

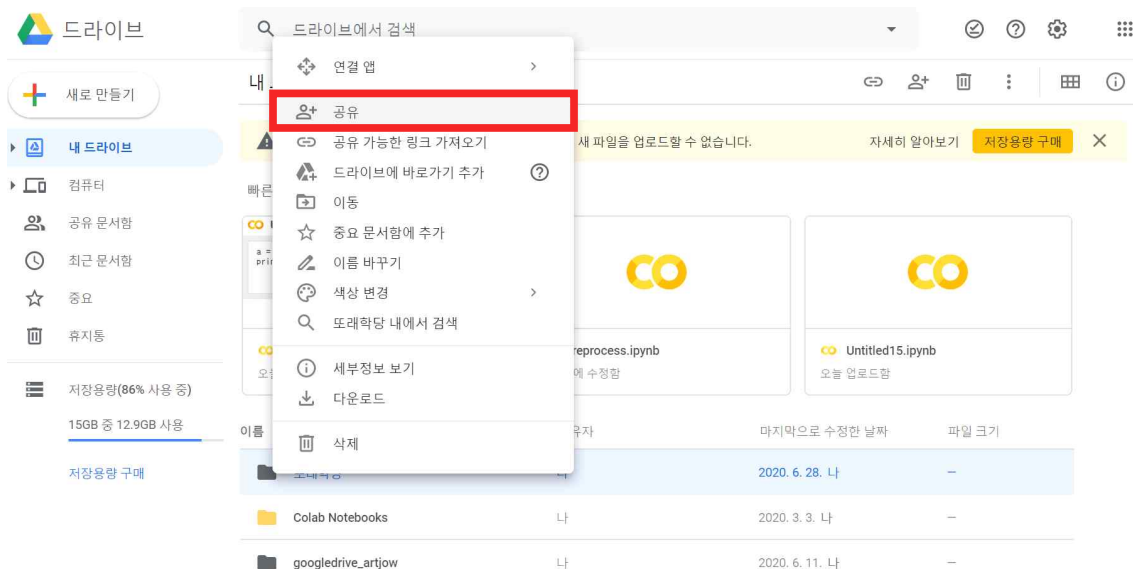
[2차시 수업]

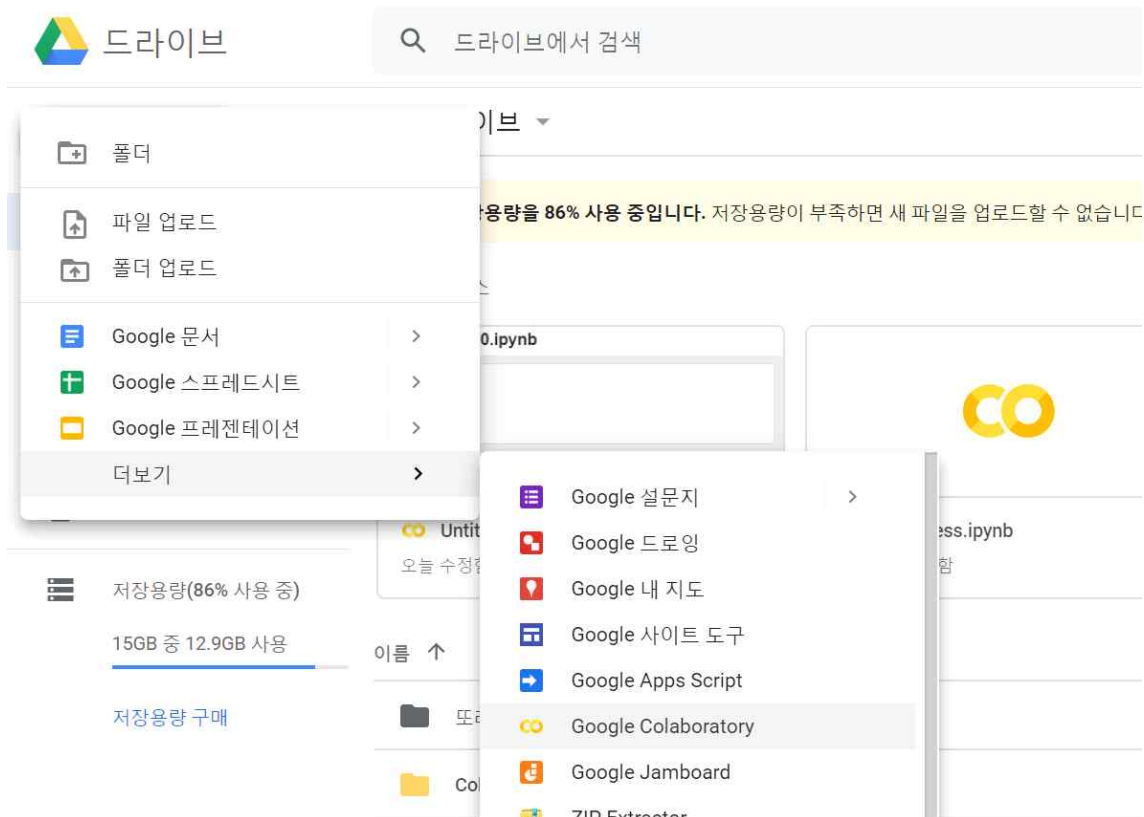
- 1) 구글콜랩, Github,구글드라이브 ;
깃허브,구글콜랩,구글드라이브 사용법 소개
- 2) 머신러닝 기초학습; 머신러닝 소개(ML Basic)
 - 체험: BIG GAN(구글콜랩에서)
 - 체험: NeuralStyleTransfer.ipynb(구글콜랩에서)
- 3) Linear Regression(선형회귀)
 - 문제풀기; Linear Regression 문제
 - 머신러닝비디오게임 영상; MarI/O - Machine Learning for Video Games

■ 1교시

1. 구글드라이브; https://www.google.com/intl/ko_ALL/drive/

구글드라이브 공유하기; 폴더 만들고, 폴더 우클릭 후 공유하기





(구글 콜랩과 연동)

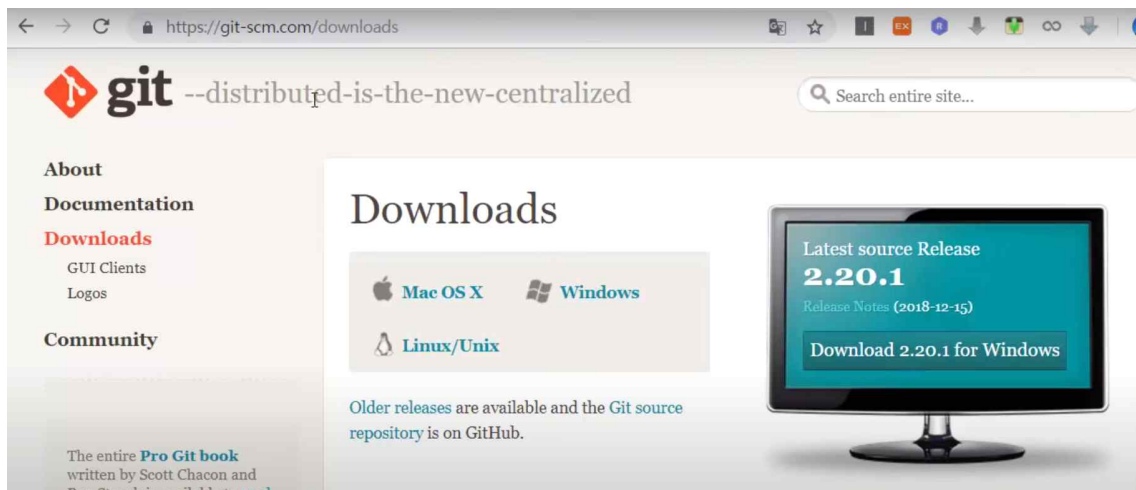
2. 깃허브

참조 사이트;

<https://www.youtube.com/watch?v=rhP5pseOJc0&list=PLRx0vPv1EmdD5FLldwTM4mKBgYjv4no81&index=1&pbjreload=101>

https://www.youtube.com/watch?v=JBN_hjgR1KQ&list=PLBHVuY1KEkULuUe_Ca3wiaFon6dPWIWAZ&index=1

1) git 설치하기; <https://git-scm.com/downloads>



2) 깃허브 사이트 접속; <https://github.com/>
https://github.com/login?return_to=%2Faccount%2Forganizations%2Fnew

3) 새로운 레포지토리 생성

새로운 리포지토리 생성

리포지토리에는 개정 내역을 포함하여 모든 프로젝트 파일이 포함됩니다. 다른 곳에 프로젝트 저장소가 있습니까?
[저장소를 가져 오십시오.](#)

소유자 * 리포지토리 이름 *

 초 형래 ▾ /

훌륭한 저장소 이름은 짧고 기억에 남습니다. 영감이 필요하십니까? 방법에 대해 [기빠](#) · [옥토](#) · [우산](#) ?

설명 (선택 사항)

☒  **공공의**
인터넷상의 모든 사람이 이 저장소를 볼 수 있습니다. 커밋 할 수 있는 사람을 선택합니다.

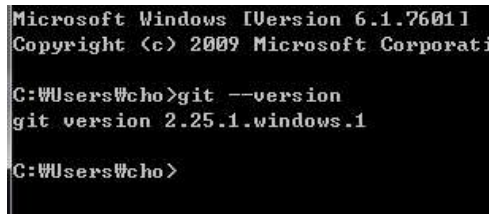
☐  **은밀한**
이 저장소를보고 커밋 할 수 있는 사람을 선택합니다.

기존 리포지토리를 가져 오는 경우가 단계를 건너 뛰십시오.

4) cmd창에서 git 버전 확인

C:\Users\cho>git --version

git version 2.25.1.windows.1



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation

C:\Users\cho>git --version
git version 2.25.1.windows.1

C:\Users\cho>
```

5) 환경설정

C:\Users\cho>git config --global

6) C 드라이브에 폴더생성

- 폴더와 깃허브 동기화

※ 폴더 생성할 드라이브로 이동

>cd\ ; c드라이브로 이동

>mkdir 디렉토리이름; 디렉토리 생성

mkdir gittest

7) 만든 디렉토리로 이동 하는 경우

C:\Users\cho>cd C:\gittest

C:\gittest>

※ 폴더내 확인 >dir



```
C:\gittest>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: B6BF-6924

C:\gittest 디렉터리

2020-07-01 오후 01:05 <DIR> .
2020-07-01 오후 01:05 <DIR> ..
2020-07-01 오후 01:09 <DIR> Peer-School-Lecture
                0개 파일                0 바이트
                3개 디렉터리 447,847,993,344 바이트 남음
```

※ 환경설정; 데이터 올리는 자의 이력 설정(아이디와 이메일 정보 기입)

C:\gittest>git config --global user.name"artjow"

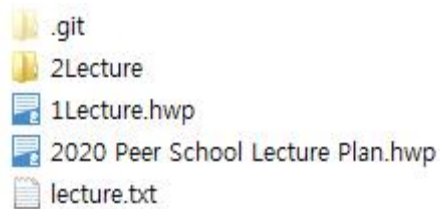
C:\gittest>git config --global user.email"artjow@naver.com"

```
C:\Wgittest>git config --global user.name"artjow"
C:\Wgittest>git config --global user.email"artjow@naver.com"
C:\Wgittest>_
```

8) 깃사이트에서 생성한 레포지토리 주소를 복사 후
cmd창에 입력< 다운로드(git clone 한다)

C:\gittest>git clone <https://github.com/chohyungrae/Peer-School-Lecture.git>

9) 내pc 폴더에 파일이 생성됨



10)cmd창에서 업로드할 파일 경로로 이동

```
C:\Wgittest>cd C:\Wgittest\Peer-School-Lecture\2Lecture_
```

11)업로드할 파일 이름 입력

> git add 파일이름

```
C:\Wgittest>git add Linear_Regression.ipynb_
```

12) commit 하기 전 올릴 파일 상태확인

>git status

```
C:\Wgittest\Peer-School-Lecture\2Lecture>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Vectors.txt
```

13) 업데이트된 것을 알린다

>git commit -m "메세지"

```
C:\Wgittest>git commit -m "Add Linear_Regression"
```

※ Staged Filed과 Committed File의 차이점

Staged Filed vs Committed File

- Staged File은 이 파일의 변화를 지속 관찰 할 대상 파일 이라는 의미 → 아직 변화 이력 저장이 안됨

- Committed File은 현재 상태를 Git Repository 저장
→ 파일의 변화 이력을 저장함

14) 푸쉬를 통해 깃허브에 올리기

git push

```
C:\Wgittest>git push_
```

3. 구글콜랩

※ 구글콜랩 소개

<https://colab.research.google.com/notebooks/welcome.ipynb>

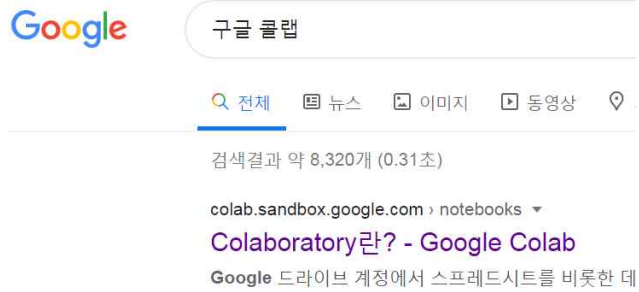
1) 구글드라이브에서 Colaboratory 검색후 설치(연결)

드라이브에 앱 연결



2) 구글콜랩 검색

<https://www.google.com/search?q=%EA%B5%AC%EA%B8%80+%EC%BD%9C%EB%9E%A9&oq=rnrmfzhffpq&aqs=chrome..69i59j2.3493j0j7&sourceid=chrome&ie=UTF-8>



(1) 콜랩연습

```
a = 5
```

```
print(a)
```

A screenshot of a Google Colab code block. The code block contains the following code:

```
a = 5  
print(a)
```

The code is highlighted in blue. Below the code block, there is a play button icon and the output '5'. Below the output, there is a second code block with the following code:

```
[2] b = 3  
    c = 4  
    print(a, b, c)
```

The code is highlighted in green. Below the code block, there is a play button icon and the output '5 3 4'.

```
x = list(range(10))
```

```
print(x)
```

A screenshot of a Google Colab code block. The code block contains the following code:

```
x = list(range(10))  
print(x)
```

The code is highlighted in blue. Below the code block, there is a play button icon and the output 'range(0, 10)'. Below the output, there is a third code block with the following code:

```
x = list(range(10))  
print(x)
```

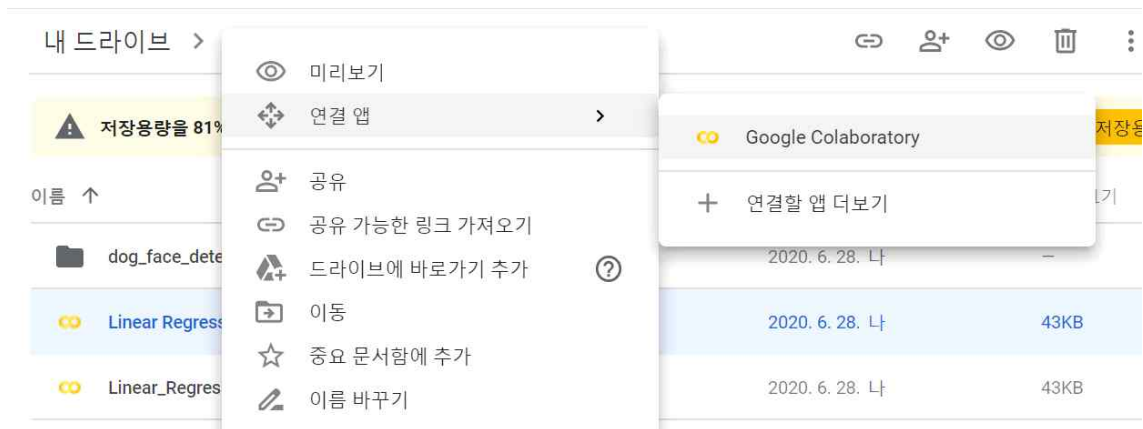
The code is highlighted in blue. Below the code block, there is a play button icon and the output 'range(0, 10)'.

(2) 깃허브 소스 구글콜랩으로 가져오기;

