About Machine Learning

KHU spondlab
학부) 최 정 훈

Contents

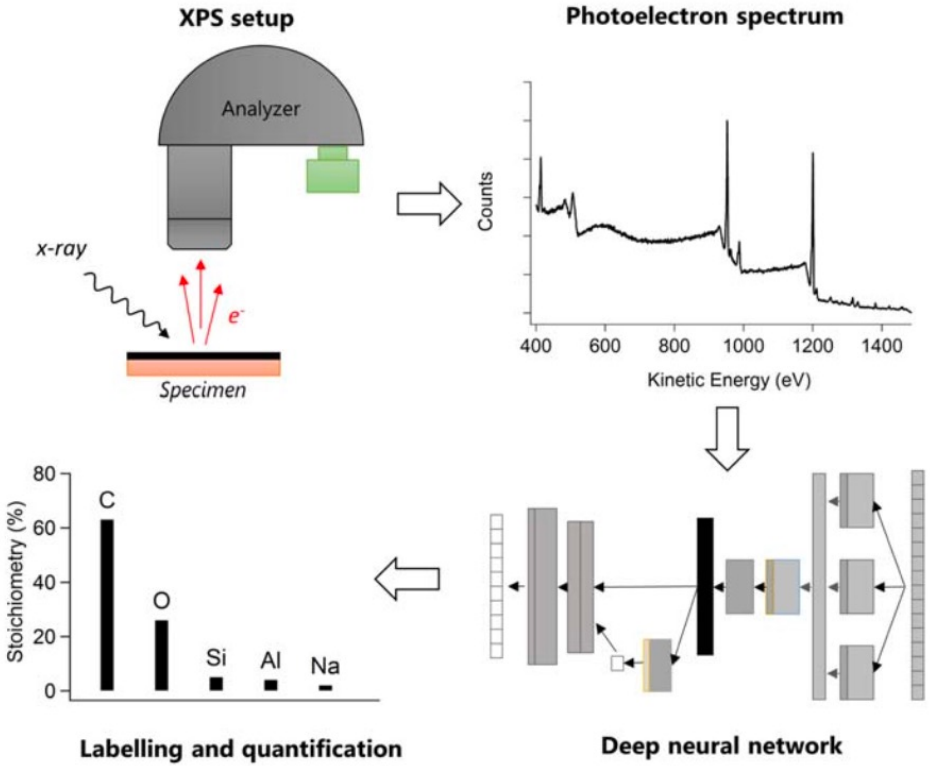
01 Model's components

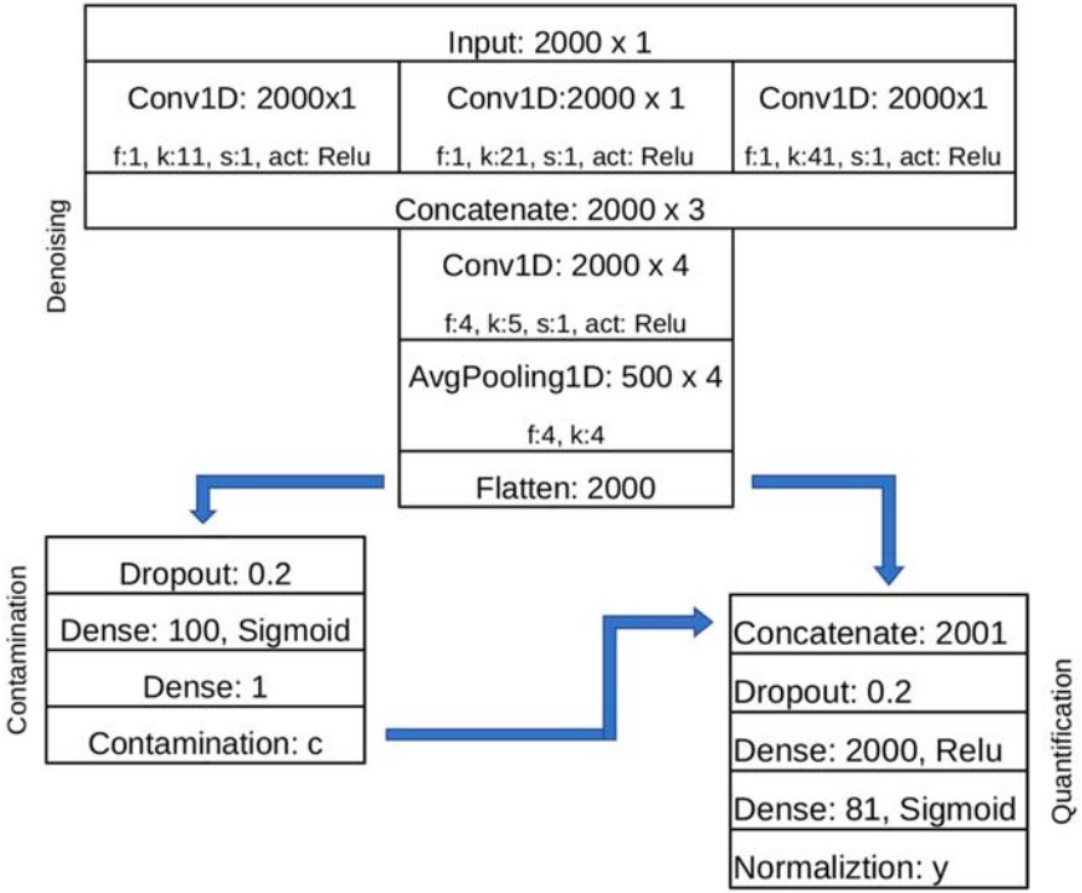
02 Deep Neural Network

03 CNN

Introduction

Training: 100k x (2000 x 1)
Test: 534 x (2000 x 1)



