인공지능을 통한 숫자인식, 이미지 인식

강의 : 박성주

01 Deep Learning Remind

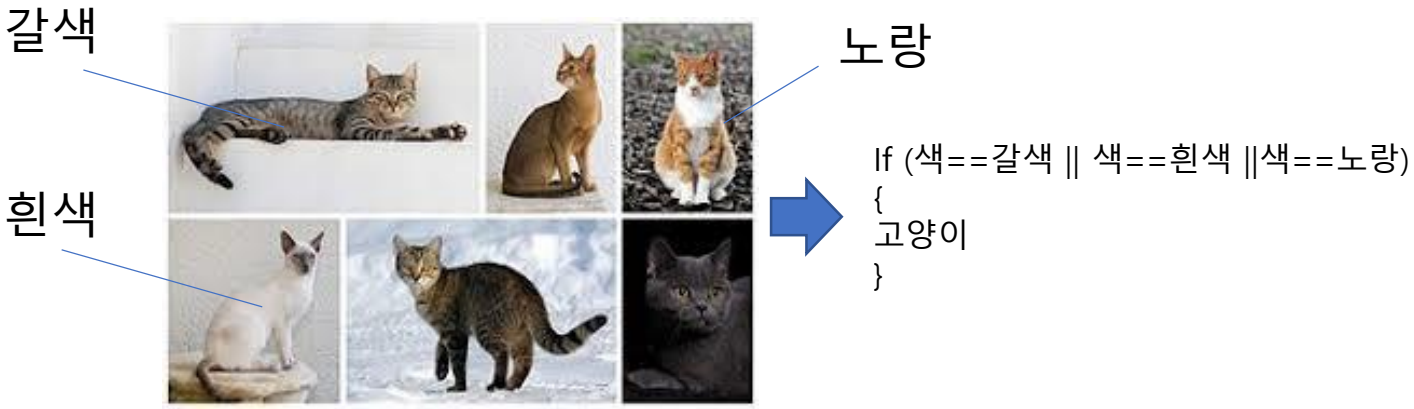
Code Vision

Hand-Craft 정의

- 모든 경우에 대해 프로그램으로 정의를 해 주어야함

Machine Learning 정의

- **“Field of study that gives computers the ability to learn without being explicitly programmed”**
- **이런 경우에는 이렇게 동작하고, 저런 경우에는 저렇게 동작하라는 등, 가능한 모든 경우의 수를 프로그래머가 정의해주지 않음**
- **그래도 데이터 학습을 통해 최적의 판단이나 예측을 가능하게 해주는 것**



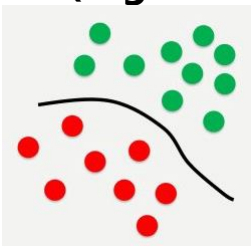
01 Deep Learning Remind

Code Vision

Machine Learning 분류

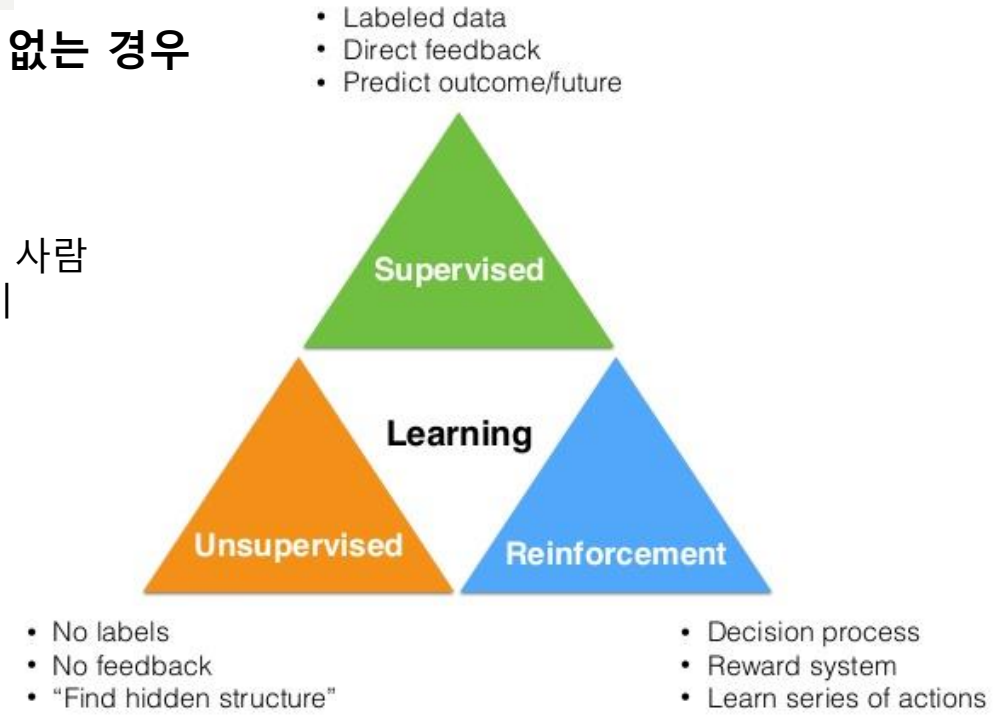
- 지도학습(supervised learning)이란 어떤 결과가 나와야하는지 사전지식을 갖고 있는 경우, 해당 출력이 나오도록 하는 규칙을 찾아낸다.
  => 입력과 출력이 데이터에 쌍으로 존재, 회귀(regression)에 해당

↔ 고양이



- 비지도 학습(Unsuperviesd)이란 입력은 있지만 출력이 없는 경우
  => 입력만 있음, 분류(clustering)에 해당

→ 가장 비슷한 사람들 끼리 묶기



01 Deep Learning Remind

Machine Learning 분류

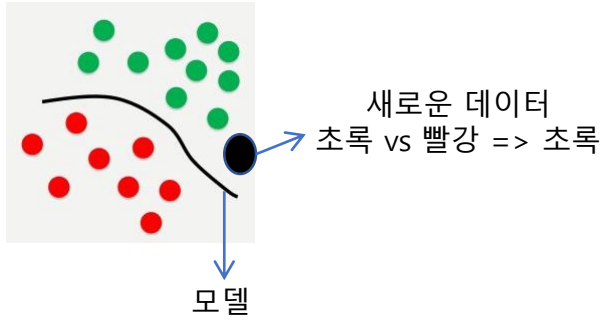
- 강화학습(reinforcement learning)는 로봇의 학습 등에 사용할 수 있으며, 자신과 환경과의 상호 관계에 따라 자신의 행동을 개선해 나가는 학습법을 말한다.

01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning) 개요

- Supervisor의 사전적 의미 : 작업 현장 통제하는 '감독관'
- 어떤 결과가 나올지 알고 있는, 이미 lebeling이 되어 출력과의 관계를 이용해서 데이터 들을 해석할 수 있는 모델을 만들고, 그것을 바탕으로 새로운 데이터를 추정 (predict)



01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning) 단계

- **1. 학습에 사용할 훈련 데이터를 정한다. => 잘 선정 해야함, 균형을 잘 맞추어야 함**
- **2. 훈련 데이터를 모은다. => 데이터가 적으면 학습이 힘듬 (무엇보다 딥러닝은 빅데이터 필요)**
- **3. 입력의 특징(feature)를 어떻게 표현 할 것인지 결정 => 일반적으로 백터 형태로 표현을 함**

데이터	라벨링 백터
	1 0
	0 1

01 Deep Learning Remind

Code Vision

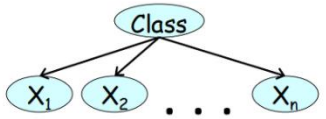
지도 학습 (Supervised Learning) 단계

- **4. 학습 알고리즘을 정한다.**

Artificial Neural Network
⇒ 딥러닝 => 인간의 뇌를 Motive로 하였기 때문에 가장 인간의 수준과 같은 지능을 낼 수 있음

Bayesian statistics

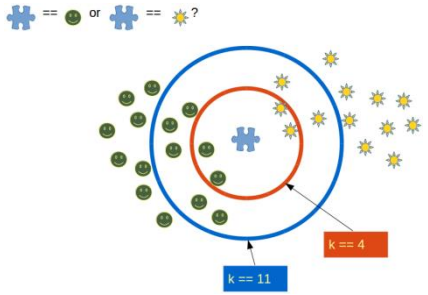
Naïve Bayes Classifier



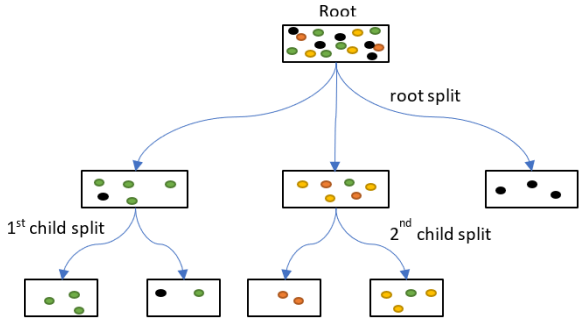
$$\frac{P(C=c^1 \mid x_1,\ldots,x_n)}{P(C=c^2 \mid x_1,\ldots,x_n)} = \frac{P(C=c^1)}{P(C=c^2)} \prod_{i=1}^{n} \frac{P(x_i \mid C=c^1)}{P(x_i \mid C=c^2)}$$

odds ratios

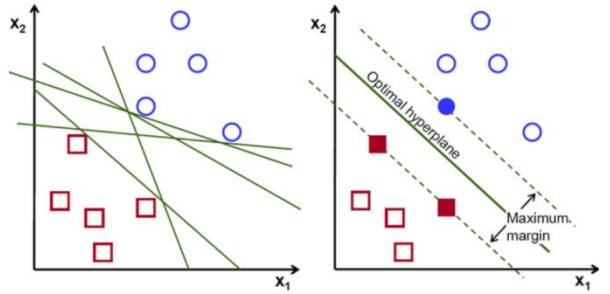
Nearest neighbor algorithm



Decision tree



Support vector machine



01 Deep Learning Remind

지도 학습 (Supervised Learning) 단계

- 5. 훈련 데이터를 이용해 학습 알고리즘을 실행한다.
  => 데이터를 어떤 방식으로 넣을 것인지 (순서대로?, 랜덤순서로?)
  => 데이터 전처리 과정

- 6. 만들어진 모델의 정확도를 평가
  => 새로운 데이터가 들어왔을 때 얼마나 정확히 맞추는지
  => Training dataset으로 학습하고, Test dataset으로 테스트 해야함

지도 학습 (Supervised Learning)

회귀 (regression) – cont.

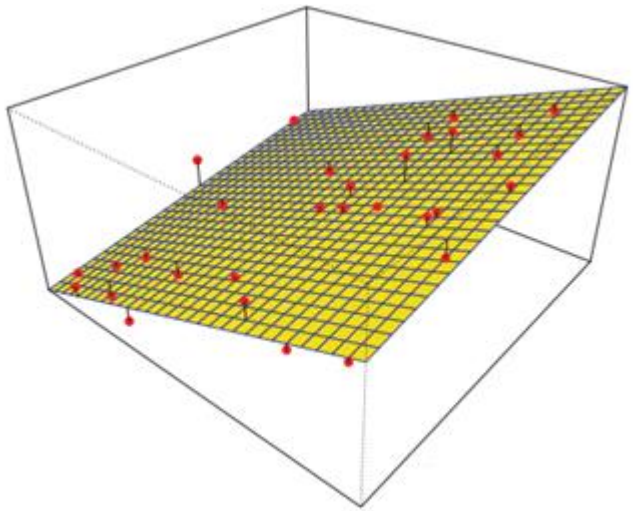
성능

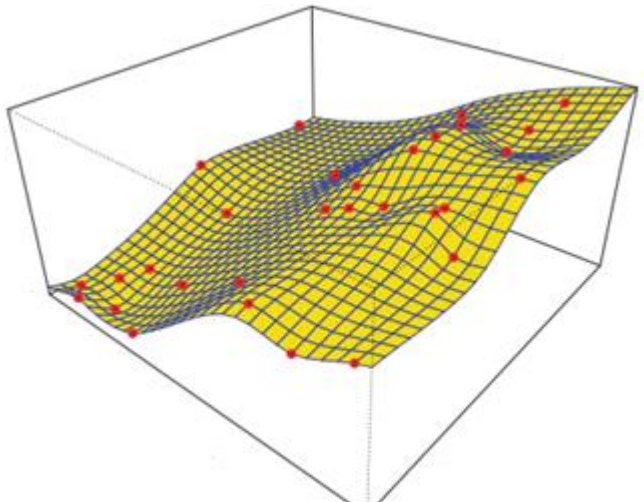
오차 : 예측값과 실제값의 차이

테스트 데이터들에 대한 (예측값 – 실제값)^{2}의 평균 또는 평균의 제곱근

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

모델의 종류(**함수의 종류**)에 영향을 받음





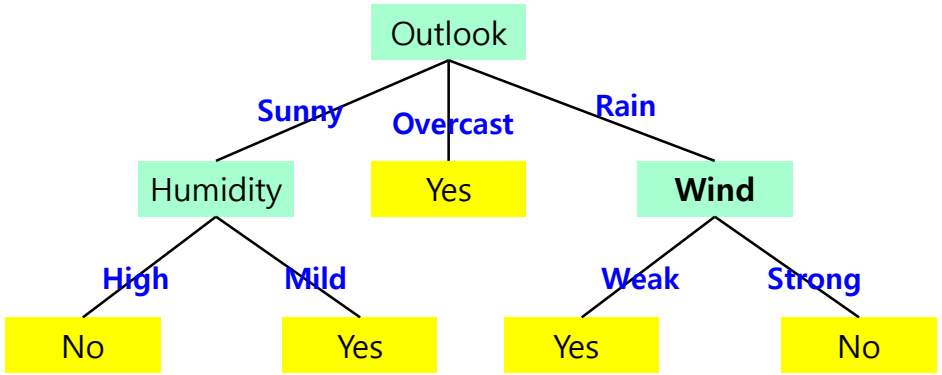
01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning)

❖ 결정트리(decision tree)

- **트리 형태**로 의사결정 **지식**을 표현한 것
  - 내부 노드(internal node) : 비교 속성
  - 간선(edge) : 속성 값
  - 단말 노드(terminal node) : 부류(class), 대표값



Day 날짜	Outlook 조망	Temperature 기온	Humidity 습도	Wind 바람	PlayTennis 테니스 여부
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

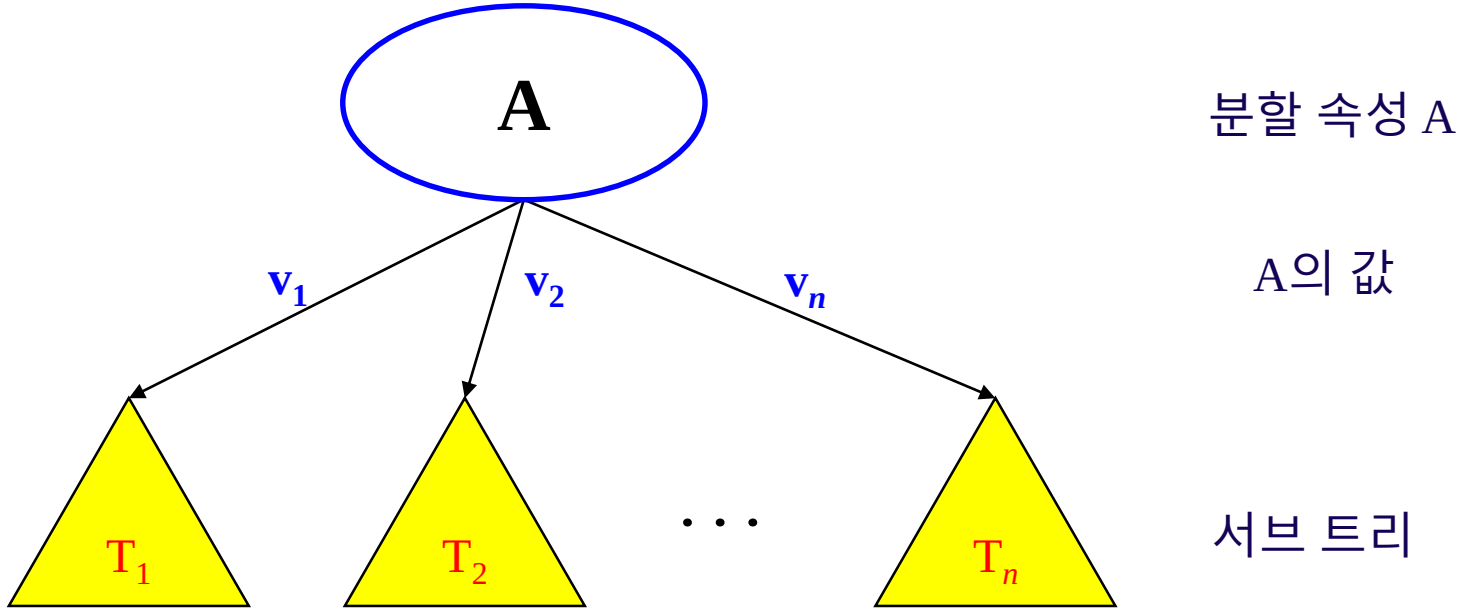
IF Outlook = Sunny **AND** Humidity = High **THEN** Answer = No

Outlook 조망	**Temperature** 기온	**Humidity** 습도	**Wind** 바람	**PlayTennis** 테니스 여부
Sunny	Hot	Mild	Weak	?
Rain	Hot	High	Weak	?

지도 학습 (Supervised Learning)

❖ 결정 트리 (decision tree) 알고리즘

- 모든 데이터를 포함한 **하나의 노드**로 구성된 **트리**에서 시작
- **반복적인 노드 분할** 과정
  1. **분할 속성**(spliting attribute)을 **선택**
  2. 속성값에 따라 **서브트리**(subtree)를 **생성**
  3. **데이터**를 속성값에 따라 **분배**



01 Deep Learning Remind

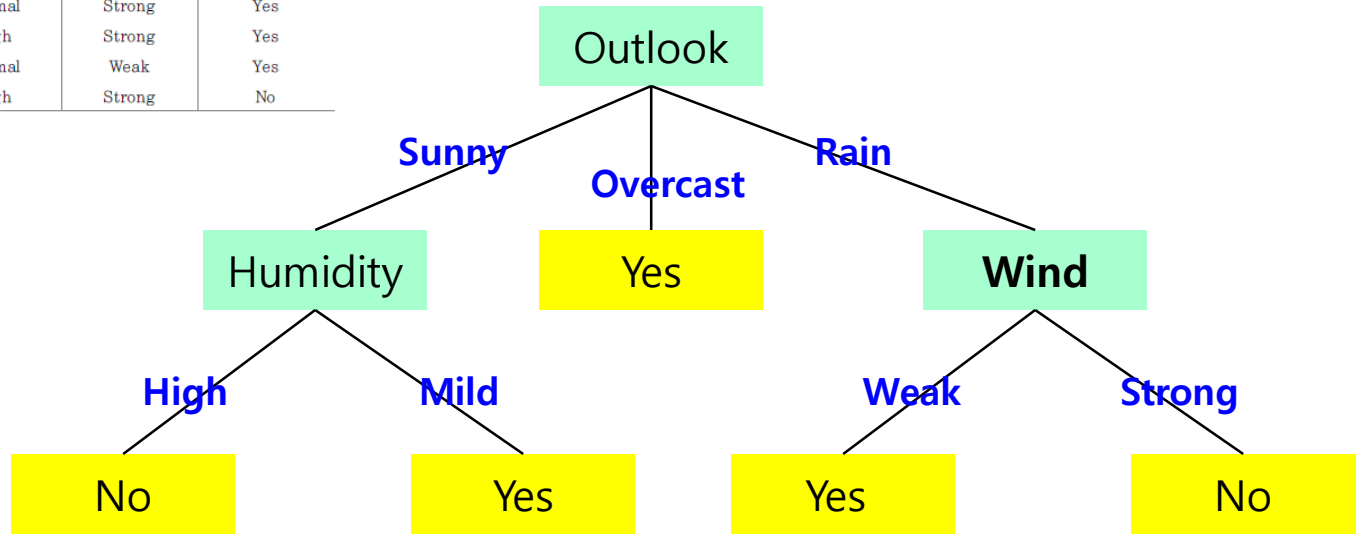
Code Vision

지도 학습 (Supervised Learning)

❖ 결정 트리 (decision tree)

- 간단한 트리

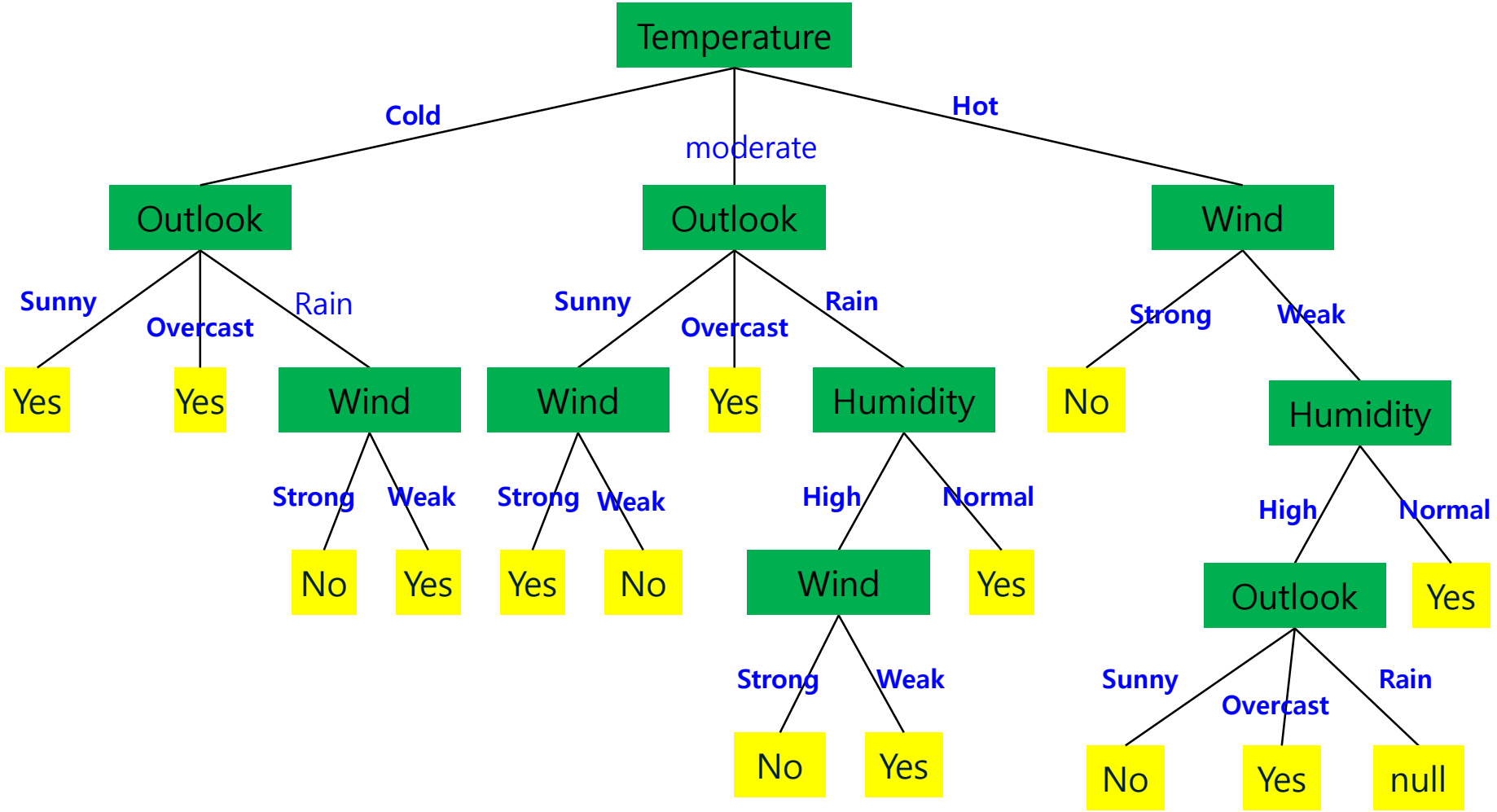
Day 날짜	Outlook 조망	Temperature 기온	Humidity 습도	Wind 바람	PlayTennis 테니스 여부
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



지도 학습 (Supervised Learning)

❖ 결정 트리 (decision tree)

- 복잡한 트리



지도 학습 (Supervised Learning)

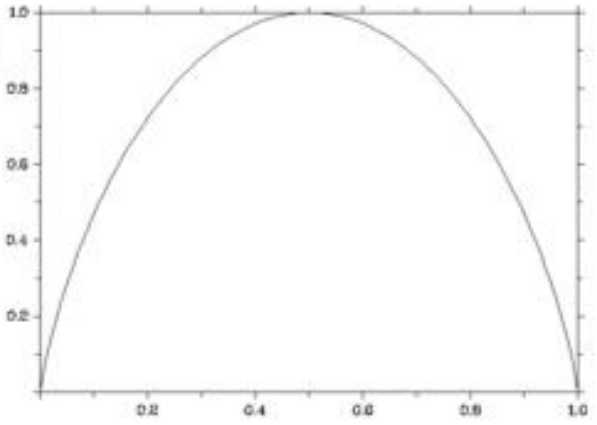
❖ 분할 속성(splitting attribute) 결정

- 어떤 속성을 선택하는 것이 효율적인가
  - **분할한 결과**가 가능하면 동질적인(pure) 것으로 만드는 속성 선택

- **엔트로피**(Entropy)
  - 동질적인 정도 측정 가능 척도
  - 원래 정보량(amount of information) 측정 목적의 척도

$$I = -\sum_{c} p(c) \log_2 p(c)$$

  - $p(c)$: 부류 c에 속하는 것의 비율



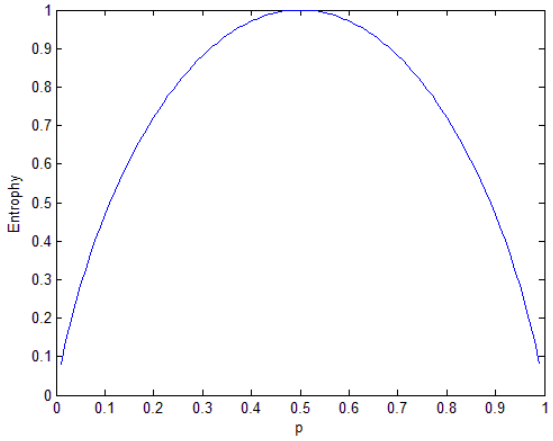
  - 2개 부류가 있는 경우 엔트로피

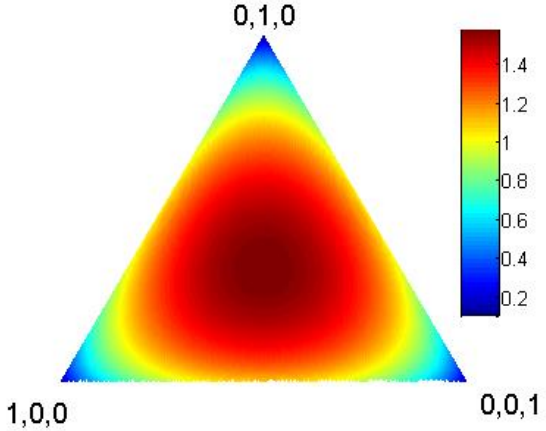
$S = \log_2 2^N = N = 2$

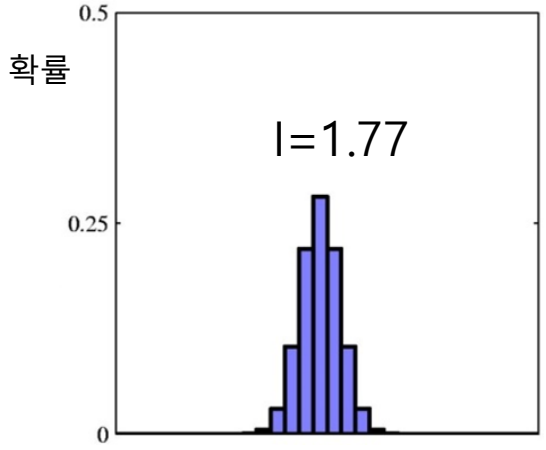
지도 학습 (Supervised Learning)

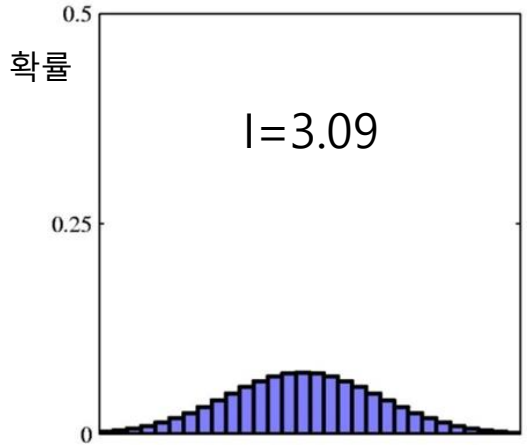
엔트로피의 특성

섞인 정도가 클 수록 큰 값





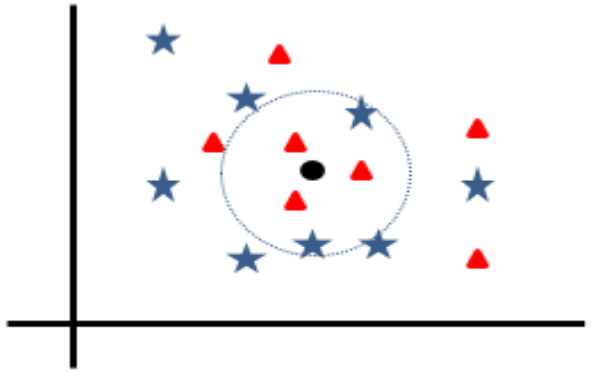




지도 학습 (Supervised Learning)

k-근접이웃 (k-nearest neighbor, KNN) 알고리즘

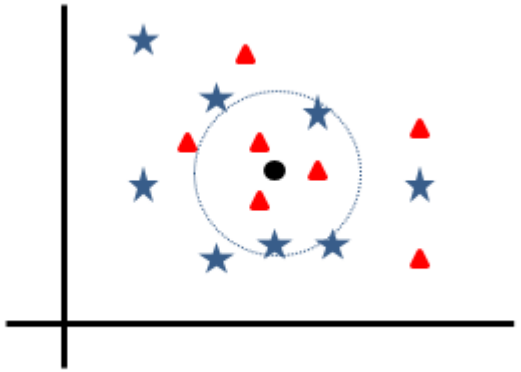
(입력, 결과)가 있는 데이터들이 주어진 상황에서,
새로운 입력에 대한 결과를 추정할 때
결과를 아는 **최근접**한 **k개**의 **데이터**에 대한 결과정보를 이용하는 방법



질의(query)와 데이터간의 **거리 계산**
효율적으로 **근접이웃 탐색**
근접 이웃 k개로 부터 **결과**를 **추정**

지도 학습 (Supervised Learning)

k-nearest neighbor (KNN) 알고리즘 – cont.



