Week1 Presentation

Linear regression, Logistic regression

GDSC Hanyang ML/DL : Basic, Jaeseung Lee

Index

- Machine Learning
  - Supervised/Unsupervised
- Linear Regression
  - Definition
  - Hypothesis
  - Cost function
- Gradient Descent
- Multi-variable regression
- Logistic Regression
  - Linear vs Logistic
  - Sigmoid function
  - Cost function

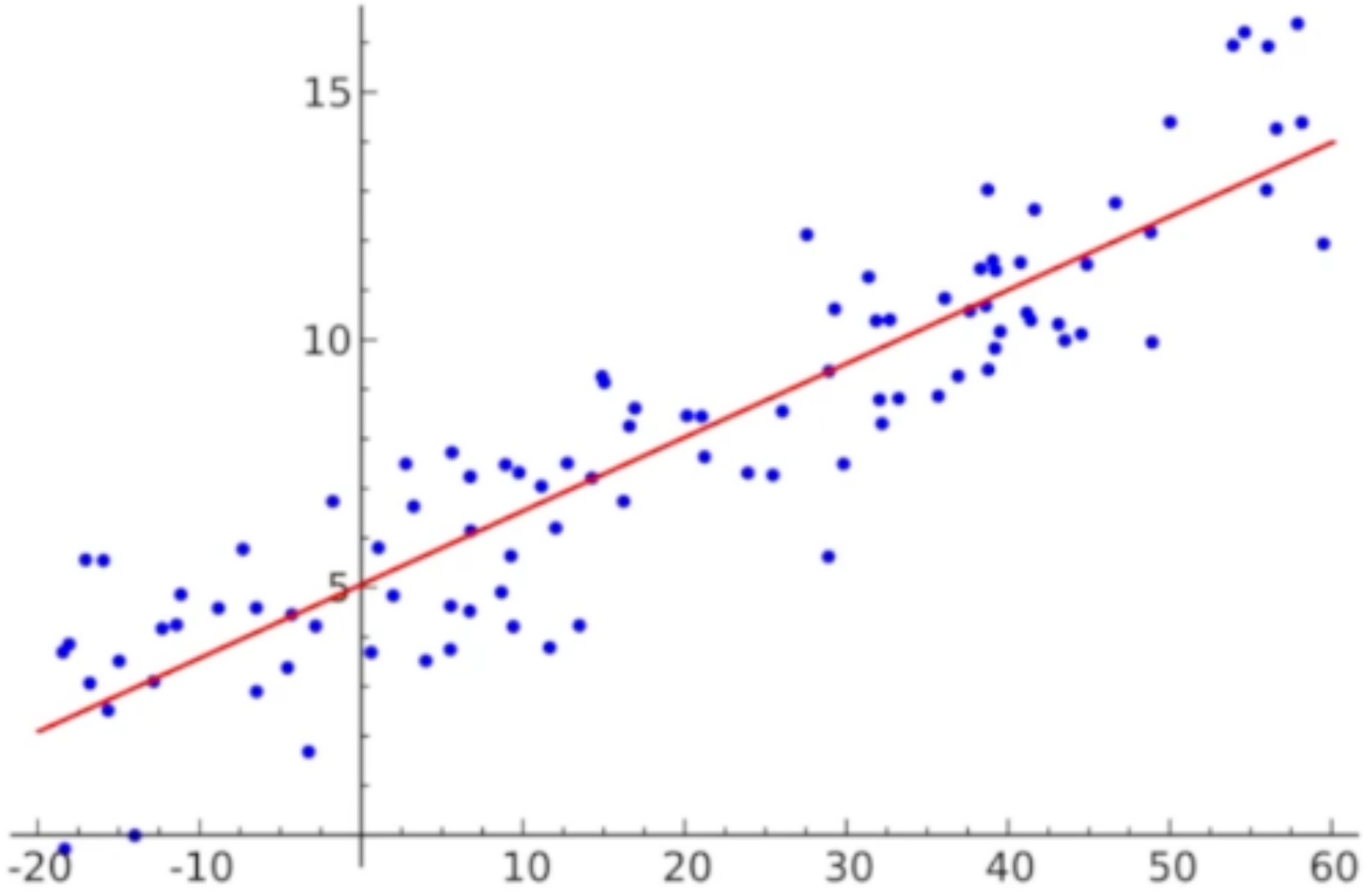
Machine Learning

Supervised / Unsupervised

- Supervised : labeling 된 데이터들을 학습
  - Regression : 점수 예측
  - Binary classification : P/F 예측
  - Multi-label classification : A,B,C,D,F 예측
- Unsupervised : labeling이 되지 않은 데이터들을 학습