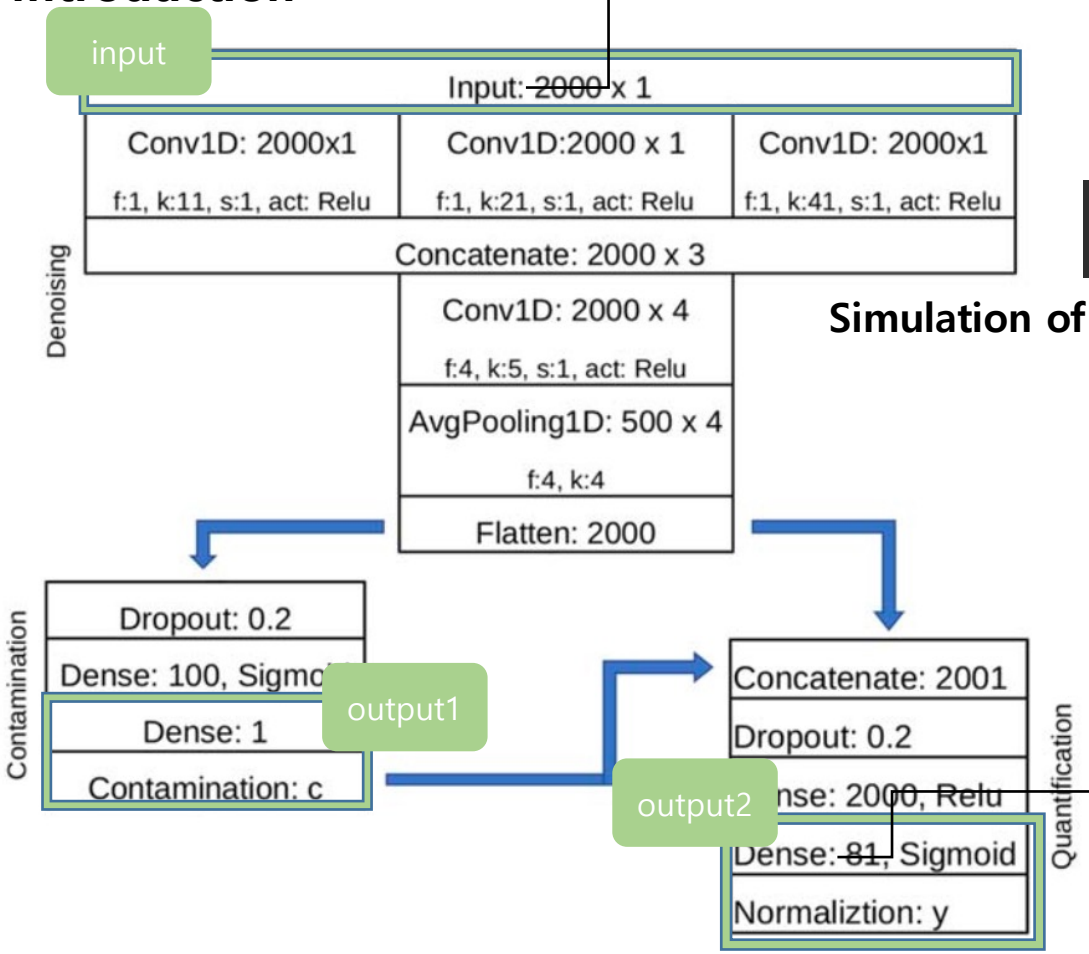


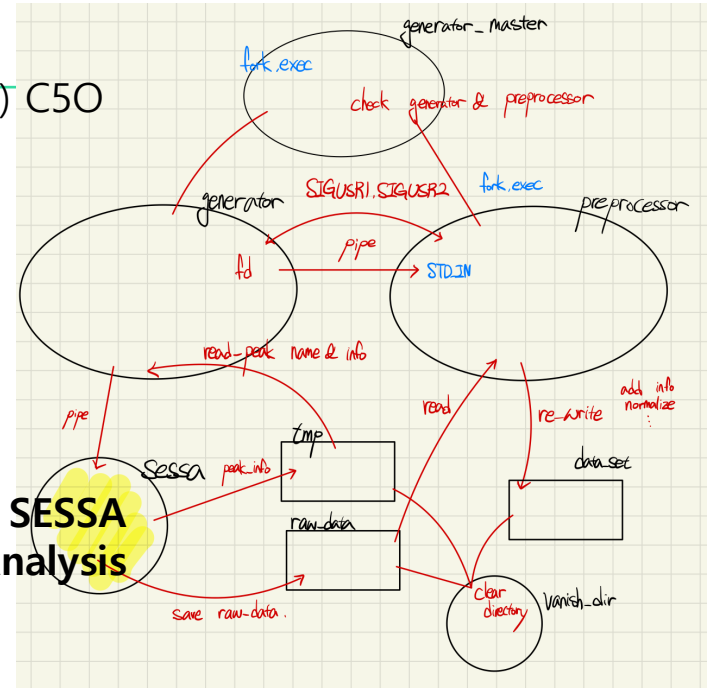
Deep neural network for x-ray photoelectron spectroscopy data analysis

Ph.D. G Drera

임의의 원소(1-5개) 임의의 비율+임의의 두께(0-40A) C5O
x10k

Introduction





NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

SESSA

Simulation of Electron Spectra for Surface Analysis

Li(3)-Bi(83) ; Pm(61) 제외
총 80개

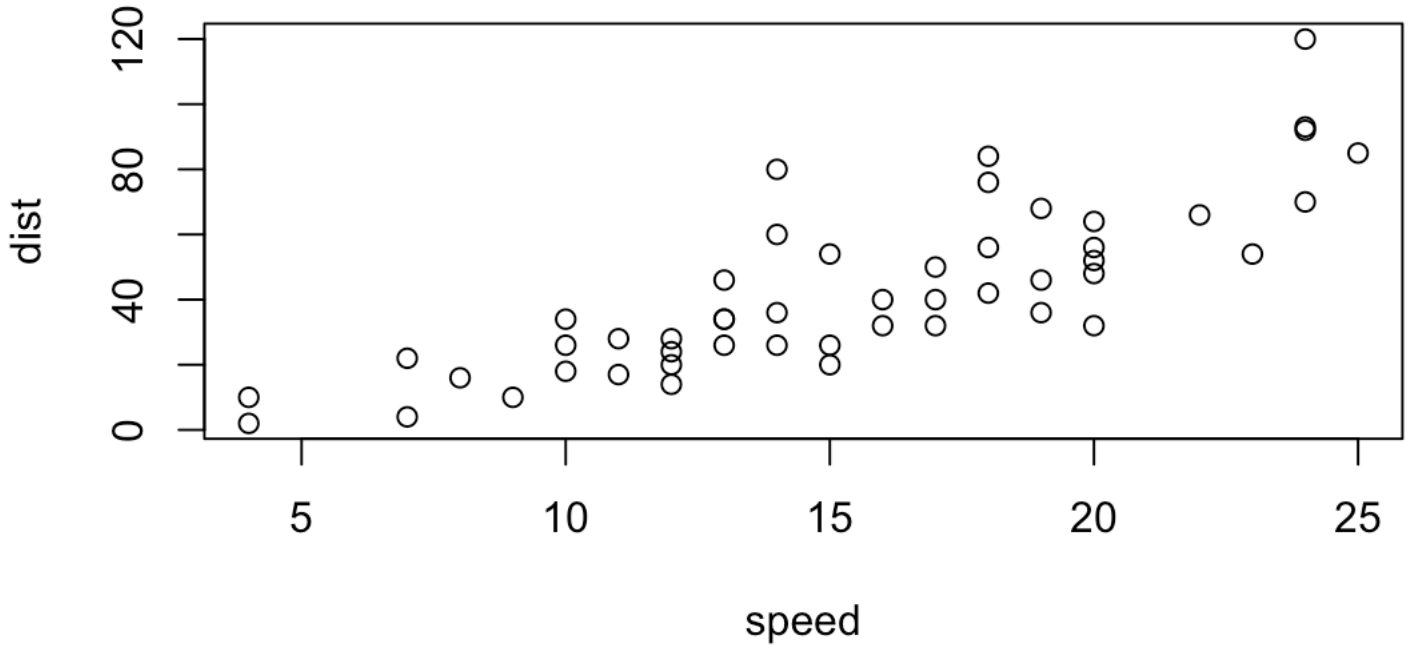
Deep neural network for x-ray photoelectron spectroscopy data analysis

Ph.D. G Drera

01. Model Components

선형 회귀 분석

	Speed	Stopping dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
9	10	34
10	11	17
11	11	28
12	12	14



1) 모델

$$y = \beta_0 + \beta_1 X + \varepsilon$$

2) Objective Function

2-1) Loss Function: 데이터 하나에 대한 목적 함수

$$J = SSE = \sum_i \varepsilon_i^2$$

$$= \sum_i (y_i - \beta_0 - \beta_1 x_i)^2$$

Least square Method

$$\frac{\partial J}{\partial \beta_0} = -2\sum_i (y_i - \beta_0 - \beta_1 x_i) = 0 \quad \Rightarrow \quad \hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\frac{\partial J}{\partial \beta_1} = -2\sum_i (y_i - \beta_0 - \beta_1 x_i) x_i = 0 \qquad \hat{\beta}_2 = \bar{y} - \hat{\beta}_1 \bar{x}$$

2-2) Cost Function: 전체 데이터의 오차; loss function 평균

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$BCE(x) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\left(h(x_i;\theta)\right) + (1-y_i)\log\left(1-h(x_i;\theta)\right)$$

과적합을 방지하기 위해,
일반적으로 DNN에서는 비용함수에 가중치에 대한 비용을 추가

Lasso $J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha\sum_{i=1}^{n} |\theta_i|$

Ridge $J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$

Elastic Net $J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + r\alpha\sum_{i=1}^{n}|\theta_i| + \frac{1-r}{2}\alpha\sum_{i=1}^{n}\theta_i^2$

$$y = \beta_0 + \beta_1 X + \beta_2 X^2$$

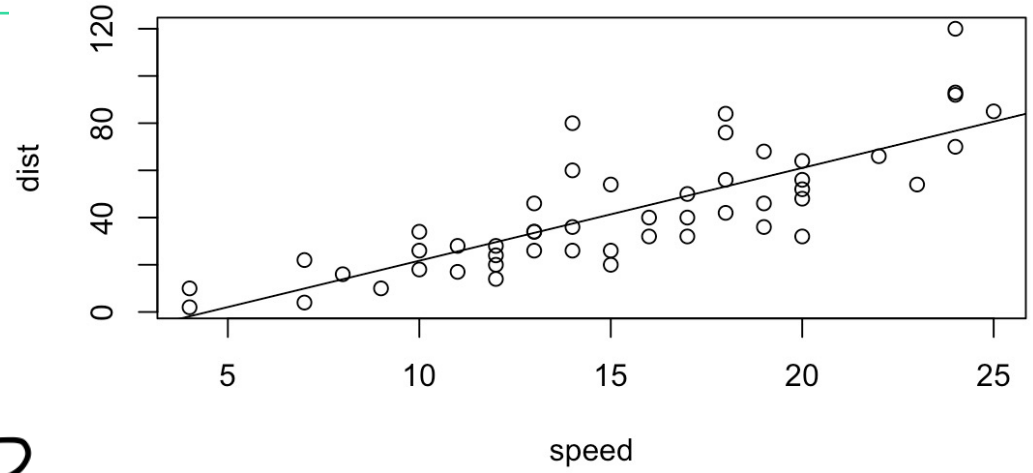
$$\begin{pmatrix} X_1^2 & X_1 & 1 \\ X_2^2 & X_2 & 1 \\ \vdots & \vdots & \vdots \\ X_n^2 & X_n & 1 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \Rightarrow AX = B$$

$$\min_X \|B - AX\|^2$$

$$\Rightarrow \frac{d}{dX} \|B - AX\|^2 = 0$$

$$-2A^T(B - AX) = 0$$

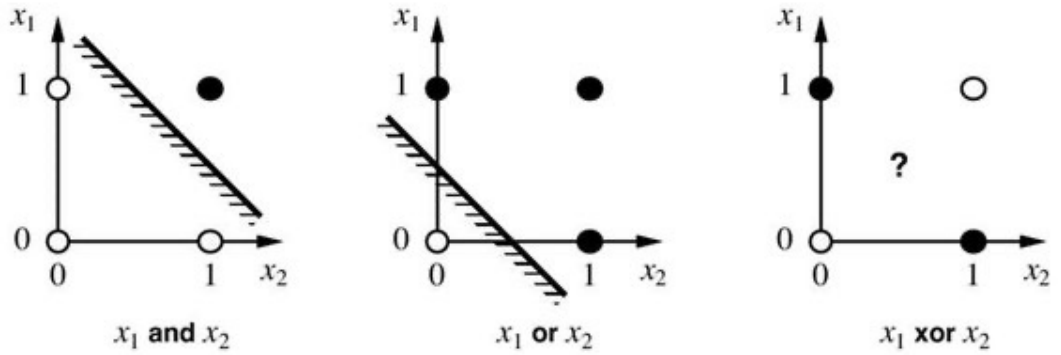
$$X = (A^TA)^{-1}A^TB \quad \text{항상 존재}$$



But,
Linear Regression 조건
1) 선형성
2) 독립성
3) 등분산성
4) 정규성

Linear separability

XOR 문제



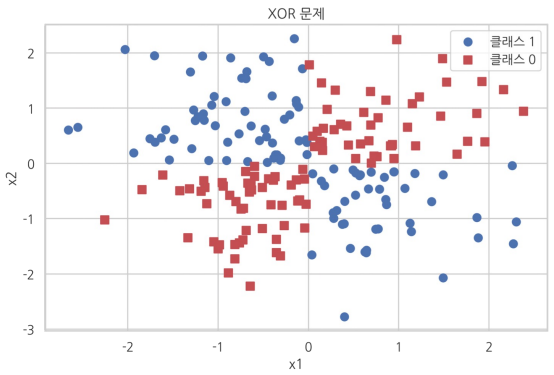
비선형성 필요

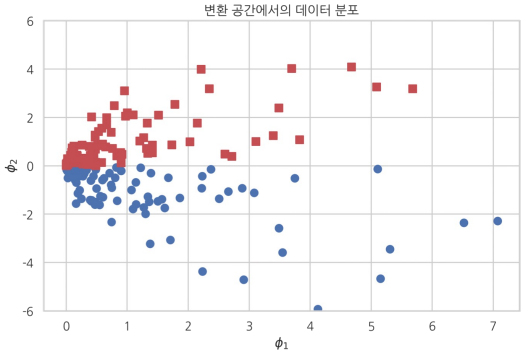
-> 고차원으로 매핑(kernel SVM, KPCA)
-> 활성화 함수(activation function)

$\phi(\cdot) : R^D \rightarrow R^M \quad (M > D)$

ex) $(x_1, x_2) \rightarrow \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$; Polynomial d=2

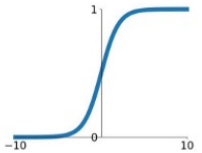
$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$



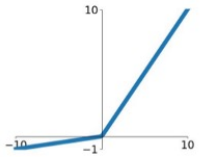


3) Activation Function

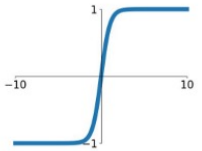
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

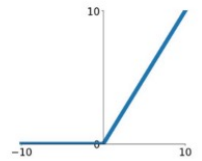


tanh
$\tanh(x)$

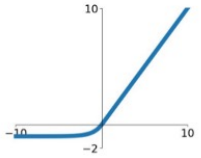


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



일반화 회귀 분석(generalized linear regression)

$g(y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots$

ex) 회귀가 아닌 이진 분류를 하는 경우 $g(y) = \ln\left(\frac{y}{1-y}\right)$; 로지스틱 함수

$y = g^{-1}(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots)$

$J = \sum_i (\hat{y}_i - y_i)^2$

$\min_{\beta} J$ 는 Newton - Raphson method를 이용해 찾아야 함.

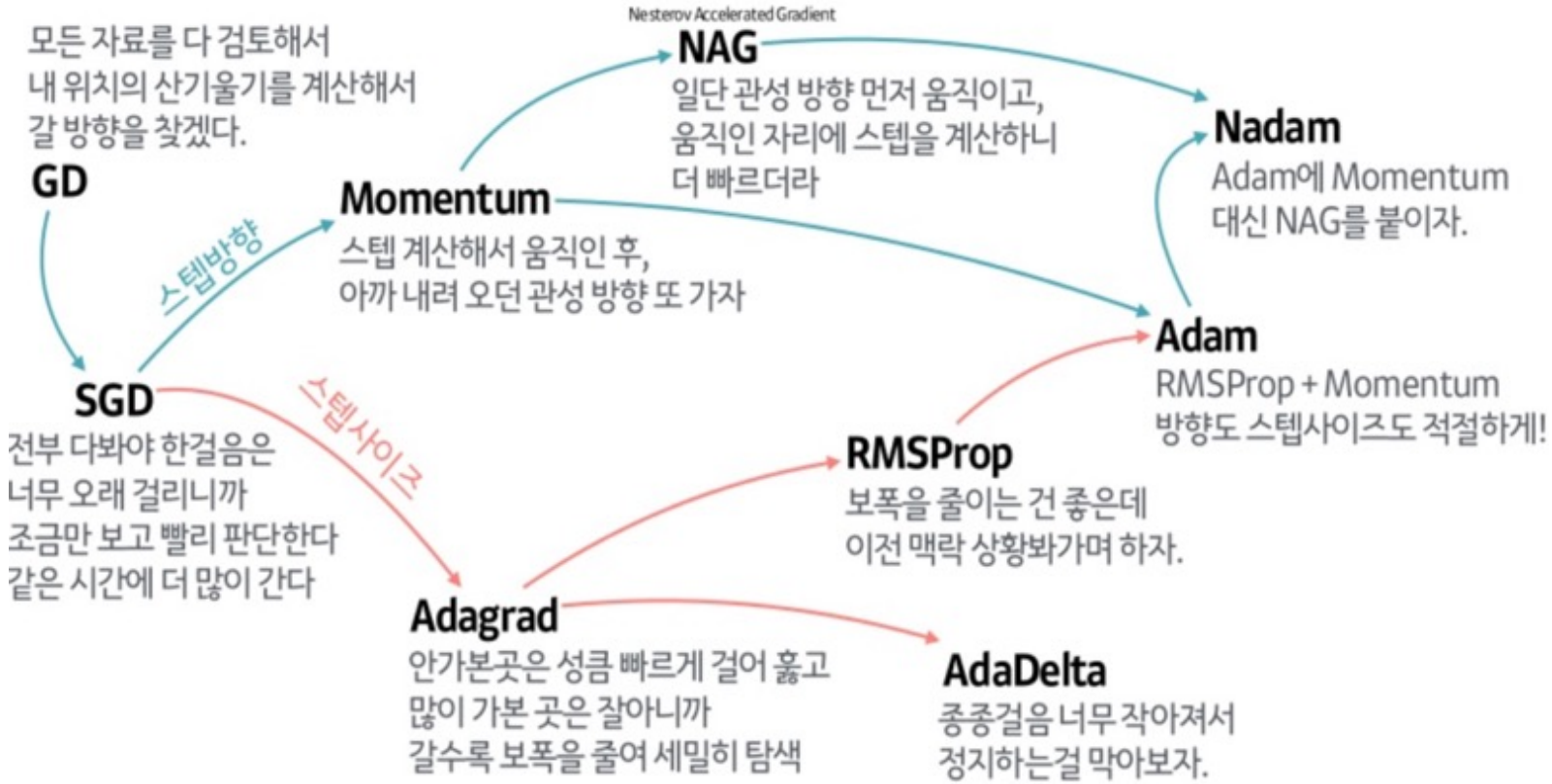
Newton-Raphson method

$$J(\beta_0, \beta_1, \beta_2)$$

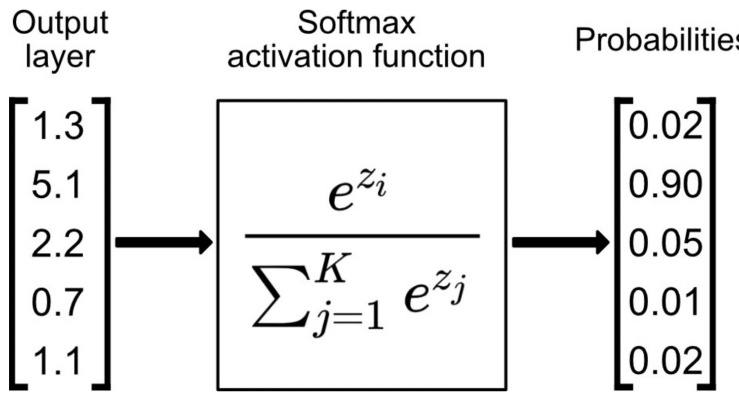
$$\min_{\beta} J \Rightarrow \beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J$$

4) Optimizer

비용이 최소가 되는 점을 찾는 방법



출력층에서 주로 사용되는 activation function



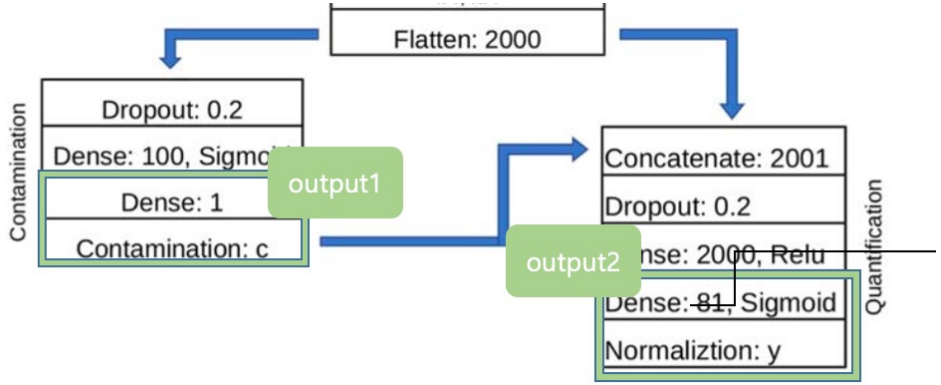
$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$X = \ln\left(\frac{y}{1-y}\right)$$

