Neural Network 2

Training 중에 일어나는 일

- Given $(\mathbf{x}^i, \mathbf{y}^i)$ $(\mathbf{i=1...m})$
- Define Hypothesis $\mathbf{H_\theta(x)}$ for predicting $\mathbf{y}^j$ from new $\mathbf{x}^j$

- Choose cost function $\mathbf{J(\theta)}$ $(\boldsymbol{\theta}_i\ \mathbf{i=1...n})$ such that
- By minimizing $\mathbf{J(\theta)}$ for fixed $(\mathbf{x}^i, \mathbf{y}^i)$ $(\mathbf{i=1...m})$
- We obtain $\boldsymbol{\theta}$ for best $\mathbf{H_\theta(x)}$

Linear regression (univariate 경우)

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

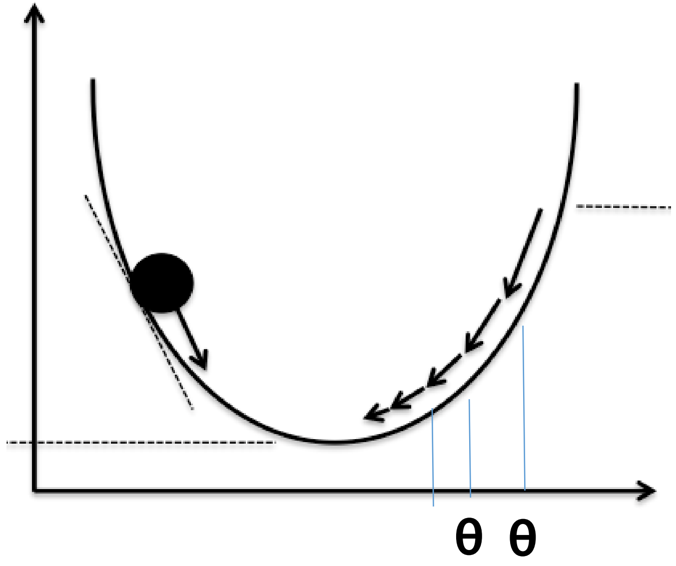
Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$

최적화 알고리즘 (gradient descent)

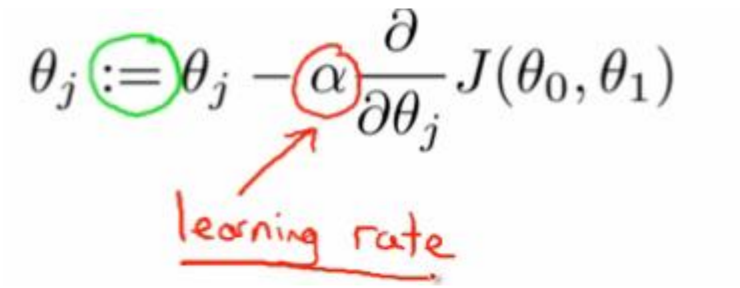
- Minimize **J(θ)** for fixed **(x^i, y^i) (i=1...m)** 하려면
- 어떤 **θ** 에 대해서도 **J(θ)** 과 $\frac{dJ(\theta)}{d\theta}$ 를 알 수 있는 방법이 필요



