Lecture 18: Neural Network and Backpropagation II

[SCS4049-02] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

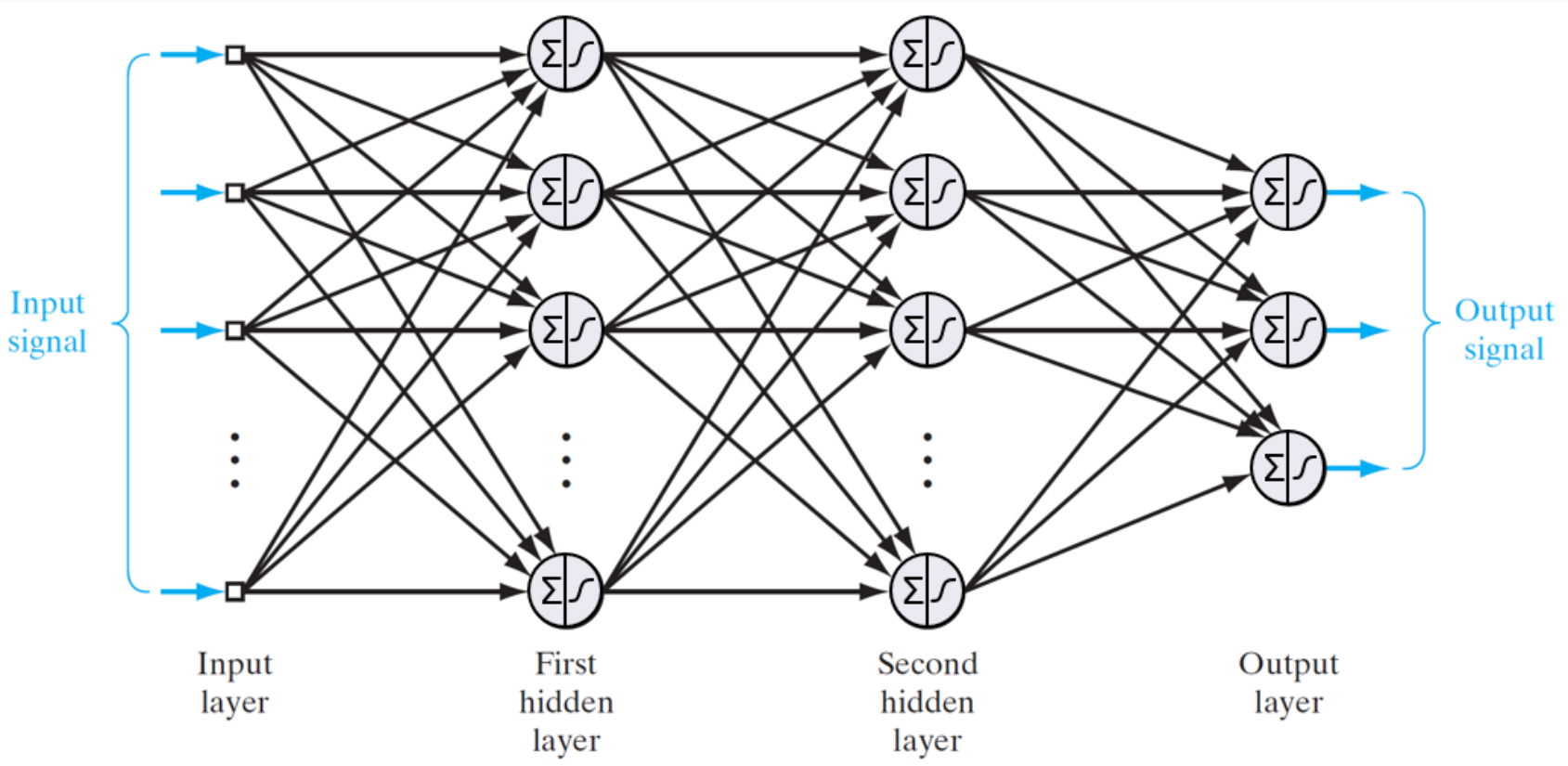
AI Department, Dongguk University

Tentative schedule

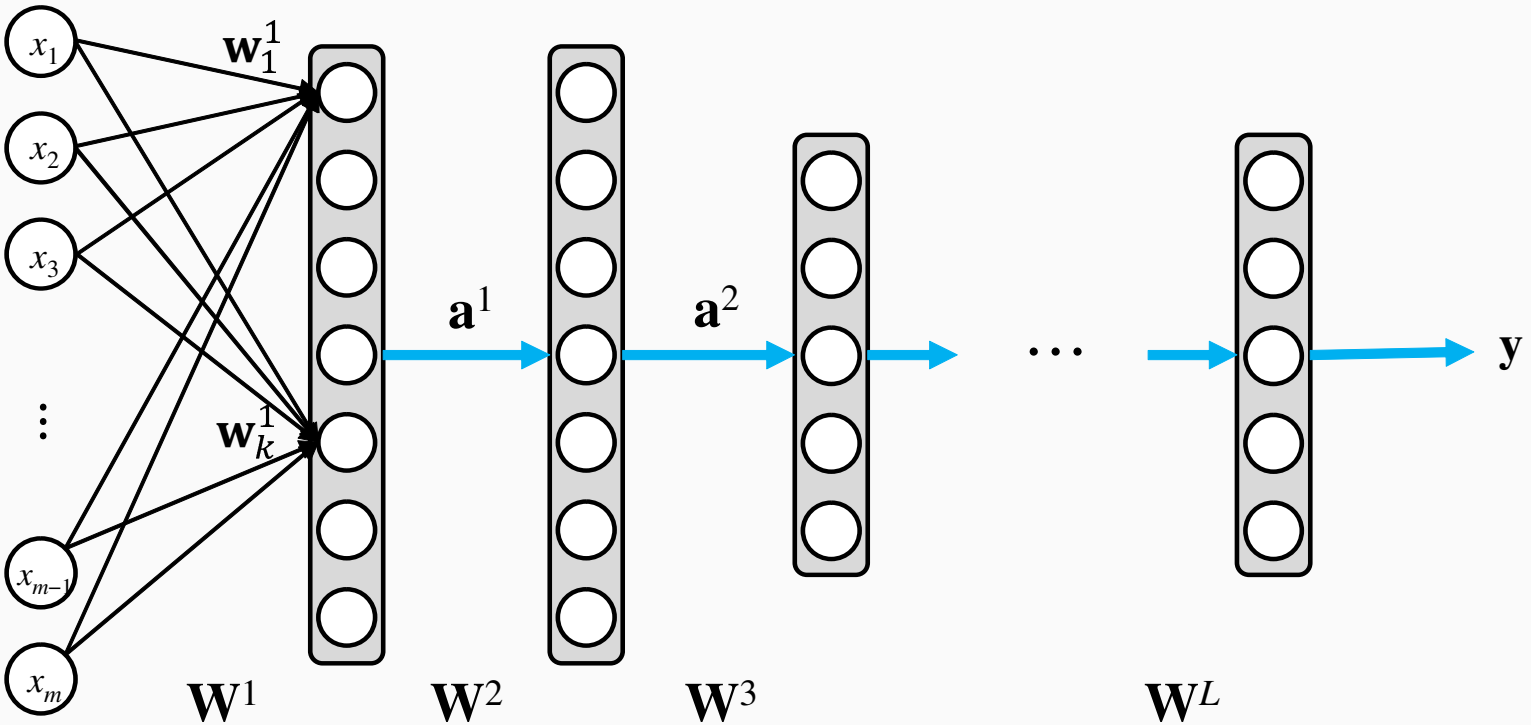
week	topic	date (수 / 월)
1	Machine Learning Introduction & Basic Mathematics	09.02 / 09.07
2	Python Practice I & Regression	09.09 / 09.14
3	AI Department Seminar I & Clustering I	09.16 / 09.21
4	Clustering II & Classification I	09.23 / 09.28
5	Classification II	(추석) / 10.05
6	Python Practice II & Support Vector Machine I	10.07 / 10.12
7	Support Vector Machine II & Decision Tree and Ensemble Learning	10.14 / 10.19
8	Mid-Term Practice & **Mid-Term Exam**	10.21 / **10.26**
9	휴강 & Dimensional Reduction I	10.28 / 11.02
10	Dimensional Reduction II & Neural networks and Back Propagation I	11.04 / 11.09
11	Neural networks and Back Propagation II & Convolutional Neural Network	11.11 / 11.16
12	Model Optimization	11.18 / 11.23
13	Recurrent Neural network	11.25 / 11.30
14	Autoencoders	12.02 / 12.07
15	**Final exam**	(휴강) / **12.14**