데이터간의 **거리 계산**

수치 데이터의 경우

유클리디언 거리(Euclidian distance)

$$X = (x_1, x_2, \ldots, x_n) \quad Y = (y_1, y_2, \ldots, y_n)$$

$$d = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2$$

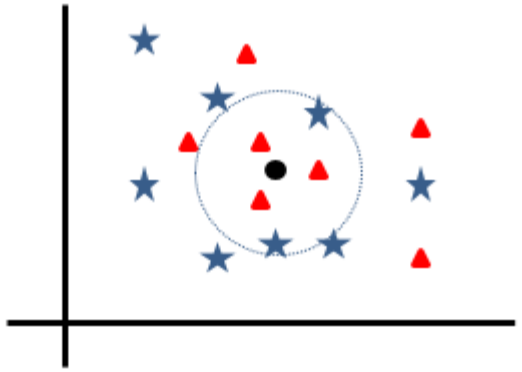
응용분야의 특성에 맞춰 개발

범주형 데이터가 포함된 경우 (카테고리 타입의 데이터라고 보면됨)

응용분야의 특성에 맞춰 개발

지도 학습 (Supervised Learning)

k-nearest neighbor (KNN) 알고리즘 – cont.



최근접 k개로 부터 **결과**를 **추정**하는 방법

분류

출력이 **범주형** 값

다수결 투표(majority voting) : 개수가 많은 범주 선택

회귀분석

출력이 수치형 값

평균 : 최근접 k개의 평균값

가중합(weighted sum) : 거리에 반비례하는 가중치 사용

지도 학습 (Supervised Learning)

k-nearest neighbor (KNN) 알고리즘 – cont.

특징

학습단계에서는 실질적인 학습이 일어나지 않고 **데이터**만 **저장**

학습데이터가 크면 **메모리 문제**

게으른 학습(**lazy learning**)

새로운 데이터가 주어지면 저장된 데이터를 이용하여 학습

시간이 많이 걸릴 수 있음

지도 학습 (Supervised Learning)

❖ 단순 베이즈 분류기(naïve Bayes classifier)

- **부류**(class) 결정지식을 **조건부 확률**(conditional probability)로 결정
  - $P(c|x_1, x_2, \cdots . x_n)$: 속성값에 대한 부류의 조건부 확률
    - c : 부류
    - x_i : 속성값
- **베이즈 정리** (Bayes theorem)

사후 확률 / 가능도 / 사전 확률

$$P(c|x_1, x_2, \cdots . x_n) = \frac{P(x_1, x_2, \cdots, x_n|c)P(c)}{P(x_1, x_2, \cdots, x_n)}$$

증거

- **가능도**(likelihood)의 **조건부 독립**(conditional independence) 가정

$$P(x_1, x_2, \cdots . x_n|c) = P(x_1|c)\ P(x_2|c) \cdots P(x_n|c)$$

$$P(c|x_1, x_2, \cdots, x_n) = \frac{P(x_1|c)\ P(x_2|c) \cdots P(x_n|c)P(c)}{P(x_1, x_2, \cdots, x_n)}$$

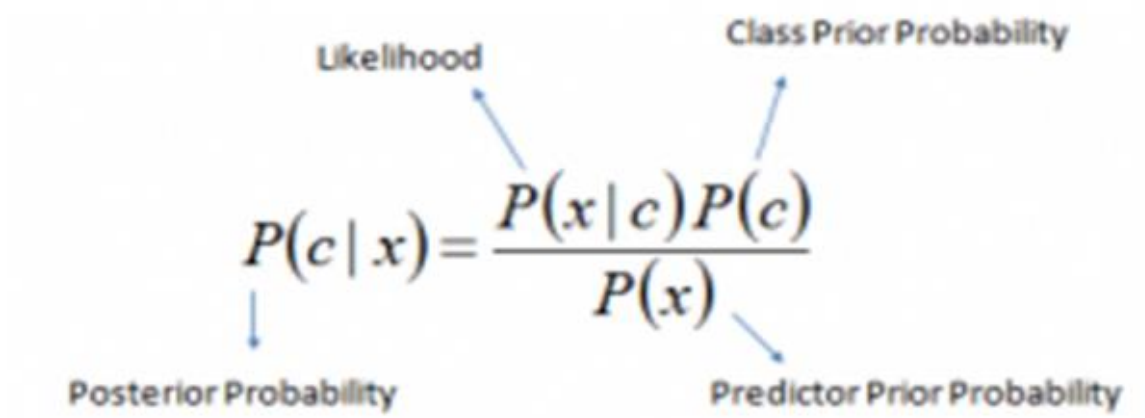
01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning)

- 단순 베이즈 분류기 (Naïve Bayes Classifier)
- => Naïve란 순진한 (경험이 부족하다는 의미)
- => 즉 A, B, C라는 사건이 있을 경우 A, B, C의 서로 관계를 모르기 때문에 독립으로 취급하자! = Naive
- => A, B, C 특징 들간의 연관성을 너무 고려해버리면 복잡해지는 경향이 있어, 단순화 시켜 쉽고 빠르게 판단을 내리는데 유용하다.

- Bayes 법칙
- $P(C|x)$ ⇒ 특정 개체 x 가 특정 그룹 c에 속할 사후 확률
- $P(C)$ ⇒ 특정 그룹 c가 발생할 빈도, 사전 고유 확률
- $P(x|C)$ ⇒ 특정 그룹 c인 경우 특정개체 x 가 거기 속할 확률



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

01 Deep Learning Remind

지도 학습 (Supervised Learning)

- **단순 베이즈 분류기 (Naïve Bayes Classifier) 예시**

- **Drew는 남자이름 ? 여자이름 ?**

사전 정보

Officer Drew

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

What is the probability of being called "*drew*" given that you are a male?

What is the probability of being a male?

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

What is the probability of being named "*drew*"?

01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning)

- 단순 베이즈 분류기 (Naïve Bayes Classifier) 예시

- Drew는 남자이름 ? 여자이름 ?

사전 정보

Officer Drew

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

P(male | drew) = 1/3 * 3/8 = 0.125

P(female | drew) = 2/5 * 5/8 = 0.25

What is the probability of being called "*drew*" given that you are a male?

What is the probability of being a male?

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

What is the probability of being named "*drew*"?

- => 눈으로 확인해도 되지만 지금과 달리 클래스가 3개이상만 되어도 Naïve Bayes Classifier를 쓰는 것이 훨씬 간단함

01 Deep Learning Remind

Code Vision

지도 학습 (Supervised Learning)

- 단순 베이즈 분류기 (Naïve Bayes Classifier) 예시

- C = {Cat, Dog, Bird}
- X={Claudia, Alberto, Drew, Karin}일 때
- Drew라는 이름을 가졌을 때, 어느 종이 선택될 확률이 많은지 계산해서 메일로 보내주세요
- 각 3개확률이 계산된 값이 몇인지도 적어주시기 바랍니다.

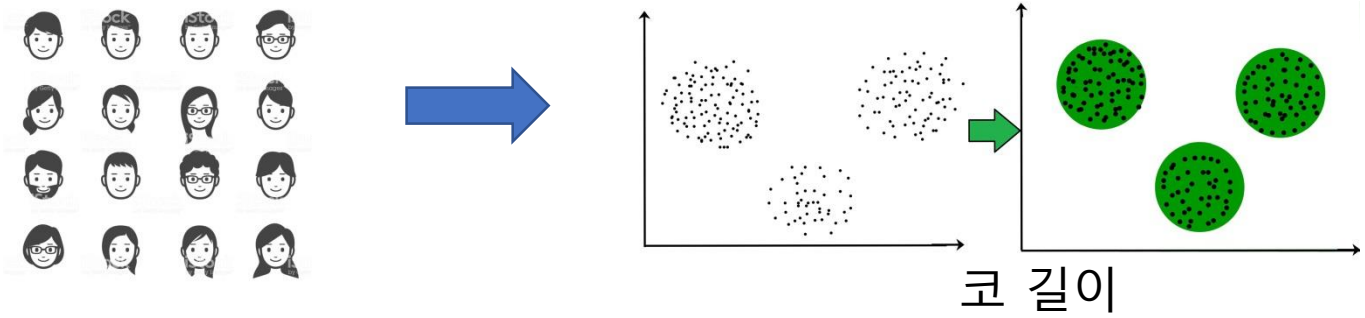
- 이메일 : stray7@naver.com
- 이메일로

Name	Animal
Claudia	Cat
Alberto	Cat
Drew	Dog
Drew	Bird
Claudia	Dog
Drew	Bird
Alberto	Cat
Karin	Bird
Alberto	Bird
Drew	Dog
Claudia	Dog
Drew	Cat
Drew	Cat

비지도 학습 (Unsupervised Learning) 개요

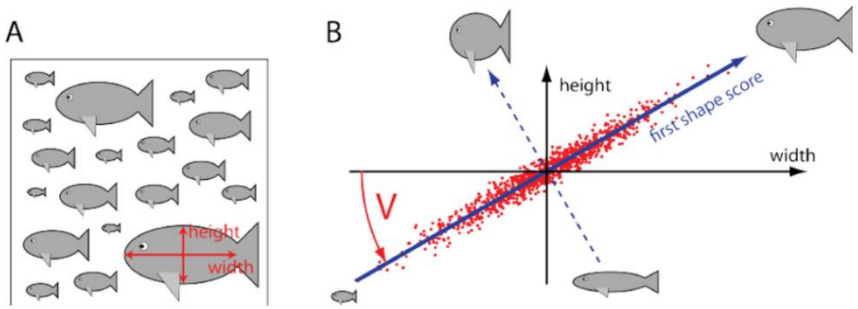
- 지도학습의 경우는 정답이 정해져 있기 때문에 error function을 구할 수 있음
- 비지도는 x
- 2가지 대표적인 예

- **Clustering**



- **Dimensionality reduction**

=> 인식에 불필요한 정보들은 버리고, 가장 두드러지는 특징만을 취함



01 Deep Learning Remind

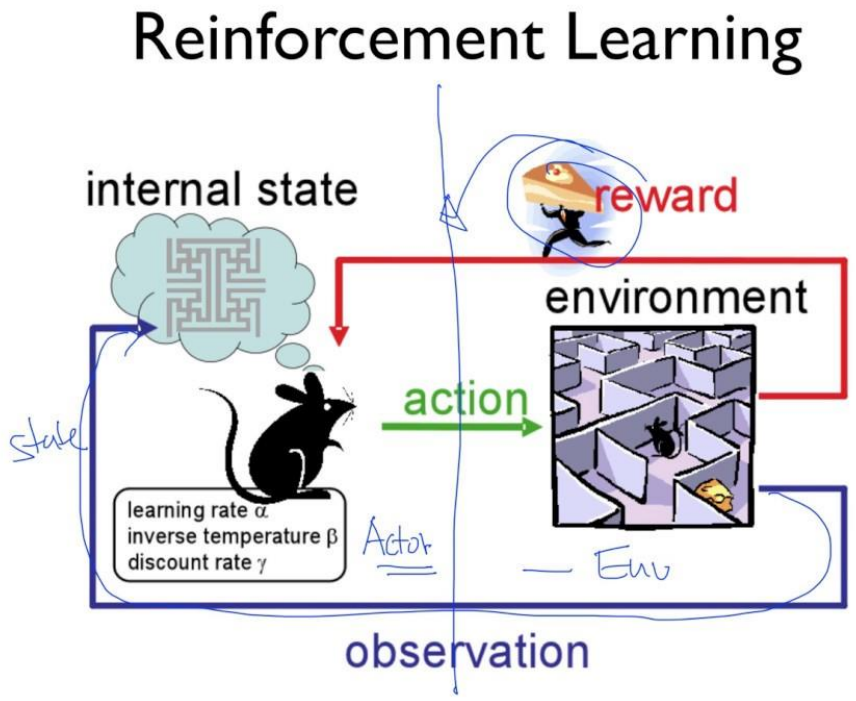
Code Vision

강화학습의 (Reinforcement Learning) 개요

- 강화학습이란, 주어진 어떤 상황에서 보상을 최대화 할 수 있는 행동에 대해 학습하는 것

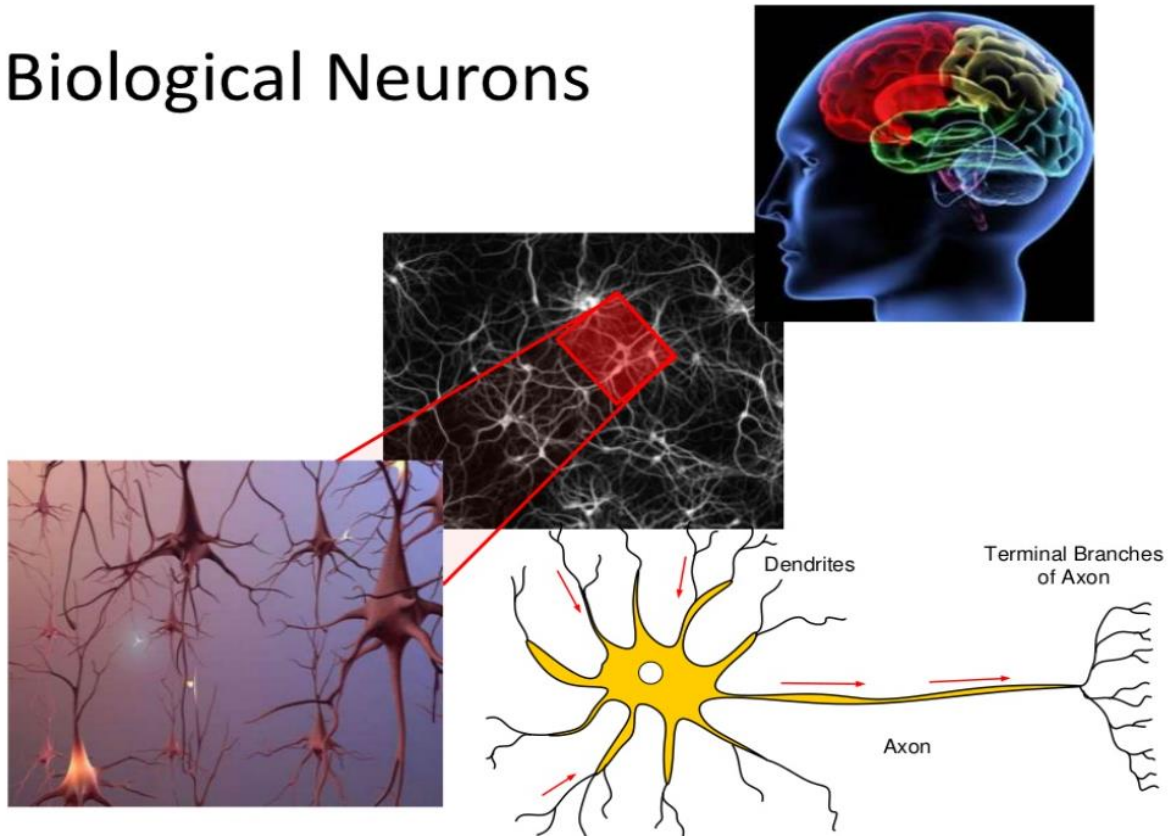
- 학습의 주체가 상황에 가장 적합한 행동을 찾기까지는 수많은 시행착오가 필요

- 복잡한 상황에서는 현재 선택한 행동이 미래의 순차적인 보상에 영향



01 Deep Learning Remind

지도학습의 신경망(Neural Network)알고리즘 => 딥러닝 이라고 함

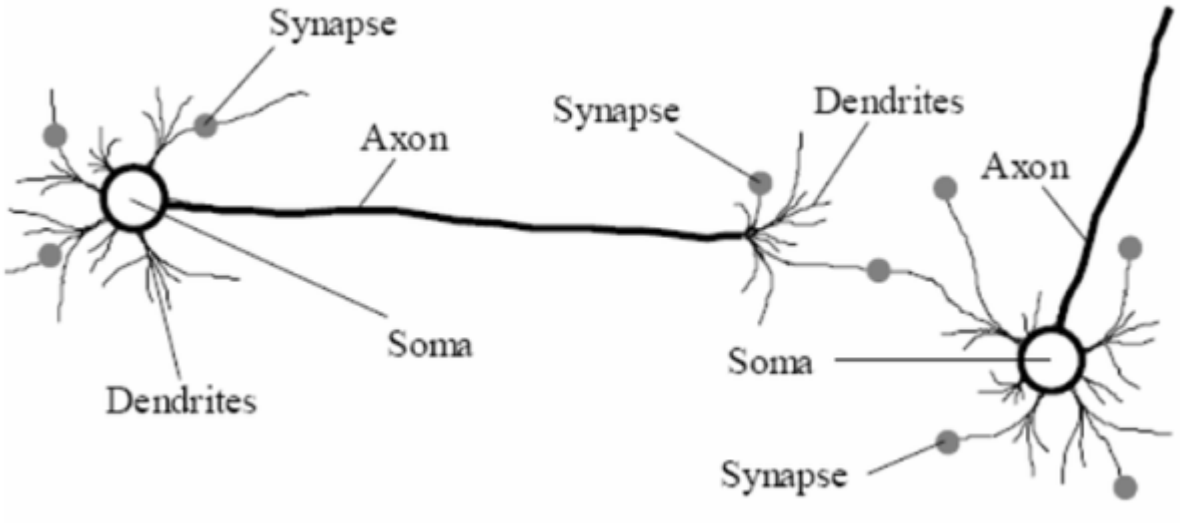


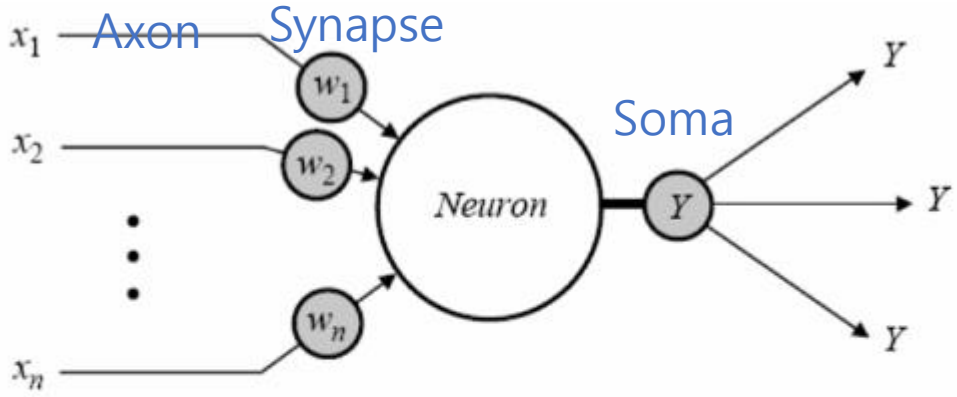
01 Deep Learning Remind

지도학습의 신경망(Neural Network)알고리즘 => 딥러닝 이라고 함

- 축삭돌기 (Axon) : 가늘고 길게 뻗어있으며, 다른 뉴런으로 신호를 전달하는 기능
- 신경세포체 (Soma) : 신경 세포의 핵을 담당하는 부분으로 여러 뉴런으로 전달되는 외부 자극에 대한 판정을 하여 다른 뉴런으로 신호를 전달할 것인지 결정함

  => **Activation Function**
- 시냅스 (Synapse) : 얇은 막의 형태이며, 다른 뉴런으로부터 어느정도의 세기로 전달할 것인지 결정 =>**Weigt**





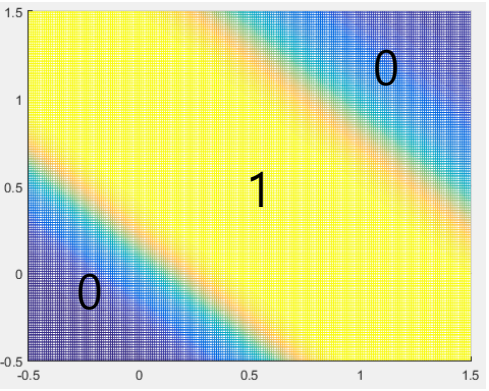
$$X = \sum_{i=1}^{n} x_i w_i \qquad Y = \begin{cases} +1, & \text{if } X \geq \theta \\ -1, & \text{if } X < \theta \end{cases}$$

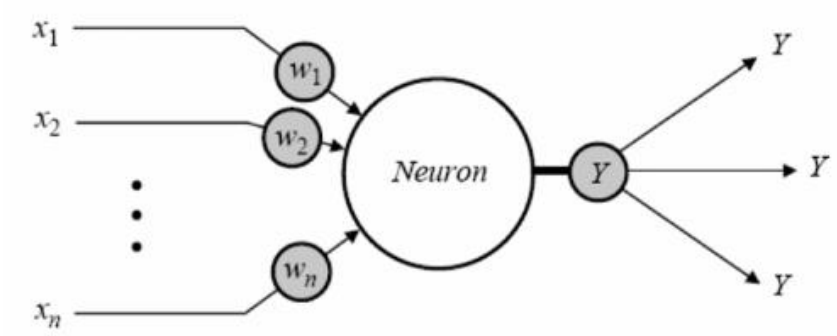
01 Deep Learning Remind

Code Vision

지도학습의 신경망(Neural Network)의 기본 구조

- Forward Propagation(전파)
- 입력(x1,x2...)에 가중치(w1,w2..)를 곱하여
  Activation Function(Y)에 전달된 후
  최종 출력이 결정됨

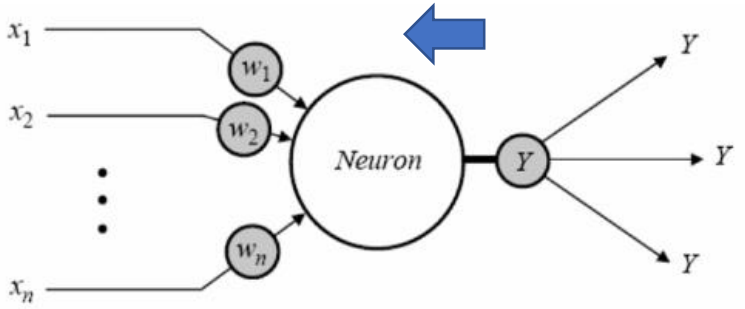




$$X = \sum_{i=1}^{n} x_i w_i \qquad Y = \begin{cases} +1, & \text{if } X \geq \theta \\ -1, & \text{if } X < \theta \end{cases}$$

W 수정 = 딥러닝의 학습과정!!

- Back Propagation(역전파)
- 위의 그림에 따르면 x1=1, x2=1일 때는
  출력 값이 1로 나와야 하는데 0이 나올 경우
  역전파라는 학습을 통해 Weight를 조정 해야함



$$X = \sum_{i=1}^{n} x_i w_i \qquad Y = \begin{cases} +1, & \text{if } X \geq \theta \\ -1, & \text{if } X < \theta \end{cases}$$

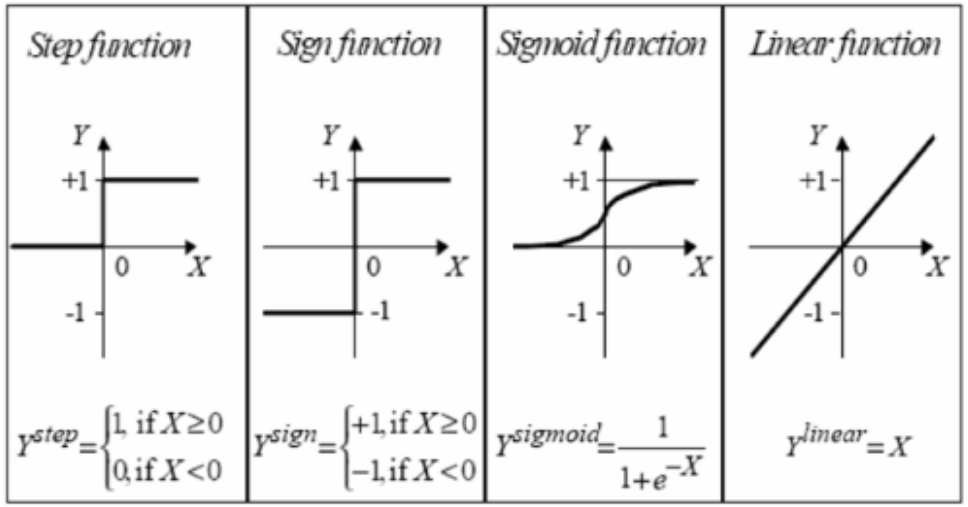
01 Deep Learning Remind

Code Vision

신경망의 역사

- 1949년 심리학자인 Donald Hebb에 의해 처음 발표됨
  => 생물학적인 신경망에서 학습이 이루어지면 특정 입력으로 들어오는 신호 자극에 잘 반응할 수 있도록 시냅스들의 세기가 결정이 된다는 사실에 주목 = Weight!

- Preceptron의 개념
  => 1957년 Rosenbalt는 "Perceptron"이라는 개념을 발표
  => 발표당시 뉴런의 활성함수(Activation Function)로 기본적인 "Step Function"을 최초로 사용
  => 입력의 중요도에 따라 출력이 결정되는 수학적 모델로서 의미가 있다



01 Deep Learning Remind

Code Vision

신경망의 역사

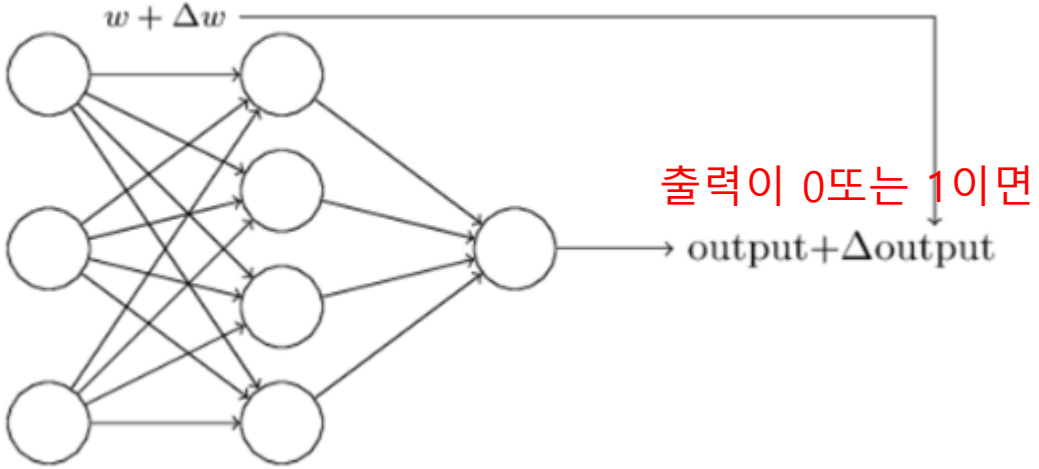
- Perceptron의 한계

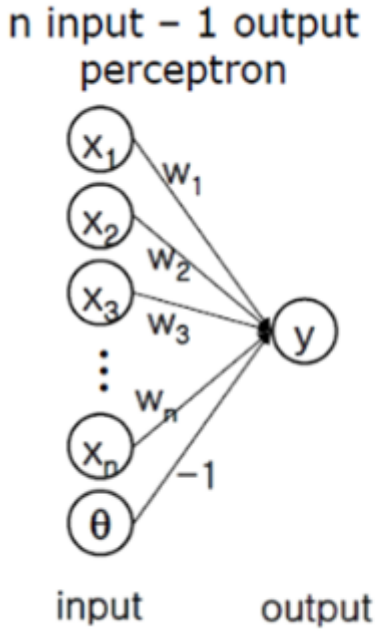
  => 신경망의 특성을 잘 반영한 것 같지만 한계가 있음

  => 출력이 0과 1처럼 극단적인 결과만 도출할 수 있음

  => 2 layer로만 구성되어 있어, 아주 단순한 결과만 도출 가능

Weight도 매우 한정된 값만 도출됨





$y=STEP(\Sigma x_i w_i-\theta)$

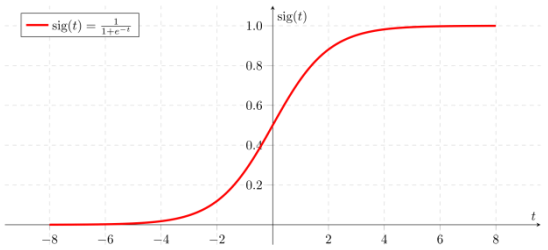
01 Deep Learning Remind

Code Vision

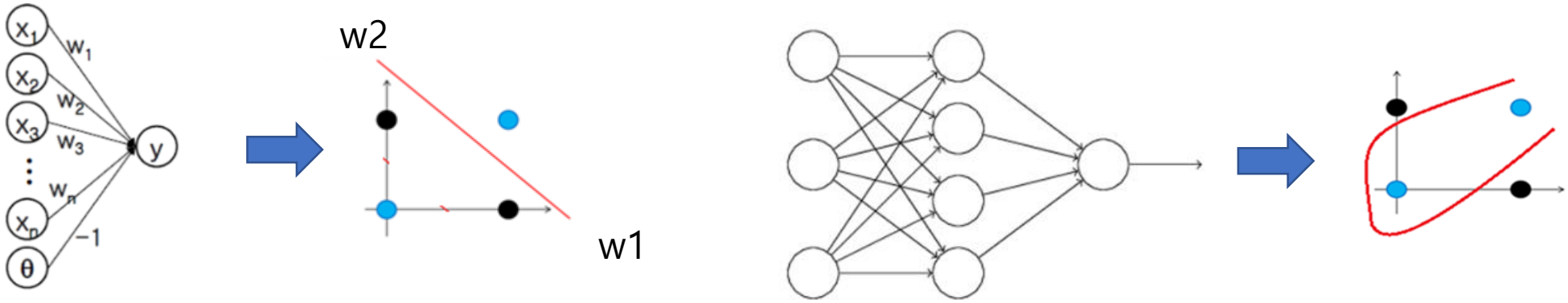
신경망의 역사

- Perceptron의 한계 해결책 1 =>Sigmoid Functions
  => 해결책은 Sigmoid 함수를 사용하는 것
  => 활성화 함수로 Sigmoid 함수를 사용하면 0~1의 연속적인 값을 가질 수 있음