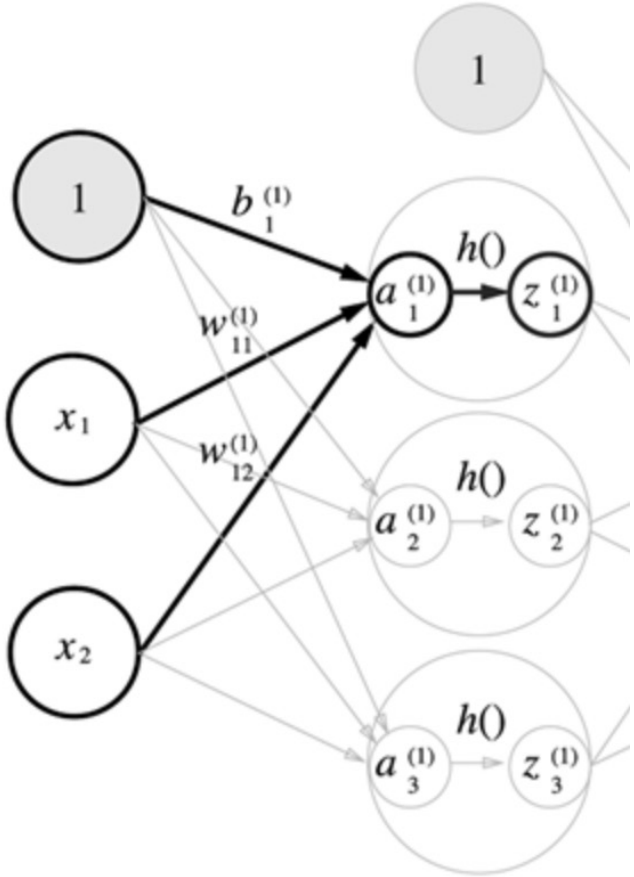
Logit: 이진 분류 시, 출력 값이 해당 라벨일 확률을 의미

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

2개 이상의 라벨로 분류할 때, 각 라벨로 분류될 확률을 의미



02. Deep Neural Network

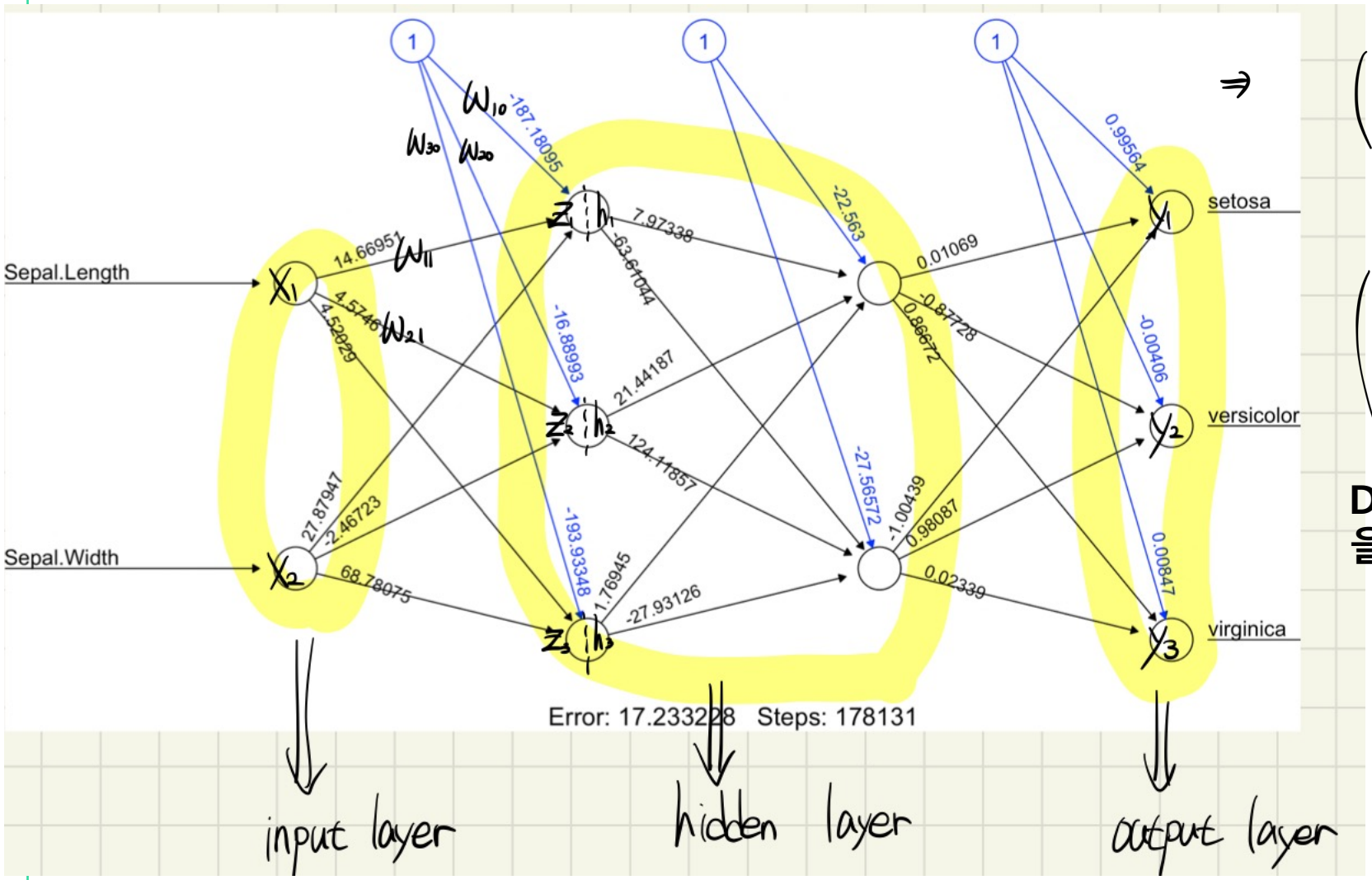


$$\left(a_1^{(k)} \quad a_2^{(k)} \quad a_3^{(k)}\right) = \left(x_1 \quad x_2\right) \begin{pmatrix} w_{11}^{(k)} & w_{12}^{(k)} & w_{13}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} & w_{23}^{(k)} \end{pmatrix} + \left(b_1^{(k)} \quad b_2^{(k)} \quad b_3^{(k)}\right)$$

Hidden layer 에서 activation function(비선형 함수)가
일반화 회귀 모델(glm)과 동일한 역할 수행

Forward propagation

$$z_1 = w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$



$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} w_{10} \\ w_{20} \\ w_{30} \end{pmatrix}$$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} f(z_1) \\ f(z_2) \\ f(z_3) \end{pmatrix} \Leftarrow \text{activation fcn}$$

DNN은 단순히 일반화 선형 회귀(glm)을 여러 겹 수행한 것과 동일함.

Back propagation; chain rule(가중치 업데이트)

ex) $w_1 = w_1 - \frac{\partial J}{\partial w_1}$

$= w_1 - \frac{\partial J}{\partial o} \times \frac{\partial o}{\partial w_1}$

$J = (y_i - \hat{y}_i)^2 = (y_i - f_2(o))^2$

$o = w_{01} + w_1 z_1 + w_2 z_2$

$\Rightarrow$

$\frac{\partial o}{\partial w_1} = z_1$

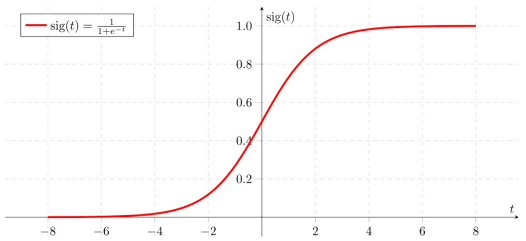
$\frac{\partial J}{\partial o} = -2(y_i - f_2(o)) f_2'(o)$

ex) $w_{11} = w_{11} - \frac{\partial J}{\partial w_{11}}$

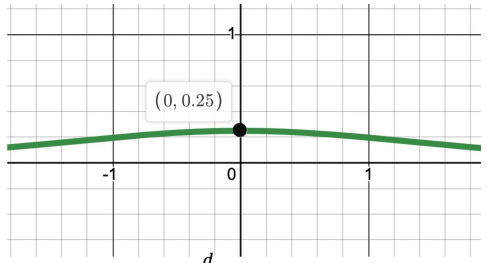
$= w_{11} - \frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial w_{11}}$ ($f_1'(y_1)$, x_1)

$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial o} \cdot \frac{\partial o}{\partial z_1} = \frac{\partial J}{\partial o} \cdot w_1$

