1 Derivations

Consider a data set, with D_{in} dimensional feature spaces and D_{out}-dimensional output (target) space. We model or approximate the function that relates the feature and target using deep neural network. To start with, let $x \in X_{train}$ be a feature vector. We will use fully connected deep neural network architecture. Specifically, we devide layers into three categories, input layer with D_{in} input units, hidden layers, and output layer with D_{out} units. Let L, $h_{hid}(x)$, $h_{out}(x)$ be the number of hidden layers, the activation function of hidden layers, and the activation function of output layer respectively. We define an activation corresponding to l-th hidden layer as

$$a_j^l = \sum_k^{D_l} W_{jk}^l \, z_k^{l-1} + b_j^l \,, \qquad 1 \leq l \leq L \tag{1}$$

where D_l is the number of units in the hidden layer l and

$$z_j^l = h_{hid}(a_j^l) \,, \qquad z_k^0 = x_j \tag{2}$$

For output layer

$$a_k^{out} = \sum_k^{D_{out}} W_{jk}^{out} + b_j^{out} \,, \qquad y_k = h_{out}(a_k^{out}) \tag{3}$$

Our purpose is to estimate or train the weights and biases that fit to the training data the best. To do that, we will use maximum likelihood estimation, which is equivalent to minimize the following Loss function

$$E(W^l, W^{out}, b^l, b^{out}) = \sum_n E_n = \sum_n \frac{1}{2} \sum_k^{D_{out}} [y_{kn} - t_{kn}]^2 \tag{4}$$

We will use gradient descent (GD) to find such minimum. Before using GD, we need to find the derivative of the total loss function w.r.t weights and biases. Backpropagation is the most convenient methode for that, for its speed and simplicity.

Using computational graph, one can see that the the individual loss E_n is a function of a^{out} alone, while a^{out} itself is a function of W^{out} and z^L. Breaking this further down, we can infer that in general, the loss is a function of $a^l, W^{l+1}, W^{l+2}, ..., W^L, W^{out}, b^{l+1}, ..., b^{out}$ for arbitrary $1 \leq l \leq L$. Therefore

$$\begin{aligned}
\frac{\partial E_n}{\partial W_{ij}^l} &= \frac{\partial E_n}{\partial a_i^l} \frac{\partial a_i^l}{\partial W_{ij}^l} = \frac{\partial E_n}{\partial a_i^l} z_j^{l-1} \equiv \delta_i^{n,l} z_j^{l-1} \\
\frac{\partial E_n}{\partial W_{ij}^{out}} &= \sum_{k,p} [y_{kn} - t_{kn}] \frac{\partial h_{out}(a_k^{out})}{\partial a_p^{out}} \frac{\partial a_p^{out}}{\partial W_{ij}^{out}} = [y_{in} - t_{in}] \, h'_{out}(a_i^{out}) \, z_j^L \\
\frac{\partial E_n}{\partial b_i^l} &= \frac{\partial E_n}{\partial a_i^l} \frac{\partial a_i^l}{\partial b_i^l} = \frac{\partial E_n}{\partial a_i^l} \equiv \delta_i^{n,l} \\
\frac{\partial E_n}{\partial b_i^{out}} &= \sum_{k,p} [y_{kn} - t_{kn}] \frac{\partial h_{out}(a_k^{out})}{\partial a_p^{out}} \frac{\partial a_p^{out}}{\partial b_i^{out}} = [y_{in} - t_{in}] \, h'_{out}(a_i^{out})
\end{aligned} \tag{5}$$

where we have used the fact that all Ws and bs are independent to each other. The error δ^l can be computed recursively using

$$\delta_i^{n,l} = \frac{\partial E_n}{\partial a_i^l} = \sum_{j,k} \frac{\partial E_n}{\partial a_j^{l+1}} \frac{\partial a_j^{l+1}}{\partial z_k^l} \frac{\partial z_k^l}{\partial a_i^l} = h'_{hid}(a_i^l) \sum_j \delta_j^{n,l+1} W_{ji}^{l+1} \tag{6}$$

From (10), one can also obtain

$$\delta_j^{n,L+1} \equiv \delta_j^{n,out} = [y_{jn} - t_{jn}]\, h'_{out}(a_j^{out}) \tag{7}$$

2 Training Algorithm

1. Initialize $W^l, W^{out}, b^l, b^{out}$
2. Compute the derivative $\partial E_n/\partial W^l$, $\partial E_n/\partial W^{out}$, $\partial E_n/\partial W^l$ as
   - Forward Propagation steps. Compute a^l, a^{out}, z^l using the following

$$a_j^l = \sum_k^{D_l} W_{jk}^l\, z_k^{l-1} + b_j^l\,, \qquad z_j^l = h_{hid}(a_j^l)\,, \qquad z_k^0 = x_j\,, \quad a_k^{out} = \sum_k^{D_{out}} W_{jk}^{out} + b_j^{out} \tag{8}$$

   - Backpropagation steps.
     - Compute $\delta^{n,l}, \delta^{n,out}$ using

$$\delta_j^{n,L+1} \equiv \delta_j^{n,out} = [y_{jn} - t_{jn}]\, h'_{out}(a_j^{out})\,, \quad \delta_i^{n,l} = h'_{hid}(a_i^l) \sum_j \delta_j^{n,l+1} W_{ji}^{l+1} \tag{9}$$

     - Compute the derivative of E_n with respect to $W^l, W^{out}, b^l, b^{out}$ as

$$\begin{aligned} \frac{\partial E_n}{\partial W_{ij}^l} &= \frac{\partial E_n}{\partial a_i^l} z_j^{l-1} \equiv \delta_i^{n,l} z_j^{l-1}\,, \quad \frac{\partial E_n}{\partial W_{ij}^{out}} = [y_{in} - t_{in}]\, h'_{out}(a_i^{out})\, z_j^L \\ \frac{\partial E_n}{\partial b_i^l} &= \frac{\partial E_n}{\partial a_i^l} \equiv \delta_i^{n,l}\,, \quad \frac{\partial E_n}{\partial b_i^{out}} = [y_{in} - t_{in}]\, h'_{out}(a_i^{out}) \end{aligned} \tag{10}$$

3. Update $W^l, W^{out}, b^l, b^{out}$ using stochastic gradient descent method:

$$X(t+1) = X(t) - \eta \frac{\partial E_n}{\partial X}\,, \qquad X = \{W^l, W^{out}, b^l, b^{out}\} \tag{11}$$

   with η is the learning rate.
4. Repeat (2)-(3) until convergence criterion achieved.