Neural Nets 2

1. **For epoch 1:**

   $g()$ is $sgn()$ and we use $sgn'() = 1$

   $w0 = 1.5$ and $w1 = 0$ the initial value of the weights of the neuron

   x the input vector of size 1

   $\{2 \rightarrow 3, 1 \rightarrow -1\}$ the learning set with 2 examples

   $\eta = 0.1$ the learning rate

   For x_1: $o(x_1) = sgn(1.5 \times 1 + 0 \times 2) = 1$

   $w_0 \leftarrow 1.5 + 0.1 \times (3 - 1) \times 1$

   $w_0 \leftarrow 1.7$

   $w_1 \leftarrow 0 + 0.1 \times (3 - 1) \times 2$

   $w_1 \leftarrow 0.4$

   For x_2: $o(x_2) = sgn(1.7 \times 1 + 0.4 \times 1) = 1$

   $w_0 \leftarrow 1.7 + 0.1 \times (-1 - 1) \times 1$

   $w_0 \leftarrow 1.5$

   $w_1 \leftarrow 0.4 + 0.1 \times (-1 - 1) \times 1$

   $w_1 \leftarrow 0.2$

   For epoch 2:

   $w0 = 1.5$ and $w1 = 0.2$

   For x 1: $o(x$ 1$) = sgn(1.5 \times 1 + 0.2 \times 2) = 1$

   $w_0 \leftarrow 1.5 + 0.1 \times (3 - 1) \times 1$

   $w_0 \leftarrow 1.7$

   $w_1 \leftarrow 0.2 + 0.1 \times (3 - 1) \times 2$

   $w_1 \leftarrow 0.6$

   For x 2: $o(x$ 2$) = sgn(1.7 \times 1 + 0.6 \times 1) = 1$

   $w_0 \leftarrow 1.7 + 0.1 \times (-1 - 1) \times 1$

   $w_0 \leftarrow 1.5$

   $w_1 \leftarrow 0.6 + 0.1 \times (-1 - 1) \times 1$

   w 1$\leftarrow 0.4$

   Which just goes to show that you do not always get convergence.

2. (a) first output unit input $= -0.3 \times 0.6 + 0.9 \times 0.8 = 0.54$, $activation = 0.632$
   second output unit input $= -0.6 \times 0.6 + -0.1 \times 0.8 = -0.44$, $activation = 0.392$
   third output unit input $= 0.4 \times 0.6 + 1.2 \times 0.8 = 1.2$, $activation = 0.769$

(b) first output unit $= (0.1 - 0.632) \times 0.632 \times (1 - 0.632) = -0.124$
second output unit $= (0.9 - 0.392) \times 0.392 \times (1 - 0.392) = 0.121$
third output unit $= (0.1 - 0.769) \times 0.769 \times (1 - 0.769) = -0.119$

(c) for hidden unit $b = 0.6 \times (1 - 0.6) \times [-0.124 \times -0.3 + 0.121 \times -0.6 + -0.119 \times 0.4] = -0.02$
for hidden unit $c = 0.8 \times (1 - 0.8) \times [-0.124 x 0.9 + 0.121 \times -0.1 + -0.119 \times 1.2] = -0.04$

(d) Weight changes for weights connecting from unit a
Original weight -0.8 becomes $-0.8 + (0.25 \times -0.02) \times 0.9 = -0.805$ Original weight -0.3 becomes $-0.3 + (0.25 \times -0.04) \times 0.9 = -0.309$

3. See lecture notes

4. See lecture notes

5. For simplicity, we will assume that the activation function is the same linear function at each node: $g(x) = cx + d$. (The argument is the same (only messier) if we allow different c_i and d_i for each node.)

(a) The outputs of the hidden layer are

$$H_j = g\left(\sum_k W_{k,j} I_k\right) = c \sum_k W_{k,j} I_k + d$$

The final outputs are

$$O_i = g\left(\sum_j W_{j,i} H_j\right) = c\left(\sum_j W_{j,i}\left(c \sum_k W_{k,j} I_k + d\right)\right) + d$$

Now we just have to see that this is linear in the inputs:

$$O_i = c^2 \sum_k I_k \sum_j W_{k,j} W_{j,i} + d\left(1 + c \sum_j W_{j,i}\right)$$

Thus we can compute the same function as the two-layer network using just a one-layer perceptron that has weights $W_{k,i} = \sum_j W_{k,j} W_{j,i}$ and an activation function $g(x) = c^2 x + d\left(1 + c \sum_j W_{j,i}\right)$.

(b) The above reduction can be used straightforwardly to reduce an n-layer network to an $(n-1)$-layer network. By induction, the n-layer network can be reduced to a single layer network. Thus, linear activation function restrict neural networks to represent only linearly functions.

6. Intuitively, the data suggest that a probabilistic prediction $P(Output = 1) = 0.8$ is appropriate. The network will adjust its weights to minimize the error function. The error is

$$E = \frac{1}{2} \sum_i (y_i - a_i)^2 = \frac{1}{2}\left[80\,(1 - a_1)^2 + 20\,(0 - a_1)^2\right] = 50 O_1^2 - 80 O_1 + 50$$

The derivative of the error with respect to the single output $a1$ is:

$$\frac{\partial E}{\partial a_1} = 100a_1 - 80$$

Setting the derivative to zero, we find that indeed $a1 = 0.8$.