Written Assignment 3

Isobel Smith

02/11/2016

1 Question 3: Neural Network

We have a neural network with 1 hidden layer. Input and hidden layer both have 2 nodes. There is 1 output node. The values of theta for bias nodes are 0.2. The vector $v^{(1)}$ for layer 1 is: [0.5,0.1,0.5,0.7] and $v^{(2)}$ for layer 2 is [1,2].

1.1 a) Calculate the activations of all nodes for x1 = 0.5 and x2 = 0.9

$a_i^{(j)}$ is the activation of unit i in layer j
$\theta^{(j)}$ is the matrix of weights controlling function mapping from layer j to layer j+1

Activation:

$$\begin{aligned} a_1^{(2)} &= g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2) \\ a_2^{(2)} &= g(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2) \\ h_\theta(x) = a_3^{(1)} &= g(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)}) \end{aligned}$$

For each layer we apply the sigmoid function on the linear combination.

The sigmoid function $= h_\theta^{(x)} = \frac{1}{1+e^{\theta_x^T}}$

The vector $v^{(1)}$ for layer 1 is: [0.5,0.1,0.5,0.7] and $v^{(2)}$ for layer 2 is [1,2]. These corrospond to the weights, or the θ values.

lets assume the value of the bias nodes are 1.

So the activation values are:

$$\begin{aligned}
a_1^{(2)} &= g(0.2 \cdot 1 + 0.5 \cdot 0.5 + 0.5 \cdot 0.9) \\
&= g(0.9) \\
&= \frac{1}{1 + e^{-0.9}} \\
&= 0.711 \\
a_2^{(2)} &= g(0.2 \cdot 1 + 0.1 \cdot 0.5 + 0.7 \cdot 0.9) \\
&= g(0.88) \\
&= \frac{1}{1 + e^{-0.88}} \\
&= 0.707 \\
h_\theta(x) = a_3^{(1)} &= g(0.2 \cdot 1 + \ \cdot^{(2)} a_1^{(2)} + 2 \cdot a_2^{(2)}) \\
&= g(2.323) \\
&= \frac{1}{1 + e^{-2.323}} \\
&= 0.911
\end{aligned}$$

1.2 b) Suppose the correct output is 1. Calculate the errors for all nodes and the updates of the weights (for one iteration)

To calculate the error we use back propogation.

$\delta_j^{(l)}$ = error of node j in layer l.

We have three layers in our neural network, so we first start with the last layer (the output layer), and work backwards. We do not calculate $\delta_1^{(1)}$ as this is the input layer. Starting with the last layer, we have:

$\delta_1^{(3)} = y^{(i)} - a_1^{(i)}$

Where $y^{(i)}$ is the actual value, and $a^{(i)}$ is the activation value.

As stated in the question, we suppose that the correct output is 1. Therefore the value of $\delta_1^{(3)}$ is:

$$\begin{aligned}
\delta_1^{(3)} &= 1 - 0.911 \\
&= 0.089
\end{aligned}$$

We then propogate backwards to calculate the errors for the next set of nodes:

$$\begin{aligned}
\delta_1^{(2)} &= \theta_{11}^{(2)} \cdot \delta_1^{(3)} \\
\delta_2^{(2)} &= \theta_{12}^{(2)} \cdot \delta_1^{(3)}
\end{aligned}$$

This is calculated to be:

$$\begin{aligned}\delta_1^{(2)} &= 1 \cdot 0.089 \\ &= 0.089 \\ \delta_2^{(2)} &= 2 \cdot 0.089 \\ &= 0.178\end{aligned}$$

Next we need to calculate the updates of all the weights. To do this we use the cost function.

The cost function for binary classification is:

$$J(\theta) = -\frac{1}{m}[\sum_{(i=1)}^{m}\sum_{(k=1)}^{K} y_k^{(i)} log(h_\theta(x_k^{(i)})log(1 - h_\theta(x^{(i)}))k)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{sl}\sum_{j=1}^{(sl+1}(\theta_{ji}^{(l)})^2$$

This cost function is run over each example, so simplifies to:

$$cost(i) = y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(h_\theta(x^{(i)}))$$

so for the output:

$$\begin{aligned}cost &= 1 \cdot log(0.089) + (1-1) \cdot log(0.089) \\ &= -2.42\end{aligned}$$

2 Question 4: Perceptron

2.1 What are the values of weights w_0, w_1 and w_2?

Please Note: answered with reference to Perceptron for Dummies slides http://www.jilp.org/cbp/Daniel-slides.PDF

We assume that the surface crosses the x_1 axis at -1, and the x_2 axis at 2.

