## 1 Backpropagation

We now derive the Backpropagation algorithm for finding the gradients of the cost function of the neural network. For our parameters $W \in \mathbb{R}^{H \times nC}$, $b^{(1)} \in \mathbb{R}^{H}$, $U \in \mathbb{R}^{H}$,$b^{(2)} \in \mathbb{R}$ and $L \in \mathbb{R}^{nC \times V}$ the cost function for our neural network is defined as follows.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}log(h_\theta(x^{(i)})) - (1-y^{(i)})log(1-h_\theta(x^{(i)}))\right] + \frac{C}{2m}\left[\sum_{j=1}^{nC}\sum_{k=1}^{H}W_{k,j}^2 + \sum_{k=1}^{H}U_k^2\right] \tag{1}$$

$$h_\theta(x^{(i)}) = g\left(U^T f\left(Wx^{(i)} + b^{(1)}\right) + b^{(2)}\right) \tag{2}$$

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3}$$

$$g(z) = sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4}$$

The derivatives of the *tanh* and *sigmoid* functions $g$ and $f$ are given by

$$\frac{d}{dz}f(z) = 1 - \left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right)^2 = 1 - f^2(z) \tag{5}$$

$$\frac{d}{dz}g(z) = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = g(z)(1-g(z)) \tag{6}$$

Notice that the derivatives of both the hyperbolic tangent and logistic functions are defined completely in terms of the functions themselves. This will simplify deriving the gradients for each parameter. Since we will implement stochastic gradient descent in order to optimize our cost function with respect to the parameters $U, b^{(2)}, W, b^{(1)}$ and $L$ we only need to observe the gradient with respect to these parameters at a single training example at a time. Thus we find the gradient for the simplified cost function below where $x$ and $y$ represent a single training example.

$$J(\theta) = -ylog(h_\theta(x)) - (1-y)log(1-h_\theta(x)) + \frac{C}{2}\left[\sum_{j=1}^{nC}\sum_{k=1}^{H}W_{k,j}^2 + \sum_{k=1}^{H}U_k^2\right] \tag{7}$$

We now derive the gradients $\frac{\partial J}{\partial U}$,$\frac{\partial J}{\partial b^{(2)}}$, $\frac{\partial J}{\partial W}$, $\frac{\partial J}{\partial b^{(1)}}$ and $\frac{\partial J}{\partial L}$. In order to simplify the gradient for each parameter notice that the regularization term (the second term in the cost function) does not depend on $b^{(1)}$, $b^{(2)}$ or $L$ thus vanishes for those terms. Also since the gradient distributes over addition we can take the derivative of this term with respect to $U$ and $W$ separately and add it in later. The gradient of the regularization term for the two parameters is

$$\frac{\partial}{\partial U}\frac{C}{2}\left[\sum_{j=1}^{nC}\sum_{k=1}^{H} W_{k,j}^2 + \sum_{k=1}^{H} U_k^2\right] = CU \tag{8}$$

$$\frac{\partial}{\partial W}\frac{C}{2}\left[\sum_{j=1}^{nC}\sum_{k=1}^{H} W_{k,j}^2 + \sum_{k=1}^{H} U_k^2\right] = CW \tag{9}$$

To simplify notation we define $z^{(1)}, z^{(2)}, a^{(1)}, a^{(2)}$ as follows.

$$\begin{aligned} z^{(1)} &= Wx + b^{(1)}, \\ a^{(1)} &= f(z^{(1)}), \\ z^{(2)} &= U^T a^{(1)} + b^{(2)}, \\ a^{(2)} &= g(z^{(2)}) \end{aligned} \tag{10}$$

Finally we let the even further simplified cost function written in terms of the above defininitions without regularization (as we will add that in later) be defined as

$$J(U, b^{(2)}, W, b^{(1)}, L) = -ylog(a^{(2)}) - (1-y)log(1-a^{(2)}) \tag{11}$$

First we derive $\frac{\partial J}{\partial U}$ with respect to (11). Using the chain rule and remembering the derivative of the sigmoid function from (6) we find

$$\begin{aligned} \frac{\partial J}{\partial U} &= \left(\frac{-y}{a^{(2)}} + \frac{(1-y)}{1-a^{(2)}}\right)\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial U}, \\ \frac{\partial J}{\partial U} &= \left(\frac{-y}{a^{(2)}} + \frac{(1-y)}{1-a^{(2)}}\right)(a^{(2)})(1-a^{(2)})\frac{\partial z^{(2)}}{\partial U}, \\ \frac{\partial J}{\partial U} &= \left(-y(1-a^{(2)}) + (1-y)a^{(2)}\right)\frac{\partial z^{(2)}}{\partial U}, \end{aligned}$$

$$\frac{\partial J}{\partial U} = \left(a^{(2)} - y\right)\frac{\partial z^{(2)}}{\partial U} \tag{12}$$

