $\frac{\partial}{\partial v_c}(u_o^T \cdot v_c) = u_o$,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = -u_o + \frac{\partial}{\partial v_c}\left[\log\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right]$$

Using the chain rule of derivatives on log,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = -u_o + \frac{1}{\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \frac{\partial}{\partial v_c}\left[\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)\right]$$

Since the derivative of a sum is the sum of derivatives,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = -u_o + \frac{1}{\sum_{w\in\text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \sum_{w\in\text{Vocab}}\left[\frac{\partial}{\partial v_c}\left(exp(u_w^T \cdot v_c)\right)\right]$$

[1] 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.

Since (i) $\frac{\partial exp(x)}{\partial x} = exp(x)$ and (ii) $\frac{\partial exp(a \cdot x)}{\partial x} = exp(a \cdot x) \cdot a$ per the chain rule,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = -u_o + \frac{1}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)} \sum_{w \in \text{Vocab}} \left(exp(u_w^T \cdot v_c) \cdot u_w \right)$$

Since $\frac{exp(u_w^T \cdot v_c)}{\sum_{w' \in \text{Vocab}} exp(u_{w'}^T \cdot v_c)}$ is the conditional probability distribution $\hat{y}_w$ per word2vec,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = -u_o + \sum_{w \in \text{Vocab}} \hat{y}_w u_w$$

Since $y_w = \begin{cases} 1 \text{ if w = o} \\ 0 \text{ if w} \neq \text{o} \end{cases}$, we can write u_o^T in the above equation as $\sum_{w \in \text{Vocab}} y_w u_w$. As such,

$$\begin{aligned} \frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} &= -\sum_{w \in \text{Vocab}} y_w u_w + \hat{y}_w u_w \\ &= \sum_{w \in \text{Vocab}} [-y_w u_w + \hat{y}_w u_w] \\ &= \sum_{w \in \text{Vocab}} u_w(-y_w + \hat{y}_w) \\ &= \sum_{w \in \text{Vocab}} u_w(\hat{y}_w - y_w) \end{aligned}$$

Note that y is a 1-hot vector with a 1 at word o. Vectorizing the above equation in terms of y, $\hat{y}$, and U,

$$\boxed{\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} = U(\hat{y} - y)}$$

(c) (5 points) Compute the partial derivatives of $J_{naive-softmax}(v_c, o, u)$ 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. In this subpart, you may use specific elements within these terms as well, such as $(y_1, y_2, \ldots)$.

Answer:

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} &= \frac{\partial}{\partial u_w}\left(-\log(\hat{y}_o)\right) \\ &= \frac{\partial}{\partial u_w}\left(-\log \frac{exp(u_o^T \cdot v_c)}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)}\right)\end{aligned}$$

Applying the divisive property of logarithms, i.e., $\log(\frac{x}{y}) = \log(x) - \log(y)$,

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} &= \frac{\partial}{\partial u_w}\left[-\left(\log(exp(u_o^T \cdot v_c) - \log \sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)\right)\right] \\ &= \frac{\partial}{\partial u_w}\left[-\log(exp(u_o^T \cdot v_c) + \log \sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)\right] \\ &= \frac{\partial}{\partial u_w}\left[-u_o^T \cdot v_c + \log \sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)\right] \\ &= -\frac{\partial u_o^T}{\partial u_w} \cdot v_c + \frac{\partial}{\partial u_w}\left[\log \sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)\right] \\ &= -\frac{\partial u_o^T}{\partial u_w} \cdot v_c + \frac{1}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \frac{\partial}{\partial u_w}\left[\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)\right]\end{aligned}$$

Since the derivative of a sum is the sum of derivatives,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} = -\frac{\partial u_o^T}{\partial u_w} \cdot v_c + \frac{1}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \sum_{w \in \text{Vocab}}\left[\frac{\partial}{\partial u_w}\left(exp(u_w^T \cdot v_c)\right)\right]$$