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$

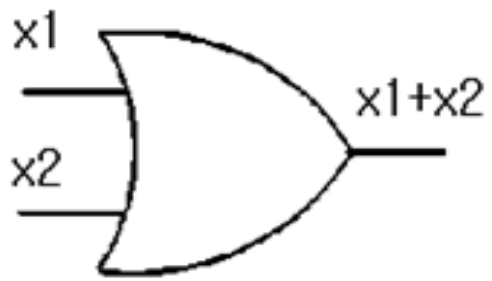


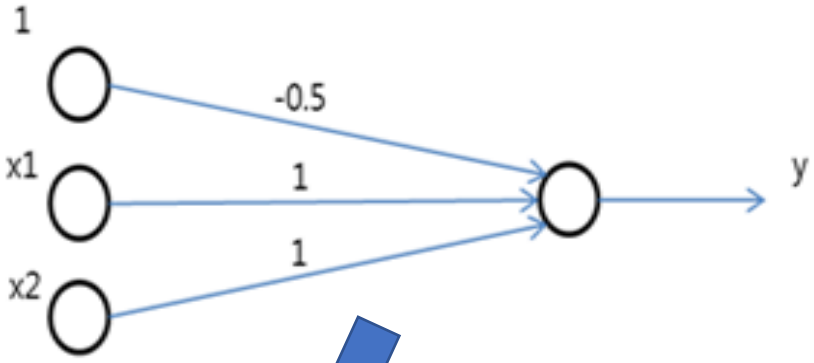
- Perceptron의 한계 해결책 2 =>Multi-layer Perceptron을 사용
  ex) 대표적인 문제가 Xor문제



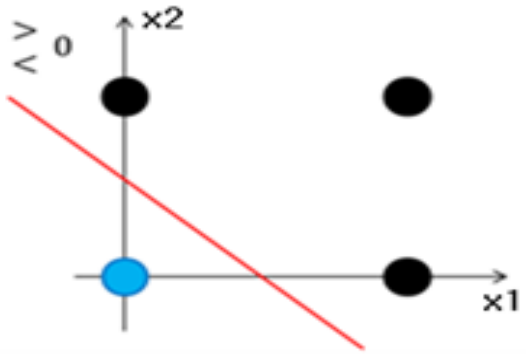
- **Neural Networks Example 1**

x1	x2	OR
0	0	0
0	1	1
1	0	1
1	1	1





$$d(\mathbf{x}) = x_1 + x_2 - 0.5 \gtrless 0$$



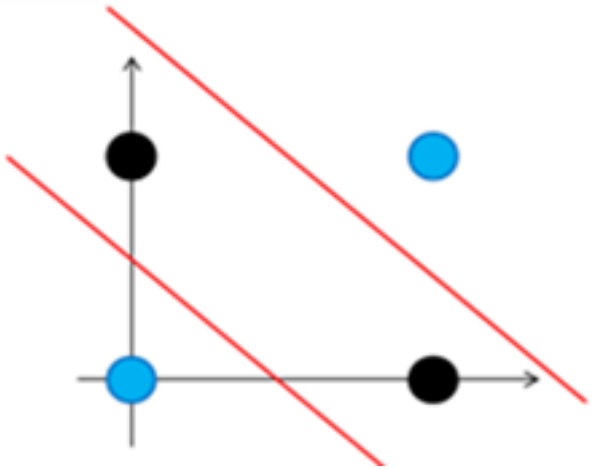
하나의 discriminant line만을 의미

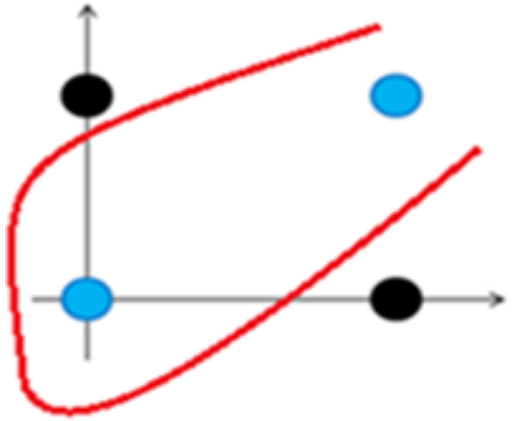
- **Single Layer Neural Network Problem**

⇒Single layer perceptron으로는 풀 수 없는 문제

⇒Single layer는 one discriminant만을 가지기 때문

⇒Neural Networks => linear $v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$



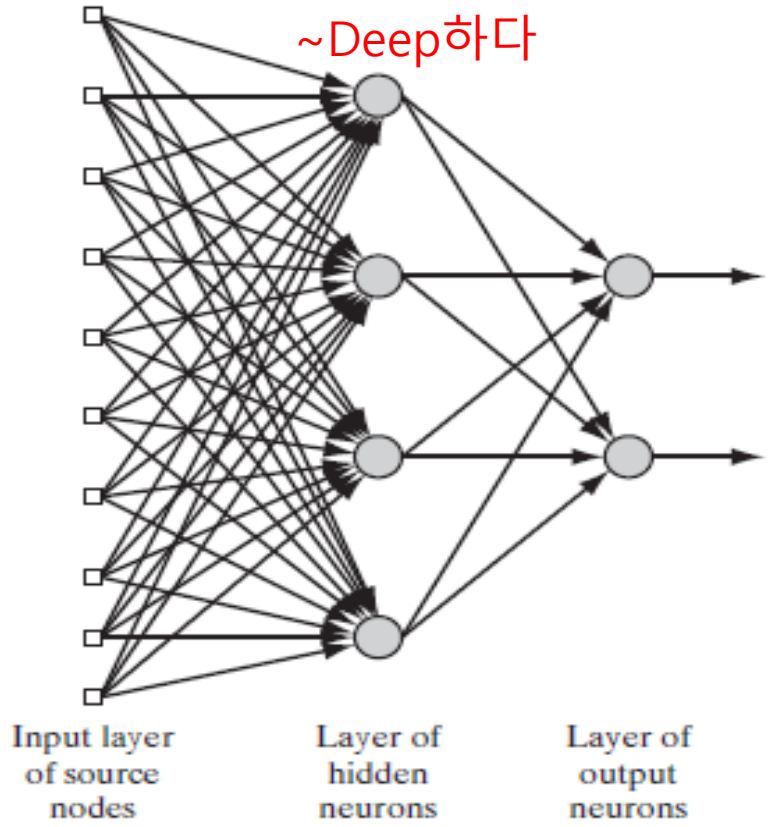


- 2개의 discriminant가 필요 -

01 Deep Learning Remind

Code Vision

- **Multi Layer Perceptron = Deep Neural Network(Deep Learning)**



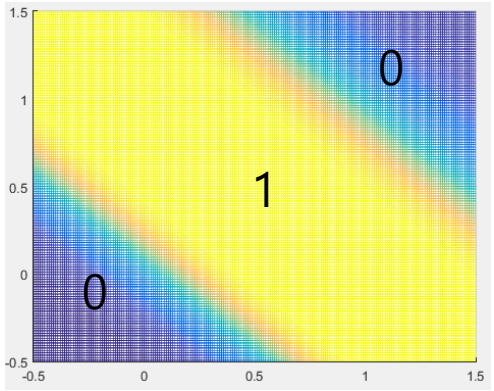
Deep learning은 Multiple Hidden layer 로 구성된 것을 의미한다.

01 Deep Learning Remind

Code Vision

신경망의 역사

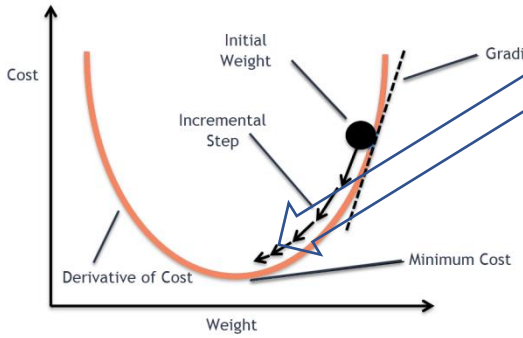
- 학습의 방법 => Gradient-Descent를 사용
  => 최적의 값을 찾아갈 때 흔히 사용하는 방법
  => 어떤 지점에 굴리기를 시작해도 그릇의 밑바닥까지 내려가면 최적 값에 도달했다라고 볼 수 있다.

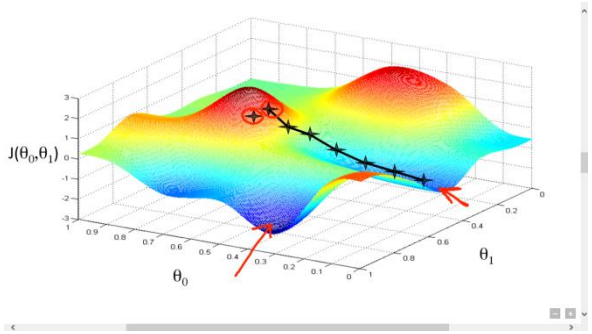


x1=1, x2=1일 때

실제 출력 값 정답
0.8 ⟺ 1

차이가 최소인 w들을 찾는것!



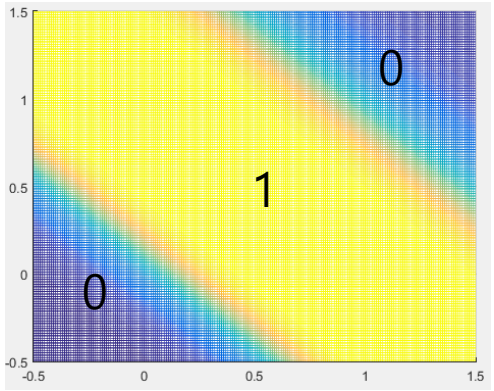


01 Deep Learning Remind

Code Vision

신경망의 역사

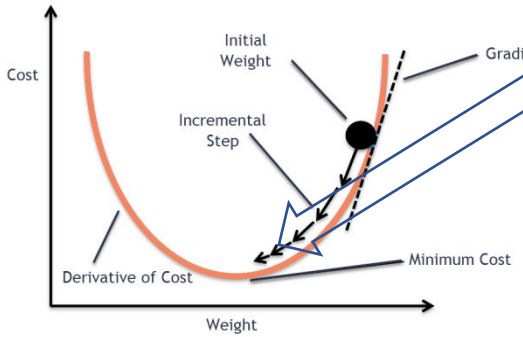
- 학습의 방법 => Gradient-Descent를 사용=> 수식은 Back propagation을 사용
  => 최적의 값을 찾아갈 때 흔히 사용하는 방법
  => 어떤 지점에 굴리기를 시작해도 그릇의 밑바닥까지 내려가면 최적 값에 도달했다라고 볼 수 있다.

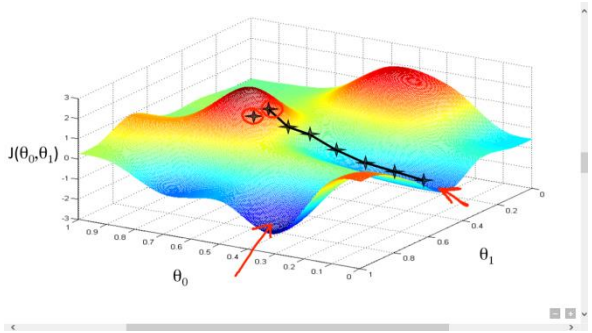


x1=1, x2=1일 때

실제 출력 값 정답
0.8 ⟺ 1

차이가 최소인 w들을 찾는것!





01 Deep Learning Remind

신경망의 역사

- Back propagation의 기원

  => 역전파를 통해 역방향으로 에러를 전파 시키면서 최적의 학습 결과를 찾는다는 의미

  => 1970년대 개발하였지만 1986년 Rumelhart와 Hinton의 논문을 통해 이 방법의 적합함이 증명됨₩

  => 1. Cost Function

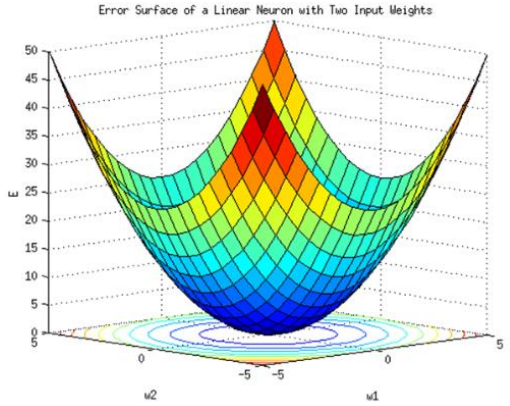
  - Cost Function의 정의

출력된 값 정답=라벨

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x)-a\|^2$$ MSE (Mean Square Function)

$\mathbf{y(x)}=\sigma(w1x1+x2w2+\cdots+b1+b2) \Rightarrow$
$y(x)$와 a가 차이가 작아지도록 하는 w, b들
을 찾아내는것

- 어느 정도 위치에 바닥이 있는지는 정확하게 알 수는 없지만 결국 바닥에 도달하게 된다.

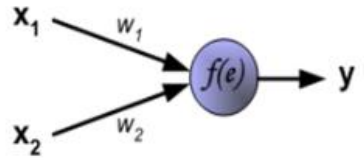


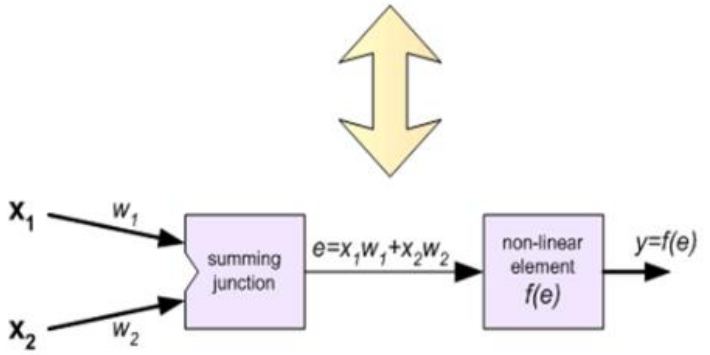
01 Deep Learning Remind

Code Vision

신경망의 역사

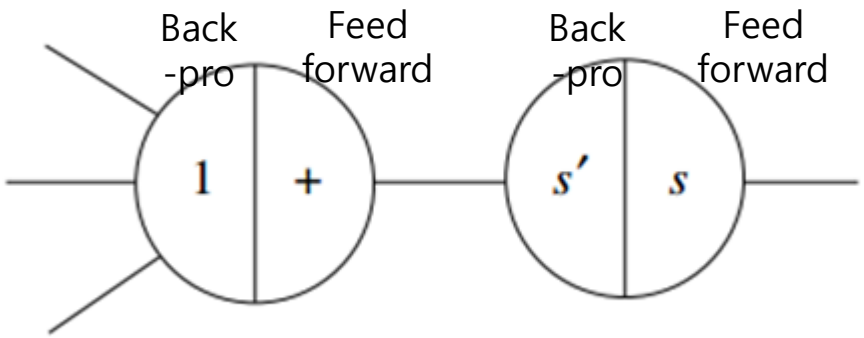
- 뉴런의 재구성
  => 입력을 합하는 부분과 활성화 함수 부분을 나눠서 생각하면 좋음!



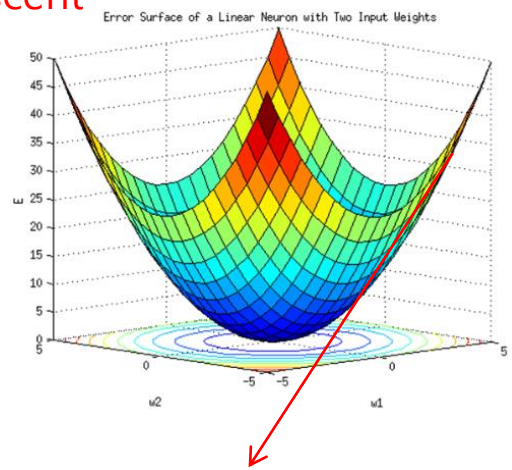


- 역전파(Back Propagation)의 기본 스텝
  => feed forward와 back propagation 두 단계로 나눠짐
  => feed forward는 단순히 출력을 내는 단계
  => back propagation은 역전파를 통해 W학습을 하는 단계

미분 관계



원래 출력 값에 미분을 하면 가장 급한 경사의 방향을 의미 = Gradient Descent



01 Deep Learning Remind

Code Vision

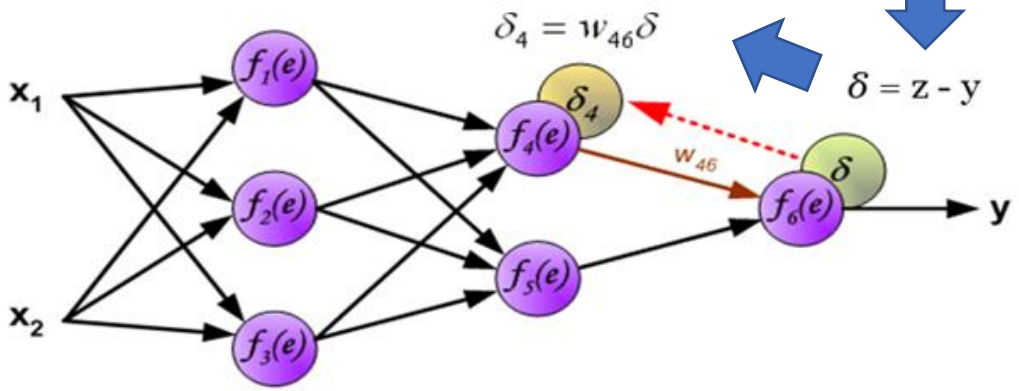
신경망의 역사

- **역전파(Back Propagation)의 2단계**
  => Feed Forward 단계에서는 훈련데이타를 신경망에 인가
  => loss function에서 큰 오차값이 생기면 그 오차값으로 w,b값을 갱신한다.
  => 2단계를 반복적으로 수행하여 최적화된 w,b 값을 생성

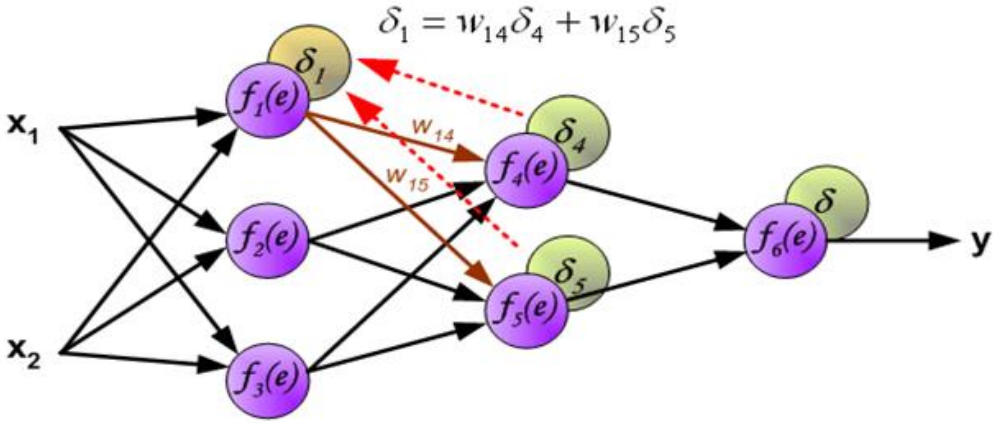
$$L(w, b)=\frac{1}{2}(z-y)^2$$

미분
= Gradient Descent

Feed forward를 통해 δ 을 구함
역 전파를 통해 반대방향으로 에러를 전파



얻어진 error로 $\delta 4$, $\delta 5$ 도 구할 수 있다.



01 Deep Learning Remind

Code Vision

신경망의 역사

- 역전파(Back Propagation)의 learning rate
  => 학습 조절 변수
  => 학습 속도를 조절하기 위해 학습조절 변수(learning rate)인 η을 곱함.

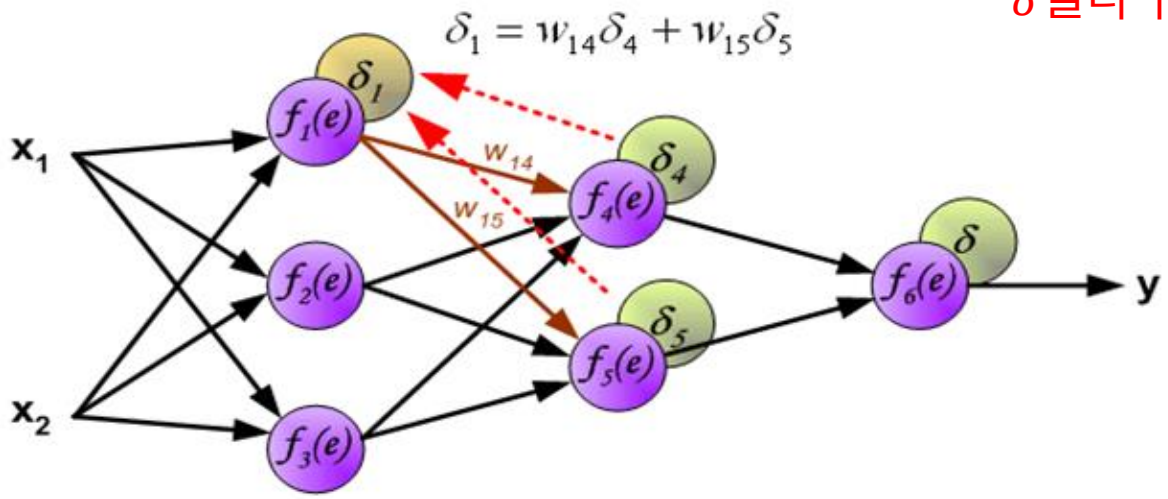
$$w^{+} = w - \eta \partial C/\partial w \qquad L(w, b)=\frac{1}{2}(z-y)^2 \Rightarrow \frac{\partial C}{\partial w}=z-y$$

- Sigmoid 함수의 좋은 성질과 delta rule

$$f'(x) = \frac{\partial f(x)}{\partial x} = f(x)(1-f(x))$$

$$w^{+} = w - \eta \frac{\partial L(w,b)}{\partial x} * f'(x)$$

δ델타 규칙은 한번더 설명!



01 Deep Learning Remind

Code Vision

신경망의 역사

- 역전파(Back Propagation)의 learning rate
  => 학습 조절 변수
  => 학습 속도를 조절하기 위해 학습조절 변수(learning rate)인 η을 곱함.

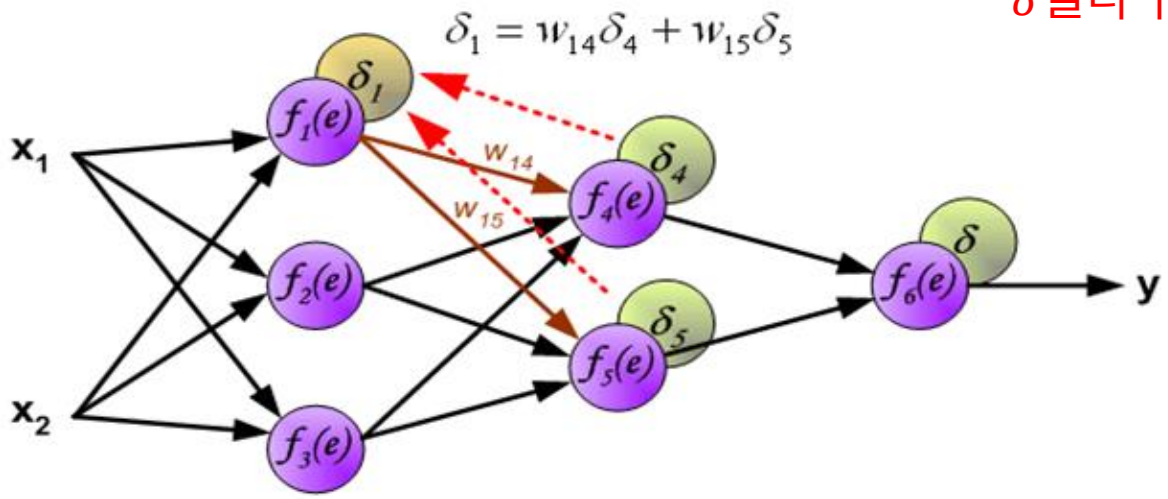
$$w^{+} = w - \eta \partial C/\partial w \qquad L(w, b)=\frac{1}{2}(z-y)^2 \Rightarrow \frac{\partial C}{\partial w}=z-y$$

- Sigmoid 함수의 좋은 성질과 delta rule

$$f'(x) = \frac{\partial f(x)}{\partial x} = f(x)(1-f(x))$$

$$w^{+} = w - \eta \frac{\partial L(w,b)}{\partial x} * f'(x)$$

δ델타 규칙은 한번더 설명!



01 Deep Learning Remind

신경망의 역사

- 역전파 (Back Propagation)은?

  => 최적의 학습 결과를 찾기 위해 역방향으로 에러를 전파

  => Back propagation이해를 위한 cost function 개념

  - 입력된 훈련 데이터에 관한 실제 출력과 기대 출력간의 차이
  - cost function이 최소값이 되도록 backpropagtion

  => Back propagation 이해를 위한 Gradient-Descent 개념

  - cost function이 최소가 되도록 하는 방법
  - cost function의 인자인 가중치(w)와 바이어스(b)를 조절함

  => Back propagation의 두 단계

  - feed forward : 입력 => 출력(최종 출력단에서 에러와 cost function을 구함)
  - back propagation : 출력부터 반대방향으로 순차적으로 편미분을 수행해가면서 뉴런의 가중치(w)의 바이어스(b) 값을 갱신

01 Deep Learning Remind

신경망의 역사

- 역전파 (Back Propagation)은?

  => 최적의 학습 결과를 찾기 위해 역방향으로 에러를 전파

  => Back propagation이해를 위한 cost function 개념

  - 입력된 훈련 데이터에 관한 실제 출력과 기대 출력간의 차이
  - cost function이 최소값이 되도록 backpropagtion

  => Back propagation 이해를 위한 Gradient-Descent 개념

  - cost function이 최소가 되도록 하는 방법
  - cost function의 인자인 가중치(w)와 바이어스(b)를 조절함

  => Back propagation의 두 단계

  - feed forward : 입력 => 출력(최종 출력단에서 에러와 cost function을 구함)
  - back propagation : 출력부터 반대방향으로 순차적으로 편미분을 수행해가면서 뉴런의 가중치(w)의 바이어스(b) 값을 갱신

01 Deep Learning Remind

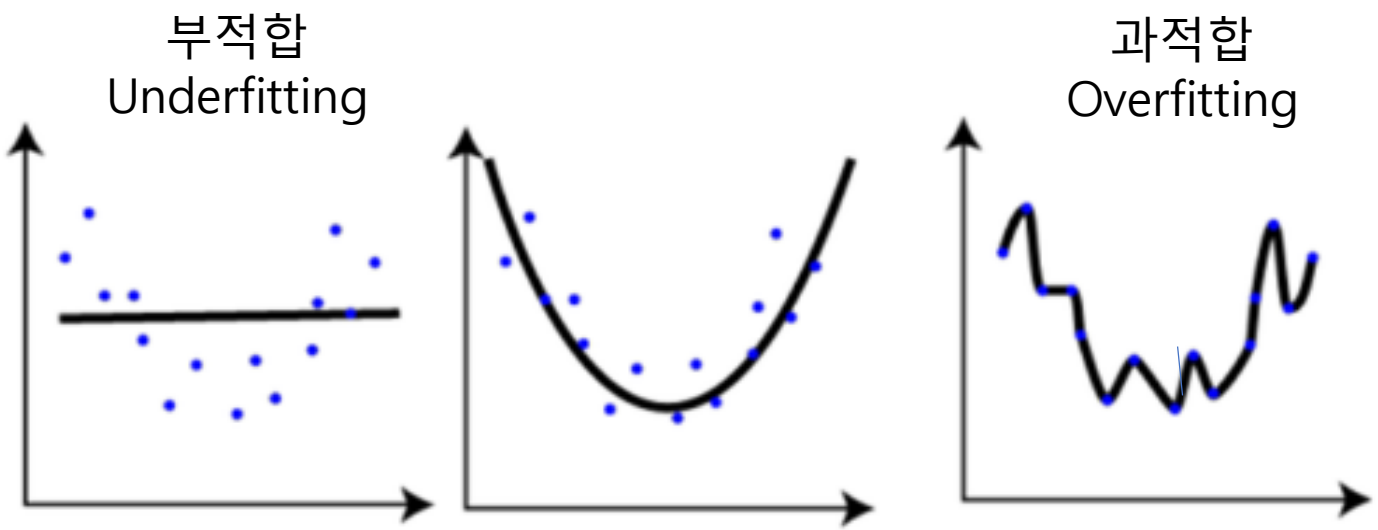
Code Vision

Overfitting

- 통계에서 좋은 결과를 얻기 위해서는 전체를 대표할 수 있는 샘플
- 전체를 잘 설명할 수 있는 모델을 만들 어야함

- 부적합 : 데이터를 잘 나타내고 있지 못함
- 과적합 : 모든 데이터에 맞춰져 새로운 데이터가 들어오면 취약함

- 이상적인 해결 방법 : 데이터의 개수를 늘리는 것=>대부분 데이터를 늘리기는 어렵다.
- 오컴의 면도날 : 사고 절약의 원리 => 두가지 주장이 있다면 그나마 간단한 쪽을 택하자



01 Deep Learning Remind

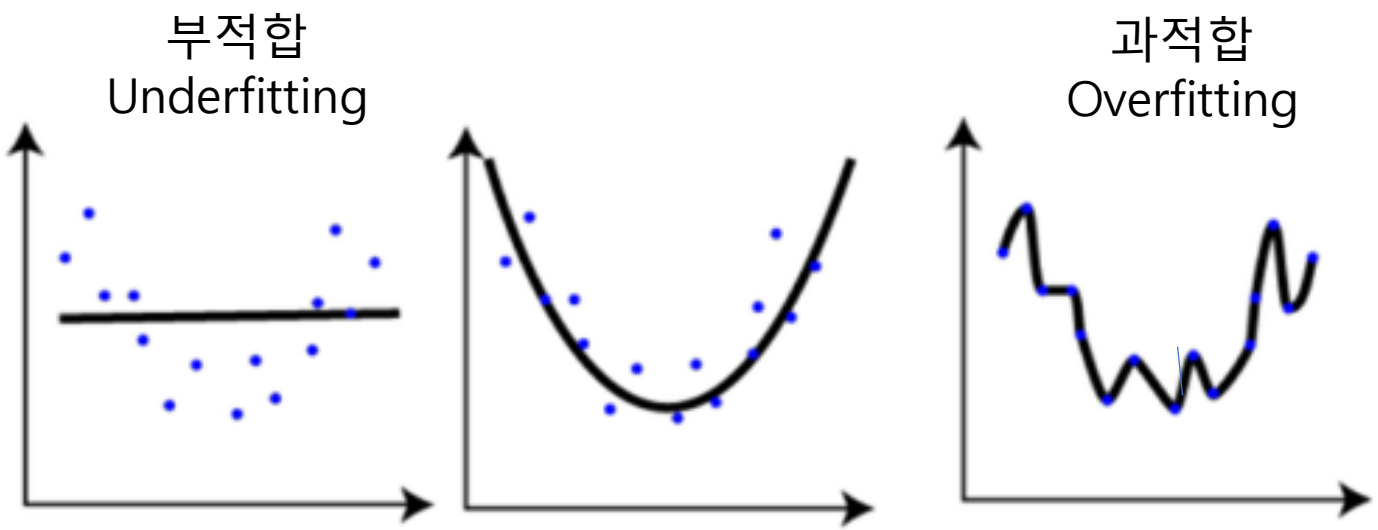
Code Vision

Overfitting

- 통계에서 좋은 결과를 얻기 위해서는 전체를 대표할 수 있는 샘플
- 전체를 잘 설명할 수 있는 모델을 만들 어야함

- 부적합 : 데이터를 잘 나타내고 있지 못함
- 과적합 : 모든 데이터에 맞춰져 새로운 데이터가 들어오면 취약함

- 이상적인 해결 방법 : 데이터의 개수를 늘리는 것=>대부분 데이터를 늘리기는 어렵다.
- 오컴의 면도날 : 사고 절약의 원리 => 두가지 주장이 있다면 그나마 간단한 쪽을 택하자



