MACHINE LEARNING

2018-07-05

REVLR

TEAM S.C.P

CONTENTS

- What is ML : Machine Learning
- Linear Regression – 선형 회귀
- function H(x) (Hypothesis function) – 가설 함수
- cost function – 비용 함수

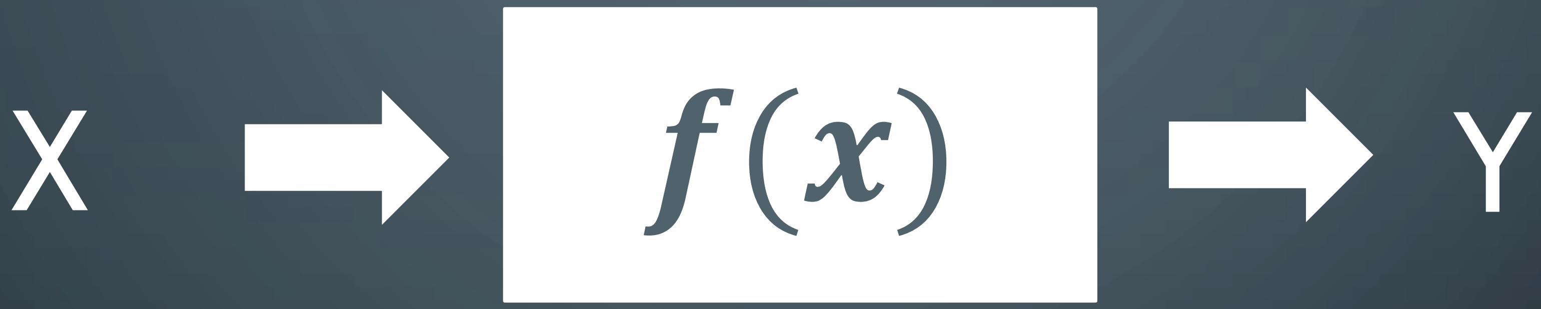
BASIC PREPERATION

- python 3.6.x (install tensorflow)

- math (differentiation)

$$f'(x)$$

EXPLICIT PROGRAMMING



X

$f(x)$

Y

ex) Starcraft 의 computer
넥서스 한대 치고 튀면
프로브 다따라옴

ML : MACHINE LEARNING

컴퓨터를 인간처럼 학습시켜 **스스로 규칙을 형성**할 수 있지 않을까 하는 시도에서 비롯되어 만들어짐

넥서스 때림 → 프로브 다 따라감 → 짐 : 학습

넥서스 때림 → 안 따라감 → 이김 or 짐

SUPERVISED/UNSUPERVISED LEARNING

- Supervised Learning:
    - learning with labeled examples
- Unsupervised Learning: un-labeled data