Since (i) $\frac{\partial exp(x)}{\partial x} = exp(x)$ and (ii) $\frac{\partial exp(a \cdot x)}{\partial x} = exp(a \cdot x) \cdot a$ per the chain rule,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} = -\frac{\partial u_o^T}{\partial u_w} \cdot v_c + \frac{1}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \sum_{w \in \text{Vocab}}\left(exp(u_w^T \cdot v_c) \cdot v_c\right)$$

Now, since $\frac{\partial u_o^T}{\partial u_w} = \begin{cases} 1 \text{ if w = o} \\ 0 \text{ if w} \neq \text{o} \end{cases}$, and also that $\frac{\partial u_o^T}{\partial u_w}$ is equivalent to $y_w = \begin{cases} 1 \text{ if w = o} \\ 0 \text{ if w} \neq \text{o} \end{cases}$, we can write $\frac{\partial u_o^T}{\partial u_w}$ in the above equation as $\sum_{w \in \text{Vocab}} y_w$. As such,

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} &= -\sum_{w \in \text{Vocab}} y_w v_c + \frac{1}{\sum_{w \in \text{Vocab}} exp(u_w^T \cdot v_c)} \cdot \sum_{w \in \text{Vocab}}\left(exp(u_w^T \cdot v_c) \cdot v_c\right) \\ &= \sum_{w \in \text{Vocab}}\left[-y_w v_c + \frac{exp(u_w^T \cdot v_c)}{\sum_{w' \in \text{Vocab}} exp(u_{w'}^T \cdot v_c)} \cdot v_c\right]\end{aligned}$$

Since $\frac{exp(u_w^T \cdot v_c)}{\sum_{w' \in \text{Vocab}} exp(u_{w'}^T \cdot v_c)}$ is the conditional probability distribution $\hat{y}_w$ per word2vec,

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_w} &= \sum_{w \in \text{Vocab}} (-y_w v_c + \hat{y_w} v_c) \\ &= \sum_{w \in \text{Vocab}} v_c \left(\hat{y_w} - y_w\right)\end{aligned}$$

(Optional) Given that y is a 1-hot vector with a 1 at word o, upon vectorizing the above equation in terms of y, $\hat{y}$, and v_c, the above equation can be equivalently rewritten as:

$$\frac{\partial J}{\partial U} = v_c(\hat{y} - y)^\top$$

Note that given that both v_c and the difference $(\hat{y} - y)$ are column vectors, to dimension balance, we transpose the second column vector in the above equation.

Case 1. Consider the case where $w = o$, i.e., the context word is the true 'outside' word.

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_{w=o}} = v_c \left(\hat{y_{w=o}} - y_{w=o}\right)$$

Since $y_{w=o} = 1$,

$$\boxed{\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_{w=o}} = v_c \left(\hat{y_{w=o}} - 1\right)}$$

Case 2. Now, consider the case where $w \neq o$, i.e., the context word is not the true 'outside' word,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_{w \neq o}} = v_c \left(\hat{y_{w \neq o}} - y_{w \neq o}\right)$$

Since $y_{w \neq o} = 0$,

$$\boxed{\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial u_{w \neq o}} = v_c \hat{y_{w \neq o}}}$$

Consolidating case 1 and 2,

$$\frac{\partial J}{\partial u_w} = \begin{cases} v_c \left(\hat{y}_w - 1\right) & \text{if } w = o \\ v_c \hat{y}_w & \text{otherwise} \end{cases}$$

(d) (1 point) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}\ (\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to U. Please write your answer in terms of $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_1}, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_2}, \ldots, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_{|\text{Vocabl}}}$. The solution should be one or two lines long.

Answer:
The derivative of a scalar y by a matrix A is given by,

$$\frac{\partial y}{\partial A_{m\times n}} = \begin{bmatrix} \frac{\partial y}{\partial A_{11}} & \frac{\partial y}{\partial A_{12}} & \cdots & \frac{\partial y}{\partial A_{1n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial A_{m1}} & \frac{\partial y}{\partial A_{m2}} & \cdots & \frac{\partial y}{\partial A_{mn}} \end{bmatrix}$$

Given u_w represents the vector for 'outside' word w, the derivative of $J_{naive-softmax}$ (which is a scalar) by U (which is a matrix) is,

$$\frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial U} = \begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} & \cdots & \frac{\partial J}{\partial u_{|V|}} \end{bmatrix}$$

where,

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

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

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

Please compute the derivative of $\sigma(x)$ with respect to x, where x is a scalar. Hint: you may want to write your answer in terms of $\sigma(x)$.