Feed-Forward Propagation and Cost Function

Deep feedforward networks



Goal of deep feedforward networks



Goal of deep feedforward networks

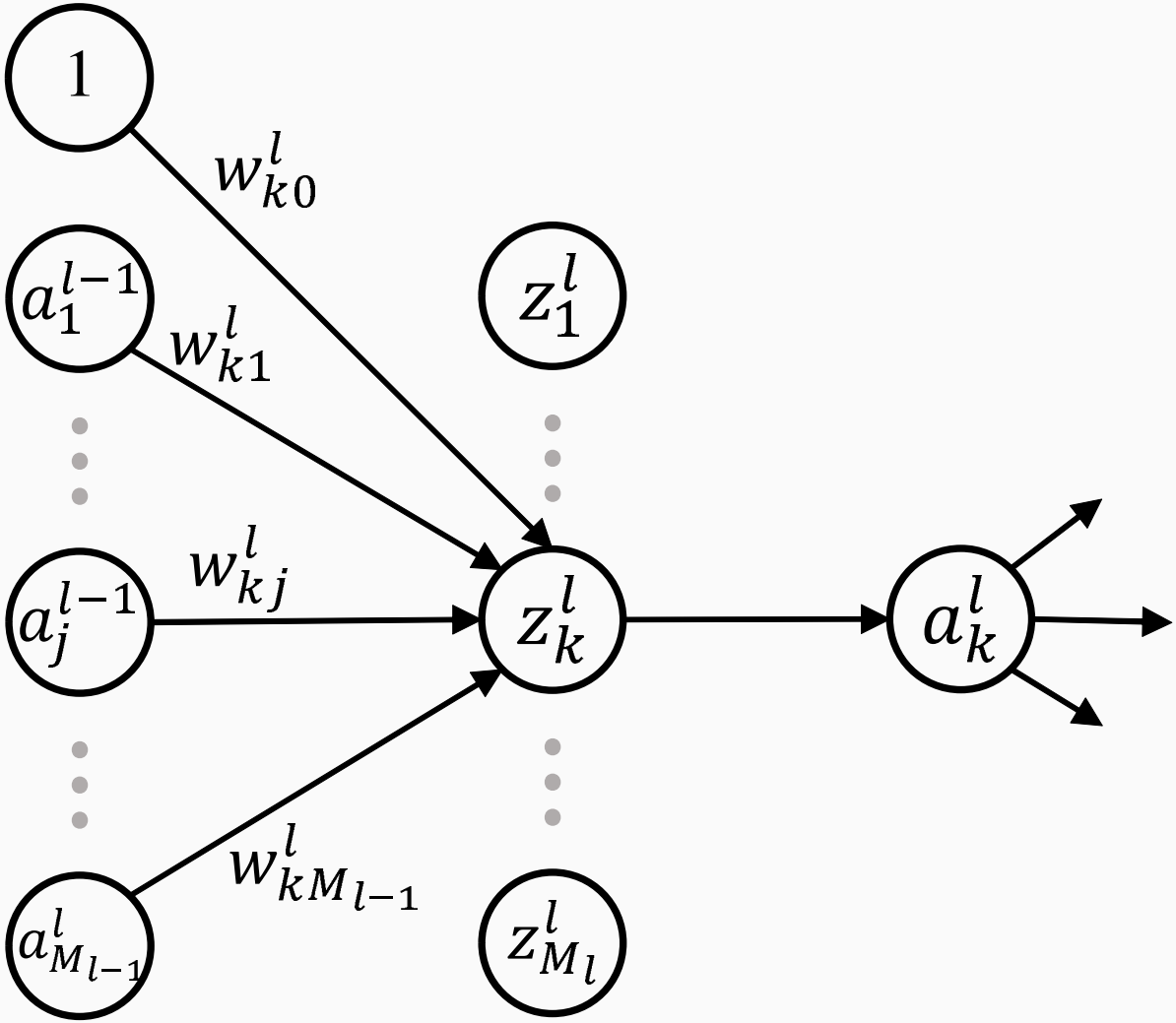
Let $\mathbf{W}^l = \begin{bmatrix} \mathbf{w}_1^l & \mathbf{w}_2^l & \cdots & \mathbf{w}_{m_l}^l \end{bmatrix}$ and $\boldsymbol{\varphi}^l$ be the activation function for layer l, then

