1 Backpropagation

1.1 Notations

Assume there are M layers. The k^{th} layer is denoted by a vector $v^{[k]} \in \mathbb{R}^{n_k}$. The weight matrix is denoted by $W^{[k]} \in \mathbb{R}^{n_k \times n_{k-1}}$. The bias vector is denoted by $b^{[k]} \in \mathbb{R}^{n_k}$. The activation function between the $k-1^{\text{th}}$ layer and the k^{th} layer is denoted by $\sigma_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$. To simplify the computation, we let the activation function be sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$, and note that

$$\sigma(x) = \begin{pmatrix} \frac{1}{1+e^{-x_1}} \\ \frac{1}{1+e^{-x_2}} \\ \vdots \\ \frac{1}{1+e^{-x_n}} \end{pmatrix}, \forall x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

Let $\{(x_i, y_i)\}_1^n$ be our training data, then the feed forward process can be formalized as

$$\begin{aligned} v_i^{[1]} &= x_i, \\ v_i^{[2]} &= \sigma(W^{[1]} v_i^{[1]} + b^{[1]}), \\ v_i^{[3]} &= \sigma(W^{[2]} v_i^{[2]} + b^{[2]}), \\ &\vdots \\ v_i^{[k]} &= \sigma(W^{[k-1]} v_i^{[k-1]} + b^{[k-1]}), \\ &\vdots \\ v_i^{[M]} &= \sigma(W^{[M-1]} v_i^{[M-1]} + b^{[M-1]}) \end{aligned} \tag{1}$$

1.2 Training

Let the last layer's output $v_i^{[M]}$ be $\hat{y}_i$. Let's use L2-norm as our loss function, for single data, we have

$$L(\hat{y}_i, y_i) = \frac{1}{2} \|\hat{y}_i - y_i\|_2^2 \tag{2}$$

To train a network, we need to first sum the loss of all data,

$$\mathcal{L} = \sum_{i=1}^{n} L(\hat{y}_i, y_i) = \sum_{i=1}^{n} \frac{1}{2} \|\hat{y}_i - y_i\|_2^2$$

The equation 1 shows that the loss function 2 actually depends on each weight matrix $W^{[i]}$, each bias vector $b^{[i]}$, and each ground truth y_i, that is,

$$\mathcal{L}(W^{[1]}, W^{[2]}, \ldots, W^{[M-1]}, b^{[1]}, b^{[2]}, \ldots, b^{[M-1]}, y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} \frac{1}{2} \|\hat{y}_i - y_i\|_2^2$$

We can ignore each y_i here since it's a constant during the computation, so

$$\mathcal{L}(W^{[1]}, W^{[2]}, \ldots, W^{[M-1]}, b^{[1]}, b^{[2]}, \ldots, b^{[M-1]}) = \sum_{i=1}^{n} \frac{1}{2} \|\hat{y}_i - y_i\|_2^2$$

Now our goal is finding each $W^{[k]}$ and $b^{[k]}$ to let each $\hat{y}_i$ be closer to y_i by minimizing $\mathcal{L}$. The process of minimizing the loss function is called *training*, not being specific to back propagation.

Let's simplify our notation. Let $\theta = (W^{[1]}, \cdots, W^{[M-1]}, b^{[1]}, \cdots, b^{[M-1]})$, then $\theta \in \mathbb{R}^D$, where

$$D = \sum_{k=1}^{M-1} n_k \times n_{k+1} + \sum_{k=1}^{M-1} n_k$$

Then the training problem becomes like this

$$\arg\min_{\theta} \sum_{i=1}^{n} \frac{1}{2}\|\hat{y}_i - y_i\|_2^2 = \arg\min_{\theta} \mathcal{L}(\theta), \text{where } \mathcal{L} : \mathbb{R}^D \to \mathbb{R}$$