TYPE OF SUPERVISED LEARNING

- Predicting Large range of value based on time spent

  - regression

- Predicting 1 or 0 value based on time spent

  - binary classification

- Predicting layers based on time spent

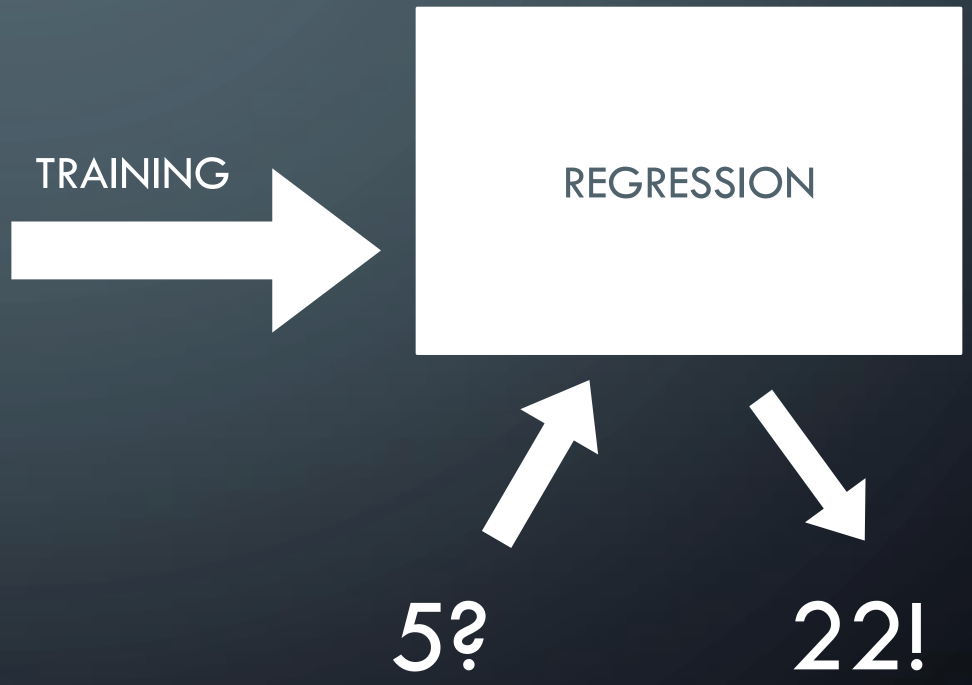
