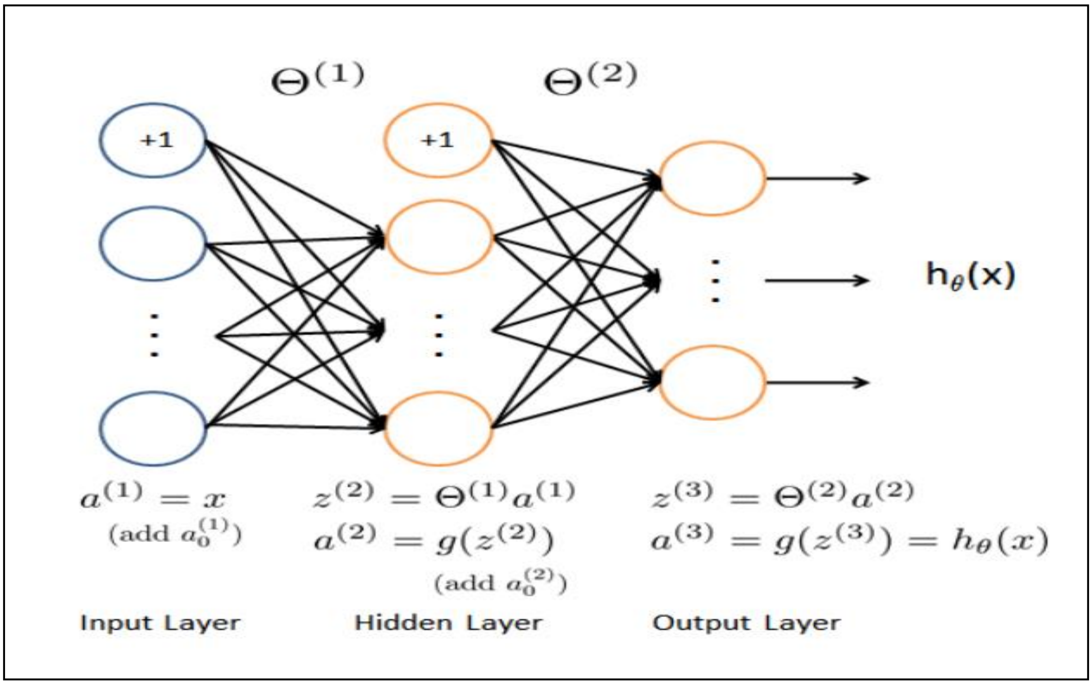
Review and some details of NN & BP algorithm.



Notation :

m = num of training sets

n = num of features

K = num of output classes

L = num of layers

s_l = num of units of the l_{th} layer. (bias not included)

After load data. We have

$$X = \begin{bmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots \\ x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$ --(m*n matrix) ;

$$Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}$$ --(m*1 matrix, and need to be transported to m*k matrix)

Add bias $x_0 = 1$ to X ,then we get $a^{(1)}$ --(m*(n+1)) .

For l_{th} layer to $(l+1)_{th}$ layer, Θ^l is a $s_{l+1} \times (s_l + 1)$ matrix.

eg. $\Theta^{(1)} = \begin{bmatrix} \theta_{10}^{(1)} & \dots & \theta_{1n}^{(1)} \\ \dots & \dots & \dots \\ \theta_{s_2 0}^{(1)} & \dots & \theta_{s_2 n}^{(1)} \end{bmatrix}$ -- ($s_2 \times (s_1 + 1)$) , s_1 =n

By $z^{(2)} = a^{(1)} (\Theta^{(1)})^T$, then we get $z^{(2)}$ -- (m*s2).

$z^{(2)} = \begin{bmatrix} z_1^{(21)} & \dots & z_n^{(21)} \\ \dots & \dots & \dots \\ z_1^{(2m)} & \dots & z_n^{(2m)} \end{bmatrix}$

Take $a^{(2)} = g(z^{(2)})$ (g refers to the sigmoid function) and add bias $a_0^{(2)} = 1$, then we get $a^{(2)}$ --(m*(s2+1)).

As the same, we can get $a^{(3)}$ --(m*k) as well.

For a NN which has only one hidden layer, the third layer is the output layer .so $a^{(3)}$ is the hypothesis $h_\Theta(x)$ --(m*k)

BP:

To take the gradient descent , we need to compute cost function J and its partial derivative .

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)})_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

Im the MATLAB program, we can use

$$\boxed{\begin{aligned} J &= sum(sum(-\mathrm{Y}.*\log(a^{(3)})-(1.-\mathrm{Y}).*\log(1.-a^{(3)})))/m \\ &\quad +(sum(sum(\Theta^{(1)}.\wedge 2))+sum(sum(\Theta^{(2)}.\wedge 2)))*\lambda/(2*m) \end{aligned}}$$

Computing the gradient:

First ,we are going to compute the error of every layer $\delta^{(l)}$.

For the output layer (here is the 3th layer):

$\delta^{(3)} = a^{(3)} - \mathrm{Y}$ (m*k)

For hidden layer like the 2th layer:

$$\delta^{(2)} = \delta^{(3)}\Theta^{(2)}.*g'\left(z^{(2)}\right) \rightarrow \delta^{(2)} = \delta^{(3)}\Theta^{(2)}.*(a^{(2)}.*(1-a^{(2)}))$$

(m * (s2+1)) = (m * k) * (k * (s2+1)) .* (m * (s2+1))

Actually, $\delta^{(2)}$ is supposed to be a (m * s2) matrix, so we need to delete the

first column of $\delta^{(2)}$.

($\delta^{(l)} = \delta^{(l+1)}\Theta^{(l)}.*(a^{(l)}.*(1-a^{(l)}))$ --(m * (s_l+1)) ,

Before computing error of the next layer,we need to delete the first column

of $\delta^{(l)}$ and get a new $\delta^{(l)}$ which is supposed to be a (m * s_l) matrix.)

As for the input layer, there is no error.

After computing errors of every layers except the input layer, wo are going

to computing the gradient. $\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

$$\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$$

$$D_{ij}^{(l)} := \frac{1}{m}\triangle_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m}\triangle_{ij}^{(l)} \quad \text{if } j = 0$$

$$\Theta_{grad}^{(1)} = ((\delta^{(2)})^T * a^{(1)}) / m + \lambda * \Theta^{(1)} / m$$

$$\Theta_{grad}^{(1)}(:,1) = \Theta_{grad}^{(1)}(:,1) - \lambda * \Theta^{(1)}(:,1) / m$$

As the same, we can get $\Theta_{grad}^{(2)}$, … , $\Theta_{grad}^{(L-1)}$.

$$\Theta_{grad} = [\Theta_{grad}^{(1)}; \Theta_{grad}^{(2)}; \ldots; \Theta_{grad}^{(L-1)}]$$