Linear Regression

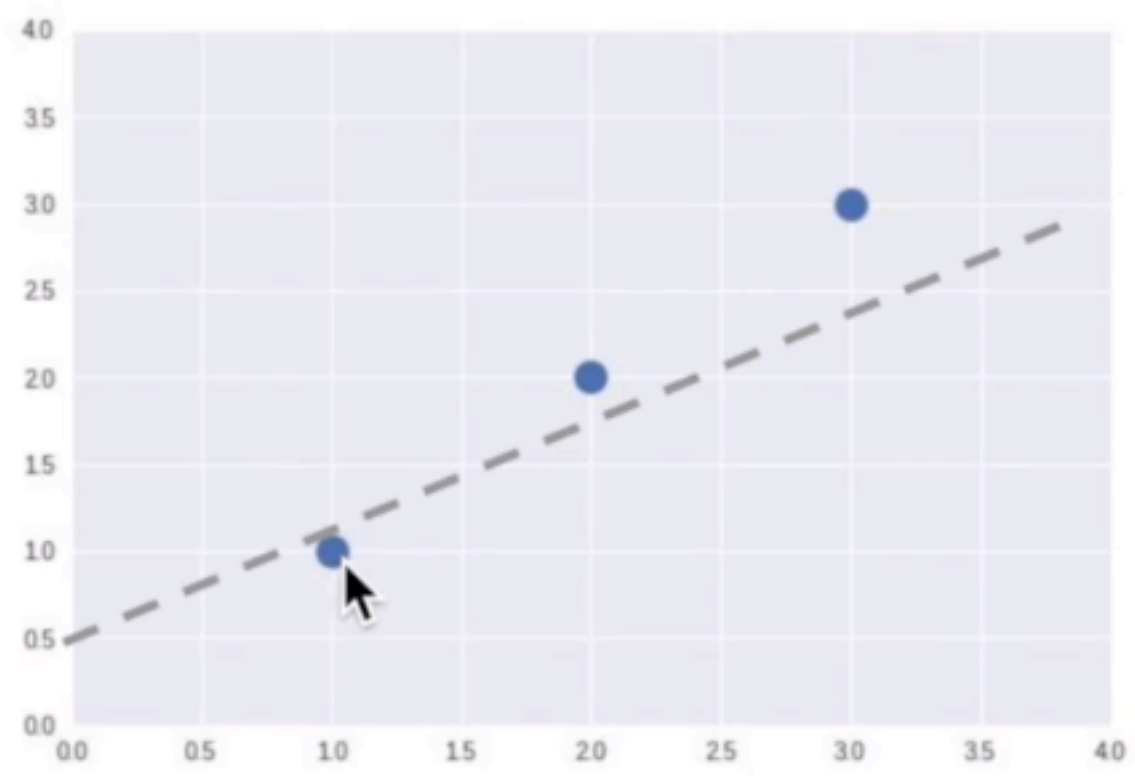
Definition



- Regression (회귀)
  - 회귀란 다시 처음으로 돌아온다는 뜻
  - 통계학에서 회귀 분석이란 관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한 뒤 적합도를 측정해 내는 분석 방법을 일컬음.
- Linear Regression (선형 회귀)
  - 두 변수 사이의 모형이 직선인 회귀 분석 방법
  - 데이터들을 잘 대변하는 직선의 방정식(**H(x) = Wx + b**)을 찾는 것이 목표

Linear Regression
Hypothesis

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



- Definition
  - Hypothesis : 가설이란 의미로, Linear Regression에서는 직선의 방정식.
- **H(x) = Wx + b**
  - H(x) : x를 변수로 하여 나온 가설의 결과 값
  - W : Weight를 의미, 직선의 기울기를 나타내는 변수
  - b : bias를 의미, y 절편(의미상)을 나타내는 변수

Linear Regression

Cost / Cost function

- Cost / Cost function
  - 가설에 의한 값과 실제 값과의 차이를 Cost(= Loss, Error)라고 함.
  - Cost function 이란 Cost를 구하기 위한 함수.

