Machine Learning notes

LWang

January 10, 2017

This notes are mostly based on Andrew Ng's Machine Learning course on Coursera

1 Backpropagation Algorithm

A network forward propgates activation to produce an output and it backward propgates error to determine weight parameters. The goal of Backpropagation Algorithm in neural network is to minimize cost function(the sum squared error between the network's output values and the given target values), $\min_\theta J(\Theta)$, more specifically, we want to compute $\frac{\partial}{\partial \Theta_{j,i}^{(l)}} J(\Theta)$

The algorithm runs as follows:

Given training set $(x^{(1),y(1)}) \dots (x^{(m)}, y^{(m)})$

set $\Delta_{i,j}^{(l)} := 0$ for all (l,i,j), we obtain a matrix full of zeros.

For training example t= 1 to m:

1. Set $a^{(1)} := X$, add x_0
2. Perform forward propgation to compute $a^{(l)}$, for l=2,3...L

$$
\begin{aligned}
a^{(1)} &= X \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \quad (\text{add} \quad a_0^{(2)}) \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(3)} &= g(z^{(3)}) \quad (\text{add} \quad a_0^{(3)}) \\
z^{(4)} &= \Theta^{(3)} a^{(3)} \\
a^{(4)} &= h_\Theta(x) = g(z^{(4)})
\end{aligned}
$$

3.Using $y^{(t)}$, compute error values $\delta^{(L)} = a^{(L)} - y^{(t)}$.

4.Compute error values for each layer, $\delta^{(L-1)}, \delta^{(L-2)} \dots \delta^{(2)}$, where

$$\delta^{(l)} = \big((\Theta^{(l)})^T \delta^{(l+1)}\big) * g'(z^{(l)})$$

The g–prime derivative term can also be written out as :

$$g'(z^{(l)} = a^{(l)} .* (1 - a^{(l)})$$

5.Update term.$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T$

$D_{i,j}^{(l)} = \frac{1}{m}\big(\Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)}\big), \qquad \text{if} \qquad j \neq 0$

$D_{i,j}^{(l)} = \frac{1}{m}\big(\Delta_{i,j}^{(l)}, \qquad \text{if} \qquad j = 0$

where $\frac{\partial}{\partial \Theta_{j,i}^{(l)}} J(\Theta) = D_{(i,j)}^{(l)}$

2 More details about Backpropgation:

Notations:

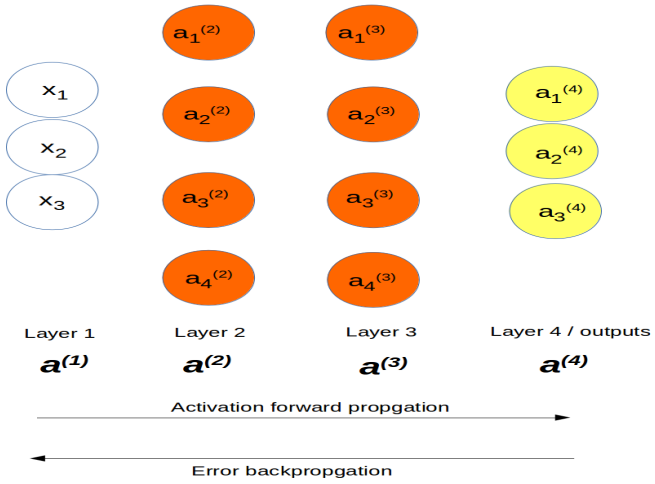

Figure 1: 4 layers neural networks

- L: layers of neural network
- s_l : number of units in the lth layer(no bias term counted)
- $z_i^{(l)}$: the ith input in the lth layer(i=1,2,... s_l)
- $a_i^{(l)}$: the ith output in the lth layer(i=0,1,...,s_l)
- $\Theta_{ij}^{(l)}$: parameter (j=0,1,...,s_l, i= 1,2,...,s_{l+1})
- $h_\Theta(x)$: the output of the neural network
- $J(\Theta)$: the cost function
- $\delta_i^{(l)}$: the computation error in the ith unit in the lth layer(i=1,2,...,s_l)

The goal is to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ for all i, j, l.