```
!git clone 주소기입
```

```
ls -ltr
```

(3) 구글드라이브에서 데이터 콜랩으로 이동



(4) 구글콜랩에서 구글드라이브와 연동

from google.colab import drive

drive.mount("/content/gdrive")

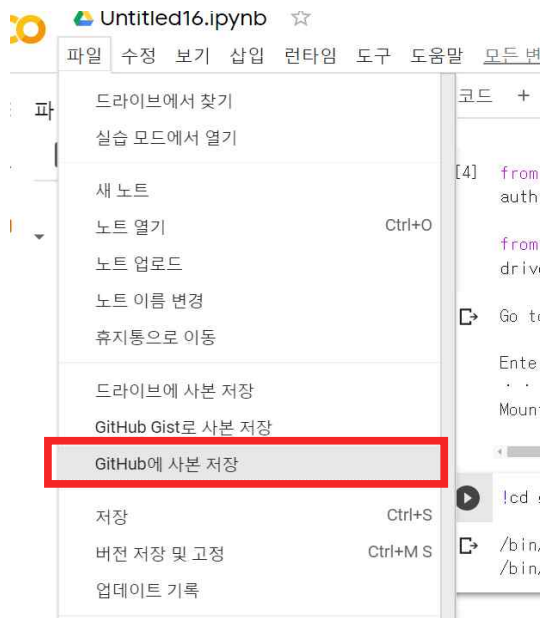
- google drive는 단 두 줄로 연동이 가능하며 해당 code block을 실행하고 권한 요청을 승인하면 소스 코드가 있는 google drive에 접근이 가능하게 됨. 즉, google drive에 있는 파일을 read 할 수도 있고, 실행 결과물들을 google drive에 저장할 수도 있게 된다.

(5) 파일 업로드

from google.colab import files

content_img = files.upload()

(6) 깃허브로 구글콜랩데이터(.ipynb)전송이 가능



GitHub으로 복사

저장소: [🔗](#)

chohyungrae/Peer-School-Lecture ▼
chohyungrae/Peer-School-Lecture
chohyungrae/PyTorch-GAN
chohyungrae/cho1
chohyungrae/marzipano

브랜치: [🔗](#)
master ▼

취소 [확인](#)

[Colaboratory 에 파일을 업로드하고 이미지 출력하기]

(7) 이미지 올리기

https://colab.research.google.com/github/nicewook/datascience_exercise/blob/master/upload_file_and_display_image.ipynb#scrollTo=yhKoG8OdcReS

가) from google.colab import files

uploaded = files.upload() # 파일 업로드 기능 실행

for fn in uploaded.keys(): # 업로드된 파일 정보 출력

print('User uploaded file "{name}" with length {length} bytes'.format(
name=fn, length=len(uploaded[fn])))

...

파일 선택

선택된 파일 없음

Cancel upload

나) ls -al



total 472

drwxr-xr-x 1 root root 4096 Jul 4 16:57 ./

drwxr-xr-x 1 root root 4096 Jul 4 16:52 ../

-rw-r--r-- 1 root root 298769 Jul 4 16:53 1.jpg

drwxr-xr-x 1 root root 4096 Jun 26 16:26 .config/

-rw-r--r-- 1 root root 165164 Jul 4 16:57 night_sky.jpg

drwxr-xr-x 1 root root 4096 Jun 26 16:26 sample_data/

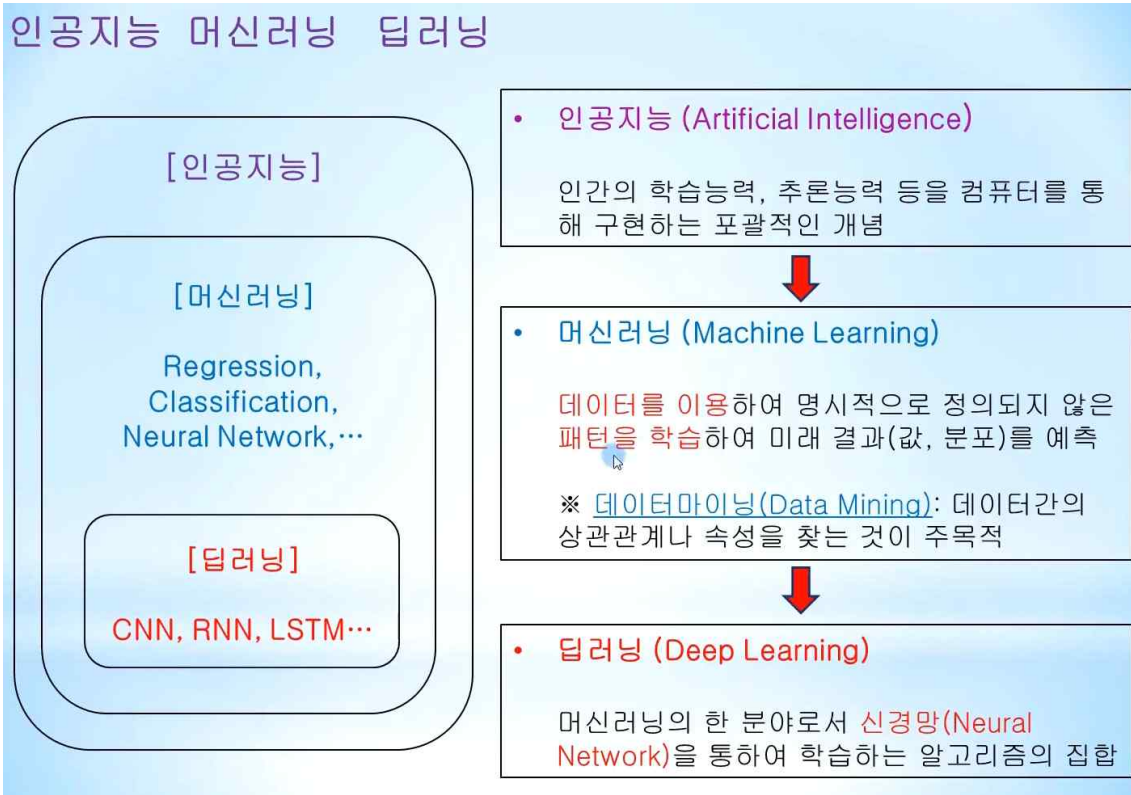
다) 이미지 보기

```
from IPython.display import Image
```

```
Image('night_sky.jpg')
```



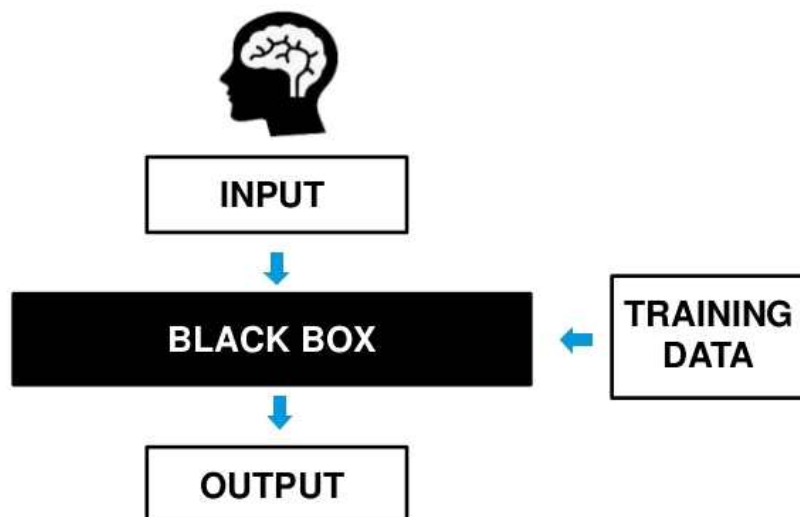
■ 2교시



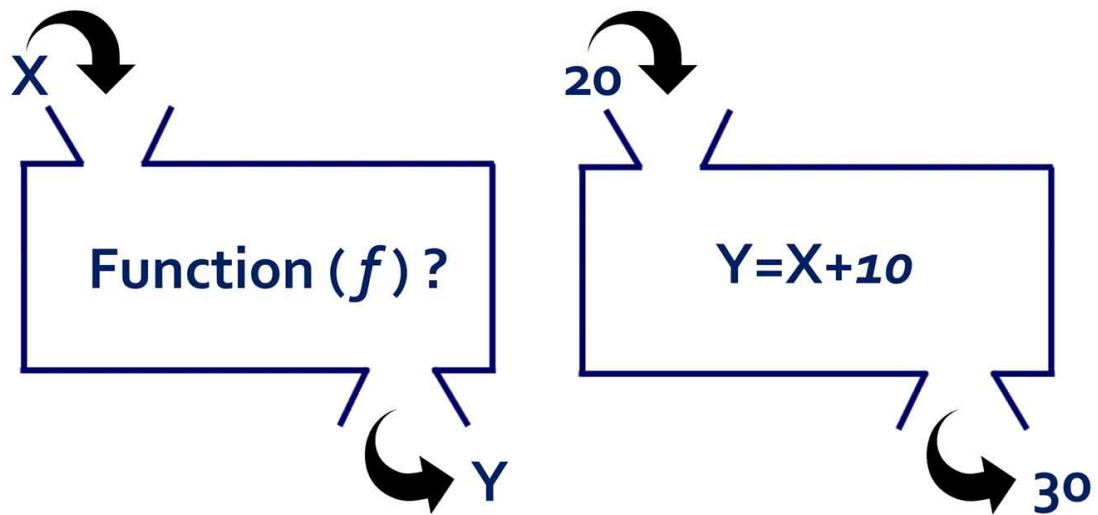
참조:

<https://www.youtube.com/watch?v=vcCaSBjpsHk&list=PLS8glc2q83OjStGjdTF2LZtc0vefCAbnX&index=1>

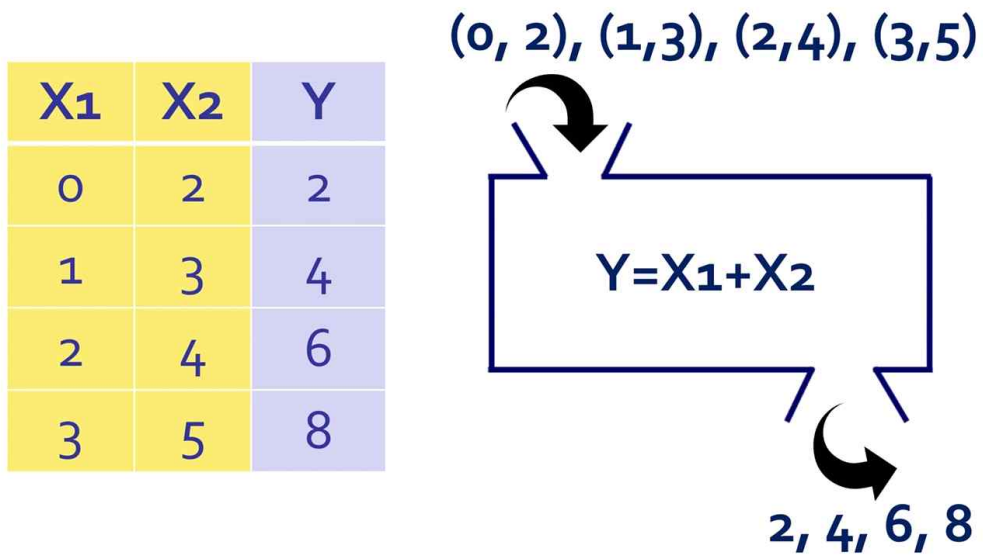
MACHINE LEARNING



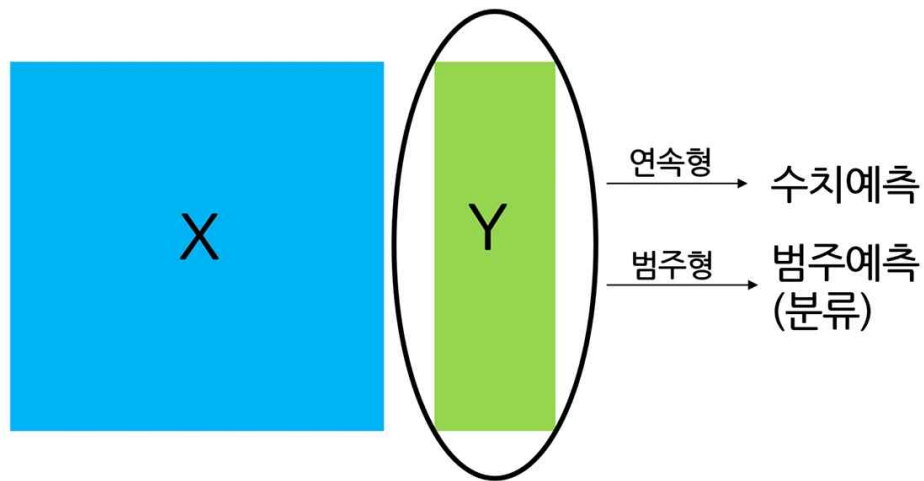
머신러닝 모델링



예측 모델링

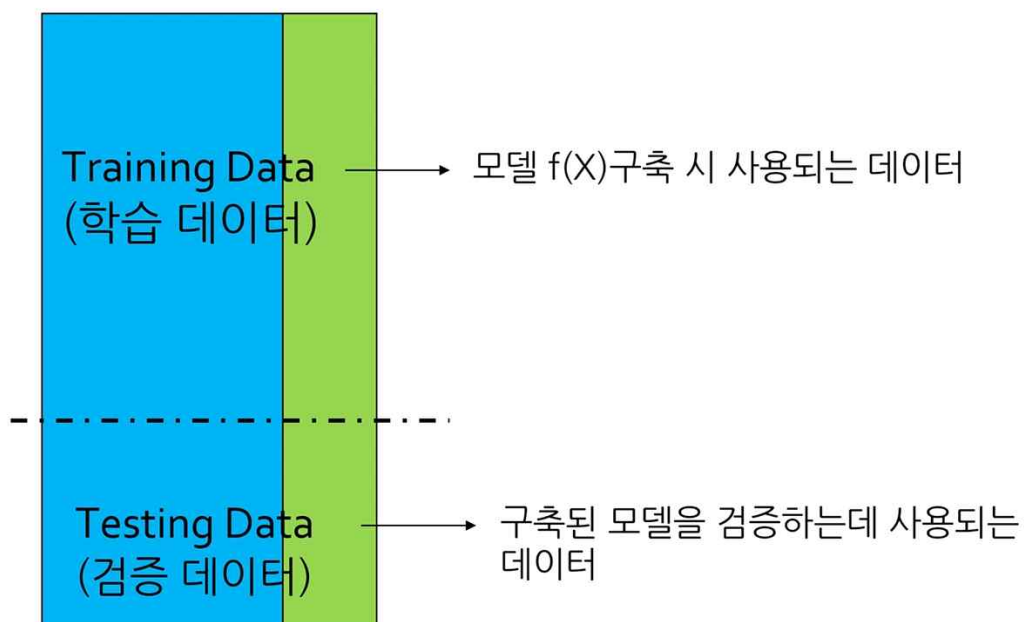


수치예측 / 범주예측 (분류)



- 연속형 데이터: 데이터 자체를 숫자로 표현
예) 가격, 길이, 압력, 두께, ...
- 범주형 데이터: 원칙적으로 숫자로 표시할 수 없는 데이터
예) 제품불량여부 (양품/불량), 고양이와 개의 분류, ...

학습데이터, 검증데이터



$$Y = \mathbf{w}_1 X_1 + \mathbf{w}_2 X_2 + \varepsilon$$

파라미터 (母數)

파라미터 추정

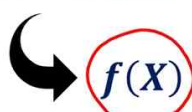
$$\sum_{i=1}^n \{Y_i - (\mathbf{w}_1 X_{1i} + \mathbf{w}_2 X_{2i})\}^2 \quad \text{Cost function (비용함수)}$$

비용함수를 최소로 하는 \mathbf{w}_1 와 \mathbf{w}_2 를 찾자!

$$\min_{\mathbf{w}_1, \mathbf{w}_2} \sum_{i=1}^n \{Y_i - (\mathbf{w}_1 X_{1i} + \mathbf{w}_2 X_{2i})\}^2$$

모델 결정 → 파라미터 추정

$$\min_{w_1, w_2} \sum_{i=1}^n \{Y_i - (w_1 X_{1i} + w_2 X_{2i})\}^2$$



$$f(X) = w_0 + w_1 X_1 + w_2 X_2 \rightarrow \text{다중선형회귀 모델}$$

$$f(X) = \frac{1}{1 + e^{-(w_0 + w_1 X_1 + w_2 X_2)}} \rightarrow \text{로지스틱회귀 모델}$$

$$\underline{f(X)} = \sum_{m=1}^n k(m) I\{(x_1, x_2) \in R_m\} \quad \text{의사결정나무 모델}$$

$$f(X) = \frac{1}{1 + \exp\left(-\left(w_0 + w_1 \left(\frac{1}{1 + e^{-(w_{01} + w_{11} X_1 + w_{21} X_2)}}\right)\right) + w_2 \left(\frac{1}{1 + e^{-(w_{02} + w_{12} X_1 + w_{22} X_2)}}\right)\right)}$$

뉴럴네트워크 모델

모델 결정 → 파라미터 추정

$$\min_W \sum_{i=1}^n \{Y_i - f(X)\}^2$$

$$f(X) = w_0 + w_1 X_1 + w_2 X_2 \quad \text{다중선형회귀 모델}$$

$$\min_{w_0, w_1, w_2} \sum_{i=1}^n \{Y_i - (w_0 + w_1 X_1 + w_2 X_2)\}^2$$

Least square estimation algorithm

$$\hat{f}(X) = \hat{w}_0 + \hat{w}_1 X_1 + \hat{w}_2 X_2$$

모델 결정 → 파라미터 추정

$$\min_W \sum_{i=1}^n \{Y_i - f(X)\}^2$$

$$f(X) = \frac{1}{1 + e^{-(w_0 + w_1 X_1 + w_2 X_2)}} \quad \text{로지스틱회귀 모델}$$

$$\min_{w_0, w_1, w_2} \sum_{i=1}^n \left\{ Y_i - \left(\frac{1}{1 + e^{-(w_0 + w_1 X_1 + w_2 X_2)}} \right) \right\}^2$$

Conjugate gradient algorithm

$$\hat{f}(X) = \frac{1}{1 + e^{-(\hat{w}_0 + \hat{w}_1 X_1 + \hat{w}_2 X_2)}}$$

모델 결정 → 파라미터 추정

$$\min_W \sum_{i=1}^n \{Y_i - f(X)\}^2$$

뉴럴네트워크 모델

$$f(X) = \frac{1}{1 + \exp \left(- \left(w_0 + w_1 \left(\frac{1}{1 + e^{-(w_{01} + w_{11} X_1 + w_{21} X_2)}} \right) + w_2 \left(\frac{1}{1 + e^{-(w_{02} + w_{12} X_1 + w_{22} X_2)}} \right) \right) \right)}$$

$$\min_{w_0, \dots, w_{22}} \sum_{i=1}^n \left\{ Y_i - \left(\frac{1}{1 + \exp \left(- \left(w_0 + w_1 \left(\frac{1}{1 + e^{-(w_{01} + w_{11} X_1 + w_{21} X_2)}} \right) + w_2 \left(\frac{1}{1 + e^{-(w_{02} + w_{12} X_1 + w_{22} X_2)}} \right) \right) \right) \right\}^2$$

Backpropagation algorithm

$$\hat{f}(X) = \frac{1}{1 + \exp \left(- \left(\hat{w}_0 + \hat{w}_1 \left(\frac{1}{1 + e^{-(\hat{w}_{01} + \hat{w}_{11} X_1 + \hat{w}_{21} X_2)}} \right) + \hat{w}_2 \left(\frac{1}{1 + e^{-(\hat{w}_{02} + \hat{w}_{12} X_1 + \hat{w}_{22} X_2)}} \right) \right) \right)}$$

모델 결정 → 파라미터 추정

$$f(X) = w_0 + w_1X_1 + w_2X_2 \quad \hat{f}(X) = \hat{w}_0 + \hat{w}_1X_1 + \hat{w}_2X_2$$

다중선형회귀 **모델** Least square estimation algorithm

$$f(X) = \frac{1}{1 + e^{-(w_0 + w_1X_1 + w_2X_2)}} \quad \hat{f}(X) = \frac{1}{1 + e^{-(\hat{w}_0 + \hat{w}_1X_1 + \hat{w}_2X_2)}}$$

로지스틱회귀 **모델** Conjugate gradient algorithm

$$f(X) = \frac{1}{1 + \exp\left(-\left(w_0 + w_1\left(\frac{1}{1 + e^{-(w_{01} + w_{11}X_1 + w_{21}X_2)}}\right) + w_2\left(\frac{1}{1 + e^{-(w_{02} + w_{12}X_1 + w_{22}X_2)}}\right)\right)\right)}$$

뉴럴네트워크 **모델** Backpropagation algorithm

$$\hat{f}(X) = \frac{1}{1 + \exp\left(-\left(\hat{w}_0 + \hat{w}_1\left(\frac{1}{1 + e^{-(\hat{w}_{01} + \hat{w}_{11}X_1 + \hat{w}_{21}X_2)}}\right) + \hat{w}_2\left(\frac{1}{1 + e^{-(\hat{w}_{02} + \hat{w}_{12}X_1 + \hat{w}_{22}X_2)}}\right)\right)\right)}$$

머신러닝 모델 학습 과정 요약

1. 모델 결정하기 (Y를 표현하기 위한 X들을 조합 방식 결정)
2. 모델을 구성하는 파라미터 찾기 (모델의 **핵심!!**)

어떻게?

가지고 있는 데이터를 이용하여

무엇을 추구하며?

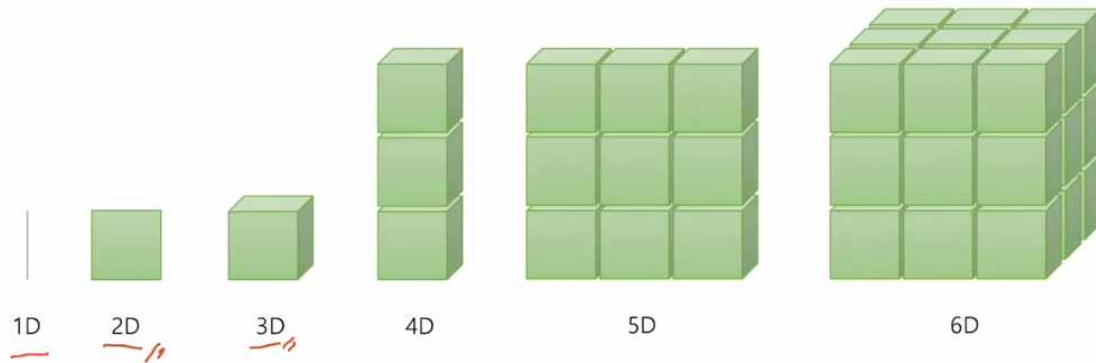
실제 데이터의 값과 최대한 같게 나오도록!

참조:

<https://www.youtube.com/watch?v=pFyFHUmXgu0&list=PLpIPLT0Pf7IoTxTCi2MEQ94MZnHaxrP0j&index=26>

※ 벡터(크기와 방향이 있는), 매트릭(행렬), 텐서(3D)

Vector, Matrix and Tensor



※ 벡터 설명 참조 사이트: https://www.youtube.com/watch?v=fNk_zzaMoSs

1. 기계학습은 학습 방식에 따라 크게 세 가지로 분류

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**

Categories of ML Problems

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Discrete	Classification	Clustering	Discrete Action Space Agent
Continuous	Regression	Dimensionality Reduction	Continuous Action Space Agent

Semi-Supervised Learning

※ Dimensionality Reduction(차원축소): Feature가 많다고 해서 모든 Feature를 사용하여 모델을 만드는 것은 좋은 방법이 아니라 설명력이 높은 Feature만

사용해야 한다는 것. 즉, 설명력이 높은 Feature만 사용하여 모델을 생성한다는 것은 모든 차원을 사용하지 않고 차원은 축소해 설명력이 높은 모델을 생성한다는 의미

※ Semi-supervised Learning(준지도학습)

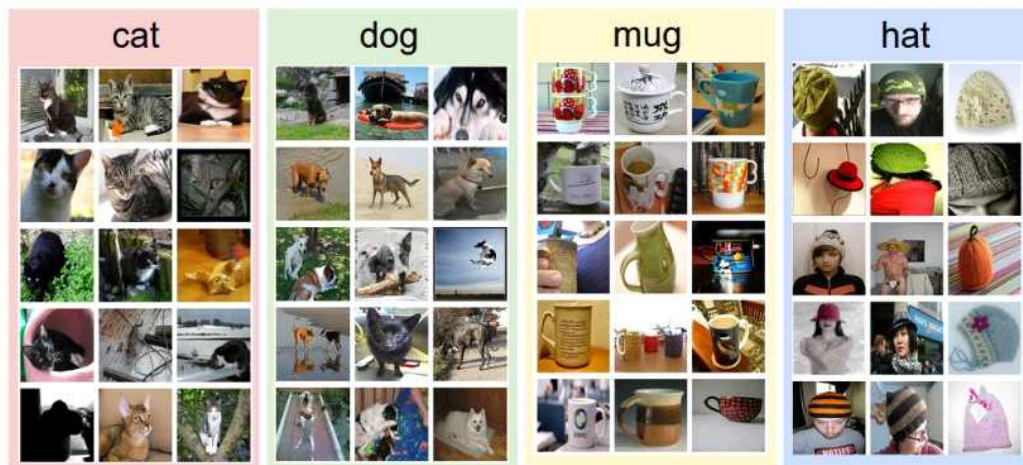
레이블이 달려있는 데이터와 레이블이 달려있지 않은 데이터를 동시에 사용해서 더 좋은 모델을 만들자는 것이 목표.

※ Clustering(군집화)

개체들이 주어졌을 때, 개체들을 몇 개의 클러스터(부분 그룹)으로 나누는 과정을 의미. 이렇게 개체들을 그룹으로 나누는 과정을 통해서, 클러스터 내부 멤버들 사이는 서로 가깝거나 비슷하게, 서로 다른 두 클러스터 사이의 멤버 간에는 서로 멀거나 비슷하지 않게 하는 것이 클러스터링의 목표.

1) Supervised Learning(지도학습)

가) 투입에 대한 산출의 답이 미리 주어진 경우



An example training set for four visual categories. In practice we may have thousands of categories and hundreds of thousands of images for each category.

나) 대표적으로 태스크로 분류 (classification)와 회귀(Regression)가 있음.

- 분류는 위와 같이 이미지 안에 있는 물체를 무엇으로 분류할지에 대한 태스크(task, 작업 단위의 실행 단위, 프로세스, 요청)이다.
- 물체가 나와 있는 사진과 그 물체가 무엇인지에 대한 정보를 같이 주어서 학습시키는 방식

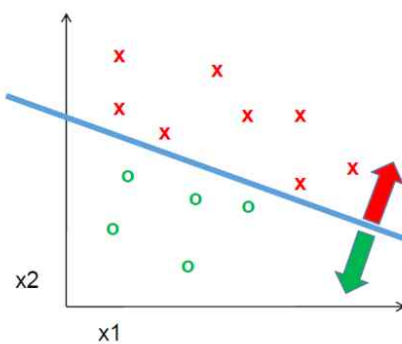
Classification Problem



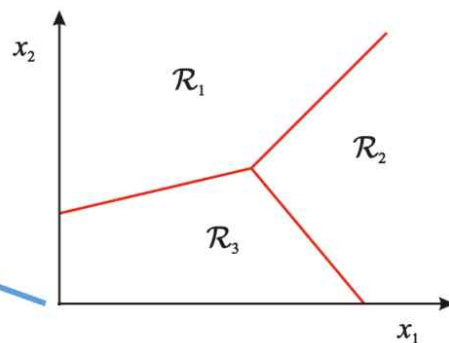
Chihuahua or Muffin?

Classification Problem

Identifying which of a set of categories a new instance belongs



Binary Classification



Multi-class Classification

- 회귀(Regression)는 연속적인 값을 추정하는 태스크.

다) Image Classification(이미지 분류)

사진 분류 태스크는

- 투입 및 산출 (Input and output): 투입은 N개의 이미지로 구성되며 K개의 물체 분류 중 하나로 산출의 답이 정해져있습니다. 이 데이터들을 훈련 집합 (training set)이라고 부른다.
- 학습 (Learning): 훈련 집합을 통해, 태스크에서 물체 분류를 잘 수행할 수 있도록 학습이 이루어진다. 이것을 분류기 (classifier)를 훈련한다, 혹은 모델 (model)을

학습한다고 지칭함

- 평가 (Evaluation): 마지막으로 학습이 잘 이루어졌는지 시험 집합 (test set)을 통해 검증 단계를 거친다. 모델의 정확도 (accuracy)가 얼마나 되는지 확인할 수 있다.

2) Unsupervised Learning(비지도 학습)

- 산출에 대한 아무런 정보가 없을 경우. 오직 투입 데이터들만 가지고 학습
- 레이블 없이 데이터를 보고 스스로 학습

● Unsupervised learning: un-labeled data

- Google news grouping
- Word clustering

3) Reinforcement Learning(강화학습)

정해진 답이 주어져있지는 않지만, 행동을 취한 뒤에 그 행동을 평가할 수 있는 경우(보상이 주어지는)/ 주로 체스, 바둑과 같은 게임 태스크를 위한 학습 방법

[머신러닝 기초(Machine learning Basics)]

※SK Planet FACEID Concept: <https://www.youtube.com/watch?v=RUdVbcefzol&t=4s>

1. data acquisition(획득)

먼저 카메라를 찍고, 그중에서 얼굴 부분만 잘라낸 image data를 확보



그림1. Image Capturing

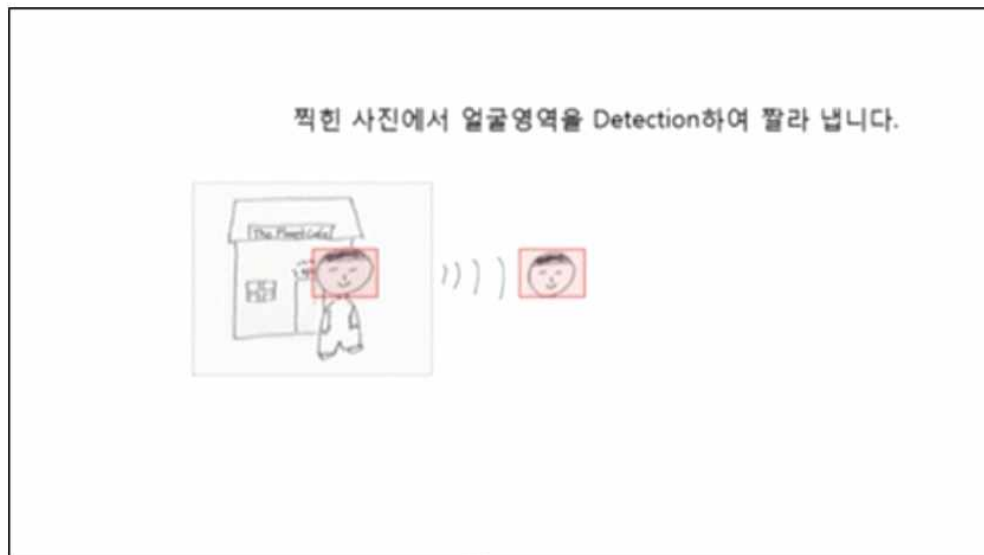


그림 2. Face Detection

feature extraction

Machine은 확보된 얼굴 영상을 그대로 인식하지는 않습니다. 보다 인식하기 쉽도록 feature를 추출합니다.

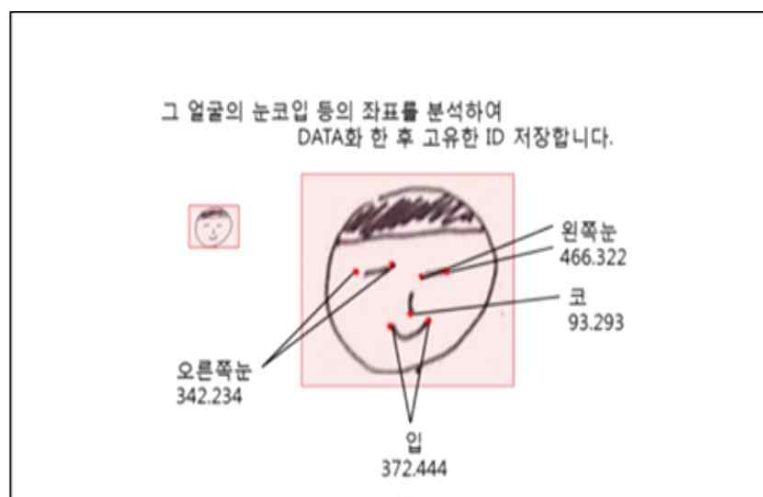


그림 3. Feature Extraction

Classifier

그리고 해당 feature를 기존에 저장되어 있던 data와 비교하여 가장 유사도가 높은 ID를 찾아냅니다. 여기서 기존에 저장되어 있던 data란 이러한 feature가 들어오면 이러한 ID일 확률이 높아요...에 대한 정보를 의미합니다. 이렇게 data를 실제 test 전에 저장하는 것을 training 한다고 하죠. 즉, training 된 classifier에 feature를 입력해 넣으면 어떤 ID에 해당하는 지 알려줍니다. 이것이 test 단계입니다.

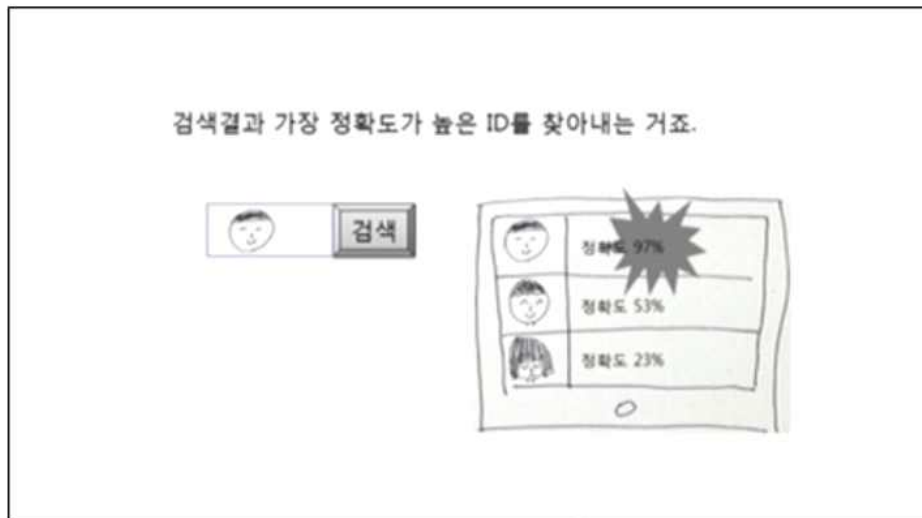


그림 4. Classifier

이를 종합해 보면 Machine Learning은 '미리 준비된 data를 이용하여 classifier를 training(learning)시키는 Training 단계'와, 'train된 classifier를 이용하여 입력된 data가 어떤 분류결과인지 확인하는 Test 단계'로 나뉘게 됩니다.

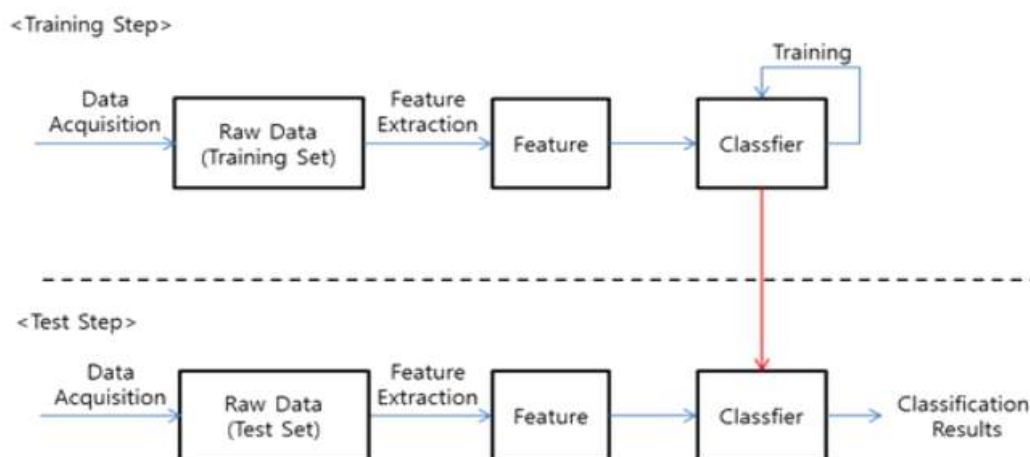


그림 5. machine learning의 개념도

2. Feature Extraction

machine learning을 위해 어떤 feature를 사용해야 좋을까요? 이 문제는 어떤 목적을 위해 machine learning을 사용하는 것인지에 따라 달라질 것입니다. 가령, 곧 급등할 주식을 찾고 싶다!!! 라고 한다면, 위 그림 5는 어떻게 표현될까요?

일단, Raw Data = '매일매일의 주식 종목별 주가' 가 될 것입니다. 그런데 이 Raw Data를 주식 종목이 처음 상장된 날부터 오늘까지의 값이라면, 그것은 너무도 방대한 양의 data일 것입니다. 이것을 classifier의 입력으로 사용했다가는 엄청난 계산량에 training도 못하고 GG를 선언하겠지요. 따라서, 그거보다는 더 압축(?)된 형태의 Feature를 Classifier의 입력으로 사용해야 할 것입니다. 가령 다음과 같은 것들이 Feature의 예가 될 수 있겠지요.

Feature = 5일/20일/60일 이동평균 값, MACD, 기업 정보(PER, PBR, ROE ...), 전저점/전고점 값, candle, 거래량 등등 이외에도 수십~수백가지 이상의 지표들이 있겠지요.

그 중 어떤 feature를 사용하면, 남들보다 조금은 더 높은 확률로 급등주를 찾을 수 있을까요? 알면 모두가 이미 워렌 버핏이 되어 있겠죠? 어려운 문제입니다. 즉, 어떤 feature를 선택하고 extract할지를 결정하는 것은 매우 어려운 문제가 됩니다. 다행히도 얼굴 인식, 음성 인식 등 특정 분야에서서는 어떤 feature가 효과적인지 많은 분들이 이미 연구해 놓았기 때문에 대부분의 엔지니어들은 가져다 쓰기만 해도 되겠지요..

※ 태깅(Tagging)이란 사이트의 관리자가 사이트의 이미지나 텍스트를 관련된 주제나 카테고리의 형태로 키워드 처리를 해주는 것으로, 쉽게 말하면 태그를 다는 것이다.

3. Classifier

Classifier는 machine learning 알고리즘의 핵심이겠지요. Neural Network, Support Vector Machine, AdaBoost, Random Forest 등 많은 알고리즘들이 있습니다. 이에 대한 소개는 다음 장에서 하기로 하고....

일반적으로 Classifier를 training하는데에는 많은 양의 data가 필요하게 됩니다. 하지만 data를 수집하는 것 자체가 굉장히 비용이 많이 듭니다. Facebook 정도 되면 tagging을 이용하여 많은 양의 얼굴 데이터를 확보하는 데 어려움이 없겠지만, 대부분의 연구소들은 그렇게 방대한 양의 얼굴데이터를 수집하기 어렵겠지요. 그래서 한정된 양의 database로 training 시키기 위해 Cross Validation 기법이나 Bootstrap 기법 등이 사용된다고 합니다.

Cross Validation 은 database중 일부를 이용해서 training set 으로 정하되, 그 일부를 바꿔가면서 training and test 하는 방법입니다. 다음 그림 6에 잘 나타나 있네요.

Bootstrap은 training set을 선택하는데 있어서, 정해진 값을 선택하는 것이 아니라, 해당 값의 확률분포를 modeling하고, 그 model에서 원하는 만큼의 data를 뽑아서 사용하는 것입니다. 가령 1,2,3,4 라는 데이터가 있으면, 1,2,4는 나올 확률이 0.2 이지만, 3은 0.4가 되죠. 이 확률값을 바탕으로 data를 resampling하여 데이터를 재구성합니다.

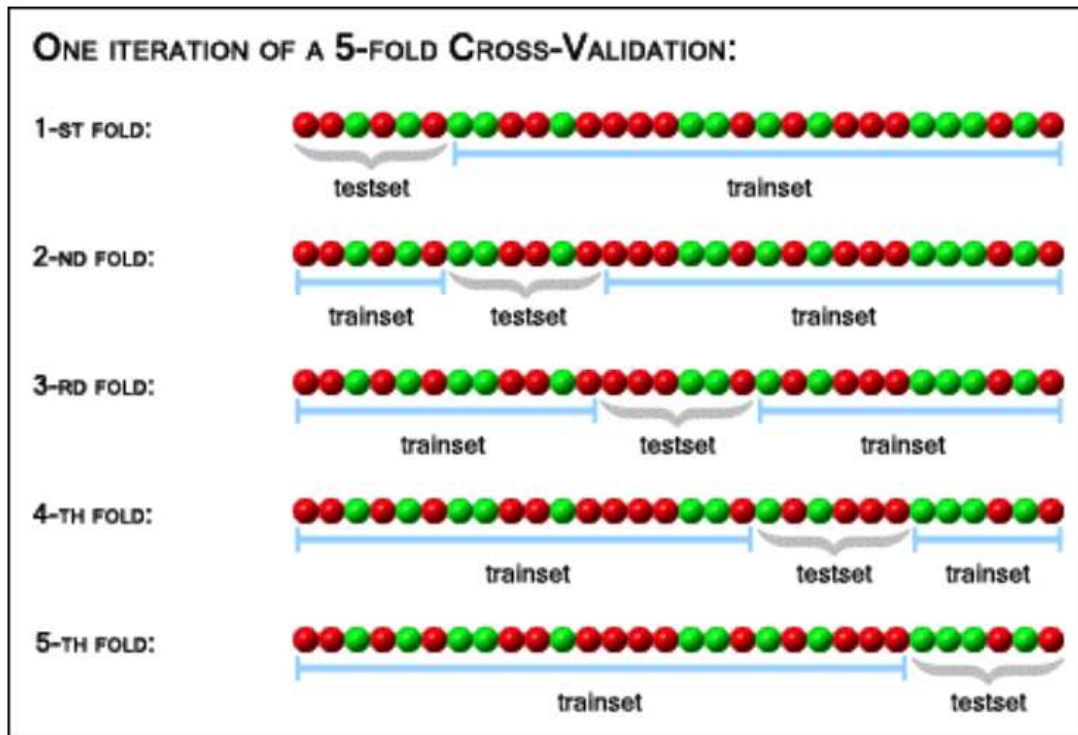


그림 6. Cross Validation

◆ BIG GAN 체험해보기

구글콜랩;

https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/biggan_generation_with_tf_hub.ipynb#scrollTo=dSAyfDfnVugs



◆ NeuralStyleTransfer.ipynb 체험해보기

구글콜랩;

<https://colab.research.google.com/github/titu1994/Neural-Style-Transfer/blob/master/NeuralStyleTransfer.ipynb#scrollTo=XANFbnpsfCj3>

■ 3교시

참조 사이트:

<https://www.youtube.com/watch?v=4Yo297HQyAk&list=UUueLU1pCvFlM8Y8sth7a6RQ&index=45>

1. 선형 회귀(Linear Regression)

- 변수 x 의 값은 독립적으로 변할 수 있는 것일 때, y 값은 계속해서 x 의 값에 의해서, 종속적으로 결정되면 x 를 독립 변수, y 를 종속 변수라고 한다.
- 선형 회귀는 한 개 이상의 독립 변수 x 와 y 의 선형 관계를 모델링한다.
- 독립 변수 x 가 1개라면 단순 선형 회귀라고 부른다.

2. 단순 선형 회귀 분석(Simple Linear Regression Analysis)

$y=Wx+b$ (단순 선형 회귀의 수식)

- 독립 변수 x , 곱해지는 값 W 는 가중치(weight), 별도로 더해지는 값 b 를 편향(bias)이라고 함.
 - 직선의 방정식에서는 각각 직선의 기울기와 절편을 의미
- W 와 b 가 없이 y 와 x 란 수식은 y 는 x 와 같다는 하나의 식밖에 표현하지 못한다. 그래프 상으로 말하면, 하나의 직선밖에 표현하지 못하는 것을 뜻함.

$y=x$

다시 말해 W 와 b 의 값을 적절히 찾아내면 x 와 y 의 관계를 적절히 모델링한 것이 된다.

3. 다중 선형 회귀 분석(Multiple Linear Regression Analysis)

$y=W_1x_1+W_2x_2+...W_nx_n+b$

예를 들면 집의 매매 가격은 단순히 집의 평수가 크다고 결정되는 게 아니라 집의 층의 수, 방의 개수, 지하철 역과의 거리와도 영향이 있다. 이러한 다수의 요소를 가지고 집의 매매 가격을 예측해보고 싶다면 y 는 여전히 1개이지만 x 는 1개가 아니라 여러 개가 되었다. 이를 다중 선형 회귀 분석이라고 함.

선형회귀모델

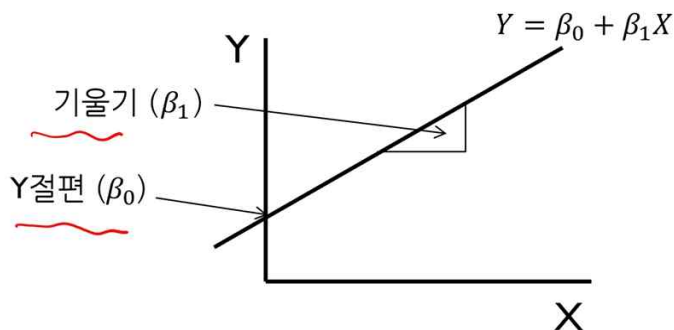