$$\mathbf{y}^1 = \boldsymbol{\varphi}^1(\mathbf{W}^1\mathbf{x}) \tag{1}$$
$$\mathbf{y}^2 = \boldsymbol{\varphi}^2(\mathbf{W}^2\mathbf{y}^1) \tag{2}$$
$$= \boldsymbol{\varphi}^2(\mathbf{W}^2\boldsymbol{\varphi}^1(\mathbf{W}^1\mathbf{x})) \tag{3}$$
$$\mathbf{y}^3 = \boldsymbol{\varphi}^3(\mathbf{W}^3\mathbf{y}^2) \tag{4}$$
$$= \boldsymbol{\varphi}^3(\mathbf{W}^3\boldsymbol{\varphi}^2(\mathbf{W}^2\mathbf{y}^1)) \tag{5}$$
$$= \boldsymbol{\varphi}^3(\mathbf{W}^3\boldsymbol{\varphi}^2(\mathbf{W}^2\boldsymbol{\varphi}^1(\mathbf{W}^1\mathbf{x}))) \tag{6}$$
$$\cdots$$
$$\mathbf{y} = \boldsymbol{\varphi}^L(\mathbf{W}^L\boldsymbol{\varphi}^{L-1}(\mathbf{W}^{L-1}\boldsymbol{\varphi}^{L-2}(\cdots\boldsymbol{\varphi}^2(\mathbf{W}^2\boldsymbol{\varphi}^1(\mathbf{W}^1\mathbf{x}))\cdots))) \tag{7}$$

Very much complex nonlinear function

$\Longrightarrow$ require universal function approximator

Feedforward propagation



Feedforward propagation

For l-th hidden layer

$$z_k^l = \sum_{j=0}^{M_{l-1}} w_{kj}^l a_j^{l-1} \qquad a_k^l = \sigma^l(z_k^l) \tag{8}$$

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1} \qquad \mathbf{a}^l = \boldsymbol{\sigma}^l(\mathbf{z}^l) \tag{9}$$

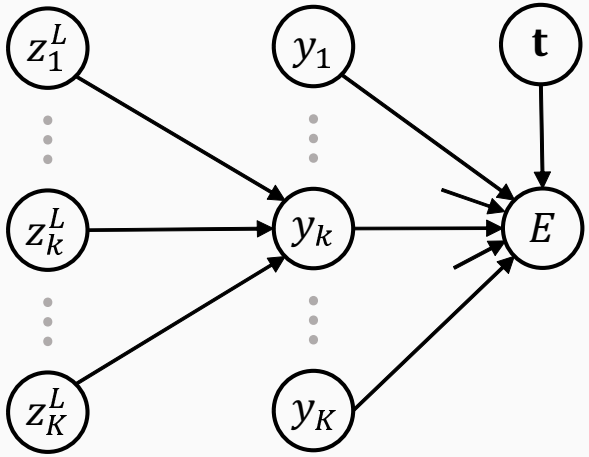
- z_k^l: layer l의 unit k로 들어오는 activation들의 weighted sum
- w_{kj}^l: layer $(l-1)$의 unit j에서 layer l의 unit k로 가는 가중치
- w_{k0}^l: layer l의 unit k에 대한 bias
- a_j^{l-1}: layer $(l-1)$의 unit j에서 나오는 activation 값
- a_0^l: layer l의 bias에 대응 $(=1)$
- a_k^l: layer l의 unit k의 activation 값
- σ^l: layer l의 activation function
- M_{l-1}: layer $(l-1)$의 hidden unit 수

Cost function

Cost function: cross-entropy

$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{K} t_j \log y_j \tag{10}$$

$$y_k = \frac{\exp(z_k)}{\sum_{j=1}^{K}\exp(z_j)} \tag{11}$$



- t_j: $\mathbf{x}$의 class label (target value)
- K: 출력층의 unit 수 = class의 수, $M_L = K$