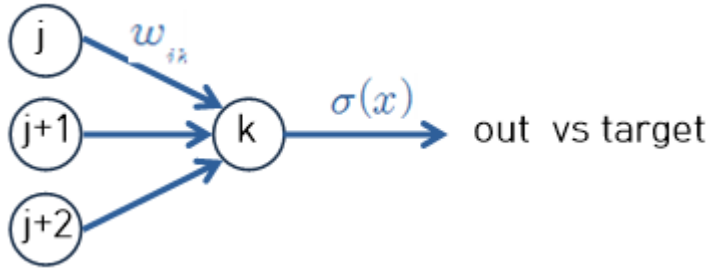
01 Deep Learning Remind

- **Multi Layer Perceptron = Deep Neural Network(Deep Learning)**

① forward propagation

1.Initialization

W1,W2, ,Wn = randn()
Bias = [1 1];



2. Activation

Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$

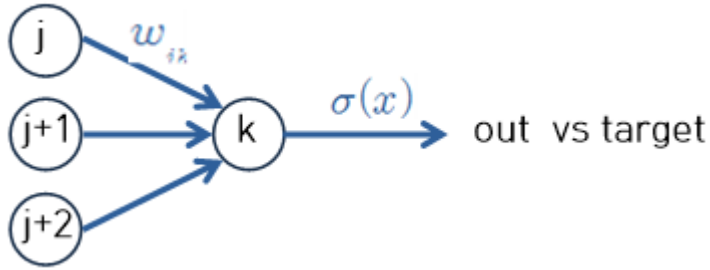
01 Deep Learning Remind

- **Multi Layer Perceptron = Deep Neural Network(Deep Learning)**

① forward propagation

1.Initialization

W1,W2, ,Wn = randn()
Bias = [1 1];



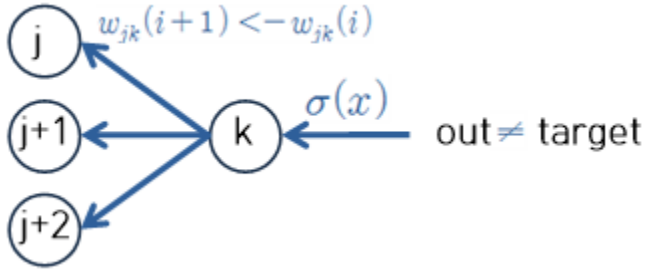
2. Activation

Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$

Code Vision

②backward propagation

3. Learning weigh



1. binary step

$$e_k = t_k - o_k$$

2.sigmoid

$$E = (t_k - o_k)^2$$

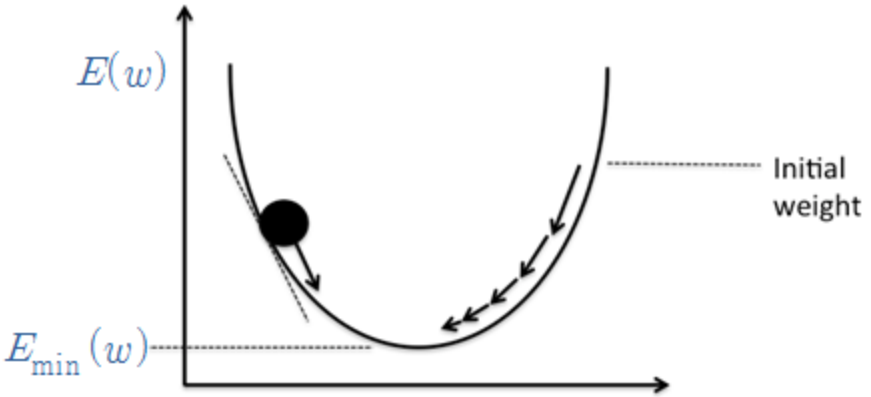
gradient descent

$$\triangle w_{jk}(n) = \alpha \times o_j \times \sum e_k$$ true error

$$\triangle w_{jk}(n) = \alpha \frac{\partial E}{\partial w_{jk}}$$

4.Repeat

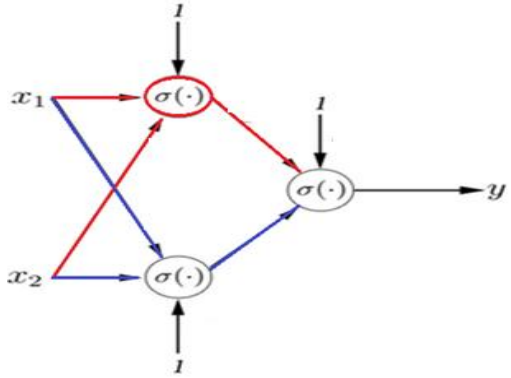
$$w(n+1) = w(n) + \triangle w(n)$$



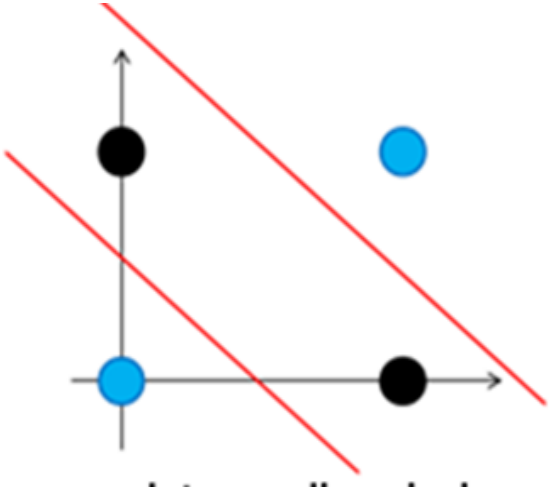
Code Vision

①toy example : XOR classification

layer = 2; node=[2 2 1]; bias_value = [1 1]; L_rate = 0.1



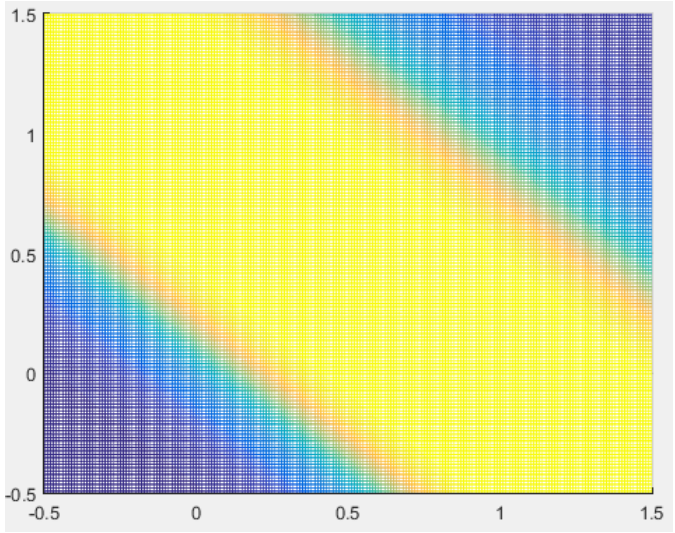

```
for i=1:30000    % iteration
a=randperm(4,1); %random training
if  mod(a,4)==1,              input=[0 0]; target=0.01;
    elseif mod(a,4)==2,    input=[0 1]; target=0.99;
    elseif mod(a,4)==3,    input=[1 0]; target=0.99;
    else                          input=[1 1]; target=0.0
 end
                                (omit)
 end
```

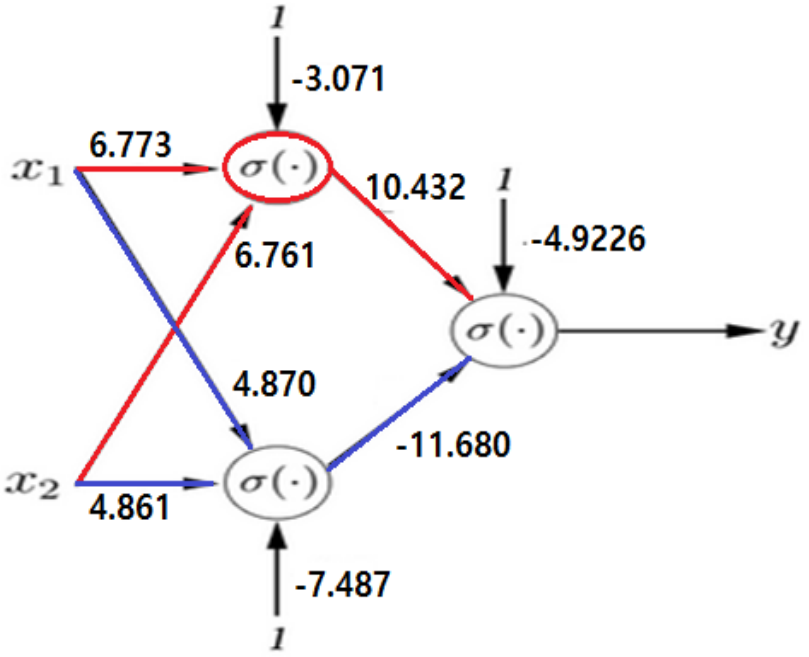


01 Deep Learning Remind

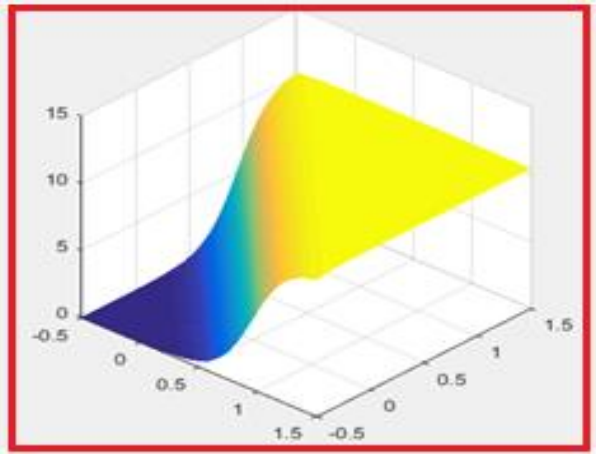
Code Vision

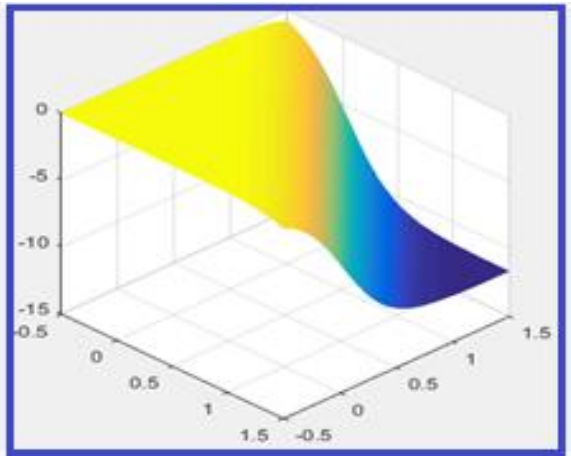
input	output
[0 0]	0.0089
[0 1]	0.9891
[1 0]	0.9891
[1 1]	0.0109

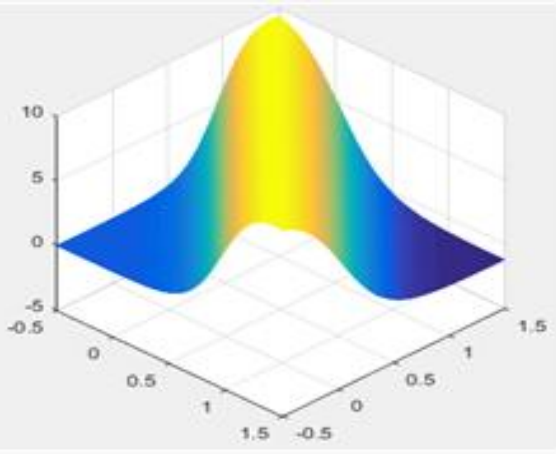




01 Deep Learning Remind





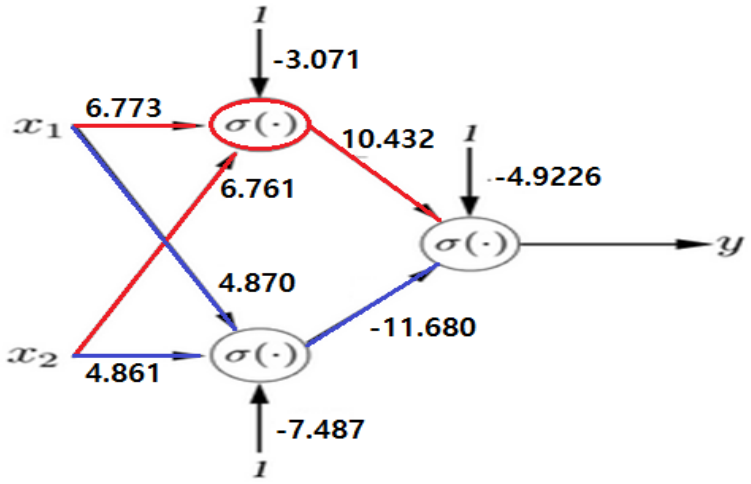


$$z1 = \frac{10.432}{1 + e^{(-6.773x_1 + 6.761x_2 - 3.071)}}$$

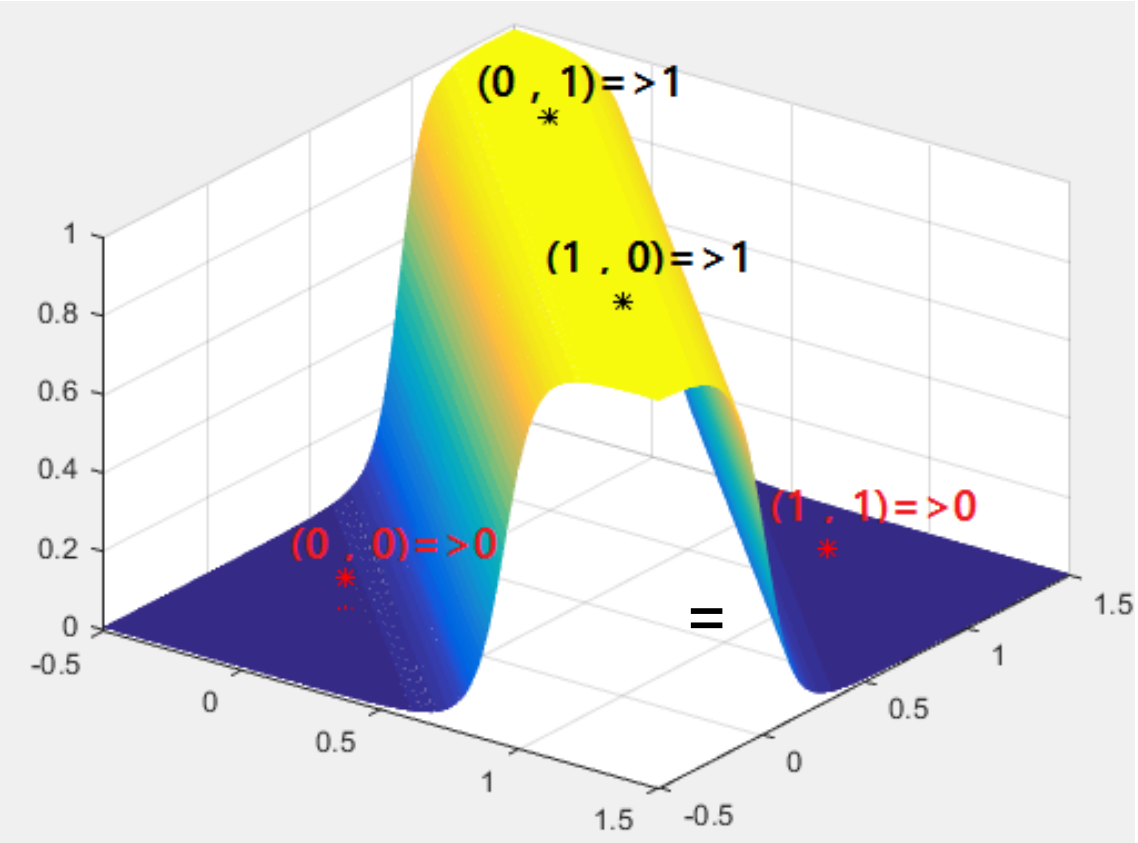
$$z2 = \frac{-11.680}{1 + e^{(4.87x_1 + 4.861x_2 - 7.487)}}$$

$$z1 + z2$$

weights plays role in scaling sigmoid function



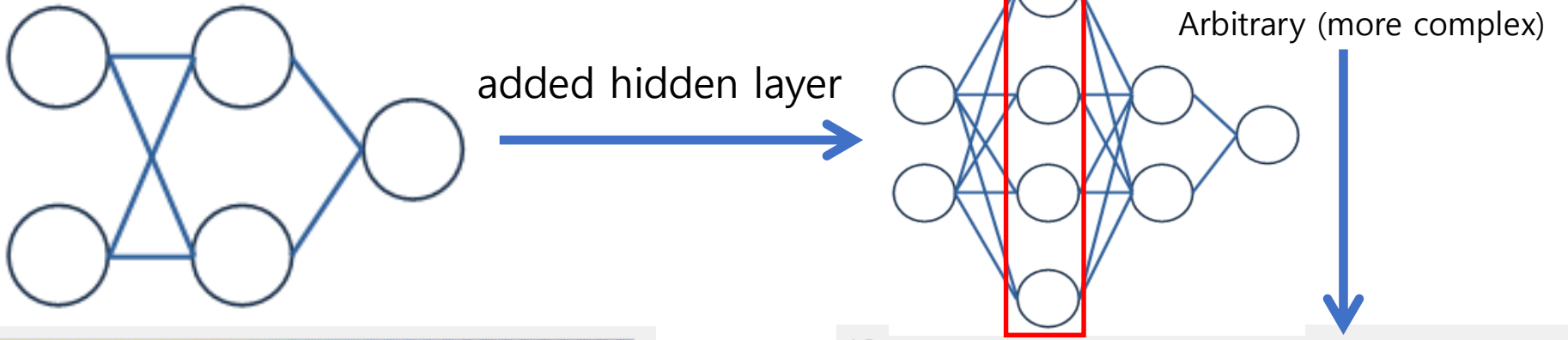
01 Deep Learning Remind

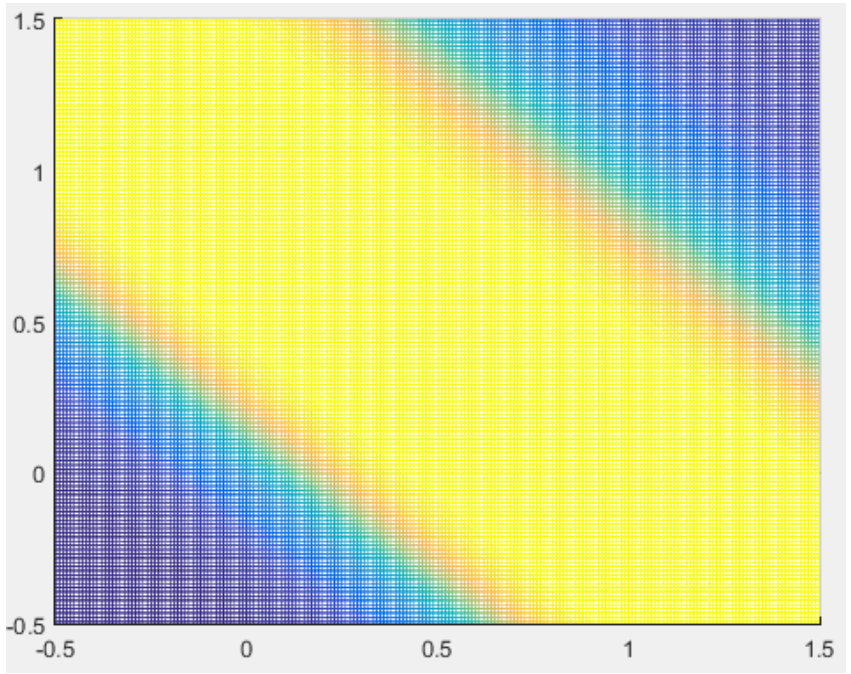


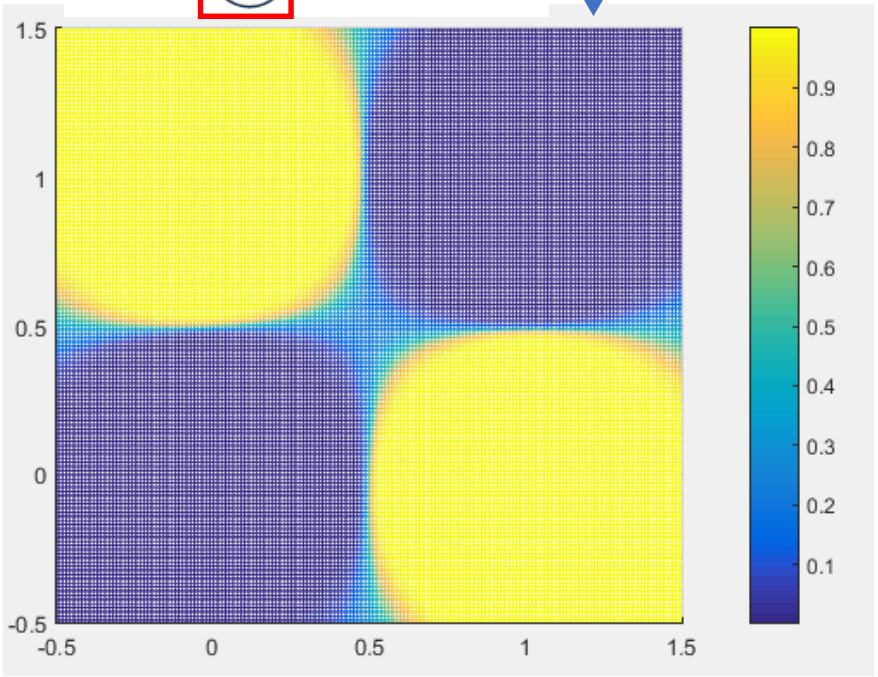
$$output = \frac{1}{1 + e^{-(z1+z2-4.923)}}$$

