# Logistic Regression - Classification

# 로지스틱 회귀 - Classification

- Binary
- 스팸 메일 탐지 : Spam or Ham
- Facebook feed : show or hide
- 신용카드 사기거래 탐지 :
  정상 거래 / 사기 거래

# 로지스틱 회귀 - Classification

## 0 / 1 encoding

- 스팸 메일 탐지 : Spam(1) or Ham(0)
- Facebook feed : show(1) or hide(0)
- 신용카드 사기거래 탐지 :
  정상적 거래(0) / 사기 거래(1)

# 로지스틱 회귀 - Classification

## Pass(1)/Fail(0) based on study hours

# Linear regression

- We know Y is 0 or 1

  $H(x) = Wx + b$

- Hypothesis can give values large than 1 or less than 0

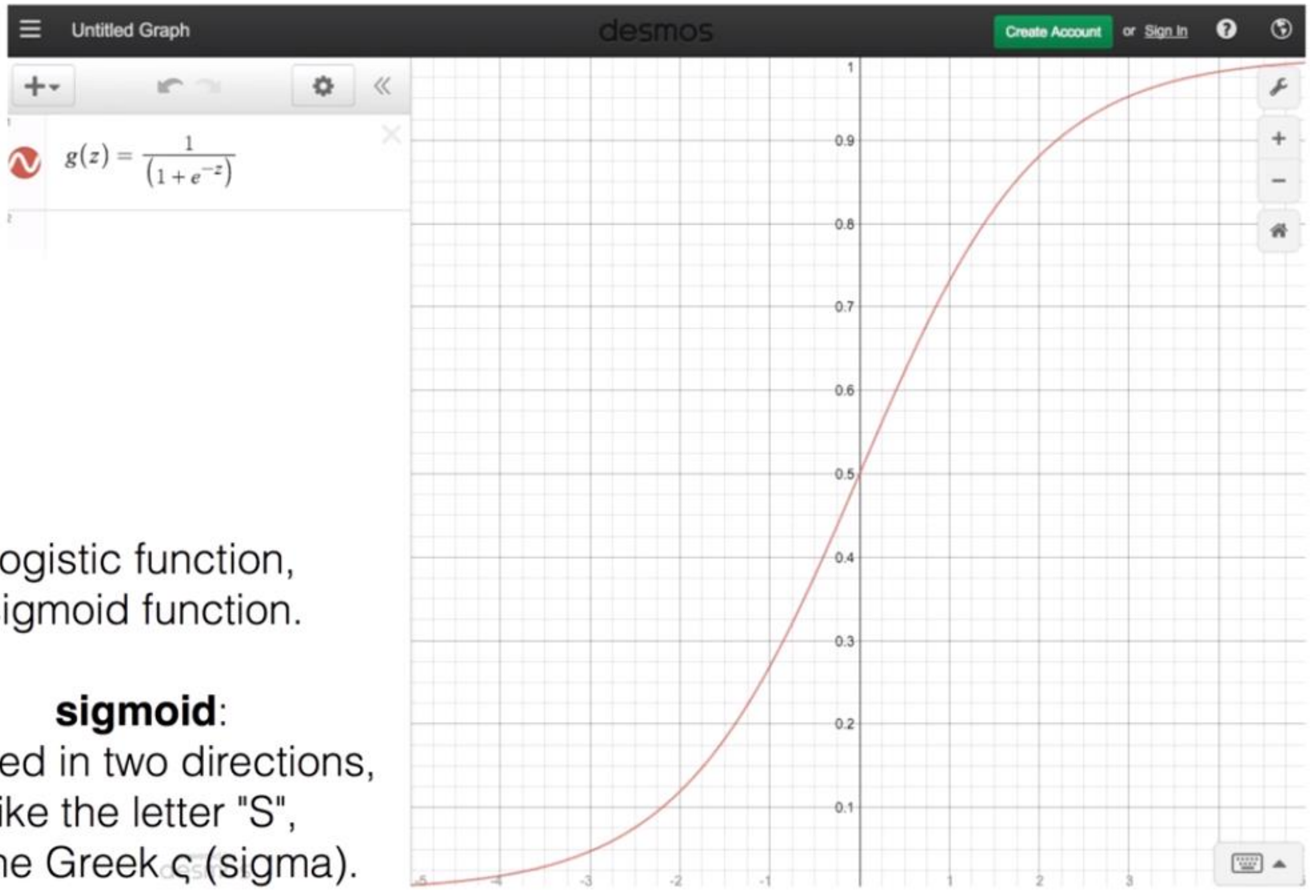
# 로지스틱 회귀 - Classification

## Logistic Hypothesis

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

# 로지스틱 회귀 - Classification

## Sigmoid




logistic function,
sigmoid function.