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

Linear Regression

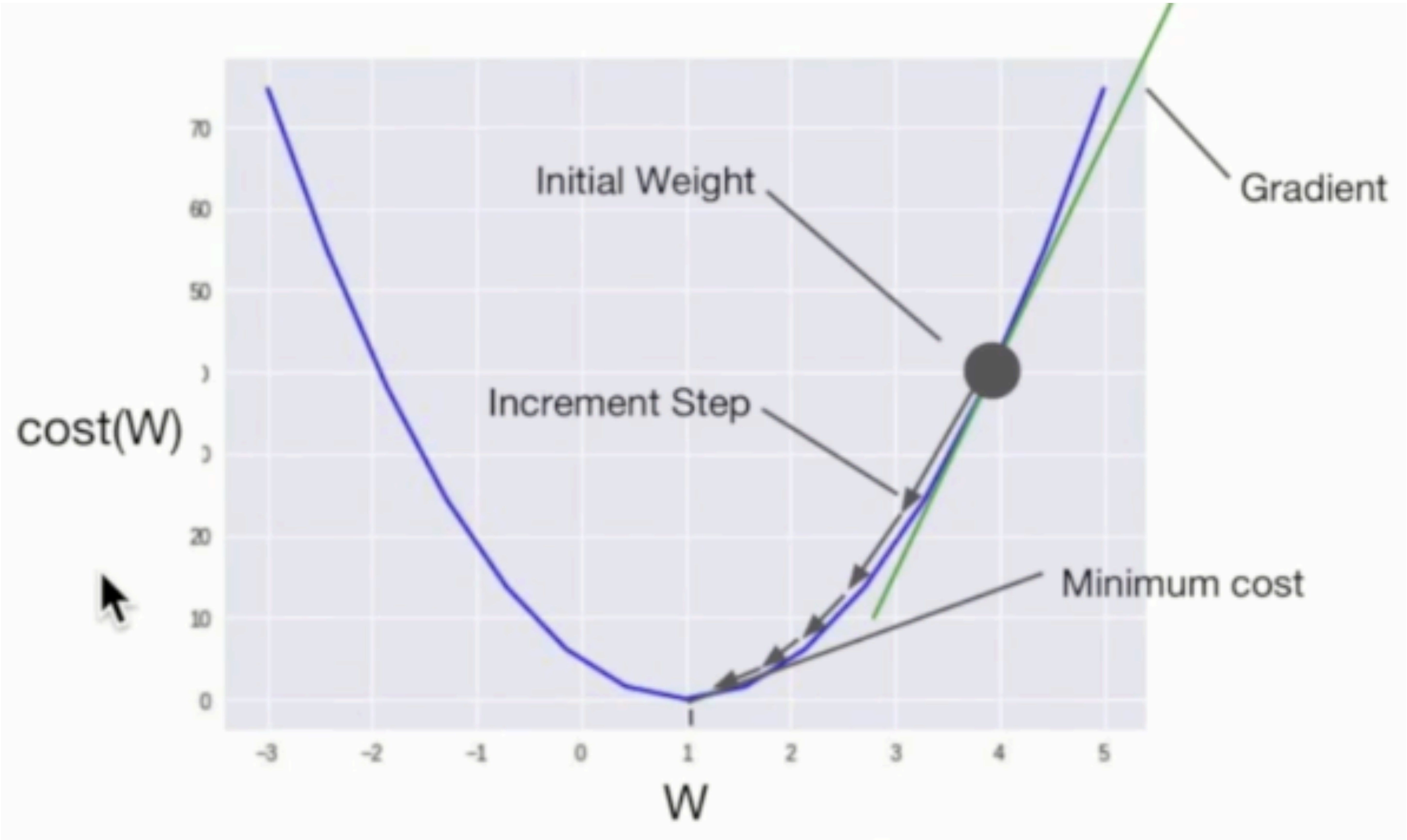
Cost / Cost function : MSE (Mean Squared Error)

