Lecture 05

Logistic (regression) classification

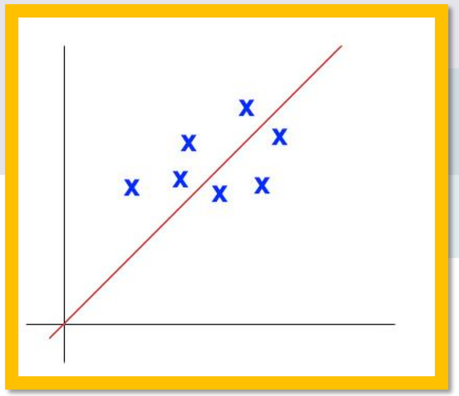
Regression

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

- **Hypothesis :**
- **Cost function :**
- **Gradient descent :**

Regression

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

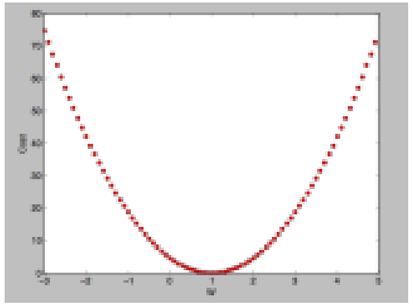


- **Hypothesis :** $H(X) = WX$
- **Cost function :** $cost(W) = \frac{1}{m}\sum(WX - y)^2$
- **Gradient descent :**

cost function 을 미분 → Gradient

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

Step size
(learning rate)



Classification

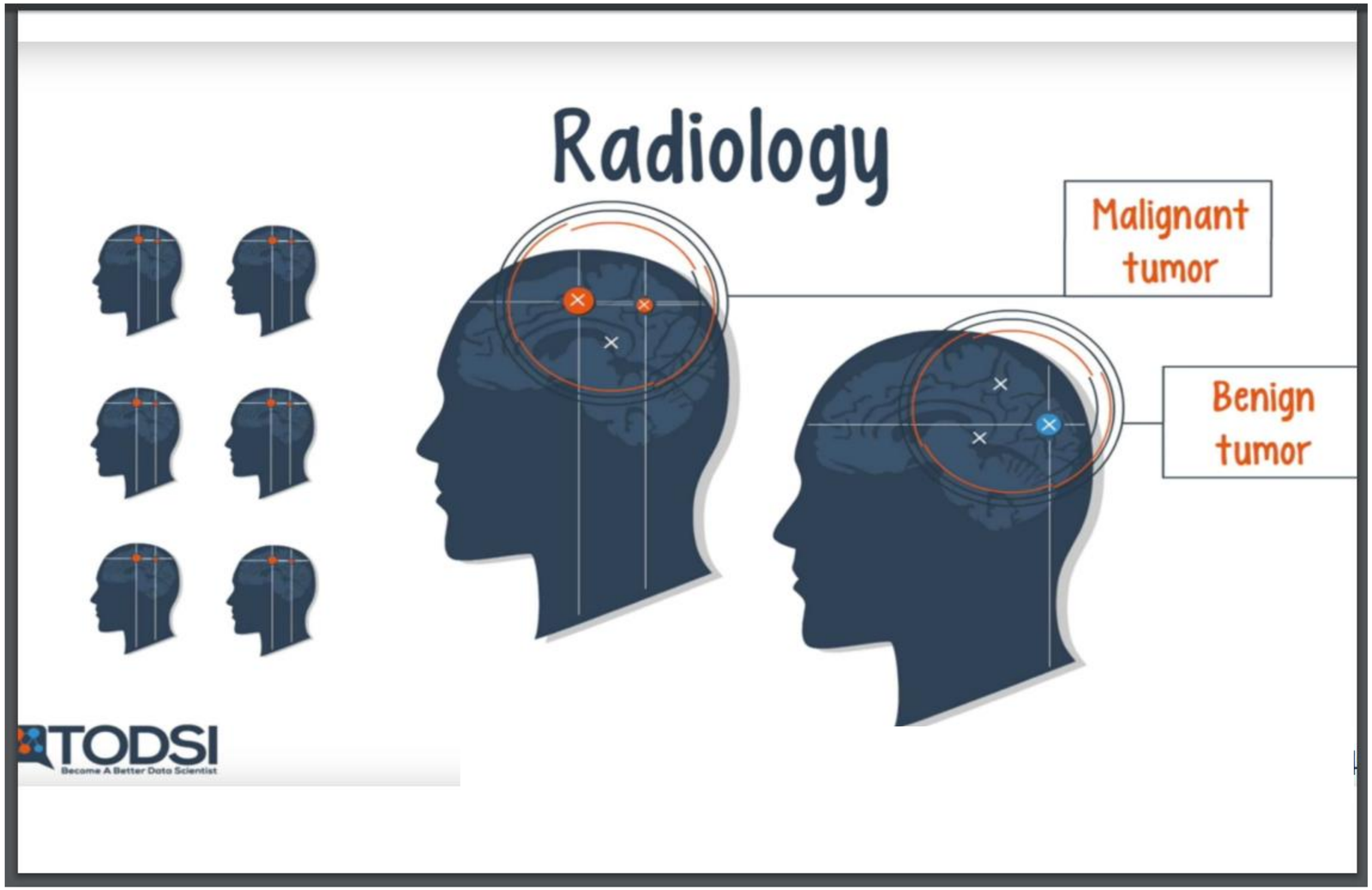
- **Spam Detection: Spam or Ham**
- **Facebook feed: show or hide**
- **Credit Card Fraudulent Transaction detection : legitimate or fraud**