Sigmoid는 미분할 경우, 최대값이 0.25로 기울기가 소실!
따라서 은닉층의 activation function으로 사용하지 않음



sigmoid



$\frac{d}{dx}sigmoid$

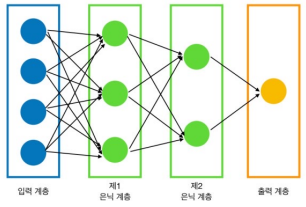
https://heytech.tistory.com/

ReLU, Leaky ReLU가 요즘 가장 많이 사용됨

02. Deep Neural Network

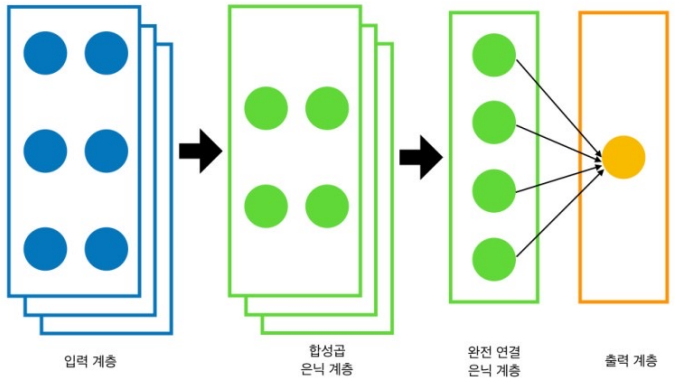
DNN

Deep Neural Network



CNN

Convolutional Neural Network



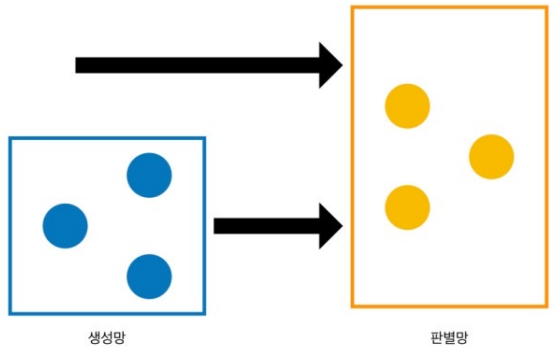
이미지 분석(vison)

로커스) 가상인간 로지

가상 이미지 생성

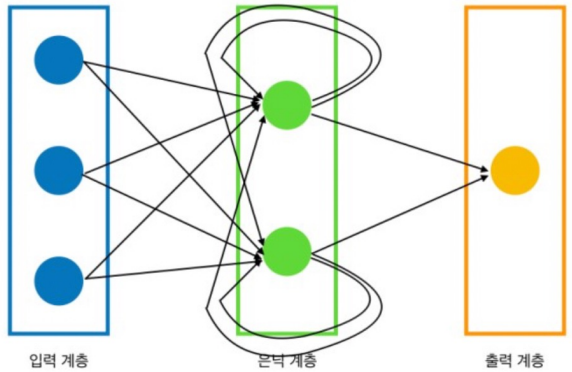
GAN

Generative Adversarial Network

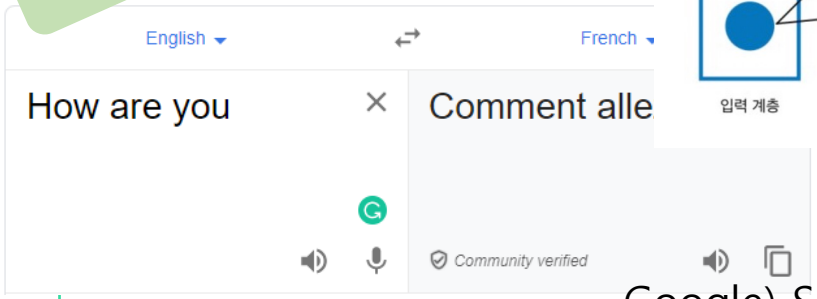


RNN

Recurrent Neural Network

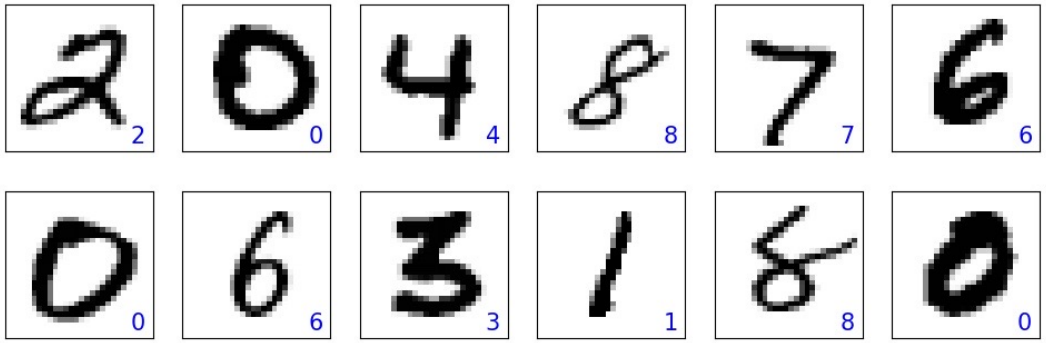


시계열 데이터/자연어



Google) Seq2Seq

03. Convolutional Neural Network



이미지 분류에 가장 많이 사용되는 MNIST
28x28x1 (8-bit gray scale)

601x601x3 (8-bit color; 0-255)
BMP

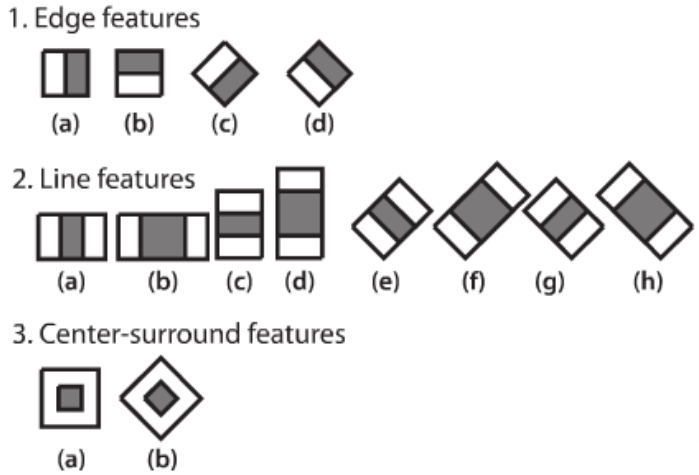
차원이 낮고, topology information이 적음

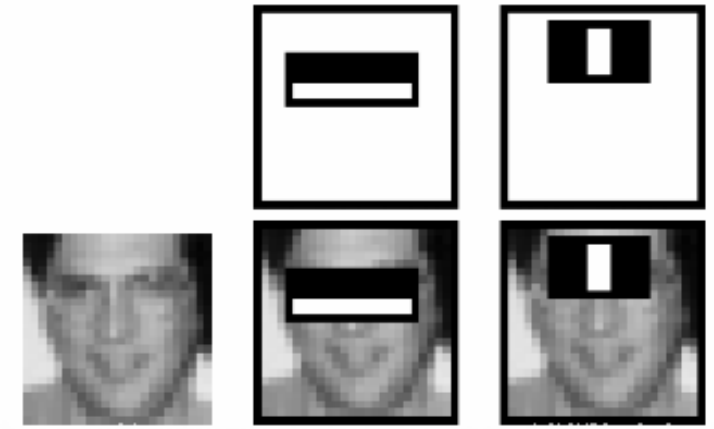
차원이 크고, topology information이 적음

**DNN은 입력 시, 1차원으로 입력(flatten) 이 때 topology information을 고려하지 않으며,
차원이 크면 많은 가중치가 필요**

CNN 사용

OpenCV Haar Cascade;





전체 이미지에서 특정 이미지를 잘라내기
Haar Cascade라는 방법을 주로 사용

CNN 모델에 전체 이미지 입력 시,
학습 불가능
(해상도 CNN 모델에 맞춰서 변형 필요)

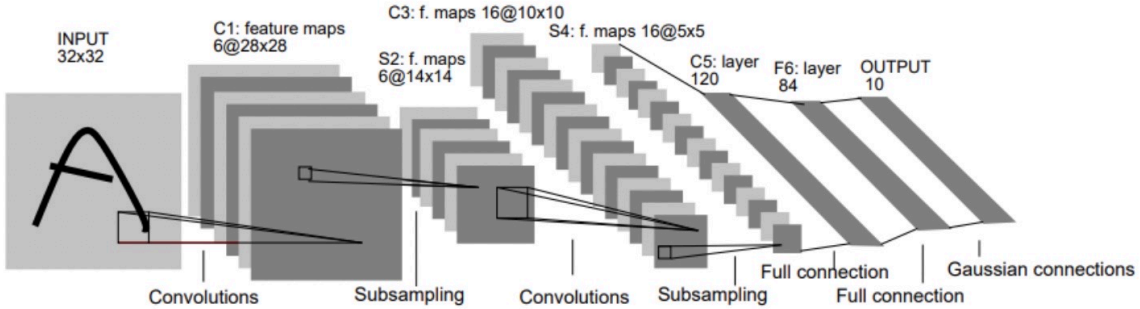


Figure 1: LeNet-5

Yan LeCun 논문, CNN의 초석

1	2	3	4
3	5	6	3
7	8	9	0
3	1	3	6

[이미지]

×

2	1
-1	2

[필터]

+ 2

[편향]

→

13	16	12
20	26	8
23	32	31

[특성]

