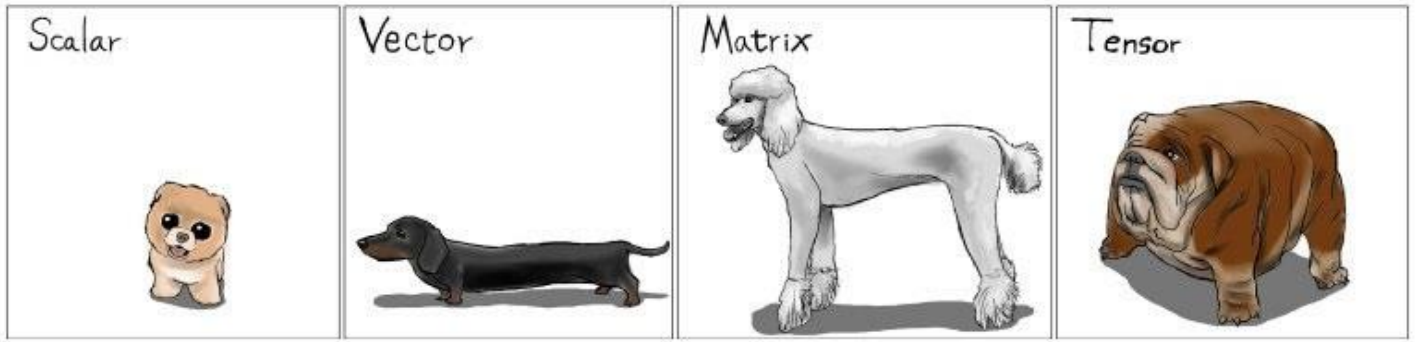
M2. 딥러닝 기본

인공신경망 기본개념 및 MNIST 과제소개

- FNN
- MNIST dataset
- 경사하강법
- Backpropagation
- Hyperparameter
- Train/Val/Test
- Overfitting/Underfitting

잠깐! 선형대수 기본상식

Tensor는 scalar, vector, matrix를 포함한 모든 dimension을 표현할 수 있다



1

$$\begin{bmatrix} 2.3 \\ 4.0 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & -2.0 \\ 5.8 & 10.2 \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

출처 : Image by Karl Stratos, medium.com

잠깐! 선형대수 기본상식

Shape : 각 차원의 element 수

1

Shape []

$$\begin{bmatrix} 2.3 \\ 4.0 \end{bmatrix}$$

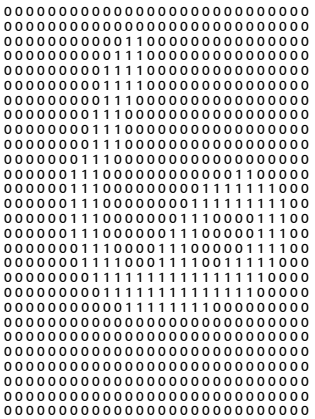
Shape [2]

$$\begin{bmatrix} 0.1 & -2.0 \\ 5.8 & 10.2 \end{bmatrix}$$

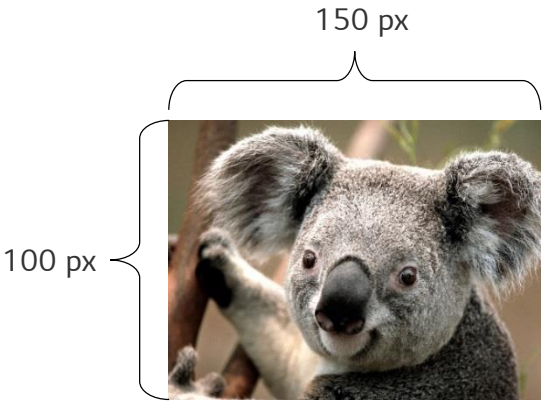
Shape [2,2]

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

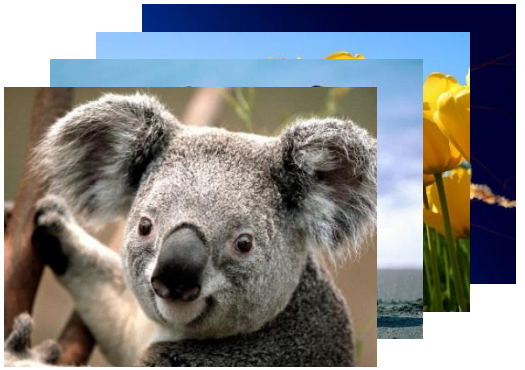
Shape [2,2,2]



Shape [28,28]



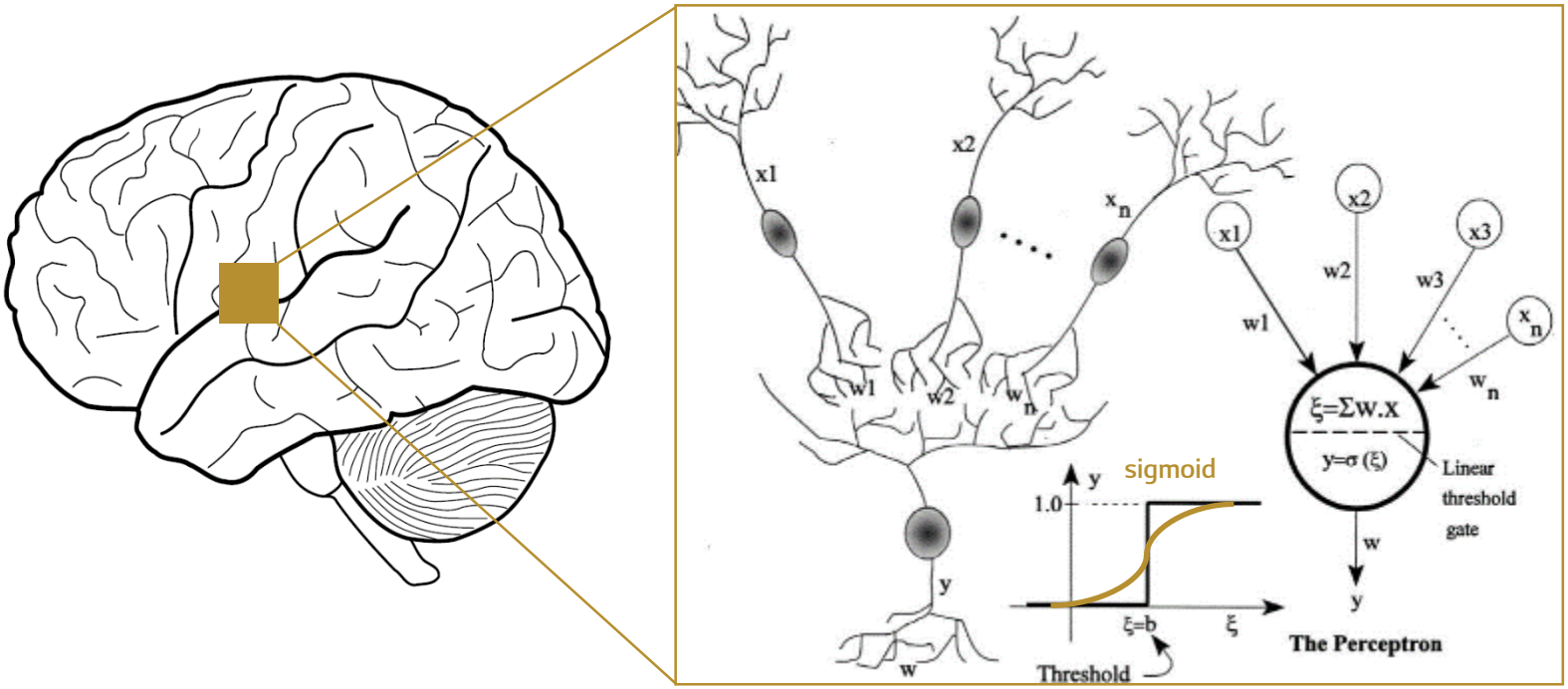
Shape [100,150,3]