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = \underbrace{\frac{\partial J(\Theta)}{\partial a_i^{(l+1)}}}_{\text{part 1}} \cdot \underbrace{\frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}}}_{\text{part 2}} \cdot \underbrace{\frac{\partial z_i^{(l+1)}}{\partial \Theta_{ij}^{(l)}}}_{\text{part 3}} \qquad \text{(using chain rule)} \tag{2.0.1}$$

$$\begin{aligned}
\text{part 1} = \frac{\partial J(\Theta)}{\partial a_i^{(l+1)}} &= \sum_{j=1}^{s_{l+2}} \frac{\partial J}{\partial z_j^{(l+2)}} \frac{\partial z_j^{(l+2)}}{\partial a_i^{(l+1)}} \\
&= \sum_{j=1}^{s_{l+2}} \delta_j^{(l+2)} \frac{\partial}{\partial a_i^{(l+1)}} \Big(\sum_{k=1}^{s_l+1} \Theta_{jk}^{(l+1)} \cdot a_k^{(l+1)} \Big) \\
&= \sum_{j=1}^{s_{l+2}} \delta_j^{(l+2)} \Theta_{ji}^{(l+1)} \qquad (i = 1, 2, \ldots, s_{l+2}, \qquad \text{no bias term})
\end{aligned} \tag{2.0.2}$$

$$\begin{aligned}
\text{part 2} = \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}} &= \frac{\partial}{\partial z_i^{(l+1)}} g(z_i^{(l+1)}) \\
&= g(z_i^{(l+1)})\big(1 - g(z_i^{(l+1)})\big) \\
&= a_i^{(l+1)}\big(1 - a_i^{(l+1)}\big) \qquad (i = 1, 2, \ldots, s_{l+1}, \qquad \text{no bias term})
\end{aligned} \tag{2.0.3}$$

$$\text{part 3} = \frac{\partial z_i^{(l+1)}}{\partial \Theta_{ij}^{(l)}} = \frac{\partial}{\partial \Theta_{ij}^{(l)}} \big(\sum_{k=0}^{s_l} \Theta_{ik} a_k^{(l)} \big)$$
$$= a_j^{(l)} \qquad (j = 0, 1, 2, \ldots, s_{l+1}, \qquad \text{bias term included}) \tag{2.0.4}$$

now, let's combine part 1 , 2 and 3,

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = \sum_{k=1}^{s_{l+2}} \delta_k^{(l+2)} \Theta_{ki}^{(l+1)} a_i^{(l+1)} \big(1 - a_i^{(l+1)}\big) a_j^{(l)} \tag{2.0.5}$$

since $\delta_i^{(l)} = \frac{\partial}{\partial z_i^{(l)}} J(\Theta)$, we will have

$$\delta_i^{(l+1)} = \frac{\partial}{\partial z_i^{(l+1)}} J(\Theta)$$

which also equals to

$$\delta_i^{(l+1)} = \text{part 1} \cdot \text{part 2}$$
$$\delta_i^{(l+1)} = \sum_{k=1}^{s_{l+2}} \delta_k^{(l+2)} \Theta_{ki}^{(l+1)} a_i^{(l+1)} \big(1 - a_i^{(l+1)}\big)$$
$$\delta^{(l+1)} = \big(\Theta^{(l+1)}\big)^T \delta^{(l+2)} a^{(l+1)} \big(1 - a^{(l+1)}\big) \tag{2.0.6}$$

Rewrite eq(6.2.5), the gradient can be neatly expressed as

$$\frac{\partial J(\Theta)}{\partial \Theta^{(l)}} = \delta^{(l+1)} \big(a^l\big)^T \tag{2.0.7}$$