

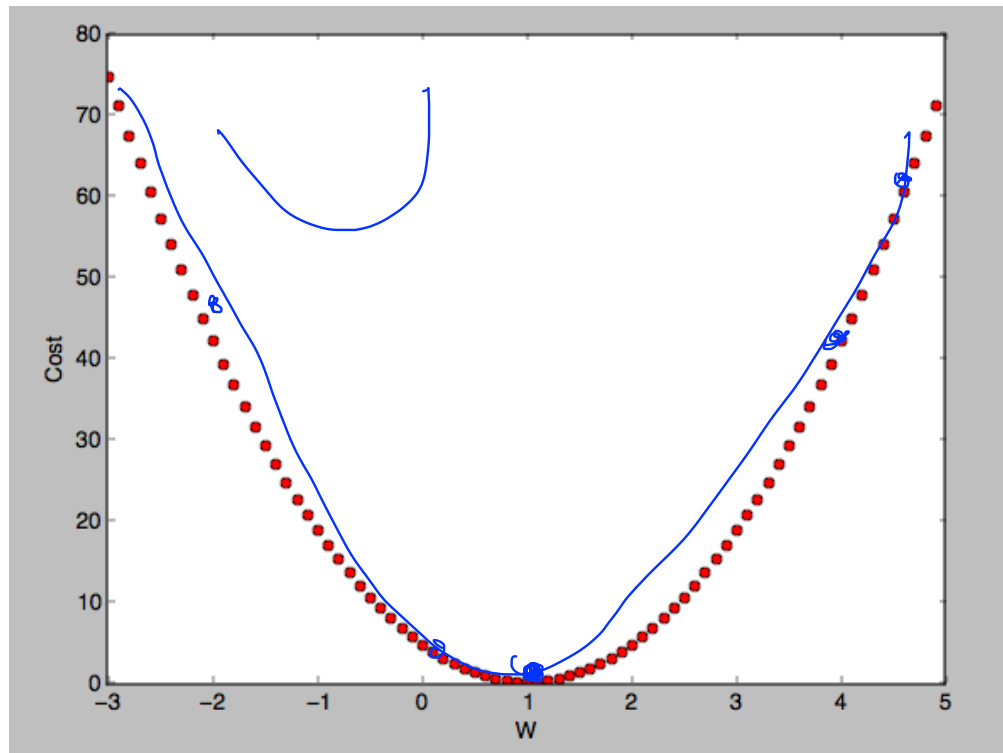
# Lecture 5-2

Logistic (regression) classification:  
cost function & gradient decent

Sung Kim <hunkim+mr@gmail.com>

# Cost

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



장점은 어느 점에서나 cost의 최저점을 찾을 수 있다는 것

# Cost function

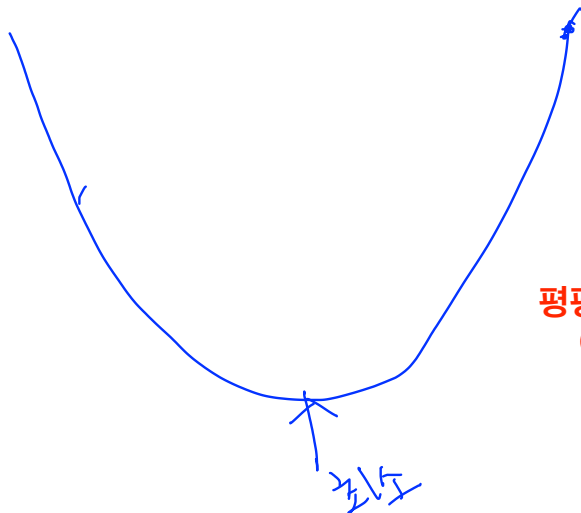
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

sigmoid가 제곱에 들어가면서 더 구불구불해짐  
 $0 < \sim < 1$

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

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

sigmoid



단점  
 어느 지점에서 시작하냐에 따라서 최저점을 못찾을 수도 있다



평평해지는 곳에서 최저점으로 인식할 수도 있다  
 이런 부분은 local minimum이라고 한다

전체에서 최저점 global minimum

즉, 이런 함수는 Gradient Descent 알고리즘에 사용할 수 없기 때문에 다시 형태를 바꿔야 한다

# New cost function for logistic

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

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

이 **c**함수는 **y**를 두 가지의 경우로 나누어서 정의한다

그러면 왜 두 가지로 나누어서 정의하는 것일까?