Stochastic gradient and backpropagation

SGD weight update rule

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \tag{12}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}} E \tag{13}$$

Credit assignment

- Weight 값들이 잘못 되었기 때문에 Cost 가 발생
  $\Longrightarrow$ 따라서 모든 weight에게 그 weight가 cost 발생에 기여한 만큼씩 gradient 를 배분해 주는 것
- 문제는 weight 들이 여러 층에 분산되어 있으며, 각 층은 비선형 함수이므로 배분 방법이 단순하지 않음 $\Longrightarrow$ backpropagation algorithm

Backpropagation

Backpropagation

출력층부터 입력층까지 역순으로 모든 layer에서 weight w_{ij}^l을 업데이트

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \tag{14}$$

Local gradient (or error of neuron)의 정의

$$\delta_j^l \triangleq \frac{\partial E}{\partial z_j^l} \tag{15}$$

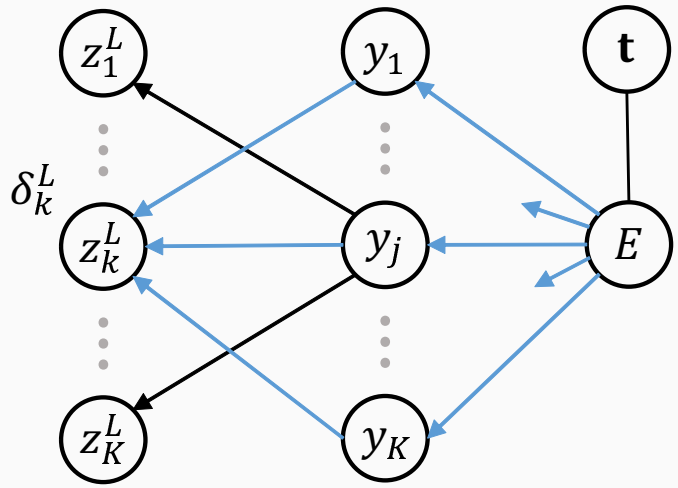
- Layer l의 unit k에서 error, 또는 하위층으로 분배할 gradient
- Activation a_j^l를 기준으로 생각할 수 있으나, nonlinear 함수를 포함하므로 다루기 어려움

Backpropagation: output layer 1/7

출력층에서의 local gradient δ_k^L 계산

$$\delta_k^L = \frac{\partial E}{\partial z_k^L} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_k^L} \tag{16}$$

- Softmax 함수는 $z_1^L, z_2^L, ..., z_K^L$에 의존
- 따라서 z_K^L의 변동은 $y_1, y_2, ..., y_K$에 모두 영향



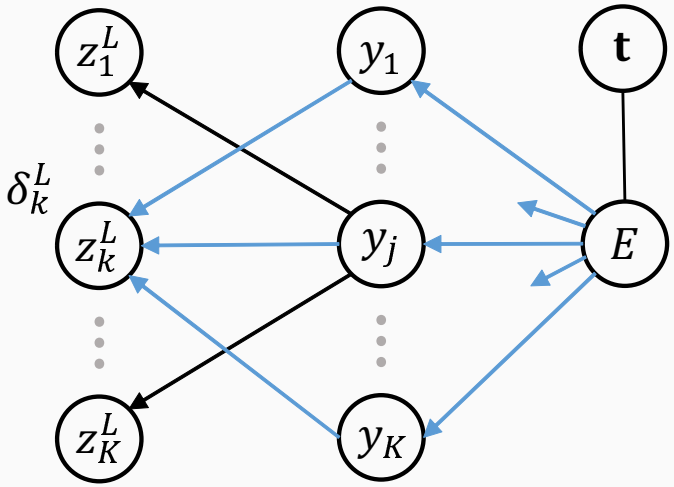
Backpropagation: output layer 2/7

$\frac{\partial y_j}{\partial z_k^L}$의 계산: $k = j$인 경우

$$\frac{\partial y_k}{\partial z_k^L} = \frac{\partial}{\partial z_k^L} \frac{\exp(z_k)}{\sum_i \exp(z_i)} \tag{17}$$

$$= \frac{\exp(z_k)}{\sum_i \exp(z_i)} \left(1 - \frac{\exp(z_k)}{\sum_i \exp(z_i)}\right) \tag{18}$$

$$= y_k(1 - y_k) \tag{19}$$



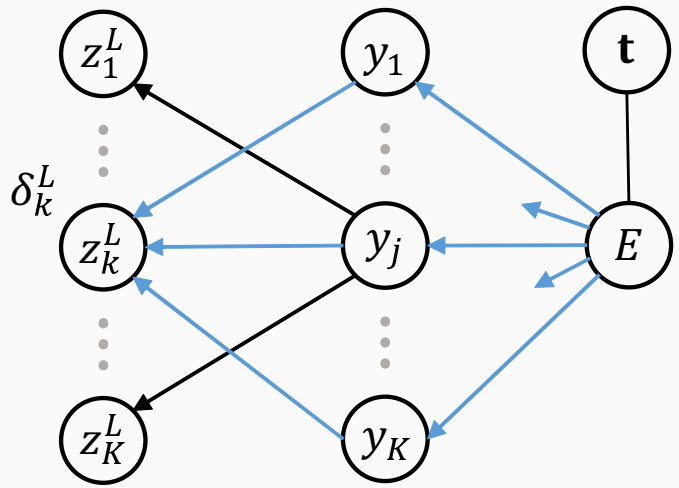
Backpropagation: output layer 3/7

$\frac{\partial y_j}{\partial z_k^L}$ 의 계산: $k \neq j$인 경우

$$\frac{\partial y_j}{\partial z_k^L} = \frac{\partial}{\partial z_k^L} \frac{\exp(z_j)}{\sum_i \exp(z_i)} \tag{20}$$

$$= -\frac{\exp(z_k)}{\sum_i \exp(z_i)} \frac{\exp(z_j)}{\sum_i \exp(z_i)} \tag{21}$$

$$= -y_k y_j \tag{22}$$



Backpropagation: output layer 4/7

From definition of local gradient

$$\delta_j^l = \frac{\partial E}{\partial z_j^l} = -\sum_j t_j \frac{\partial}{\partial z_k} \log y_j = -\sum_j t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_k} \tag{23}$$

$$= -\frac{t_k}{y_k} \frac{\partial y_j}{\partial z_k} - \sum_{j \neq k} \frac{t_j}{y_j} \frac{\partial y_j}{\partial z_k} \tag{24}$$

$$= -\frac{t_k}{y_k} y_k (1 - y_k) - \sum_{j \neq k} \frac{t_j}{y_j} (-y_k y_j) \tag{25}$$

$$= -t_k + t_k y_k + \sum_{j \neq k} t_j y_k \tag{26}$$

$$= -t_k + \sum_{j=1}^{K} t_j y_k \tag{27}$$

$$= -(t_k - y_k) \tag{28}$$

Backpropagation: output layer 5/7

In matrix form

$$\boldsymbol{\delta}^L = \frac{\partial E}{\partial \mathbf{z}^L} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}^L}\right)^T \left(\frac{\partial E}{\partial \mathbf{y}}\right) \tag{29}$$

$$= \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_j}{\partial z_1} & \cdots & \frac{\partial y_K}{\partial z_1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial z_k} & \cdots & \frac{\partial y_j}{\partial z_k} & \cdots & \frac{\partial y_K}{\partial z_k} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial z_K} & \cdots & \frac{\partial y_j}{\partial z_K} & \cdots & \frac{\partial y_K}{\partial z_K} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \vdots \\ \frac{\partial E}{\partial y_j} \\ \vdots \\ \frac{\partial E}{\partial y_K} \end{bmatrix} \tag{30}$$

$$= \begin{bmatrix} \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_1} \\ \vdots \\ \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_k} \\ \vdots \\ \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_K} \end{bmatrix} = \begin{bmatrix} -(t_1 - y_1) \\ \vdots \\ -(t_k - y_k) \\ \vdots \\ -(t_K - y_K) \end{bmatrix} \tag{31}$$

Backpropagation: output layer 6/7

With vector form

$$\mathbf{y} = \boldsymbol{\varphi}(\mathbf{z}^L) \tag{32}$$

$$\boldsymbol{\delta}^L = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}^L}\right)^T \left(\frac{\partial E}{\partial \mathbf{y}}\right) \tag{33}$$

$$= \left(\boldsymbol{\varphi}'(\mathbf{z}^L)\right)^T \nabla_{\mathbf{y}} E \tag{34}$$

$$= -(\mathbf{t} - \mathbf{y}) \tag{35}$$

Backpropagation: output layer 7/7

$\frac{\partial E}{\partial w_{kj}^L}$ 의 계산

$$z_k^L = \sum_{j=0}^{M_{L-1}} w_{kj}^L a_j^{L-1} \tag{36}$$

$$\frac{\partial z_k^L}{\partial w_{kj}^L} = a_j^{L-1} \tag{37}$$

$$\frac{\partial E}{\partial w_{kj}^L} = \frac{\partial E}{\partial z_k^L}\frac{\partial z_k^L}{\partial w_{kj}^L} = -(t_k - y_k)a_j^{L-1} = \delta_k^L a_j^{L-1} \tag{38}$$

그러므로 weight update rule은

$$w_{kj}^L \leftarrow w_{kj}^L - \eta \frac{\partial E}{\partial w_{kj}^L} \tag{39}$$

$$= w_{kj}^L - \eta \delta_k^L a_j^{L-1} \tag{40}$$

$$= w_{kj}^L + \eta(t_k - y_k)a_j^{L-1} \tag{41}$$

$$\tag{42}$$

Backpropagation: local gradient propagation

From δ_j^{l+1} to δ_j^l

$$\delta_k^l = \frac{\partial E}{\partial z_k^l} = \sum_j \frac{\partial E}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_k^l} = \sum_j \frac{\partial z_j^{l+1}}{\partial z_k^l} \delta_j^{l+1} \tag{43}$$

$$z_j^{l+1} = \sum_{i=1} w_{ji}^{l+1} a_i^l = \sum_{i=0} w_{ji}^{l+1} \sigma^l(z_i^l) \tag{44}$$

$$\frac{\partial z_j^{l+1}}{\partial z_k^l} = w_{jk}^{l+1} \sigma^{l\prime}(z_k^l) \tag{45}$$

그러므로

$$\delta_k^l = \sigma^{l\prime}(z_k^l) \sum_j w_{jk}^{l+1} \delta_{jk}^{l+1} \tag{46}$$

Backpropagation: weight in layer l

$\frac{\partial E}{\partial w_{kj}^l}$ 의 계산

$$z_k^l = \sum_{j=0}^{M_{l-1}} w_{kj}^l a_j^{l-1} \tag{47}$$

$$\frac{\partial z_k^l}{\partial w_{kj}^l} = a_j^{l-1} \tag{48}$$

$$\frac{\partial E}{\partial w_{kj}^l} = \frac{\partial E}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l} = \delta_k^l a_j^{l-1} \tag{49}$$

그러므로 weight update rule은

$$w_{kj}^l \leftarrow w_{kj}^l - \eta \frac{\partial E}{\partial w_{kj}^l} \tag{50}$$

$$= w_{kj}^l - \eta \delta_k^l a_j^{l-1} \tag{51}$$

Backpropagation: summary

1. **Input $\mathbf{x}$**: 입력층에서 activation $\mathbf{a}^1$을 계산
2. **Feedforward**: 각 층 $l = 2, 3, ..., L$에 대해서 다음을 계산

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1} \quad \mathbf{a}^l = \boldsymbol{\sigma}^l(\mathbf{z}^l) \tag{52}$$

3. **Output error $\boldsymbol{\delta}^L$**: 출력층에서 비용함수를 이용해 local gradient 계산

$$\boldsymbol{\delta}^L = -(\mathbf{t} - \mathbf{y}) \tag{53}$$

4. **Backpropagate the error**: 출력층부터 거꾸로 다음 식을 계산

$$\delta_k^l = \sigma^{l\prime}(z_k^l) \sum_j w_{jk}^{l+1} \delta_{jk}^{l+1} \tag{54}$$