0, 1 encoding

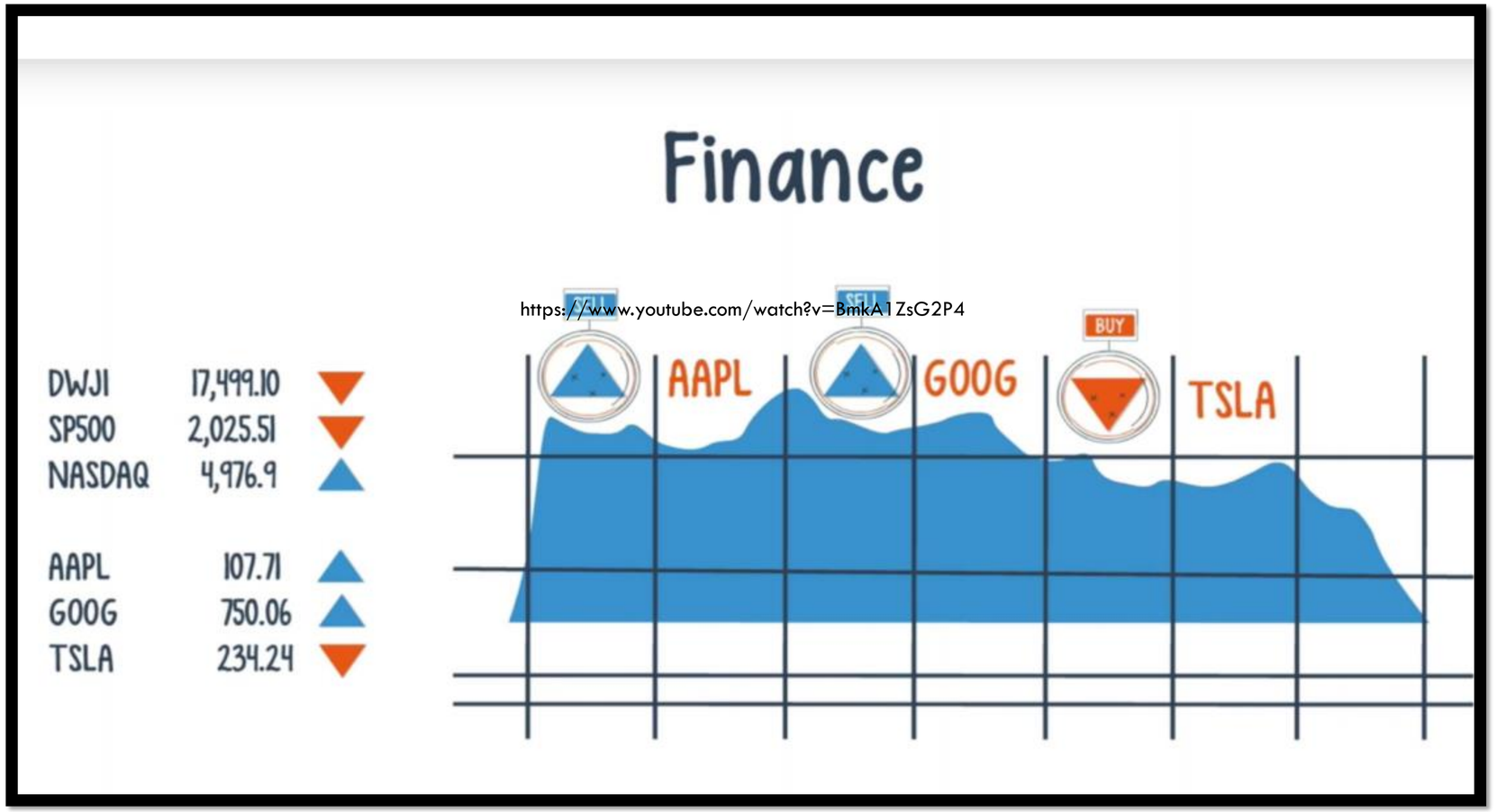
- **Spam Detection: Spam (1) or Ham (0)**
- **Facebook feed: show (1) or hide (0)**
- **Credit Card Fraudulent Transaction detection : legitimate (0) or fraud (1)**

