Knowledges:

BP
反向传播法 ⇒ 传统神经网络训练方法

：随机设定初值，计算当前网络output，与标签残差去改变前面各层参数，直至收敛。

本质即 为梯度下降法求得 $J(\theta)$ 的偏导数，确定下降的方向。

梯度扩散：在深层网络结构中，残差传递至最前面层已变得太小，即梯度扩散了。

Neural Network:

I. Cost function:

来源 ⇒ Logistic Regression:

$$J(\theta) = -\frac{1}{m} \cdot \sum_{i=1}^{m} \left[y_i \cdot \log h_\theta(x_i) + (1-y_i) \cdot \log 1 - h_\theta(x_i) \right]$$

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 .$$

Neural Network:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\cdot\left[y_k^{(i)}\cdot\log h_\theta(x^{(i)})_k + (1-y_k^{(i)})\cdot\log(1-h_\theta(x^{(i)}))_k\right]$$

$$+\frac{\lambda}{2m}\cdot\sum_{l=1}^{L-1}\cdot\sum_{j=1}^{S_l}\sum_{i=1}^{S_{(l+1)}}\left(\theta_{ij}^{(l)}\right)^2$$

II. Forward Propagation: (FP)

$a^{(1)} = x$ (No need to add bias unit)

$z^{(2)} = \theta^{(1)}\cdot a^{(1)}$

$a^{(2)} = g(z^{(2)})$ ($a^{(2)}$ add $a_0^{(2)}$)

$z^{(3)} = \theta^{(2)}\cdot a^{(2)}$

$a^{(3)} = g(z^{(3)})$ ($a^{(3)}$ add $a_0^{(3)}$)

$\vdots$

$z^{(k+1)} = \theta^{(k)}\cdot a^{(k)}$

$a^{(k+1)} = g(z^{(k+1)})$ ($a^{(k+1)}$ add $a_0^{(k+1)}$)

$\vdots$

How to get the expression of $\delta^{(l)} \Rightarrow$

① $\delta^l = (\theta^l)^T \cdot \frac{\partial E^{(l+1)}}{\partial x^{(l+1)}}$ → $x^{(l+1)} = \theta^{(l)} \cdot a^{(l)} + b$

$E = \frac{1}{2} \sum_{i=1}^{m} (y_{output} - y_{real})^2$

反向传播

现在我们获得了 $\frac{dE}{dy}$，接下来便可以根据链式法则得出 $\frac{dE}{dx}$。

$= \frac{\partial E}{\partial y} = y_{output} - y_{real}$

$$\frac{\partial E}{\partial x} = \frac{dy}{dx}\frac{\partial E}{\partial y} = \frac{d}{dx}f(x)\frac{\partial E}{\partial y}$$

z　a　g

其中，当 $f(x)$ 是 S 型激活函数 $\Rightarrow f(x) = \frac{1}{1+e^{-z}}$

时，$\frac{d}{dx}f(x) = f(x)(1 - f(x))$ 。

$f'(x) = f(x)(1-f(x))$

$= a^l \cdot (1 - a^l)$

$\delta^l = (\theta^l)^T \cdot \frac{\partial E^{(l+1)}}{\partial x^{(l+1)}}$

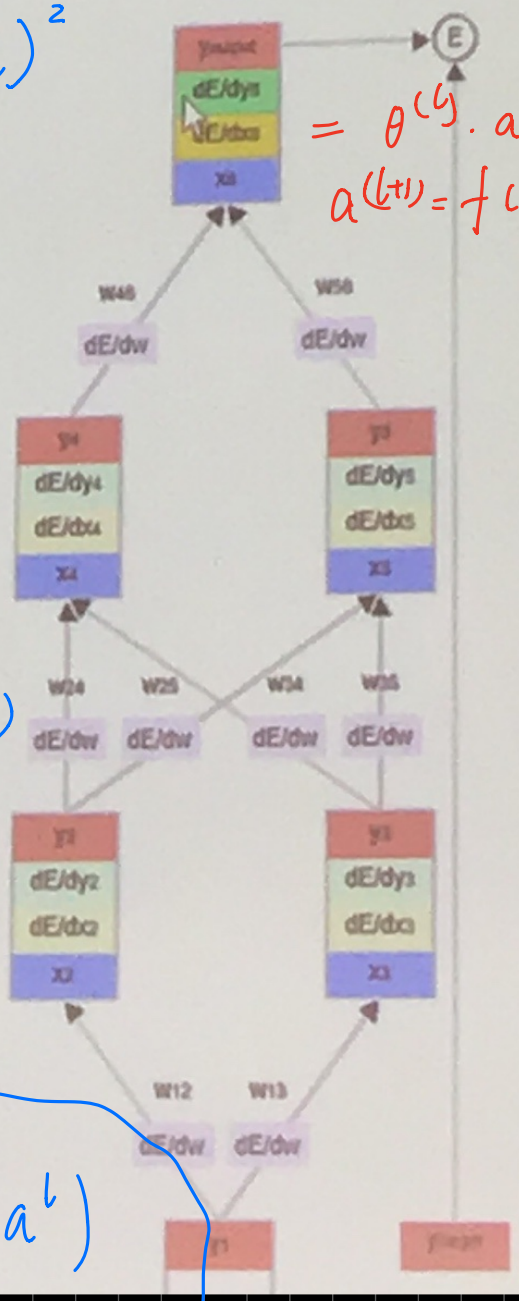
$\frac{\partial E^{(l+1)}}{\partial y^{(l+1)}} \cdot a^l \cdot (1 - a^l)$