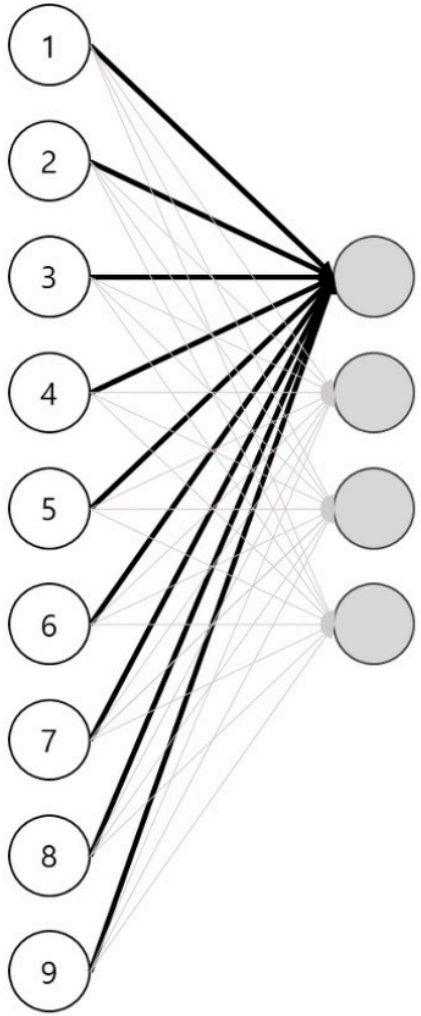
CNN에서의 가중치

1	2	3
4	5	6
7	8	9

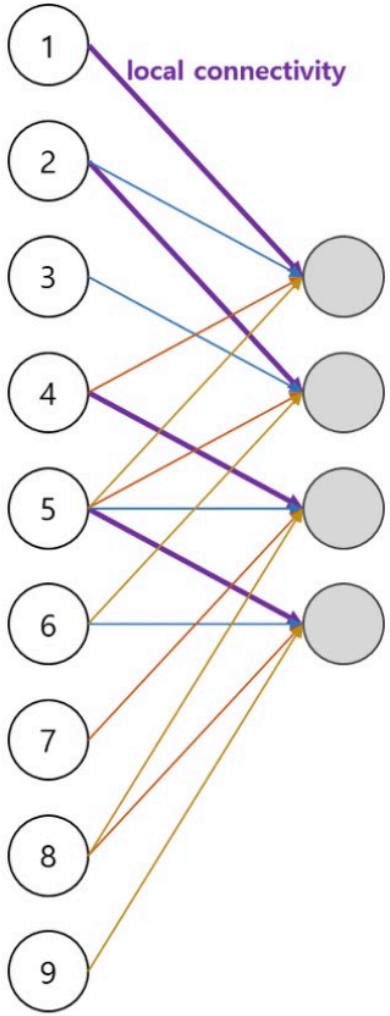
[이미지]

1	2
3	4

[필터]



[Fully Connected Layer]

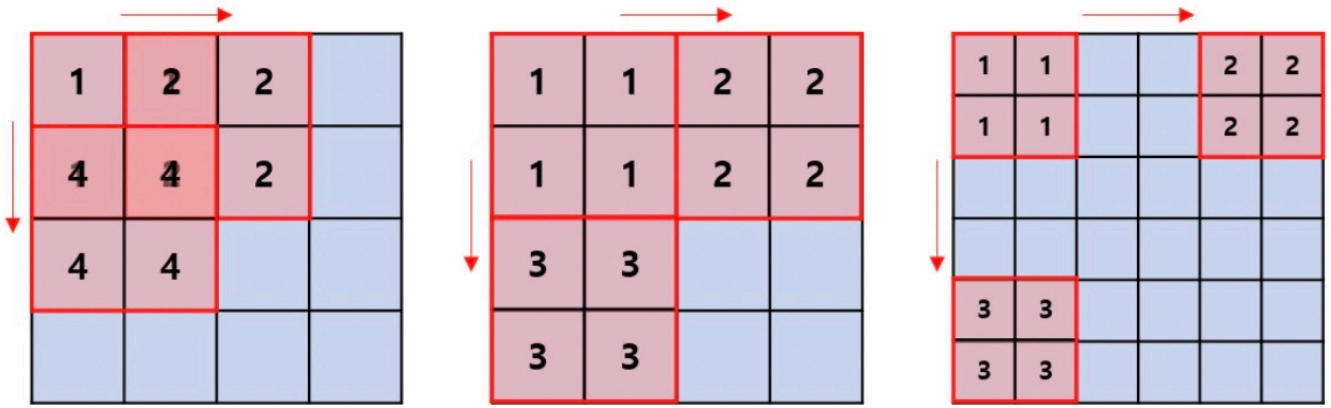


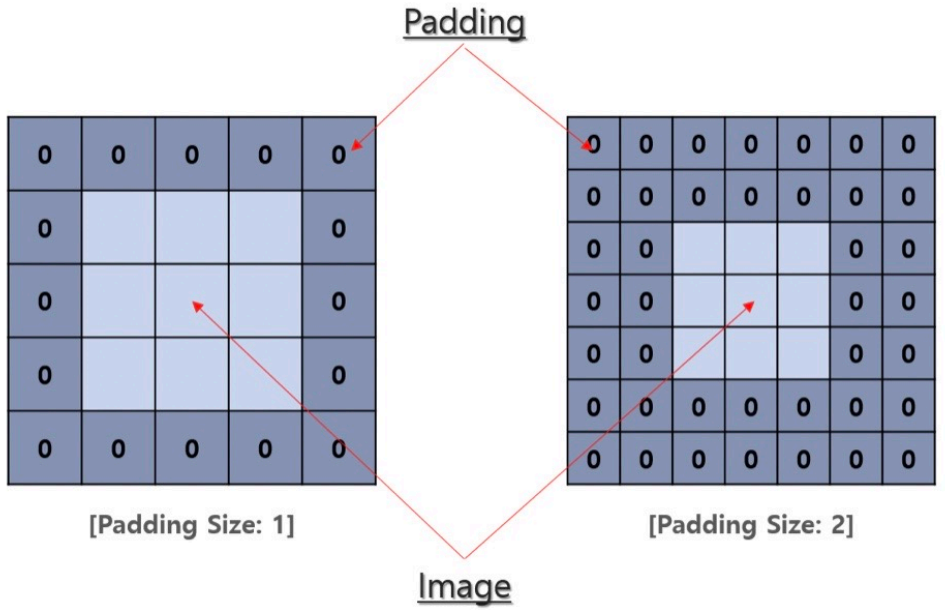
[Convolutional Layer]

DNN일 경우, (9+1) x 4 = 40 개의 가중치를
CNN을 사용하면, (4+1) x 1 = 5 개의 가중치만 이용

DNN 보다 빠른 연산 속도

Stride: 필터를 입력 데이터에 합성할 때, 움직이는 간격

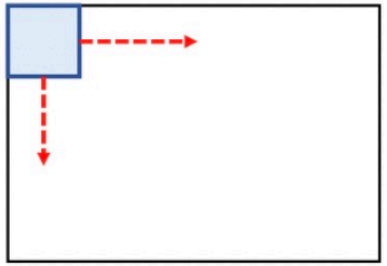




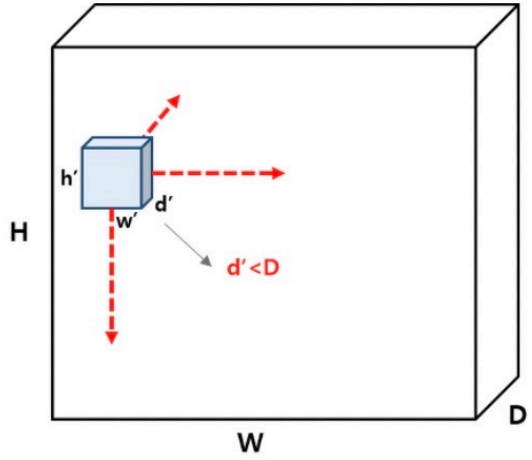
Padding: 합성 시, 특성의 크기가 작아지는 것을 막아 주기 위해 모서리를 0 으로 채움

1D Convolution / 2D Convolution / 3D Convolution

[1D Convolution]

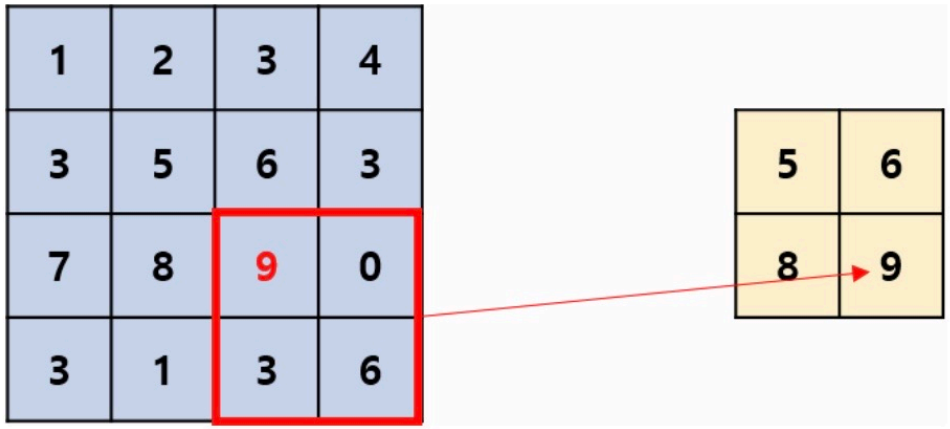


[2D Convolution]