Radiology

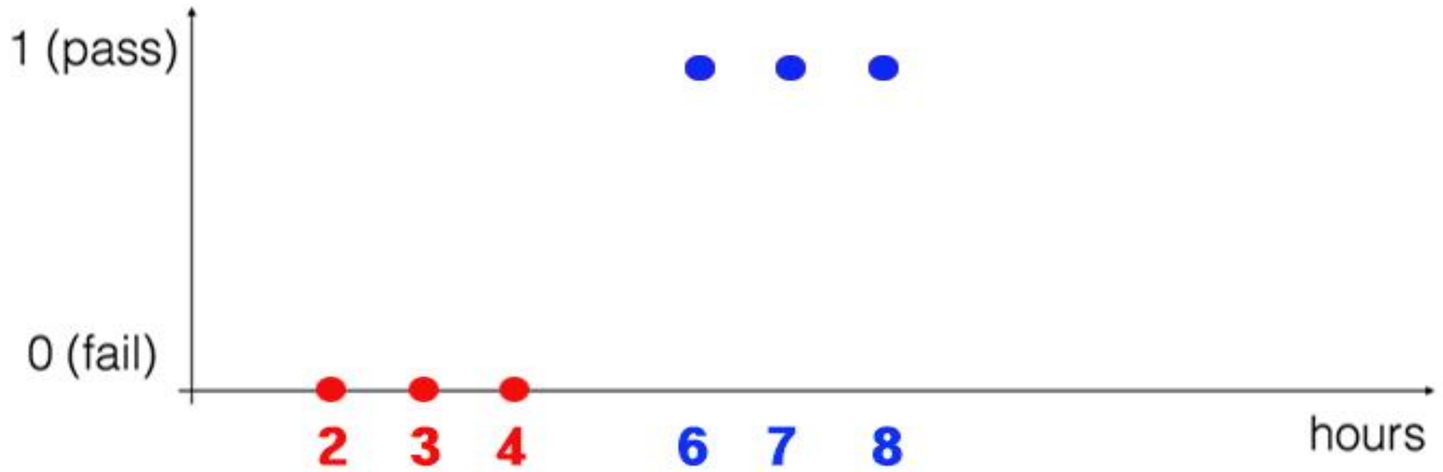
https://www.youtube.com/watch?v=BmkA1ZsG2P4