01 Deep Learning Remind

Code Vision

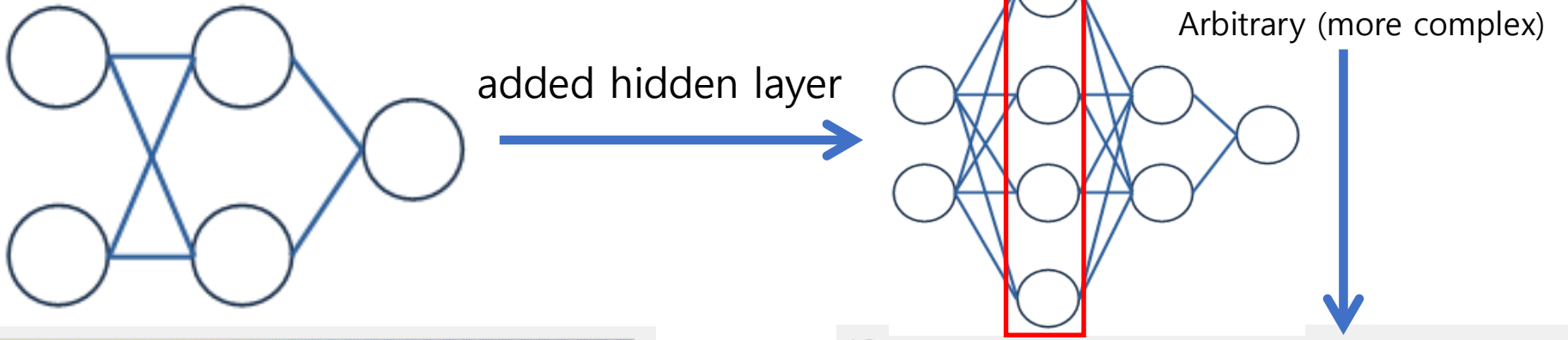


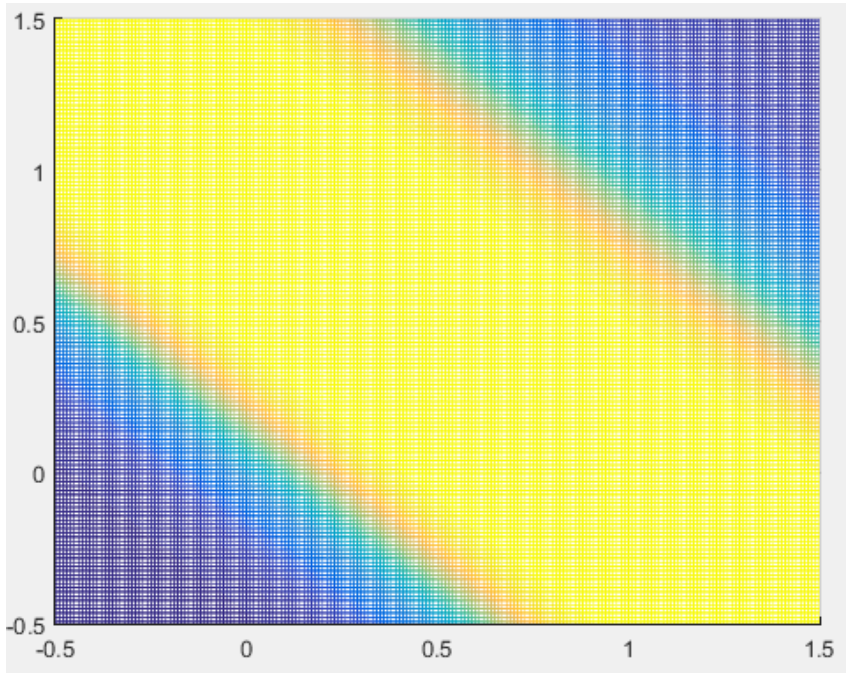


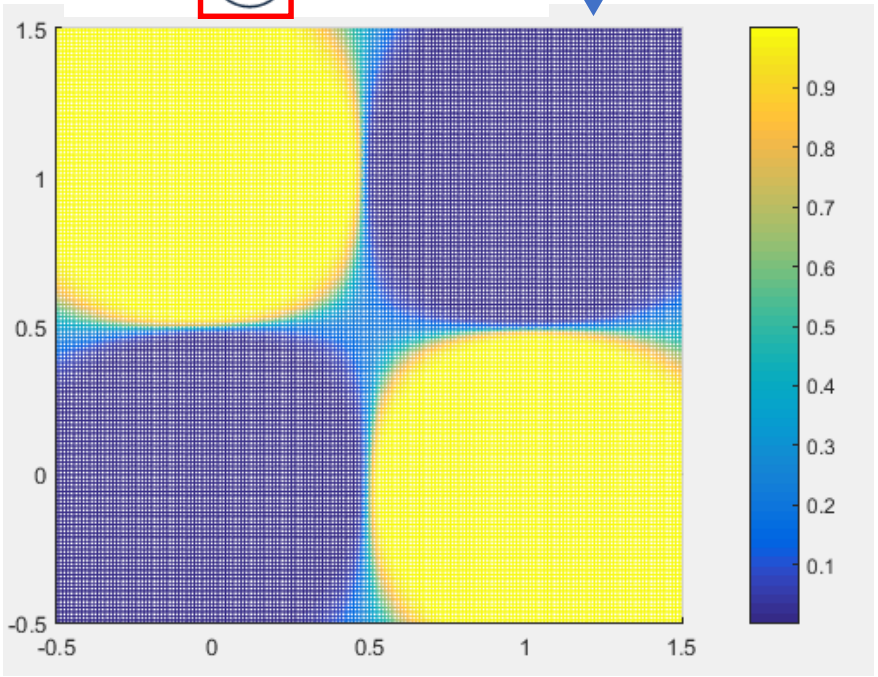


01 Deep Learning Remind

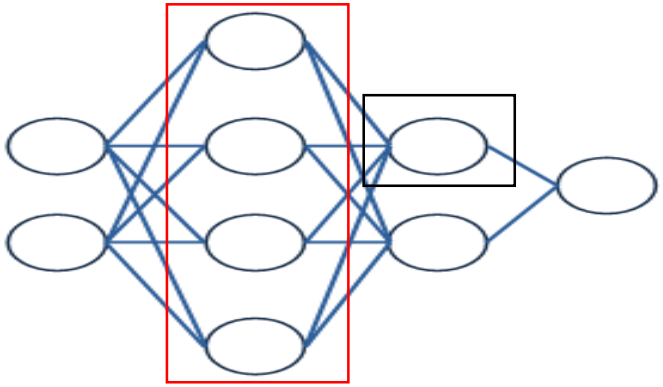
Code Vision



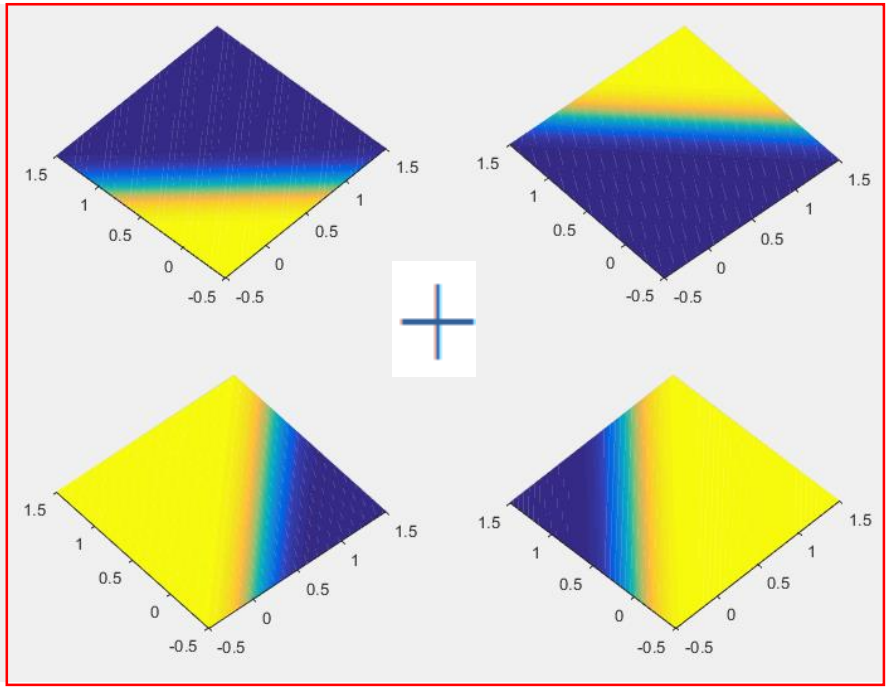




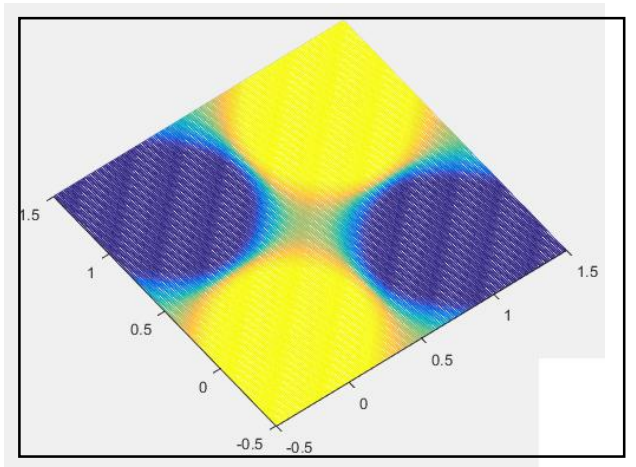
01 Deep Learning Remind



weighted units

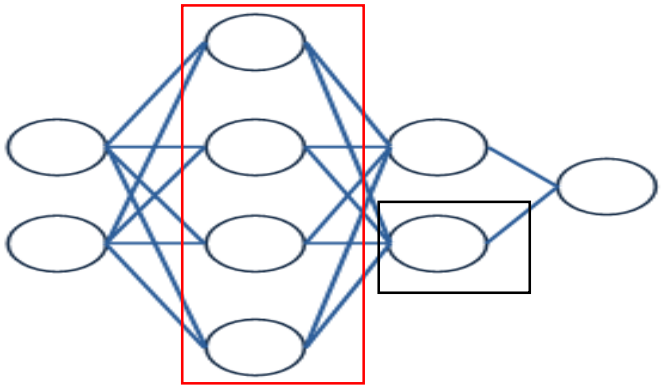


=

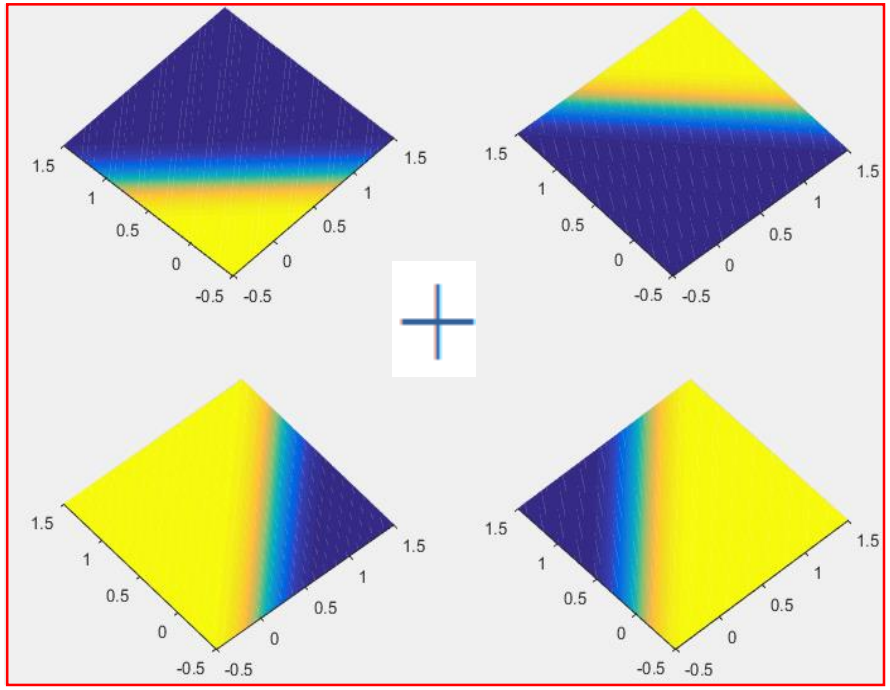


01 Deep Learning Remind

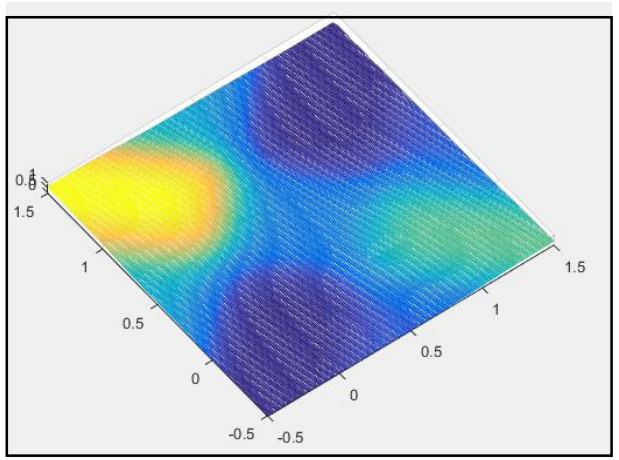
Code Vision



weighted units

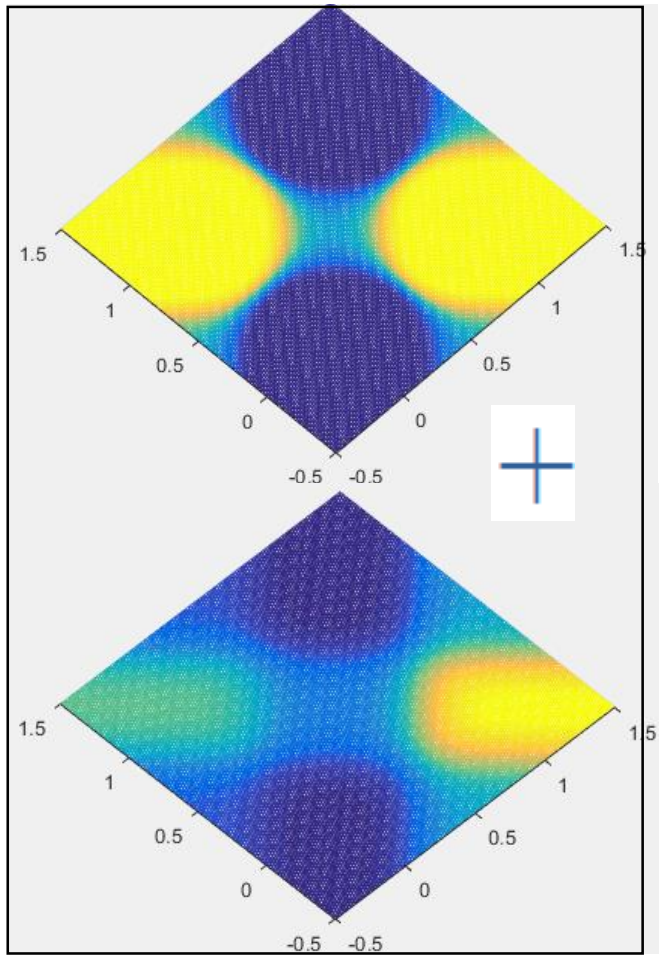


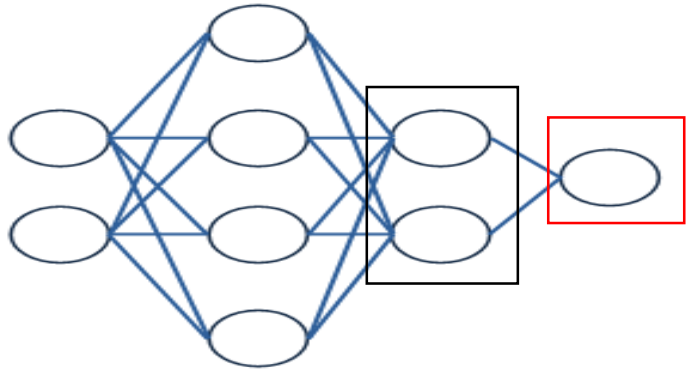
=

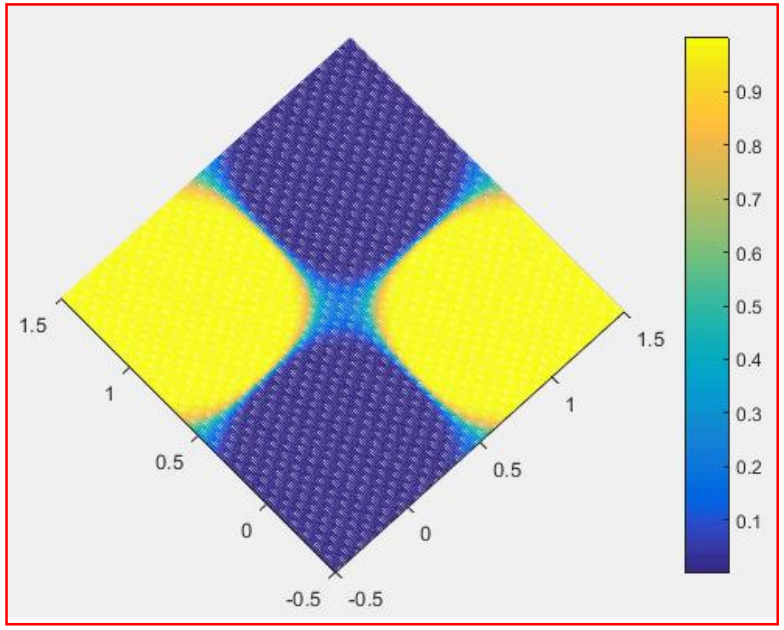


01 Deep Learning Remind-(DNN)

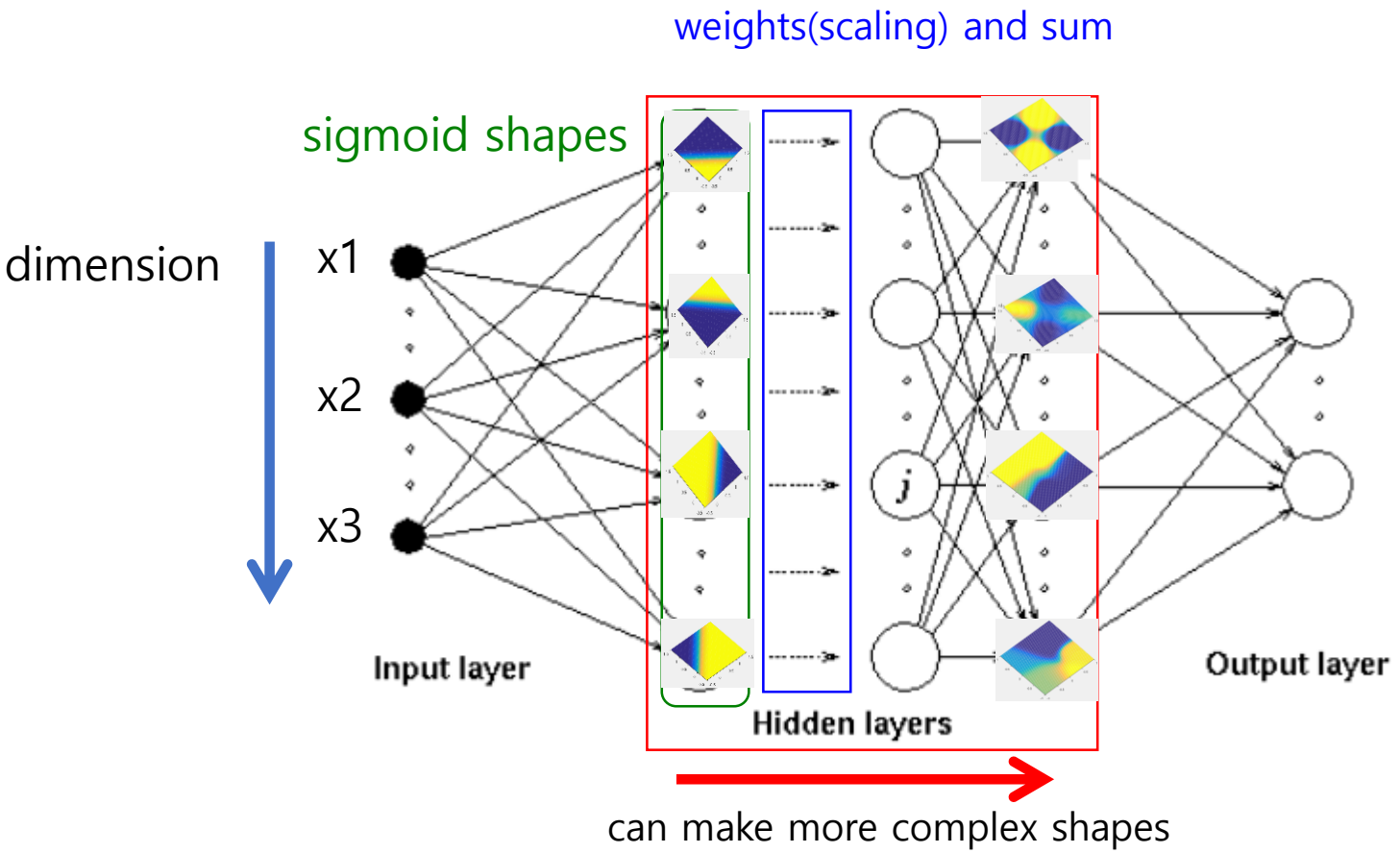
Code Vision





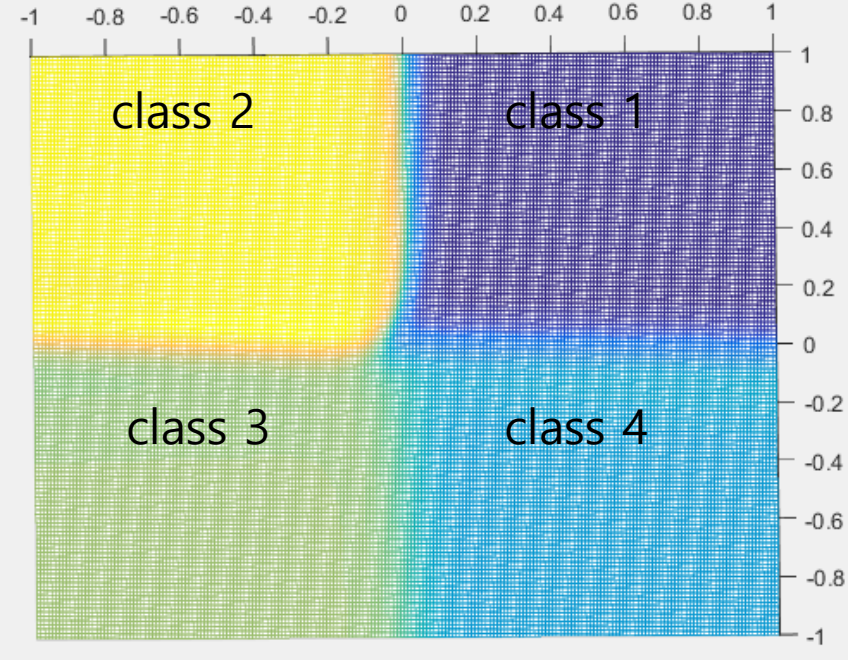


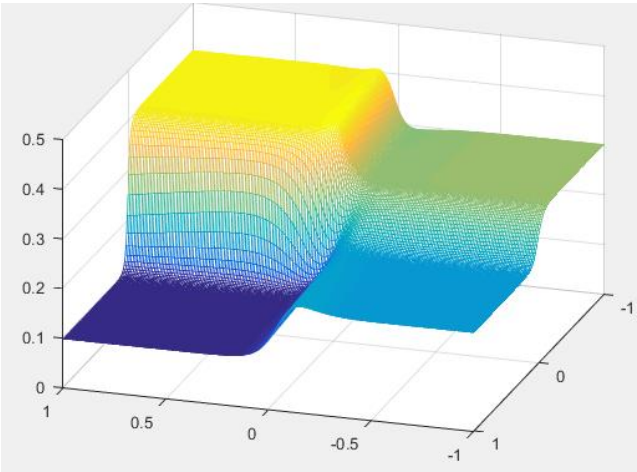
01 Deep Learning Remind-(DNN)

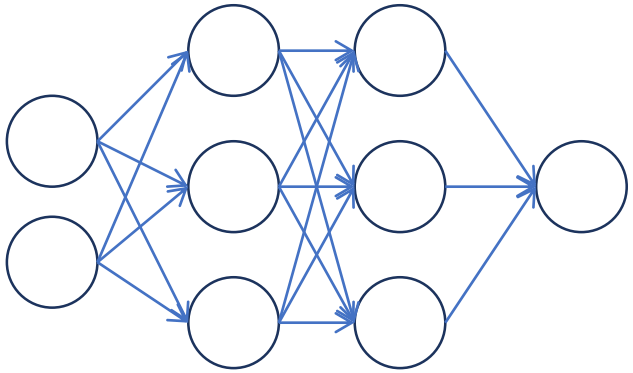


Code Vision

②toy example : Quadrant classification



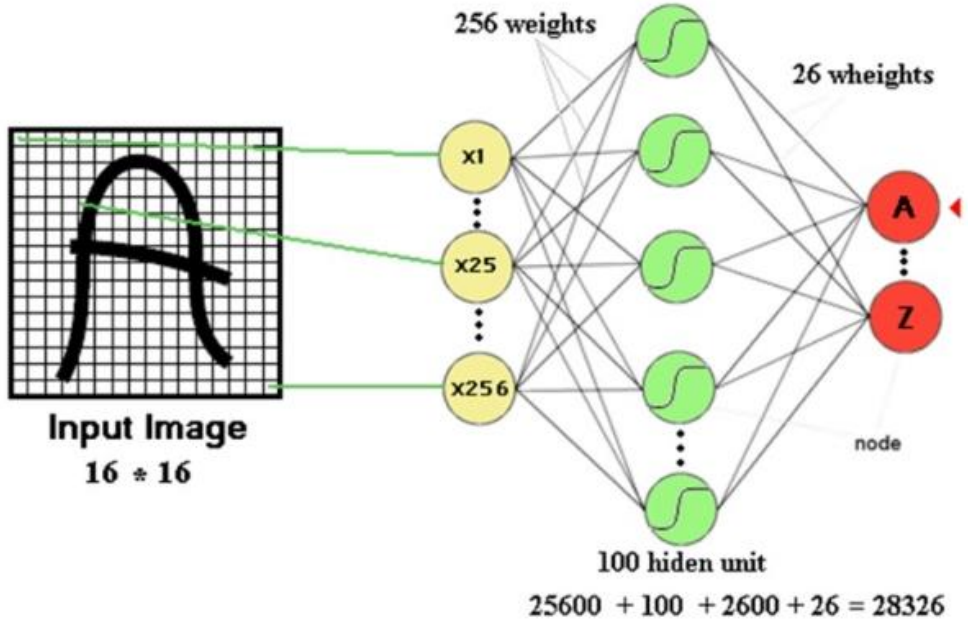


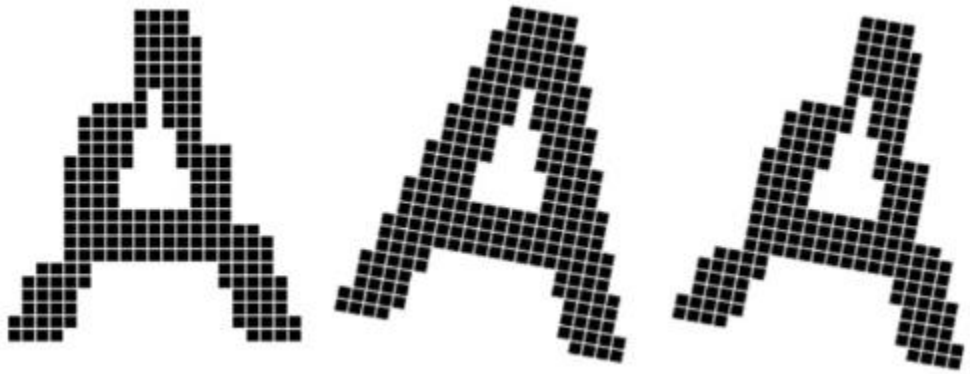


02 Deep Learning Remind-(CNN)

Code Vision

Fully-connected neural network Limitation





Fully connected layer for object recognition ...

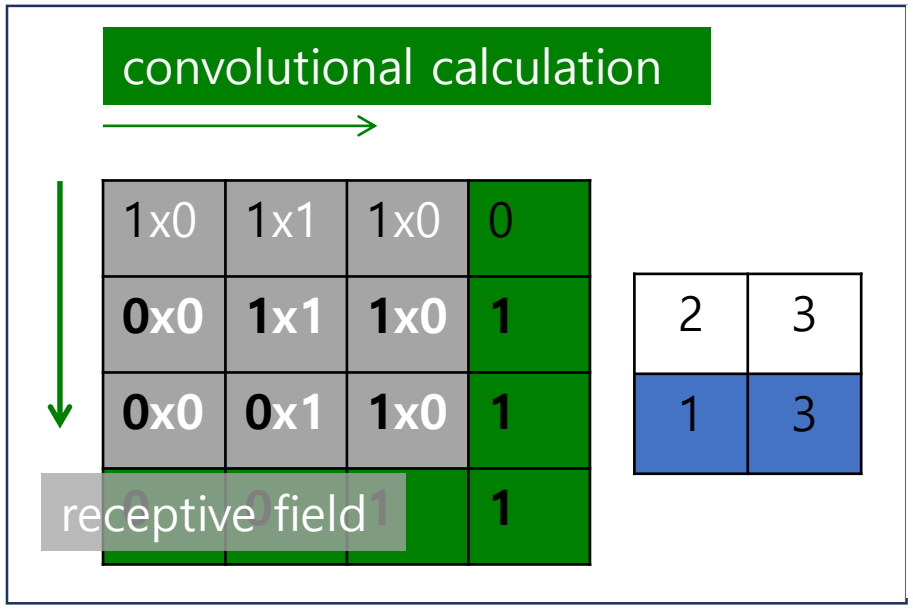
- increases the capacity of the system
- requires a larger training set.
- takes many times
- has no built-in invariance with respect to translations, or local distortions of the inputs.

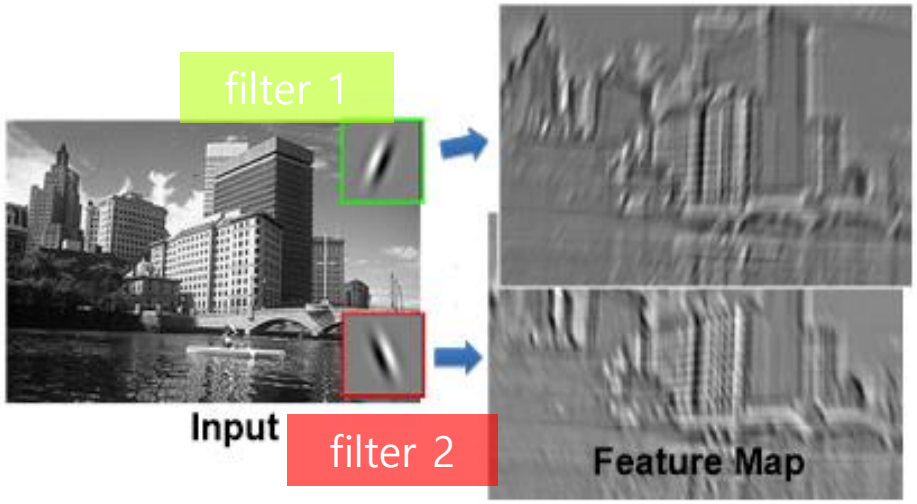
http://laonple.blog.me/220654387455

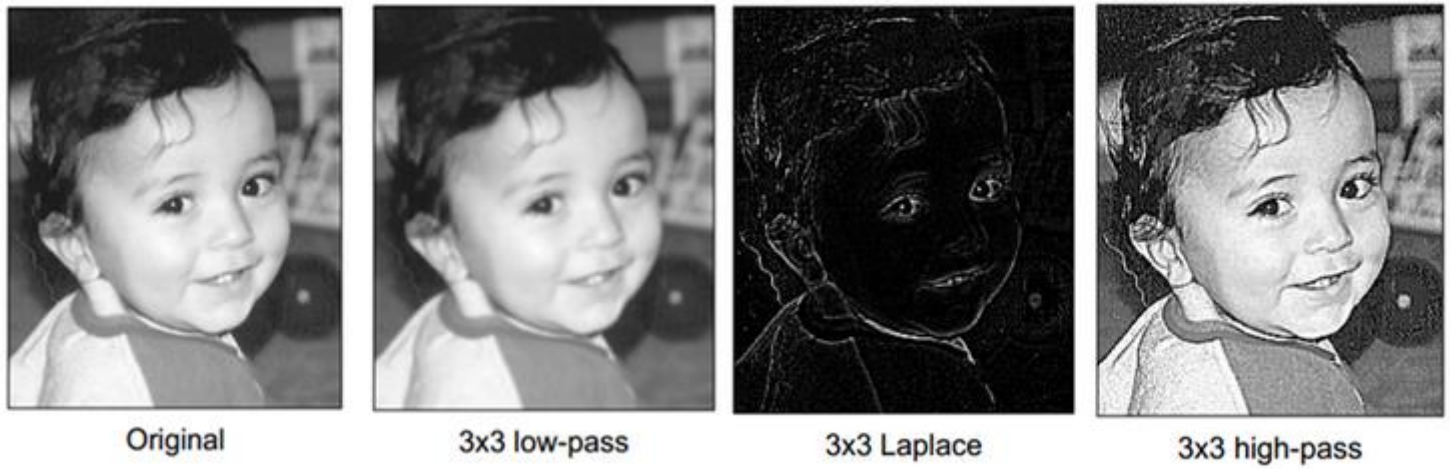
02 Deep Learning Remind-(CNN)

Code Vision

CNN Principle Convolution

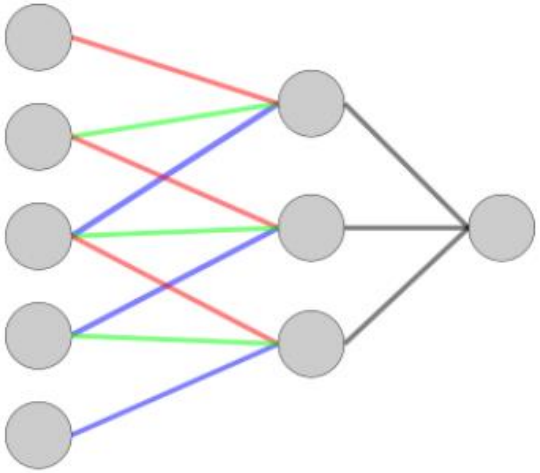






02 Deep Learning Remind-(CNN)

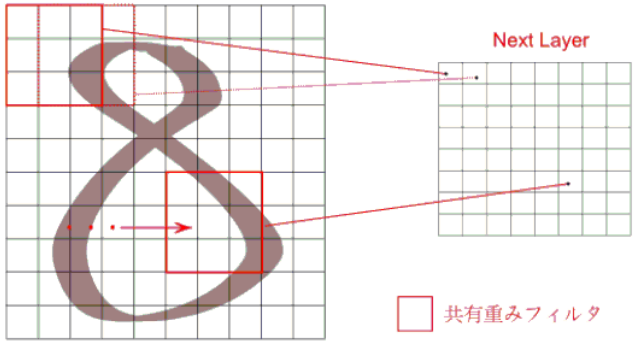
Suitable for classifying image



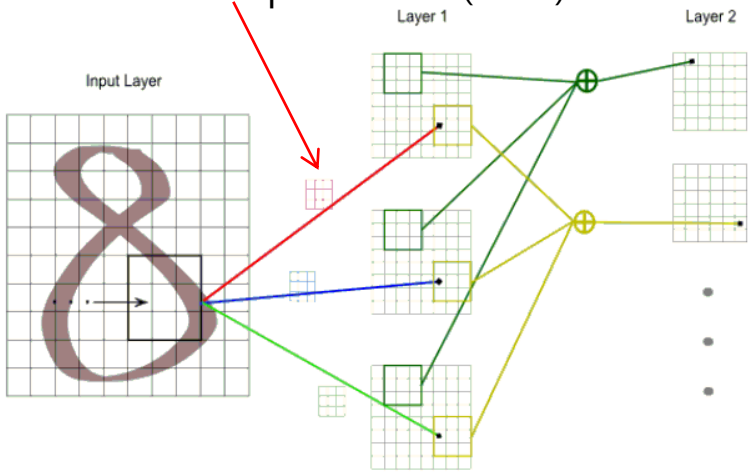
Convolutional Neural Networks work well for many object recognition