Shape [4,100,150,3]

인공신경망 개요

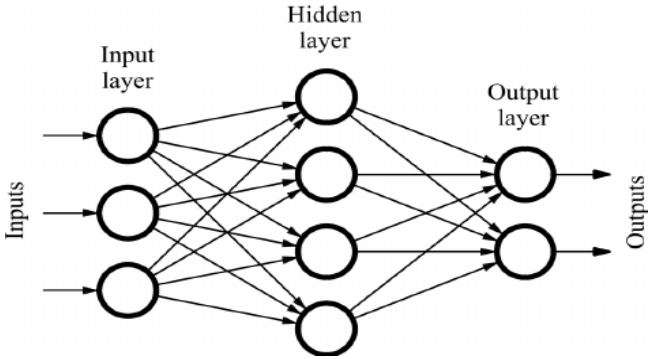
두뇌 세포를 모사한 인공신경망(ANN; Artificial Neural Network) 알고리즘



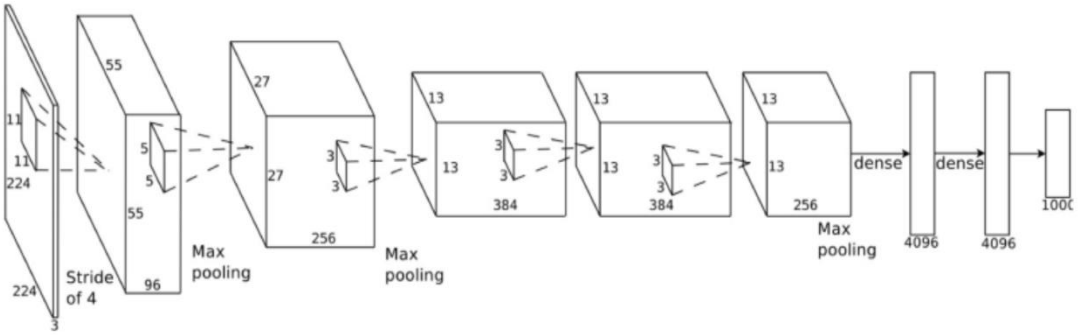
구조에 따른 인공신경망 분류

인공 뉴런 연결 방식에 따른 딥러닝 모델 기본 아키텍처 (FNN / CNN / RNN)

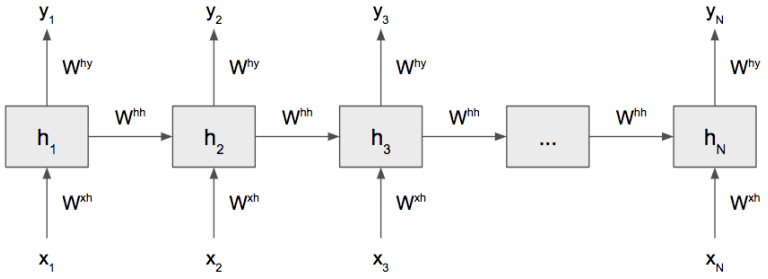
Feedforward Neural Network



Convolutional Neural Network



Recurrent Neural Network



LG CNS에 입사할 것인가?

성장 가능성이
있는가?

팀원 중에
이상한 사람이 있나?

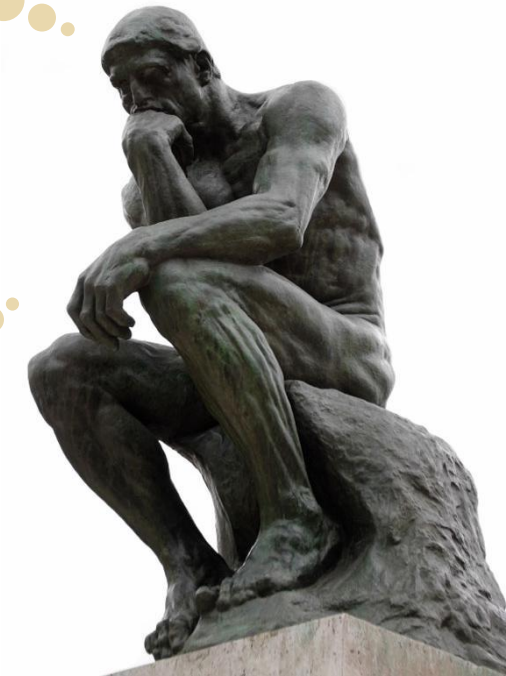
출퇴근이 편한가?

연봉은 적절한가?

조직문화가
수평적인가?

내가 잘 할 수 있는
업무인가?

밥은 맛있나?



LG CNS에 입사할 것인가?

30

-20

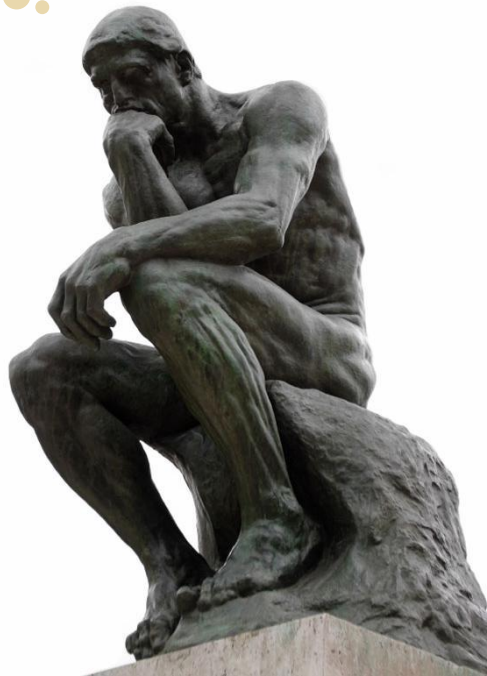
15

10

15

10

100



LG CNS에 입사할 것인가?

기준 : **80점** 이상 만족 시 입사 결정!

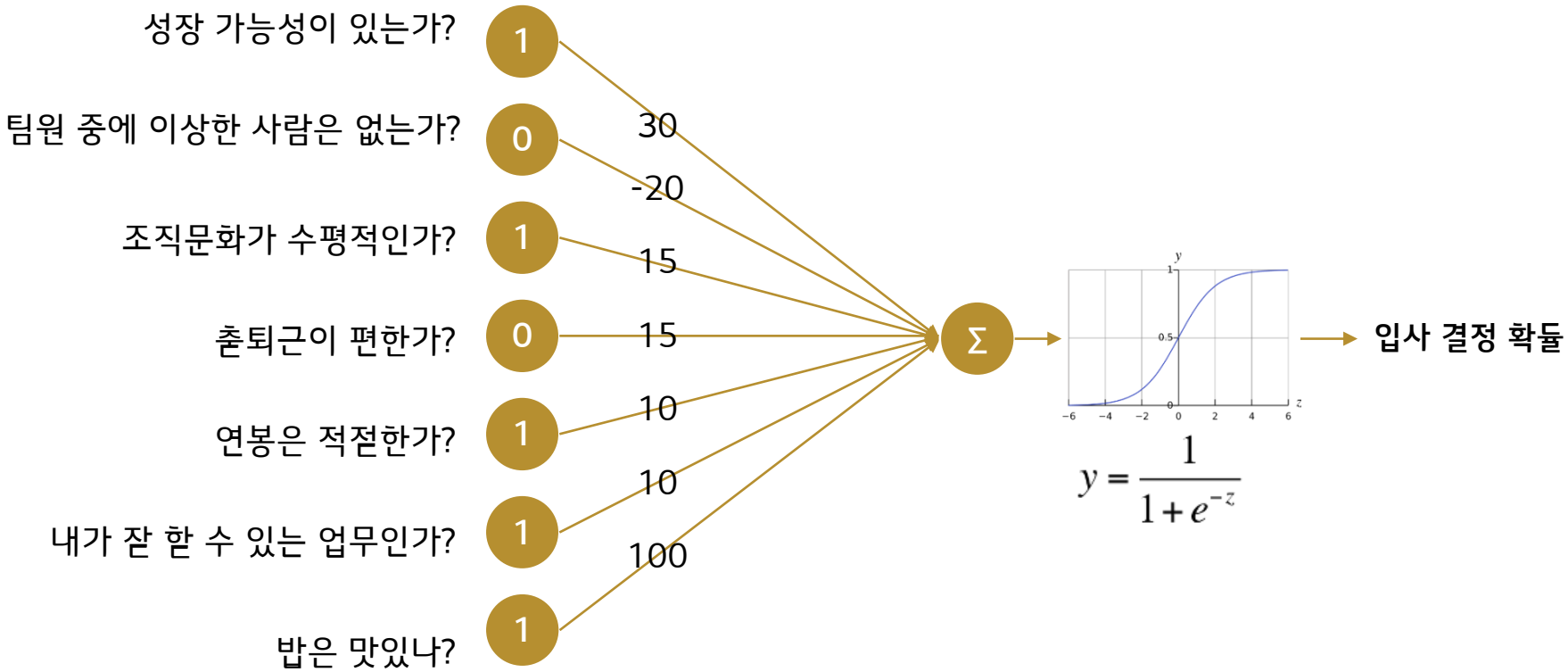
조건	Weight (w)	Data (x)	Weighted Sum $(\sum wx)$
성장 가능성이 있는가?	30	O	30*1
팀원 중에 이상한 사람은 없는가?	-20	X	-20*0
조직문화가 수평적인가?	15	O	15*1
출퇴근이 편한가?	15	X	15*0
연봉은 적절한가?	10	O	10*1
내가 잘 할 수 있는 업무인가?	10	O	10*1
밥은 맛있나?	100	O	100*1

