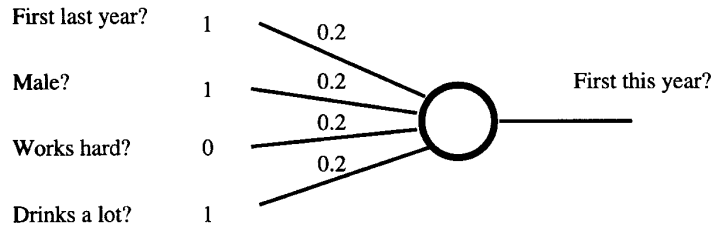


COMP 6721 Applied Artificial Intelligence (Winter 2021)

Worksheet #5: Neural Networks

Perceptron. Calculate your first neuron activation for the *Perceptron* (only 100 billion—1 more to go!):



Activation function:

$$f(\vec{x}) = \begin{cases} 1, & \text{if } \vec{x} \cdot \vec{w} \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

(use a threshold of 0.55):

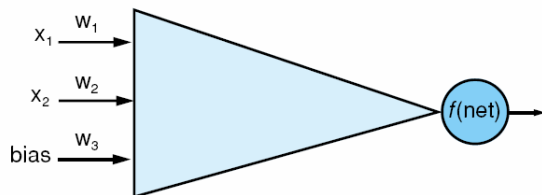
$$f(\vec{x}) = \dots\dots\dots$$

Perceptron Learning. Ok, so for the first training example, the perceptron did not produce the right output. To learn the correct result, it has to adjust the weights: $\Delta w = \eta(T - O)$, where $\eta = 0.05$ (our *learning rate*). T is the expected output and O the output produced by the perceptron. Remember to only update weights for *active* connections:

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	'A' this year?
Richard	Yes	Yes	No	Yes	No
Alan	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	No

Student	w_1	w_2	w_3	w_4	$f(\vec{x})$	ok?
Richard	0.2	0.2	0.2	0.2		
Alan						
Alison						

Delta Rule. In the generalized delta rule for training the perceptron, we add a *bias* input that is always one and has its own weight (here w_3). We want the perceptron to learn the two-dimensional data shown on the right:



x_1	x_2	Output
1.0	1.0	1
9.4	6.4	-1
2.5	2.1	1
8.0	7.7	-1
0.5	2.2	1

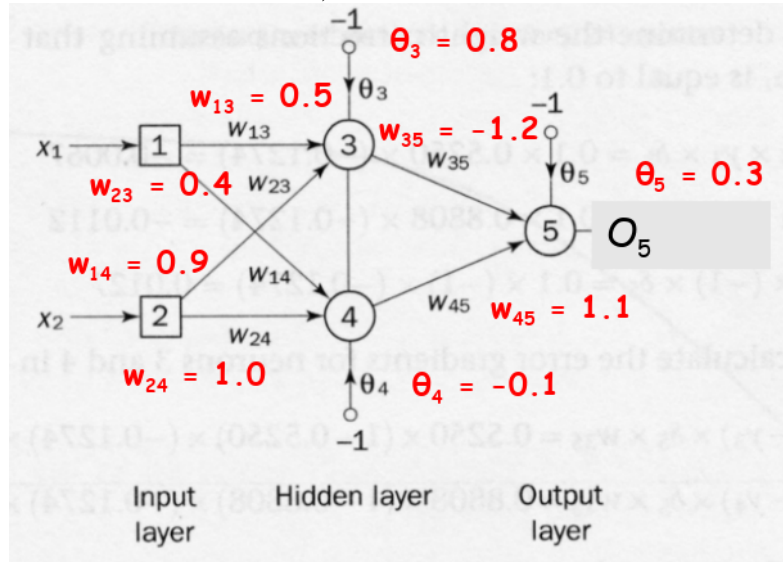
Assume we use the sign function with a threshold of 0 and a learning rate $\eta = 0.2$. The weights are initialized randomly as shown in the table. Apply the generalized delta rule for updating the weights: $\Delta w_i = \eta(T - O)x_i$

Data	w_1	w_2	w_3	$f(\vec{x})$	ok?
#1	0.75	0.5	-0.6		
#2					
#3					
#4					

Neural Network for XOR. To learn a non-linearly separable function like XOR, we'll use a neural network with a hidden layer (the weights have been initialized randomly):

x_1	x_2	$x_1 \text{ XOR } x_2$
1	1	0
0	0	0
1	0	1
0	1	1

$$O_i = \text{sigmoid}\left(\sum_j w_{ji}x_j\right) = \frac{1}{1 + e^{-(\sum_j w_{ji}x_j)}}$$



Step 1. Compute the output for the three neurons O_3, O_4 and O_5 for the input ($x_1 = 1, x_2 = 1$):

$O_3 = \dots\dots\dots O_4 = \dots\dots\dots O_5 = \dots\dots\dots$

Step 2. The next step is to calculate the error

$$\delta_k \leftarrow g'(x_k) \times \text{Err}_k = O_k(1 - O_k) \times (O_k - T_k)$$

starting from the output neuron O_5 : $\delta_5 = O_5(1 - O_5) \times (O_5 - T_5) = \dots\dots\dots$

Step 3. Now we calculate the error terms for the hidden layer:

$$\delta_h \leftarrow g'(x_h) \times \text{Err}_h = O_h(1 - O_h) \times \sum_{k \in \text{Outputs}} w_{kh} \delta_k$$

For the two neurons (3), (4) in the hidden layer:

- $\delta_3 = O_3(1 - O_3)\delta_5 w_{35} = \dots\dots\dots$
- $\delta_4 = O_4(1 - O_4)\delta_5 w_{45} = \dots\dots\dots$

Step 4. Now we can update our weights using

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}, \text{ where } \Delta w_{ij} = -\eta \delta_j x_i$$

Here, we assume a constant learning rate $\eta = 0.1$:

- $\Delta w_{14} = \dots\dots\dots$
- $\Delta w_{24} = \dots\dots\dots$
- $\Delta w_{45} = \dots\dots\dots$
- $\Delta \theta_5 = \dots\dots\dots$

And finally update the weights ($w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$):

- $w_{14} = w_{14} + \Delta w_{14} = \dots\dots\dots$
- $w_{24} = w_{24} + \Delta w_{24} = \dots\dots\dots$
- $w_{45} = w_{45} + \Delta w_{45} = \dots\dots\dots$
- $\theta_5 = \theta_5 + \Delta \theta_5 = \dots\dots\dots$