  - multi-label classification

REGRESSION DATA

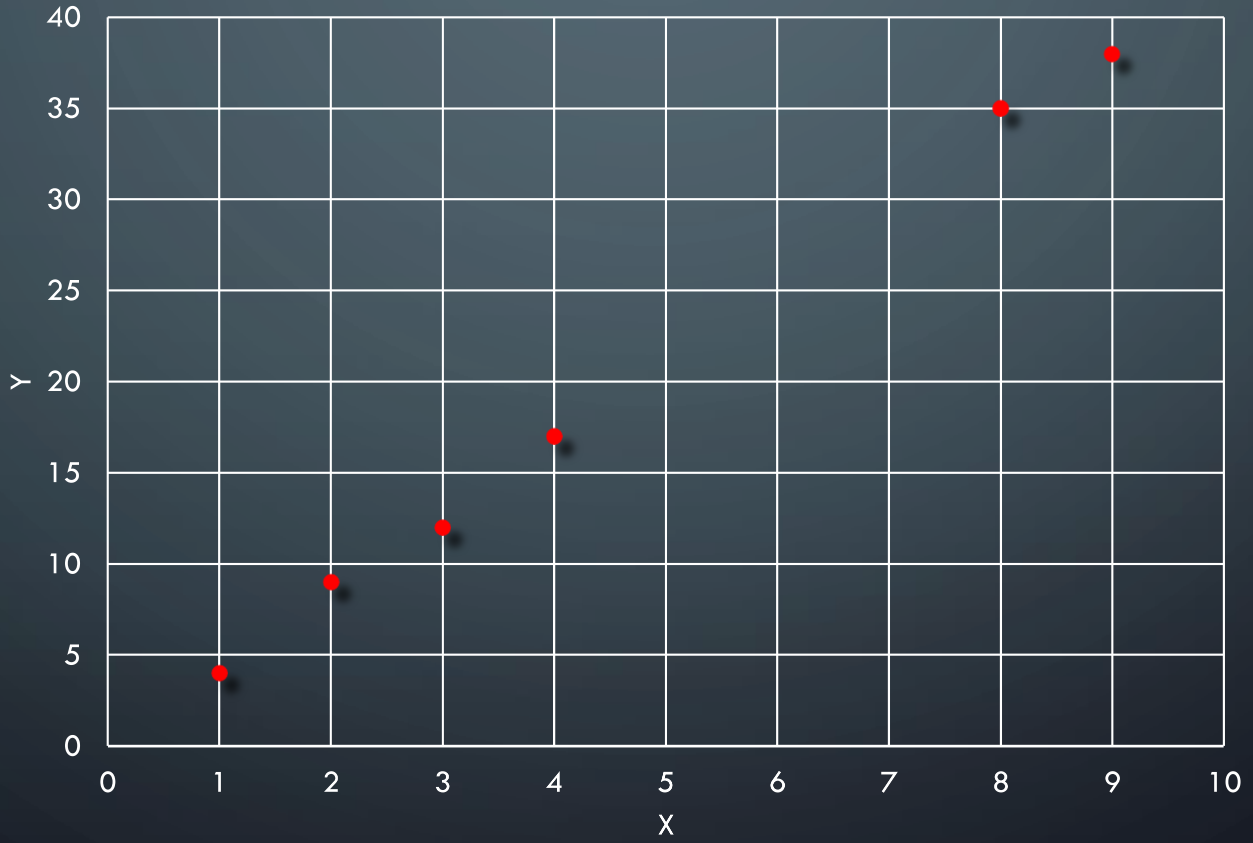
X(play times)	Y(wins)
1	4
2	9
3	12
4	17
8	35
9	38

REGRESSION MODEL

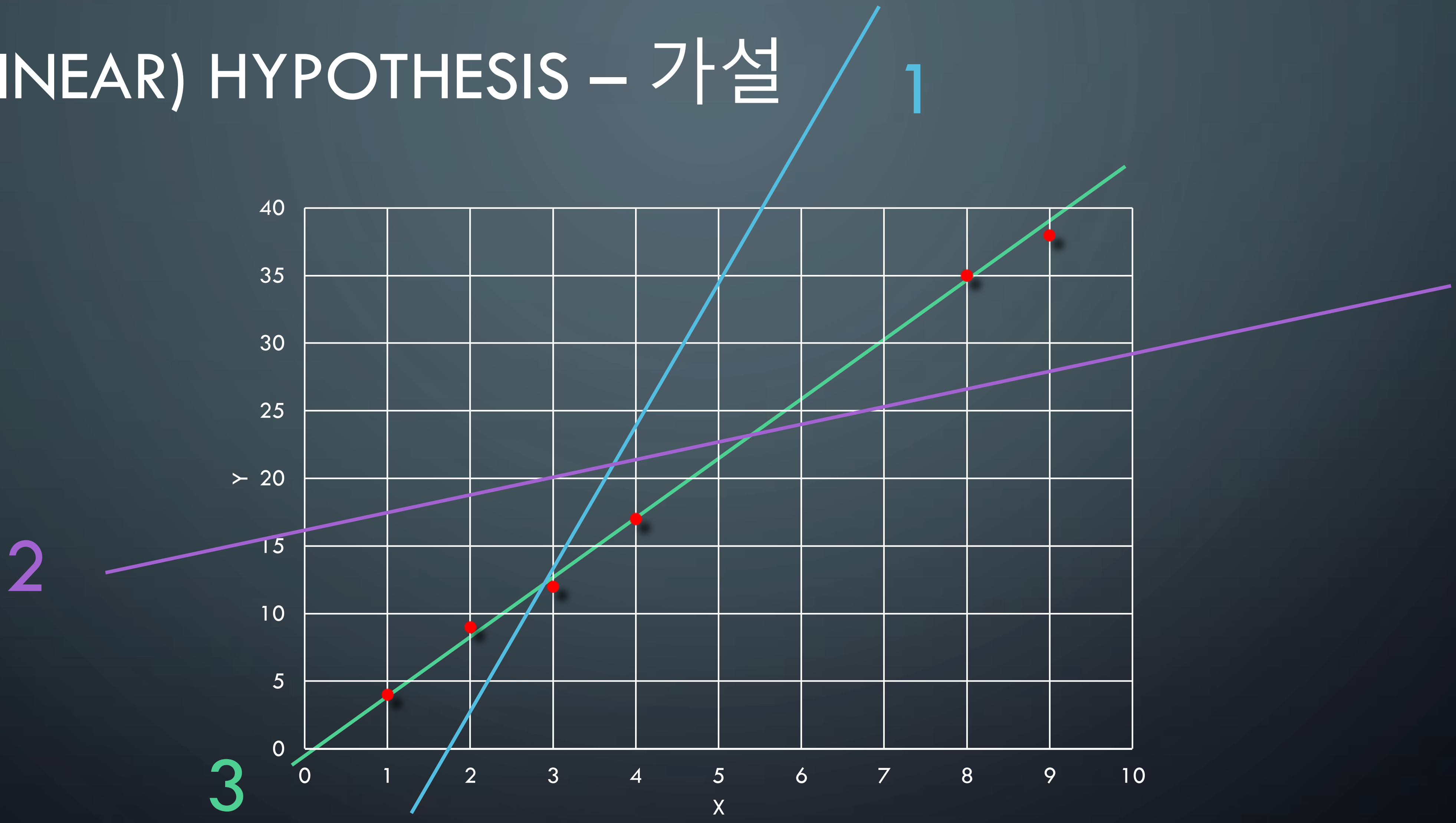
X(play times)	Y(wins)
1	4
2	9
3	12
4	17
8	35
9	38



