STANFORD UNIVERSITY

CS224D: DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Assignment 1

Xiaomo Liu

August 23, 2015

1 SOFTMAX

$$\text{softmax}(\mathbf{x}) = \text{softmax}(\mathbf{x} + c) \tag{1.1}$$

Proof:

$$\begin{aligned}
\text{softmax}(\mathbf{x} + c) &= \frac{e^{\mathbf{x}+c}}{\sum_{\mathbf{x}} e^{\mathbf{x}+c}} \\
&= \frac{e^{\mathbf{x}} e^{c}}{\sum_{\mathbf{x}} e^{\mathbf{x}} e^{c}} \\
&= \frac{e^{c} \times e^{\mathbf{x}}}{e^{c} \times \sum_{\mathbf{x}} e^{\mathbf{x}}} \\
&= \frac{e^{\mathbf{x}}}{\sum_{\mathbf{x}} e^{\mathbf{x}}} = \text{softmax}(\mathbf{x})
\end{aligned} \tag{1.2}$$

In practice, this is trick can be used to maintain numerical stability.

2 NEURAL NETWORK BASICS

2.1 GRADIENT OF SIGMOD FUNCTION

The sigmod function in neural networks is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

where x is s scalar. Thus, the gradient of sigmod function is

$$
\begin{aligned}
\nabla\sigma(x) = \frac{\partial\sigma(x)}{\partial x} &= \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} \\
&= \frac{\partial}{\partial z}(z^{-1})\cdot\frac{\partial}{\partial x}(1+e^{-x}) \\
&= \frac{1}{(1+e^{-x})^2}\cdot e^{-x} \\
&= \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}}\cdot\left(1-\frac{1}{1+e^{-x}}\right) \\
&= \sigma(x)\cdot(1-\sigma(x))
\end{aligned}
\tag{2.2}
$$

2.2 Gradient of Cross Entropy

When using a neural network to perform classification and prediction, it is usually better to use cross-entropy error than classification error, and somewhat better to use cross-entropy error than mean squared error to evaluate the quality of the neural network.

$$
\mathrm{CE}(\mathbf{y},\hat{\mathbf{y}}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.3}
$$

Since $\mathbf{y}$ is a one-hot vector, then only y_k is one, other dimensions of $\mathbf{y}$ are all zero. Thus, cross entropy will be $CE(\mathbf{y},\hat{\mathbf{y}}) = -\log(\hat{y}_k)$. The gradient of cross entropy becomes

$$
\frac{\partial \mathrm{CE}(\mathbf{y},\hat{\mathbf{y}})}{\partial\theta} = -\frac{\partial \log(\mathrm{softmax}(\theta_k))}{\partial\theta} \tag{2.4}
$$

The softmax function of θ_k is

$$
\mathrm{softmax}(\theta_k) = \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} \tag{2.5}
$$

Thus,

$$
\hat{y}_k = \log(\mathrm{softmax}(\theta_k)) = \theta_k - \log(\sum_j e^{\theta_j}) \tag{2.6}
$$

For θ_i that $i = k$, the derivate is

$$
\frac{\partial}{\partial\theta_i}(\theta_k - \log(\sum_j e^{\theta_j})) = 1 - \frac{e^{\theta_i}}{\sum_j e^{\theta_j}} \tag{2.7}
$$

Otherwise, for θ_i that $i \neq k$, the derivate is

$$
\frac{\partial}{\partial\theta_i}(\theta_k - \log(\sum_j e^{\theta_j})) = -\frac{e^{\theta_i}}{\sum_j e^{\theta_j}} \tag{2.8}
$$

Combining these two cases together

$$
\frac{\partial CE(\mathbf{y},\hat{\mathbf{y}})}{\partial\theta_i} = t_i - \hat{y}_i \tag{2.9}
$$

where $t_i = 1$ when $i = k$.

2.3 Gradient of One-hidden-layer Neural Network

The cost function J for this neural network is

$$J = \mathrm{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.10}$$

Thus, the gradients with respect to the input vector $\mathbf{x}$ is

$$\frac{\partial J}{\partial x_i} = \sum_k \frac{\partial J}{\partial h_k} \cdot \frac{\partial h_k}{\partial x_i} \tag{2.11}$$

The first part of the gradient, based on the chain rule, is

$$\frac{\partial J}{\partial h_k} = \sum_j \frac{\partial J}{\partial \theta_j} \cdot \frac{\partial \theta_j}{\partial h_k} \tag{2.12}$$

and

$$\theta_j = \sum_k h_k W_2(k, j) + b_2(j) \tag{2.13}$$

Thus,

$$\frac{\partial J}{\partial h_k} = \sum_j (t_j - \hat{y}_j) \cdot W_2(k, j) \tag{2.14}$$

The second part of the gradient, based on the chain rule, is

$$\begin{aligned}\frac{\partial h_k}{\partial x_i} &= \frac{\partial \sigma(z_k)}{\partial z_k} \cdot \frac{\partial(\sum_i x_i W_1(i,k) + b_1(k))}{\partial x_i} \\ &= \sigma'(z_k) \cdot W_1(i,k)\end{aligned} \tag{2.15}$$

where $z_k = \mathbf{x}\mathbf{W}_1(*, k) + b_1(k)$.
The final gradient $\frac{\partial J}{\partial x}$ is

$$\begin{aligned}\frac{\partial J}{\partial x_i} &= \sum_k \frac{\partial J}{\partial h_k} \cdot \frac{\partial h_k}{\partial x_i} \\ &= \sum_k \sum_j (t_j - \hat{y}_j) \cdot W_2(k, j) \cdot \sigma'(z_k) \cdot W_1(i,k)\end{aligned} \tag{2.16}$$

2.4 Parameters of Neural Network

In the one-hidden layer neural networks, assuming the input is D_x-dimensional, the output is D_y-dimensional, and H hidden units, the number of parameters is

$$(D_x + 1) \times H + (D_y + 1) \times H = (D_x + D_y + 2) \times H \tag{2.17}$$

3 WORD2VEC

3.1 GRADIENT OF INPUT WORD VECTOR

The word prediction in `word2vec` model with softmax function is

$$\hat{y}_i = \Pr(\text{word}_i | \hat{\boldsymbol{r}}, \boldsymbol{w}) = \frac{\exp(\boldsymbol{w}_i^\top \hat{\boldsymbol{r}})}{\sum_{j=1}^{|V|} \exp(\boldsymbol{w}_j)} \tag{3.1}$$

The cross entropy error between the predicted and actual output probabilities is

$$J = \text{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i y_i \log(\hat{y}_i) \tag{3.2}$$

and because $\boldsymbol{y}$ is a one-hot vector, the object function becomes as follows, if word$_i$ is in the context:

$$\begin{aligned} J &= -\log(\hat{y}_i) \\ &= u_i - \log \sum_{j=1}^{|V|} \exp(u_j) \end{aligned} \tag{3.3}$$

where $u_j = \boldsymbol{w}_j^\top \hat{\boldsymbol{r}} = \sum_{k=1}^{|h|} w_{k,j} r_k$. Thus, the gradient of J with respect to $\hat{r}$ is

$$\begin{aligned} \frac{\partial J}{\partial r_k} &= \sum_j \frac{\partial J}{\partial u_j} \cdot \frac{\partial u_j}{\partial r_k} \\ &= \sum_j^{|V|} (t_j - \hat{y}_j) w_{k,j} \end{aligned} \tag{3.4}$$

where $t_j = 1$ if $j = i$, otherwise $t_j = 0$. The vector version of the gradient is

$$\frac{\partial J}{\partial \hat{\boldsymbol{r}}} = \sum_j^{|V|} (t_j - \hat{y}_j) \boldsymbol{w}_j \tag{3.5}$$

3.2 GRADIENT OF OUTPUT WORD VECTOR

Thus, the gradient of J with respect to $w_{i,j}$ is

$$\begin{aligned} \frac{\partial J}{\partial w_{i,j}} &= \frac{\partial J}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{i,j}} \\ &= (t_j - \hat{y}_j) r_i \end{aligned} \tag{3.6}$$

where $t_j = 1$ if $j = i$, otherwise $t_j = 0$. The vector version of the gradient is

$$\frac{\partial J}{\partial \boldsymbol{w}_i} = (t_j - \hat{y}_j) \hat{\boldsymbol{r}} \tag{3.7}$$