The perceptron outputs, for each x:

$$o(x_1, ..., x_n) = \begin{cases} 1 & if w_0 + w_1x_1 ... + w_nx_n > 0 \\ -1 & otherwise \end{cases}$$

The decision surface of a perceptron is the representation of the perceptron. Our perceptron has two x values, and so is the equation of a line in 3D space.

$$y = w_1x_1 + w_2x_2 + w_0$$

2.2 Designing perceptrons

2.2.1 Design a 2-layer perceptron that implements the Booean function A AND (NOT B)

The Boolean function A AND (NOT B) is the function that inputs A and B and will return a positive value if the sum of A and NOT B are both True. In this case,$x_1 is A$, and $x_2 is B$.

The values of the inputs are boolean, so they are binary, and only take values of 0 or 1.

Table 1: A AND (NOT B)

A	¬	B	out
1	0	1	0
1	1	0	1
0	0	1	0
0	1	0	0

We choose the weights for w_1 and w_2 to reflect A and ¬B:

$$w_0 = -5$$
$$w_1 = 10$$
$$w_2 = -30$$

so our classifier remains:

$$o(x_1, ..., x_n) = \begin{cases} 1 & if w_0 + w_1 x_1 ... + w_n x_n > 0 \\ -1 & otherwise \end{cases}$$

for example:

$$x_1 = 1$$
$$x_2 = 0$$
$$then$$
$$-5 + 1 \cdot 10 + 0 \cdot -30 = 15$$

Therefore correctly is classified as 1, or True.
and if we test the other way: for example:

$$x_1 = 1$$
$$x_2 = 1$$
$$then$$
$$-5 + 1 \cdot 10 + 1 \cdot -30 = -25$$

The value is = 0, so is classified as 0, or False

2.2.2 Design a two-layer network of perceptrons that implements A XOR B

A XOR B is true when one of the values is false:

Table 2: A XOR B

A	B	out
1	1	0
1	0	1
0	1	1
0	0	0

At the first layer we shall check the logical AND:
The truth table for AND is

Table 3: A AND B

A	B	out
1	1	1
1	0	0
0	1	0
0	0	0

the weights for this layer are:

$$\begin{aligned} w_0 &= -10 \\ w_1 &= 10 \\ w_2 &= 10 \end{aligned}$$

so our classifier remains:

$$o(x_1, ..., x_n) = \begin{cases} 1 & if w_0 + w_1 x_1 ... + w_n x_n > 0 \\ -1 & otherwise \end{cases}$$

so, for example:

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 0 \\ &then \\ -10 + 10 \cdot 1 + 10 \cdot 0 &= 0 \end{aligned}$$

Therefore would be classified as 0, False.
The next layer of the perceptron is OR: The truth table for this is:
the weights for this layer are:

$$\begin{aligned} w_0 &= -10 \\ w_1 &= 20 \\ w_2 &= 20 \end{aligned}$$

Table 4: A OR B

A	B	out
1	1	1
1	0	1
0	1	1
0	0	0

so our OR classifier remains:

$$o(x_1, ..., x_n) = \begin{cases} 1 & if w_0 + w_1 x_1 ... + w_n x_n > 0 \\ -1 & otherwise \end{cases}$$

so, for example:

$$x_1 = 1$$
$$x_2 = 0$$
$$then$$
$$-10 + 20 \cdot 1 + 20 \cdot 0 = 10$$

This classified as 1 (10>0) so is True.
As we defined earlier, the truth table for XOR is:

Table 5: A XOR B

A	B	out
1	1	0
1	0	1
0	1	1
0	0	0

Therefore if we put our perceptron together we can see if it outputs the correct value: IN this table, A and B represent the input values. We compute A AND B at the same time as A OR B, the outputs of the two nodes are fed into the final node, giving the output.

Table 6: layered perceptron

A	B	$a_1^{(2)}$(AND)	$a_2^{(2)}$(OR)	out
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0