(1) 주어진 **θ** 에 대해 **J(θ)**, $\frac{dJ(\theta)}{d\theta}$ 계산

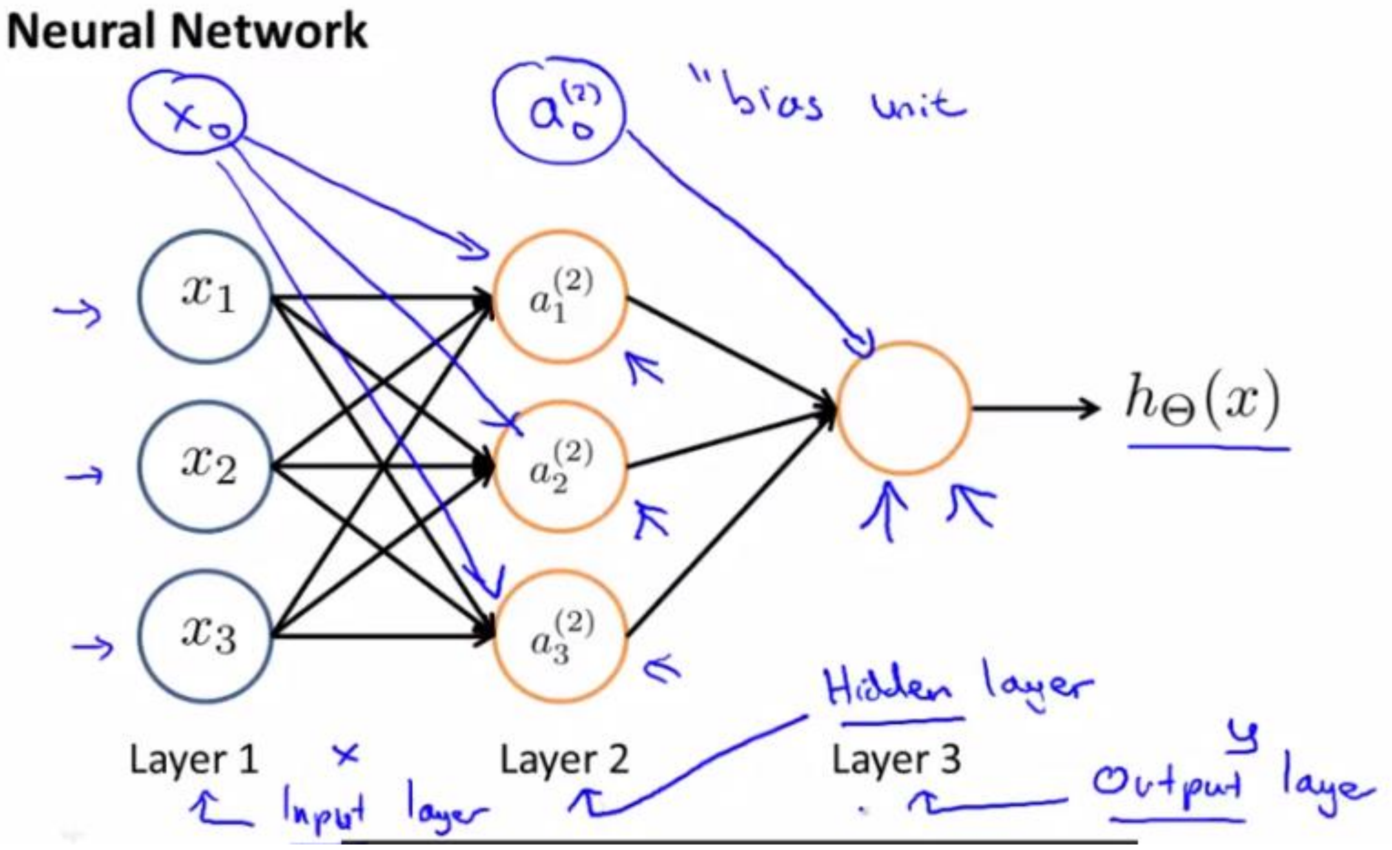
(2) 새로운 **θ** 계산 (아래 공식 참조)

(3) 새로운 J(**θ**) 계산하고 더 나으면 (1)로 돌아가 반복



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Model



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$\Theta^{(2)}$

Neural network

- 레이어간 weight matrix들이 바로 **θ**
- 주어진 모든 $\boldsymbol{\theta^{l}_{jk}}$ 에 대해서 **J(θ)** 과 $\frac{\partial J(\theta)}{\partial \theta^{l}_{jk}}$ 값을.. 알 수 있나?