= 165 ≥ 80

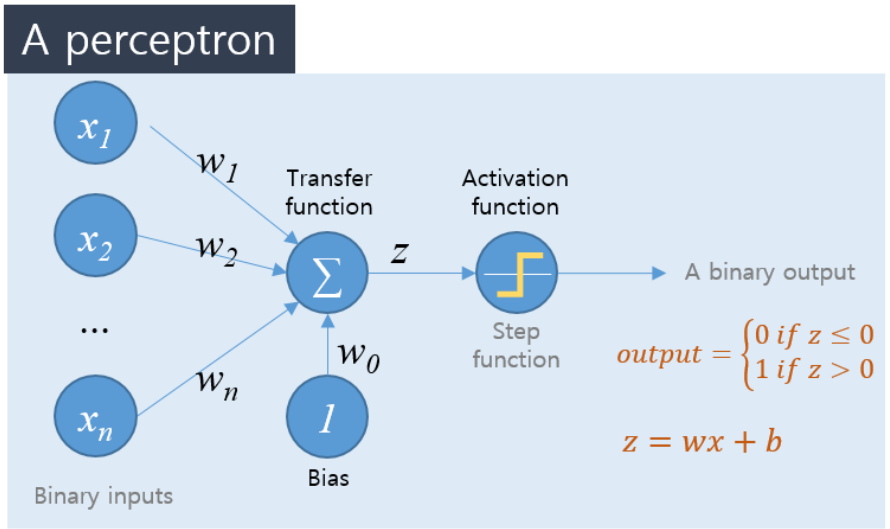
LG CNS 입사 결정!

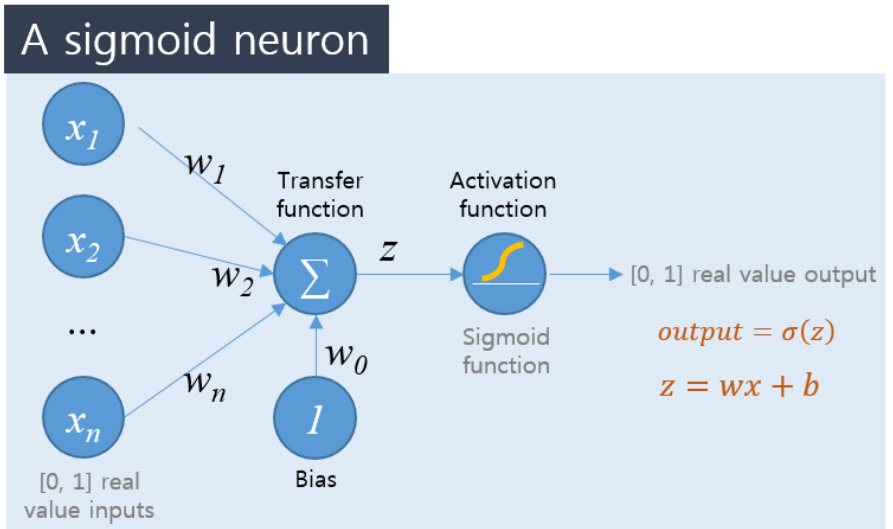
Feedforward Neural Network

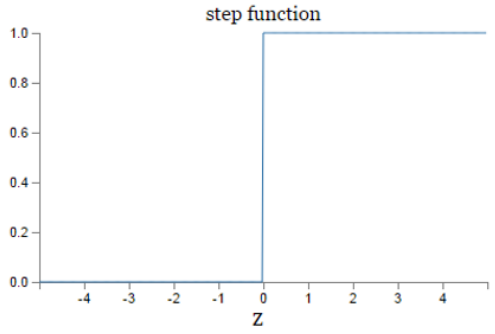
LG CNS에 입사할 것인가?



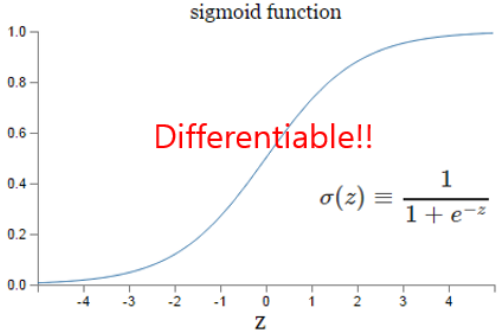
시그모이드 뉴런 (Sigmoid Neuron)







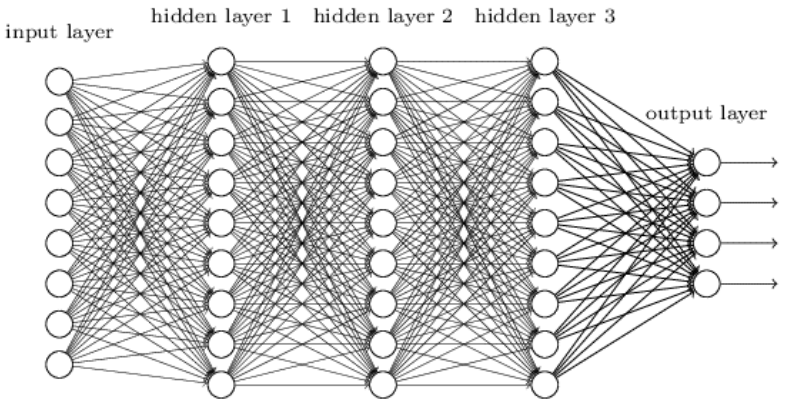
Activation function

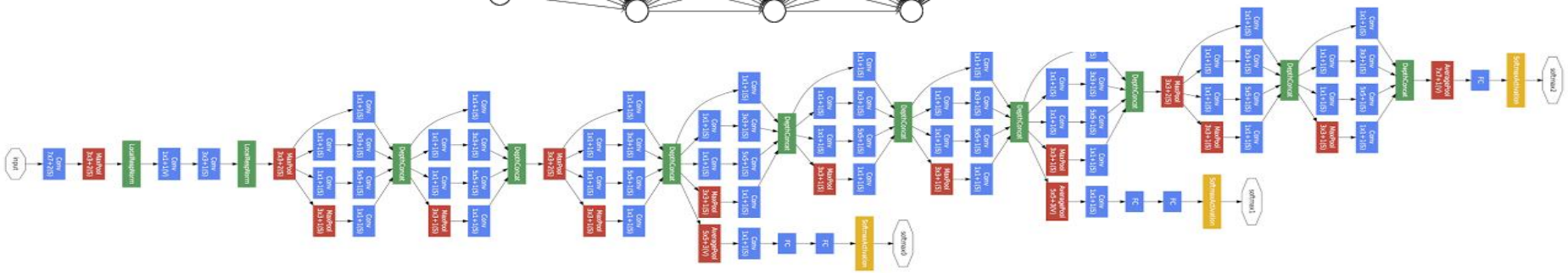


Activation function

DEEP Feedforward Neural Network

이러한 네트워크를 깊고 깊게(DEEP) 쌓으면 …





구체적인 feature

- Edge
- Color
- …

- 뾰족한 귀
- 둥그런 코
- …

추상화된 feature

- 강아지의 득징이 보이는가?
- 고양이의 득징이 보이는가?