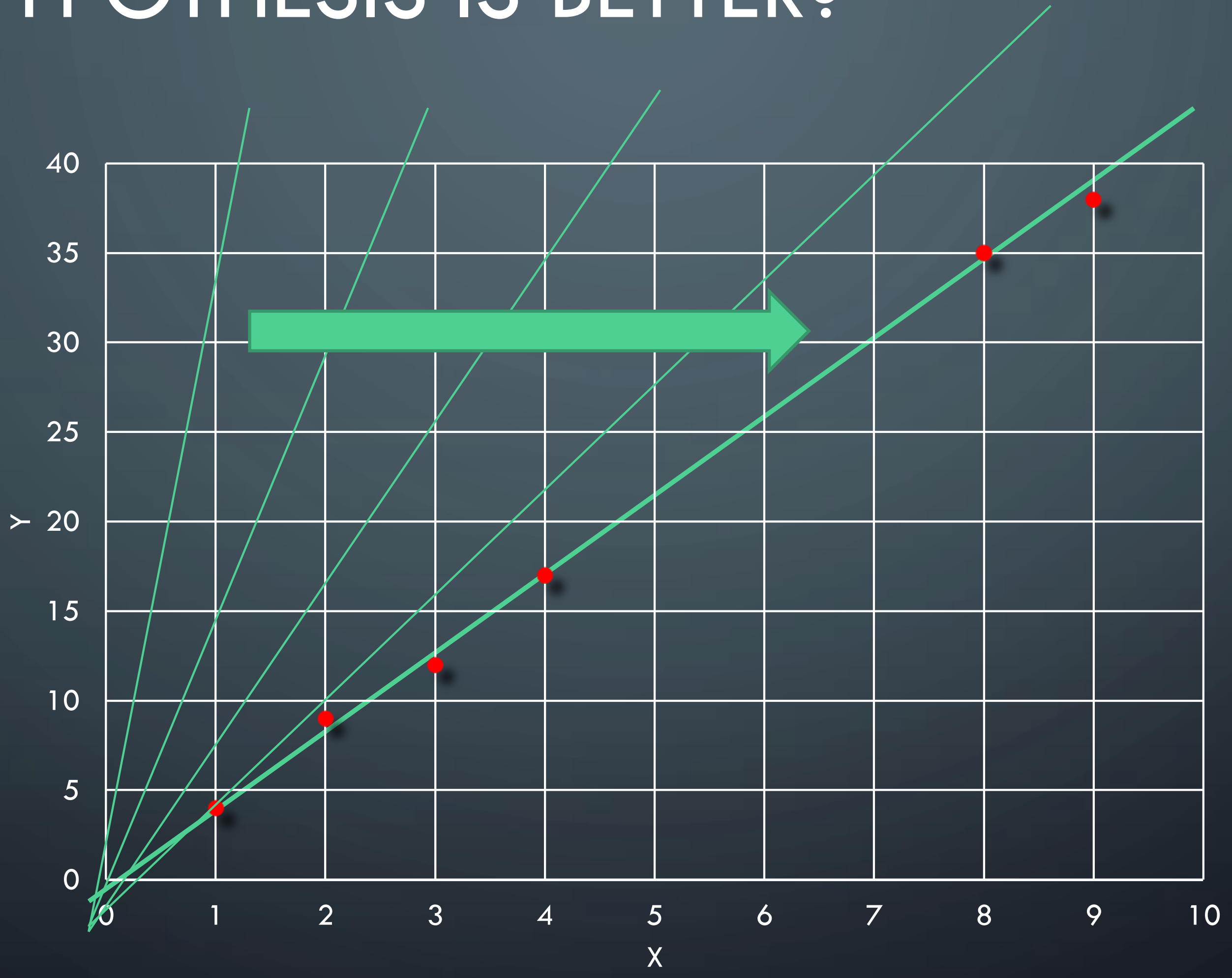
(LINEAR) HYPOTHESIS – 가설



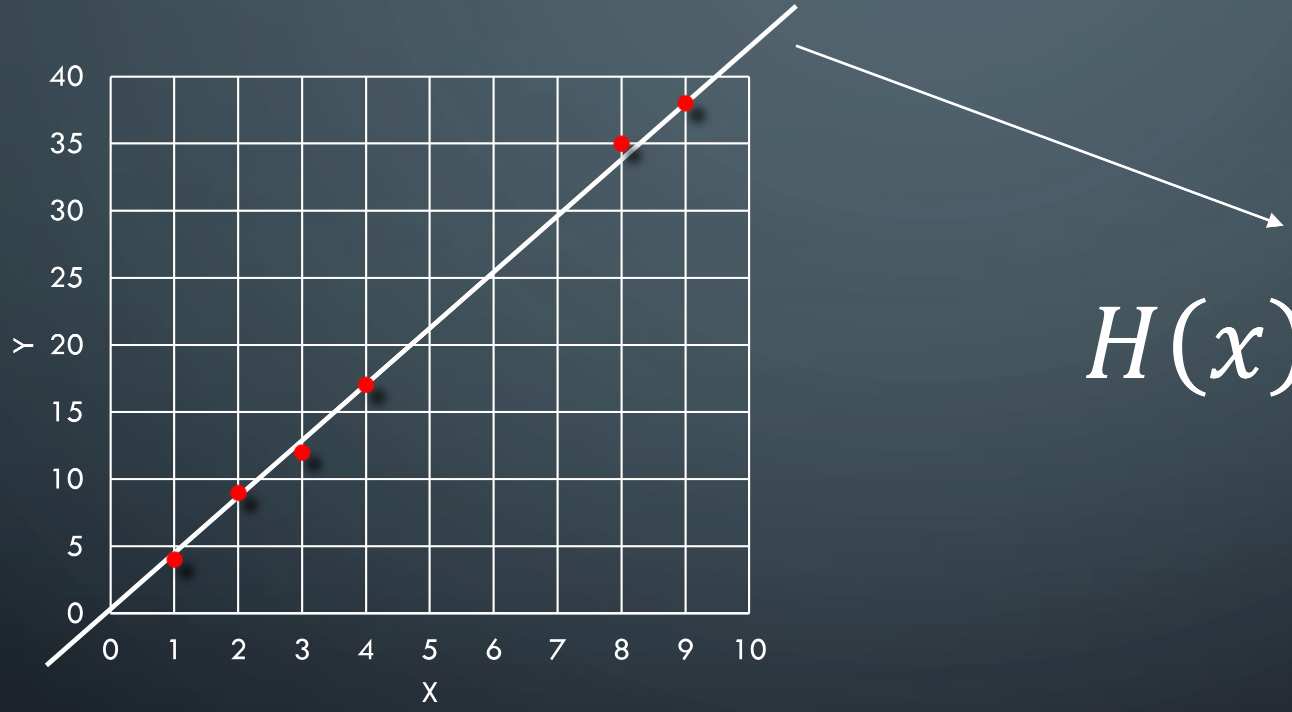
(LINEAR) HYPOTHESIS – 가설



WHICH HYPOTHESIS IS BETTER?

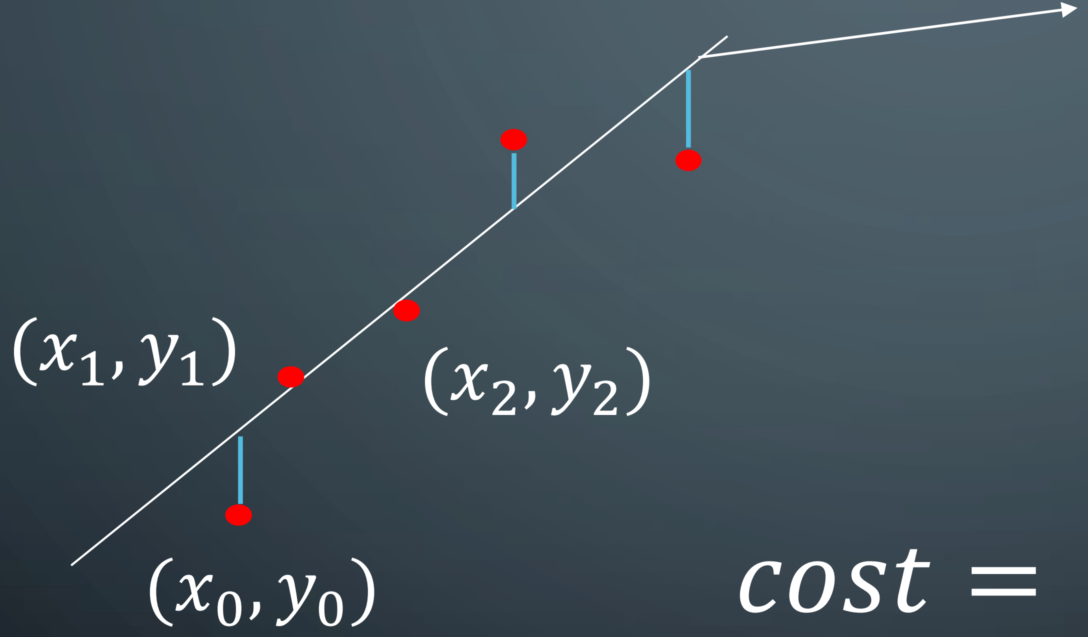


WHICH HYPOTHESIS IS BETTER?



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

COST FUNCTION



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

$$H(x_i) - y_i$$

$$cost = \frac{1}{m}\sum_{i=1}^{m}\left(H(x_i) - y_i\right)^2$$

COST FUNCTION

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

이차함수!

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

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i + b - y_i)^{2}$$

COST FUNCTION

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x_i) - y_i)^{\ 2}$$ W에 대해 미분!

