CS224N: NATURAL LANGUAGE PROCESSING WITH DEEP LEARNING ASSIGNMENT #1

ANTHONY HO

1. (a) For any input vector $\boldsymbol{x}$ and any constant c,

$$\begin{aligned}
\text{softmax}(\boldsymbol{x}+c)_i &= \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} \\
&= \frac{e^c e^{x_i}}{\sum_j e^c e^{x_j}} \\
&= \frac{e^c e^{x_i}}{e^c \sum_j e^{x_j}} \\
&= \frac{e^{x_i}}{\sum_j e^{x_j}} \\
&= \text{softmax}(\boldsymbol{x})_i
\end{aligned} \tag{1}$$

Since (1) is true for any arbitrary element i, we can conclude that:

$$\text{softmax}(\boldsymbol{x}) = \text{softmax}(\boldsymbol{x}+c)$$

(b) Please see the coding portion of the assignment.

2. (a) First, we can rearrange the definition of the sigmoid function to obtain:

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

Now we can derive the gradient of the sigmoid function w.r.t. x, assuming x is a scalar.

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

(b) First, let's consider the fact that $\boldsymbol{y}$ is the one-hot label vector, i.e.

$$y_i = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases}$$

where k is the index of the true label.
Therefore, we can simplify the cross entropy function as:

$$\text{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i y_i \log(\hat{y}_i) = -\log(\hat{y}_k)$$

To derive the gradient w.r.t the inputs of a softmax function when cross entropy loss is used for evaluation, let's consider its individual elements:

$$\begin{aligned}
\frac{\partial}{\partial \theta_i}\mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= \frac{\partial}{\partial \theta_i}\left[-\log(\hat{y}_k)\right] \\
&= \frac{\partial}{\partial \theta_i}\left[-\log\left(\frac{e^{\theta_k}}{\sum_j e^{\theta_j}}\right)\right] \\
&= \frac{\partial}{\partial \theta_i}\left[-\theta_k + \log\sum_j e^{\theta_j}\right] \\
&= -\frac{\partial \theta_k}{\partial \theta_i} + \frac{\sum_j e^{\theta_j}\frac{\partial \theta_j}{\partial \theta_i}}{\sum_j e^{\theta_j}}
\end{aligned} \tag{2}$$

By noting that:

$$\frac{\partial \theta_j}{\partial \theta_i} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

We can simpify (2) as:

$$\frac{\partial}{\partial \theta_i}\mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -y_i + \frac{e^{\theta_i}}{\sum_j e^{\theta_j}} = -y_i + \hat{y}_i$$

Thus, the gradient w.r.t the inputs of a softmax function when cross entropy loss is used for evaluation is:

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

(c) Let's denote:

$$\begin{aligned}
\boldsymbol{\theta_1} &= \boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1} \\
\boldsymbol{\theta_2} &= \boldsymbol{h}\boldsymbol{W_2} + \boldsymbol{b_2}
\end{aligned}$$

By applying chain rule, we can rewrite the gradient as:

$$\frac{\partial J}{\partial \boldsymbol{x}} = \frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{x}} = \frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta_2}}\frac{\partial \boldsymbol{\theta_2}}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta_1}}\frac{\partial \boldsymbol{\theta_1}}{\partial \boldsymbol{x}}$$

The first component is simply the result of part (b):

$$\frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta_2}} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

The second component is:

$$\frac{\partial \boldsymbol{\theta_2}}{\partial \boldsymbol{h}} = \frac{\partial}{\partial \boldsymbol{h}}\left(\boldsymbol{h}\boldsymbol{W_2} + \boldsymbol{b_2}\right) = \boldsymbol{W_2}^\top$$

The third component is a $H \times H$ matrix and can be computed by examining its individual elements:

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta_1}}\right)_{ij} = \left(\frac{\partial \sigma(\boldsymbol{\theta_1})}{\partial \boldsymbol{\theta_1}}\right)_{ij} = \frac{\partial(\sigma(\boldsymbol{\theta_1}))_i}{\partial \theta_{1j}} = \frac{\partial \sigma(\theta_{1i})}{\partial \theta_{1j}} = \begin{cases} \sigma'(\theta_{1i}), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

where $\sigma'(\theta_{1i}) = \sigma(\theta_{1i})\left(1 - \sigma(\theta_{1i})\right)$ is the sigmoid gradient as shown in part (a).
Thus, we can define:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta_1}} = \boldsymbol{S}(\boldsymbol{\theta_1})$$

where $\boldsymbol{S}$ denotes a $H \times H$ diagonal matrix where the diagonal elements are $\sigma'(\theta_i)$ for $i = 1, \cdots, H$.
The fourth component is similar to the second component:

$$\frac{\partial \boldsymbol{\theta_1}}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}}\left(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}\right) = \boldsymbol{W_1}^\top$$