Answer:

$$\begin{aligned}
\frac{\partial \sigma}{\partial x} &= \frac{\partial}{\partial x}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{\partial}{\partial x}\left[\left(1+e^{-x}\right)^{-1}\right] \\
&= \left[-\left(1+e^{-x}\right)^{-2}\right]\left[-e^{-x}\right] \\
&= \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \\
&= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right) \\
&= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}+1-1}{1+e^{-x}}\right) \\
&= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}+1}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \boxed{\sigma(x)(1-\sigma(x))}
\end{aligned}$$

(f) (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 $w_1, w_2, \cdots, w_K$ and their outside vectors as $\mathbf{u_1}, \cdots, \mathbf{u_K}$. For this question, assume that the K negative samples are distinct. In other words, $i \neq j$ implies $w_i \neq w_j$ for $i, j \in \{1, ..., K\}$. Note that $o \notin \{w_1, ..., w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$J_{neg-sample}(v_c, o, U) = -\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log \sigma(-u_k^T v_c))$$

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

Please repeat parts (b) and (c), computing the partial derivatives of $J_{neg-sample}$ with respect to v_c, with respect to u_o, and with respect to a negative sample u_k. Please write your answers in terms of the vectors u_o, v_c, and 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 (e) to help compute the necessary gradients here.

Answer:

The loss function contains two terms:

(i) First term: the log of the probability that the center word and true outside word came from the corpus data.

(ii) Second term: the sum of logs of the probabilities that the center word and outside context words did not come from the corpus data.

(1) Computing the partial derivatives of $J_{neg-sample}$ w.r.t. v_c, the word vector of the 'center' word c.

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}(v_c, o, U)}{\partial v_c} &= \frac{\partial}{\partial v_c}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right] \\
&= \frac{\partial}{\partial v_c}\left[-\log\left(\sigma\left({u_o}^T v_c\right)\right)\right] - \frac{\partial}{\partial v_c}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-{u_k}^T v_c\right)\right)\right] \\
&= \frac{\partial}{\partial v_c}\left[-\log\left(\sigma\left({u_o}^T v_c\right)\right)\right] - \sum_{k=1}^{K}\left[\frac{\partial}{\partial v_c} \log\left(\sigma\left(-{u_k}^T v_c\right)\right)\right] \\
&= \left(-\frac{1}{\sigma\left(u_o T_{v_c}\right)}\right)\left(u_o \sigma\left(u_o^T v_c\right)\left(1 - \sigma\left(u_o^T v_c\right)\right)\right) - \sum_{k=1}^{K} \frac{-u_k \sigma\left(-u_k^T v_c\right)\left(1 - \sigma\left(-u_k^T v_c\right)\right)}{\sigma\left(-u_k^T v_c\right)} \\
&= -u_o\left(1 - \sigma\left(u_o^T v_c\right)\right) + \sum_{k=1}^{K} u_k\left(1 - \sigma\left(-u_k^T v_c\right)\right) \\
&= \boxed{u_o(\sigma(u_o^T v_c) - 1) - \sum_{k=1}^{K} u_k\left(\sigma(-u_k^T v_c) - 1\right)}
\end{aligned}$$

(2) Computing the partial derivatives of $J_{neg-sample}$ w.r.t. u_o, the word vector of the 'outside' word o.

