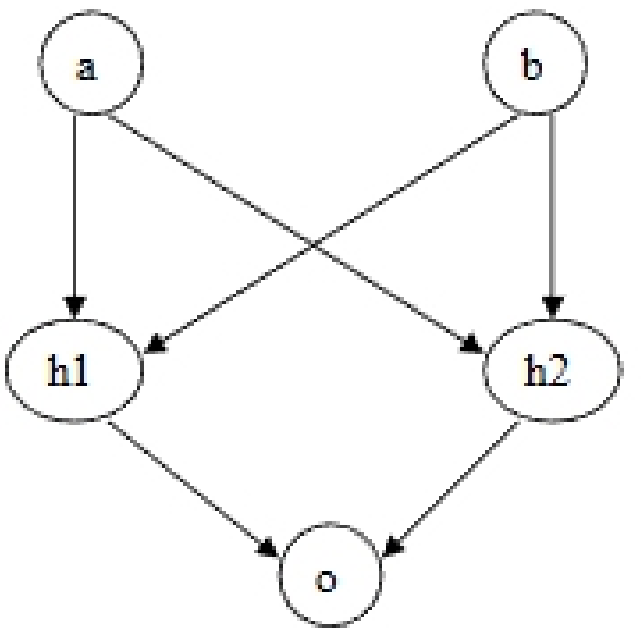
Neural Network Assignment
Due April 10, 2003

1. You are given a simple Perceptron with 3 inputs, A, B, and C that uses a step function with threshold of 0.5. Find weights between 0 and 1 such that the output is:
    a. (A and B) or C
    b. A and (B or C)
2. You are given the following Perceptron to train on the xor function. Fill out the table provided in order to determine what the new weights will be after two training iterations. Use the weight modification rules provided in class for a sigmoidal excitation function, with the following change. For the hidden and output neurons, there is a bias that is applied to the sigmoid functions. This can be interpreted as implied neuron that always fires and whose weight is set equal to that bias. This allows one to learn biases for the neurons as well as learning the weights. Use $\beta = 0.5$.



					$W_{bias,h1}$	$W_{a,h1}$	$W_{b,h1}$	$W_{bias,h2}$	$W_{a,h2}$	$W_{b,h2}$	$W_{bias,o}$	$W_{h1,o}$	$W_{h2,o}$
Initial	weights				0.68	0.38	0.83	0.5	0.71	0.43	0.30	0.19	0.19
Timestep	Case	err_o	err_{h1}	err_{h2}	$\Delta w_{bias,h1}$	$\Delta w_{a,h1}$	$\Delta w_{b,h1}$	$\Delta w_{bias,h2}$	$\Delta w_{a,h2}$	$\Delta w_{b,h2}$	$\Delta w_{bias,o}$	$\Delta w_{h1,o}$	$\Delta w_{h2,o}$
1	a=0,b=0		-0.0279		-0.00312	0	0		0	0			
	a=0,b=1												
	a=1,b=0												
	a=1,b=1												
					$W_{bias,h1}$	$W_{a,h1}$	$W_{b,h1}$	$W_{bias,h2}$	$W_{a,h2}$	$W_{b,h2}$	$W_{bias,o}$	$W_{h1,o}$	$W_{h2,o}$
New	weights					0.37989							
2	a=0,b=0												
	a=0,b=1												
	a=1,b=0												
	a=1,b=1												

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

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

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

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

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

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

$$\delta_{out} = -\left(F(\vec{w},\vec{x}) - c\right),$$

then clearly we can rewrite the performance as

$$P_{(x,c)} = -\delta_{out}^2.$$

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

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

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 $w_{i[i],j[h]}$. 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 $f\left(\vec{f}(\vec{x} \cdot \vec{w}_{in}) \cdot \vec{w}_{hid}\right)$ 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

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

where $\tilde{f}(x) \equiv {\partial f(x)}/{\partial x}$. 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

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

This leads to performance derivatives that can be written as