- reduces number of free parameters
- brings us some translation invariance

the number of input neurons :28 x 28
but many fewer weights by sharing these



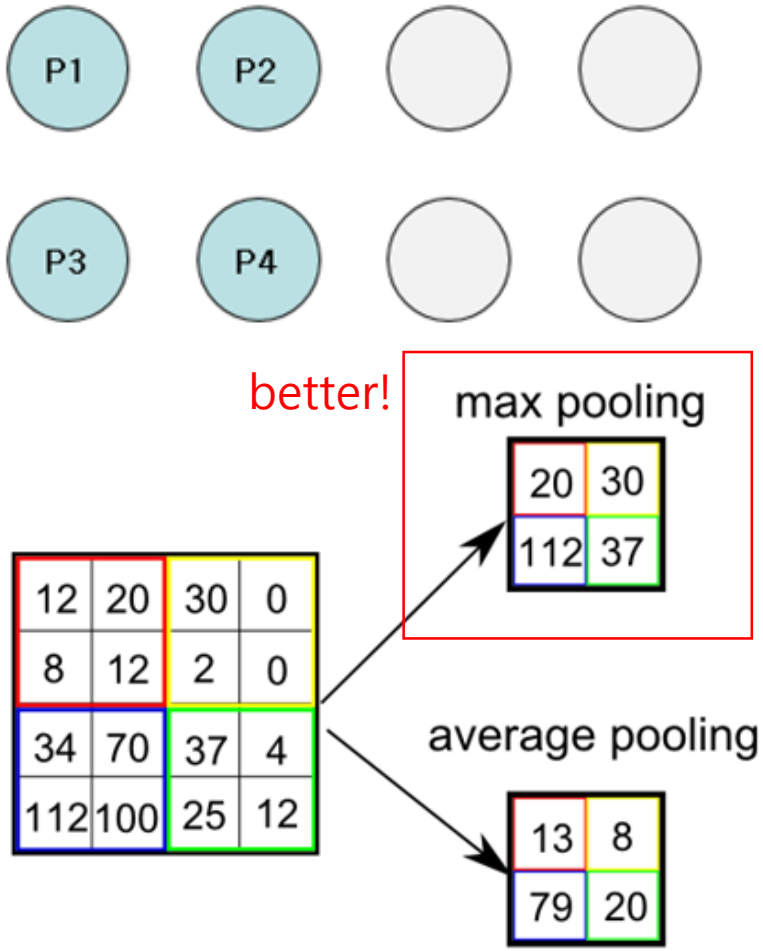
receptive field (filter)



feature maps

02 Deep Learning Remind-(CNN)

Suitable for classifying image



average : (P1+P2+P3+P4)/4

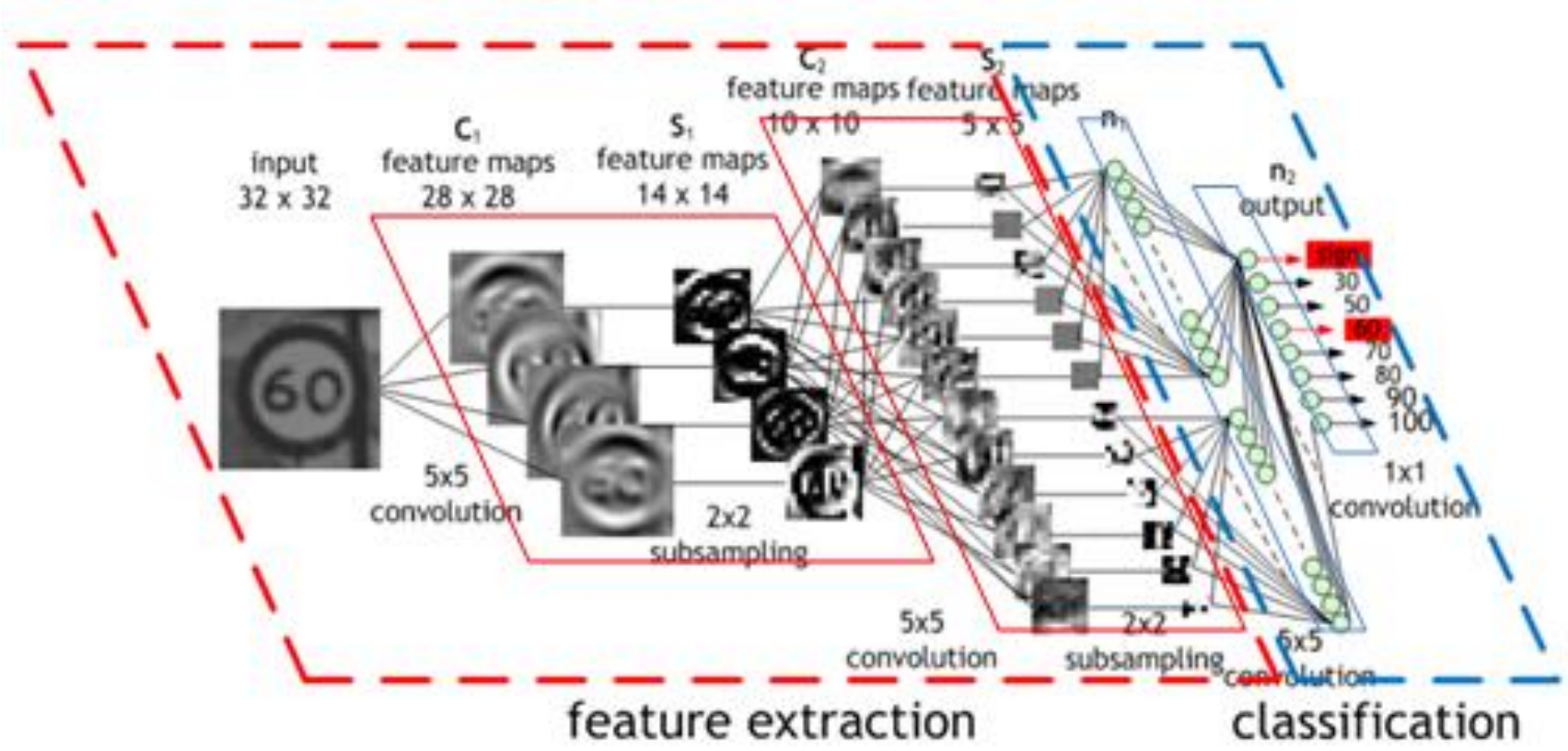
max : max(P1, P2, P3, P4)

- max function -

In neuroloy, the strongest signal is transmitted, others are ignored

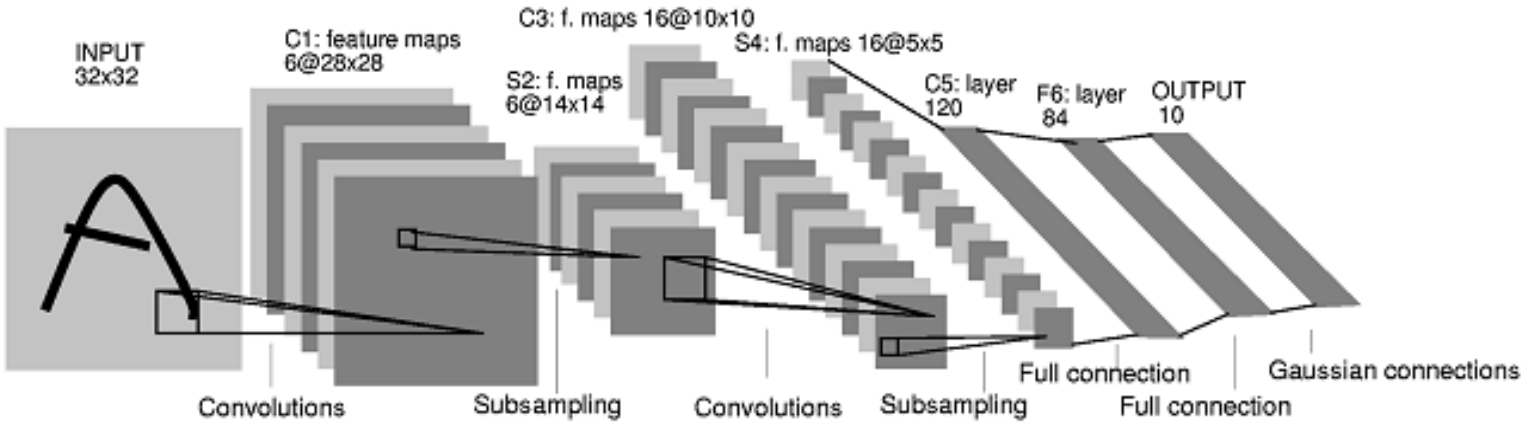
02 Deep Learning Remind-(CNN)

General CNN Architecture



02 MNIST Classification

문자,숫자 구분을 위한 CNN Architecture



input : 32 x 32 (for NMIST)

C1 : convolution layer (6 kernal size 5 x 5 x 6)

S2 : subsampling layer (6 kernal size 2 x 2, average, not overlaping)

C3 : convolution layer (16 kernal size 5 x 5 x 16)

S4 : subsampling layer (scale size 2 x 2, average, not overlaping)

C5 , F6 : fully connected layer (for classification)

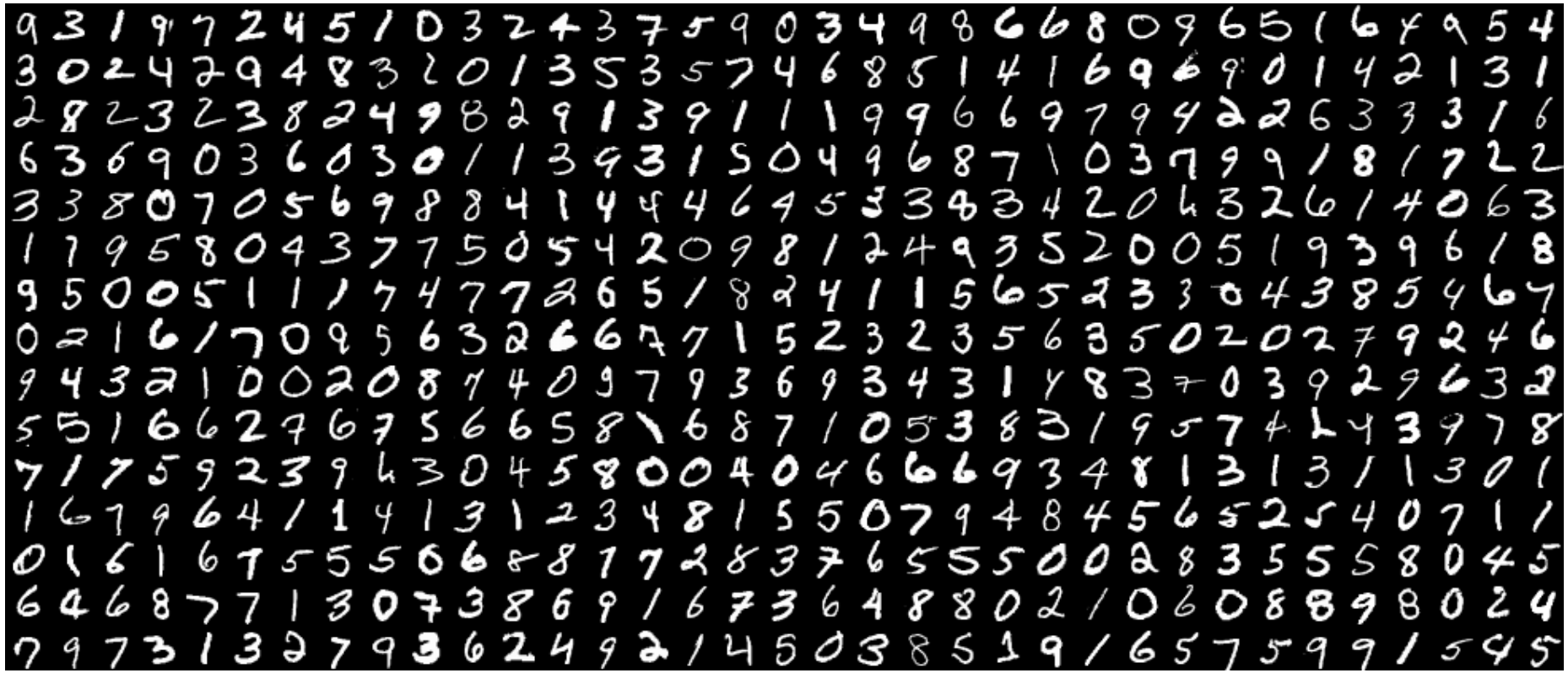
activation function : sigmoid (only apply to convolution layer)

Gradient-Based learning Applied to Document Recogntition (Yann LeCun 1990)

02 MNIST Classification

Code Vision

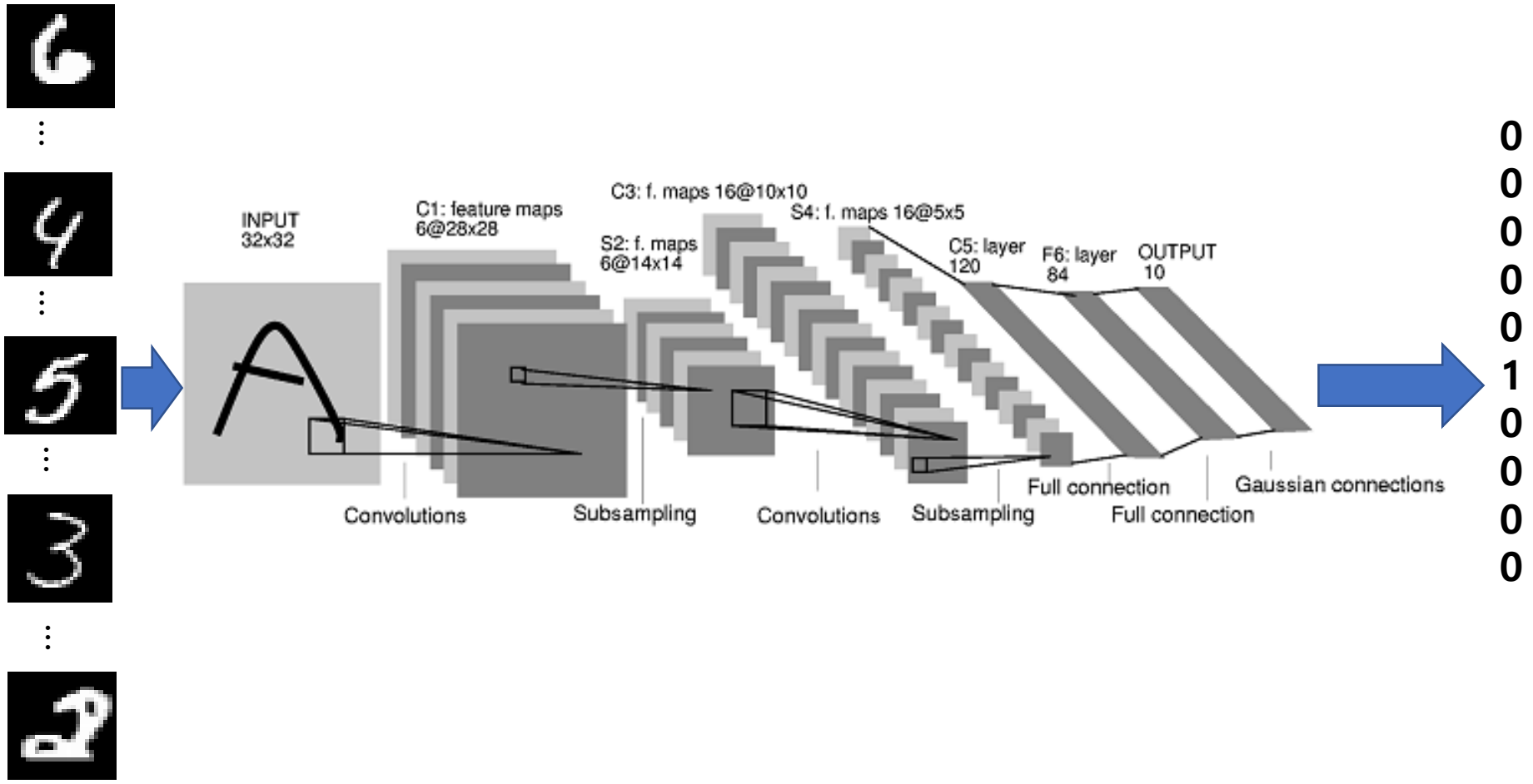
MNIST DATASET : Training Set – 60,000
Test Set – 10,000



02 MNIST Classification

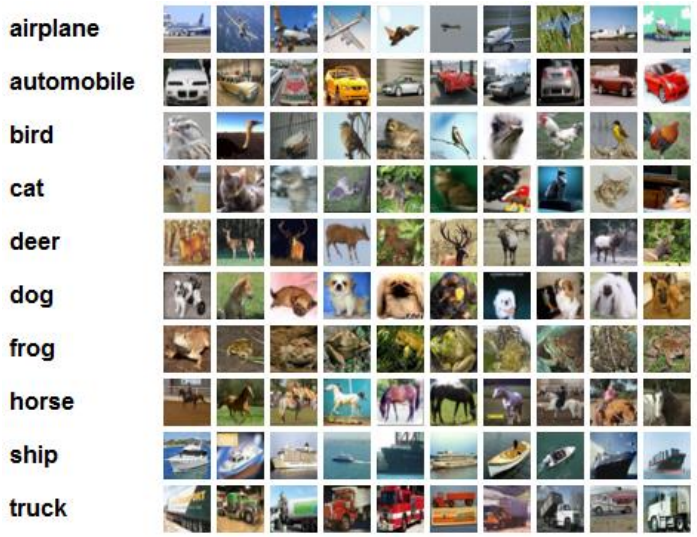
Code Vision

MNIST DATASET : Training Set – 60,000
Test Set – 10,000

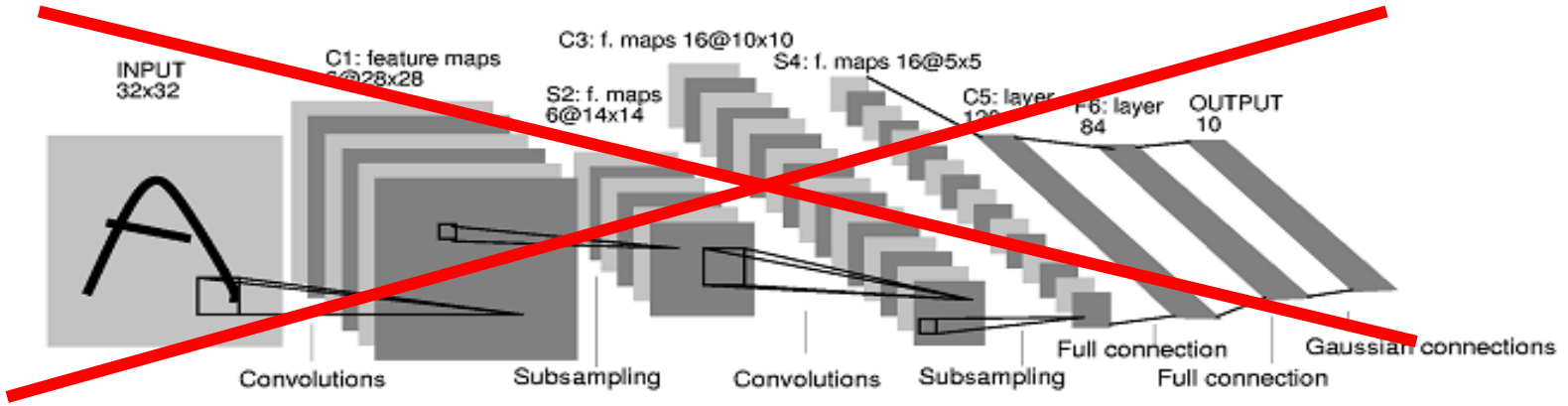


MNIST Classification 실습

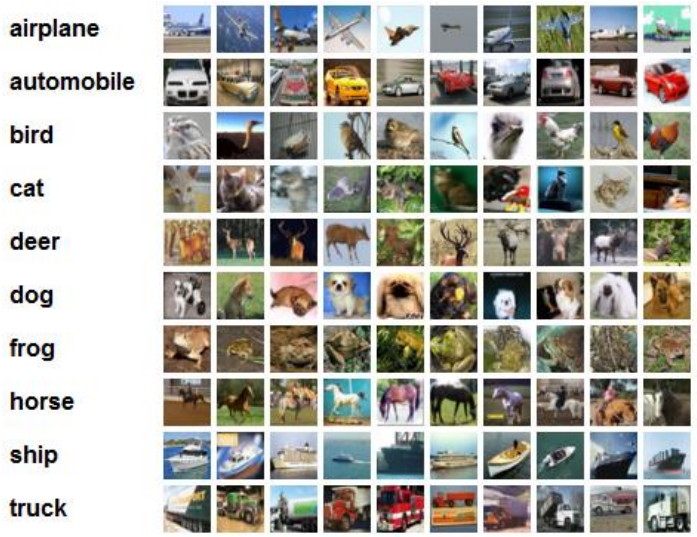
03 Image Classification



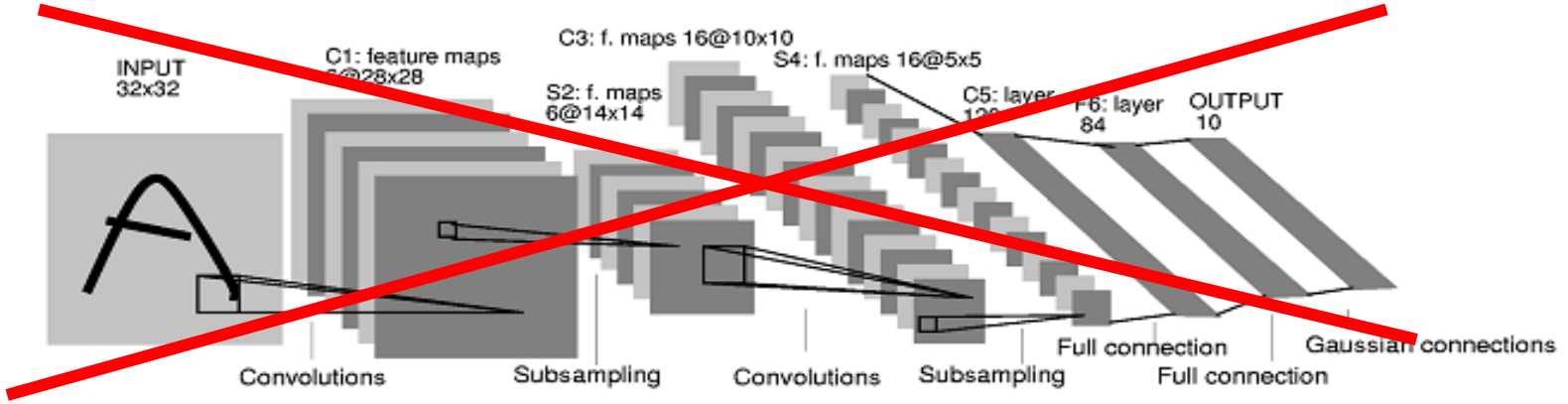
- 기존 Lenet-5의 문제점
  ⇒ 파라미터 수가 적음
  ⇒ 파라미터 수를 늘릴 경우
  Activation function에 문제



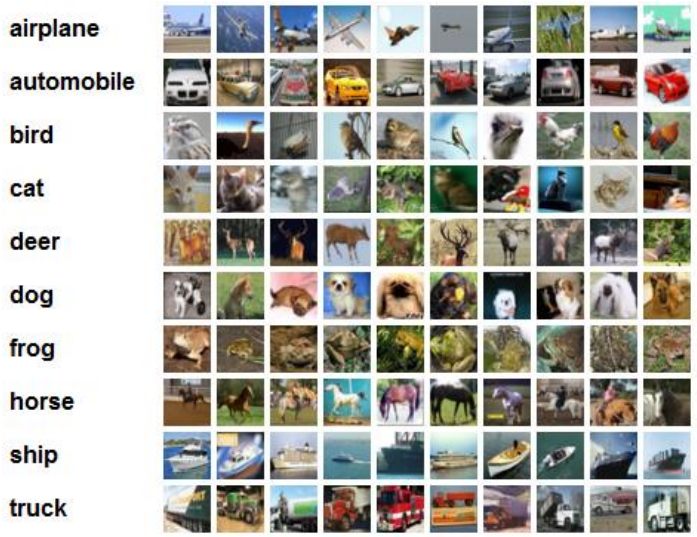
03 Image Classification



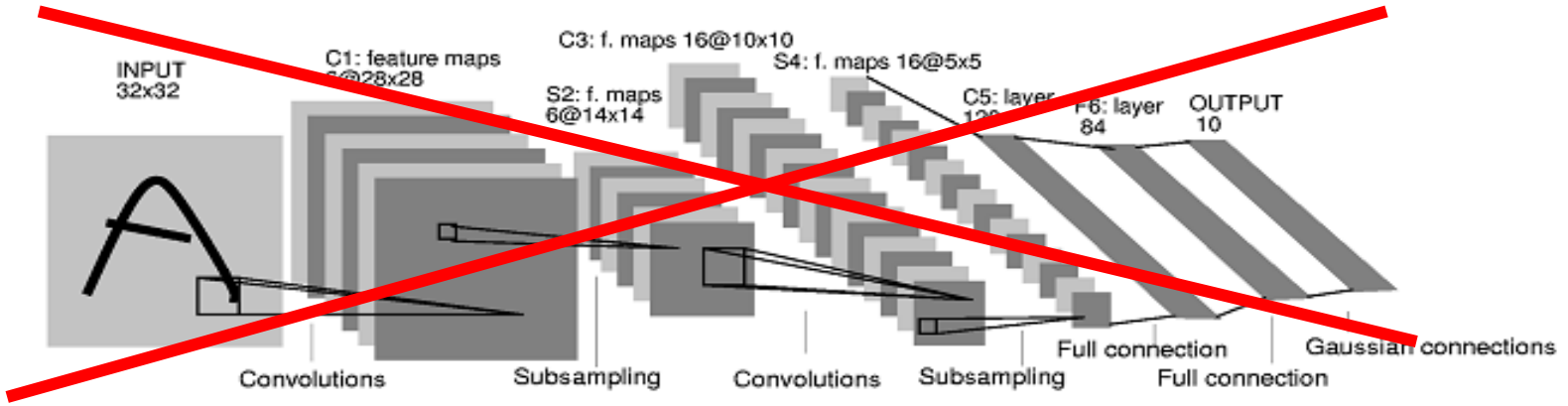
- 파라미터가 적은 문제
  => 실제 물체의 경우 문자와 달리 복잡성이 훨씬 높게 때문에 기존의 Lenet-5 모델의 적은 수의 파라미터로는 실제 물체 이미지의 정보를 담을 수 없음



03 Image Classification



- **Activation Function 문제**
  => 기존 Sigmoid Function의 경우 네트워크의 층수를 늘리면 Vanishing Gradient 현상이 발생함



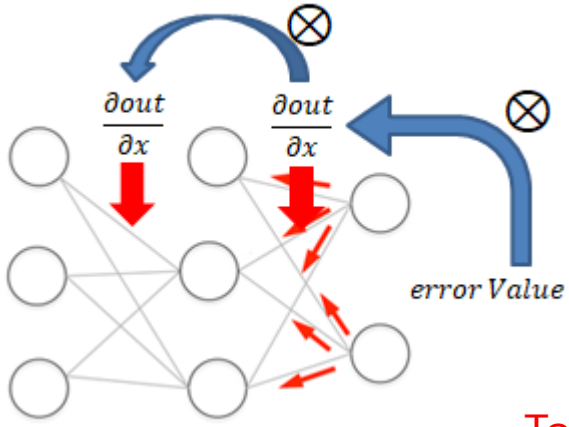
03 Image Classification

Code Vision

- **Activation Function 문제**
  => 기존 Sigmoid Function의 경우
  네트워크의 층수를 늘리면 Vanishing Gradient 현상이 발생함

- Back-propagation

⇒ 학습과정을 통해 weight가 수정되는 과정

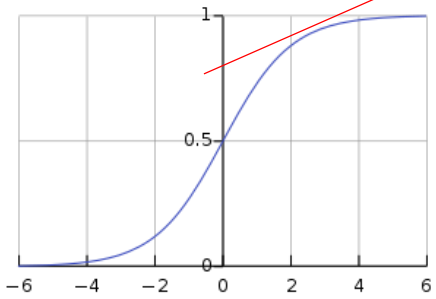


- 층이 깊어지면

$$w = w - \frac{\partial out}{\partial x} \times \frac{\partial out}{\partial x} \times \cdots$$

- $0 < \frac{\partial out}{\partial x} < 1$일 경우 반복적으로 0< <1 값이 곱해지면 gradient값이 0에 가까워져 vanishing gradient 현상이 발생

Too low!!