- **J(θ) :** (x^i, y^i) (i=1...m) 에 대해서 y^i 와 NN의 예측의 차이

➔ **forward propagation for ALL, fixed (x^i, y^i) (i=1...m)**

- $\frac{\partial J(\theta)}{\partial \theta^{l}_{jk}}$: NN 의 결과 예측값에 미치는 θ^{l}_{jk}의 영향의 크기

➔ **backpropagation for ALL, fixed (x^i, y^i) (i=1...m)**

NN training 알고리즘

1. Randomly initialize the weights
2. Implement forward propagation to get $h_\Theta(x^{(i)})$ for any $x^{(i)}$
3. Implement the cost function

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

4. Implement backpropagation to compute partial derivatives
5. Use gradient checking to confirm that your backpropagation works. Then disable gradient checking.

---

6. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta.

When we perform forward and back propagation, we loop on every training example:

```
for i = 1:m,
   Perform forward propagation and backpropagation using example (x(i),y(i))
   (Get activations a(l) and delta terms d(l) for l = 2,...,L
```

$$\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) \;.*\; a^{(l)} \;.*\; (1 - a^{(l)})$$

Backpropagation algorithm

→ Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j). (used to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

For $i = 1$ to m ← $(x^{(i)}, y^{(i)})$.

- Set $a^{(1)} = x^{(i)}$
- → Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
- → Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
- → Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ ~~$\delta^{(1)}$~~
- → $\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ ← $[ij]$

$\triangle^{(l)} := \triangle^{(l)} + \delta^{(l+1)} (a^{(l)})^T$.

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

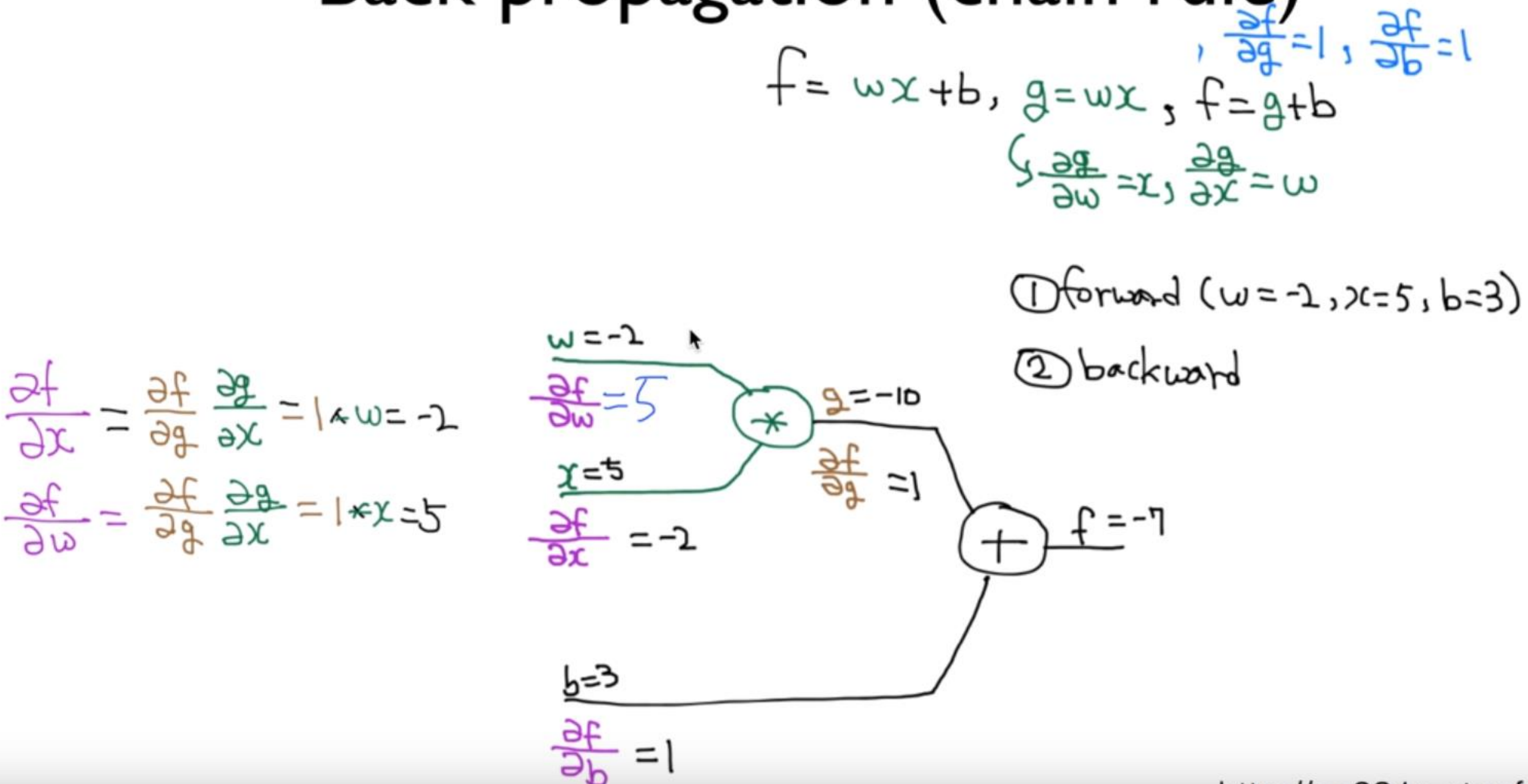
→ $D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)}$ if $j = 0$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