$\Downarrow$

$\delta^{(l+1)}$

$= \theta^{(l)} \cdot a^{(l)}$

$a^{(l+1)} = f(x^{(l+1)})$



III. Back Propagation (BP):

I. Calculate $\delta_j^{(l)}$ for each $j \in (1, S_l)$

$l \in (2, L)$

$\delta_j^{(l)}$ → error for node j in layer l.

the first layer is input layer, there is no error for every node j.

How to calculate ⟹ right to the left (or back)

error of layer L ⟸ $\delta^{(L)} = a^{(L)} - y$

error of layer $(L-1)$, calculate using $\delta^{(L)}$ ⟸ $\delta^{(L-1)} = (\theta^{(L-1)})^T \cdot \delta^{(L)} .* g'(z^{(L-1)})$

$\frac{\partial E}{\partial a}$

$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial (z)} = g'(z)$

$= (\theta^{(L-1)})^T \cdot \delta_j^{(L)} \cdot a^{(L-1)} .* (1 - a^{(L-1)})$

$g(z) = \frac{1}{1+e^{-z}}$

$g'(z) = \left(\frac{1}{1+e^{-z}}\right)\left(1 - \frac{1}{1+e^{-z}}\right)$

$= g(z) \cdot (1 - g(z))$

II. Calculate $\Delta_{ij}^{(l)}$

How ⟹ ① Set $\Delta_{ij}^{(l)} = 0$ for all l, i, j

② $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \cdot \delta_i^{(l+1)}$

$$\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + \delta_i^{(l+1)} \cdot a_j^{(l)}$$

III Calculate $D_{ij}^{(l)} \Leftrightarrow \frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}}$, 即 cost function 对第 l 层的 i 行 j 列参数的偏导数

$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}} = D_{ij}^{(l)} = \begin{cases} \frac{1}{m} \cdot \Delta_{ij}^{(l)} + \lambda \cdot \theta_{ij}^{(l)}, & j \neq 0 \Rightarrow \text{other unit} \\ \frac{1}{m} \cdot \Delta_{ij}^{(l)}, & \text{if } j = 0 \Rightarrow \text{bias unit} \end{cases}$$

IV. Gradient checking $\Rightarrow$ numerical approximation of $\frac{\partial J(\theta)}{\partial \theta}$

$$\frac{\partial J(\theta)}{\partial \theta_i} \approx \text{gradApprox} = \frac{J(\theta_1, \theta_2 \cdots \theta_i + \varepsilon \cdots \theta_n) - J(\theta_1, \theta_2 \cdots \theta_i - \varepsilon \cdots \theta_n)}{2\varepsilon}$$

△ ε is usually 10^{-3}, 10^{-4} or even smaller.

△ Note that the most important thing about

gradient checking is that to calculate each $\frac{\partial J(\theta)}{\partial \theta_i}$, we remain other θ_j $(j \neq i)$ unchanged

△ After checking, turn off the gradient checking and just use backpropagation code for learning.

because gradient checking is very slow!

V. Random initialization

Reason: -We can't use the all zero initialization to init $\vec{\theta}$ as we've done in Logistic Regression - because this will cause symmetric problems.

Application:

OCR: Optical Character Recognition.

Pipeline process: