Deep Learning Study

Deep Feedforward Networks

신 성 호

2023.

6. Deep Feedforward Network

- 딥 러닝의 본질적인 모델

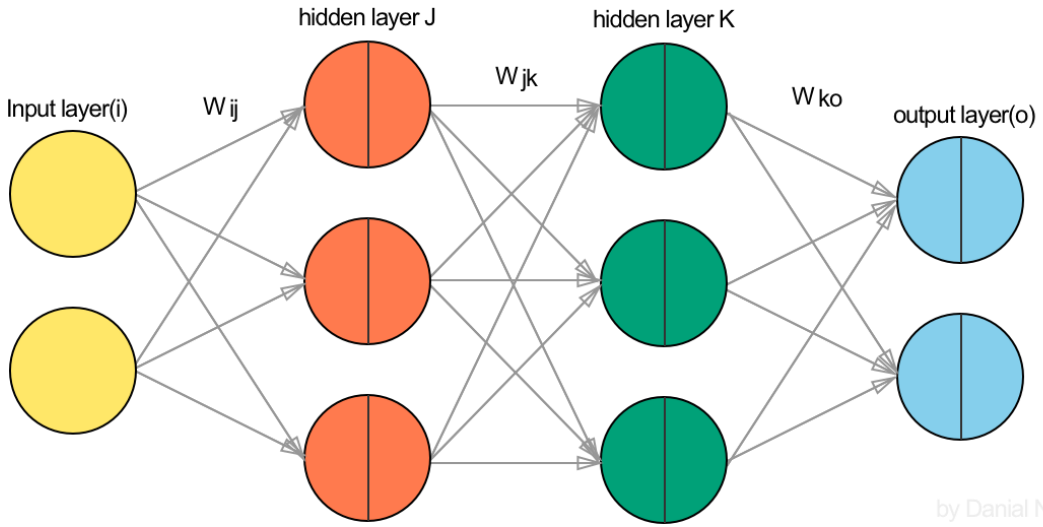
> **Deep feedforward networks**, also called **feedforward neural networks**, or **multilayer perceptrons** (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f^*(\boldsymbol{x})$ maps an input $\boldsymbol{x}$ to a category y. A feedforward network defines a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameters $\boldsymbol{\theta}$ that result in the best function approximation.

- Feedforward (순방향)
  - x로부터 함수를 거쳐 y까지 일련의 정보의 흐름

6. Deep Feedforward Network

- Depth(깊이)
  - $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$

- Hidden Layer(은닉층)
  - 중간 레이어들의 출력은 직접적으로 쓰이지 않음

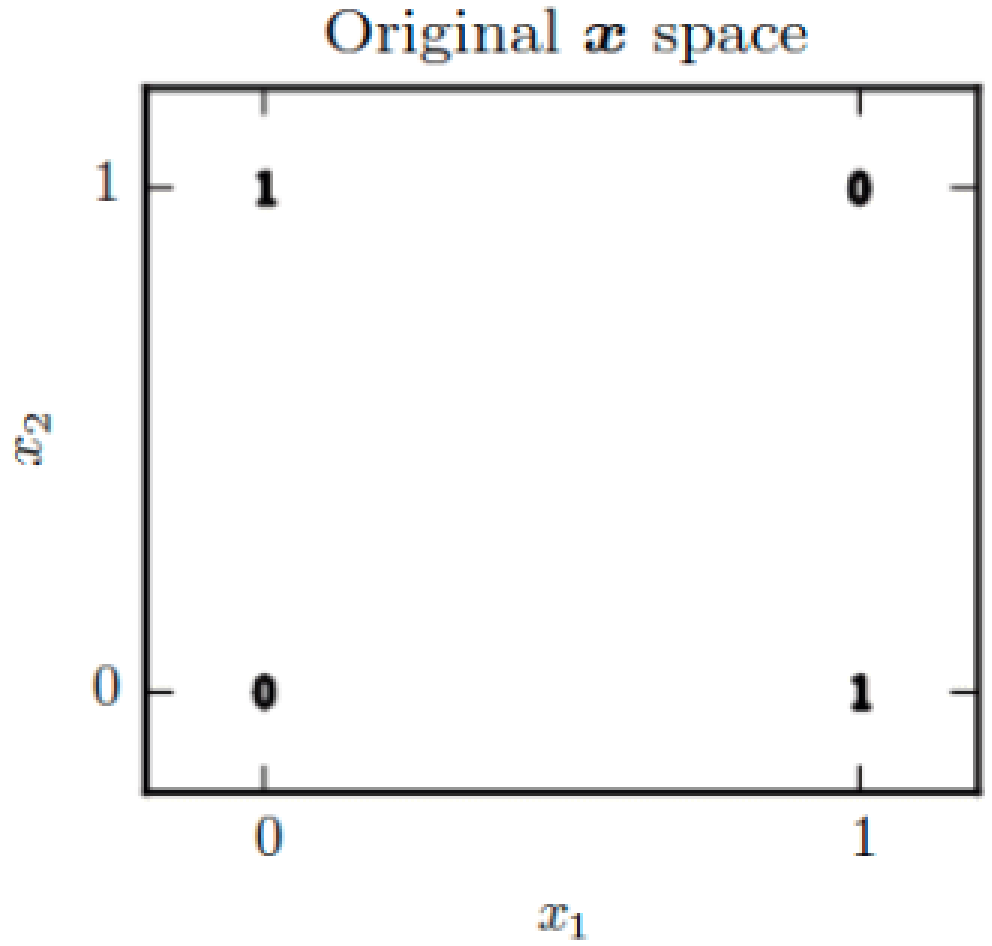


6. Deep Feedforward Network

- nonlinear transformation $\phi(\boldsymbol{x})$
  - 매우 일반적인 파이를 사용
  - 직접 파이를 손보기
  - 딥러닝을 통해 학습

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^{\top} \boldsymbol{w}$$

Ex. Learning XOR



입력		출력
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Ex. Learning XOR

- XOR 함수 $y = f^*(\boldsymbol{x})$

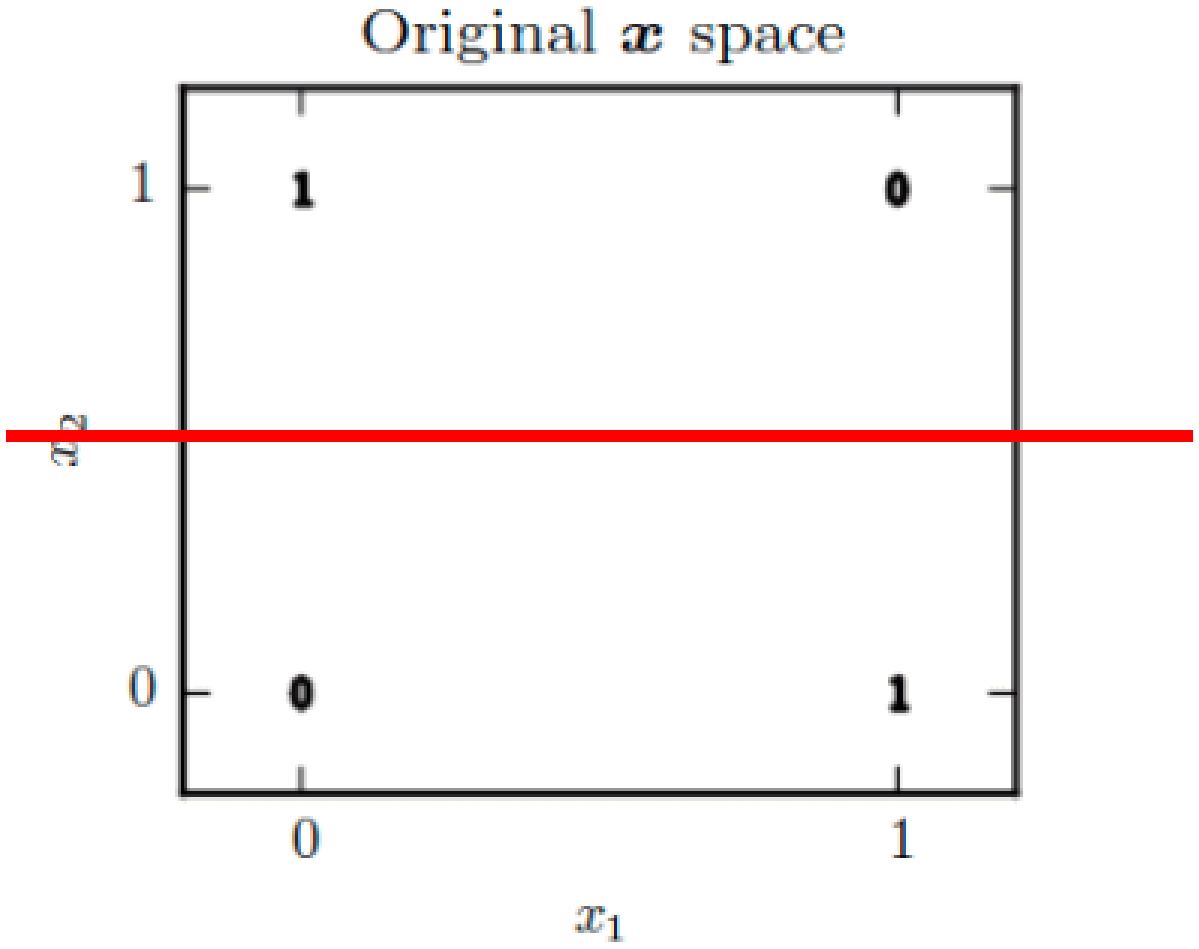
- $\mathbb{X} = \left\{ [0,0]^\top, [0,1]^\top, [1,0]^\top, [1,1]^\top \right\}$

- $J(\theta) = \frac{1}{4} \sum_{\boldsymbol{x} \in \mathbb{X}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta}))^2$

- $f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{x}^\top \boldsymbol{w} + b$

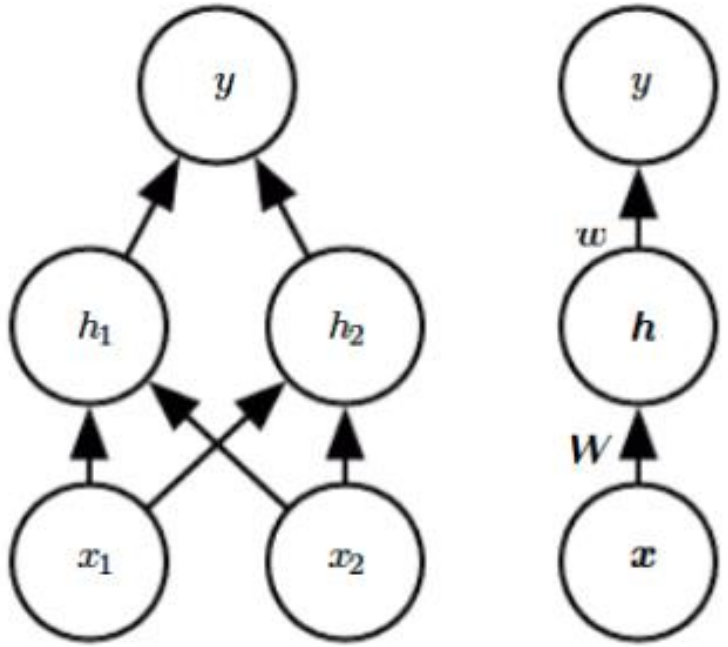
Ex. Learning XOR

- $\boldsymbol{w} = \mathbf{0}$ $b = \frac{1}{2}$



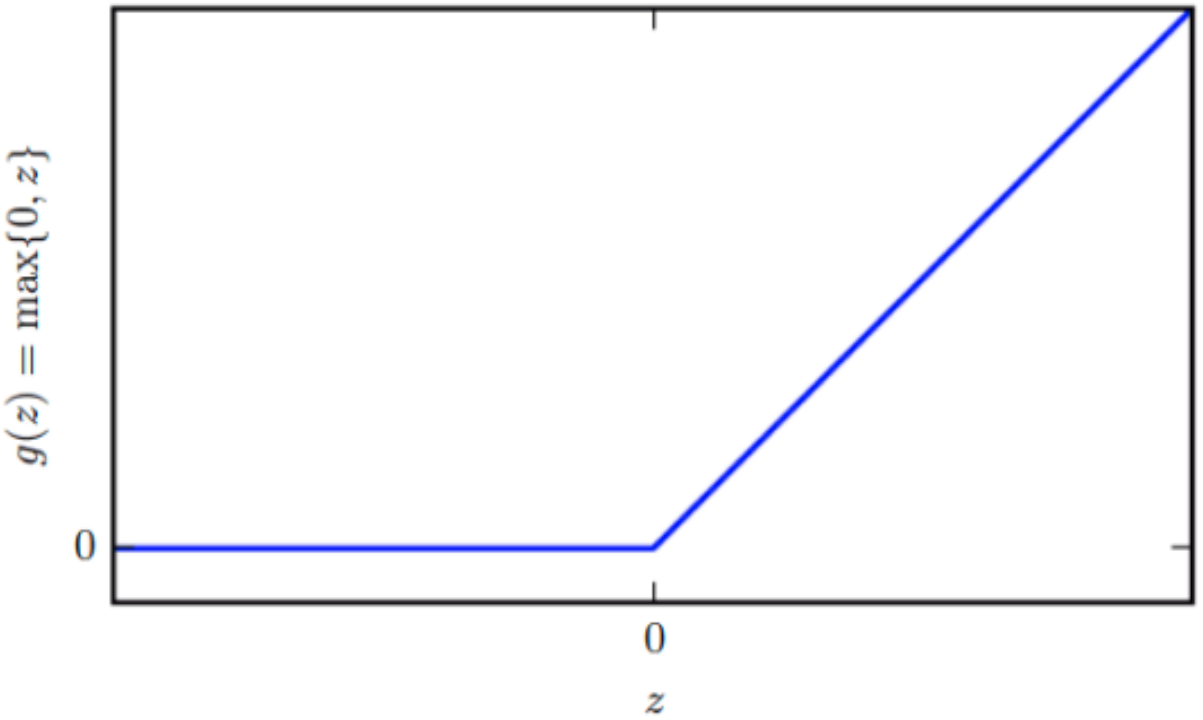
Ex. Learning XOR

- $f^{(1)}(\boldsymbol{x}) = \boldsymbol{h} = g(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c})$
- $f(\boldsymbol{h}) = \boldsymbol{h}^\top \boldsymbol{w} + b$
- $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{w}$



Ex. Learning XOR

- Relu (Rectified Linear Unit) 함수



Ex. Learning XOR

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b$$

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Ex. Learning XOR

$$\boldsymbol{XW} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

$$\boldsymbol{XW} + \boldsymbol{c} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

$$h = \max(0, \boldsymbol{XW} + \boldsymbol{c}) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Ex. Learning XOR

We finish with multiplying by the weight vector $\boldsymbol{w}$:

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

6.2 Cost Functions

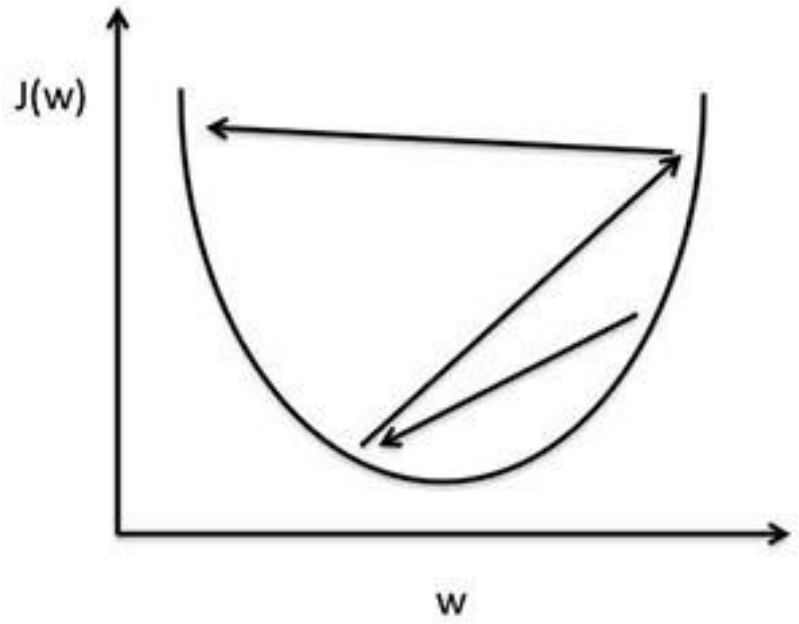
- MLE : Cross- Entropy

$$J(\theta) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{y} \mid \boldsymbol{x})$$

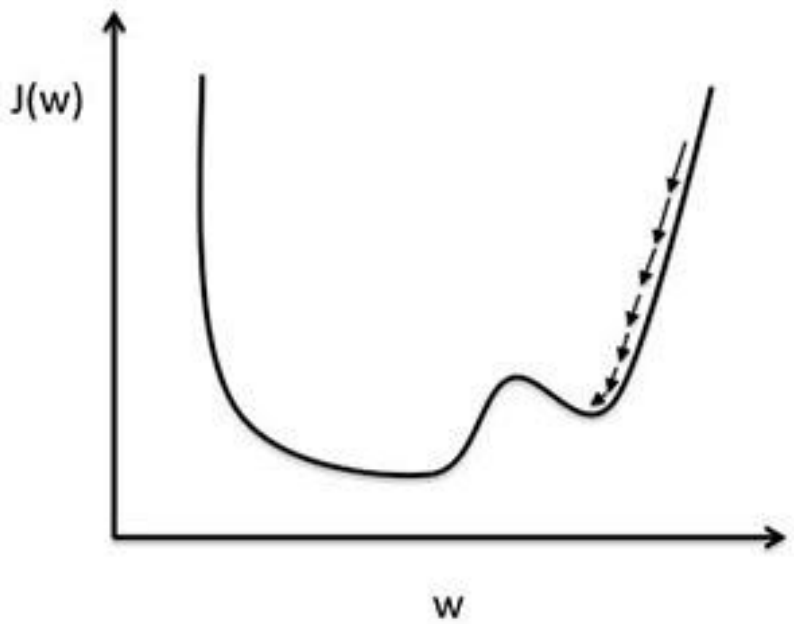
  - 최소값을 찾기 힘들다는 특성이 있음
  - 이산 변수에 대해 모델들이 확률을 0 또는 1로 표현하기 어려움
  - 출력의 밀집도(density)를 조절

6.2 Cost Functions

- Satruration & Gradient Vanishing



Large Learning Rate



Small Learning Rate

6.2 Cost Functions

- Function & Functional

$$f(x) = x^2 - 4x + 5$$

$x = 3 \;\; f(x) = 2$

$$J[y(x)] = \int_0^1 (y^2 - 4xy + 5x^2)dx$$

$y(x) = 2x \quad J[y(x)] = \int_0^1 (4x^2 - 8x^2 + 5x^2)dx = \frac{1}{3}$

6.2 Cost Functions

- Calculus of variations

$$f^* = \arg\min_f \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} \|\boldsymbol{y} - f(\boldsymbol{x})\|^2$$

$$f^*(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}\sim p_{data}(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}]$$

$$f^* = \arg\min_f \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} \|\boldsymbol{y} - f(\boldsymbol{x})\|_1$$

6.2 Cost Functions

- Calculus of variations

$$H[p] = -\mathbb{E}_x \log p(x).$$

$$H[p] = -\int p(x) \log p(x) dx.$$

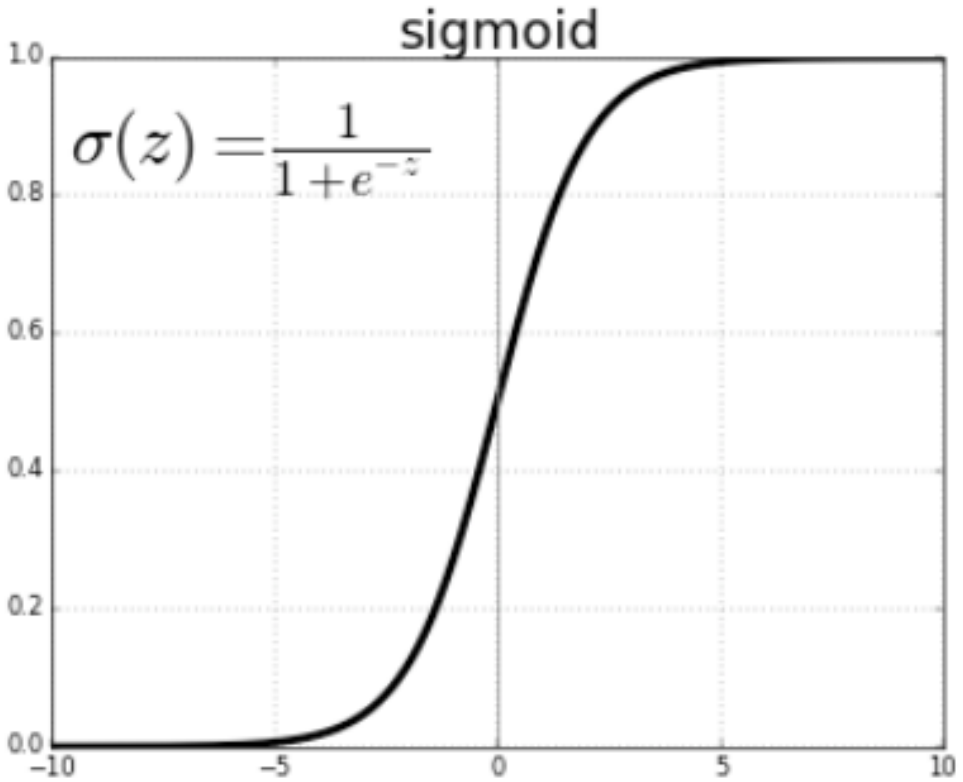
$$\mathcal{L}[p] = \lambda_1 \left(\int p(x) dx - 1 \right) + \lambda_2 \left(\mathbb{E}[x] - \mu \right) + \lambda_3 \left(\mathbb{E}[(x-\mu)^2] - \sigma^2 \right) + H[p] \quad (19.50)$$

$$= \int \left(\lambda_1 p(x) + \lambda_2 p(x) x + \lambda_3 p(x)(x-\mu)^2 - p(x) \log p(x) \right) dx - \lambda_1 - \mu\lambda_2 - \sigma^2 \lambda_3. \quad (19.51)$$

$$p(x) = \mathcal{N}(x; \mu, \sigma^2).$$

6.2.2 Output Units

- Cost function 선택 -> Output unit 선택
- Sigmoid Units



6.2.2 Output Units

- 베르누이 분포

$$P(y = 1 \mid \boldsymbol{x}) = \max\left\{0, \min\left\{1, \boldsymbol{w}^\top \boldsymbol{h} + b\right\}\right\}$$

$$\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{h} + b)$$

6.2.2 Output Units

$$\log \tilde{P}(y) = yz$$

$$\tilde{P}(y) = \exp(yz)$$

$$P(y) = \frac{\exp(yz)}{\sum_{y'=0}^{1} \exp(y'z)}$$

$$p(y) = \sigma((2y-1)z)$$

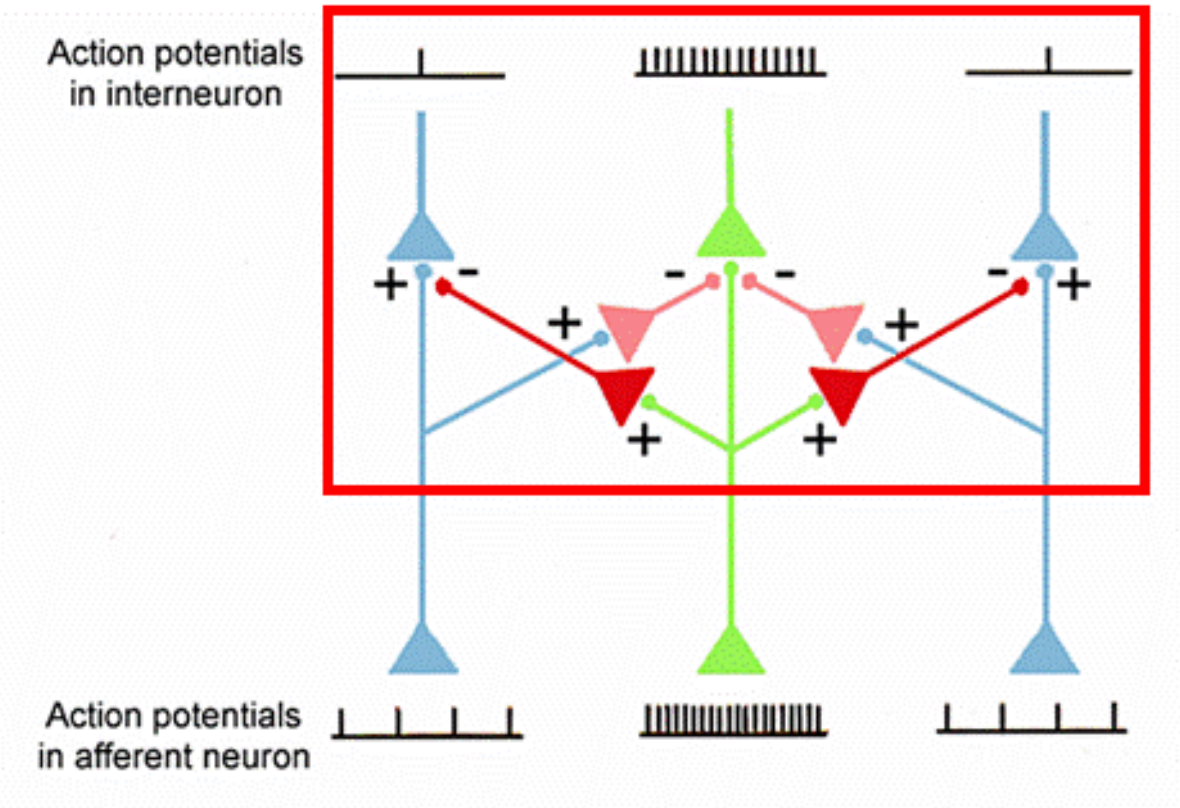
$$J(\boldsymbol{\theta}) = -\log P(y \mid \boldsymbol{x})$$

$$= -\log \sigma((2y-1)z)$$

$$= \zeta((1-2y)z)$$

6.2.2 Output Units

- Softmax

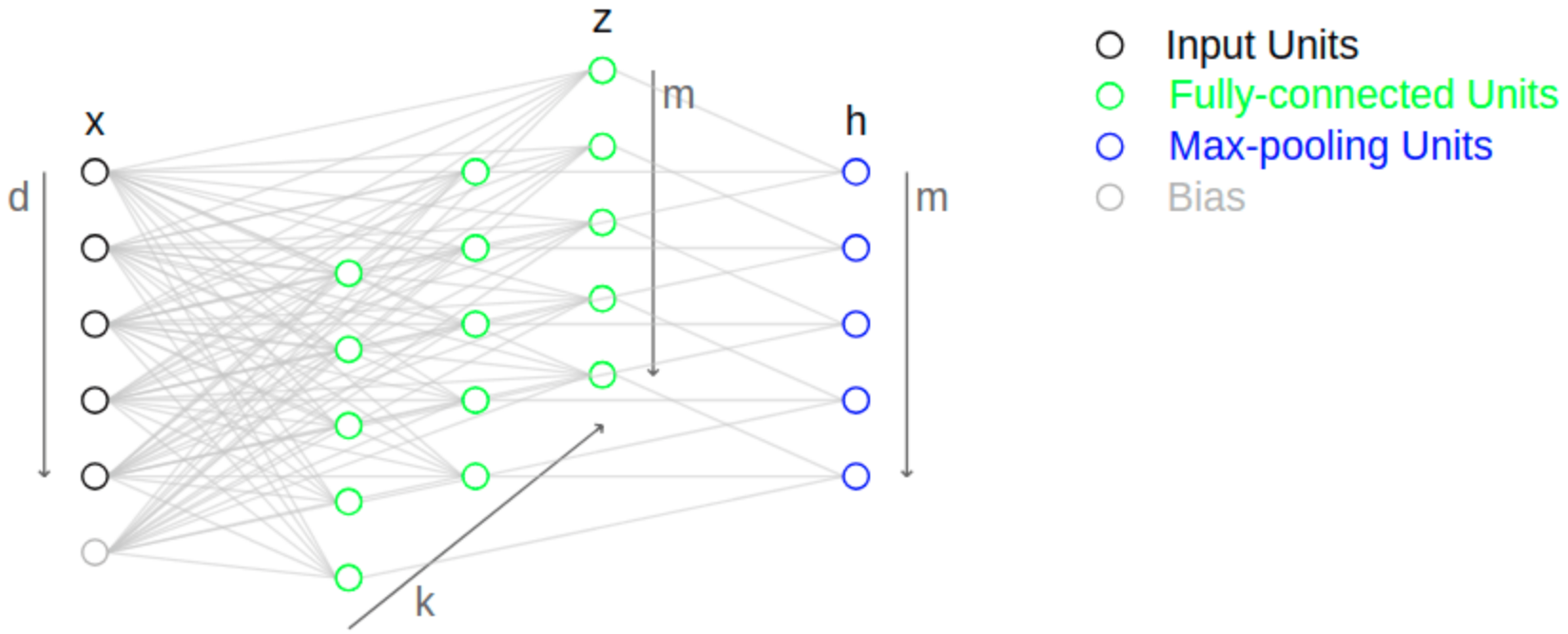


6.3 Hidden Units

- ReLu & Maxout Unit

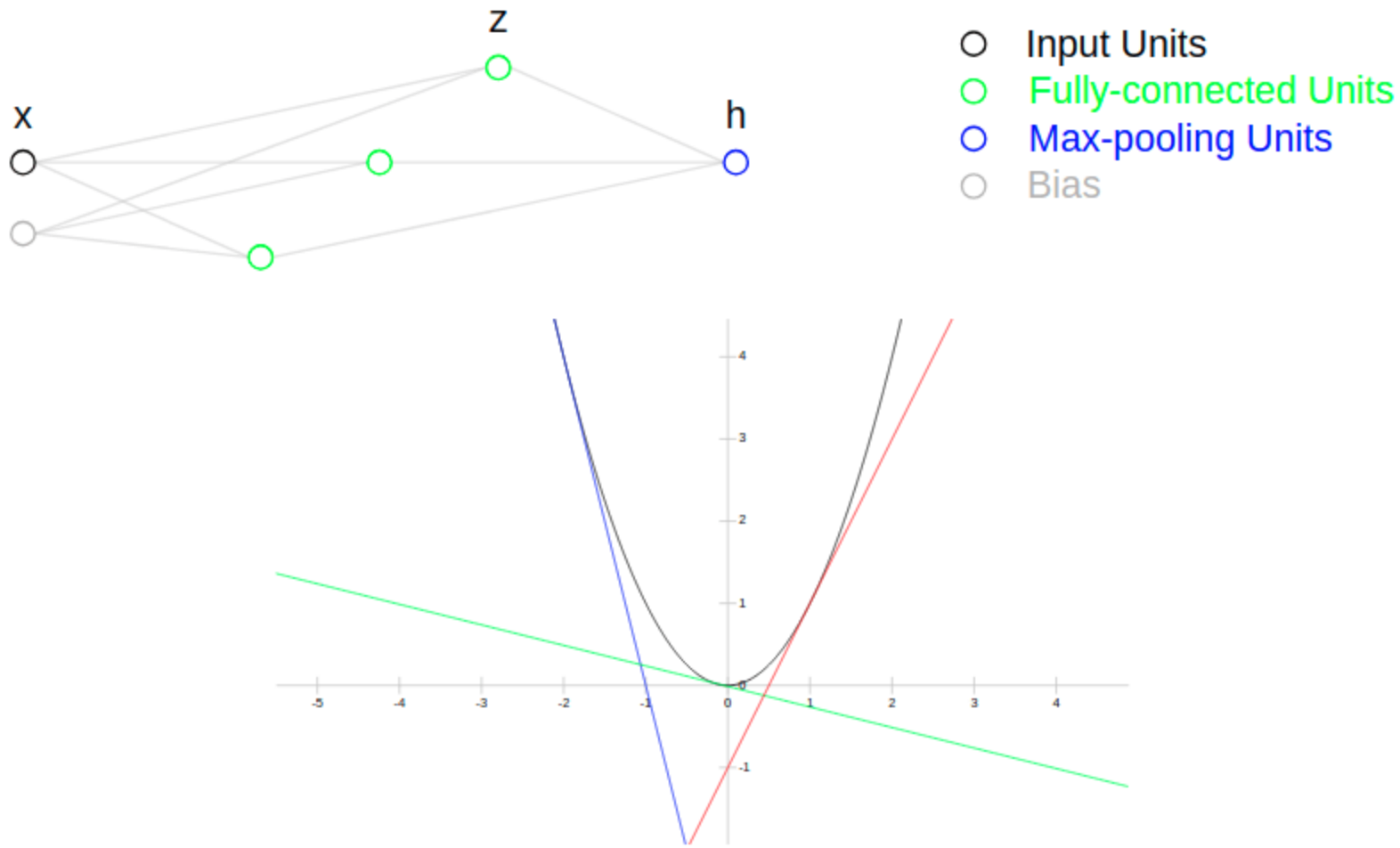
$$g(\boldsymbol{z})_i = \max_{j \in [1,k]} z_j$$

$$z_{ij} = \boldsymbol{x}^\top \boldsymbol{W}_{\ldots ij} + b_{ij}$$



6.3 Hidden Units

- ReLu & Maxout Unit



6.4 Architecture Design

- 공짜 점심은 없다 이론
  - 최적화 이론이 학습해야할 함수에 적합하지 않음
  - 학습 알고리즘이 오버피팅되어 엉뚱한 함수를 학습

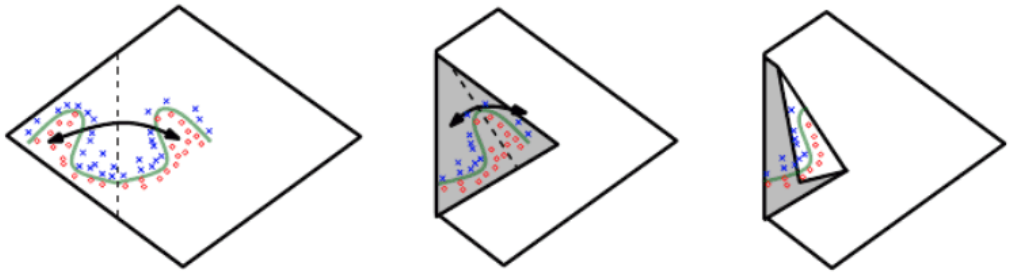
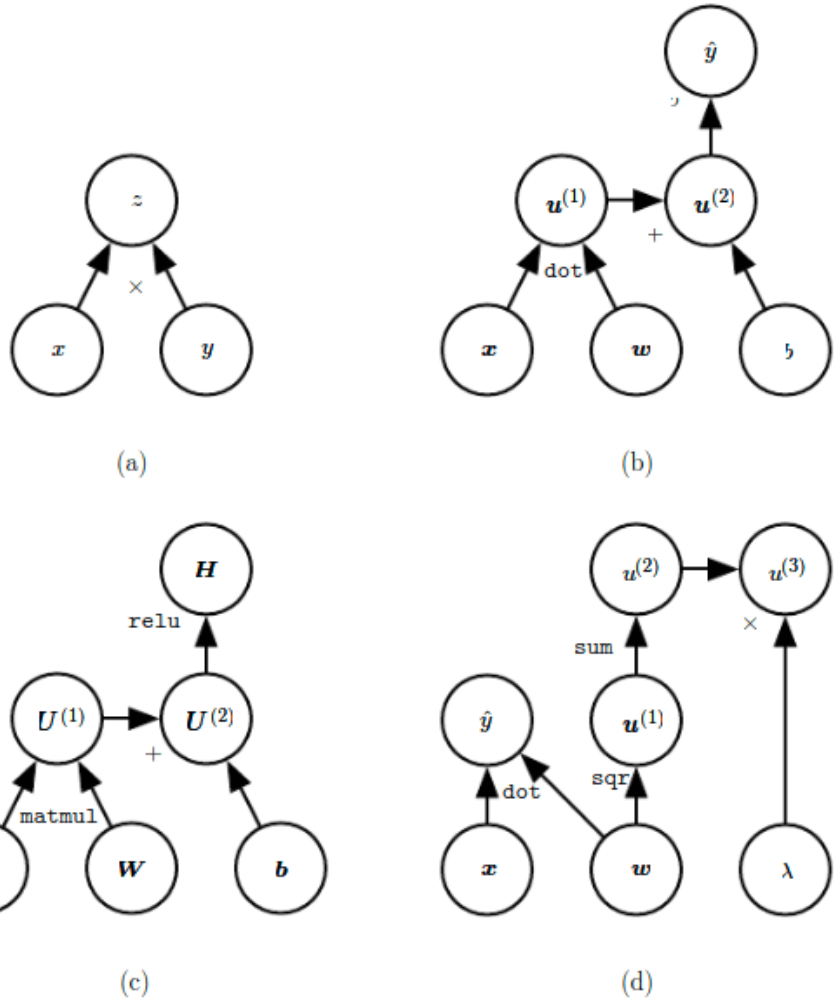
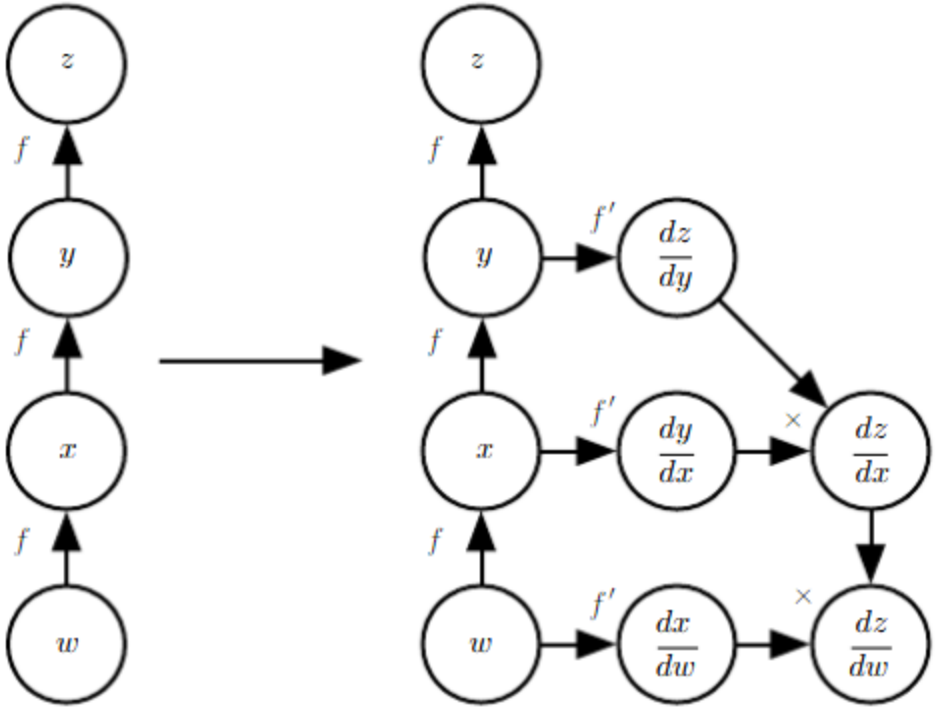


Figure 6.5: An intuitive, geometric explanation of the exponential advantage of deeper rectifier networks formally by Montufar *et al.* (2014). *(Left)* An absolute value rectification unit has the same output for every pair of mirror points in its input. The mirror axis of symmetry is given by the hyperplane defined by the weights and bias of the unit. A function computed on top of that unit (the green decision surface) will be a mirror image of a simpler pattern across that axis of symmetry. *(Center)* The function can be obtained by folding the space around the axis of symmetry. *(Right)* Another repeating pattern can be folded on top of the first (by another downstream unit) to obtain another symmetry (which is now repeated four times, with two hidden layers). Figure reproduced with permission from Montufar *et al.* (2014).

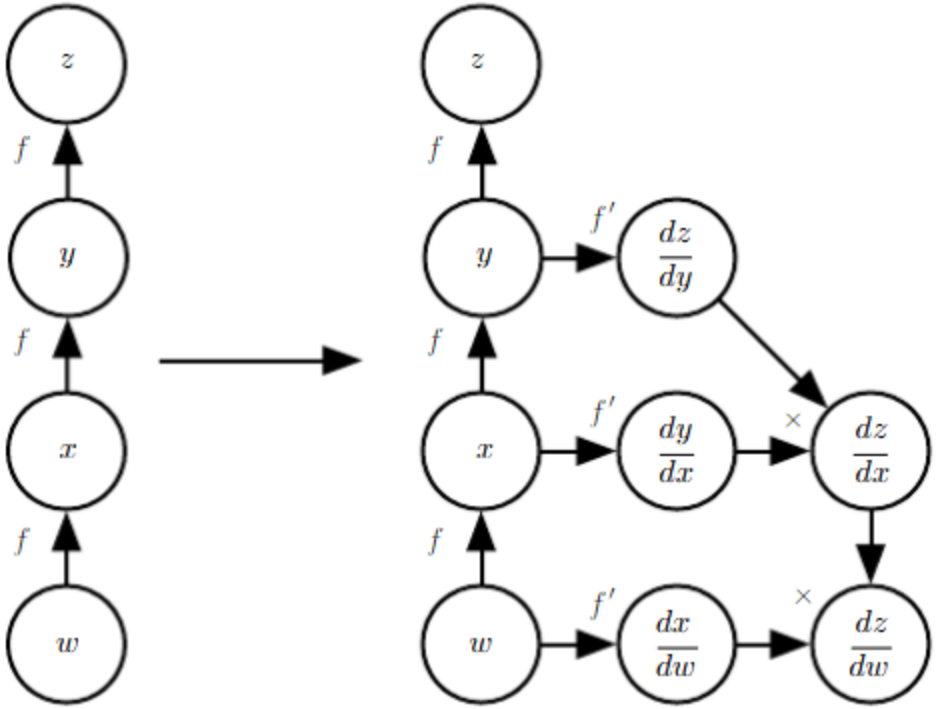
6.5 Back-Propagation



6.5 Back-Propagation



6.5 Back-Propagation



6.5 Back-Propagation