- 선형회귀모델: 출력변수 Y를 입력변수 X들의 선형결합으로 표현한 모델



선형결합: 변수들을 (상수 배와) 더하기 빼기를 통해 결합

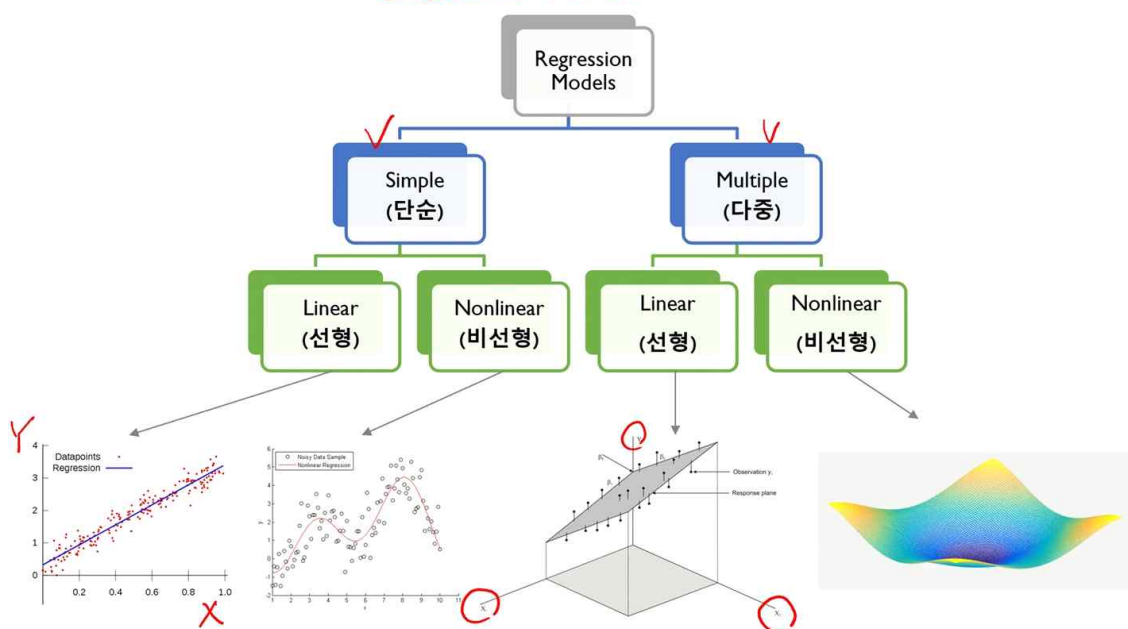
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

- X변수 한 개가 Y를 표현하는 경우: $Y = \beta_0 + \beta_1 X$ (직선 식)

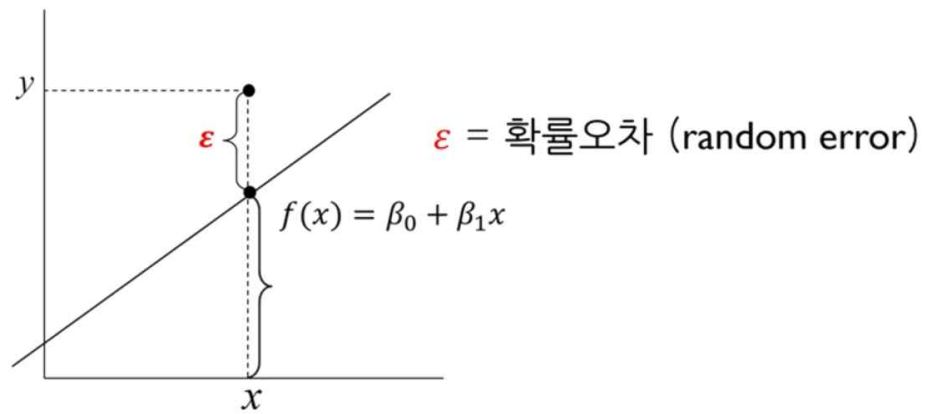


선형회귀모델 분류

X변수의 수, X변수와 Y변수의 선형성 여부에 따라 구분



선형회귀 모델 가정

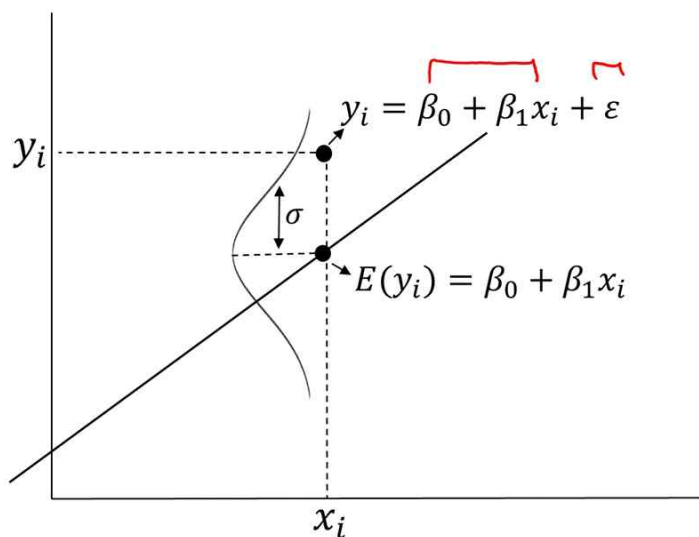


확률오차 가정 : $\varepsilon_i \sim \text{정규분포}$ $E(\varepsilon_i) = 0$ $V(\varepsilon_i) = \sigma^2$ for all i .

$$\varepsilon_i \sim N(0, \sigma^2), i = 1, 2, \dots, n$$

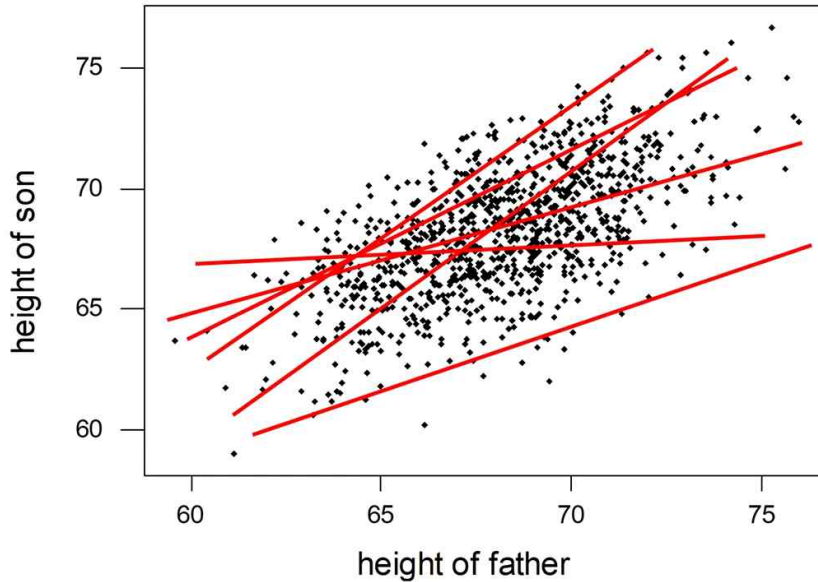
선형회귀 모델 가정

$$Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma^2), i = 1, 2, \dots, n$$



선형회귀 모델

$$E(Y) = f(X) = \beta_0 + \beta_1 X_1$$



선형회귀 모델

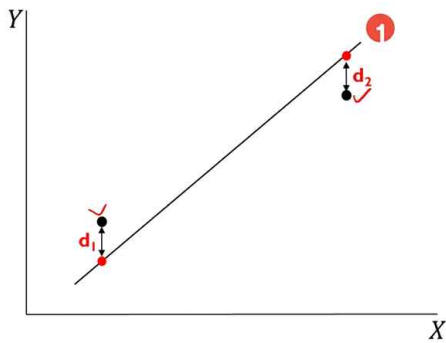
$$E(Y) = f(X) = \beta_0 + \beta_1 X_1$$

파라미터 (Parameter)

파라미터를 찾자 (추정하자)

가지고 있는 데이터들의 함수식으로!

파라미터 추정



$$d_1 + d_2 + \dots + d_n = 0$$

$$d_1^2 + d_2^2 + \dots + d_n^2 \geq 0$$

$$d_1 = Y_1 - E(Y_1) \\ = Y_1 - (\beta_0 + \beta_1 X_1)$$

$$\sum_{i=1}^n d_i^2 = \sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$

$$\min_{\beta_0, \beta_1} \sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$

Cost function
(비용함수)

파라미터 추정

$$\min_{\beta_0, \beta_1} \sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$

Cost function (비용함수)

Algorithm

$$\hat{\beta}_0, \hat{\beta}_1$$

$$f(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$

파라미터 추정 알고리즘

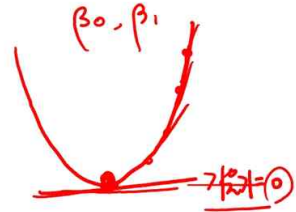
$$\min_{\beta_0, \beta_1} \sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$

Cost function (비용함수)

- Cost function is convex \rightarrow globally optimal solution exists (전역 최적해 존재)

$$\frac{\partial C(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{i=1}^n Y_i - (\beta_0 + \beta_1 X_i) = 0$$

$$\frac{\partial C(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{i=1}^n Y_i - (\beta_0 + \beta_1 X_i) X_i = 0$$



파라미터 추정 알고리즘

Question. Find estimator of β_0 and β_1 (i.e., $\hat{\beta}_0$ and $\hat{\beta}_1$)

Step 1. $\sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$

Step 2. $\min_{\beta_0, \beta_1} \sum_{i=1}^n \{Y_i - (\beta_0 + \beta_1 X_i)\}^2$

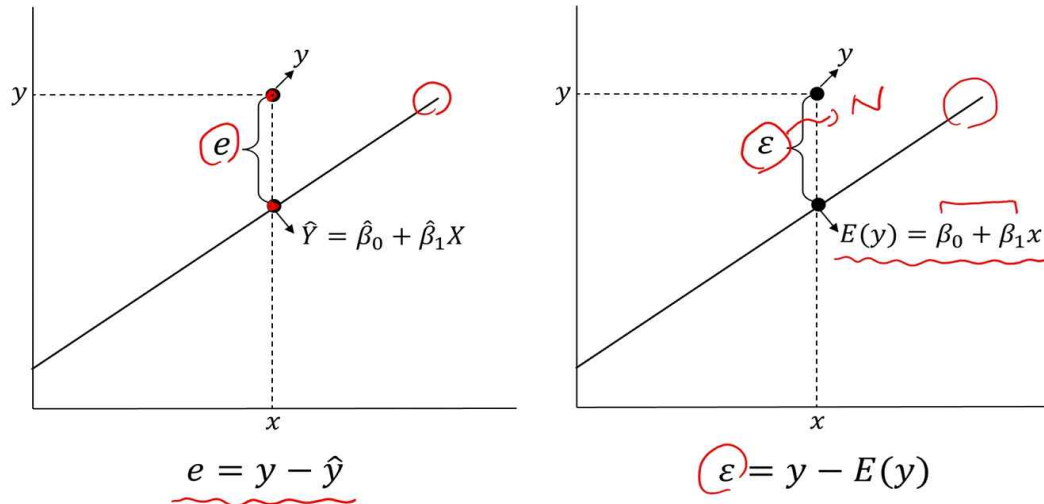
Step 3.
$$\begin{aligned} \frac{\partial C(\beta_0, \beta_1)}{\partial \beta_0} &= -2 \sum_{i=1}^n Y_i - (\beta_0 + \beta_1 X_i) = 0 \\ \frac{\partial C(\beta_0, \beta_1)}{\partial \beta_1} &= -2 \sum_{i=1}^n Y_i - (\beta_0 + \beta_1 X_i) X_i = 0 \end{aligned}$$

Solutions. $\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}, \quad \hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$

Algorithm: Least Squares Estimation Algorithm (최소제곱법, 최소자승법)

※ 알고리즘; 어떤 문제를 풀기위한 체계적인 방법론, 집합

잔차 (Residual)



- 잔차 e 는 확률오차 ε 이 실제로 구현된 값

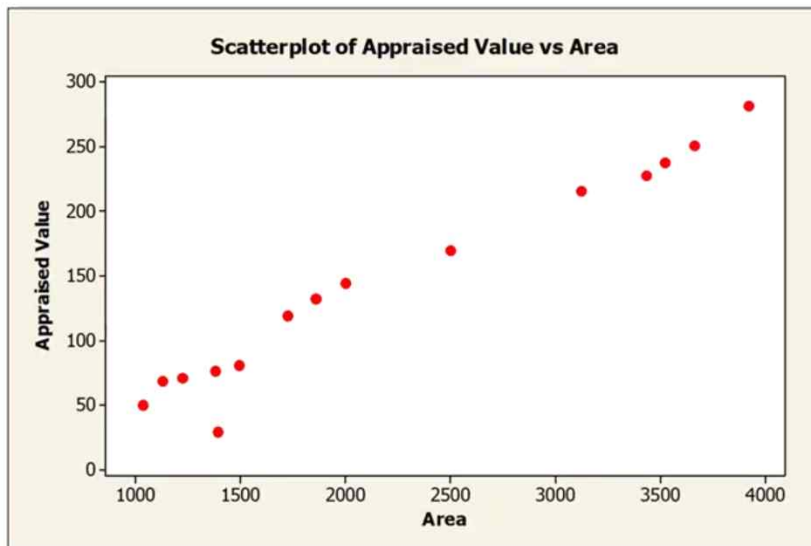
※ 잔차와 오차의 차이; 확률오차는 예측 값, 잔차는 확률오차의 실제로 구현된 값

◆ 선형회귀모델 문제풀기

선형회귀 모델 예제

관측치	Area, 집크기 (X)	Appraised Value, 집가격 (Y)
1	1380	76
2	3120	216
3	3520	238
4	1130	69
5	1030	50
6	1720	119
7	3920	282
8	1490	81
9	1860	132
10	3430	228
11	2000	145
12	3660	251
13	2500	170
14	1220	71
15	1390	29

선형회귀분석 모델 예제



선형회귀 모델 예제

The regression equation is

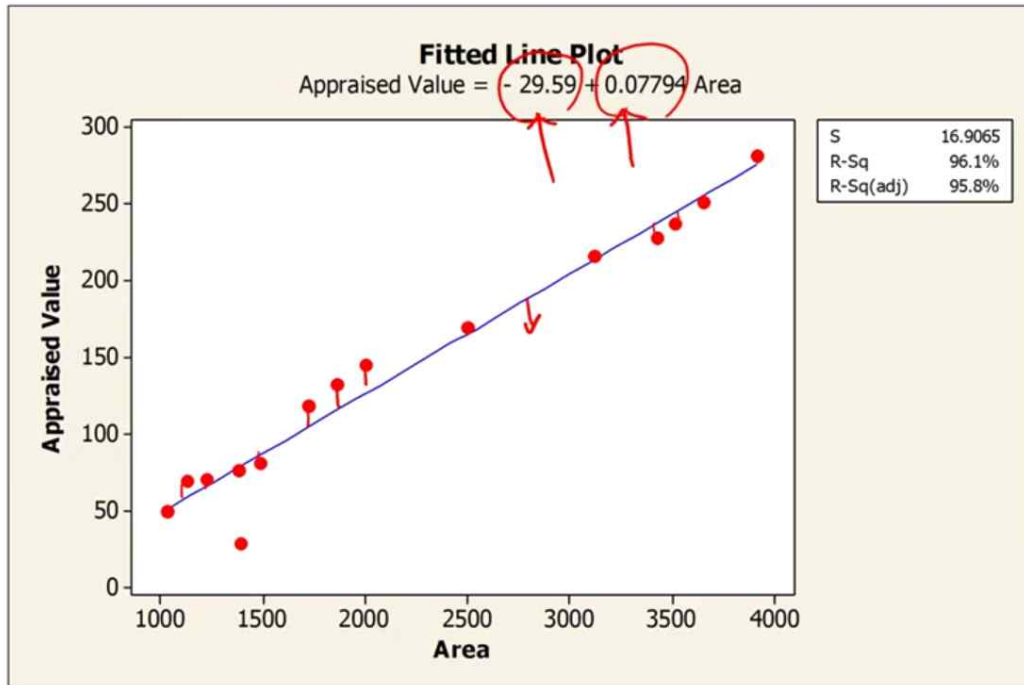
$$\text{Appraised Value (집 가격)} = -29.6 + 0.0779 \text{ Area (집 크기)}$$

$$\hat{\beta}_1 = 0.077939$$

집 크기가 1 square foot 증가할 때마다 집 가격은 0.077 (\$ thousand) 증가

\$ 77

선형회귀 모델 예제



※ 오차가 가장 작은 추정값 = Fitted Line Plot: Appraised Xvalue(평가 값)

파라미터 추정 알고리즘

Least square estimator

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \quad \hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

- Estimator(추정량): 샘플의 함수 (a function of the samples)
- 추정량의 용도: 알려지지 않은 파라미터를 추정 β_0, β_1
- 추정량의 종류
 - (1) 점추정 (point estimator), (2) 구간추정 (interval estimator)

파라미터에 대한 점추정 (Point Estimator)

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \quad \varepsilon_i \sim N(0, \sigma^2), \quad i = 1, 2, \dots, n$$

$$\beta_0 \text{에 대한 점추정 식: } \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

$$\beta_1 \text{에 대한 점추정 식: } \hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

$$\sigma^2 \text{에 대한 점추정 식: } \hat{\sigma}^2 = \left(\frac{1}{n-2} \right) \sum_{i=1}^n e_i^2$$

↓ ↓
샘플 잔차 (residual)

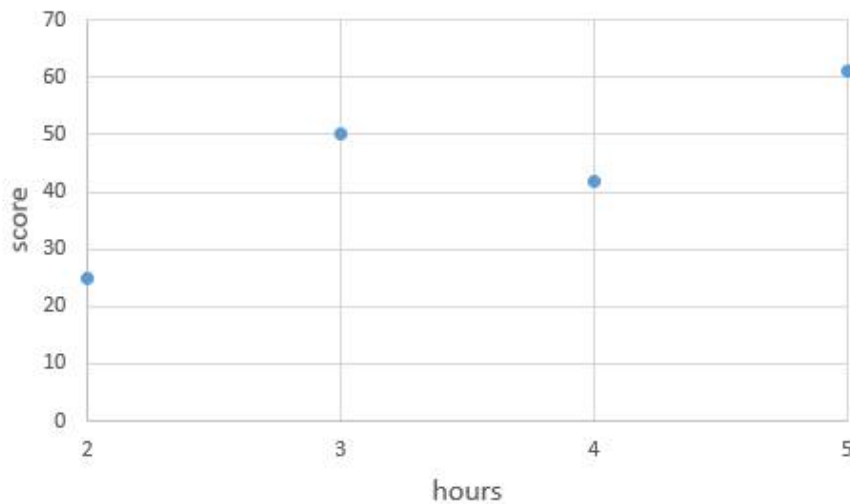
■ 선형회귀분석 식 구하기

1. 가설(Hypothesis) 세우기

어떤 학생의 공부 시간에 따라서 다음과 같은 점수를 얻었다.

hours(x)	score(y)
2	25
3	50
4	42
5	61

>> 이를 좌표 평면에 그려보면



학생이 6시간, 7시간, 8시간을 공부하였을 때의 성적을 예측한다면?

x와 y의 관계를 유추하기 위해서 수학적으로 식을 세워보게 되는데 이러한 식을 가설 (Hypothesis)이라고 함

$H(x)=Wx+b$ (H(x)에서 H는 Hypothesis를 뜻함)

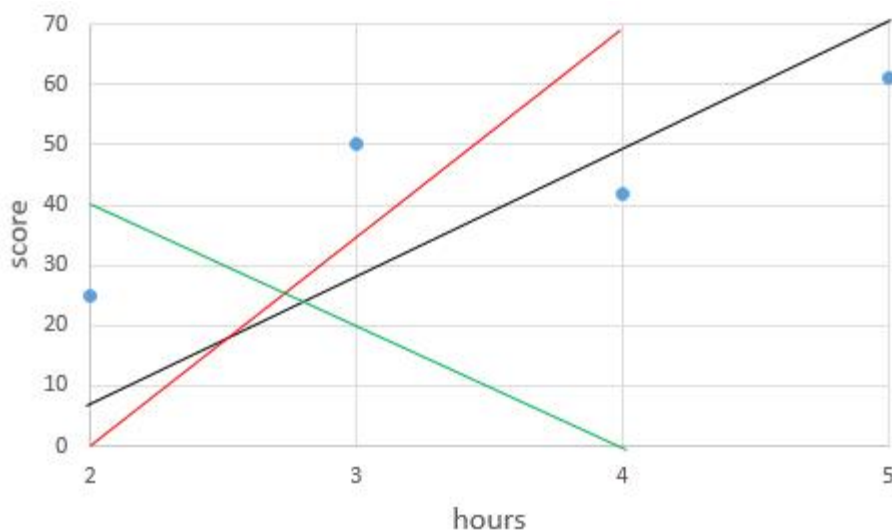
- 선형 회귀의 가설은

W와 b의 값에 따라서 천차만별로 그려지는 직선의 모습.

- W는 직선의 기울기이고 b는 절편으로 직선을 표현함

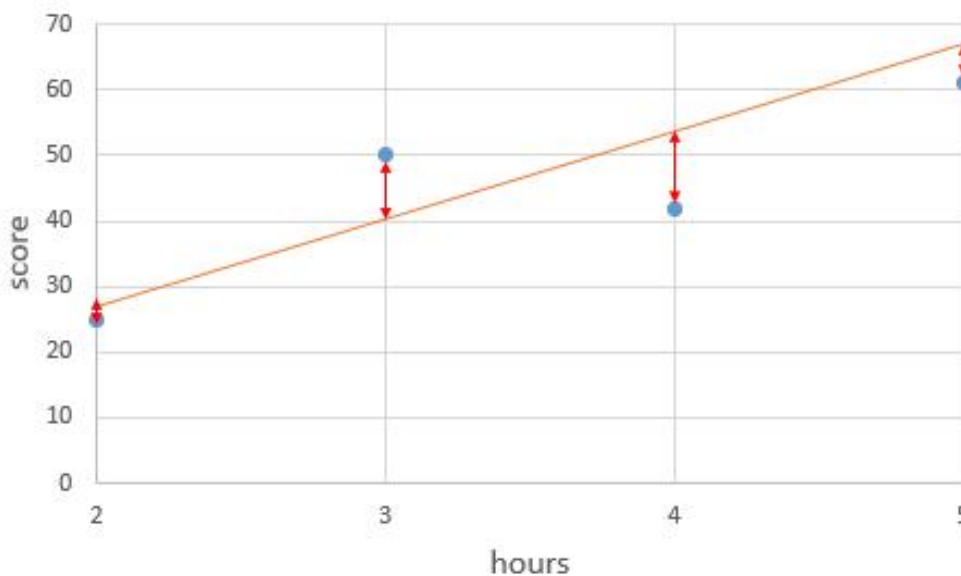
- 선형 회귀는 주어진 데이터로부터 y와 x의 관계를 가장 잘 나타내는 직선을 그리는 것을 뜻함.

그리고 어떤 직선인지 결정하는 것은 W와 b의 값이므로 선형 회귀에서 해야할 일은 결국 적절한 W와 b를 찾아내는 일이 된다.



■ 비용 함수(Cost function) : 평균 제곱 오차(MSE)

- 머신 러닝은 W 와 b 를 찾기 위해서 실제 값과 가설로부터 얻은 예측값의 오차를 계산하는 식을 세우고, 이 식의 값을 최소화하는 최적의 W 와 b 를 찾아내는 것
- 실제값과 예측값에 대한 오차에 대한 식을 목적 함수(Objective function) 또는 비용 함수(Cost function) 또는 손실 함수(Loss function)라고 함
- 함수의 값을 최소화하거나, 최대화하거나 하는 목적을 가진 함수를 목적 함수(Objective function)라고 함
- 값을 최소화하려고 하면 이를 비용 함수(Cost function) 또는 손실 함수(Loss function)라고 함
- 비용 함수는 단순히 실제값과 예측값에 대한 오차를 표현하면 되는 것이 아니라, 예측값의 오차를 줄이는 일에 최적화 된 식이어야 한다.
- 회귀 문제의 경우 비용함수는 주로 평균 제곱 오차(Mean Squared Error, MSE)가 사용



- 오차(error)란?