- Sigmoid function -

03 Image Classification

LeNet (1990, Yann LeCun , just for NMIST)

이후 개선된 네트워크 (이 후 네트워크 부터 이미지 구분 가능)

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

ZF Net (2013, Matthew Zeiler-Rob Fergus , winning in ILSVRC 2013) => modifying AlexNet's hyper-parameter and increasing the size of convolution layer

GoogLeNet (2014, Szegedy in Google, winning in ILSVRC 2014)

ResNet (2015, Kaimming He in microsoft, winning in ILSVRC 2015)
=> first introducing residual framework

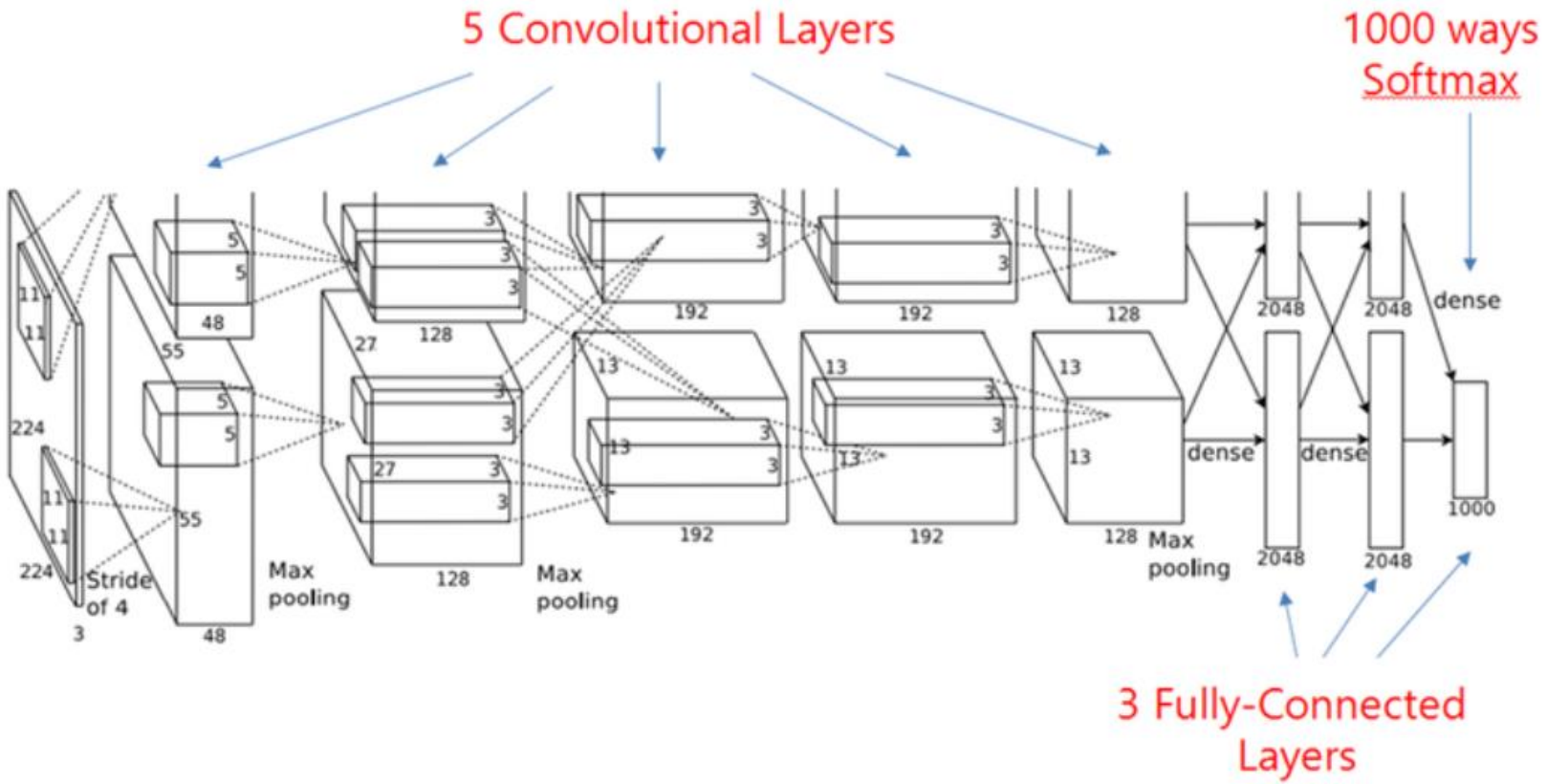
03 Image Classification

Code Vision

LeNet (1990, Yann LeCun , just for NMIST)

이후 개선된 네트워크 (이 후 네트워크 부터 이미지 구분 가능)

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

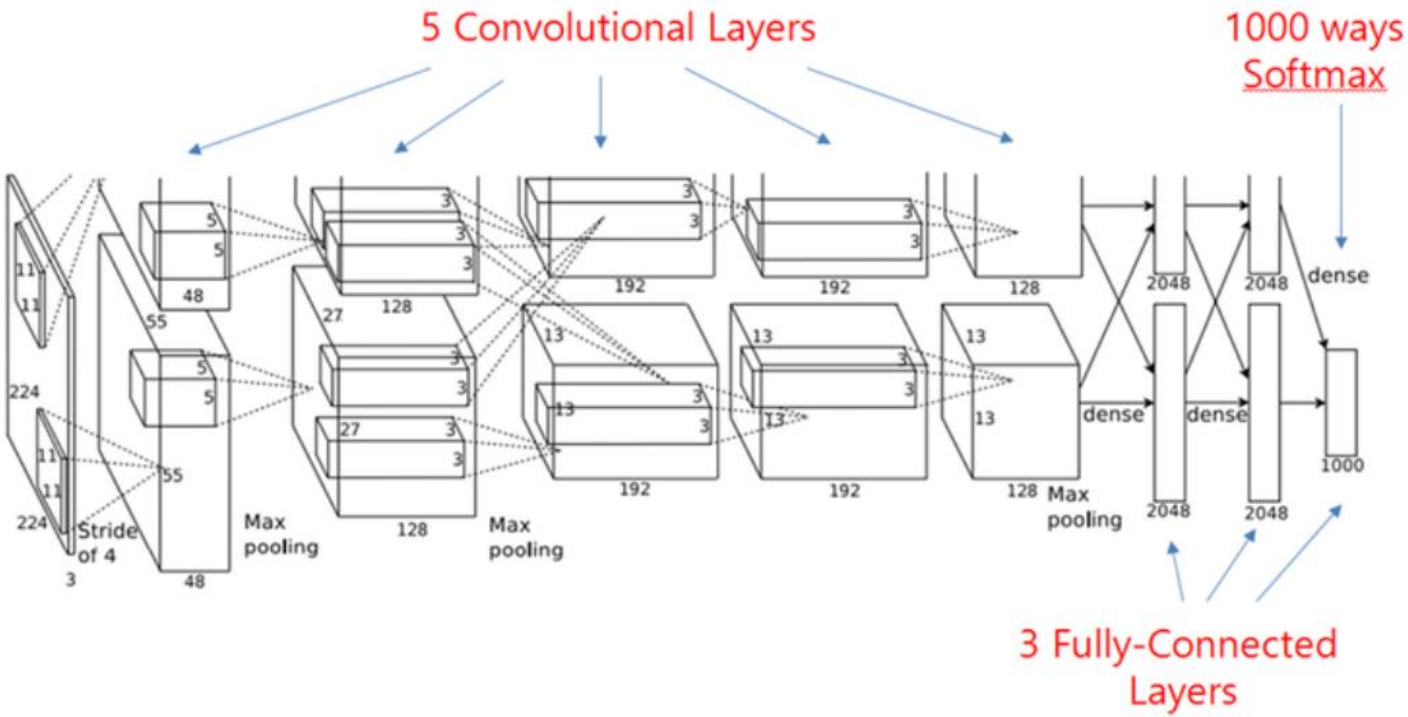


03 Image Classification

Code Vision

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

- 마지막 Classification 층에 softmax를 사용
- 숫자와 달리 이미지는 RGB 3개의 채널을 가짐
- Activation Function : Sigmoid => Relu (Vanishing 문제 해결)
- Overlapped pooling 을 새로 도입
- Over-fitting 문제 해결



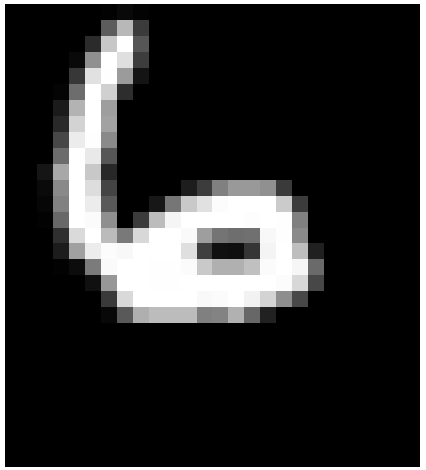
03 Image Classification

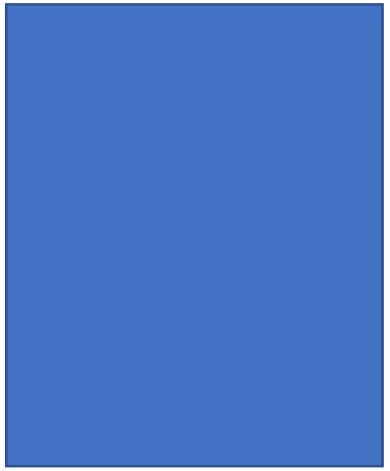
Code Vision

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

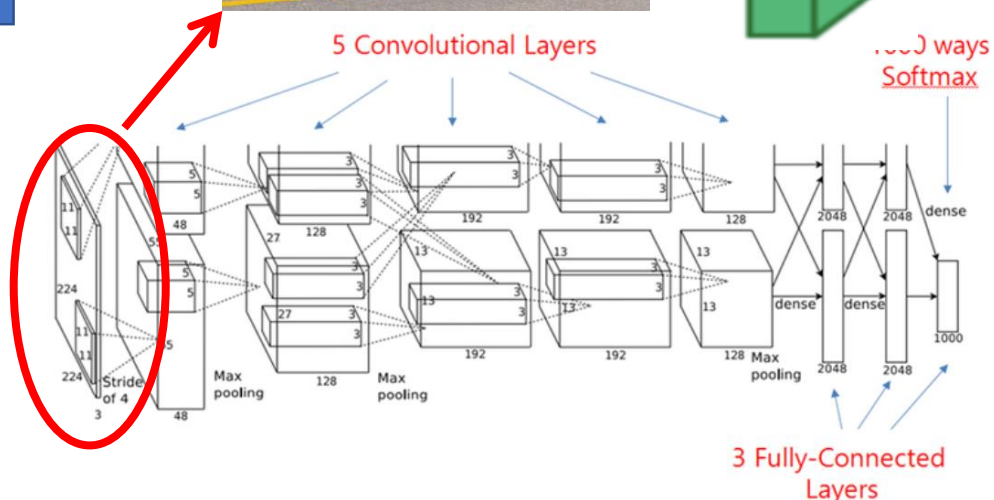
- 숫자와 달리 이미지는 RGB 3개의 채널을 가짐

흑백 : 가로 x 세로 x(1채널 흑백으로만)





흑백 : 가로 x 세로 x(3채널)
모든 색은 : R(빨강)+G(초록)+B(파랑)의 조합



03 Image Classification

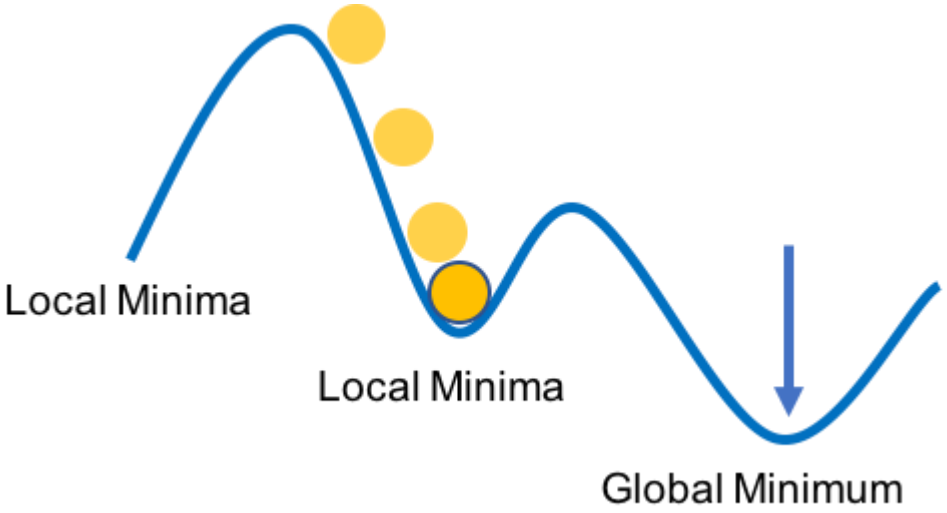
Code Vision

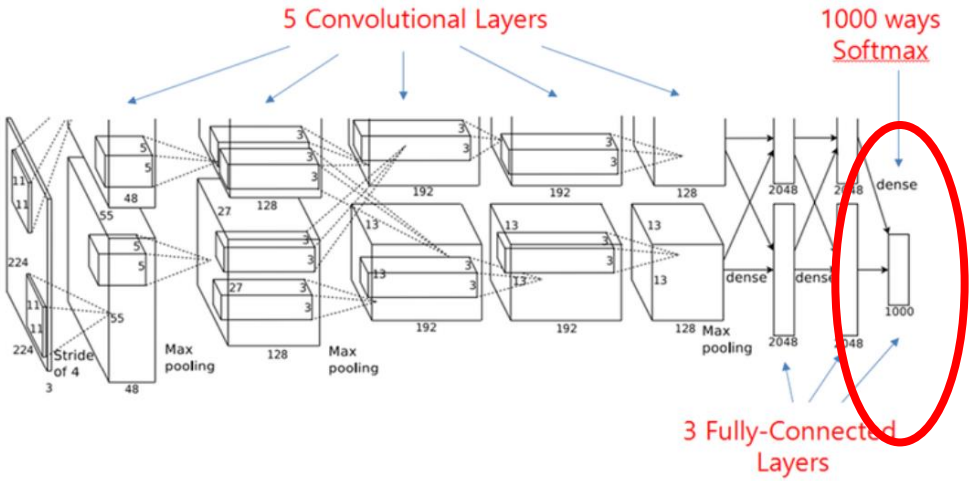
AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

- 마지막 Classification 층에 softmax를 사용
  => Loss 함수에서 Mean Square Error를 쓰면 Global Minima로 가기 힘듬(최적화가 힘듬) => Cross Entropy 식을 사용

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$



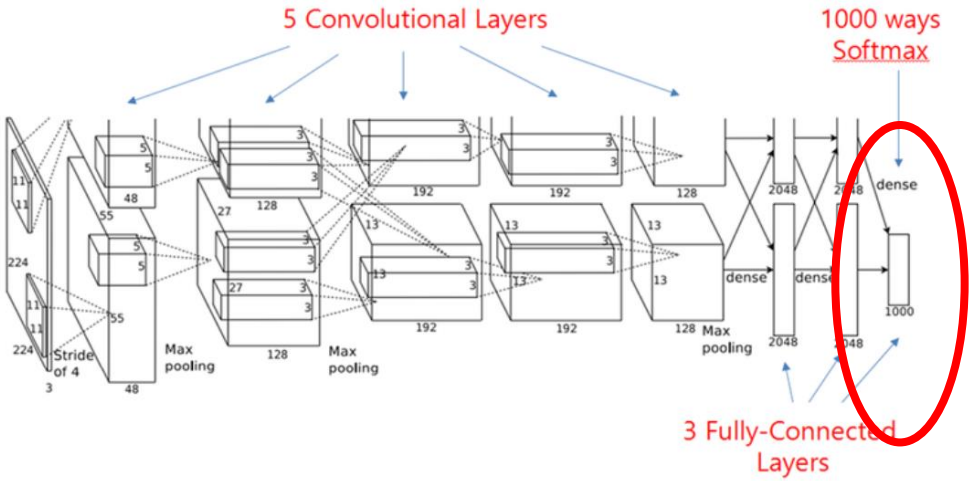


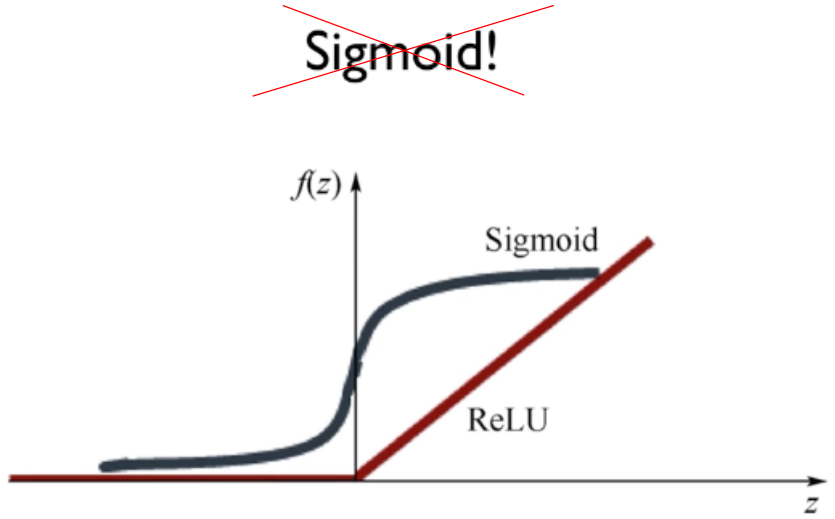
03 Image Classification

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

- Activation Function : Sigmoid => Relu (Vanishing 문제 해결)

- Relu 설명

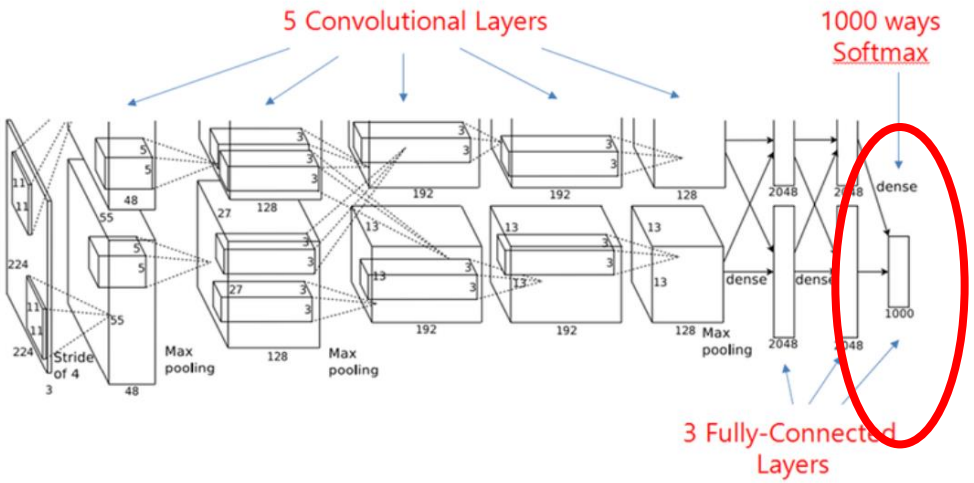


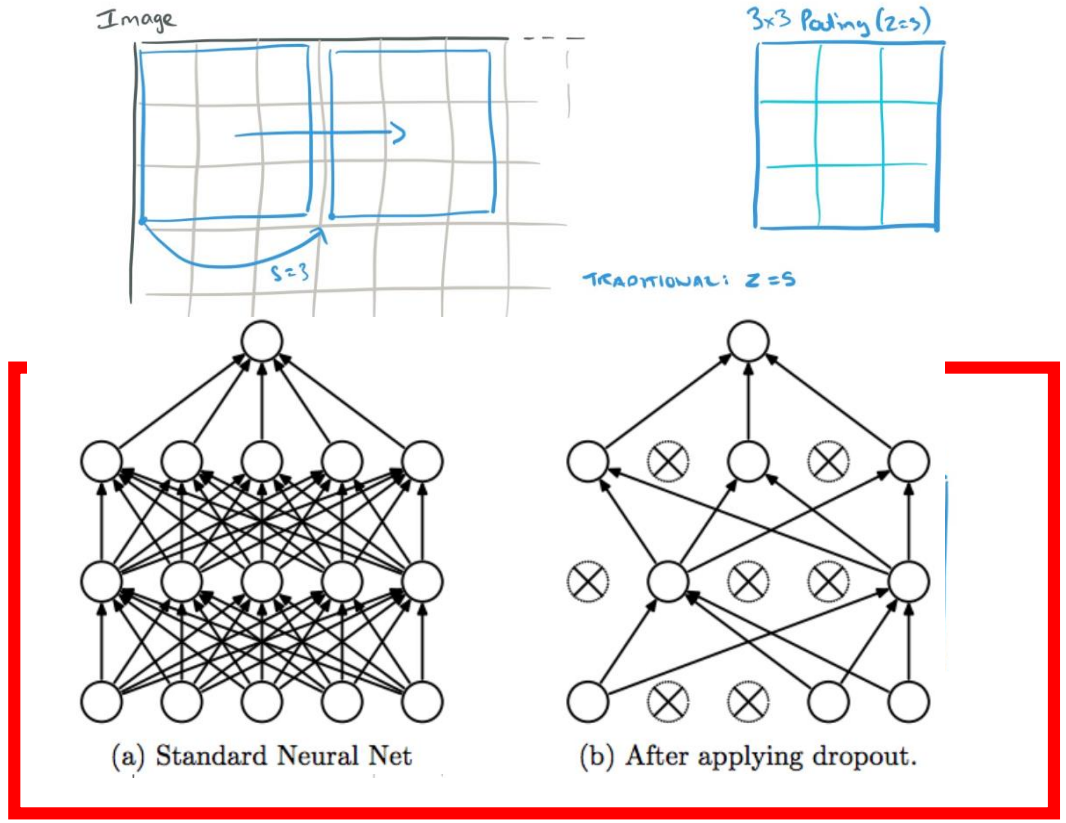


03 Image Classification

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

- Overlapped pooling 을 새로 도입(최근에 쓰지 않음)



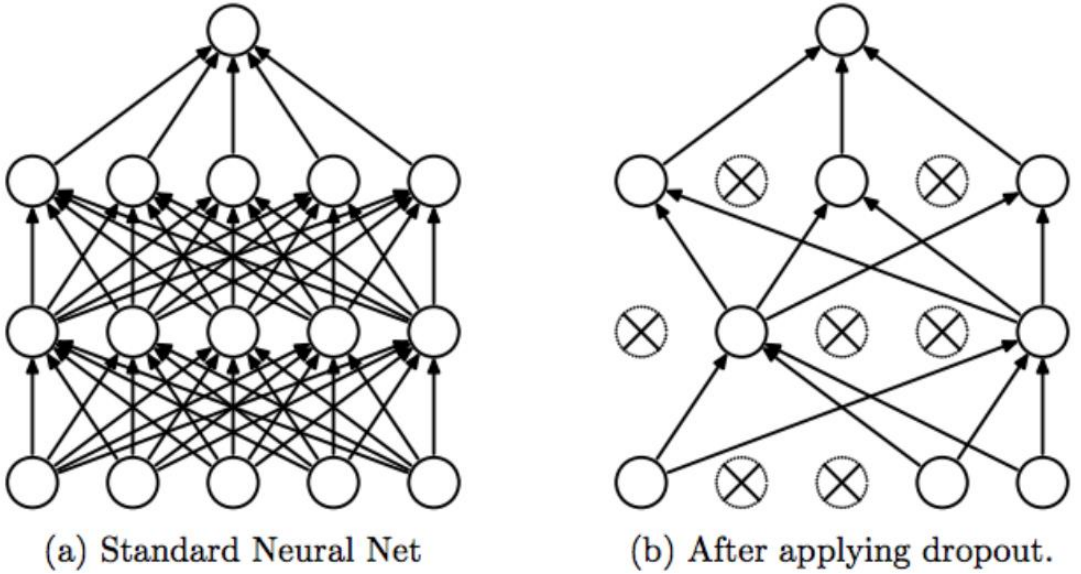


03 Image Classification

Code Vision

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

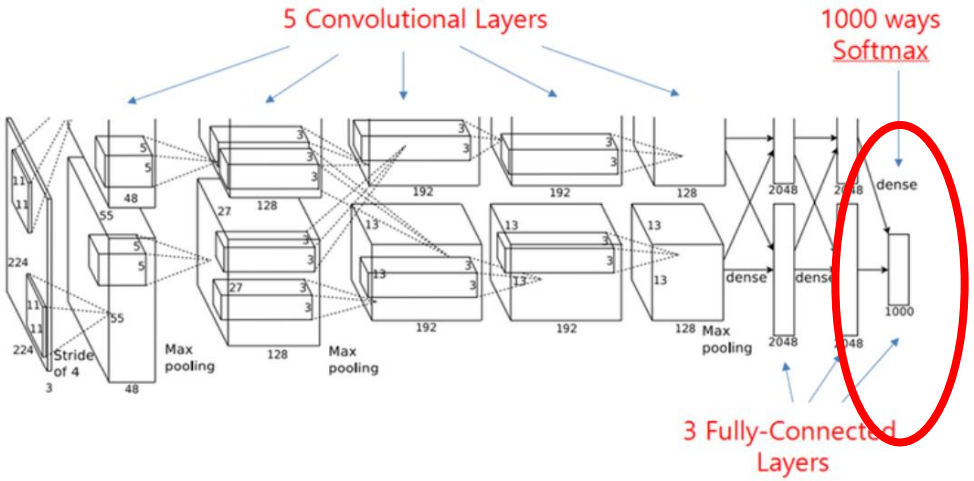
- **Dropout(최근에 쓰지 않음)**



(a) Standard Neural Net

(b) After applying dropout.

- **Dropout 설명**



지도 학습 (Supervised Learning)

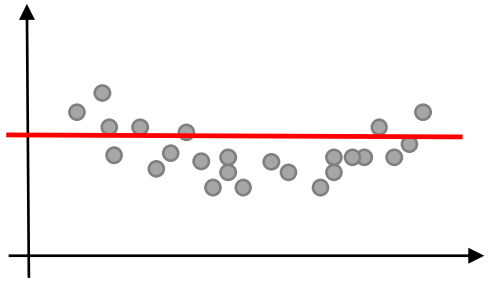
회귀의 과적합(overfitting)과 부적합(underfitting)

과적합

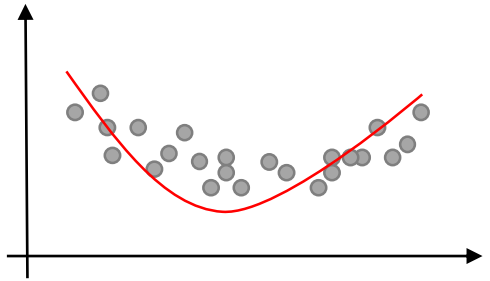
지나치게 복잡한 모델(함수) 사용

부적합

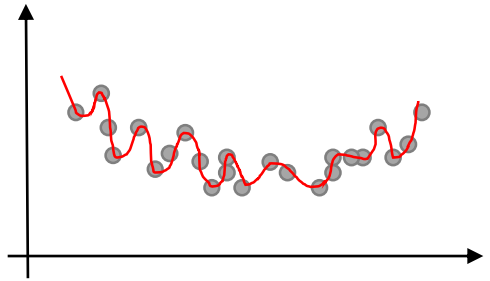
지나치게 단순한 모델(함수) 사용



부적합(underfitting)



정적합(good fitting)



과적합(overfitting)

지도 학습 (Supervised Learning)

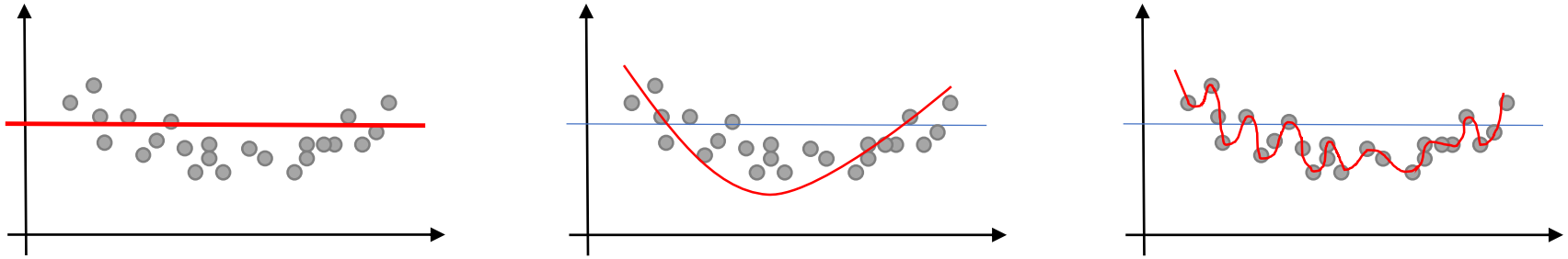
회귀의 과적합(overfitting) 대응 방법

모델의 복잡도(model complexity)를 성능 평가에 반영

=> 모델의 복잡도가 낮아지는 방향으로 최적화가 됨

목적함수 = 오차의 합 + (가중치)*(모델 복잡도)

벌점(penalty) 항



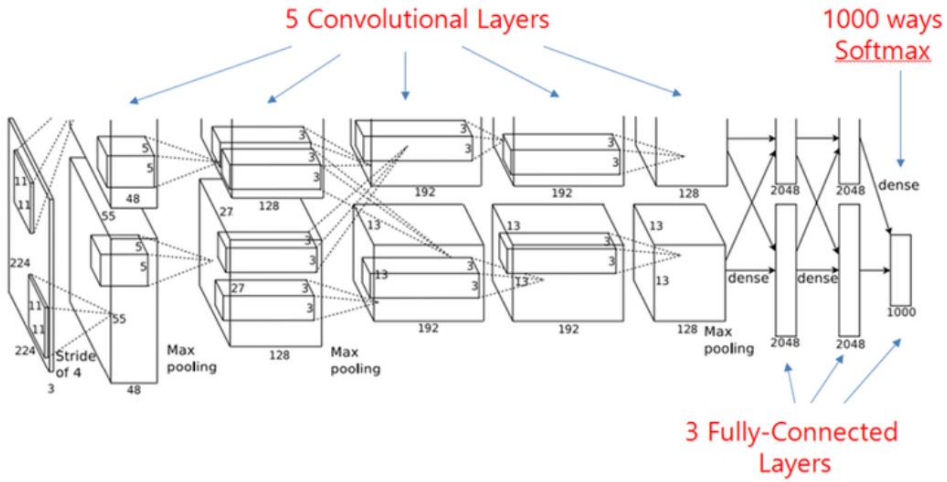
부적합(underfitting) 정적합(good fitting) 과적합(overfitting)

03 Image Classification

Code Vision

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

- **AlexNet 구조**



input :224 x 224 x 3 (RGB)
Layer 1 : convolution layer (96 kernels of size 11 x 11 x 3 with a stride of 4 pixel
Layer 2 : subsampling layer (256 kernels of size 5 x 5 x 48 , overlapping max pooling)
(respons-normalized -> max pooling)
Layer 3 : subsampling layer (384 kernels of size 3 x 3 x 256 ,overlapping max pooling)
(respons-normalized -> max pooling)
Layer 4 : convolution layer (384 kernels of size 3 x 3 x 192)
Layer 5 : convolution layer (256 kernels of size 3 x 3 x 192)
Layer 6 , 7 :fully connected layer

Activation function : Rectified Linear Units is applied to all layers.
Drop out is applied to all fully connected layers

03 Image Classification

Code Vision

AlexNet (2012 , Krizhevesky-Hinton , winning in ILSVRC 2012)

Cifar-10 (Image) Classification

Cifar-10 [32 x 32 x 3 image , 10 class]
Training set :40000 set
validation set :10000 set
test set :10000 set

02 MNIST Classification

Code Vision

- Progress : Neural Network

Xor Classification → Digit Classification(NMIST) → Convolution Neural Network Image Classification (Cifar-10)

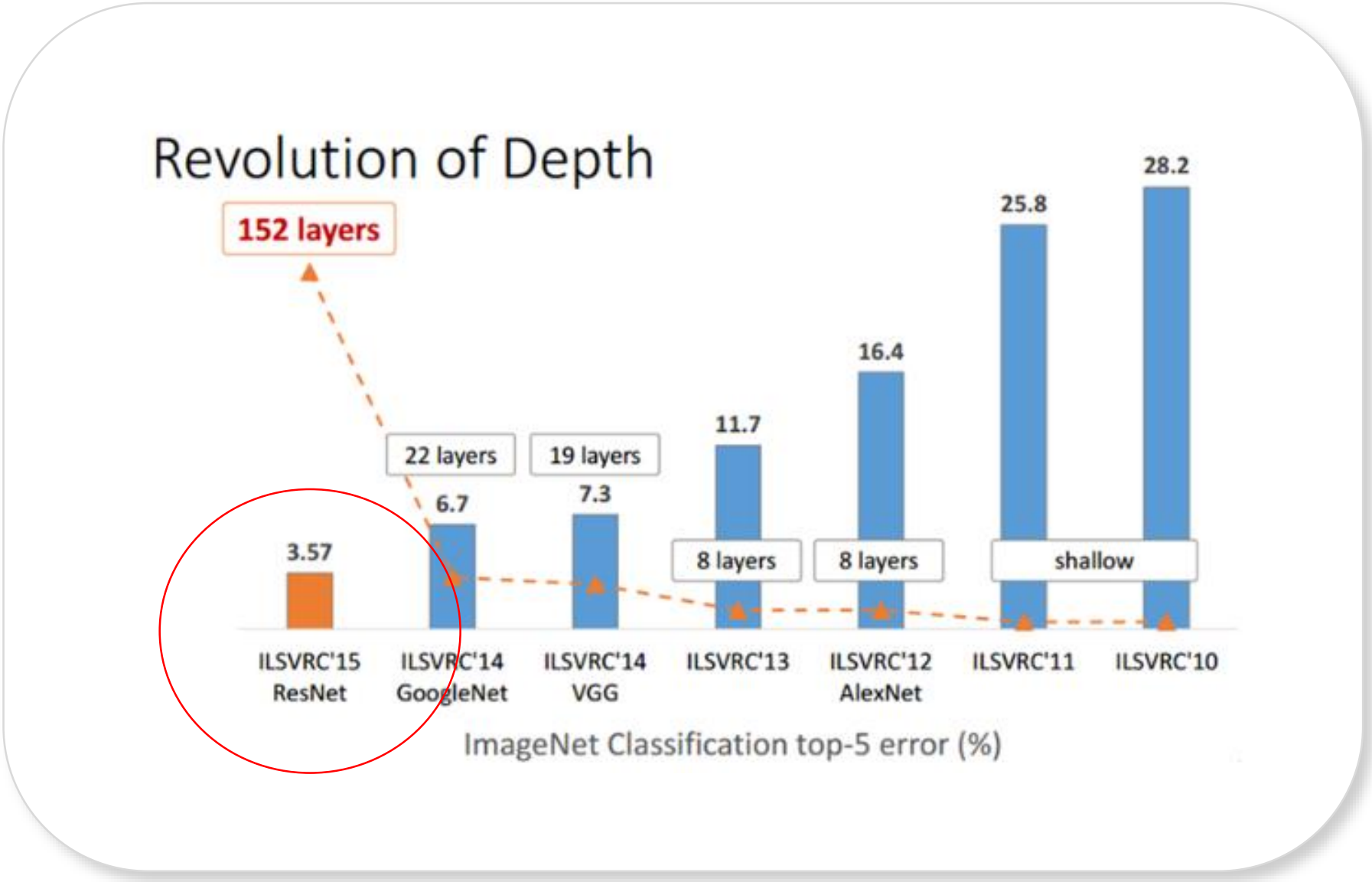
Image Classification 실습

01 Residual Network 소개

Code Vision

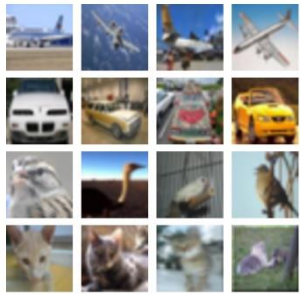
- 저자 Kaiming He, 2015.09
- 2015년 9월 ImageNet Large Scale Visual Recognition Challenge에서 Image Classification에서 압도적으로 우승
- 2014년 우승 네트워크의 오류 율을 절반으로 줄임
- Residual Network의 원리와 구조 소개