1.3 Main Computations

To compute $\nabla L(\theta)$. Let $z^{[k]} = W^{[k]}a^{[k-1]} + b^{[k]}$ then $a^{[k]} = \sigma(z^{[k]})$. Let j be the index represents neurons of each layer. The notation becomes

$$\begin{aligned}
&z_j^{[k]} = (W^{[k]}a^{[k-1]} + b^{[k]})_j = (W^{[k]}a^{[k-1]})_j + b_j^{[k]},\\
&a_j^{[k]} = \sigma(z^{[k]})_j,\\
&k = 2, \dots, M, \quad j = 1, \dots, n_k
\end{aligned}$$

The core concept of back propagation is using deep level derivatives to compute shallow level derivative. This can be achieved with the help of chain rule. First we derive the derivative of $z_j^{[k]}$(the j^{th} neuron of the k^{th} layer).

$$\begin{aligned}
\frac{\partial L(\theta)}{\partial z_j^{[M]}} &= \frac{\partial L(\theta)}{\partial a_j^{[M]}}\frac{\partial a_j^{[M]}}{\partial z_j^{[M]}} = \frac{\partial L(\theta)}{\partial a_j^{[M]}}\sigma'(z_j^{[M]}) = (a_j^{[M]} - y_j)\sigma'(z_j^{[M]}) = (a_j^{[M]} - y_j)a_j^{[M]}(1 - a_j^{[M]}),\\
\frac{\partial L(\theta)}{\partial z_j^{[M-1]}} &= \frac{\partial}{\partial z_j^{[M-1]}} \frac{1}{2}\sum_{k=1}^{n_M}\Big(y_k - \sigma\big(W^{[M]}\sigma(z^{[M-1]}) + b^{[M]}\big)_k\Big)^2\\
&= \nabla_{z^{[M]}} L(\theta) \cdot \frac{\partial z^{[M]}}{\partial z_j^{[M-1]}}\\
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} \frac{\partial z_m^{[M]}}{\partial z_j^{[M-1]}}\\
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} \left(\frac{\partial}{\partial z_j^{[M-1]}} \sum_{s=1}^{n_{M-1}} w_{ms}^{[M]}\sigma(z_s^{[M-1]}) + b_m^{[M]}\right)\\
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} w_{mj}^{[M]}\sigma'(z_j^{[M-1]})\\
&= \sigma'(z_j^{[M-1]})\left((W^{[M]})^T\frac{\partial L(\theta)}{\partial z^{[M]}}\right)_j\\
&= \left((W^{[M]})^T\frac{\partial L(\theta)}{\partial z^{[M]}}\right)_j a_j^{[M-1]}(1 - a_j^{[M-1]}),\\
&\quad\vdots
\end{aligned}$$

$$\frac{\partial L(\theta)}{\partial z_j^{[1]}} = \left((W^{[2]})^T \frac{\partial L(\theta)}{\partial z^{[2]}}\right)_j a_j^{[1]}(1 - a_j^{[1]})$$

From above we can see that we *back propagate* the derivative since we need $\partial z^{[M]}$ to yield $\partial z_j^{[M-1]}$. The partial derivative of $z_j^{[k]}$ can help us get the gradient w.r.t weight and bias.

$$\begin{aligned}\frac{\partial L(\theta)}{\partial b_j^{[k]}} &= \frac{\partial L(\theta)}{\partial z_j^{[k]}} \frac{\partial z_j^{[k]}}{\partial b_j^{[k]}} = \frac{\partial L(\theta)}{\partial z_j^{[k]}}, \\ \frac{\partial L(\theta)}{\partial w_{js}^{[k]}} &= \frac{\partial L(\theta)}{\partial z_j^{[k]}} \frac{\partial z_j^{[k]}}{\partial w_{js}^{[k]}} = \frac{\partial L(\theta)}{\partial z_j^{[k]}} a_s^{[k]}\end{aligned}$$

And hence we can back propagate each $\partial b_j^{[k]}$ and $\partial w_{js}^{[k]}$.

For the case with multiple training instances (x_i, y_i), $i = 1, \ldots, N$. The loss function becomes

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \|y_i - a_i^{[[M]}\|_2^2$$

The computation are all the same but the notation will become a little bit complicated.