Now we only need to find the partial $\frac{\partial z^{(2)}}{\partial U}$ in (12) in order to derive the gradient.

$$\frac{\partial z^{(2)}}{\partial U} = \frac{\partial}{\partial U}\left(U^T a^{(1)} + b^{(2)}\right) = (a^{(1)})^T \tag{13}$$

Putting together (12) and (13) we arrive at

$$\frac{\partial J}{\partial U} = \left(a^{(2)} - y\right)(a^{(1)})^T \tag{14}$$

Also using the exact same steps to arrive at (12) we can solve for $\frac{\partial J}{\partial b^{(2)}}$

$$\frac{\partial J}{\partial b^{(2)}} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial b^{(2)}} = \left(a^{(2)} - y\right) \tag{15}$$

We find $\frac{\partial J}{\partial W}$ using the same derivation as in (12) and applying the chain rule again.

$$\frac{\partial J}{\partial W} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial W} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W} \tag{16}$$

To simplify the derivation we take all of the partial derivatives in (16) with respect to a single element, $W_{ij}$. First notice that only the $i^{th}$ element of $a^{(1)}$ and $z^{(1)}$ depend on the $ij^{th}$ element of $W$, thus we have

$$\frac{\partial J}{\partial W_{ij}} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial a_i^{(1)}} \frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial W_{ij}} \tag{17}$$

Now we can find each partial separately and combine them.

$$\frac{\partial z^{(2)}}{\partial a_i^{(1)}} = \frac{\partial}{\partial a_i^{(1)}} \left(U^T a^{(1)} + b^{(2)}\right) = U_i^T \tag{18}$$

From the derivative of the hyperbolic tangent function found in (5) we have

$$\frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} = \frac{\partial}{\partial z_i^{(1)}} f(z_i^{(1)}) = 1 - f(z_i^{(1)})^2 = 1 - (a_i^{(1)})^2 \tag{19}$$

$$\frac{\partial z_i^{(1)}}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left(Wx + b^{(1)}\right) = x_j \tag{20}$$

Combining (17), (18), (19) and (20) we end up with

$$\frac{\partial J}{\partial W_{ij}} = \left(a^{(2)} - y\right) U_i^T \left(1 - (a_i^{(1)})^2\right) x_j \tag{21}$$

In order to find the gradient in matrix form we need to introduce new notation. Let $\diamond$ be defined as an element-wise multiplication of two matrices or vectors of the same size. Now we have the full gradient as

$$\frac{\partial J}{\partial W} = \left(a^{(2)} - y\right) U \diamond \left(1 - (a^{(1)})^2\right) x^T \tag{22}$$

To find $\frac{\partial J}{\partial b^{(1)}}$ we apply the same steps we used to arrive at (17)

$$\frac{\partial J}{\partial b_i^{(1)}} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial a_i^{(1)}} \frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial b_i^{(1)}} \tag{23}$$

$$\frac{\partial z_i^{(1)}}{\partial b_i^{(1)}} = \frac{\partial}{\partial b_i^{(1)}} \left(Wx + b^{(1)}\right) = 1 \tag{24}$$

We combine (17), (18), (19) and (24) and find

$$\frac{\partial J}{\partial b^{(1)}} = \left(a^{(2)} - y\right) U \diamond \left(1 - (a^{(1)})^2\right) \tag{25}$$

In order to find the last parameter gradient $\frac{\partial J}{\partial L}$, we find $\frac{\partial J}{\partial x}$ where $x$ is the training example and then map this gradient to its corresponding piece of the $\frac{\partial J}{\partial L}$ gradient. We use the same steps as (17) but this time every element of $a^{(1)}$ and $z^{(1)}$ depend on the $i^{th}$ element of $x$.

$$\frac{\partial J}{\partial x_i} = \left(a^{(2)} - y\right) \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial x_i} \tag{26}$$

Letting $W_{\bullet i}$ designate the $i^{th}$ column of $W$,

$$\frac{\partial z^{(1)}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(Wx + b^{(1)}\right) = W_{\bullet i} \tag{27}$$