Therefore, the the gradient with respect to the inputs $\boldsymbol{x}$ to an one-hidden-layer neural network is:

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{x}} &= \frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta_2}} \frac{\partial \boldsymbol{\theta_2}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta_1}} \frac{\partial \boldsymbol{\theta_1}}{\partial \boldsymbol{x}} \\ &= (\hat{\boldsymbol{y}} - \boldsymbol{y})\, \boldsymbol{W_2}^\top \boldsymbol{S}(\boldsymbol{\theta_1}) \boldsymbol{W_1}^\top \\ &= (\hat{\boldsymbol{y}} - \boldsymbol{y})\, \boldsymbol{W_2}^\top \boldsymbol{S}(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}) \boldsymbol{W_1}^\top\end{aligned}$$

Or, equivalently:

$$\frac{\partial J}{\partial \boldsymbol{x}} = (\hat{\boldsymbol{y}} - \boldsymbol{y})\, \boldsymbol{W_2}^\top \circ \sigma(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}) \circ (1 - \sigma(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}))\boldsymbol{W_1}^\top$$

where $\circ$ denotes the Hadamard product of two vectors.

(d) The dimensions of the weights and biases are as follows:

Parameter	Dimension
W_1	$D_x \times H$
b_1	$1 \times H$
W_2	$H \times D_y$
b_2	$1 \times D_y$

Therefore, the number of parameters in this neural network is:

$$\#\text{ parameters} = D_x H + H + H D_y + D_y = (D_x + 1)H + D_y(H + 1)$$

(e) Please see the coding portion of the assignment.

(f) Please see the coding portion of the assignment.

(g) Please see the coding portion of the assignment.

3. (a) Let's denote:

$$\begin{aligned}\theta_w &= \boldsymbol{u}_w^\top \boldsymbol{v}_c \\ \boldsymbol{\theta} &= \boldsymbol{U}^\top \boldsymbol{v}_c\end{aligned}$$

where θ_w is a scalar, $\boldsymbol{u}_w$ and $\boldsymbol{v}_c$ are column vectors of dimensions $N \times 1$, $\boldsymbol{\theta}$ is a column vector of dimension $V \times 1$, and $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_V]$ is a matrix of dimension $N \times V$.
The softmax predictions for every word can then be written as:

$$\hat{\boldsymbol{y}} = \frac{\exp(\boldsymbol{U}^\top \boldsymbol{v}_c)}{\sum_{w=1}^{V} \exp(\boldsymbol{u}_w^\top \boldsymbol{v}_c)} = \frac{\exp(\boldsymbol{\theta})}{\sum_{w=1}^{V} \exp(\theta_w)}$$

where $\hat{\boldsymbol{y}}$ is a column vector of softmax predictions for every word of dimension $V \times 1$.
By using chain rule and the result of 2(b), the gradient of the cross entropy cost w.r.t. $\boldsymbol{v}_c$ can be derived as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{v}_c} J_{\text{softmax–CE}} &= \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{v}_c} \frac{\partial J}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{v}_c} \frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta}} \\ &= \boldsymbol{U}\,(\hat{\boldsymbol{y}} - \boldsymbol{y})\end{aligned}$$

where $\boldsymbol{y}$ is a column vector of expected word of dimension $V \times 1$.

(b) As in the previous part, we can apply chain rule and the result of 2(b):

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{softmax–CE}} &= \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{u}_k} \frac{\partial J}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} \frac{\partial \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k}\,(\hat{\boldsymbol{y}} - \boldsymbol{y}) \qquad (3)\end{aligned}$$

Rewriting matrix multiplication in (3) explicitly:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{softmax–CE}} &= \sum_{j}^{V} (\hat{\boldsymbol{y}} - \boldsymbol{y})_j \left(\frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} \right)_{\cdot j} \\ &= \sum_{j}^{V} (\hat{y}_j - y_j) \left(\frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} \right)_{\cdot j}\end{aligned}$$

where $\left(\frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} \right)_{\cdot j}$ is the j-th column of $\frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k}$ which is a $N \times V$ matrix. It can be simplified to:

$$\left(\frac{\partial \boldsymbol{U}^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} \right)_{\cdot j} = \begin{cases} \boldsymbol{v}_c, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases}$$