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

- H(x) - y
  - 가설의 결과값과 실제 값과의 차이를 구한다.
- ^2
  - 제곱의 이유 : 가설의 결과값과 실제 값과의 차이가 중요한데, 그 차이가 양수/음수로 서로 더하고 빼지므로 오차의 의미가 퇴색됨.
  - 절대값을 사용해도 되지 않을까 ?
    - 된다. 그러나 제곱 방식이 더 효율적인 Cost function을 이룰 수 있음. (선형 vs 비선형 속도 차이)

Gradient descent

One of method to minimize cost



- Gradient : 비용함수 기울기
- W에 그 기울기를 곱하여 W에서 빼는 방법.
- 빼면 뺄수록 비용이 최소가 되는 지점에 가까워짐.
- 기울기가 음수일 때, W값이 증가 되기 때문에 결국 최소 비용에 가까워지므로 적용 가능.
- 단점은 Convex function (볼록함수)가 아닌 경우 local minimum에 빠지는 경우가 생길 수 있음.

Gradient descent

Formal definition

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

$$\Downarrow$$

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

$$W := W - \alpha\frac{\partial}{\partial W}\frac{1}{2m}\sum_{i=1}^{m}\left(W(x_i) - y_i\right)^2$$

$$W := W - \alpha\frac{1}{2m}\sum_{i=1}^{m}2\left(W(x_i) - y_i\right)x_i$$

$$\boxed{W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(W(x_i) - y_i\right)x_i}$$

- Gradient 를 구하기 위해 Cost function을 미분하고, 그 값을 뺀 기울기에서 뺀 것
- 편의를 위해 bias를 제거, alpha(learning rate)로 gradient descent의 속도를 조절함

Gradient descent

Implement

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (W(x_i) - y_i) x_i$$

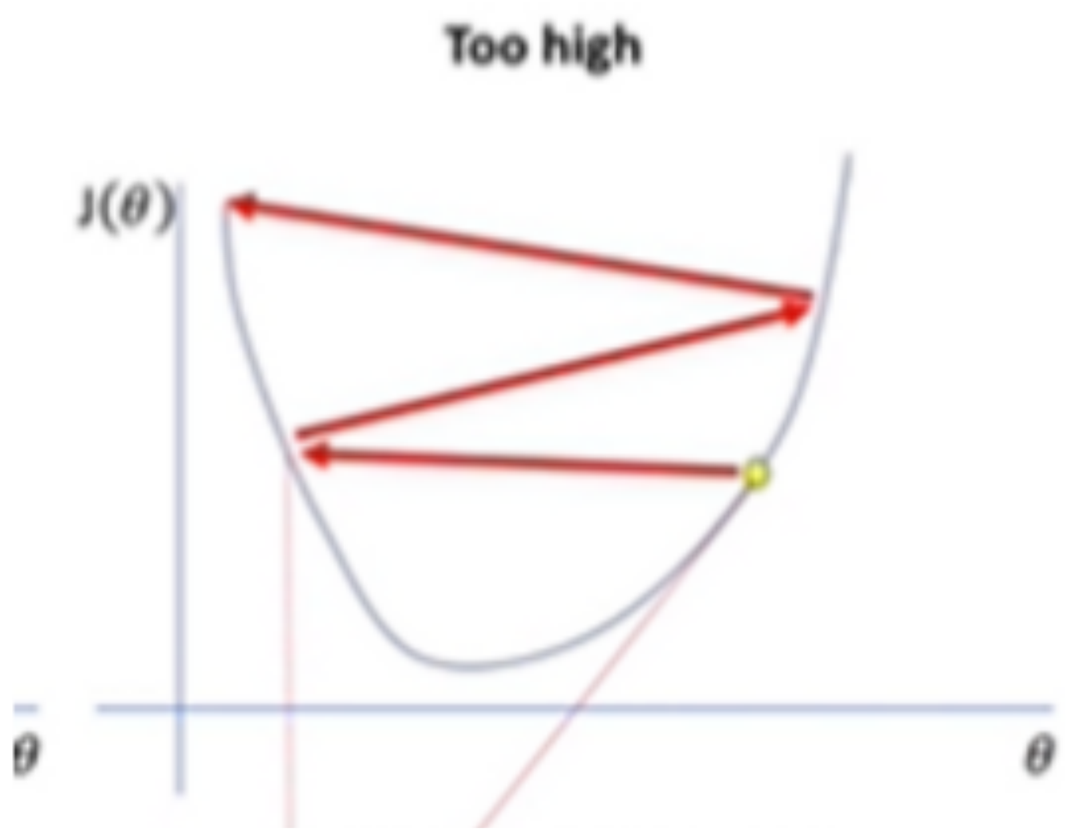
```
for step in range(300):
  hypothesis = W * X
  cost = tf.reduce_mean(tf.square(hypothesis - Y))

  learning_rate = 0.01
  gradient = tf.reduce_mean(tf.multiply(tf.multiply(W, X) - Y, X))
  descent = W - tf.multiply(learning_rate, gradient)
```

