

CS224N: NATURAL LANGUAGE PROCESSING WITH DEEP LEARNING ASSIGNMENT #1

ANTHONY HO

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

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

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

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

- (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.

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

- (b) First, let's consider the fact that \mathbf{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:

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

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:

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

By noting that:

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

We can simplify (2) as:

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

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

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

(c) Let's denote:

$$\begin{aligned}
\boldsymbol{\theta}_1 &= \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1 \\
\boldsymbol{\theta}_2 &= \mathbf{h}\mathbf{W}_2 + \mathbf{b}_2
\end{aligned}$$

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

$$\frac{\partial J}{\partial \mathbf{x}} = \frac{\partial \text{CE}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{x}} = \frac{\partial \text{CE}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \boldsymbol{\theta}_2} \frac{\partial \boldsymbol{\theta}_2}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}_1} \frac{\partial \boldsymbol{\theta}_1}{\partial \mathbf{x}}$$

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

$$\frac{\partial \text{CE}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \boldsymbol{\theta}_2} = \hat{\mathbf{y}} - \mathbf{y}$$

The second component is:

$$\frac{\partial \boldsymbol{\theta}_2}{\partial \mathbf{h}} = \frac{\partial}{\partial \mathbf{h}} (\mathbf{h}\mathbf{W}_2 + \mathbf{b}_2) = \mathbf{W}_2^\top$$

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

$$\left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}_1} \right)_{ij} = \left(\frac{\partial \sigma(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} \right)_{ij} = \frac{\partial \sigma(\theta_{1i})}{\partial \theta_{1j}} = \begin{cases} \sigma'(\theta_{1i}), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

where $\sigma'(\theta_{1i}) = \sigma(\theta_{1i})(1 - \sigma(\theta_{1i}))$ is the sigmoid gradient as shown in part (a).

Thus, we can define:

$$\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}_1} = \mathbf{S}(\boldsymbol{\theta}_1)$$

where \mathbf{S} denotes a $H \times H$ diagonal matrix where the diagonal elements are $\sigma'(\theta_i)$ for $i = 1, \dots, H$.

The fourth component is similar to the second component:

$$\frac{\partial \theta_1}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} (\mathbf{x} \mathbf{W}_1 + \mathbf{b}_1) = \mathbf{W}_1^\top$$

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

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{x}} &= \frac{\partial \text{CE}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \theta_2} \frac{\partial \theta_2}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \theta_1} \frac{\partial \theta_1}{\partial \mathbf{x}} \\ &= (\hat{\mathbf{y}} - \mathbf{y}) \mathbf{W}_2^\top \mathbf{S}(\theta_1) \mathbf{W}_1^\top \\ &= (\hat{\mathbf{y}} - \mathbf{y}) \mathbf{W}_2^\top \mathbf{S}(\mathbf{x} \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_1^\top \end{aligned}$$

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

Parameter	Dimension
\mathbf{W}_1	$D_x \times H$
\mathbf{b}_1	$1 \times H$
\mathbf{W}_2	$H \times D_y$
\mathbf{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:

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

where θ_w is a scalar, \mathbf{u}_w and \mathbf{v}_c are column vectors of dimensions $N \times 1$, $\boldsymbol{\theta}$ is a column vector of dimension $V \times 1$, and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_V]$ is a matrix of dimension $N \times V$.

The softmax predictions for every word can then be written as:

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

where $\hat{\mathbf{y}}$ 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. \mathbf{v}_c can be derived as:

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

where \mathbf{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):

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

Rewriting matrix multiplication in (3) explicitly:

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

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

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

Therefore, the gradient can be simplified to:

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

Note that:

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

(c) Let's denote:

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

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

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

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

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

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

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

This cost function is much more efficient to compute than the softmax-CE loss because the computation of $\frac{\partial J}{\partial \mathbf{v}_c}$ for softmax-CE loss scales as V while the computation of $\frac{\partial J}{\partial \mathbf{v}_c}$ 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)
- (e)
- (f)
- (g)
- (h)

4. (a)
- (b)
 - (c)
 - (d)
 - (e)
 - (f)
 - (g)