Therefore,the gradient can be simplified to:

$$\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{softmax–CE}} = (\hat{y}_k - y_k)\, \boldsymbol{v}_c$$

Specifically,

$$\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{softmax–CE}} = \begin{cases} (\hat{y}_k - 1)\, \boldsymbol{v}_c, & \text{if } k = o \\ \hat{y}_k \boldsymbol{v}_c, & \text{otherwise} \end{cases}$$

(c) Let's denote:

$$\begin{aligned}\theta_o &= \boldsymbol{u}_o^\top \boldsymbol{v}_c \\ \theta_k &= -\boldsymbol{u}_k^\top \boldsymbol{v}_c\end{aligned}$$

The gradient of the negative sampling loss w.r.t. $\boldsymbol{v}_c$ is:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{v}_c} J_{\text{neg-sample}} &= \frac{\partial}{\partial \boldsymbol{v}_c} \left[-\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \right] \\ &= \frac{\partial}{\partial \boldsymbol{v}_c} \left[-\log(\sigma(\theta_o)) - \sum_{k=1}^{K} \log(\sigma(\theta_k)) \right] \\ &= -\frac{1}{\sigma(\theta_o)} \frac{\partial \sigma(\theta_o)}{\partial \theta_o} \frac{\partial \theta_o}{\partial \boldsymbol{v}_c} - \sum_{k=1}^{K} \frac{1}{\sigma(\theta_k)} \frac{\partial \sigma(\theta_k)}{\partial \theta_k} \frac{\partial \theta_k}{\partial \boldsymbol{v}_c} \\ &= -\frac{1}{\sigma(\theta_o)} \sigma(\theta_o)(1 - \sigma(\theta_o)) \frac{\partial \theta_o}{\partial \boldsymbol{v}_c} - \sum_{k=1}^{K} \frac{1}{\sigma(\theta_k)} \sigma(\theta_k)(1 - \sigma(\theta_k)) \frac{\partial \theta_k}{\partial \boldsymbol{v}_c} \\ &= (\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) - 1) \frac{\partial \boldsymbol{u}_o^\top \boldsymbol{v}_c}{\partial \boldsymbol{v}_c} + \sum_{k=1}^{K} (\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c) - 1) \frac{\partial(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c} \\ &= (\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) - 1) \boldsymbol{u}_o - \sum_{k=1}^{K} (\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c) - 1) \boldsymbol{u}_k\end{aligned}$$

Similarly, the gradient of the negative sampling loss w.r.t. $\boldsymbol{u}_o$ is:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{u}_o} J_{\text{neg-sample}} &= (\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) - 1) \frac{\partial \boldsymbol{u}_o^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_o} + \sum_{k=1}^{K} (\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c) - 1) \frac{\partial(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)}{\partial \boldsymbol{u}_o} \\ &= (\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) - 1) \boldsymbol{v}_c\end{aligned}$$

And the gradient of the negative sampling loss w.r.t. $\boldsymbol{u}_k$ is (changing summation indices from k to j to avoid confusion):

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{neg-sample}} &= (\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) - 1) \frac{\partial \boldsymbol{u}_o^\top \boldsymbol{v}_c}{\partial \boldsymbol{u}_k} + \sum_{j=1}^{K} (\sigma(-\boldsymbol{u}_j^\top \boldsymbol{v}_c) - 1) \frac{\partial(-\boldsymbol{u}_j^\top \boldsymbol{v}_c)}{\partial \boldsymbol{u}_k} \\ &= (1 - \sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \boldsymbol{v}_c\end{aligned}$$

This cost function is much more efficient to compute than the softmax-CE loss because the computation of $\frac{\partial J}{\partial \boldsymbol{v}_c}$ for softmax-CE loss scales as V while the computation of $\frac{\partial J}{\partial \boldsymbol{v}_c}$ for negative sampling loss scales as K, resulting in a speed-up ratio of K/V, which could make a huge difference if one has a big vocabulary.