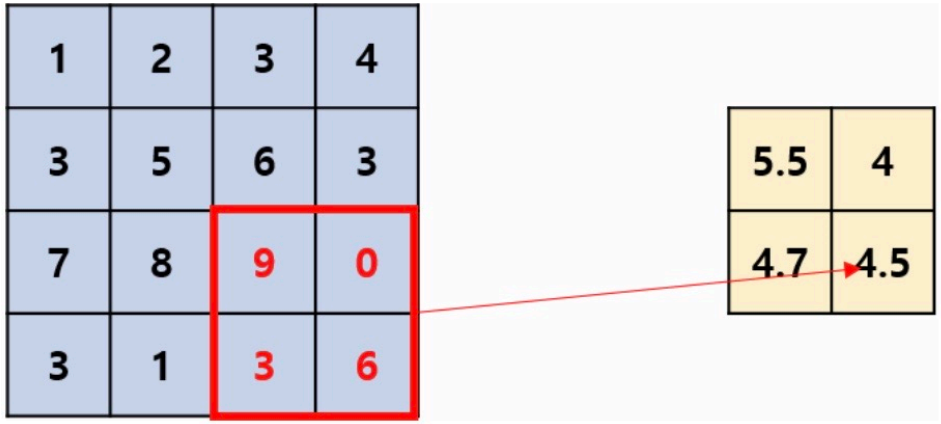


[3D Convolution]

Pooling Layer: 데이터의 공간적 특성을 유지하면서, 크기를 줄여 줌 -> 가중치를 줄여 과적합(overfitting) 해결 도움

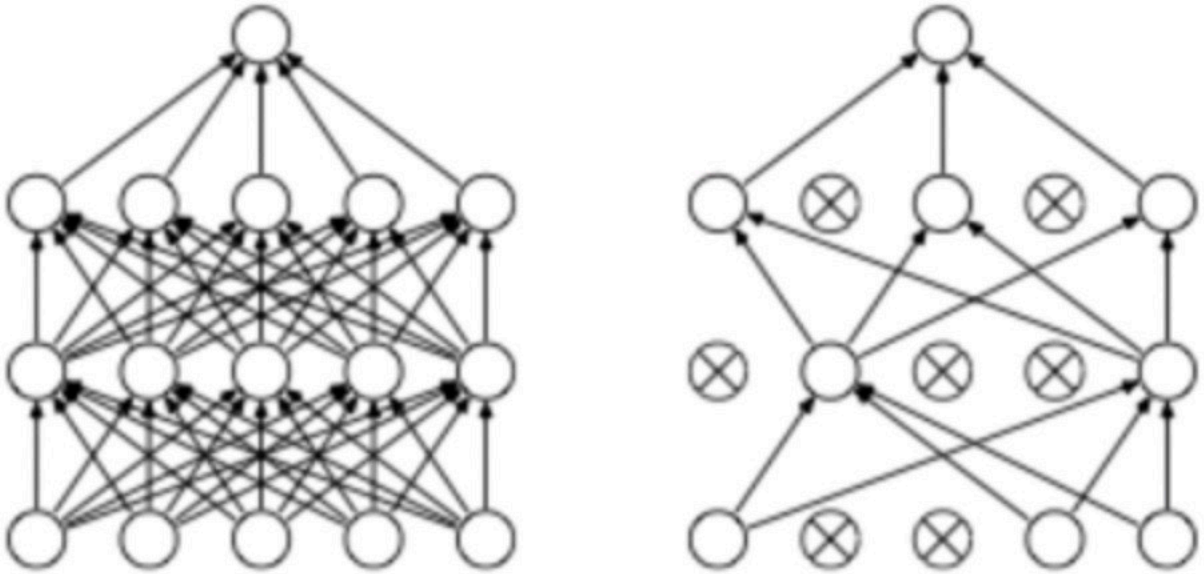


Max Pooling



Average Pooling

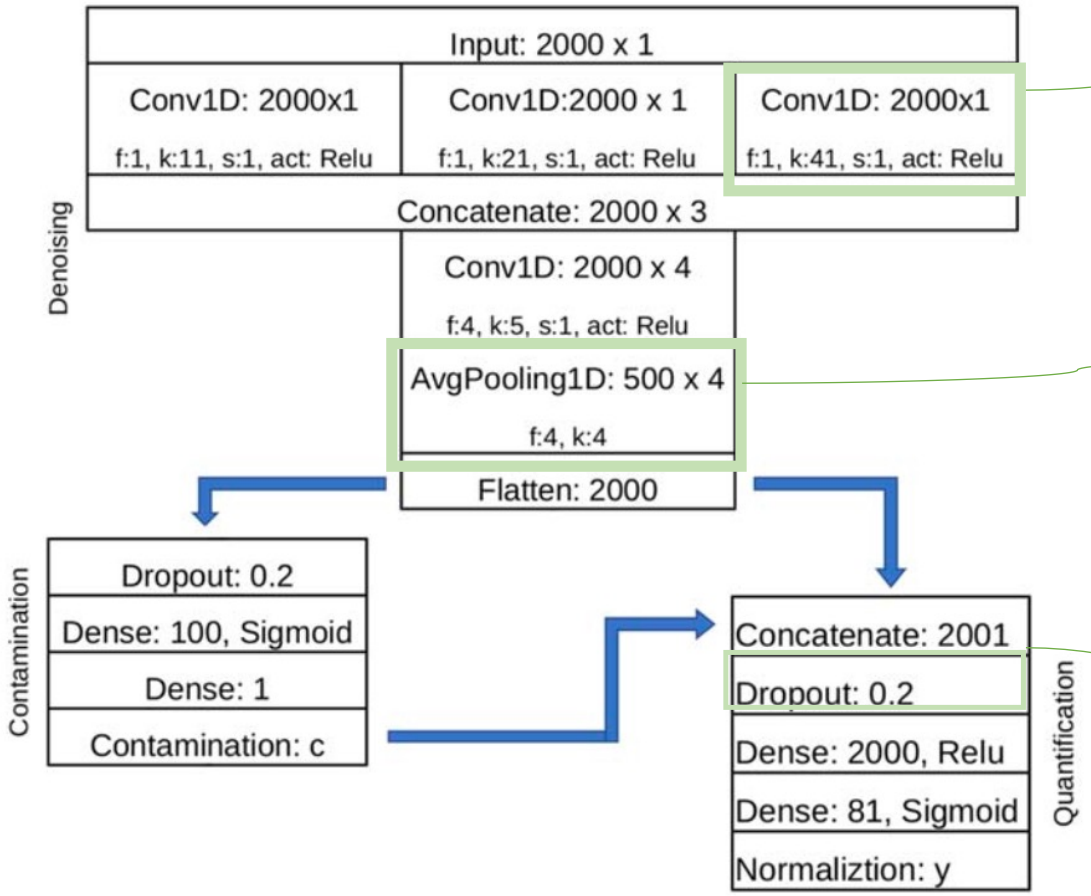
Drop-Out Layer: L1(Lasso), L2(Ridge)와 함께 과대적합을 방지하기 위해 사용



(a) Standard Neural Net (b) After applying dropout.