딥러닝 정의

Deep learning is a class of machine learning algorithms that :

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.

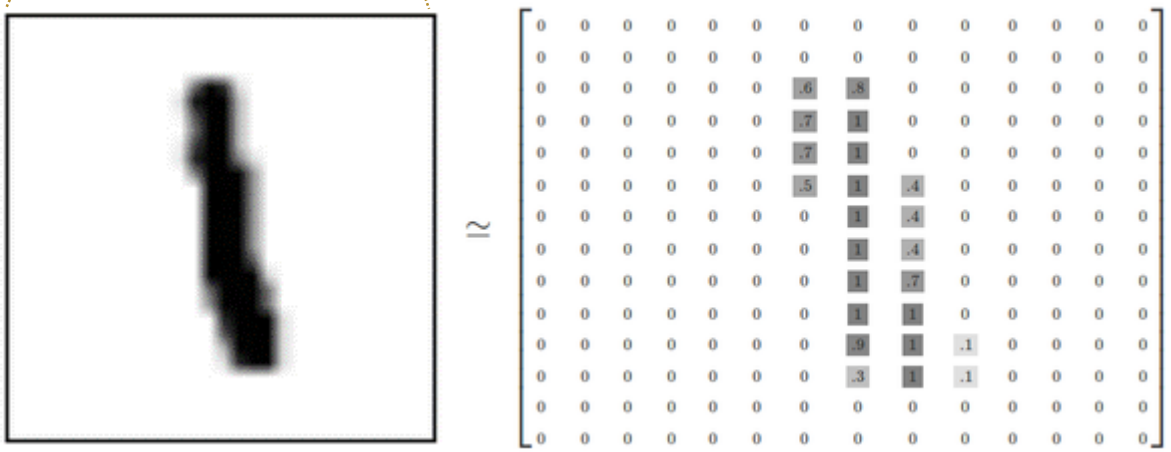
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

*참고 : https://en.wikipedia.org/wiki/Deep_learning#Definitions

필기체 숫자(MNIST*) 인식

MNIST dataset

- ✓ 숫자의 각 이미지는 28*28 흑백 이미지 (784픽셀)
- ✓ 정답 라벨(Ground Truth)은 길이 10의 열벡터



[0.]
[1.]
[0.]
[0.]
[0.]
[0.]
[0.]
[0.]
[0.]
[0.]

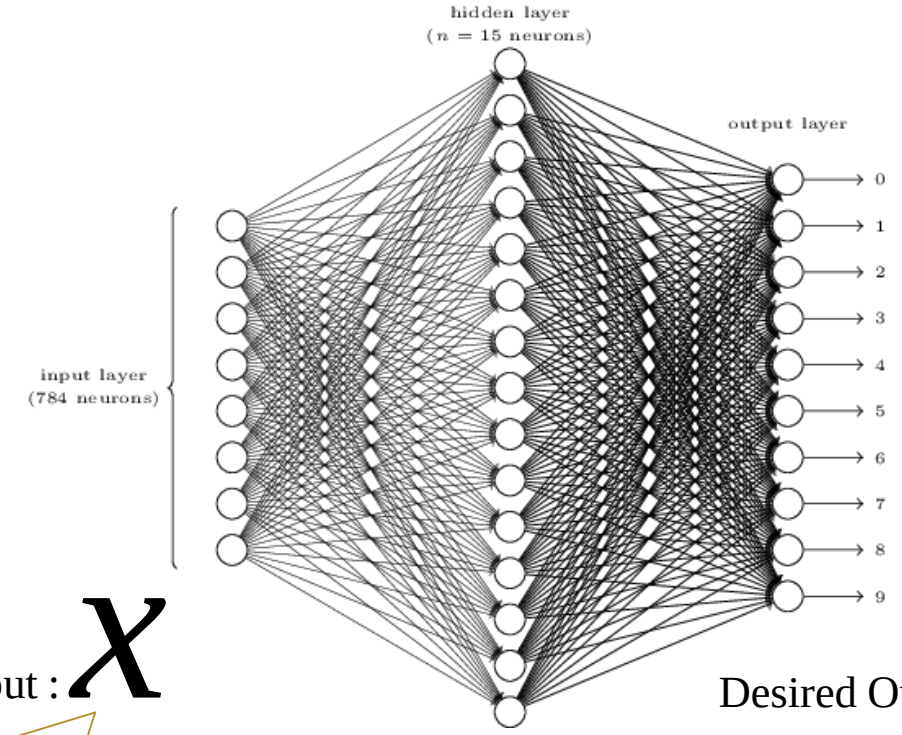
* Modified National Institute of Standards and Technology 데이터셋으로 필기체 숫자 이미지 모음이며 60,000개의 트레이닝 이미지와 10,000개의 테스트 이미지를 포함

필기체 숫자(MNIST) 인식

인공신경망 학습시키기 = weights와 biases 찾기

Dataset for training

image	digit
image	digit
image	digit
.....	
image	digit



Input : x

회색조 이미지에 대한 벡터도,
각 원소는 0~1사이의 값을 가짐
(255로 normalized)