- 실제 구현한 코드 (Lab03)
- tf.square() : 제곱 함수
- tf.reduce_mean() : 평균을 구하는 함수 (reduce는 차원을 줄인다는 의미)

Gradient descent

More details



- Alpha (learning rate)
  - Learning rate가 작으면 손실함수의 최소값을 구하는 데 오래 걸림.
  - Learning rate가 너무 크면 줄어드는 방향이 아니라 발산하여 찾을 수 없게 된다.
  - 따라서 Learning rate scheduling 을 하여 이를 적절히 조절한다.
- Learning rate scheduling
  - 반복 횟수에 따라 반영 Learning rate를 다르게 (초반에 크게, 후반에 작게) 한다.

Multi-variable regression

$$H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x_1, x_2, x_3) - y_i)^2$$

Hypothesis using matrix

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

$$H(X) = XW$$

- Multi-variable 일 때는 변수의 개수가 늘어나고, 그만큼 가중치의 개수도 늘어남.
- 이때 행렬(Matrix)을 사용하면 효과적으로 다변수들을 다룰 수 있음.
    - 표현상에서 훨씬 간단해지며, 실제 코드 효용성도 높아지는 결과를 가져옴.

Multi-variable regression

Code efficiency

```python
w1 = tf.Variable(tf.random.normal([1]))
w2 = tf.Variable(tf.random.normal([1]))
w3 = tf.Variable(tf.random.normal([1]))
b = tf.Variable(tf.random.normal([1]))
```

```python
for i in range(1000 + 1):
  with tf.GradientTape() as tape:
    hypothesis = w1 * x1 + w2 * x2 + w3 * x3 + b
    cost = tf.reduce_mean(tf.square(hypothesis - Y))
  w1_grad, w2_grad, w3_grad, b_grad = tape.gradient(cost, [w1, w2, w3, b])

  w1.assign_sub(learning_rate * w1_grad)
  w2.assign_sub(learning_rate * w2_grad)
  w3.assign_sub(learning_rate * w3_grad)
  b.assign_sub(learning_rate * b_grad)
```

Variable

```python
W = tf.Variable(tf.random.normal([3, 1]))
b = tf.Variable(tf.random.normal([1]))
```