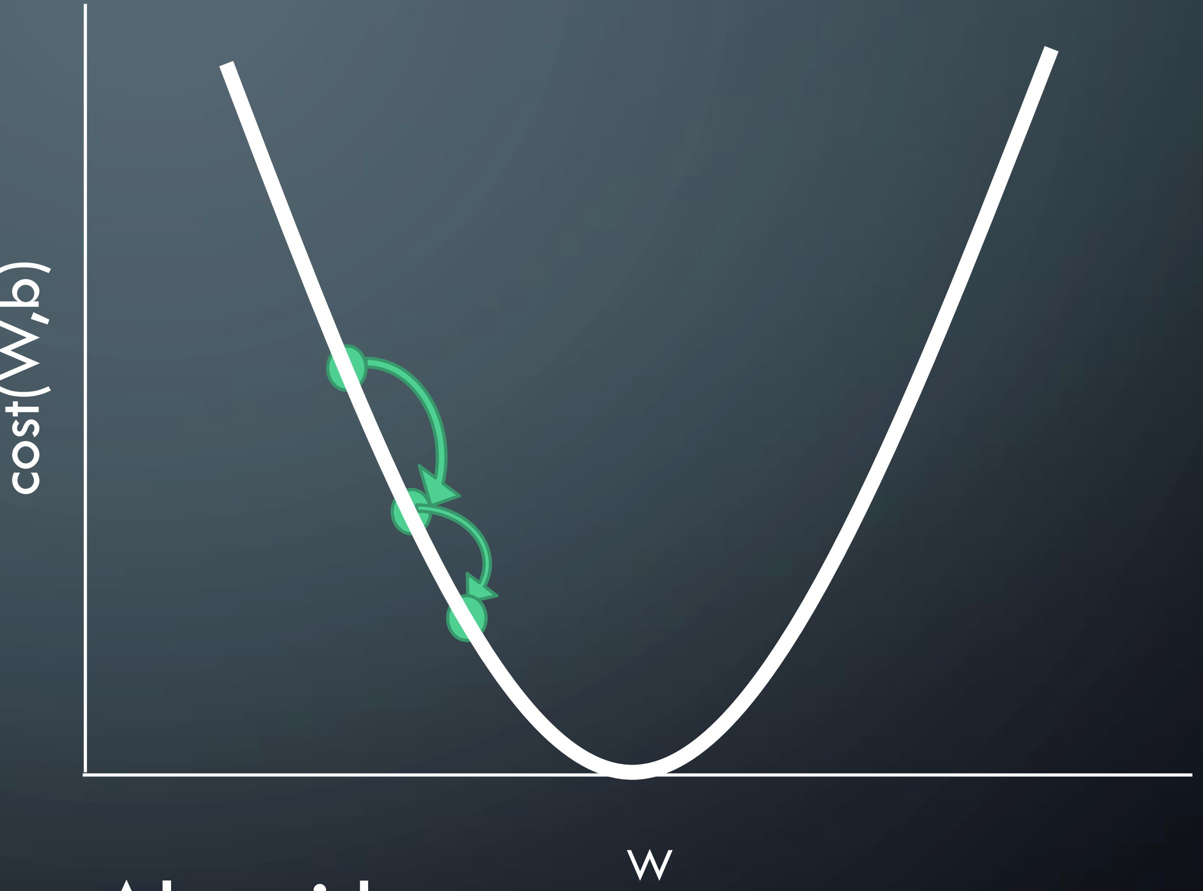
$$\frac{\partial}{\partial W}cost(W,b) = \frac{2}{m}\sum_{i=1}^{m}(x_iW + b - y_i)x_i$$

$$\therefore W \Leftarrow W \ - \ \alpha\frac{\partial}{\partial W}cost(W,b)$$ (α는 0.1 정도의 수)

COST FUNCTION

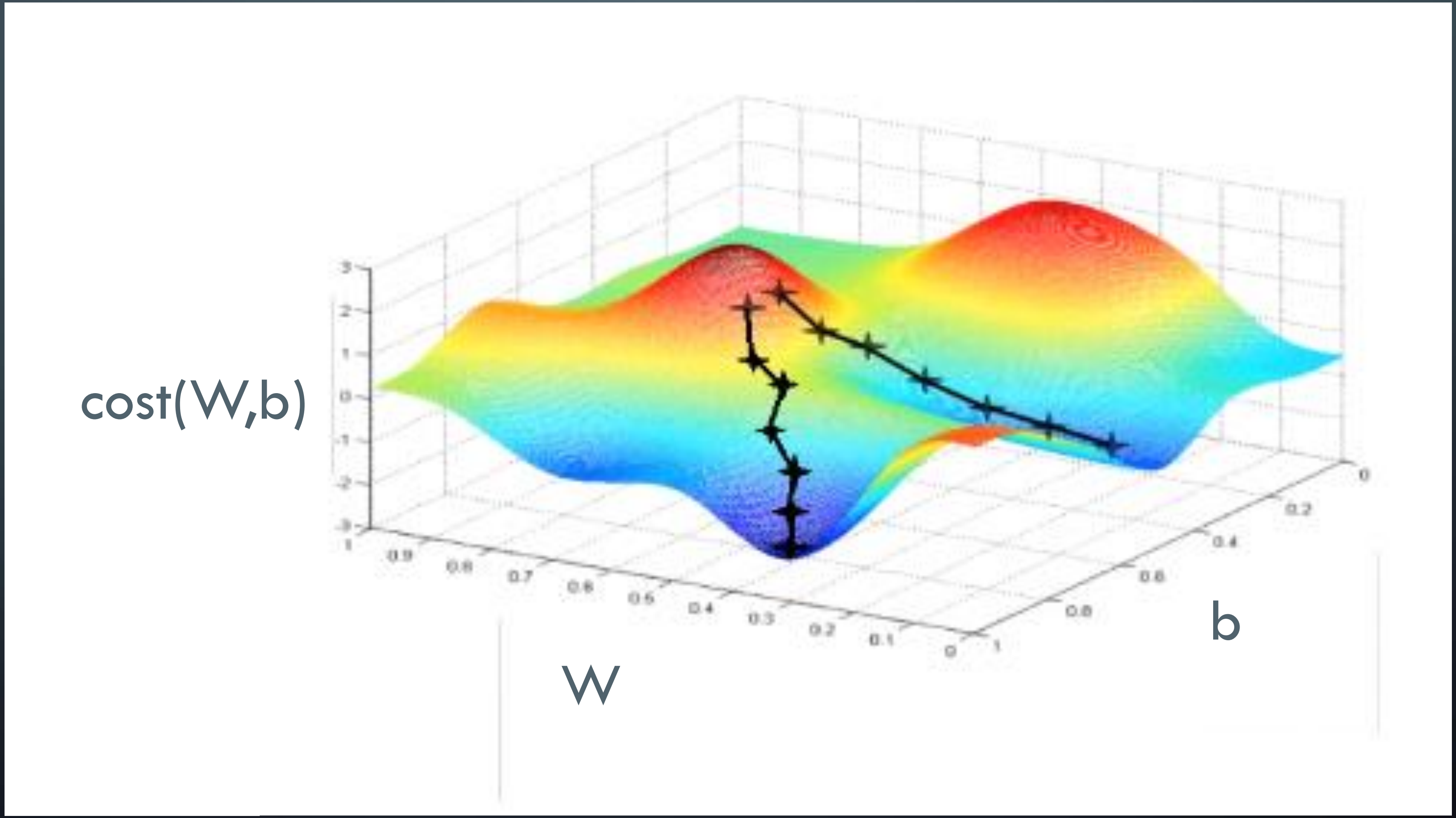
$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x_i) - y_i)^{\,2}$$