**sigmoid**:
Curved in two directions,
like the letter "S",
or the Greek ς (sigma).

# 로지스틱 회귀 - Classification

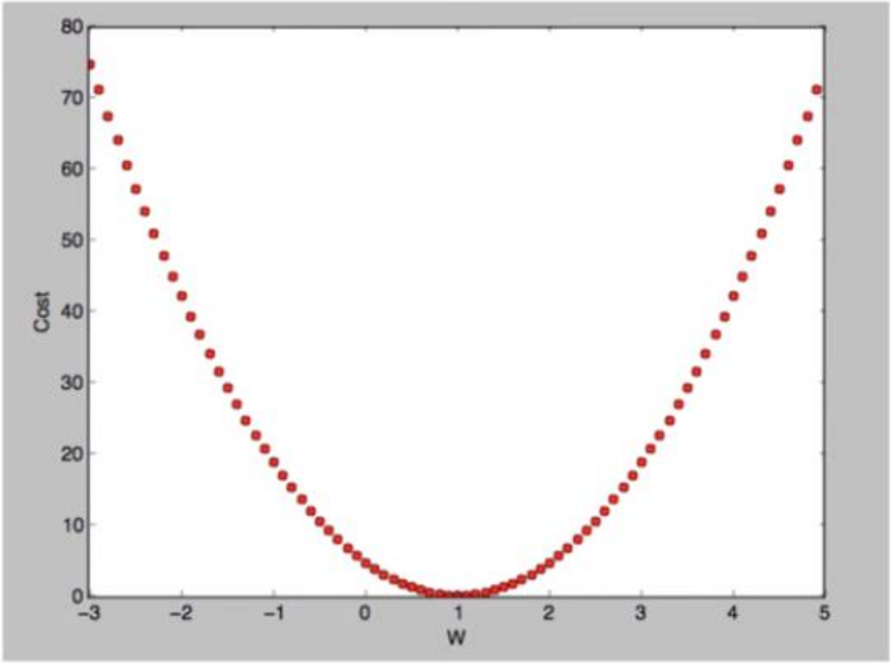
## Logistic Hypothesis

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

# 로지스틱 회귀 - Classification

## Cost

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



# 로지스틱 회귀 - Classification

## Cost function

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

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

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

# 로지스틱 회귀 - Classification

## New cost function for logistic

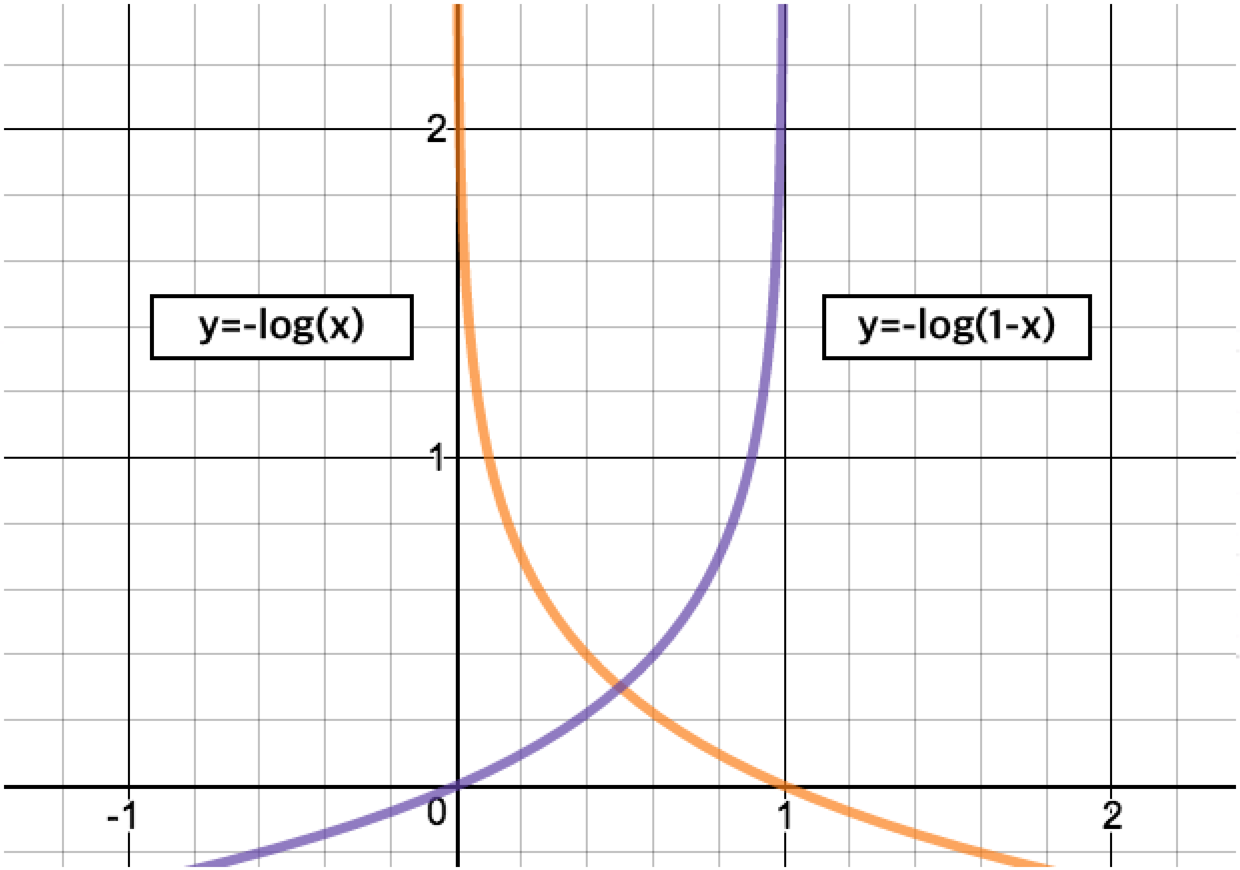
$$cost(W) = \frac{1}{m} \sum \quad c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

# 로지스틱 회귀 - Classification

## understanding cost function

$$C(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$



# 로지스틱 회귀 - Classification

## Cost function

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

$$c(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

# 로지스틱 회귀 - Classification

## Minimize cost - Gradient decent algorithm

$$cost(W) = -\frac{1}{m}\sum y log(H(x)) + (1-y) log(1-H(x))$$

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

```
# cost function
cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis)))

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```