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_o} &= \frac{\partial}{\partial u_o}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right] \\
&= \frac{\partial}{\partial u_o}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] - \frac{\partial}{\partial u_o}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]
\end{aligned}$$

Since $o \notin \{w_1, ..., w_K\}$, $\frac{\partial}{\partial u_o}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right] = 0$. As such,

$$\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_o} = \frac{\partial}{\partial u_o}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] - 0$$

Using (e), $\frac{\partial \sigma}{\partial x} = \sigma(x)(1-\sigma(x))$. So,

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_o} &= -\left[\frac{1}{\sigma\left(u_o^T v_c\right)}\right]\left[\sigma\left(u_o^T v_c\right)\left(1-\sigma\left(u_o^T v_c\right)\right) v_c\right] \\
&= -v_c\left(1-\sigma(u_o^T v_c)\right) \\
&= \boxed{v_c\left(\sigma(u_o^T v_c) - 1\right)}
\end{aligned}$$

(3) Computing the partial derivatives of $J_{neg-sample}$ w.r.t. u_k, the word vector of one of the K negative samples.

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_k} &= \frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right] \\
&= \frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] - \frac{\partial}{\partial u_k}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]
\end{aligned}$$

Since $o \notin \{w_1, ..., w_K\}$, $\frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] = 0$. As such,

$$\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_k} = 0 - \frac{\partial}{\partial u_k}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]$$

Since the derivative of a sum is the sum of derivatives,

$$\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_k} = \sum_{k=1}^{K}\left[\frac{\partial}{\partial u_k} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]$$

Now, derivatives of all of the terms with u_w where $w \neq k$ are 0, while only the term with u_w where $w = k$ remains. Also, using (e), $\frac{\partial \sigma}{\partial x} = \sigma(x)(1-\sigma(x))$. This yields,

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_k} &= -\frac{-v_c \sigma\left(-u_k^T v_c\right)\left(1-\sigma\left(-u_k^T v_c\right)\right)}{\sigma\left(-u_k^T v_c\right)} \\
&= v_c\left(1-\sigma\left(-{u_k}^T v_c\right)\right) \\
&= \boxed{-v_c\left(\sigma(-u_k^T v_c) - 1\right), \forall\, k \in [1, K]}
\end{aligned}$$

The negative sampling loss function is much more compute and memory efficient because it takes into account only K sampled word vectors ($O(K)$) whereas the naive softmax loss utilizes all the word vectors in the entire vocabulary ($O(|V|)$) to normalize the unnormalized probabilities.

(g) (2 point) 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 $w_1, w_2, \ldots, w_K$ and their outside vectors as $u_1, \ldots, u_K$. 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 $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$J_{\text{neg-sample}}\ (v_c, o, U) = -\log\left(\sigma\left(u_o^\top v_c\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-u_k^\top v_c\right)\right)$$

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

Compute the partial derivative of $J_{\text{neg-sample}}$ with respect to a negative sample u_k. Please write your answers in terms of the vectors v_c and u_k, where $k \in [1, K]$. Hint: break up the sum in the loss function into two sums: a sum over all sampled words equal to u_k and a sum over all sampled words not equal to u_k.

Answer:

$$\begin{aligned}\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} &= \frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right] \\ &= \frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] - \frac{\partial}{\partial u_k}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]\end{aligned}$$

Since $o \notin \{w_1, ..., w_K\}$, $\frac{\partial}{\partial u_k}\left[-\log\left(\sigma\left(u_o^T v_c\right)\right)\right] = 0$. As such,

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = 0 - \frac{\partial}{\partial u_k}\left[\sum_{k=1}^{K} \log\left(\sigma\left(-u_k^T v_c\right)\right)\right]$$

Note that K negative samples, $i \in [1, K]$, were drawn from the vocabulary which cannot be assumed to be distinct. As such, let's break up the sum in the loss function into two sums: (i) a sum over all sampled words w_i equal to w_k and, (ii) a sum over all sampled words w_i not equal to w_k. Further, note that here we are iterating over the indices of the the words w instead of indices of the vectors u.

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = -\frac{\partial}{\partial u_k}\left[\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \log\left(\sigma\left(-u_i^T v_c\right)\right) + \sum_{i \in \{1,\ldots,K\}: w_i \neq w_k} \log\left(\sigma\left(-u_{i;w_i \neq w_k}^T v_c\right)\right)\right]$$

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = -\left[\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \frac{\partial}{\partial u_k} \log\left(\sigma\left(-u_i^T v_c\right)\right) + \sum_{i \in \{1,\ldots,K\}: w_i \neq w_k} \frac{\partial}{\partial u_k} \log\left(\sigma\left(-u_{i;w_i \neq w_k}^T v_c\right)\right)\right]$$

Now, $\frac{\partial}{\partial u_k} \log\left(\sigma(-u_{i;w_i \neq w_k}^T v_c)\right) = 0$. Thus,

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = -\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \frac{\partial}{\partial u_k} \log\left(\sigma\left(-u_i^T v_c\right)\right)$$

or equivalently.