$$W \Leftarrow W - \alpha\frac{\partial}{\partial W}cost(W,b)$$

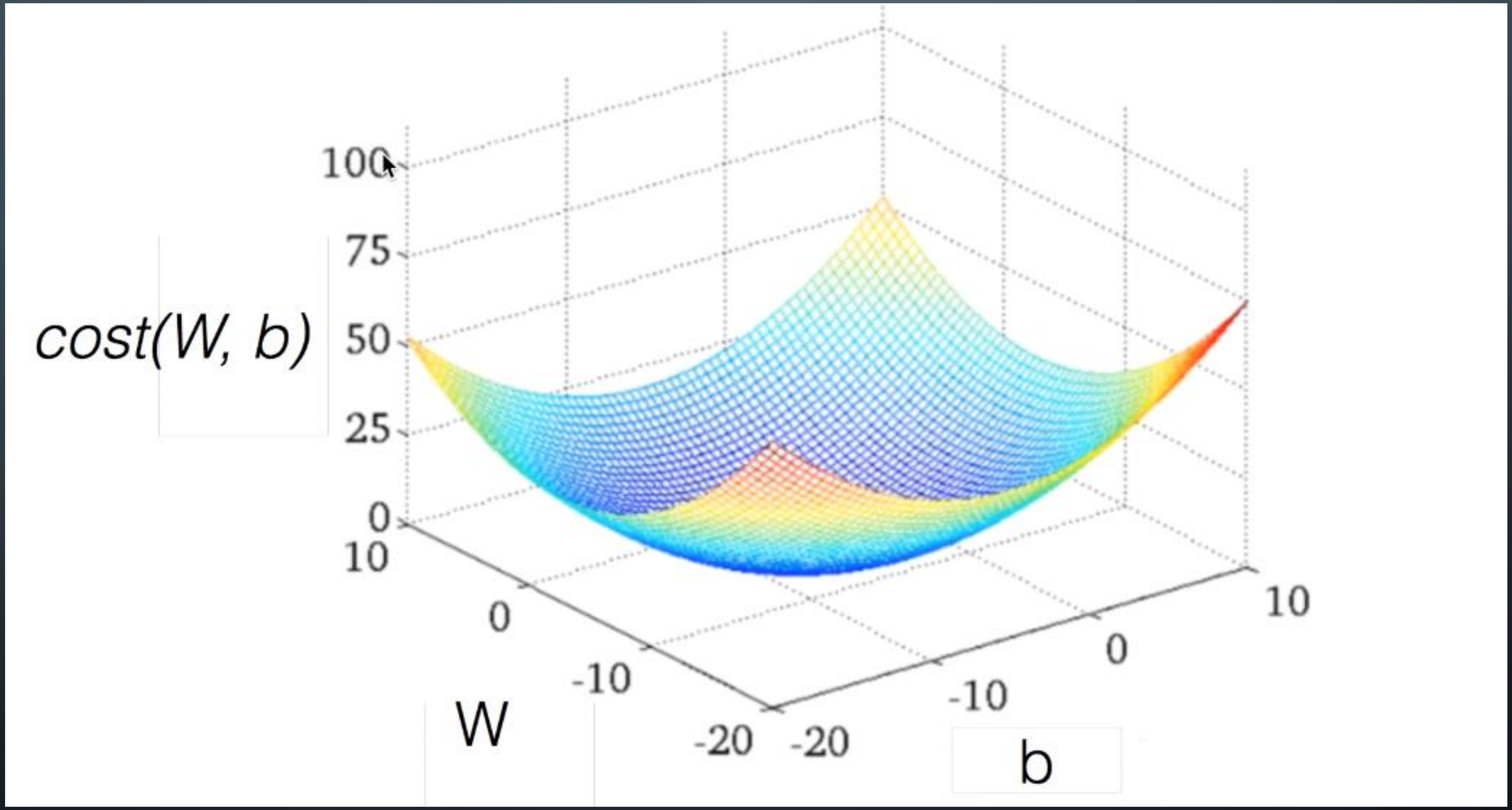


Gradient Descent Algorithm

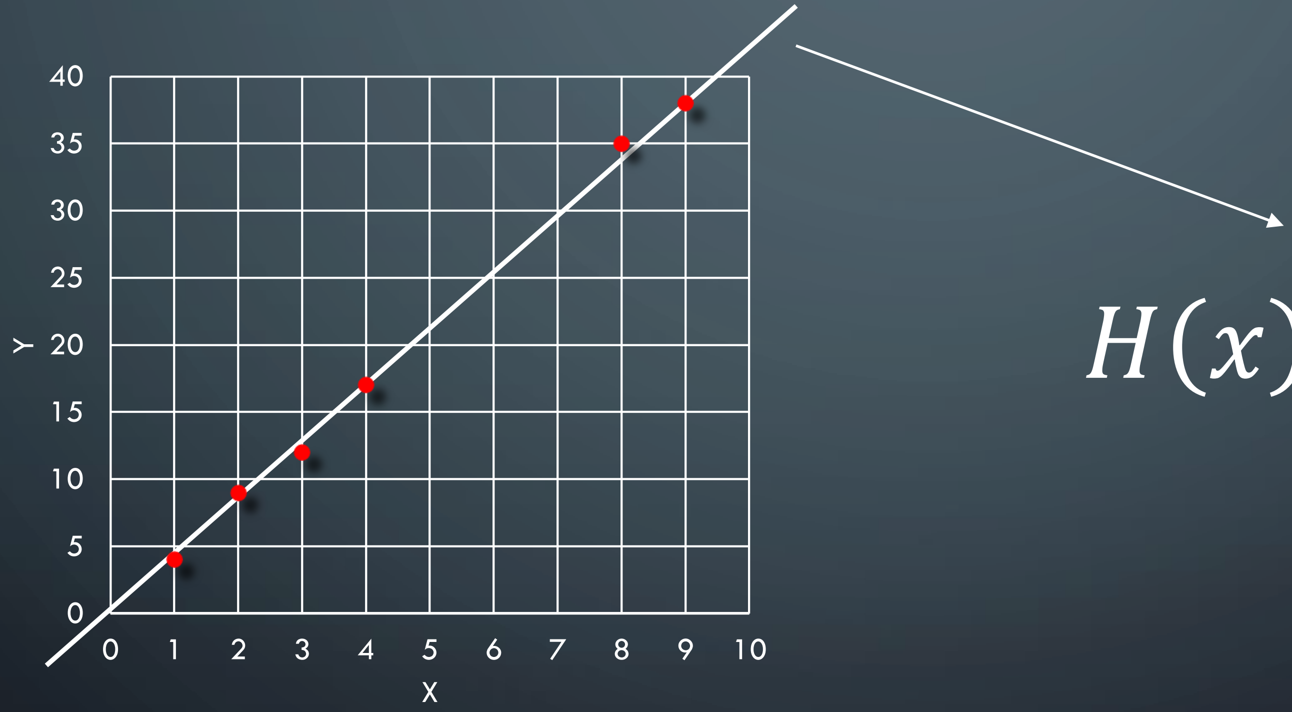
COST FUNCTION



COST FUNCTION



BEST LINEAR HYPOTHESIS

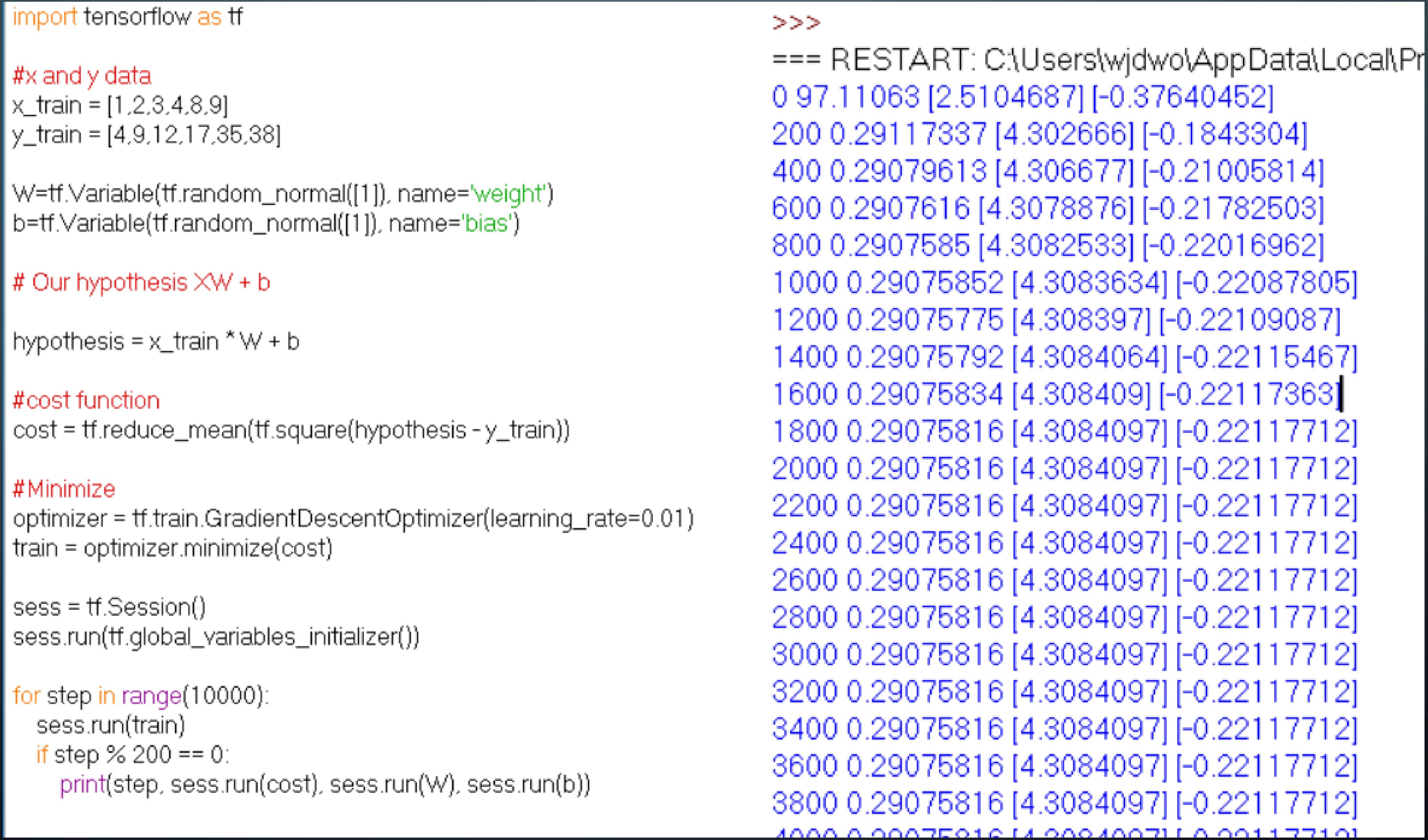


$$H(x) = Wx + b$$

IMPLEMENT VIA TENSORFLOW

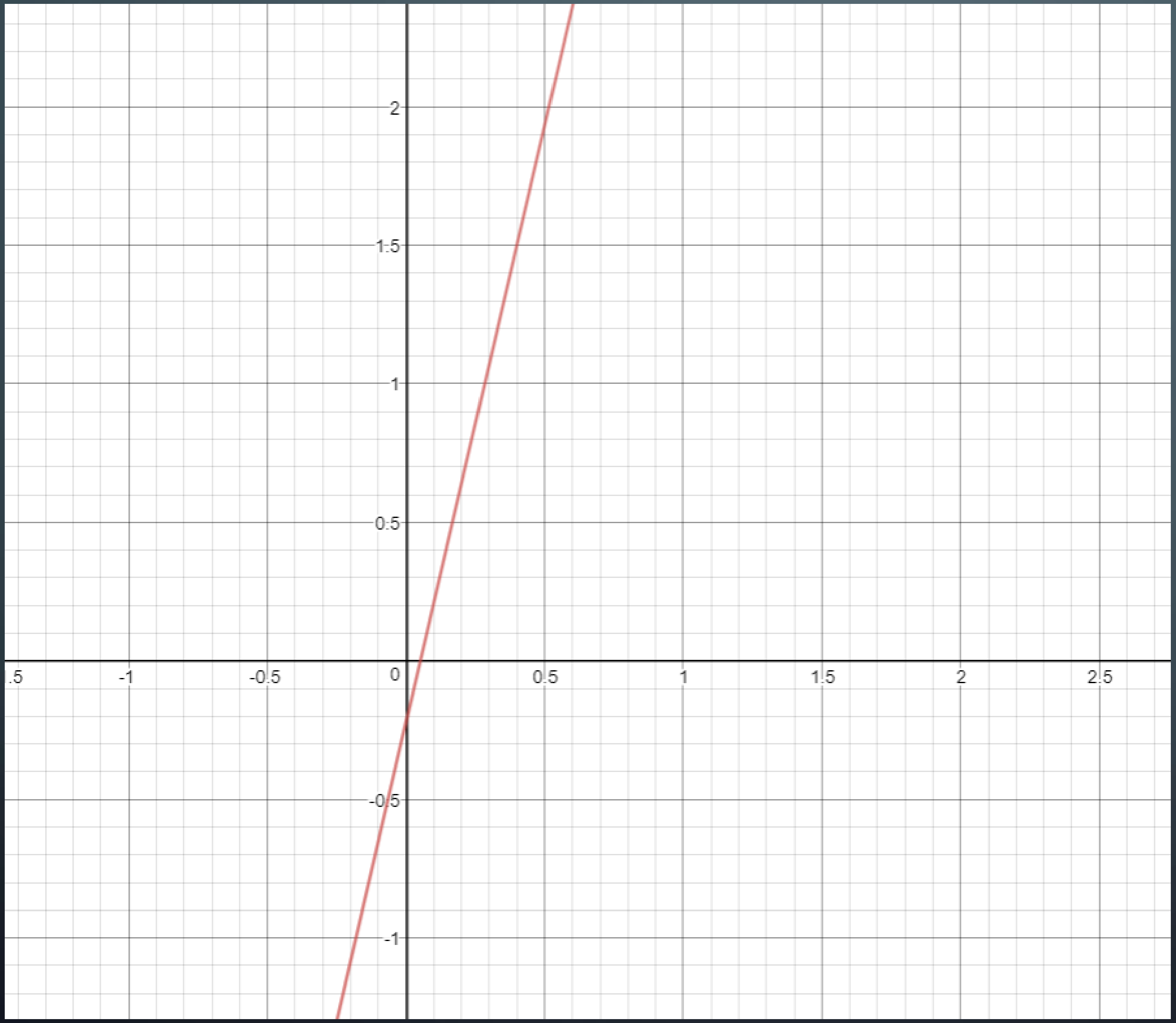
IT분야가 다 그렇듯 알고리즘은 소수의 천재가 만들어내고 대부분의 엔지니어는 그것들을 적재적소에 활용하는 역할을 맡는다. 알려진 알고리즘들은 이미 함수로 구현까지 끝나 있다. 따라서 이 문서를 보고 있을 대부분의 개발자들은 각 알고리즘의 장점과 단점을 외우고 호출방식을 익히는 것이 실용적인 면에서 훨씬 중요하다.

- in 꺼무위키



```python
import tensorflow as tf

#x and y data
x_train = [1,2,3,4,8,9]
y_train = [4,9,12,17,35,38]

W=tf.Variable(tf.random_normal([1]), name='weight')