- 1) 오차는 주어진 데이터에서 각 x 에서의 실제값 y 와 위의 직선에서 예측하고 있는 $H(x)$ 값의 차이
- 2) 위의 그림에서 \updownarrow 는 각 점에서의 오차의 크기를 보여줌.
오차를 줄여가면서 W 와 b 의 값을 찾아내기 위해서는 전체 오차의 크기를 구해야 한다.
- 3) 오차의 크기를 측정하기 위한 가장 기본적인 방법은 각 오차를 모두 더하는 방법.
- 4) 위의 $y=13x+1$ 직선이 예측한 예측값을 각각 실제값으로부터 오차를 계산하여 표를 만들어보면 아래와 같다.

hours(x)	2	3	4	5
실제값	25	50	42	61
예측값	27	40	53	66
오차	-2	10	-7	-5

- 수식적으로 '오차 = 실제값 - 예측값' 이라고 정의한 후에 모든 오차를 제곱하여 더하는 방법을 사용

- 다시 말해 위의 그림에서의 모든 점과 직선 사이의 ↑ 거리를 제곱하고 모두 더한다.

이를 수식으로 표현하면

$$\sum_{i=1}^n \left[y^{(i)} - H(x^{(i)}) \right]^2 = (-2)^2 + 10^2 + (-7)^2 + (-5)^2 = 178$$

이때 데이터의 개수인 n으로 나누면, 오차의 제곱합에 대한 **평균**을 구할 수 있는데 이를 평균 제곱 오차(Mean Squared Error, MSE)라고 함.

수식은 아래와 같음

$$\frac{1}{n} \sum_{i=1}^n \left[y^{(i)} - H(x^{(i)}) \right]^2 = 178/4 = 44.5$$

y=13x+1의 예측값과 실제값의 평균 제곱 오차의 값은 44.5이다.

- 평균 제곱 오차의 값을 최소값으로 만드는 W와 b를 찾아내는 것이 정답인 직선을 찾아내는 일인데

평균 제곱 오차를 W와 b에 의한 비용 함수(Cost function)로 재정의해 보면 다음과 같다.

$$cost(W, b) = \frac{1}{n} \sum_{i=1}^n \left[y^{(i)} - H(x^{(i)}) \right]^2$$

모든 점들과의 오차가 클수록 평균 제곱 오차는 커지며, 오차가 작아질수록 평균 제곱 오차는 작아진다.

그러므로 이 평균 제곱 오차, 즉, Cost(W,b)를 최소가 되게 만드는 W와 b를 구하면

결과적으로 y 와 x 의 관계를 가장 잘 나타내는 직선을 그릴 수 있다.

$$W, b \rightarrow \text{minimize } \text{cost}(W, b)$$

◆ 선형 회귀(Linear Regression) 문제 풀기

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

x	Y
1	1
2	2
3	3

- $W=1, \text{cost}(W)=0$

$$\frac{1}{3}((1 * 1 - 1)^2 + (1 * 2 - 2)^2 + (1 * 3 - 3)^2)$$

- $W=0, \text{cost}(W)=4.67$

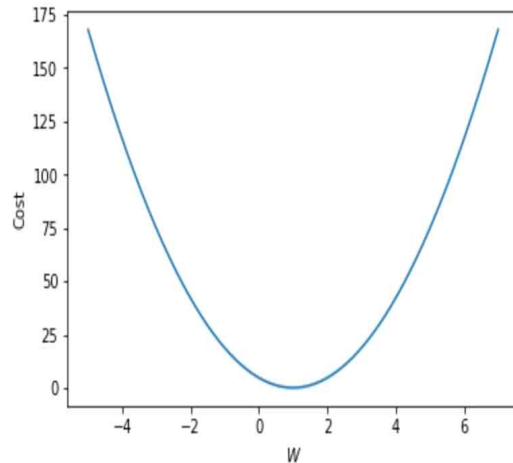
- $W=2, \text{cost}(W)=?$

■ 옵티마이저(Optimizer) : 경사하강법(Gradient Descent)

Gradient Descent: Intuition

- 곡선을 내려가자
- 기울기가 클수록 더 멀리!
- “Gradient” 를 계산하자

$$\frac{\partial cost}{\partial W} = \nabla W$$



Gradient Descent: The Math

$$cost(W) = \frac{1}{m} \sum_{i=1}^m (W x^{(i)} - y^{(i)})^2$$

$$\nabla W = \frac{\partial cost}{\partial W} = \frac{2}{m} \sum_{i=1}^m (W x^{(i)} - y^{(i)}) x^{(i)}$$

$$W : = W - \alpha \nabla W$$

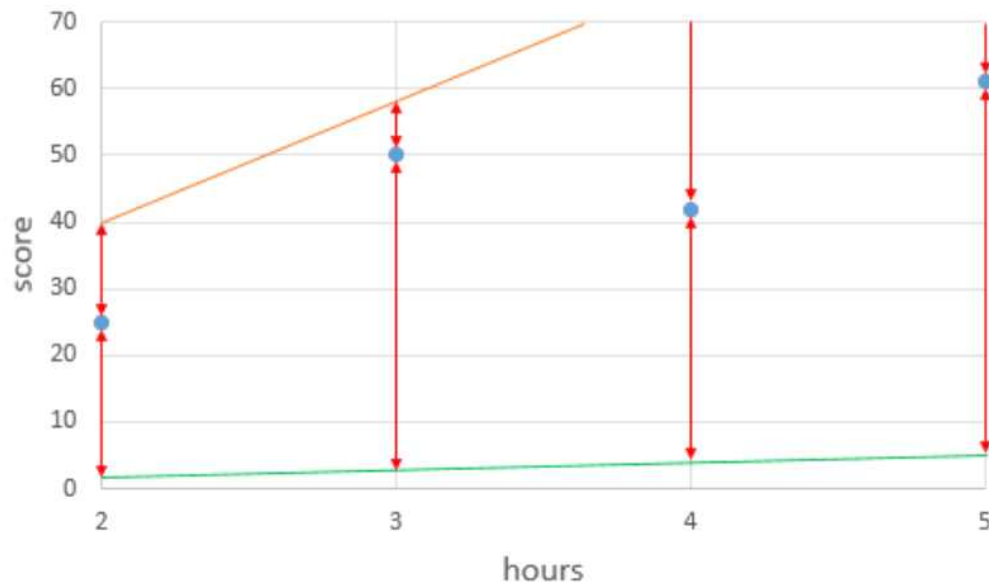
Learning rate

Gradient

- 선형 회귀 및 머신 러닝, 딥 러닝의 학습은 비용 함수를 최소화하는 매개 변수인 W와 b를 찾기 위한 작업.

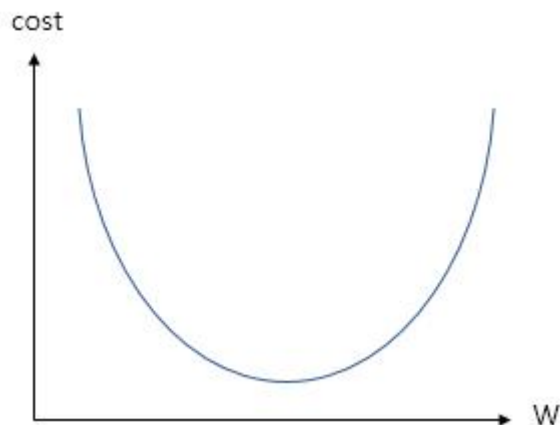
- 이때 사용되는 알고리즘을 옵티마이저(Optimizer) 또는 최적화 알고리즘이라고 부름.

- 옵티마이저를 통해 적절한 W 와 b 를 찾아내는 과정을 머신 러닝에서 학습(training)이라고 부른다.
- 가장 기본적인 옵티마이저 알고리즘이 경사 하강법(Gradient Descent)
- W 는 가중치(weight)를 뜻하고, 직선의 방정식에서 직선의 기울기를 의미함
따라서 그래프는 기울기 W 가 지나치게 높거나, 낮을 때 어떻게 오차가 커지는지를 보여준다.



- 주황색선은 기울기 W 가 20일 때, 초록색선은 기울기 W 가 1일 때를 보여줌.
- 각각 $y=20x$, $y=x$ 에 해당되는 직선
- ↑는 각 점에서의 실제값과 두 직선의 예측값과의 오차
- 기울기가 지나치게 크면 실제값과 예측값의 오차가 커지고, 기울기가 지나치게 작아도 실제값과 예측값의 오차가 커진다.
- b 또한 마찬가지인데 b 가 지나치게 크거나 작으면 오차가 커진다.

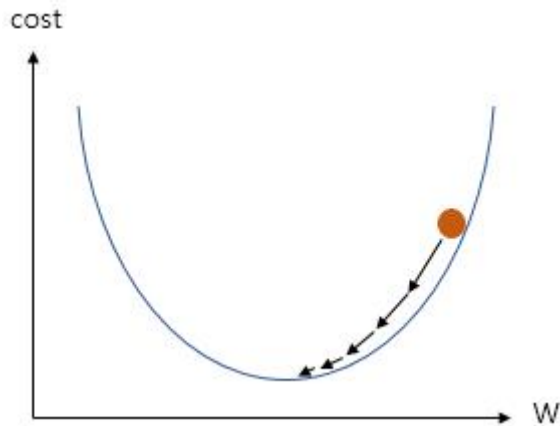
$y=Wx$ 라는 가설 $H(x)$



기울기 W 가 무한대로 커지면 커질수록 cost의 값 또한 무한대로 커지고, 반대로 기

울기 W 가 무한대로 작아져도 cost의 값은 무한대로 커진다.

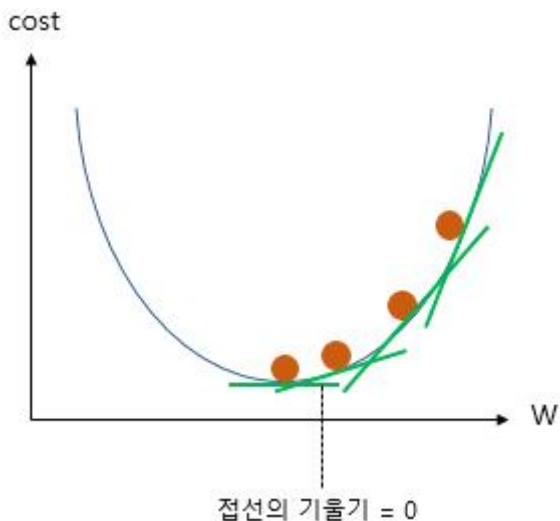
- 그래프에서 cost가 가장 작을 때는 볼록한 부분의 맨 아래 부분이다.
- 기계가 해야 할 일은 cost가 가장 최소값을 가지게 하는 W 를 찾는 일(볼록한 부분의 맨 아래 부분의 W 의 값)



- 임의의 랜덤값 W 값을 정한 뒤에, 맨 아래의 볼록한 부분을 향해 점차 W 의 값을 수정해나간다.

- 위의 그림은 W 값이 점차 수정되는 과정을 보여줌

이를 경사 하강법(Gradient Descent)이라고 함. 경사 하강법은 한 점에서의 순간 변화율 또는 접선에서의 기울기



- 위의 그림에서 초록색 선은 W 가 임의의 값을 가지게 되는 네 가지의 경우에 대해서, 그래프 상으로 접선의 기울기를 보여줌

- 맨 아래의 볼록한 부분으로 갈수록 접선의 기울기가 점차 작아지고, 맨 아래의 볼록한 부분에서는 결국 접선의 기울기가 0이 된다.

- 즉, cost가 최소화 되는 지점은 접선의 기울기가 0이 되는 지점(미분값이 0이 되는 지점)

- 경사 하강법은 비용 함수(Cost function)를 미분하여 현재 W 에서의 접선의 기울기

를 구하고, 접선의 기울기가 낮은 방향으로 W 의 값을 변경하고 다시 미분하고 이 과정을 접선의 기울기가 0인 곳을 향해 W 의 값을 변경하는 작업을 반복하는 것이다.

□ 비용 함수(Cost function)=목적함수

$$cost(W, b) = \frac{1}{n} \sum_{i=1}^n \left[y^{(i)} - H(x^{(i)}) \right]^2$$

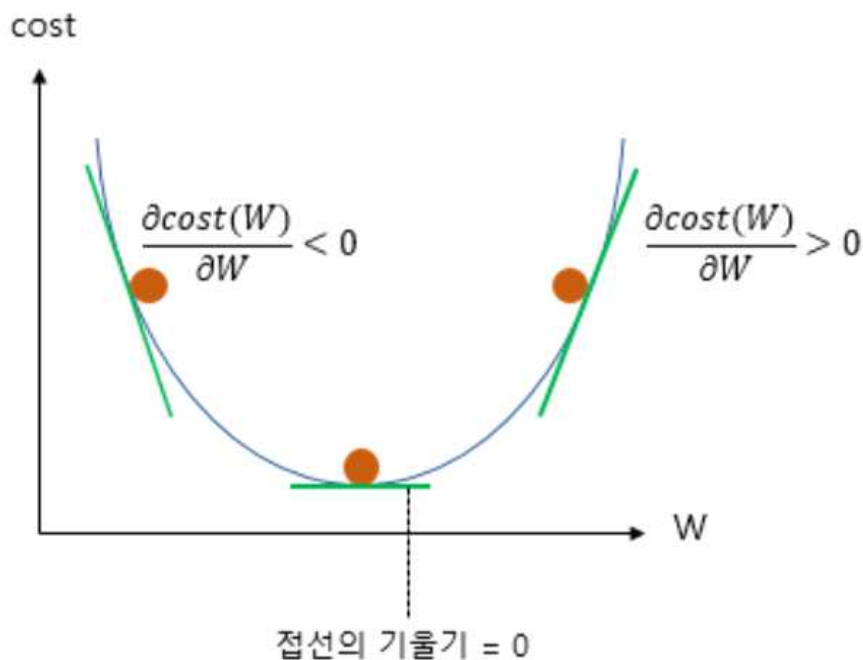
비용(cost)를 최소화하는 W 를 구하기 위해 W 를 업데이트하는 식은 다음과 같고 이를 접선의 기울기가 0이 될 때까지 반복한다.

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

위의 식은 현재 W 에서의 접선의 기울기와 α 와 곱한 값을 현재 W 에서 빼서 새로운 W 의 값으로 한다는 것을 의미한다.

- α 는 여기서 학습률(learning rate).

- 먼저 α 는 생각하지 않고 현재 W 에서 현재 W 에서의 접선의 기울기를 빼는 행위가 어떤 의미가 있는지 알아보자



위의 그림은 접선의 기울기가 음수일 때, 0일때, 양수일 때의 경우를 보여준다.
접선의 기울기가 음수일 때는

$$W := W - \alpha(\text{음수기울기}) = W + \alpha(\text{양수기울기})$$

즉, 기울기가 음수면 W의 값이 증가하게 되는데 이는 결과적으로 접선의 기울기가 0인 방향으로 W의 값이 조정된다.

접선의 기울기가 양수라면 위의 수식은 아래와 같이 표현

$$W := W - \alpha(\text{양수기울기})$$

기울기가 양수면 W의 값이 감소하게 되는데 이는 결과적으로 기울기가 0인 방향으로 W의 값이 조정된다.

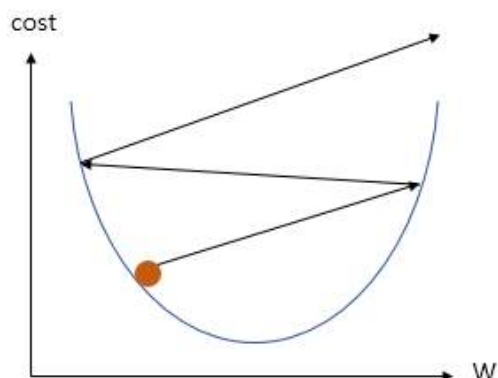
결국, 아래의 수식은 접선의 기울기가 음수거나, 양수일 때 모두 접선의 기울기가 0인 방향으로 W의 값을 조정한다.

$$W := W - \alpha \frac{\partial}{\partial W} \text{cost}(W)$$

- 학습률(learning rate) α 는 어떤 의미?

학습률 α 은 W의 값을 변경할 때, 얼마나 크게 변경할지를 결정한다. 또는 W를 그래프의 한 점으로 보고 접선의 기울기가 0일 때까지 경사를 따라 얼마나 큰 폭으로 이동할지 결정한다.

- 학습률 α 의 값을 무작정 크게 하면 접선의 기울기가 최소값이 되는 W를 빠르게 찾을 수 있을 것 같지만 그렇지 않다.



- 위의 그림은 학습률 α 가 지나치게 높은 값을 가질 때, 접선의 기울기가 0이 되는 W를 찾아가는 것이 아니라 W의 값이 발산하는 상황을 보여준다.

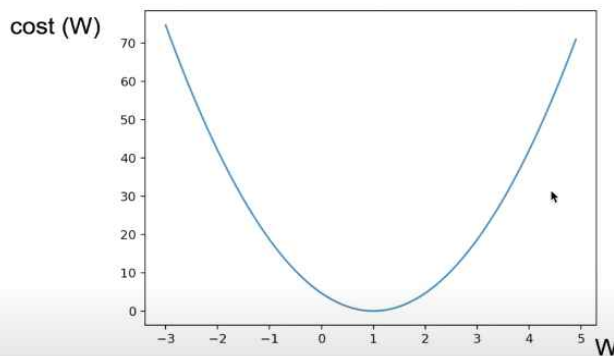
- 반대로 학습률 α 가 지나치게 낮은 값을 가지면 학습 속도가 느려지므로

적당한 α 의 값을 찾아내는 것이 중요함

- 실제 경사 하강법은 W 와 b 에 대해서 **동시에** 경사 하강법을 수행하면서 최적의 W 와 b 의 값을 찾아간다.

- 풀고자 하는 각 문제에 따라 가설, 비용 함수, 옵티마이저는 전부 다를 수 있으며 선형 회귀에 가장 적합한 비용 함수와 옵티마이저가 알려져있는데 MSE와 경사 하강법이 각각 이에 해당됩니다.

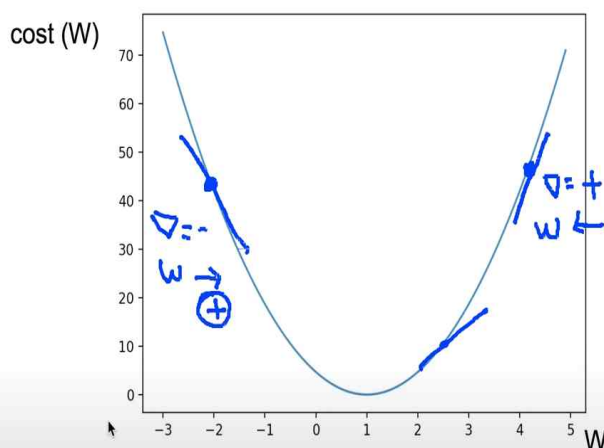
Gradient descent



$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

Gradient descent



$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

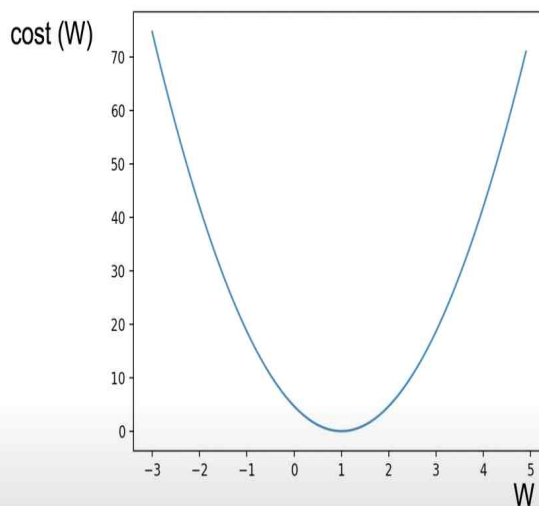
※ 오른쪽 그래프의 경우

기울기는 양수(+)방향, 가중치(weight)는 음수(-)방향

왼쪽 그래프의 경우

기울기는 양수(-)방향, 가중치(weight)는 음수(+)방향

Gradient descent



$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