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = -\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \frac{\partial}{\partial u_k} \log\left(\sigma\left(-u_k^T v_c\right)\right)$$

Using the chain rule on log,

$$\frac{\partial J_{\text{neg-sample}}\ (v_c, o, U)}{\partial u_k} = -\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \frac{1}{\sigma\left(-u_k^\top v_c\right)} \times \frac{\partial}{\partial u_k}\left(\sigma\left(-u_k^\top v_c\right)\right)$$

Using question (e), $\frac{\partial \sigma}{\partial x} = \sigma(x)(1-\sigma(x))$. So,

$$\begin{aligned}
\frac{\partial J_{\text{neg-sample}}\left(v_c, o, U\right)}{\partial u_k} &= -\sum_{i \in \{1,\ldots,K\}: w_i = w_k} \frac{-v_c \sigma\left(-u_k^T v_c\right)\left(1-\sigma\left(-u_k^T v_c\right)\right)}{\sigma\left(-u_k^T v_c\right)} \\
&= \sum_{i \in \{1,\ldots,K\}: w_i = w_k} v_c\left(1-\sigma\left(-{u_k}^T v_c\right)\right) \\
&= \boxed{\sum_{i \in \{1,\ldots,K\}: w_i = w_k} -v_c\left(\sigma(-u_k^T v_c)-1\right)}
\end{aligned}$$

(h) (3 points) Suppose the center word w_t and the context window is $[w_{t-m}, \cdots, w_{t-1}, w_t, w_{t+1}, \cdots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of word2Vec, the total loss for the context window is:

$$J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m}, U) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} J(v_c, w_{t+j}, U) \tag{2}$$

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

Write down three partial derivatives:

(i) $\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})/\partial U$
(ii) $\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})/\partial v_c$
(iii) $\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})/\partial v_w$ when $w \neq c$

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

Answer:

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})}{\partial U} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{J(v_c, w_{t+j}, U)}{\partial U}$$

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})}{\partial v_c} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{J(v_c, w_{t+j}, U)}{\partial v_c}$$

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \cdots, w_{t+m})}{\partial v_{w;w \neq c}} = 0$$

2 Coding: Implementing word2vec (18 points)

(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. Briefly explain in at most three sentences what you see in the plot.

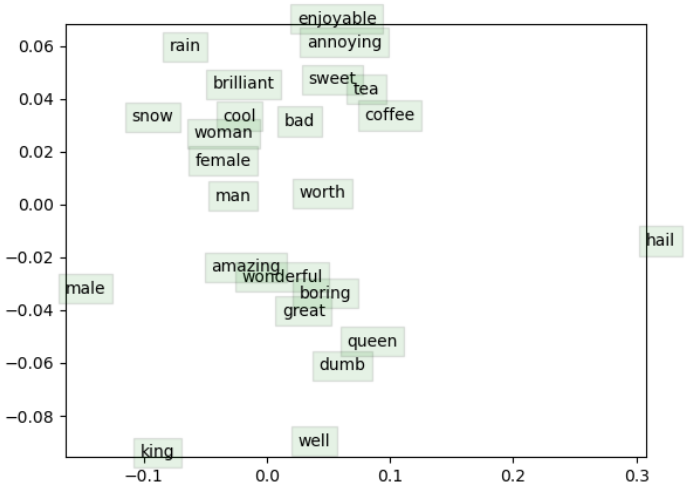

Figure 1: The learned word vectors in the 2D space.

Here we see that some antonyms, like “enjoyable” and “annoying” group together. Other similar terms seem to clump together like “woman” and “female”, “tea” and “coffee”, “amazing” and “wonderful”, etc.