

Neural Network Assignment

- Attached is a derivation for the backpropagation rules for Perceptrons that use a sigmoidal excitation function. Derive an appropriate error function and weight update rule for Perceptrons that use a tanh excitation function. Make the rules as simple as possible by choosing appropriate substitutions – such as

$$o_{i[h]} \text{ for } \tanh(\vec{x}_i \cdot \vec{w}_i)_{i[h]}. \text{ (Hint: } \frac{\partial(\tanh(x))}{\partial x} = \text{sech}^2(x) = 1 - \tanh^2(x)\text{)}$$

Derivation of Backpropagation for Sigmoid

For each time step, we present all training data to our multi-layered Perceptron. The change in the weight from “neuron” i to “neuron” j (where i 's output is being fed into j) is given by

$$\Delta w_{i,j} = \beta \sum_{\langle x,c \rangle \in T} \frac{\partial P_{\langle x,c \rangle}}{\partial w_{i,j}}.$$

Where β is a parameter that controls how quickly the weights change, and $P_{\langle x,c \rangle}$ is the performance of the network for the training data with input x and correct output c . This performance is defined as

$$P_{\langle x,c \rangle} = -(F(\vec{w}, \vec{x}) - c)^2.$$

If we define the output error (i.e., the error at the last level of our Perceptron) to be

$$\mathbf{d}_{out} = -(F(\vec{w}, \vec{x}) - c),$$

then clearly we can rewrite the performance as

$$P_{\langle x,c \rangle} = -\mathbf{d}_{out}^2.$$

We can now calculate the derivative of our performance with respect to a given weight $w_{i,j}$ as

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i,j}} = -2\mathbf{d} \frac{\partial \mathbf{d}}{\partial w_{i,j}} = 2\mathbf{d} \frac{\partial F(\vec{w}, \vec{x})}{\partial w_{i,j}}$$

We will now adopt the convention that neuron i will be identified as an input neuron by the addition of the symbol [i], as a hidden neuron by the addition of the symbol [h] (we could use [h1] and [h2] if we had two hidden layers, and as an output neuron by the addition of the symbol [o]). Therefore a weight connecting input neuron i to hidden neuron j would be represented by $w_{i[j],j[h]}$. Additionally, all inputs to a hidden neuron i will be represented as \vec{x}_i , and the weights associated with those inputs by \vec{w}_i . The output of this neuron is therefore $f(\vec{x}_i \cdot \vec{w}_i)_{i[h]}$ or $o_{i[h]}$. We will also use the convention that for an output neuron j , the inputs can be represented as $\vec{f}(\vec{x} \cdot \vec{w})$, so that the output from that neuron is $f(\vec{f}(\vec{x} \cdot \vec{w}_{in}) \cdot \vec{w}_{hid})$ or $o_{j[o]}$. This notation allows us to see that there are two special cases to consider when taking the derivative of $F(\vec{w}, \vec{x})$ with respect to $w_{i,j}$.

The first case is if $w_{i,j} \in \vec{w}_{hid}$. Using the chain rule we find

$$\frac{\partial F(\vec{w}, \vec{x})}{\partial w_{i[h],j[o]}} = \tilde{f}(\vec{f}(\vec{x} \cdot \vec{w}_{in}) \cdot \vec{w}_{hid}) f(\vec{x}_i, \vec{w}_i)_{i[h]},$$

where $\tilde{f}(x) \equiv \frac{\partial f(x)}{\partial x}$. For the sigmoid function $\tilde{f}(x) = f(x)(1 - f(x))$.

The second case is if $w_{i,j} \in \vec{w}_{in}$. After a double application of the chain rule we find

$$\frac{\partial F(\vec{w}, \vec{x})}{\partial w_{i[i],j[h]}} = \tilde{f}(\vec{f}(\vec{x} \cdot \vec{w}_{in}) \cdot \vec{w}_{hid}) w_{hid} \tilde{f}(\vec{x} \cdot \vec{w})_{j[h]} x_i.$$

This leads to performance derivatives that can be written as

Neural Network Assignment

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i[h],j[o]}} = 2f(\bar{x}_i, \bar{w}_i)_{i[h]} \tilde{f}(\bar{f}(\bar{x} \cdot \bar{w}_{in}) \cdot \bar{w}_{hid}) \mathbf{d}_{out}, \text{ or}$$

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i[h],j[o]}} = 2f(\bar{x}_i, \bar{w}_i)_{i[h]} f(\bar{f}(\bar{x} \cdot \bar{w}_{in}) \cdot \bar{w}_{hid}) (1 - f(\bar{f}(\bar{x} \cdot \bar{w}_{in}) \cdot \bar{w}_{hid})) \mathbf{d}_{out},$$

for the hidden neurons. If we use the alternative identifications $o_{i[h]} \equiv f(\bar{x}_i \cdot \bar{w}_i)_{i[h]}$ and $o_{j[o]} = f(\bar{f}(\bar{x} \cdot \bar{w}_{in}) \cdot \bar{w}_{hid})$, this becomes

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i[h],j[o]}} = 2o_{i[h]}o_{j[o]}(1 - o_{j[o]}) \mathbf{d}_{out}.$$

Likewise, for the input neurons, we find

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i[i],j[h]}} = 2\tilde{f}(\bar{f}(\bar{x} \cdot \bar{w}_{in}) \cdot \bar{w}_{hid}) w_{hid} \tilde{f}(\bar{x} \cdot \bar{w})_{j[h]} x_i \mathbf{d}_{out}, \text{ or}$$

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i[i],j[h]}} = 2x_i o_{j[h]} (1 - o_{j[o]}) w_{j[h],k[o]} o_{j[h]} (1 - o_{j[h]}) \mathbf{d}_{out}.$$

If we define the error for the hidden neurons to then be $\mathbf{d}_{j[h]} \equiv w_{j[h],k[o]} o_{j[h]} (1 - o_{j[o]}) \mathbf{d}_{out}$, and the inputs x_i to be considered as the outputs from the input neurons $o_{i[i]}$, then we can finally combine both equations into

$$\frac{\partial P_{\langle x,c \rangle}}{\partial w_{i,j}} = 2o_i o_j (1 - o_j) \mathbf{d}_j.$$