```
# Minimize: Gradient Descent using derivative:  
W -= learning_rate * derivative  
learning_rate = 0.1  
gradient = tf.reduce_mean((W * X - Y) * X)  
descent = W - learning_rate * gradient  
update = W.assign(descent)
```

$$cost(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

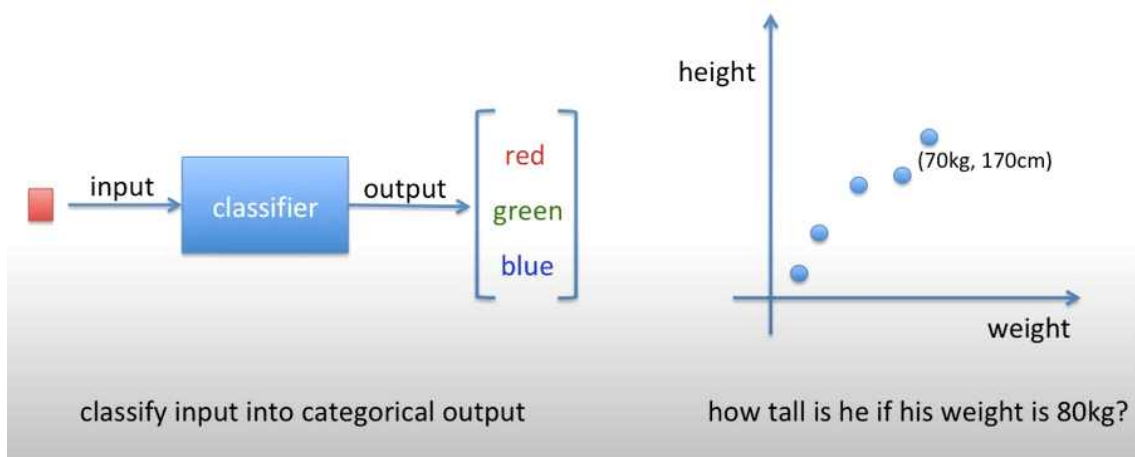
참조;

https://www.youtube.com/watch?v=TxIVr-nk1so&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=6

[Linear Regression(선형회귀)]

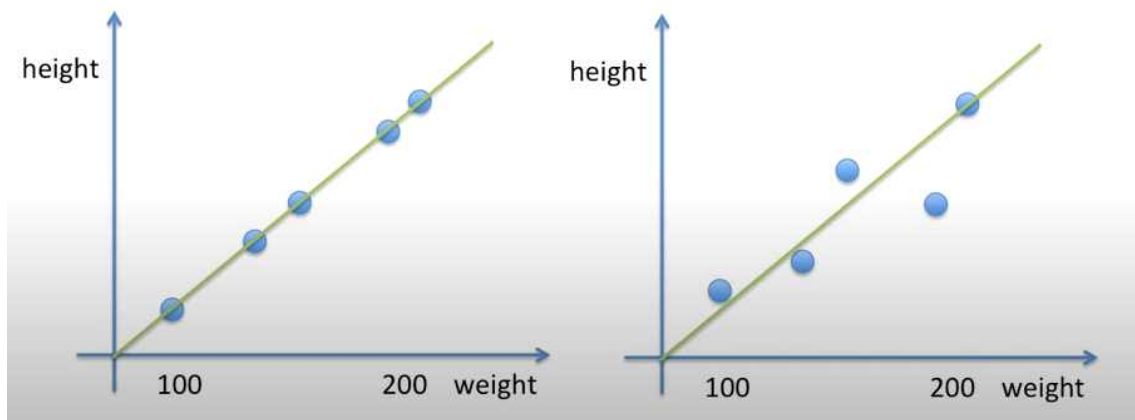
참조사이트: <https://www.youtube.com/watch?v=MwadQ74iE-k>

Classification VS Regression

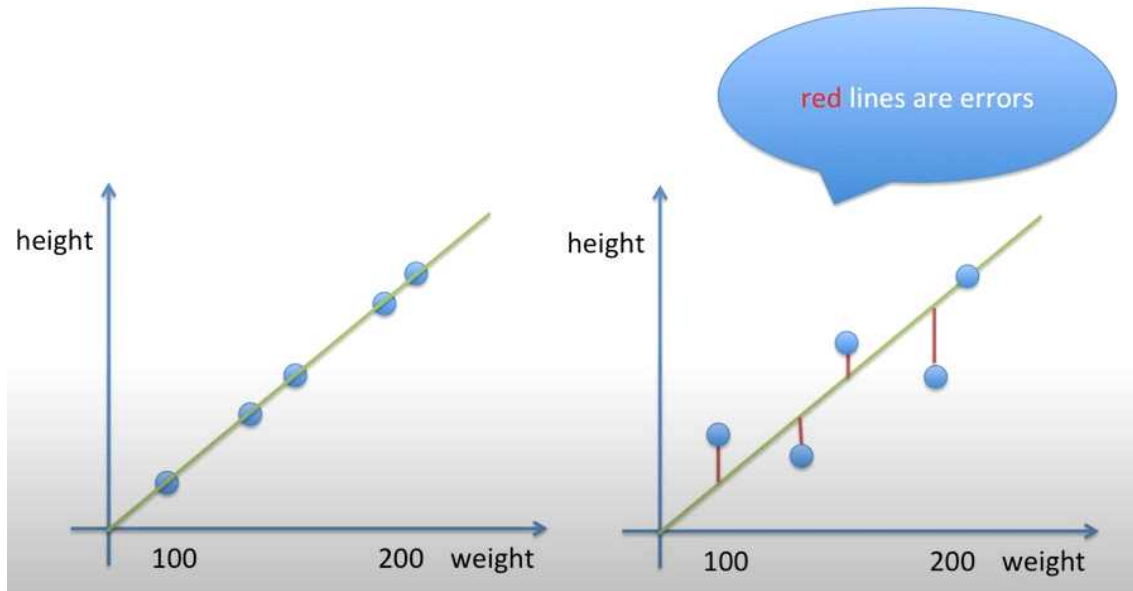


※ categorical: 범주에 속하는

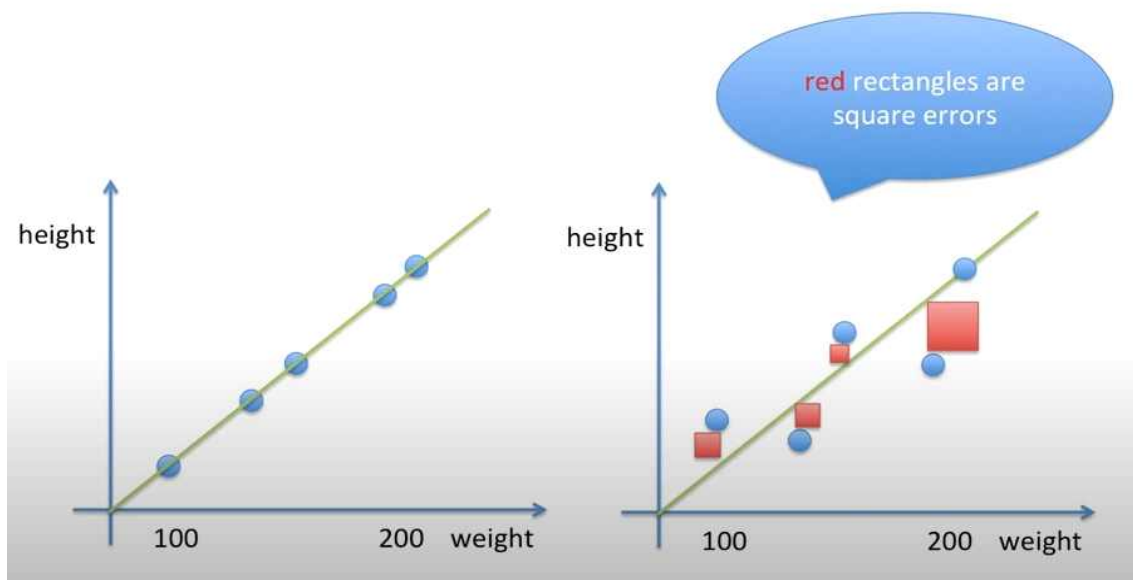
Which line will predict more likely to true height?



Error – difference between prediction and real value

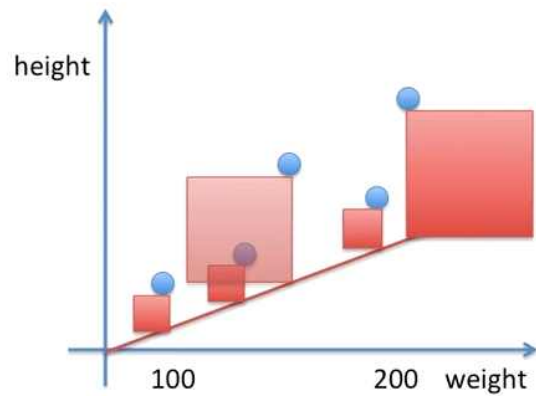
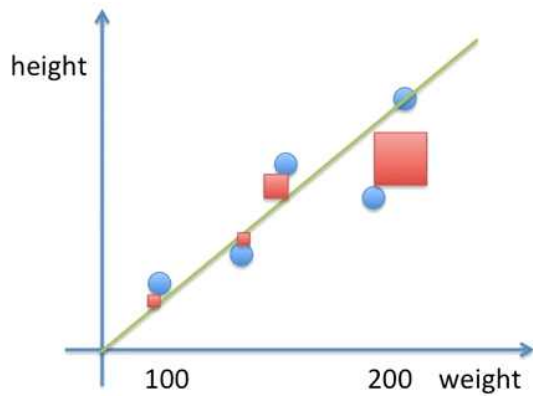


Square Error– (difference between prediction and real value)²



※ square; 정사각형, rectangles; 직사각형
square error; 에러에 제곱한 값(error X2)

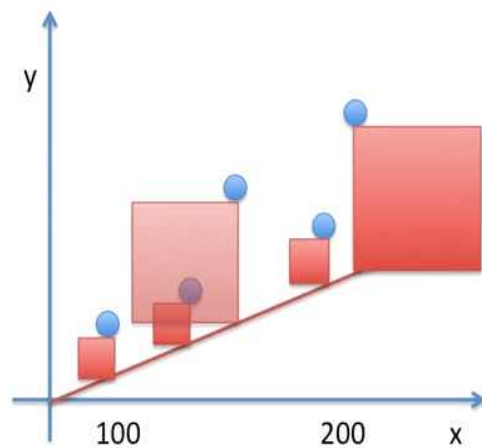
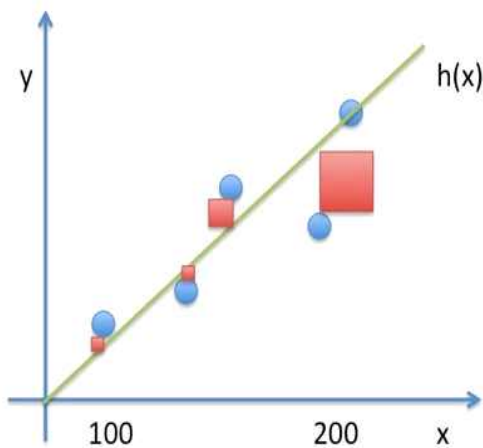
Mean Square Error



green line has less error, therefore prediction will be more likely to be true value (height)

Find linear equation has Least Mean Square (LMS) Error

$$\begin{aligned}\text{Error} &= h(x) - y \\ \text{Square Error} &= (h(x) - y)^2 \\ \text{Mean Square Error} &= 1/n \sum (h(x) - y)^2\end{aligned}$$

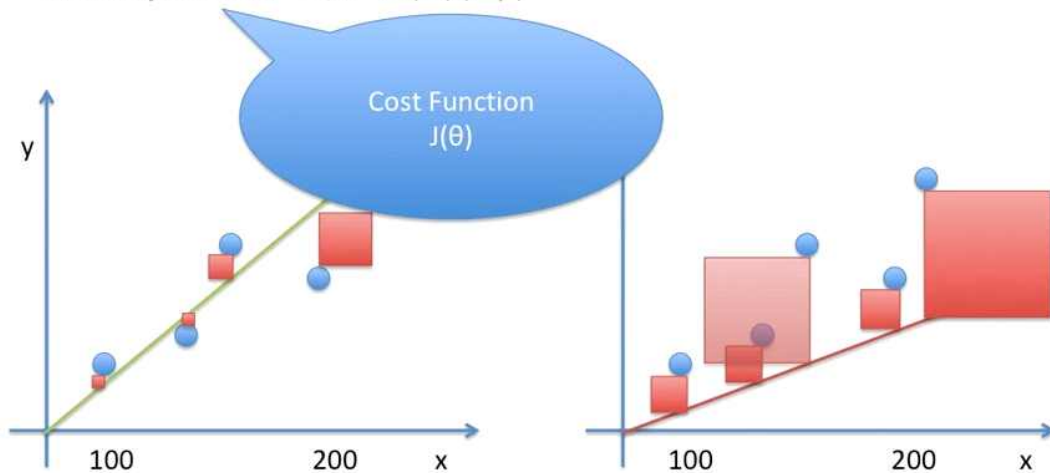


Find linear equation has Least Mean Square (LMS) Error

$$\text{Error} = h(x) - y$$

$$\text{Square Error} = (h(x) - y)^2$$

$$\text{Mean Square Error} = 1/n * \sum (h(x) - y)^2$$



※ Mean Square Error=Cost Function(비용함수; 실제값과 가설과의 차이)

“우리는 Cost Function을 구하는 것이 목적(목적함수)”

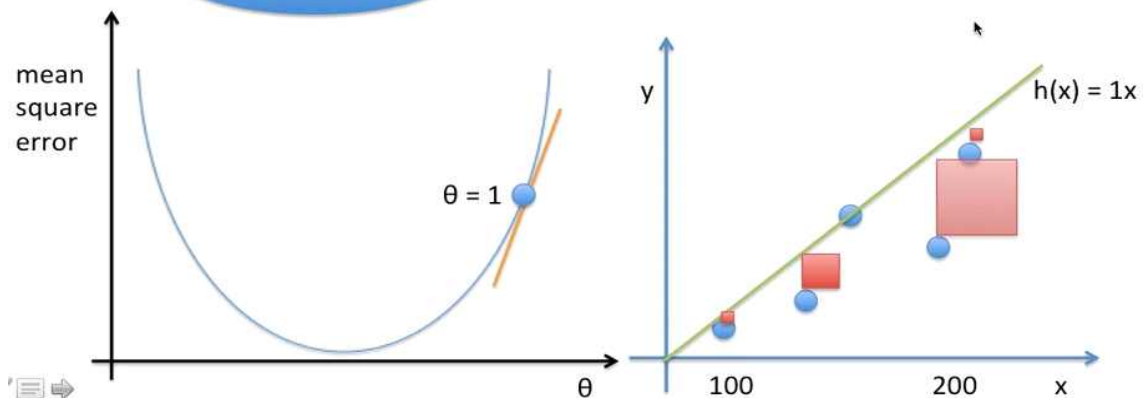
Gradient Decent to choose θ in order to minimize cost function

$$h(x) = \theta x$$

we assign
initial $\theta = 1$

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} \text{ (Mean Square Error)}$$

we assign
initial $\alpha = 0.01$



※ assign; 양수

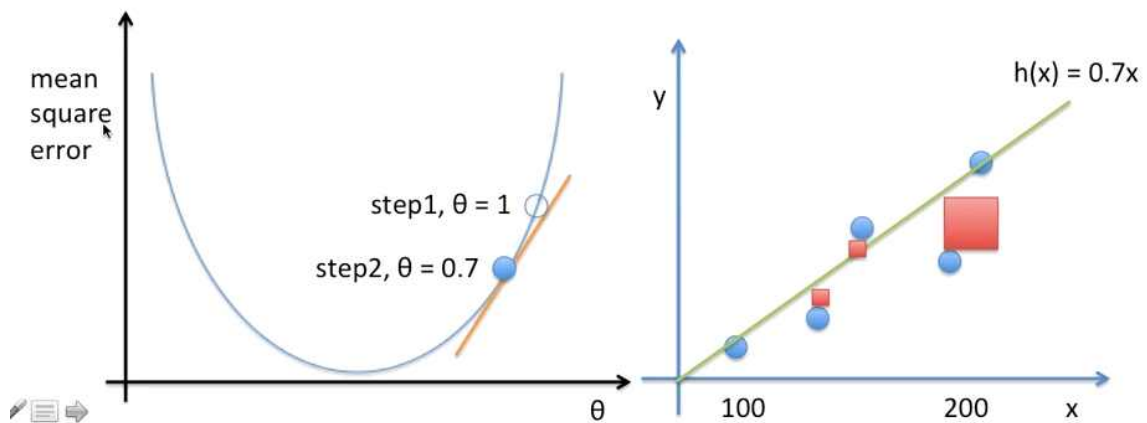
Gradient Decent to choose θ in order to minimize cost function

$$h(x) = \theta x$$

Repeat until converge:

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} \text{ (Mean Square Error)}$$

converge means
gradient is near zero

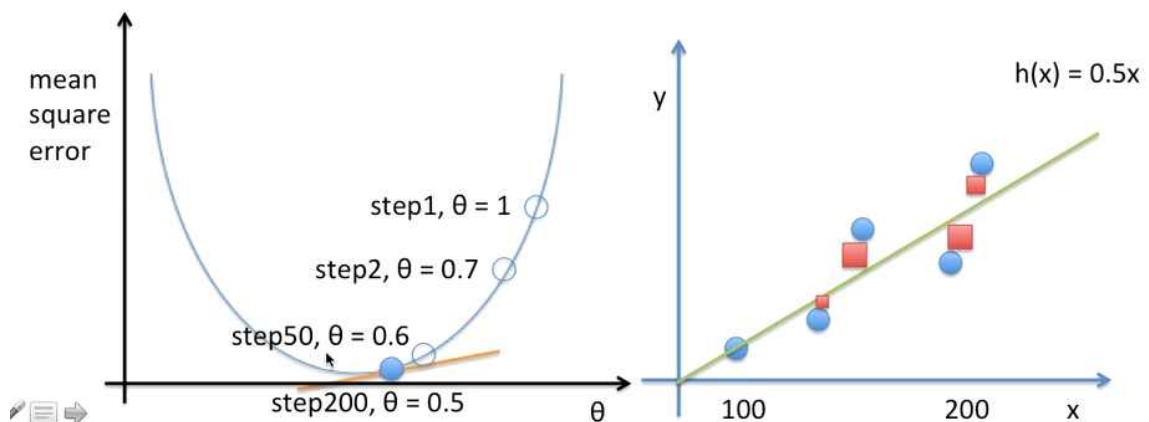


Gradient Decent to choose θ in order to minimize cost function

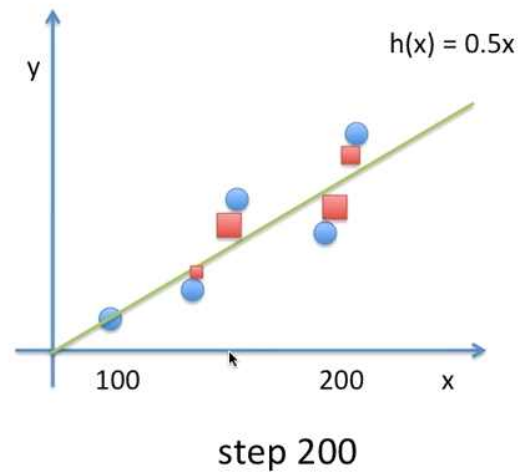
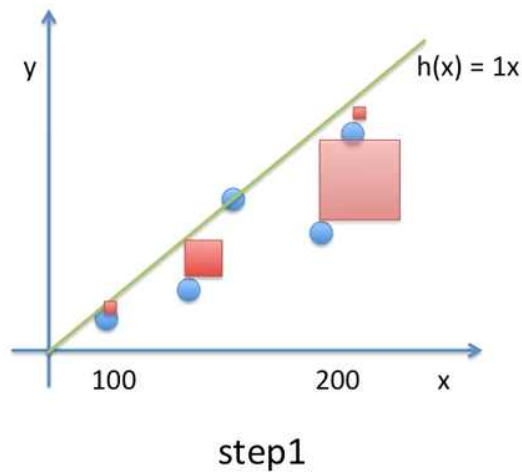
$$h(x) = \theta x$$

Repeat until converge:

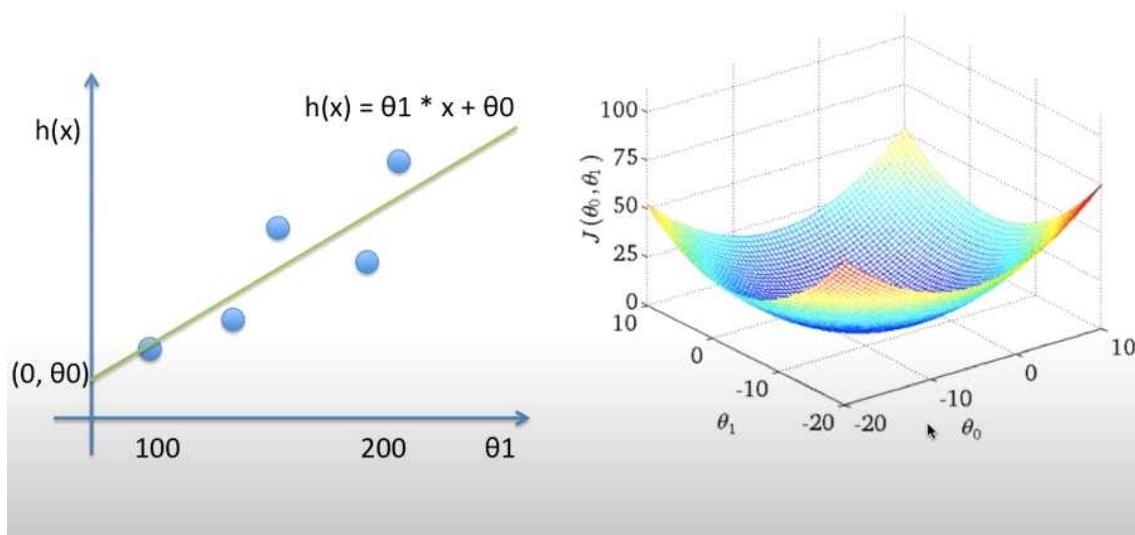
$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} \text{ (Mean Square Error)}$$



compare step1 and step 200



Q&A – What if we have more θ ? $h(x) = \theta_1 * x + \theta_0$

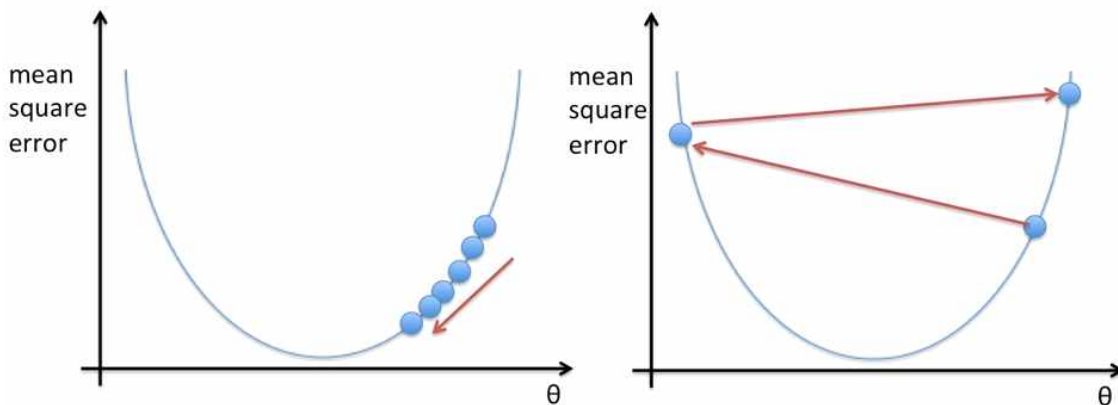


Q&A – How to decide learning rate?

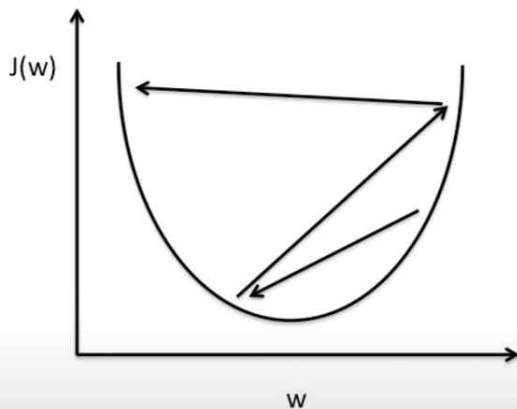
We must set the learning rate to an appropriate value since learning rate determines how fast or slow we will move towards the optimal θ

If learning rate is very large, we may skip the optimal solution.

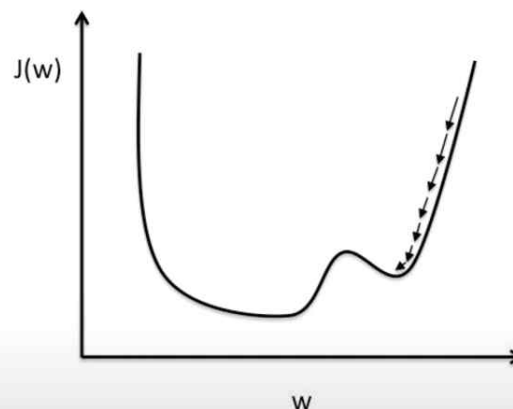
If learning rate is too small, finding optimal solution will take huge steps.



Learning rate: NaN!



Large learning rate: Overshooting.



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

Recap

- Hypothesis

$$H(x) = Wx + b$$

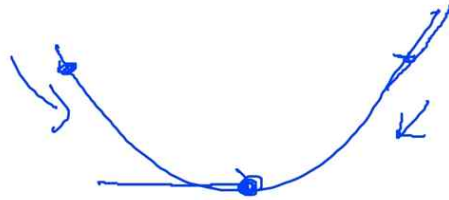
Handwritten notes: A blue arrow points from 'Hypothesis' to $H(x)$. Another blue arrow points from x to a circled '학습' (learning) label.

- Cost function

$$\text{cost}(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

Handwritten notes: A blue arrow points from 'prediction' to $H(x^{(i)})$. Another blue arrow points from 'true' to $y^{(i)}$. There are also blue arrows pointing to the W and b parameters in the cost function.

- Gradient descent algorithm



Linear Regression

- 기본적인 트레이닝 세트(데이터 세트)가 $x=1, y=1$...과 같이 있을 경우, 이 데이터들의 그래프를 그려보면 $x=y$ 꼴 그래프가 될 것이다.
- 이때, $x=y$ 그래프가 정답이라고 가정을 하고! 이 선과 가장 가까운 일직선 라인(Linear)과 가까운 좌표값을 정답이라고 처리하는 학습 모델 방법을 말한다.

$$\text{Hypothesis : } H(x) = Wx + b$$

- Linear Regression 학습방법 기본이 되는 저 방정식 따라서 각각의 값들이 얼마나 $x=y$ 꼴 그래프와 가장 가까이 있는지를 구하는 식을 알면, Linear Regression 학습법을 ML 적용시킬 수 있다.

Cost Function = Loss Function

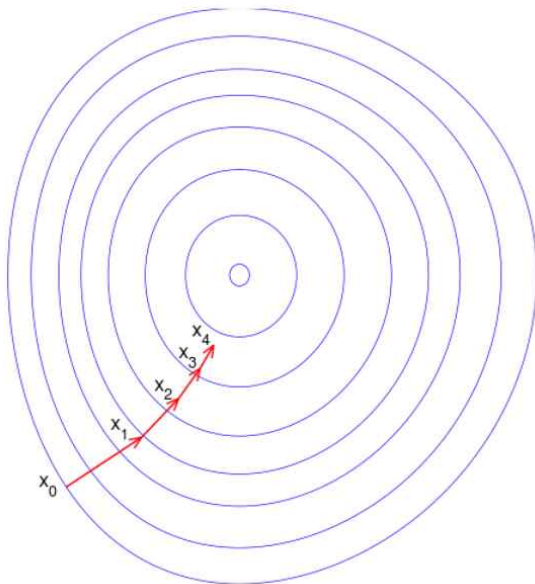
- Linear Regression에서 정답이라고 가정한 선($y=x$ 그래프선)과 가장 가까운 거리의 값을 찾는 방법을 일컫어 Cost Function (=Loss Function) 이라고 한다.
- 즉, 가설과 실제 데이터의 차이값을 볼 수 있는 방법을 말한다.
- $H(x) - y$ 의 값(정답 값 $H(x)$ 에서 실제 값인 y 를 뺀 식)을 구하자. 하지만, $H(x) - y$ 를 할 경우 -(음수) 값이 나올 수도 있으므로 제곱을 해서 구한다. $\Rightarrow (H(x)-y)^2$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

- 즉, minimize cost (W, b)를 구하는 것이 Linear Regression의 학습이 된다.

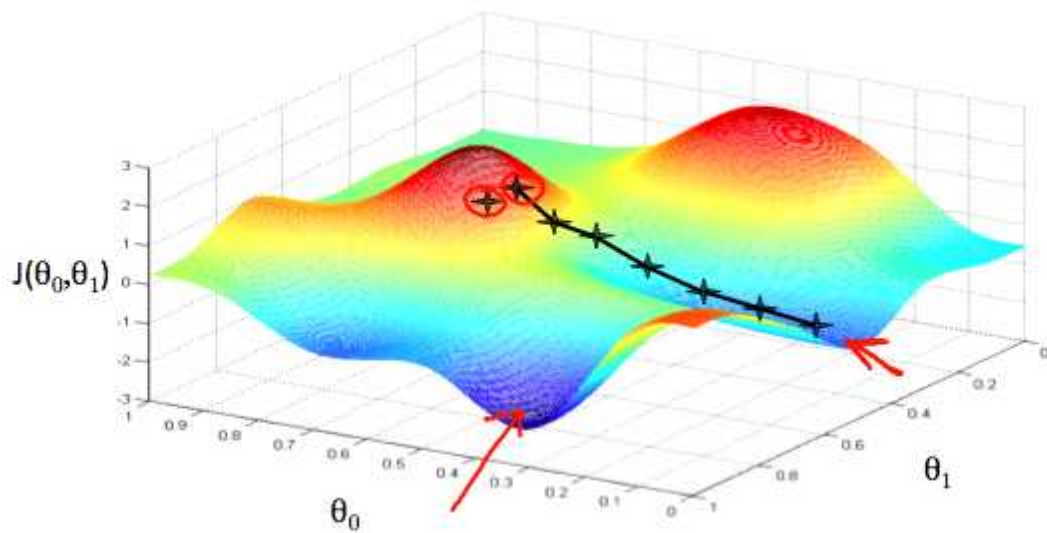
Gradient descent

Gradient descent 방법은 미분의 개념을 최적화 문제에 적용한 대표적 방법 중 하나로서 함수의 local minimum을 찾는 방법 중 하나.



[gradient descent 방법 (그림출처: 위키피디아)]

다음은 θ_0 을 x축, θ_1 을 y축, $J(\theta_0, \theta_1)$ 을 z축으로 하는 그래프의 예이다.



이 그래프에서 가장 아래있는 점의 x, y 좌표 즉 θ_0 와, θ_1 을 알아내는 것이 좋은 hypothesis function 만들기의 목표이다.


이 최소값은 cost function의 미분을 통해서 구할 수 있는데, 어떤 점에서 미분을 해서 나온 기울기 값이 최소값으로 향하는 방향을 제시한다는 것이다.

이런 방법으로 θ 를 구하는 게 gradient descent 의 algorithm이다.

한편 이렇게 미분을 통해 나온 기울기 값으로 점점 최소값으로 향할 때 얼마만큼 이동할지는 learning rate인 상수값으로 설정하는데 보통 learning rate는 α 라고 쓴다.

※ 미분계산기 사이트

https://www.derivative-calculator.net/



Derivative Calculator

Calculate derivatives online – with steps and graphing!

Also check the [Integral Calculator!](#)

[Calculadora de Derivadas en español](#)
[Ableitungsrechner auf Deutsch](#)

Advertisement

MARRIOTT BONVOY

한국 메리어트 호텔 프로모션 및 패키지를 경험하세요. 자세히 보기