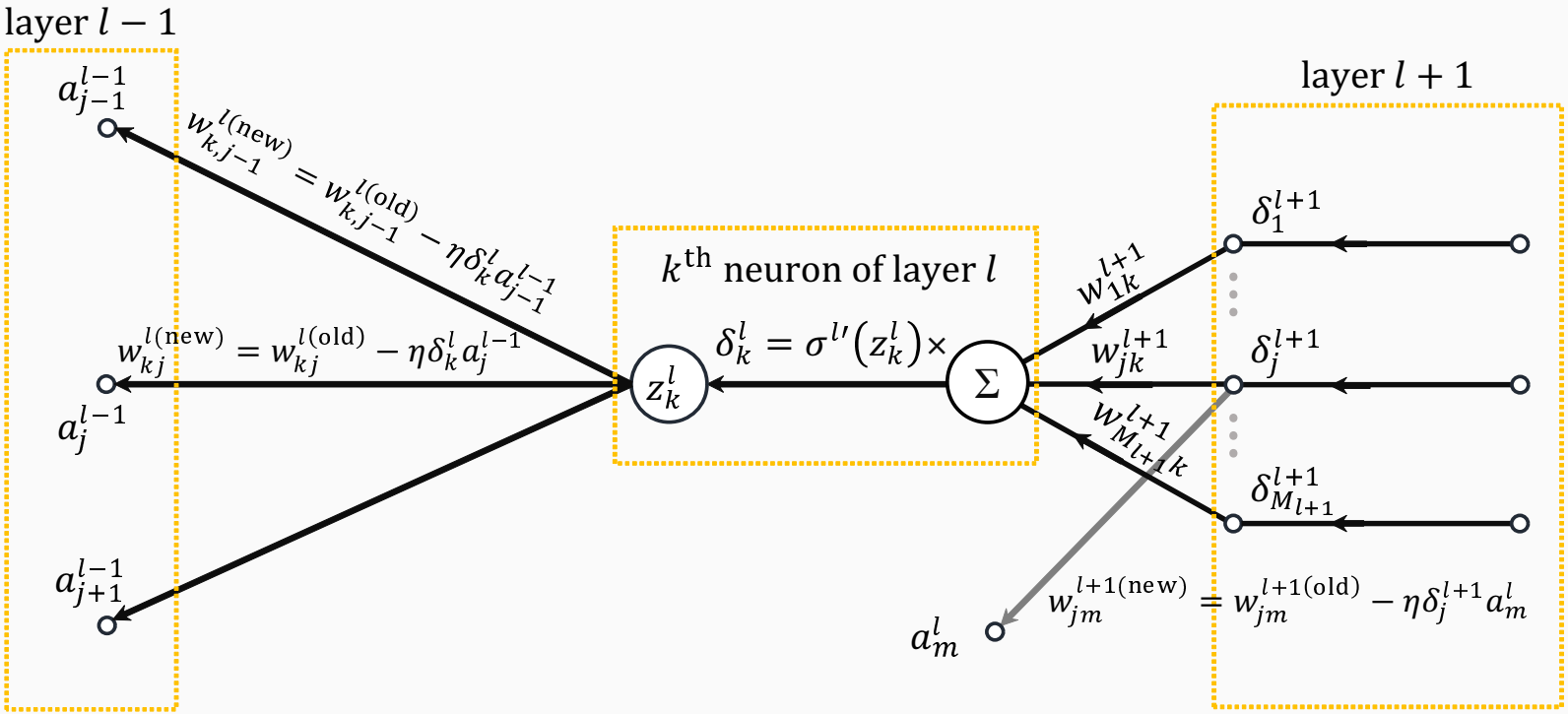
5. **Output the gradient**: 각 weight에 대해 gradient 계산

$$\frac{\partial E}{\partial w_{kj}^l} = \delta_k^l a_j^{l-1} \tag{55}$$

6. **Update weight**:

$$w_{kj}^l \leftarrow w_{kj}^l - \eta \delta_k^l a_j^{l-1} \tag{56}$$

Backpropagation: summary



Appendix

Reference and further reading

- “Chap 10 | Introduction to Artificial Neural Network with Keras” and “ Chap 11 | Training Deep Neural Networks” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow
- “Lecture 9 | Neural Networks andDeep Learning” of Kwang Il Kim, Machine Learning (2019)