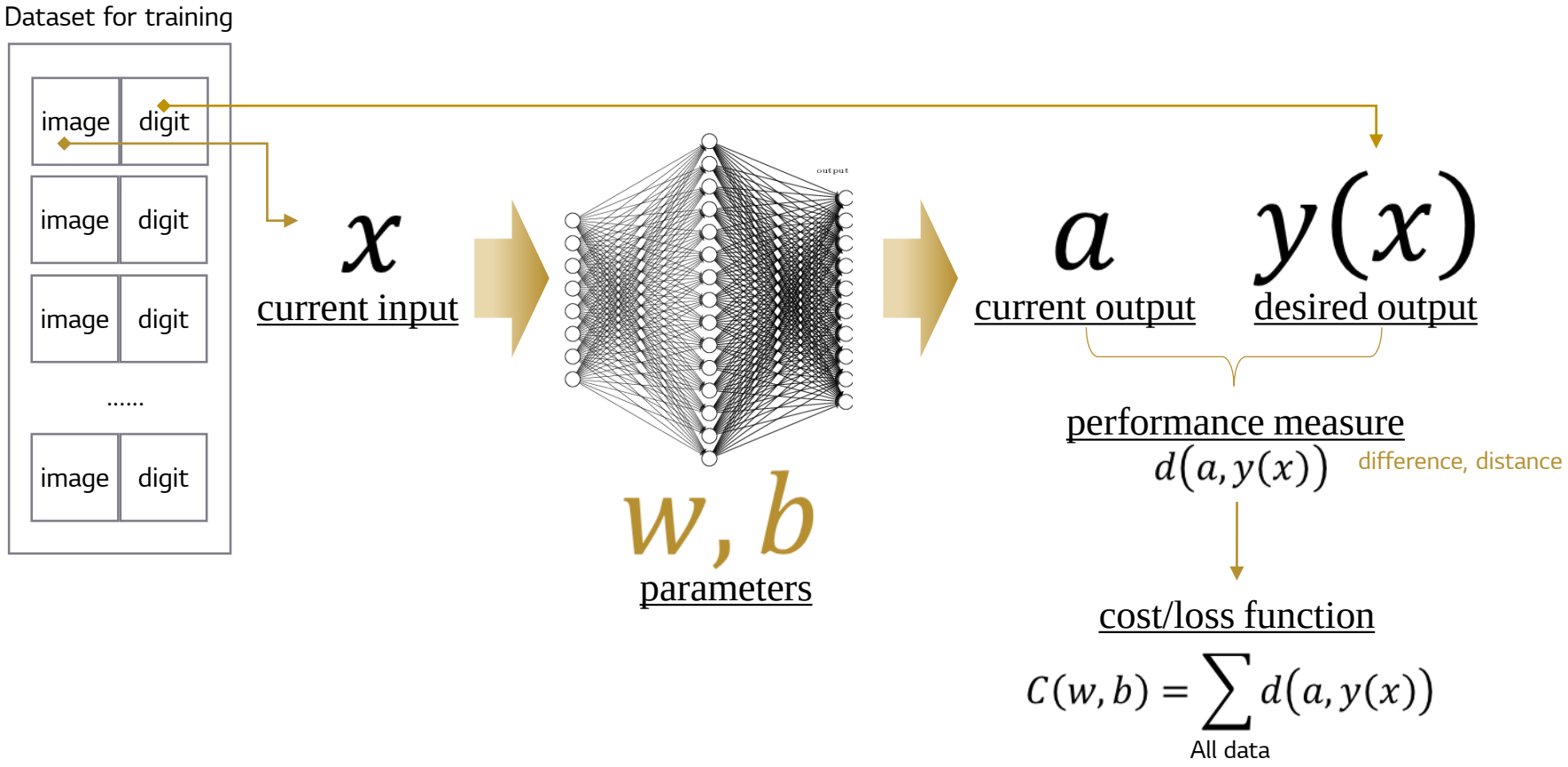
w : all weights, b : all biases

Desired Output : $y(x)$

10 dimension의 벡터도,
만일 정답이 '6'이라면
y(x) = (0,0,0,0,0,0,1,0,0,0)T

필기체 숫자(MNIST) 인식

인공신경망 학습시키기 = weights와 biases 찾기

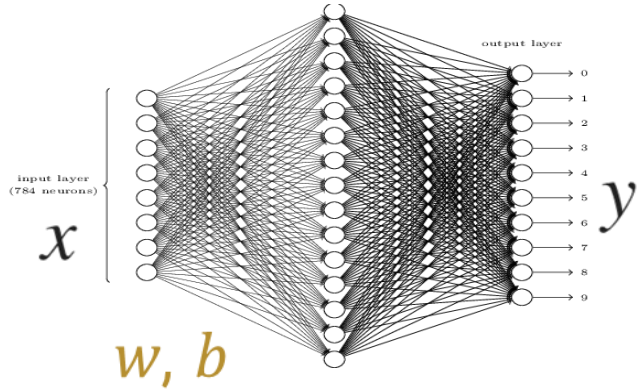


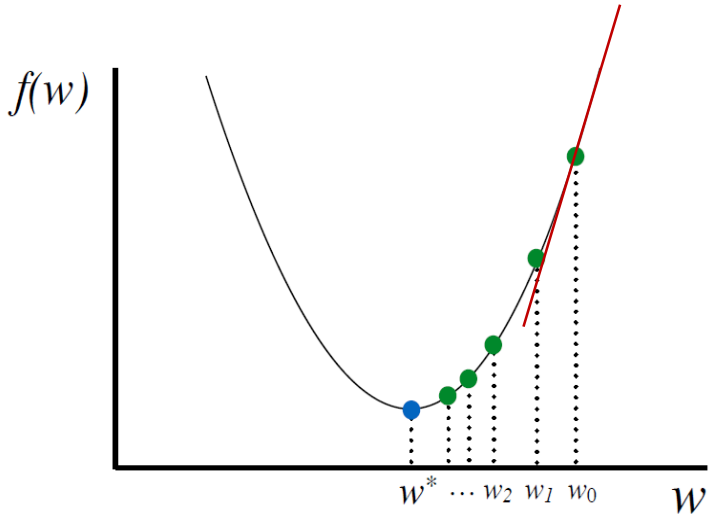
$d(a, y(x))$가 'a'와 '$y(x)$'의 차이를 나타내는 non-negative value라면,
학습의 목적은 $C(w,b) = 0$ or $minimize\ C(w,b)$이 되도록 하는 w, b 찾기이다.

경사하강법 : Gradient Descent

Learning이란 어떻게 일어나는가?

목적 모델의 예측값과 실제 정답값의 차이를 최소화하는 파라미터(w, b) 구하기



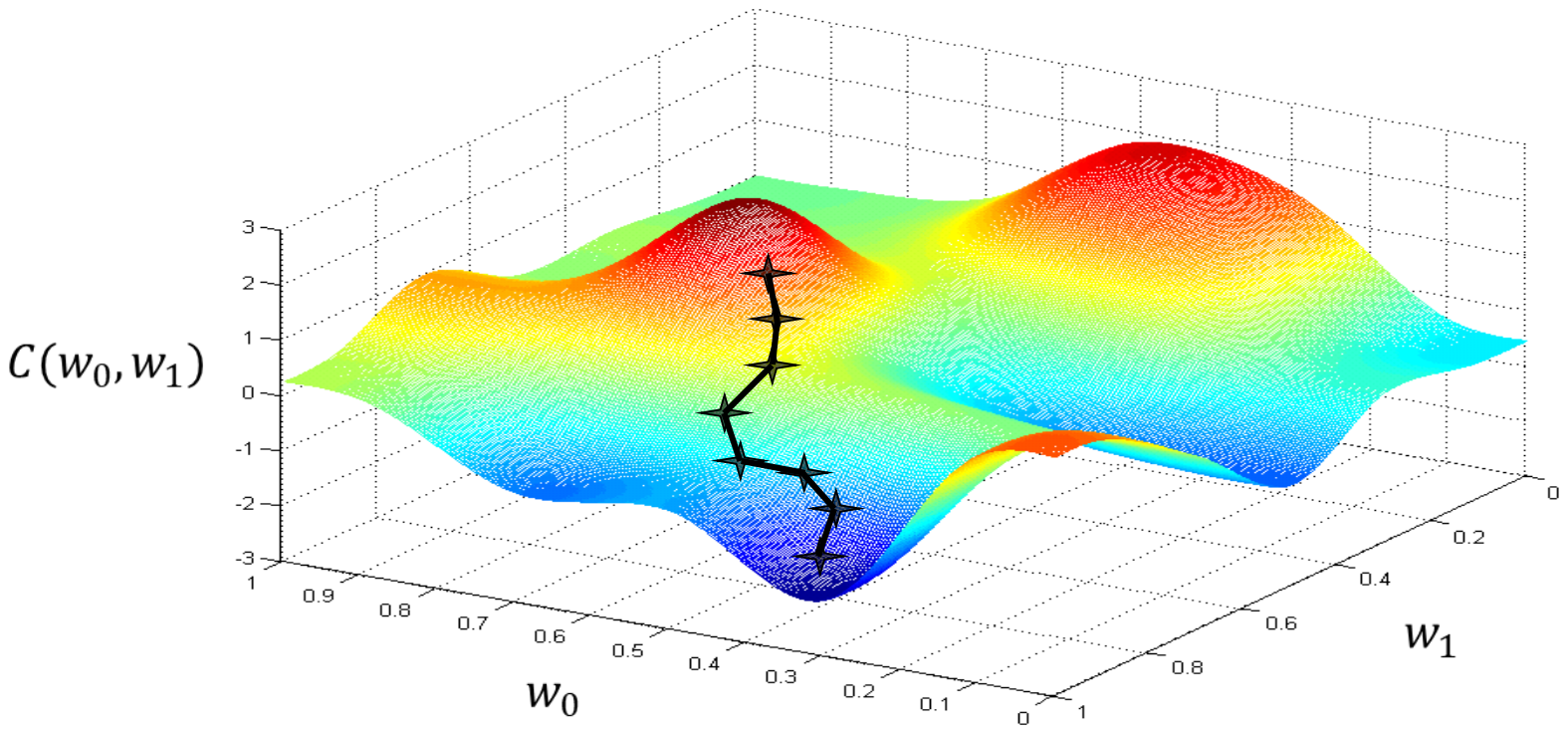


input data	x
Parameter	w, b
모델의 예측값	a
실제 정답값	y
Cost Function (Quadratic Cost function)	$C(w,b) = \frac{1}{2n}\sum_{i=1}^{n}(a_i - y_i)^2$

$$w_{j+1} \leftarrow w_j - \alpha\frac{\partial C(w)}{\partial w_j}$$

한 번에 최적의 parameter를 찾는 게 아니라
여러 번 좋은 방향으로 update 하며 '보정'해나가는 개념!

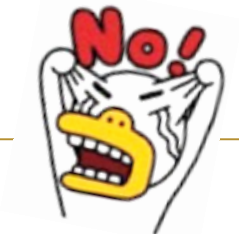
경사하강법 : Gradient Descent



하지만 이런 parameter들이 단순히 weight 한두어개가 아니라 어어엄청(보통 몇백만~몇십억) 많이 있은 것이기 때문에....

시각화는 묻돈 머릿속에서 상상하는 것도 몹시 어렵습니다.

우리는 parameter 수 만큼의 다차원 실수공간 기울기 계산(미분...!)은 동해 한번에 고려해야 합니다.

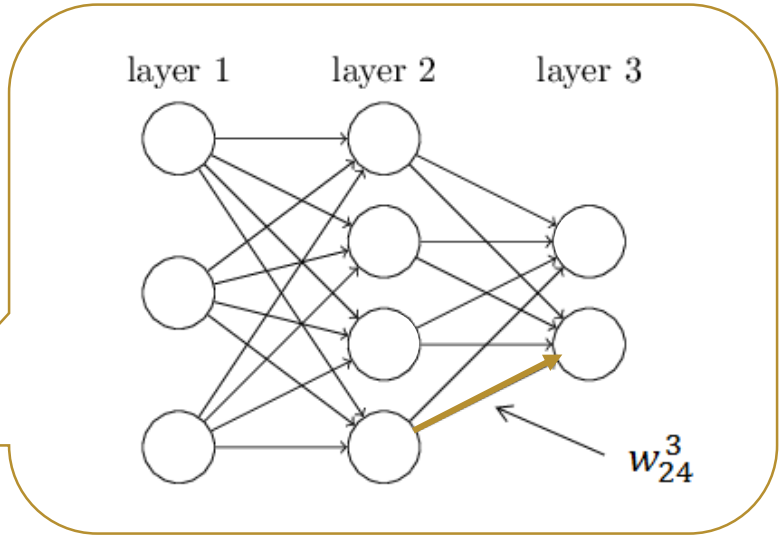


오류역전파 : Backpropagation

무수한 파라미터들.... Cost에 대한 기울기(Gradient)를 언제 다 구하지!?

득정 weight가
Cost에 미치는 영향은?

$$\frac{\partial C(w)}{\partial w_{jk}^{l}}$$



다음과 같이
Error δ를 정의하자

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

l번째 레이어의 j번째 노드 output이
Cost에 미치는 영향

Gradient를 쉽고 빠르게 구하도록 도와주는 Backpropagation의 4대 방정식

1. Error at the output layer	2. Error relationship between two adjacent layers	3. Gradient of C in terms of bias	4. Gradient of C in terms of weight
$\delta^L = \nabla_a C \odot \sigma'(z^L)$	$\delta^l = \sigma'(z^l) \odot \left((w^{l+1})^T \delta^{l+1}\right)$	$\nabla_{b^l} C = \delta^l$	$\nabla_{w^l} C = \delta^l (a^{l-1})^T$

신경망의 맨 마지막 레이어 L에서의 Error δ^L은 구할 수 있다

$(l+1)$번째 레이어의 Error δ^{l+1}를 알 수 있다면, l 번째 레이어의 Error δ^l도 구할 수 있다

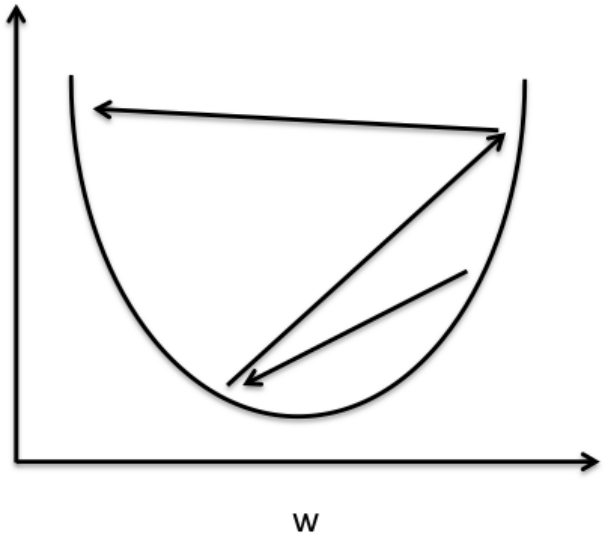
l 번째 레이어의 Error δ^l를 안다면,
l 번째 레이어의 bias와 weight에 대한 Gradient를 구할 수 있다

Hyperparameter : Learning rate

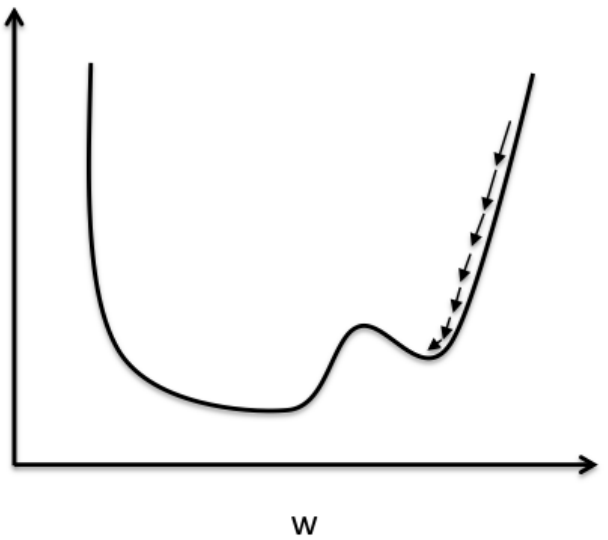
파라미터 업데이트의 정도를 조절하는 learning rate

$$w_{j+1} \leftarrow w_j - \alpha \frac{\partial C(w)}{\partial w_j}$$

Learning rate (>0)



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

용어정리 : Hyperparameter

학습하는 데 필요하지만, 학습은 되지 않는 하이퍼파라미터

Hyperparameters

- 모델의 capacity(or complexity)를 결정
- Training동안 학습되지 않음 → 사람(신경망 설계자)이 설정해줘야 함
- 예 : learning rate, weight decay, hidden size, layer 수, mini-batch size, feature 수 등..

$$w_{j+1} \leftarrow w_j - \alpha \frac{\partial C(w)}{\partial w_j}$$

Parameters

- 모델에 의해 Training동안 학습됨 → 사람이 설정하는 것이 아니라, 데이터를 통해 결정됨
- 예 : 인공신경망의 weights & biases 등..

용어 정리

Epoch, Iteration, Mini-batch size

1 epoch	모든 Training data가 한 번씩 forward pass와 backward pass를 진행
1 iteration	한 번의 forward pass와 backward pass
Mini-batch size	한 iteration을 진행할 Training data 예제의 수

예제 Training data 50,000건이 있고 mini-batch size가 1,000이라면
1 epoch을 수행하는 데 50 iteration이 진행된다.

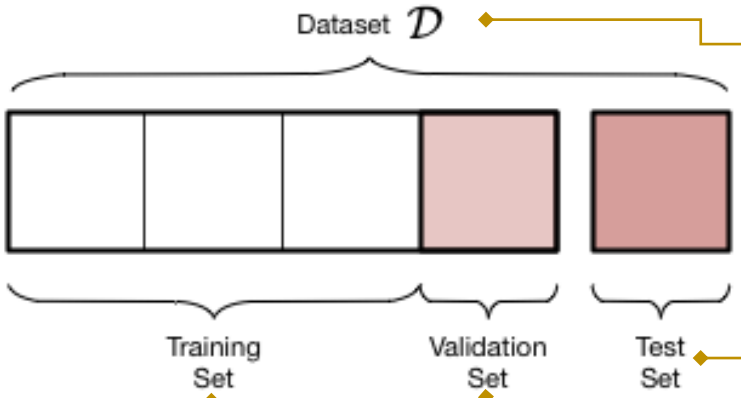
Generalization

우리가 만든 알고리즘은 훈련 시 본 적이 없는 새로운 입력 데이터에 대해서도 잘 수행되어야 한다

Generalization

The ability to perform well on *previously unobserved* inputs is called **generalization**

Generalization을 위해 데이터를 나누어 이용하자!



각 set의 example들은 모두 동일한 분포에서 수집되었다고 가정

모델 학습 및 hyperparameter 튜닝에 *전혀 관여하지 않은 set*
Test error = Generalization error
(=새로운 입력에 대한 error의 기댓값)

적절한 hyperparameter를 찾기 위해 이용
→ validation error가 최소화 되는 경우의 hyperparameter를 선택

모델 학습 시 이용, 모델은 training error를 줄이기 위해 노력
→ 단순 최적화(optimization)에 해당

Overfitting & Underfitting

내 모델이 과연 제대로 학습되었을까... ?

✓ Training set에 대한 error가 작아졌는가?

→ 모델은 **내가 알려준 데이터(Training set)**에 대해 정답은 잘 맞춰야 한다

→ If not?

Underfitting

모델이 training set에 대해 충분히 낮은 training error에 도달하지 못한 경우 발생

✓ Training error와 test error의 갭이 작은가?

→ 모델은 **내가 알려준 데이터**도 잘 맞춰야 하지만, **본 적이 없는 데이터(Test set)**에 대해서도 잘 맞춰야 한다.

→ If not?

Overfitting

Training error와 Test error의 갭이 너무 큰 경우 발생,
즉, 알려 주는 데이터는 잘 맞추는데 처음 보는 데이터는 잘 못맞추는 경우

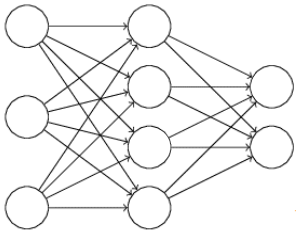
→ 태스크를 일반화하지 못하고 알려 준 데이터와 정답을 그냥 닫닫 외웠을 뿐이다..!

Overfitting & Underfitting

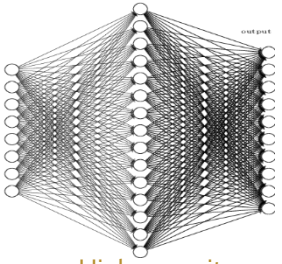
Model capacity를 조절하여 overfitting, underfitting을 해결할 수 있다

Capacity

"모델이 얼마나 복잡한 관계를 표현할 수 있는가"



Low capacity



High capacity

Capacity가 낮은 모델은 training set을 학습하는 데 어려움을 겪는다

즉, 너무 모델이 단순해서 데이터의 특징을 파악하기 힘들다

➡ **Underfitting 발생!**

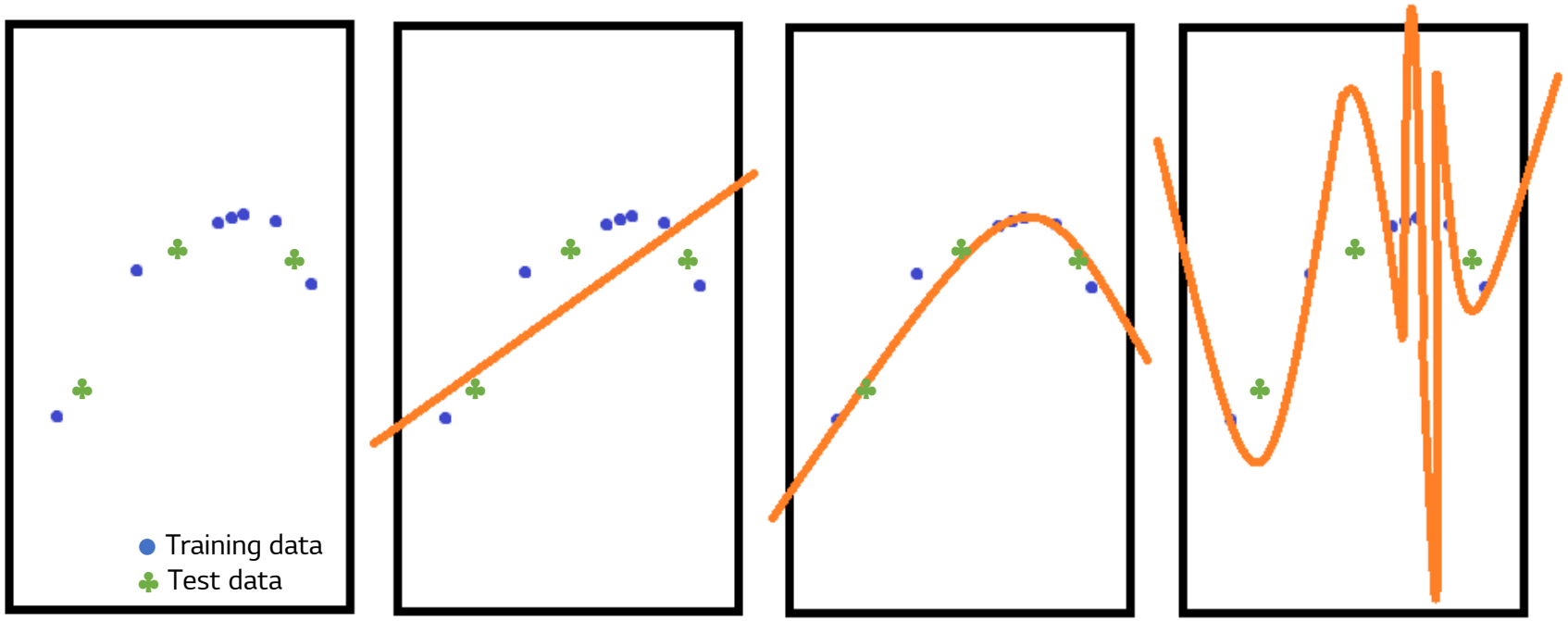
Capacity가 높은 모델은 training set의 속성을 외워버림으로써 과적합(overfit)이 되어,

test set에 대해선 잘 작동하지 않을 우려가 있다

➡ **Overfitting 발생!**

Overfitting & Underfitting

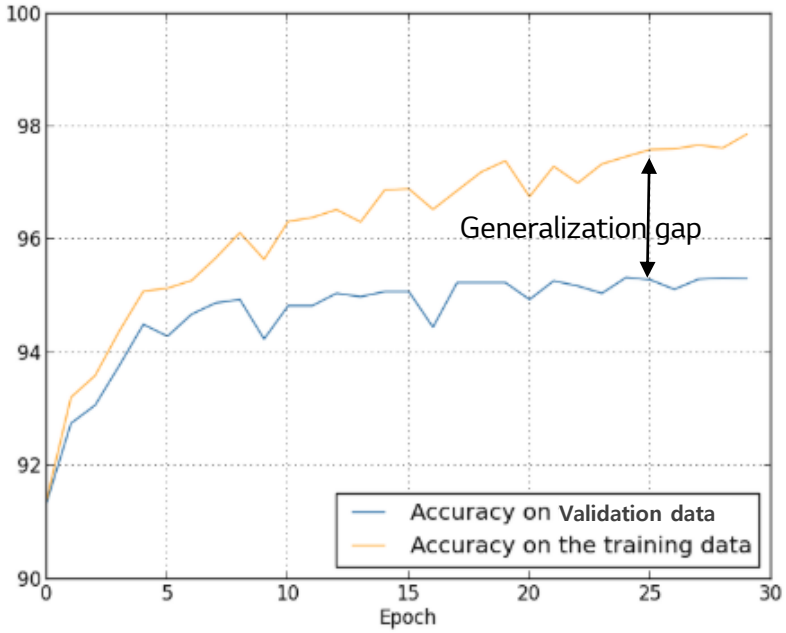
적절한 Capacity를 찾으면 Overfitting과 Underfitting을 방지할 수 있다



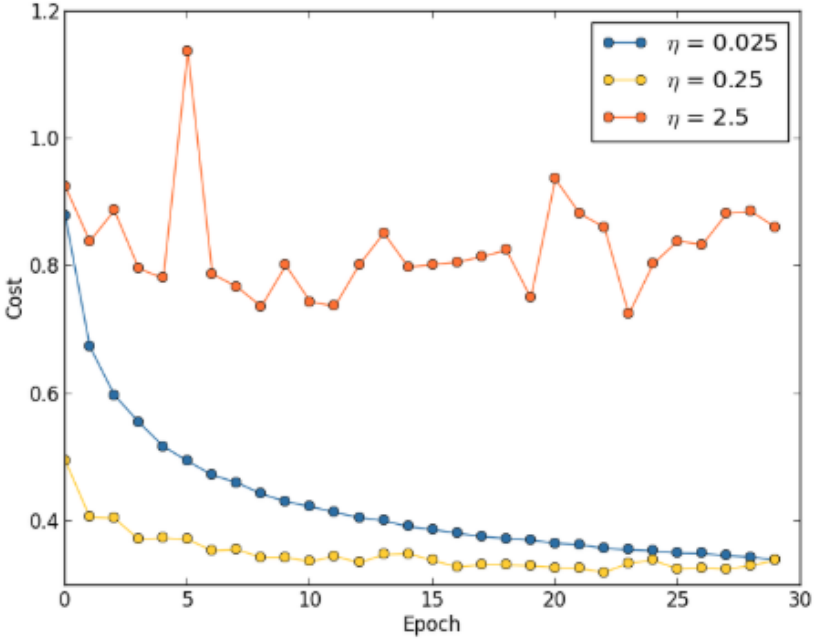
	Low capacity Underfitting	Appropriate Capacity	High capacity Overfitting
Training data	못맞춤	어느정도 잘맞춤	엄청 잘맞춤
Test data	못맞춤	어느정도 잘맞춤	못맞춤

Learning curve

Learning curve를 보고 모형 개선방안을 찾아보자



Overfitting의 조짐이 보이는가?!



적절한 Learning rate 찾기

Performance measure

Task 목적에 따라 알맞은 measure가 달라진다

Measurement

모델이 얼마나 과제를 잘 수행하는지를 객관적으로 수치화한 "성능 평가 지표"

이전에 보지 못했던 데이터에 대해 얼마나 잘 수행하는지가 중요하기 때문에
Test set을 사용하여 성능을 측정

예) 양불판정 task

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive,** Power	**False positive,** Type I error
	Predicted condition negative	**False negative,** Type II error	**True negative**

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

아 모르겠고, 우린 그냥 분류 정확도 높여주세요!

$$\text{Recall} = \frac{tp}{tp + fn}$$

(검출율)

분량제품을 판매하게 되면 브랜드이미지 타격이 큽니다.
사람이 재검수해도 좋으니 낌새만 있으면 다 골라내주세요!

$$\text{Precision} = \frac{tp}{tp + fp}$$

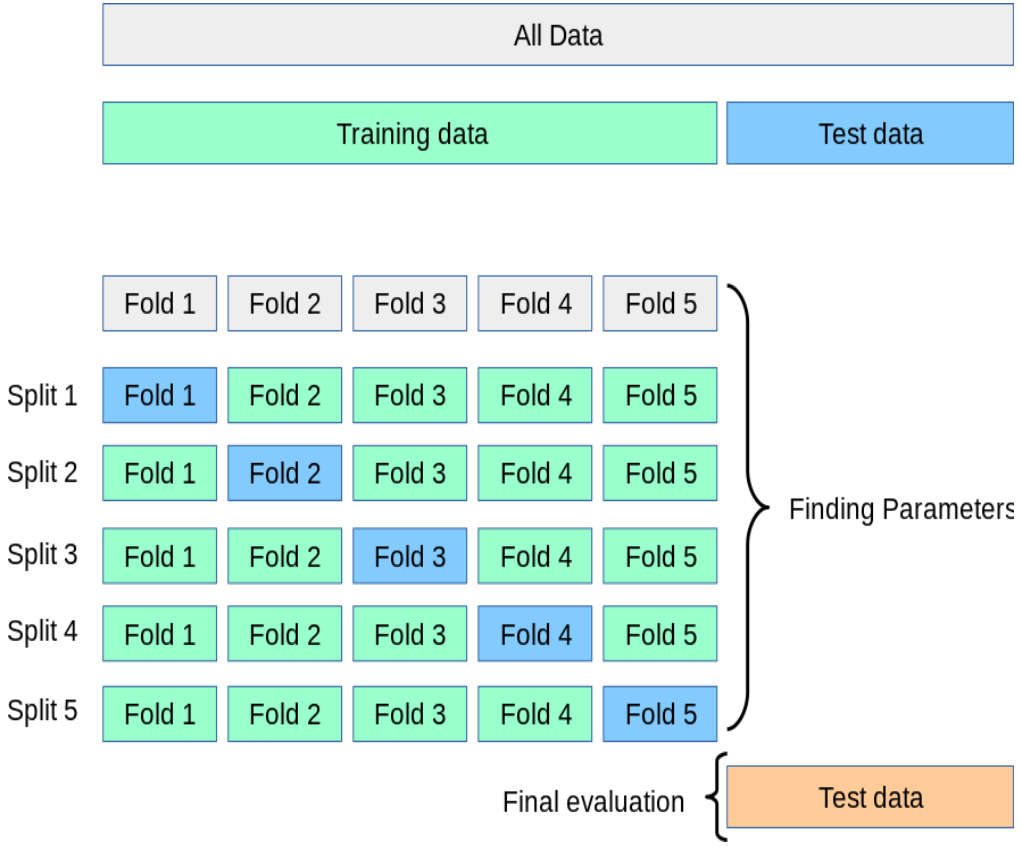
(정밀도)

모델이 분량이라고 판정한거 담당자가 다 보기 힘들어요...
모델이 아무거나 다 분량이라고 해도 곤란해요

출처 : https://en.wikipedia.org/wiki/Precision_and_recall

k-Fold Cross Validation

데이터셋의 크기가 작은 경우 train/val/test 분배에 따라 성능 지표 변동이 크게 달라질 수 있는 문제가 발생, 이를 해결하기 위해 모든 데이터가 최소 한 번은 validation set셋으로 사용되도록 하는 방법



[작업 프로세스]

1. Training set과 Test set으로 데이터를 나눈다. (Test set은 최종성능 평가를 위해 따로 빼놓는다.)
2. Training을 K개의 fold로 나눈다.(그림의 경우 K=5)
3. K개의 fold 중 1개는 Validation set으로 지정하고, K-1개는 Training set으로 지정한다.
4. K-1개의 Training set으로 모델을 생성하고, Validation set에 대해 예측을 진행하여, 이에 대한 성능지표를 추출한다.
5. 다음 fold에서는 Validation set을 바꿔서 지정하고, 이전 fold에서 Validation 역할을 했던 Set은 다시 Training set으로 활용한다.
6. 이를 모든 fold에 대해 K번 반복한다.