Calculate the Derivative of ...

sin(sqrt(e*x + a) / 2) Go!

CLR + - * / ^ √ ()

This will be calculated:

$$\frac{d}{dx} \left[\sin \left(\frac{\sqrt{e^x + a}}{2} \right) \right]$$

Not what you mean? Use parentheses! Set differentiation variable and order in "Options".

Advertisement

MARRIOTT BONVOY

MARRIOTT.CO.KR 에서 최저가 보장 혜택을 만나 보세요. 자세히 보기

About Help Examples Options Practice

The Derivative Calculator lets you calculate derivatives of functions online – for free!

Our calculator allows you to check your solutions to calculus exercises. It helps you practice by showing you the full working (step by step differentiation).

The Derivative Calculator supports computing first, second, ..., fifth derivatives as well as differentiating functions with many variables (partial derivatives), implicit differentiation and calculating roots/zeros. You can also check your answers! Interactive graphs/plots help visualize and better understand the functions.

For more about how to use the Derivative Calculator, go to "Help" or take a look at the examples.

And now: Happy differentiating!

미분 Differentiation

{ Difference 차이
 Δx x 1...6 $\Delta x=5$ x 2...7 $\Delta x=5$
 Δy y 2...3 $\Delta y=1$

{ Rate, 비율
 $\frac{\Delta y}{\Delta x} = \frac{\text{변화}}{\text{시간}} = \text{가속도}$ $\frac{\text{거리}}{\text{시간}} = \text{속도}$

$$\frac{v}{v} = ***$$

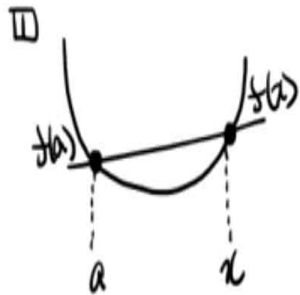
Instantaneous
순간 변화율
① 한점 가속도



Mean
평균 변화율
② 두 점 가속도

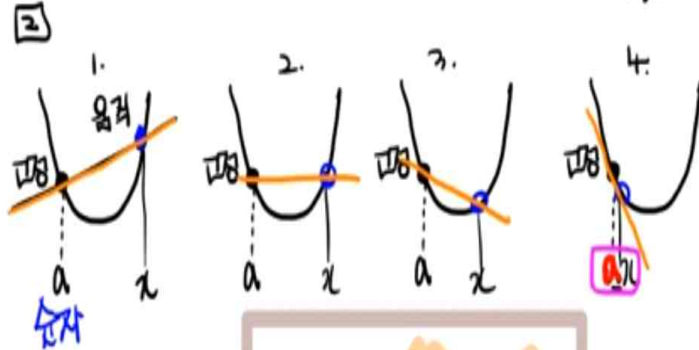


수학적 정의



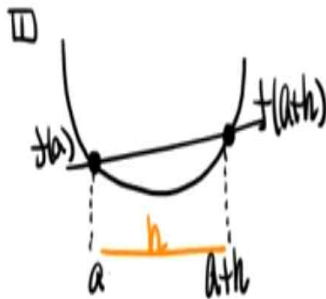
$$\frac{f(x) - f(a)}{x - a}$$

두 점 사이의
시간 변화율

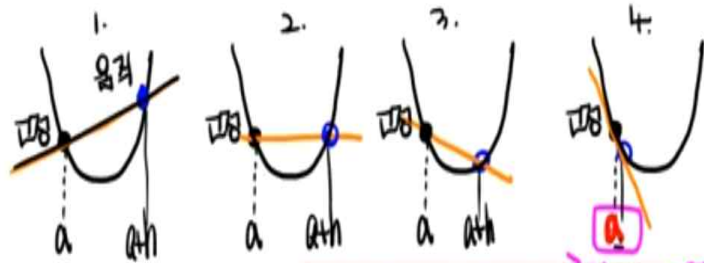


$$\lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a} = f'(a) = \frac{f'(a)}{1} = \text{미분계수}$$

한 점에서의
시간 변화율



$$\frac{f(a+h) - f(a)}{h}$$



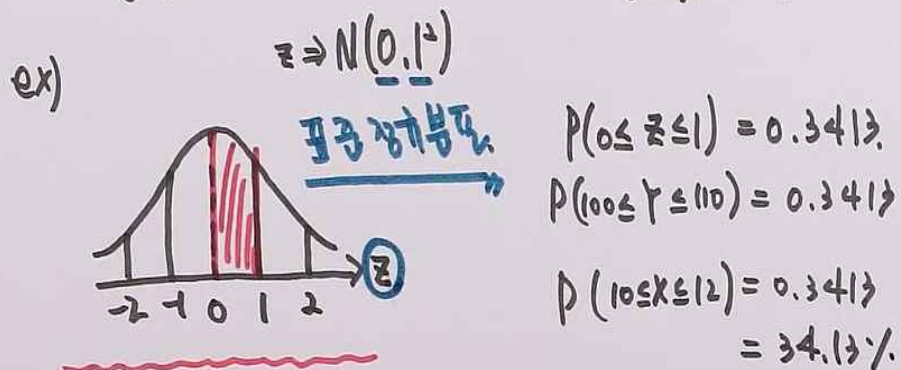
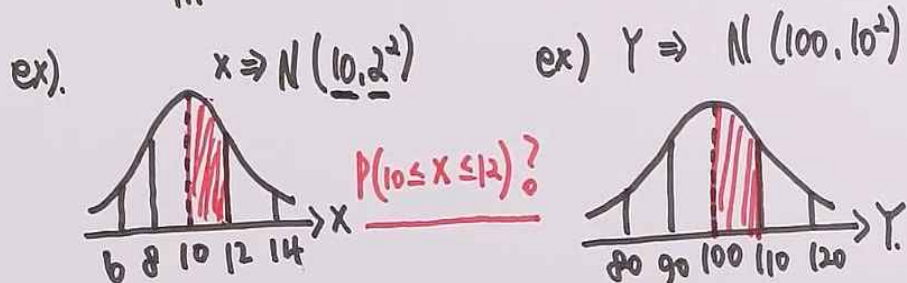
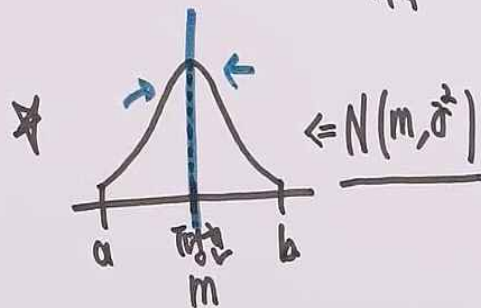
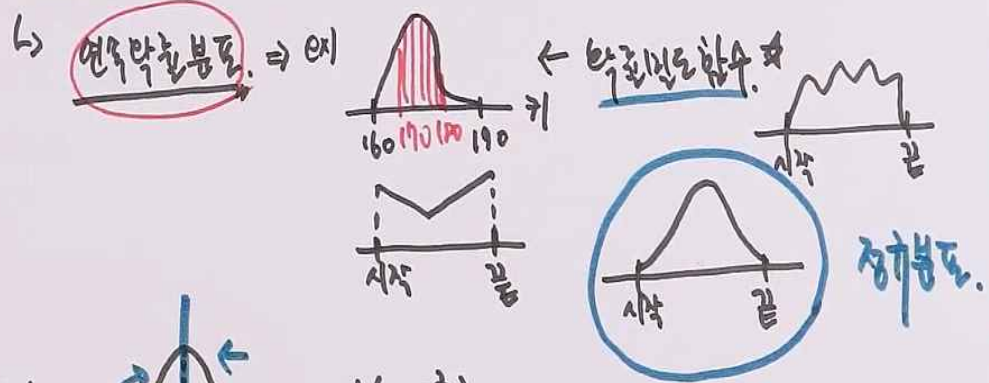
$$\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} = f'(a) = \frac{f'(a)}{1} = \text{미분계수}$$

한 점에서의
시간 변화율

참조: <https://www.youtube.com/watch?v=uLztni4-7B0&t=680s>

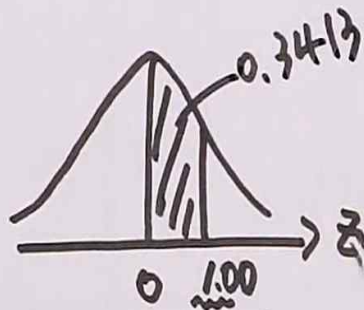
※ 정규분포

◎ 정규분포 / 정규분포의 확률계산

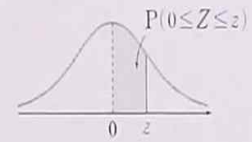


● 표준정규분포표

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0	0.004	0.008	0.012	0.016	0.0199	0.0239	0.0279	0.0319	0.0359
0.1	0.0398	0.0438	0.0478	0.0517	0.0557	0.0596	0.0636	0.0675	0.0714	0.0753
0.2	0.0793	0.0832	0.0871	0.091	0.0948	0.0987	0.1026	0.1064	0.1103	0.1141
0.3	0.1179	0.1217	0.1255	0.1293	0.1331	0.1368	0.1406	0.1443	0.148	0.1517
0.4	0.1554	0.1591	0.1628	0.1664	0.17	0.1736	0.1772	0.1808	0.1844	0.1879
0.5	0.1915	0.195	0.1985	0.2019	0.2054	0.2088	0.2123	0.2157	0.219	0.2224
0.6	0.2257	0.2291	0.2324	0.2357	0.2389	0.2422	0.2454	0.2486	0.2517	0.2549
0.7	0.258	0.2611	0.2642	0.2673	0.2704	0.2734	0.2764	0.2794	0.2823	0.2852
0.8	0.2881	0.291	0.2939	0.2967	0.2995	0.3023	0.3051	0.3078	0.3106	0.3133
0.9	0.3159	0.3186	0.3212	0.3238	0.3264	0.3289	0.3315	0.334	0.3365	0.3389
1	0.3413	0.3438	0.3461	0.3485	0.3508	0.3531	0.3554	0.3577	0.3599	0.3621
1.1	0.3643	0.3665	0.3686	0.3708	0.3729	0.3749	0.377	0.379	0.381	0.383
1.2	0.3849	0.3869	0.3888	0.3907	0.3925	0.3944	0.3962	0.398	0.3997	0.4015
1.3	0.4032	0.4049	0.4066	0.4082	0.4099	0.4115	0.4131	0.4147	0.4162	0.4177
1.4	0.4192	0.4207	0.4222	0.4236	0.4251	0.4265	0.4279	0.4292	0.4306	0.4319
1.5	0.4332	0.4345	0.4357	0.437	0.4382	0.4394	0.4406	0.4418	0.4429	0.4441
1.6	0.4452	0.4463	0.4474	0.4484	0.4495	0.4505	0.4515	0.4525	0.4535	0.4545
1.7	0.4554	0.4564	0.4573	0.4582	0.4591	0.4599	0.4608	0.4616	0.4625	0.4633
1.8	0.4641	0.4649	0.4656	0.4664	0.4671	0.4678	0.4686	0.4693	0.4699	0.4706
1.9	0.4713	0.4719	0.4726	0.4732	0.4738	0.4744	0.475	0.4756	0.4761	0.4767
2	0.4772	0.4778	0.4783	0.4788	0.4793	0.4798	0.4803	0.4808	0.4812	0.4817
2.1	0.4821	0.4826	0.483	0.4834	0.4838	0.4842	0.4846	0.485	0.4854	0.4857
2.2	0.4861	0.4864	0.4868	0.4871	0.4875	0.4878	0.4881	0.4884	0.4887	0.489
2.3	0.4893	0.4896	0.4898	0.4901	0.4904	0.4906	0.4909	0.4911	0.4913	0.4916
2.4	0.4918	0.492	0.4922	0.4925	0.4927	0.4929	0.4931	0.4932	0.4934	0.4936
2.5	0.4938	0.494	0.4941	0.4943	0.4945	0.4946	0.4948	0.4949	0.4951	0.4952
2.6	0.4953	0.4955	0.4956	0.4957	0.4959	0.496	0.4961	0.4962	0.4963	0.4964
2.7	0.4965	0.4966	0.4967	0.4968	0.4969	0.497	0.4971	0.4972	0.4973	0.4974
2.8	0.4974	0.4975	0.4976	0.4977	0.4977	0.4978	0.4979	0.4979	0.498	0.4981
2.9	0.4981	0.4982	0.4982	0.4983	0.4984	0.4984	0.4985	0.4985	0.4986	0.4986
3	0.4987	0.4987	0.4987	0.4988	0.4988	0.4989	0.4989	0.4989	0.499	0.499
3.1	0.499	0.4991	0.4991	0.4991	0.4992	0.4992	0.4992	0.4992	0.4993	0.4993
3.2	0.4993	0.4993	0.4994	0.4994	0.4994	0.4994	0.4994	0.4995	0.4995	0.4995
3.3	0.4995	0.4995	0.4995	0.4996	0.4996	0.4996	0.4996	0.4996	0.4996	0.4997



표준정규
분포표



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	.0000	.0040	.0080	.0120	.0160	.0199	.0239	.0279	.0319	.0359
0.1	.0398	.0438	.0478	.0517	.0557	.0596	.0636	.0675	.0714	.0753
0.2	.0793	.0832	.0871	.0910	.0948	.0987	.1026	.1064	.1103	.1141
0.3	.1179	.1217	.1255	.1293	.1331	.1368	.1406	.1443	.1480	.1517
0.4	.1554	.1591	.1628	.1664	.1700	.1736	.1772	.1808	.1844	.1879
0.5	.1915	.1950	.1985	.2019	.2054	.2088	.2123	.2157	.2190	.2224
0.6	.2257	.2291	.2324	.2357	.2389	.2422	.2454	.2486	.2517	.2549
0.7	.2580	.2611	.2642	.2673	.2704	.2734	.2764	.2794	.2823	.2852
0.8	.2881	.2910	.2939	.2967	.2995	.3023	.3051	.3078	.3106	.3133
0.9	.3159	.3186	.3212	.3238	.3264	.3289	.3315	.3340	.3365	.3389
1.0	.3413	.3438	.3461	.3485	.3508	.3531	.3554	.3577	.3599	.3621
1.1	.3643	.3665	.3686	.3708	.3729	.3749	.3770	.3790	.3810	.3830

참조: <https://www.youtube.com/watch?v=cUDr0BydCGg>