3.3 Gradient of negative sampling

The loss function of negative sampling is

$$J(\hat{\boldsymbol{r}}, \boldsymbol{w}_i, \boldsymbol{w}_{1,\dots,K}) = -\log(\sigma(\boldsymbol{w}_i^\top \hat{\boldsymbol{r}})) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{w}_k^\top \hat{\boldsymbol{r}})) \tag{3.8}$$

where $\sigma(\cdot)$ is the sigmoid function. The gradient of J with respect to $\hat{\boldsymbol{r}}$ is

$$\tag{3.9}$$

While the gradient of J with respect to the outwords w_i where $i \neq k$ is

$$\begin{aligned}
\frac{\partial J}{\partial w_{i,j}} &= -\frac{\partial \log(\sigma(u_j))}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{i,j}} = -\frac{\partial \log(\sigma(\sum_{j=1}^{k} w_{i,j} \hat{r}_j))}{\partial w_{i,j}} \\
&= -\frac{1}{\sigma(u_j)} \cdot \frac{\partial \sigma(u_j)}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{i,j}} \\
&= -\frac{\sigma(u_j)(1-\sigma(u_j))}{\sigma(u_j)} \hat{r}_j \\
&= \left(\sigma(\boldsymbol{w}_k^\top \hat{\boldsymbol{r}}\right) - 1) \hat{r}_j
\end{aligned} \tag{3.10}$$

where $u_j = \boldsymbol{w}_k^\top \hat{\boldsymbol{r}} = \sum_{j=1}^{k} w_{i,j} \hat{r}_j$. The vector version of this gradient is

$$\frac{\partial J}{\partial \boldsymbol{w}_i} = \left(\sigma(\boldsymbol{w}_j^\top \hat{\boldsymbol{r}}) - 1\right) \hat{\boldsymbol{r}} \tag{3.11}$$

The gradient of J with respect to negative samples w_k are different from that of positive sample. The computation of their gradients are as following,

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{w}_k} &= \left(1 - \sigma(-\boldsymbol{w}_k^\top \hat{\boldsymbol{r}})\right) \hat{\boldsymbol{r}} \\
&= \sigma(\boldsymbol{w}_k^\top \hat{\boldsymbol{r}}) \hat{\boldsymbol{r}}
\end{aligned} \tag{3.12}$$

The reason we can do this conversion is due to the property of $\sigma(x)$ as follows

$$\begin{aligned}
1 - \sigma(-x) &= 1 - \frac{1}{1+e^x} \\
&= \frac{e^x}{1+e^x} \\
&= \frac{1}{e^{-x}+1} = \sigma(x)
\end{aligned} \tag{3.13}$$

Thus, we combine the gradient of positive and negative samples into one equation

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{w}_j} &= \begin{cases} (\sigma(\boldsymbol{w}_j^\top \hat{\boldsymbol{r}}) - 1) \hat{\boldsymbol{r}} & \text{for } w_j = w_i \\ \sigma(\boldsymbol{w}_j^\top \hat{\boldsymbol{r}}) \hat{\boldsymbol{r}} & \text{for } w_j = w_1, \dots, w_k \end{cases} \\
&= (\sigma(\boldsymbol{w}_j^\top \hat{\boldsymbol{r}}) - t_j) \hat{\boldsymbol{r}}
\end{aligned} \tag{3.14}$$

where $t_j = 1$ if $j = i$, otherwise $t_j = 0$.
Next, let's compute the gradient of J with respect to $\hat{\boldsymbol{r}}$.

$$\begin{aligned}\frac{\partial J}{\partial \hat{\boldsymbol{r}}} &= \sum_j \frac{\partial J}{\partial u_j} \cdot \frac{\partial u_j}{\partial \hat{\boldsymbol{r}}} \\ &= \frac{\partial J}{\partial u_i} \cdot \frac{\partial u_i}{\partial \hat{\boldsymbol{r}}} + \sum_{j=1}^{k} \frac{\partial J}{\partial u_j} \cdot \frac{\partial u_j}{\partial \hat{\boldsymbol{r}}} \\ &= \left(\sigma(\boldsymbol{w}_i^\top \hat{\boldsymbol{r}}) - t_i\right) \boldsymbol{w}_i - \sum_{k=1}^{K} \left(\sigma(\boldsymbol{w}_k^\top \hat{\boldsymbol{r}}) - t_k\right) \boldsymbol{w}_k\end{aligned} \tag{3.15}$$

3.4 Gradient of skip-gram with negative sampling

In the skip-gram model, given the input word$_i$, it need to predict its context output words with a window size C, i.e. (word$_{i-C}$, ..., word$_{i+C}$).

$$J_s(\text{word}_{i-C,\dots,i+C}) = \sum_{-c \le j \le c, j \ne 0} F(\boldsymbol{v}'_{w_{i+j}} | \boldsymbol{v}_{w_i}) \tag{3.16}$$

In negative sampling, the cost function of each input vs. output words pair $F(\boldsymbol{v}'_{w_{i+j}} | \boldsymbol{v}_{w_i})$ is

$$F(v'_{w_{i+j}} | v_{w_i}) = -\log(\sigma(\boldsymbol{v}'^\top_{i+j} \boldsymbol{v}_i)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{v}'^\top_k \boldsymbol{v}_i)) \tag{3.17}$$

Thus, the gradient of J_s with respect to output word vector $v'_{w_{i+j}}$ is

$$\begin{aligned}\frac{\partial J_s}{\partial \boldsymbol{v}'_{w_{i+j}}} &= \frac{\partial F(\boldsymbol{v}'_{w_{i+j}} | \boldsymbol{v}_{w_i})}{\partial \boldsymbol{v}'_{w_{i+j}}} \\ &= (\sigma(\boldsymbol{v}'^\top_{w_{i+j}} \boldsymbol{v}_{w_i}) - t_{i+j}) \boldsymbol{v}_{w_i}\end{aligned} \tag{3.18}$$

where $t_{i+j} = 1$ if w_{i+j} is a positive sample and $t_{i+j} = 0$ otherwise.
While, the gradient of J_s with respect to input word vector v_{w_i} is

$$\begin{aligned}\frac{\partial J_s}{\partial \boldsymbol{v}_{w_i}} &= \sum_{-C \le j \le C} \frac{\partial F(\boldsymbol{v}'_{w_{i+j}} | \boldsymbol{v}_{w_i})}{\partial \boldsymbol{v}_{w_i}} \\ &= \sum_{-C \le j \le C} \left(\left(\sigma(\boldsymbol{v}'^\top_{w_{i+j}} \boldsymbol{v}_{w_i}) - t_i\right) \boldsymbol{v}_{w_i} - \sum_{k=1}^{K} \left(\sigma(\boldsymbol{v}'_{w_k} \boldsymbol{v}_{w_i}) - t_k\right) \boldsymbol{v}'_{w_k} \right)\end{aligned} \tag{3.19}$$

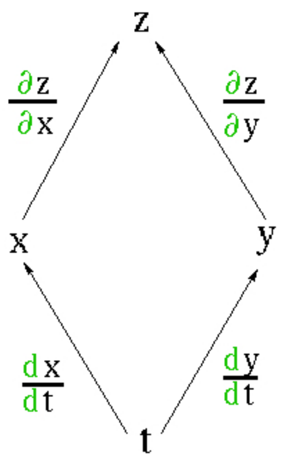

Figure 5.1: Multi-variate chain rule

3.5 Gradient of CBOW with negative sampling

4 Sentiment Analysis

4.1 Reason to Use Regularization

4.2

5 Appendix: Chain Rule

In calculus, chain rule is a formula for computing the derivate of the function composition. For example, if $y = f(u)$ and $u = g(x)$, the derivate of y with respect to x is

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} \tag{5.1}$$

Now let's consider a more complicated case for computing chain rule. That is the chain rule for multivariate. Now functions f and g are expressed in terms of their components as $y = f(\boldsymbol{u}) = f(u_1, ..., u_n)$ and $u_k = g_k(\boldsymbol{x}) = g_k(x_1, ..., x_m)$. Then, the partial derivate of y with respect to x_j is

$$\frac{\partial y}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial y}{\partial u_k} \cdot \frac{\partial u_k}{dx_j} \tag{5.2}$$

5.1 Proof of multivariate chain rule