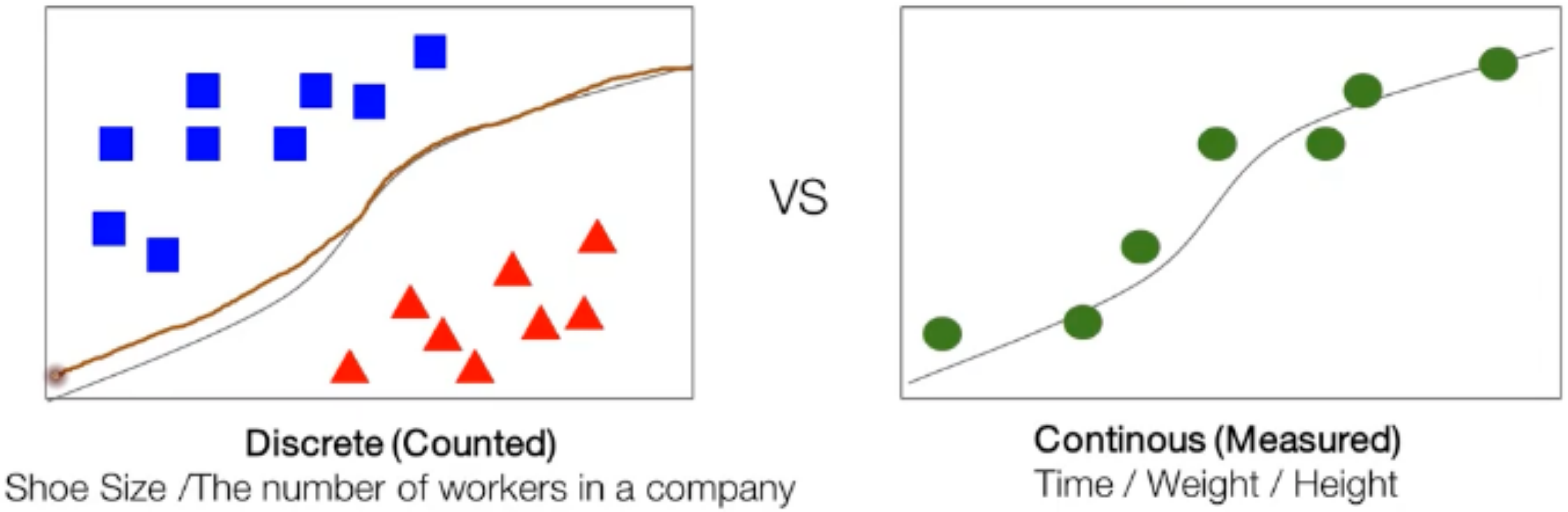
```python
n_epochs = 2000 # 반복 횟수
for i in range(n_epochs + 1):
  with tf.GradientTape() as tape:
    cost = tf.reduce_mean(tf.square(predict(X) - Y))
  W_grad, b_grad = tape.gradient(cost, [W, b])

  W.assign_sub(learning_rate * W_grad)
  b.assign_sub(learning_rate * b_grad)
```