Using the same results for the partial derivatives from (18), (19) and the above two equations we have

$$\frac{\partial J}{\partial x_i} = \left(a^{(2)} - y\right) \left[U \diamond \left(1 - (a^{(1)})^2\right)\right]^T W_{\bullet i} \tag{28}$$

To find $\frac{\partial J}{\partial x_i}$ in vector form notice that (28) gives a row vector if we multiply by the full matrix $W$ thus we simply need the transpose of this row vector to arrive at the column vector gradient for $x$

$$\frac{\partial J}{\partial x} = \left(a^{(2)} - y\right) W^T \left[U \diamond \left(1 - (a^{(1)})^2\right)\right] \tag{29}$$

To see how to map (29) back to $\frac{\partial J}{\partial L}$ notice that $x$ is an $nC$ column vector where $n$ is the vector size of each word and $C$ is the window size or number of context words in each sample. This means we have $C$ columns of $L$ to update for each training sample. Thus we can write $x$ as a $C$ long vector with each element corresponding to a word vector.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_C \end{bmatrix}$$

Each of these $C$ $n$-dimensional vectors, $x_i$, correspond to a column of $L$. Thus we can update the columns of $L$ using this breakdown of word vectors of $x$ and the gradient found in (29). In order to do this we must keep track what column each of the word vectors of $x$ correspond to.

Our final derivations for $\frac{\partial J}{\partial b^{(1)}}$, $\frac{\partial J}{\partial b^{(2)}}$ and $\frac{\partial J}{\partial L}$ are given in (25), (15) and (29) respectively since these gradients don't depend on our regularization term. To find the final form of $\frac{\partial J}{\partial W}$ we combine (9) and (22). Similarly to find the final form of $\frac{\partial J}{\partial U}$ we combine (8) and (14).

$$\frac{\partial J}{\partial W} = \left(a^{(2)} - y\right) U \diamond \left(1 - (a^{(1)})^2\right) x^T + CW \tag{30}$$

$$\frac{\partial J}{\partial U} = \left(a^{(2)} - y\right) (a^{(1)})^T + CU \tag{31}$$

This completes the derivation of the parameters to the cost function (7) for our neural network model.

## 2 Softmax and Logistic Regression

We now show that the Softmax classification model with only two classes is equivalent to logistic regression model with a single weight vector. We begin with the general Sofmax model and setting the number of classes to 2 we then derive logistic regression with a single weight vector.

Recall that Sofmax classification is defined as

$$p(y = i|x;\theta) = \phi_i = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} \theta_j^T x} \tag{32}$$

Thus the hypothesis function $h_\theta(x)$ should output a $k$-dimensional vector of probabilities that $x$ belongs to each class as defined in (32) where $k$ represents the number of classes. However, since the probability over all classes must sum to one, we have that $p(y = k|x;\theta) = 1 - \sum_{i=1}^{k-1} p(y = i|x;\theta) = 1 - \sum_{i=1}^{k-1} \phi_i$. Since $p(y = k|x;\theta)$ can be expressed as a function of all other $k-1$ probabilities, our hypothesis function only needs to output the first $k - 1$ probabilities.

$$h_\theta(x) = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} \tag{33}$$

Now we set $k = 2$ since we are considering Softmax with only two classes and we obtain

$$\begin{aligned}
h_\theta(x) &= \phi_1 = \frac{e^{\theta_1^T x}}{\sum_{j=1}^{2} \theta_j^T x} \\
h_\theta(x) &= \frac{e^{\theta_1^T x}}{e^{\theta_1^T x} + e^{\theta_2^T x}} \\
h_\theta(x) &= \frac{e^{\theta_1^T x}}{e^{\theta_1^T x} + e^{\theta_2^T x}} * \frac{\frac{1}{e^{\theta_1^T x}}}{\frac{1}{e^{\theta_1^T x}}} \\
h_\theta(x) &= \frac{1}{1 + \frac{e^{\theta_2^T x}}{e^{\theta_1^T x}}} \\
h_\theta(x) &= \frac{1}{1 + e^{(\theta_2 - \theta_1)^T x}} \\
h_\theta(x) &= \frac{1}{1 + e^{-(\theta_1 - \theta_2)^T x}}
\end{aligned} \tag{34}$$

We see that the final result derived in (34) is exactly equivalent to the hypothesis function for logistic regression with a single weight matrix, namely $\theta = \theta_1 - \theta_2$.