Intuition

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

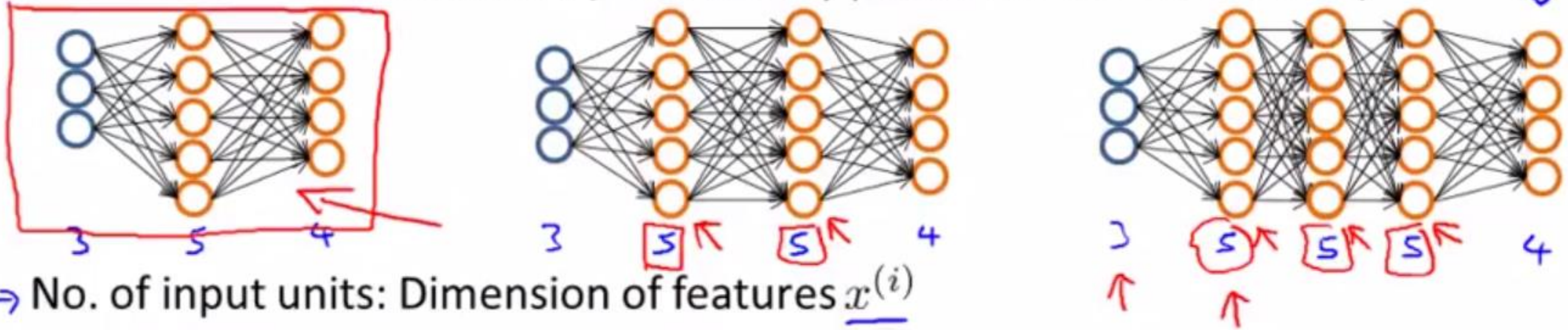
Back propagation (chain rule)



http://cs231n.stanfor

Others – choosing network shape

Pick a network architecture (connectivity pattern between neurons)



- No. of input units: Dimension of features $x^{(i)}$
- No. output units: Number of classes

Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)

Others

- ReLU() is better than sigmoid
- Why randomize the initialize $\boldsymbol{\theta}_{jk}^{l}$?
- Dropout option
- https://www.coursera.org/learn/machine-learning/lecture/zYS8T/autonomous-driving

실습

- http://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
- http://machinelearningmastery.com/setup-python-environment-machine-learning-deep-learning-anaconda/