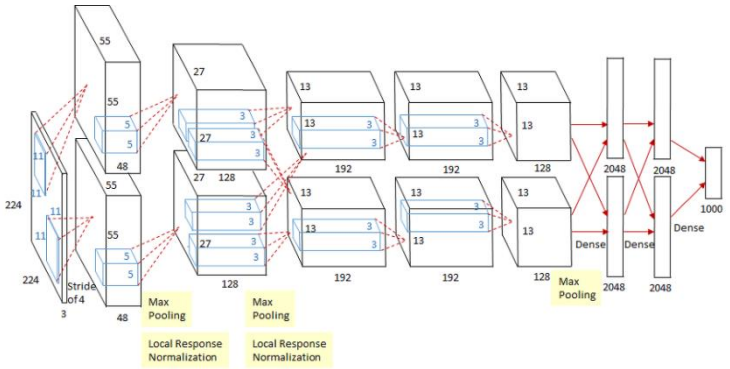
01 Residual Network 소개



01 Residual Network 소개

- Image Classification : 간단한 Task

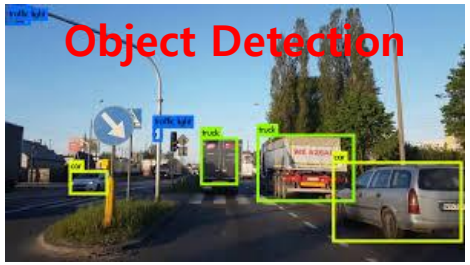




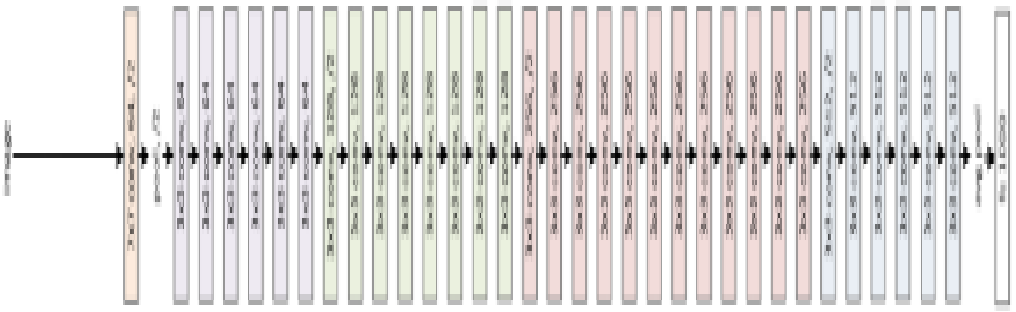
얕은 층

7-layer

- **Object Detection, Human Pose Detection, Segmentation: 복잡한 Task**



깊은 층 34-layer



점차 Deeper한 네트워크가 요구되는 상황!!

01 Residual Network 소개

Code Vision

- Back-propagation

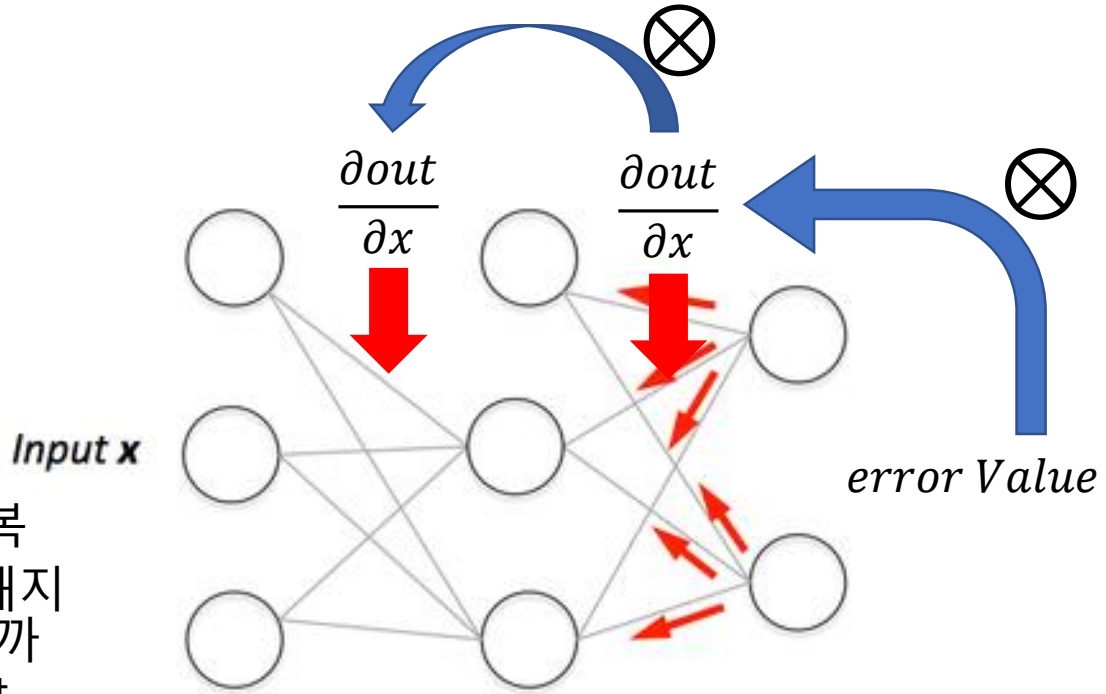
⇒ 학습과정을 통해 weight가 수정되는 과정

- 층이 깊어지면

$$w = w - \frac{\partial out}{\partial x} \times \frac{\partial out}{\partial x} \times \cdots$$

- $0 < \frac{\partial out}{\partial x} < 1$일 경우 반복적으로 0< <1 값이 곱해지면 gradient값이 0에 가까워져 vanishing gradient 현상이 발생

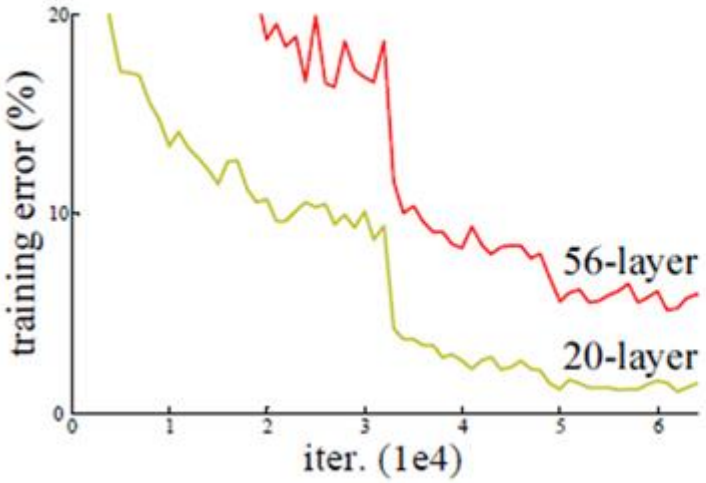
- >1일 경우 exploding gradient 현상이 발생

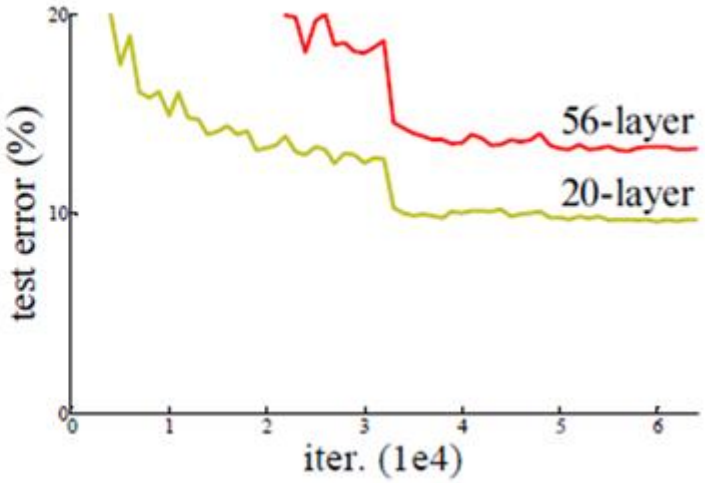


01 Residual Network 소개

Code Vision

- Vanishing Gradient 현상에 대한 실험 결과
  - Cifar-10 데이터베이스로 Image classification 성능 비교 진행
  - Training error에서 20-layer보다 56-layer의 오차가 더 커져 결과가 더 안 좋음
  - Test error의 경우도 위와 마찬가지





02 Residual Network 원리

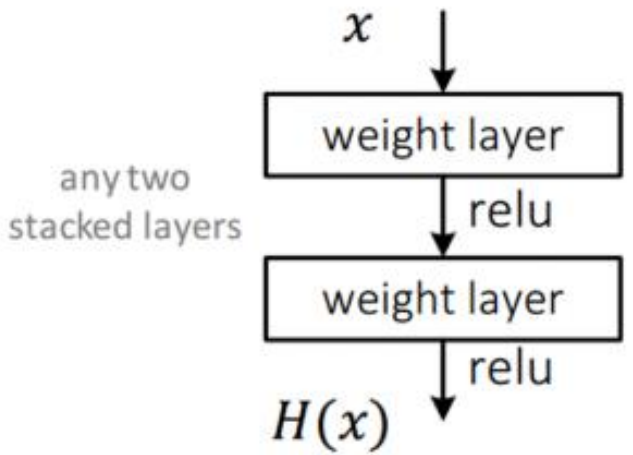
- Residual Network 와 일반 Network와의 차이
  - 일반적인 Network는 학습을 통해 최적의 $H(x)$를 얻어야 하고, 그에 맞춰 weight 값이 결정되어야 함

  - Residual Network는 $H(x) - x$를 얻는 것으로 목표를 수정함

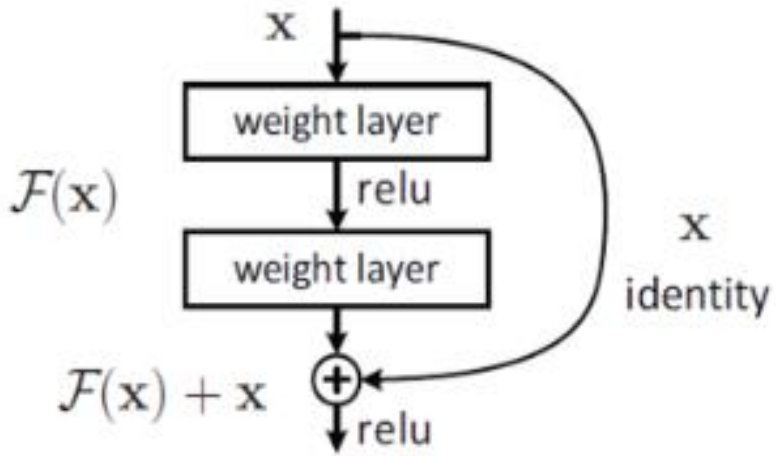
  - 여기서 $F(x) = H(x) - x$라면, 결과적으로 $H(x) = F(x) + x$가 된다.

  - 입력에서 바로 출력으로 연결되는 shortcut(지름길)이 생기게 됨

  - 아래의 그림이 Residual Network의 기본 단위가 된다.



- 일반 Network -



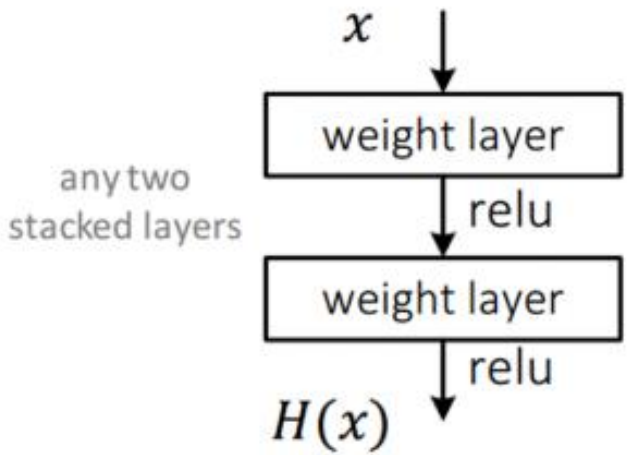
- Residual Network -

02 Residual Network 원리

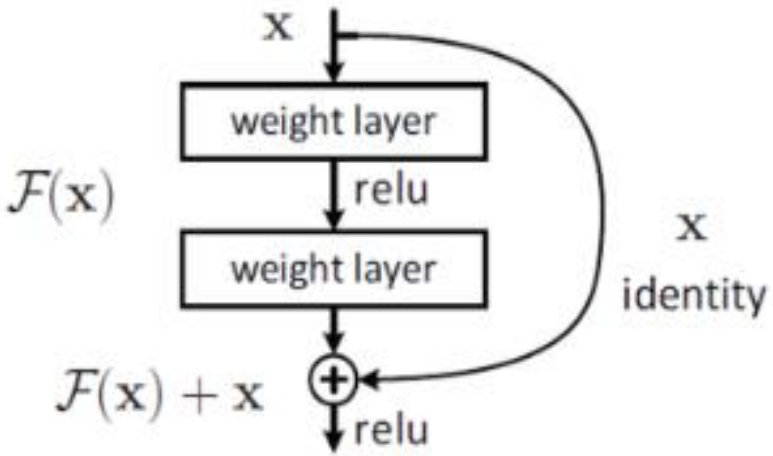
- Residual Network의 의미와 효과
  - 일반적인 경우 $H(x)$의 학습 값이 0~1사이에서 랜덤으로 요구되어지기 때문에 (ex 맞다 1, 틀리다 0) 학습이 어려움

  - Residual Network의 경우
    $$\mathrm{F}(x) = H(x) - x \Rightarrow 0$$
    가 되어야 하므로 학습방향이 0으로 결정 되어 있어, 입력의 작은 움직임(fluctuation)을 쉽게 검출할 수 있다.

  - 이런 의미에서 $F(x)$의 작은 움직임, 즉 나머지(residual)을 학습한다는 관점에서 Residual Network라고 명칭을 하게 된다.



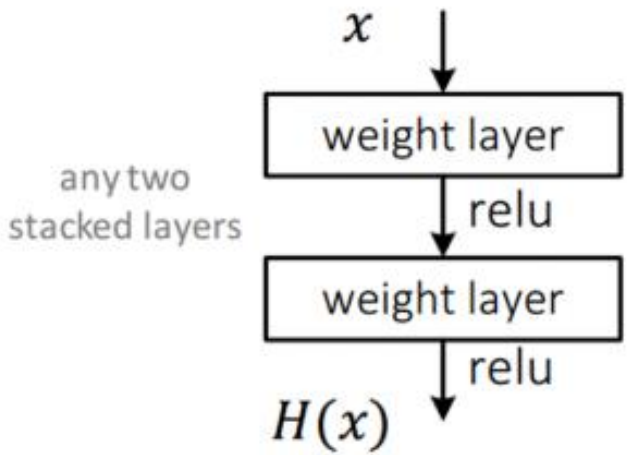
- 일반 Network -



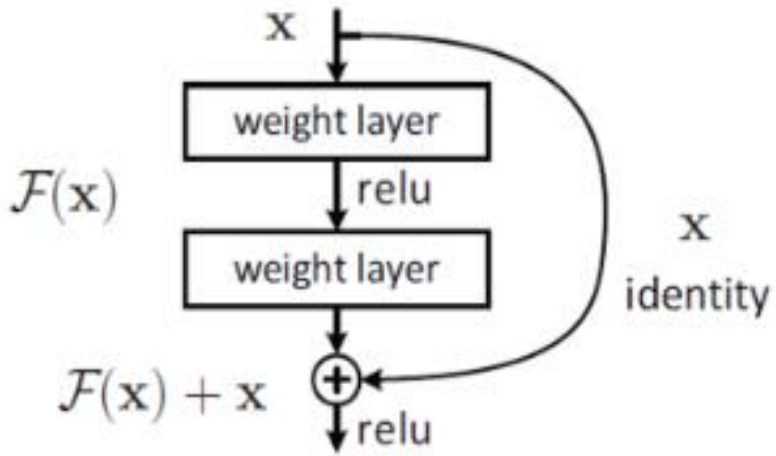
- Residual Network -

02 Residual Network 원리

- Residual Network의 의미와 효과
  - x가 그대로 출력에 연결이 되기 때문에 파라미터 수에 영향이 없음
  - shortcut(지름길)을 통한 연산량 증가는 없음
  - 몇 개의 layer를 건너 뛰면서 입력과 출력이 연결이 되므로 forward나 backward path가 단순해 짐



- 일반 Network -

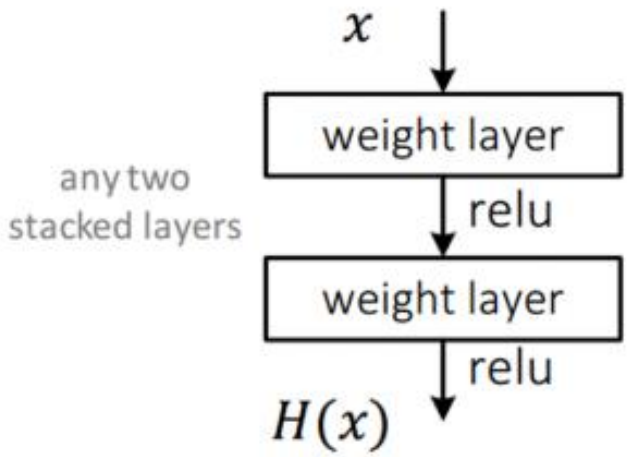


- Residual Network -

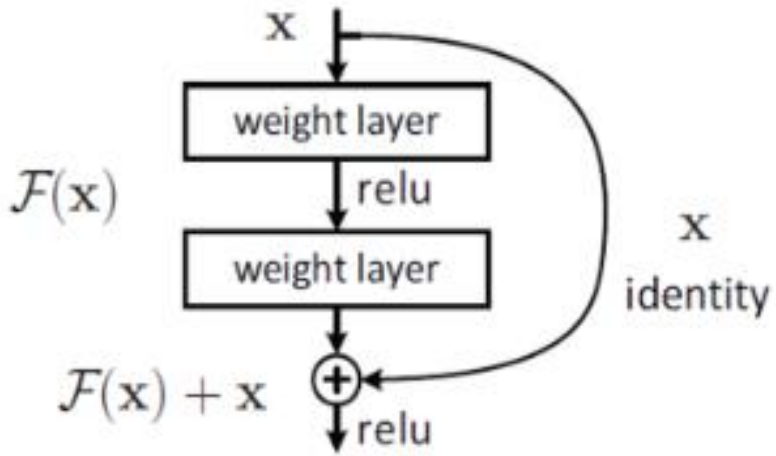
03 Residual Network 원리

- Residual Network의 의미와 효과 정리
  - 깊은 망에서도 최적화가 잘 이루어 진다.

  - 파라미터 수(인간에게는 뉴런)를 늘리면서도 Vanishing Gradient 현상을 해결하여, 정확도를 개선하며, 보다 복잡한 Task(Human Pose Detection, Object Detection등..)을 수행할 수 있게 된다.



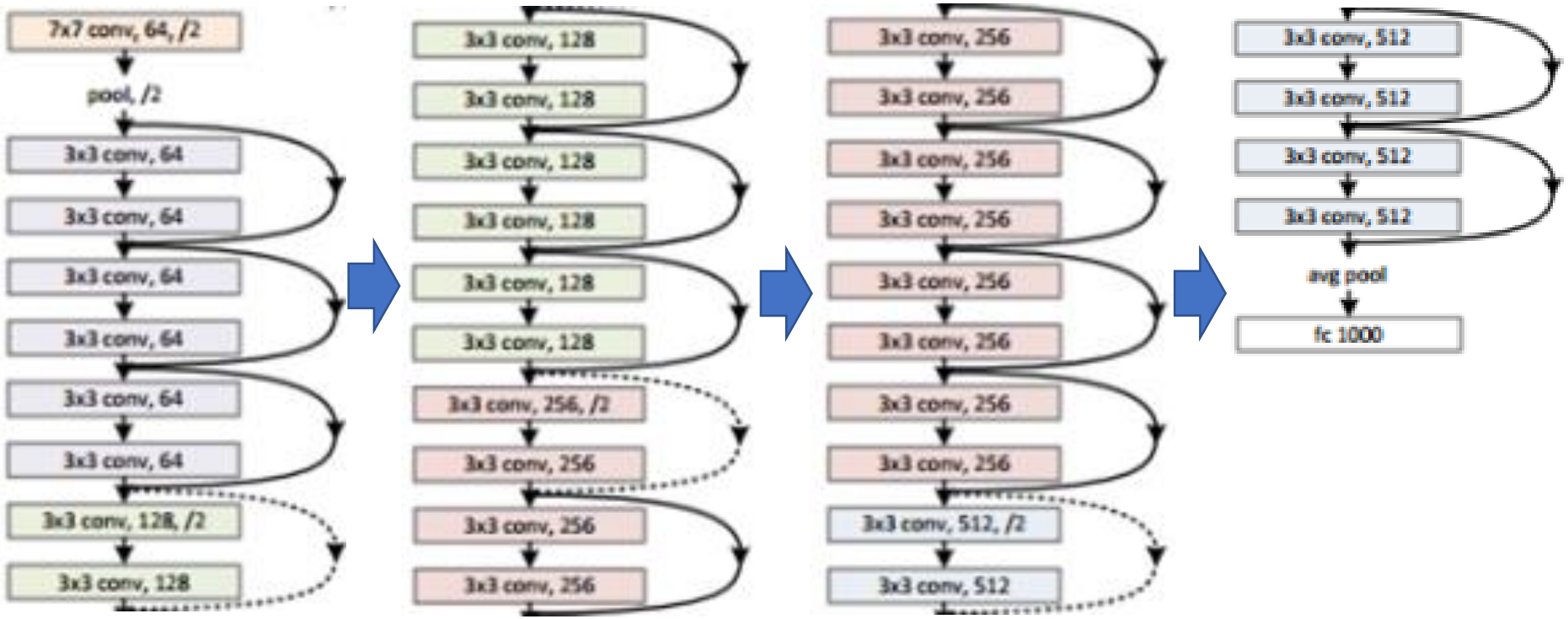
- 일반 Network -



- Residual Network -

03 Residual Network 구조

- 32-layer resnet
- Convolution Filter 은 3x3을 사용
- Fully Convolution 층을 사용하지 않음
- Dropout 사용하지 않음



03 Residual Network 구조

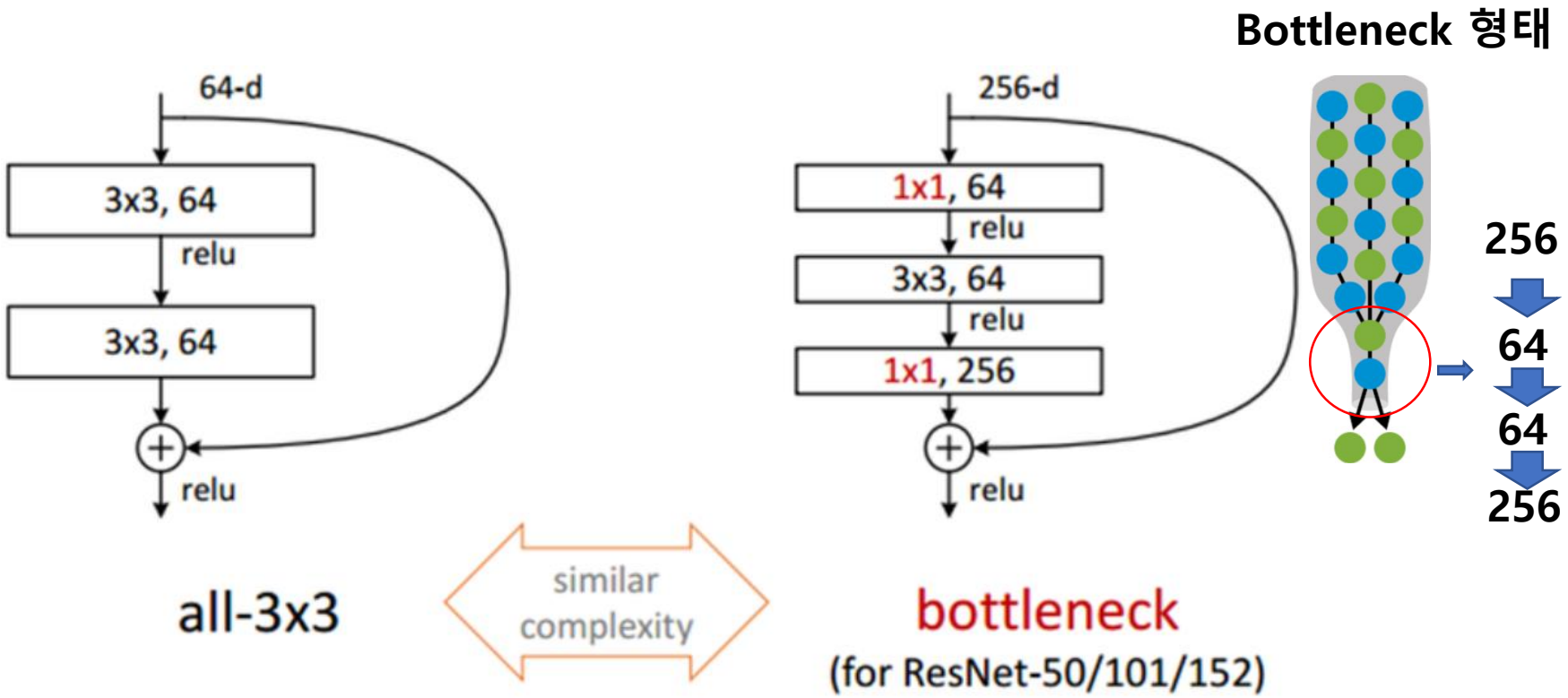
Code Vision

- Resnet의 종류 및 구성
- 18-layer, 34-layer, 50-layer, 101-layer, 152-layer

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

03 Residual Network 구조

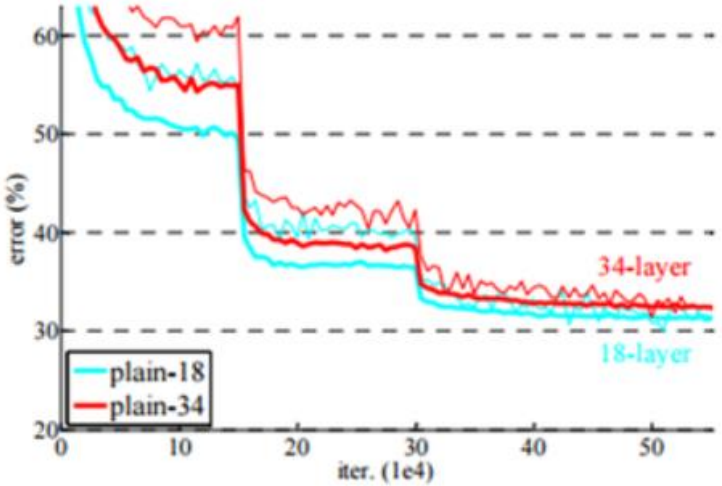
- 일반적인 all-3x3 구조 : 18-layer, 34-layer
- Bottleneck 구조 : 50-layer, 101-layer, 152-layer
- 50-layer이상의 경우 연산 량이 대폭 증가하므로, Bottleneck 구조를 활용하여 연산 량을 절감시킴

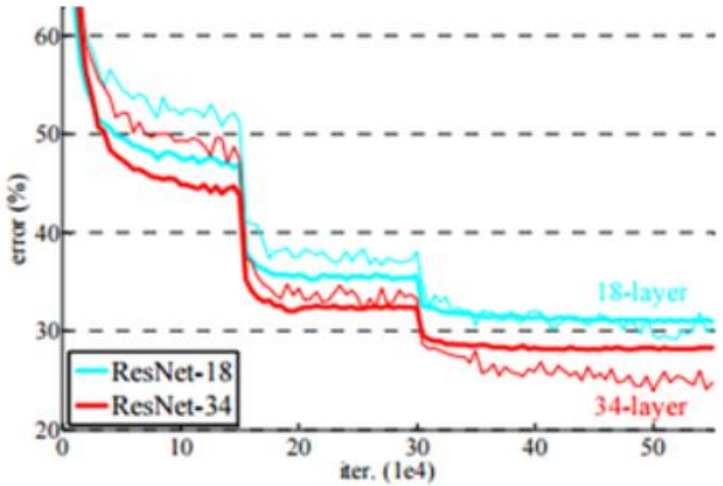


04 Residual Network 성능

- 일반적인 Network 성능 비교 : 18-layer > 34-layer
- Residual Network 성능 비교 : 18-layer < 34-layer
- 수렴 속도 : 일반적인 Network < Residual Network

Top-1 error	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**





ResNet 실습