COMP 2211 Exploring Artificial Intelligence

Multilayer Perceptron - Derivation of Backpropagation

Dr. Desmond Tsoi

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology, Hong Kong SAR, China

Sigmoid Function

- Sigmoid function is typically used as a transfer function between neurons. It is continuous and differentiable:

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

- Useful property of sigmoid function is the simplicity of computing its derivative.

$$\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$$

Error Calculation

- Given a set of training data point t_k and output layer output O_k, we can write the error as:

$$E = \frac{1}{2} \sum_{k \in K} (O_k - t_k)^2$$

- We want to calculate $\frac{\partial E}{\partial W_{ij}}$ and $\frac{\partial E}{\partial W_{jk}}$. The rate of change of the error with respect to the given connective weight, so we can minimize them.
- Now, we consider two cases:
  1. Output layer node
  2. Hidden layer node

Case 1: Output Layer Node

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{k\in K}(O_k - t_k)^2 \\
\frac{\partial E}{\partial W_{jk}} &= \frac{\partial(\frac{1}{2}\sum_{k\in K}(O_k - t_k)^2)}{\partial W_{jk}} \\
&= (O_k - t_k)\frac{\partial O_k}{\partial W_{jk}} = (O_k - t_k)\frac{\partial \sigma(x_k)}{\partial W_{jk}} \\
&= (O_k - t_k)\sigma(x_k)(1-\sigma(x_k))\frac{\partial x_k}{\partial W_{jk}} \\
&= (O_k - t_k)O_k(1-O_k)O_j
\end{aligned}
$$

Note

The summation disappears in the derivative. It is because when we take the partial derivative with respect to the j-th node, the only term that survives in the error is j-th, and thus we can ignore the remaining terms in the summation.

- $\sigma(x_k) = O_k$
- $\frac{\partial x_k}{\partial W_{jk}} = \frac{\partial(W_{jk}O_j)}{\partial W_{jk}} = O_j$

- For notation purposes, define δ_k to be the expression $(O_k - t_k)O_k(1-O_k)$, so we can rewrite the equation above as

$$
\frac{\partial E}{\partial W_{jk}} = O_j\delta_k
$$

Case 2: Hidden Layer Node

$$
\begin{aligned}
\frac{\partial E}{\partial W_{ij}} &= \frac{\partial(\frac{1}{2}\sum_{k\in K}(O_k - t_k)^2)}{\partial W_{ij}} = \sum_{k\in K}(O_k - t_k)\frac{\partial O_k}{\partial W_{ij}} = \sum_{k\in K}(O_k - t_k)\frac{\partial \sigma(x_k)}{\partial W_{ij}} \\
&= \sum_{k\in K}(O_k - t_k)\sigma(x_k)(1-\sigma(x_k))\frac{\partial x_k}{\partial W_{ij}} = \sum_{k\in K}(O_k - t_k)O_k(1-O_k)\frac{\partial x_k}{\partial O_j}\cdot\frac{\partial O_j}{\partial W_{ij}} \\
&= \sum_{k\in K}(O_k - t_k)O_k(1-O_k)\frac{\partial (W_{jk}O_j)}{\partial O_j}\cdot\frac{\partial O_j}{\partial W_{ij}} \\
&= \sum_{k\in K}(O_k - t_k)O_k(1-O_k)W_{jk}\cdot\frac{\partial O_j}{\partial W_{ij}} = \frac{\partial O_j}{\partial W_{ij}}\cdot\sum_{k\in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= \frac{\partial \sigma(x_j)}{\partial W_{ij}}\cdot\sum_{k\in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= \frac{\partial \sigma(x_j)}{\partial x_j}\frac{\partial x_j}{\partial W_{ij}}\cdot\sum_{k\in K}(O_k - t_k)O_k(1-O_k)W_{jk}
\end{aligned}
$$

Note

The summation does not disappear because the layers are fully connected, each of the hidden unit outputs affect the state of each output unit.

Case 2: Hidden Layer Node (Cont'd)

$$
\begin{aligned}
\frac{\partial E}{\partial W_{ij}} &= \sigma(x_j)(1-\sigma(x_j))\frac{\partial x_j}{\partial W_{ij}} \cdot \sum_{k \in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= O_j(1-O_j)\frac{\partial x_j}{\partial W_{ij}} \cdot \sum_{k \in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= O_j(1-O_j)\frac{\partial (W_{ij}O_i)}{\partial W_{ij}} \cdot \sum_{k \in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= O_j(1-O_j)O_i \cdot \sum_{k \in K}(O_k - t_k)O_k(1-O_k)W_{jk} \\
&= O_i O_j(1-O_j)\sum_{k \in K}\delta_k W_{jk}
\end{aligned}
$$

- Similar to before, we will define all terms besides O_i to be $\delta_j = O_j(1-O_j)\sum_{k \in K}\delta_k W_{jk}$, so we have

$$
\frac{\partial E}{\partial W_{ij}} = O_i \delta_j
$$

How Weights Affect Errors?

- For an output layer node $k \in K$

$$\frac{\partial E}{\partial W_{jk}} = O_j \delta_k$$

where $\delta_k = (O_k - t_k) O_k (1 - O_k)$

- For a hidden layer node $j \in J$

$$\frac{\partial E}{\partial W_{ij}} = O_i \delta_j$$

where $\delta_j = O_j (1 - O_j) \sum_{k \in K} \delta_k W_{jk}$

How About the Bias?

- If we incorporate the bias term θ into the equation you will find that

$$\frac{\partial O}{\partial \theta} = 1$$

- This is why we view the bias term as output from a node which is always one. This holds for any layer l, a substitution into the previous equations gives us that

$$\frac{\partial E}{\partial \theta} = \delta_l$$

The Back Propagation Algorithm using Gradient Descent

1. Run the network forward with your input data to get the network output.
2. For each output node, compute

$$\delta_k = (O_k - t_k)O_k(1 - O_k)$$

3. For each hidden node, compute

$$\delta_j = O_j(1 - O_j)\sum_{k \in K} \delta_k W_{jk}$$

4. Update the weights and biases as follows:
   - Given

$$\Delta W_{xy} = -\eta \delta_y O_x$$
$$\Delta\theta = -\eta \delta_x$$

   - Apply

$$W_{xy} \leftarrow W_{xy} + \Delta W_{xy}$$
$$\theta \leftarrow \theta + \Delta\theta$$

where η denotes learning rate. Typically, η is a value between 0 and 1.

That's all!

Any questions?