(d) For skip-gram, the cost for a context centered around c is:

$$J_{\text{skip-gram}}(w_{t-m}, \cdots, w_{t+m}) = \sum_{-m \le j \le m, j \ne 0} F(w_{t+j}, \boldsymbol{v}_c)$$

where

$$F(o, \boldsymbol{v}_c) = \begin{cases} J_{\text{softmax–CE}}(o, \boldsymbol{v}_c, \cdots) = \text{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i^V y_i \log(\hat{y}_i), & \text{for softmax–CE loss} \\ J_{\text{neg-sample}}(o, \boldsymbol{v}_c, \cdots) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^K \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)), & \text{for negative sampling loss} \end{cases}$$

Therefore, the gradients w.r.t. the word vectors for the skip-gram model are:

$$\frac{\partial}{\partial \boldsymbol{v}_k} J_{\text{skip-gram}}(w_{t-m}, \cdots, w_{t+m}) = \begin{cases} \sum_{-m \le j \le m, j \ne 0} \frac{\partial F(w_{t+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}, & \text{if } k = c \\ 0, & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{skip-gram}}(w_{t-m}, \cdots, w_{t+m}) = \sum_{-m \le j \le m, j \ne 0} \frac{\partial F(w_{t+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{u}_k}$$

For CBOW, the cost is:

$$J_{\text{CBOW}}(w_{t-m}, \cdots, w_{t+m}) = F(w_t, \hat{\boldsymbol{v}})$$

where

$$\hat{\boldsymbol{v}} = \sum_{-m \le j \le m, j \ne 0} \boldsymbol{v}_{w_{t+j}}$$

Therefore, the gradients w.r.t. the word vectors for the CBOW model are:

$$\frac{\partial}{\partial \boldsymbol{v}_k} J_{\text{CBOW}}(w_{t-m}, \cdots, w_{t+m}) = \begin{cases} \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}} \frac{\partial \hat{\boldsymbol{v}}}{\partial \boldsymbol{v}_k} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}, & \text{if } t-m \le k \le t+m \text{ and } k \ne t \\ 0, & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial \boldsymbol{u}_k} J_{\text{CBOW}}(w_{t-m}, \cdots, w_{t+m}) = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \boldsymbol{u}_k}$$

(e) Please see the coding portion of the assignment.

(f) Please see the coding portion of the assignment.

(g) Figure 1 shows the visualization for the word vectors. First we can immediately notice it that all the adjectives cluster together, while articles and punctuations scatter around. Also, we observe that many pairs of adjectives that have opposite meanings form vectors that point to approximately the same direction, such as bad $\rightarrow$ good, boring $\rightarrow$ enjoyable, dumb $\rightarrow$ brilliant, and waste $\rightarrow$ worth.

(h)

4. (a) Please see the coding portion of the assignment.

(b) Regularization helps prevent overfitting and reduce model complexity, which would help increase prediction and generalization power to new datasets.

(c) Please see the coding portion of the assignment.

(d)

(e)

(f)

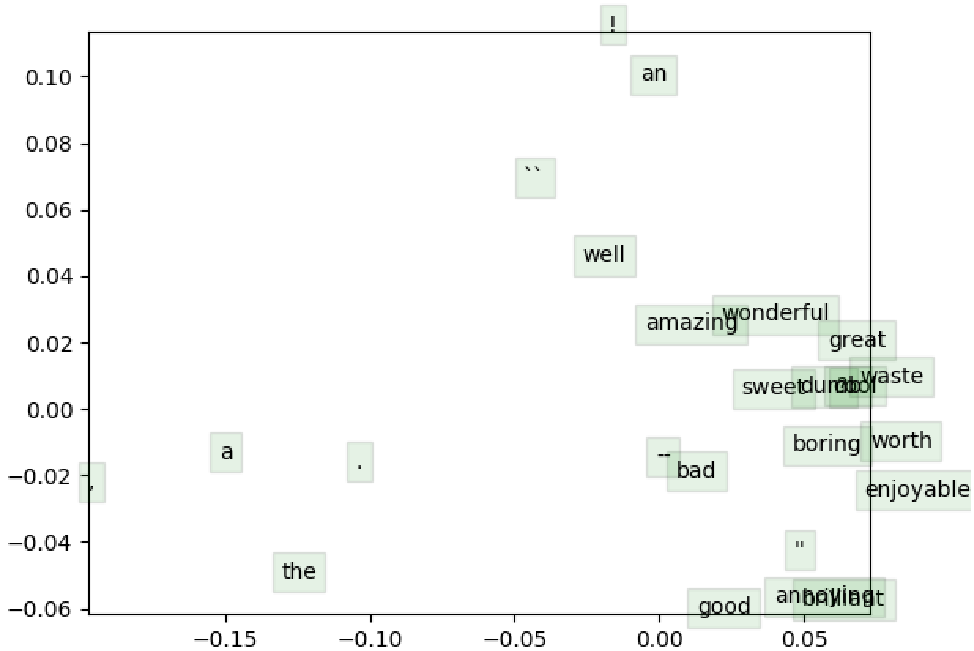

Figure 1. Visualization for the word vectors.

(g)