예측 시에는 모든 뉴런이 참여하지만, 훈련 시 hidden layer unit 일부가 랜덤하게 사용되지 않음

특정 hidden layer unit에 의존할 수 없도록 만들어서 일반적이고 안정적으로 학습

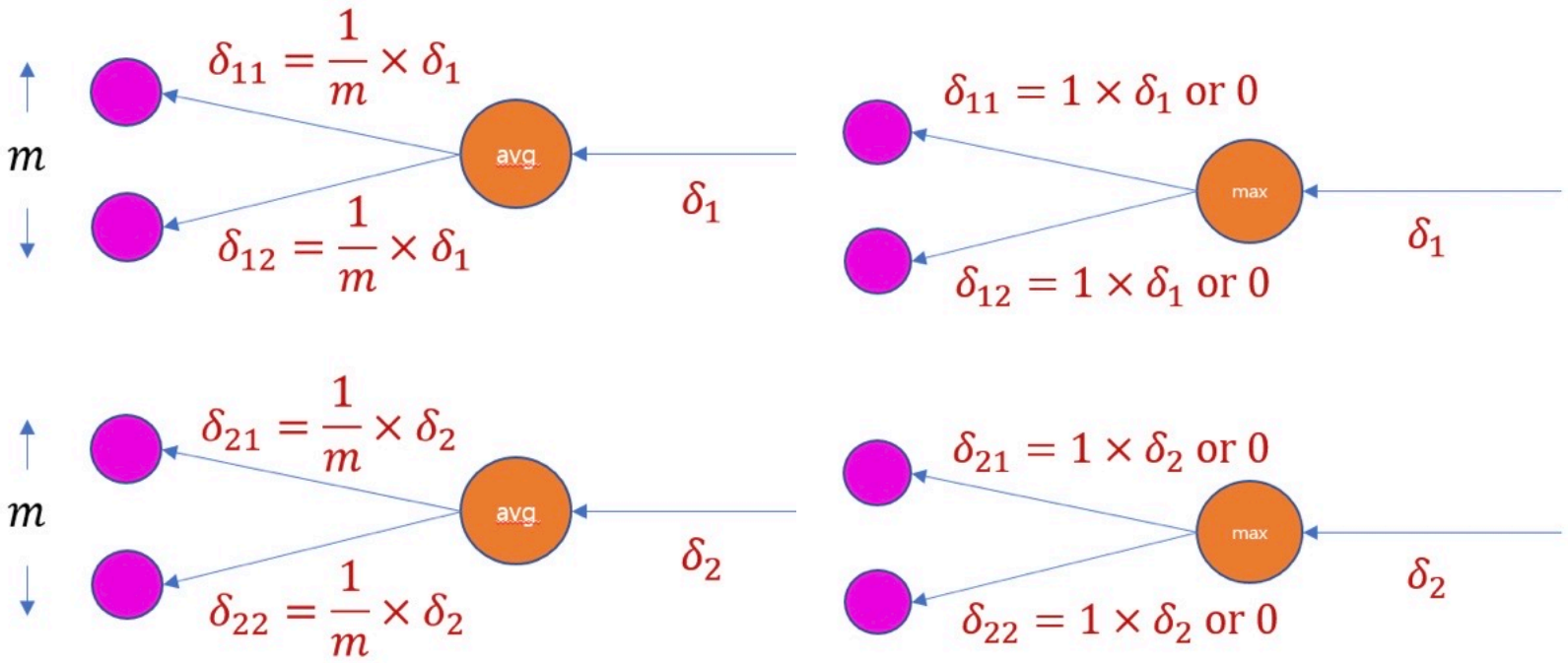


1D Convolution Layer
Filter 1개 ; kernel size: 41
Stride: 1
Padding: 1(출력 동일하기 때문에)
Activation function: ReLU

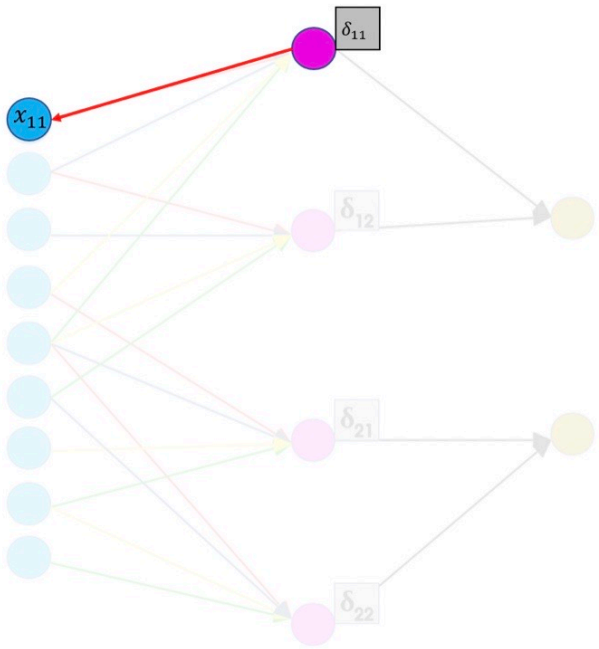
Average Pooling layer
Filter 4: ; kernel size: 4
Stride: 4
(500x4x4)/4=2000 이기 때문에

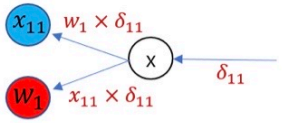
DropOut Layer
p_drop=0.2
p_keep=(1-0.2)=0.8

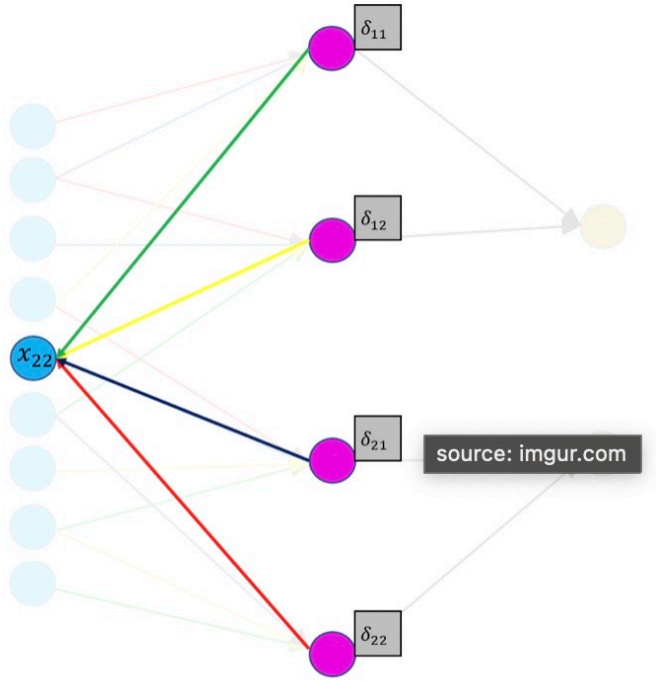
CNN Back propagation

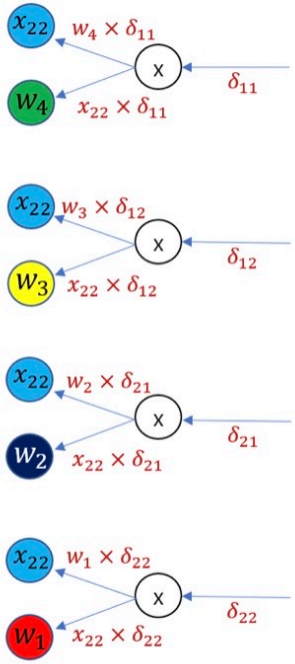


CNN Back propagation

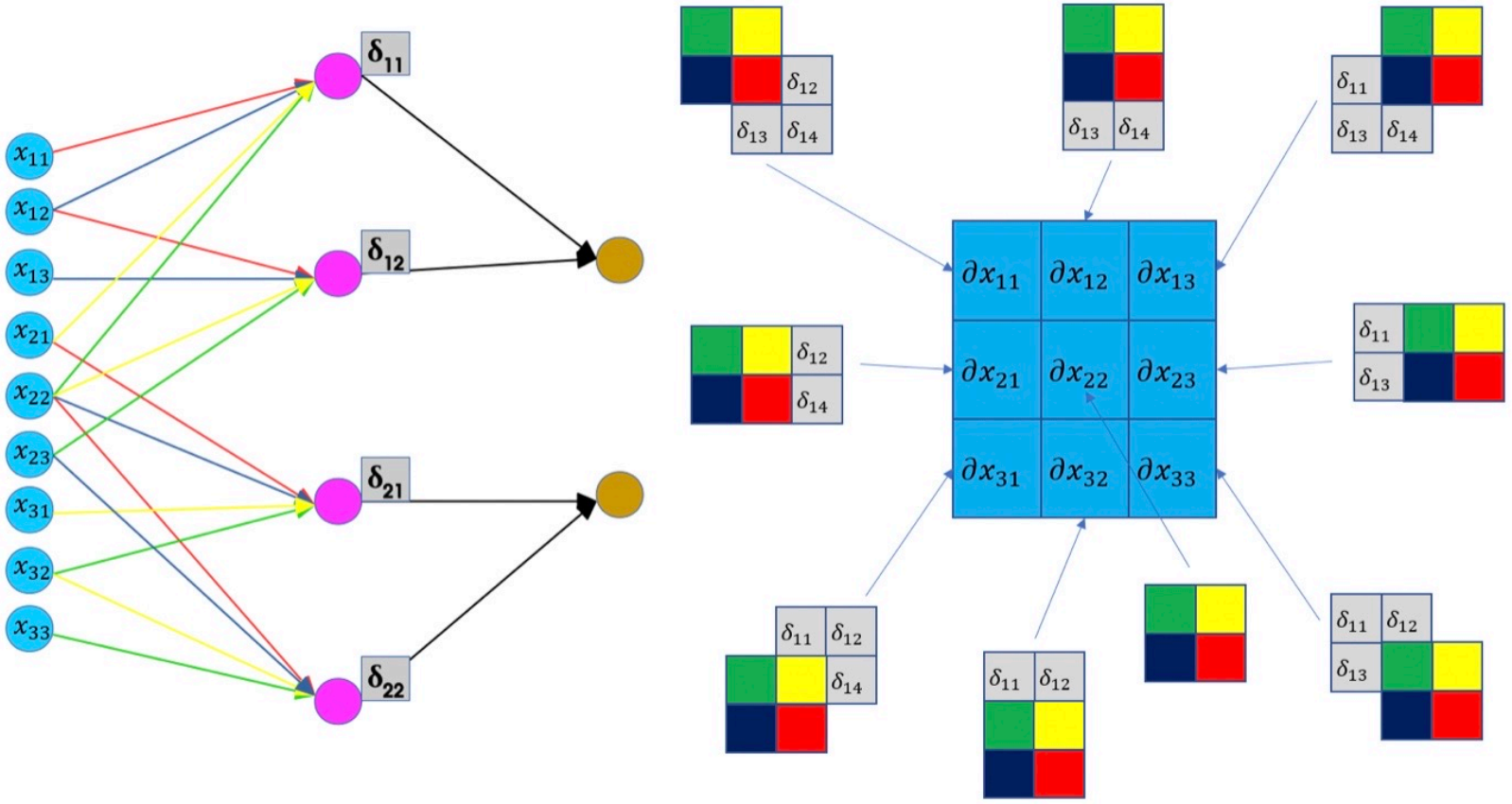








CNN Back propagation



Thank you!