b=tf.Variable(tf.random_normal([1]), name='bias')

# Our hypothesis XW + b

hypothesis = x_train * W + b

#cost function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))

#Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for step in range(10000):
    sess.run(train)
    if step % 200 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

```
>>>
=== RESTART: C:\Users\wjdwo\AppData\Local\Pr
0 97.11063 [2.5104687] [-0.37640452]
200 0.29117337 [4.302666] [-0.1843304]
400 0.29079613 [4.306677] [-0.21005814]
600 0.2907616 [4.3078876] [-0.21782503]
800 0.2907585 [4.3082533] [-0.22016962]
1000 0.29075852 [4.3083634] [-0.22087805]
1200 0.29075775 [4.308397] [-0.22109087]
1400 0.29075792 [4.3084064] [-0.22115467]
1600 0.29075834 [4.308409] [-0.22117363]
1800 0.29075816 [4.3084097] [-0.22117712]
2000 0.29075816 [4.3084097] [-0.22117712]
2200 0.29075816 [4.3084097] [-0.22117712]
2400 0.29075816 [4.3084097] [-0.22117712]
2600 0.29075816 [4.3084097] [-0.22117712]
2800 0.29075816 [4.3084097] [-0.22117712]
3000 0.29075816 [4.3084097] [-0.22117712]
3200 0.29075816 [4.3084097] [-0.22117712]
3400 0.29075816 [4.3084097] [-0.22117712]
3600 0.29075816 [4.3084097] [-0.22117712]
3800 0.29075816 [4.3084097] [-0.22117712]
4000 0.29075816 [4.3084097] [-0.22117712]
```



$x = 5\,, \quad y \approx 21.3209$

0x00

sigmoid function

softmax regression

Nural Network