Matrix

Logistic Regression

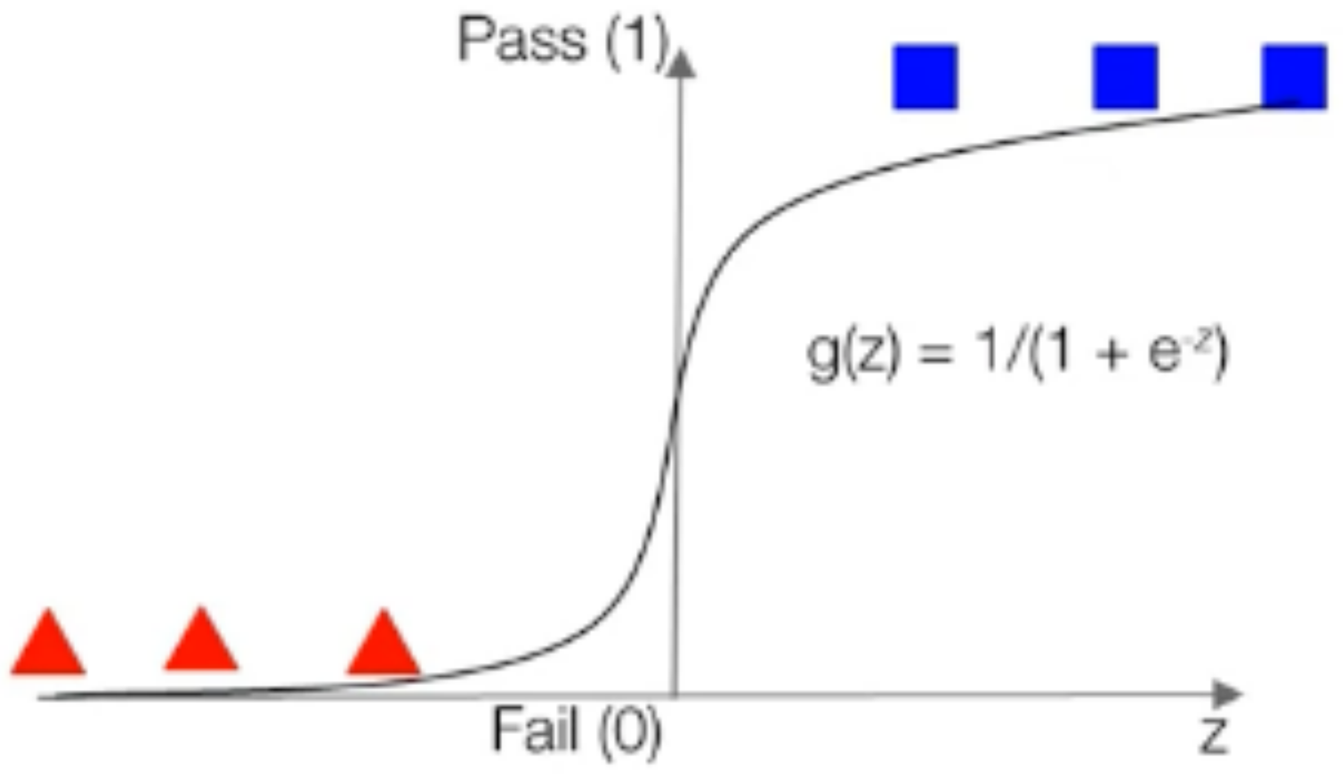
Linear vs Logistic

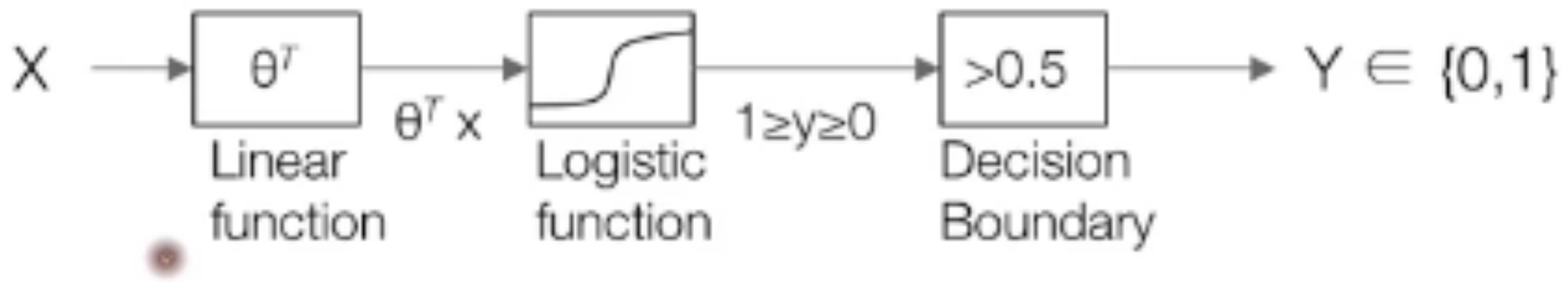


- 데이터 형식에 따라 Linear function을 사용할 지 Logistic function을 사용할지 나뉨
  - 발 사이즈, 회사 규모 등 불연속적인 데이터들을 분류할 때는 Logistic function
  - 시간, 몸무게, 키 등 연속적인 데이터들의 구간을 예측할 때는 Linear function