# understanding cost function

cost 함수에서 cost의 값은

우리가 예측한 값이 실제값과 비슷하거나 같으면 작아지고,  
예측값과 실제값의 차이가 크면 그 값이 커지는 함수이다

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

$$\frac{1}{1+e^{-2}}$$

Log

H(x)에 익스포넨셜이 쓰이므로,

구부러짐을 잡기 위해 역함수인 log 함수를 쓰는 게 기본 아이디어

$$y=0$$

$$H(x)=0, \text{ cost}=0$$

$$H(x)=1, \text{ cost}=\infty \uparrow$$

~~Cost~~  $y=1$   
실제값

$$H(x)=1 \rightarrow \text{cost}(1)=0$$

예측값

$$H(x)=0 \rightarrow \text{cost}=\infty \uparrow$$

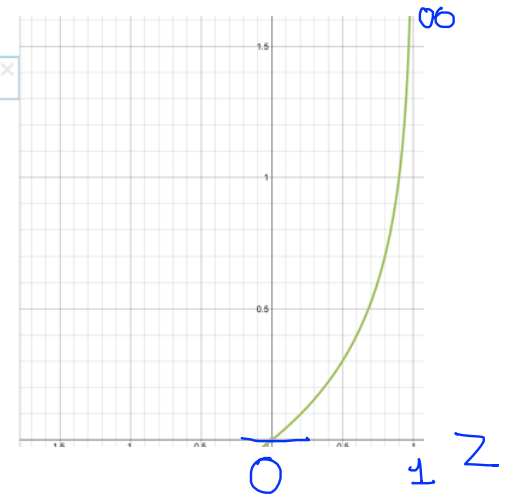
cost  
//  
 $g(z) = -\log(z)$   
 $g(z) = -\log(z)$

$g(z) = -\log(z)$



$$-\log(1-z)$$

$-\log(1-z)$



Hypothesis에서 1보다 큰 값을 낼 수 없으므로 신경쓰지 않고

# Cost function

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

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

마이너스 붙인 것

두 개를 하나로 합침

$$C(H(x), y) = \underline{-y} \log(H(x)) - \underline{(1 - y)} \log(1 - H(x))$$

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

~~]~~

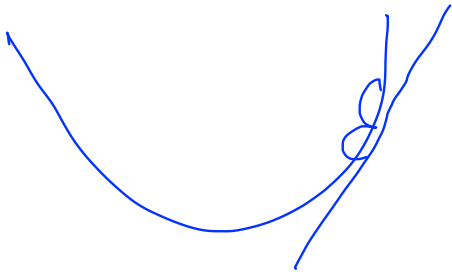
$$y=0,$$

~~]~~

$$c = -1 * \log(1 - H(x))$$

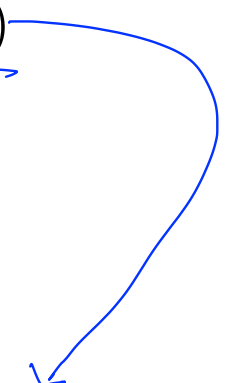
# Minimize cost - Gradient decent algorithm

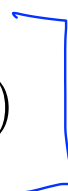
$$\underline{\text{cost}(W)} = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$



$$W := W - \underbrace{\left[ \alpha \frac{\partial}{\partial W} \text{cost}(W) \right]}$$

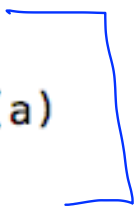
# Gradient decent algorithm

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


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


```
# 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)
```





**Next**  
Multinomial  
classification (Softmax)