Logistic Regression

Sigmoid function



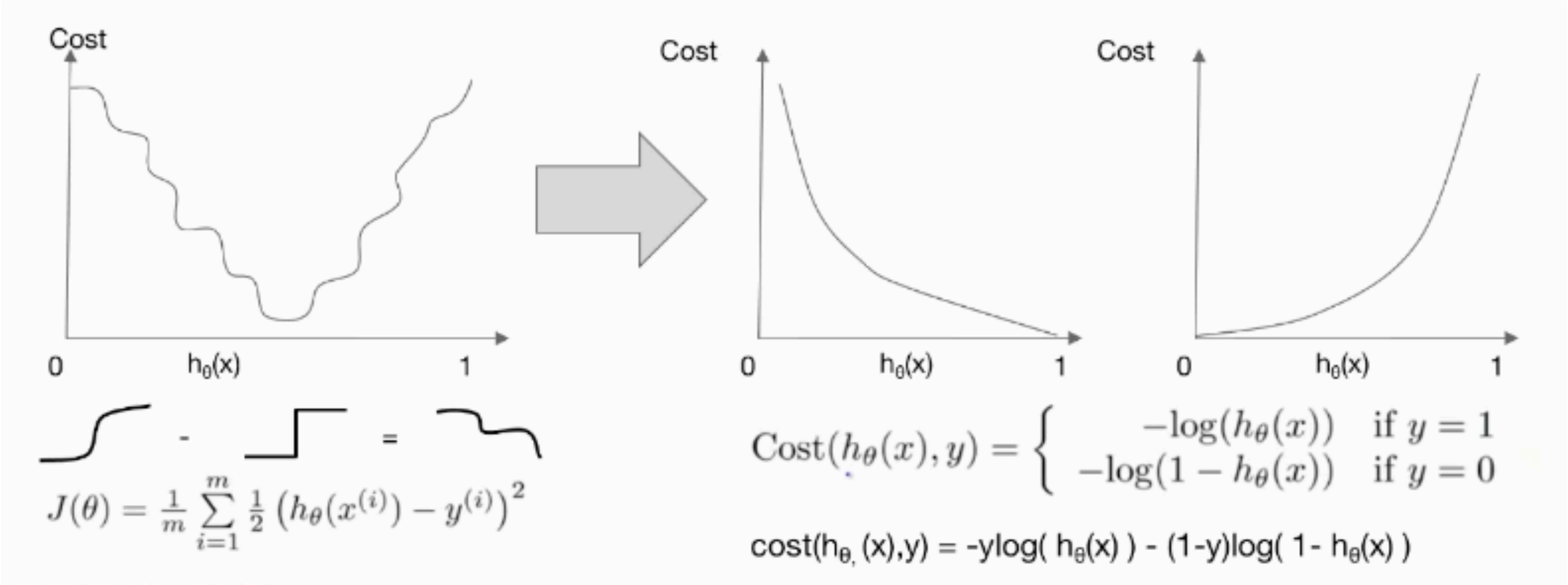


Where we define g(z) -> z is a real number -> $g(z) = e^z/(e^z + 1) = 1/(1 + e^{-z})$

- Why sigmoid?
  - Logistic은 1 또는 0을 구분해야 하기 때문에, 이에 적합한 Sigmoid function을 사용하여 데이터들을 구분한다.
  - z가 작아질 수록 0에 가까워 지고 ((1 + e^(-z)) -> infinity), 커질 수록 1에 가까워지는 특성이 있다.

Logistic Regression

Cost function



- Why use different cost function ?
  - Sigmoid function을 이전에 사용한 비용함수에 넣게 되면 저렇게 구불구불하게 되어 Gradient를 구하는데 비효율적임.
  - 따라서 Sigmoid function에 맞게 비용함수를 새로 지정하여 깔끔하게 2차함수 형태로 나올 수 있게 함.

Logistic Regression

Implement

```
def logistic_regression(features):
  hypothesis = tf.divide(1., 1. + tf.exp(-tf.matmul(features, W) + b))
  return hypothesis

def loss_fn(hypothesis, labels):
  cost = -tf.reduce_mean(labels * tf.math.log(hypothesis)
    + (1 - labels) * tf.math.log(1 - hypothesis))
  # cost function 수식 재현
  return cost
```

- 실제 코드로 구현한 모습 (Lab05)

Thank you for listening

Week1 Presentation

Linear regression, Logistic regression

GDSC Hanyang ML/DL : Basic, Jaeseung Lee