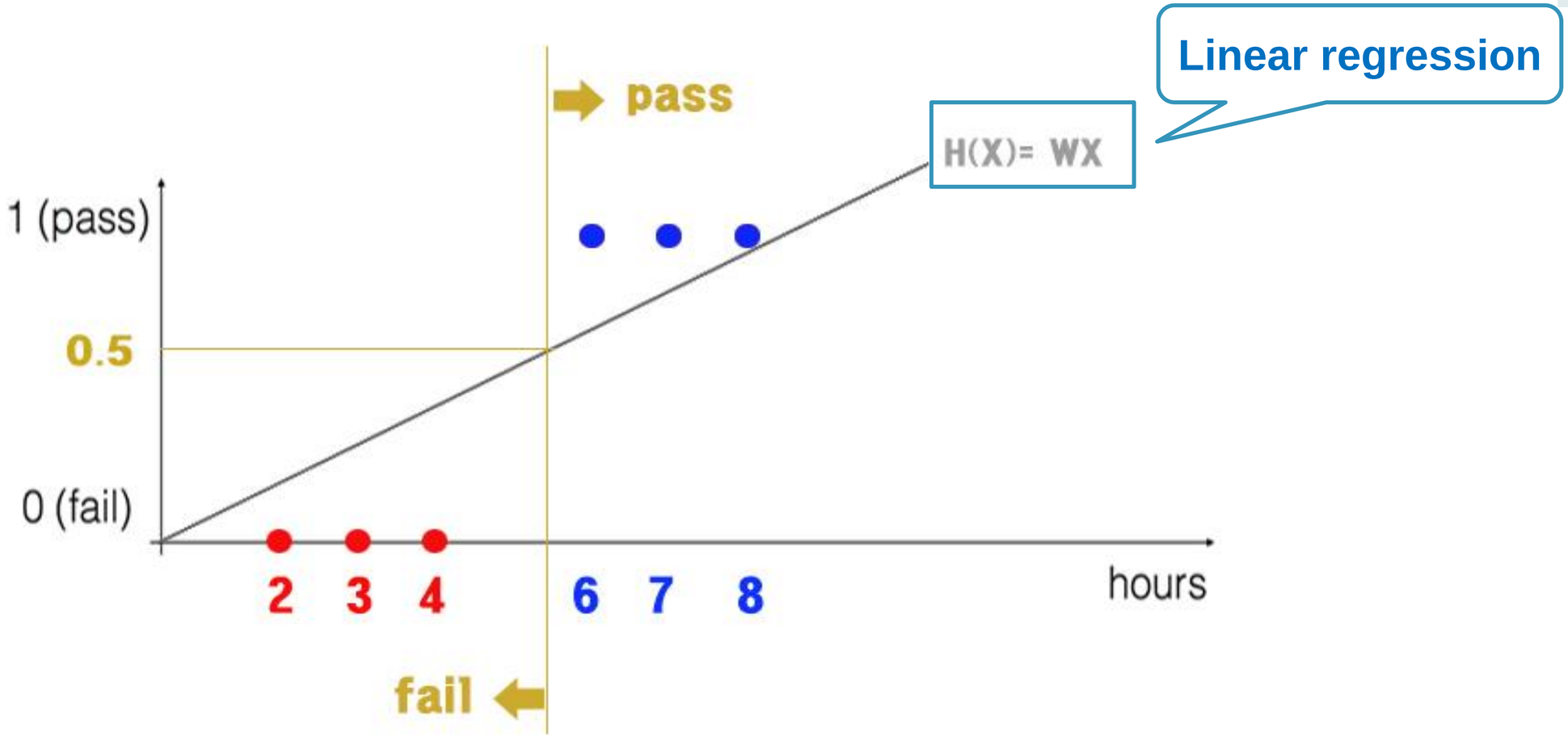
Finance


Finance
https://www.youtube.com/watch?v=BmkA1ZsG2P4
DWJI 17,499.10
SP500 2,025.51
NASDAQ 4,976.9
AAPL 107.71
GOOG 750.06
TSLA 234.24
AAPL
GOOG
BUY
TSLA

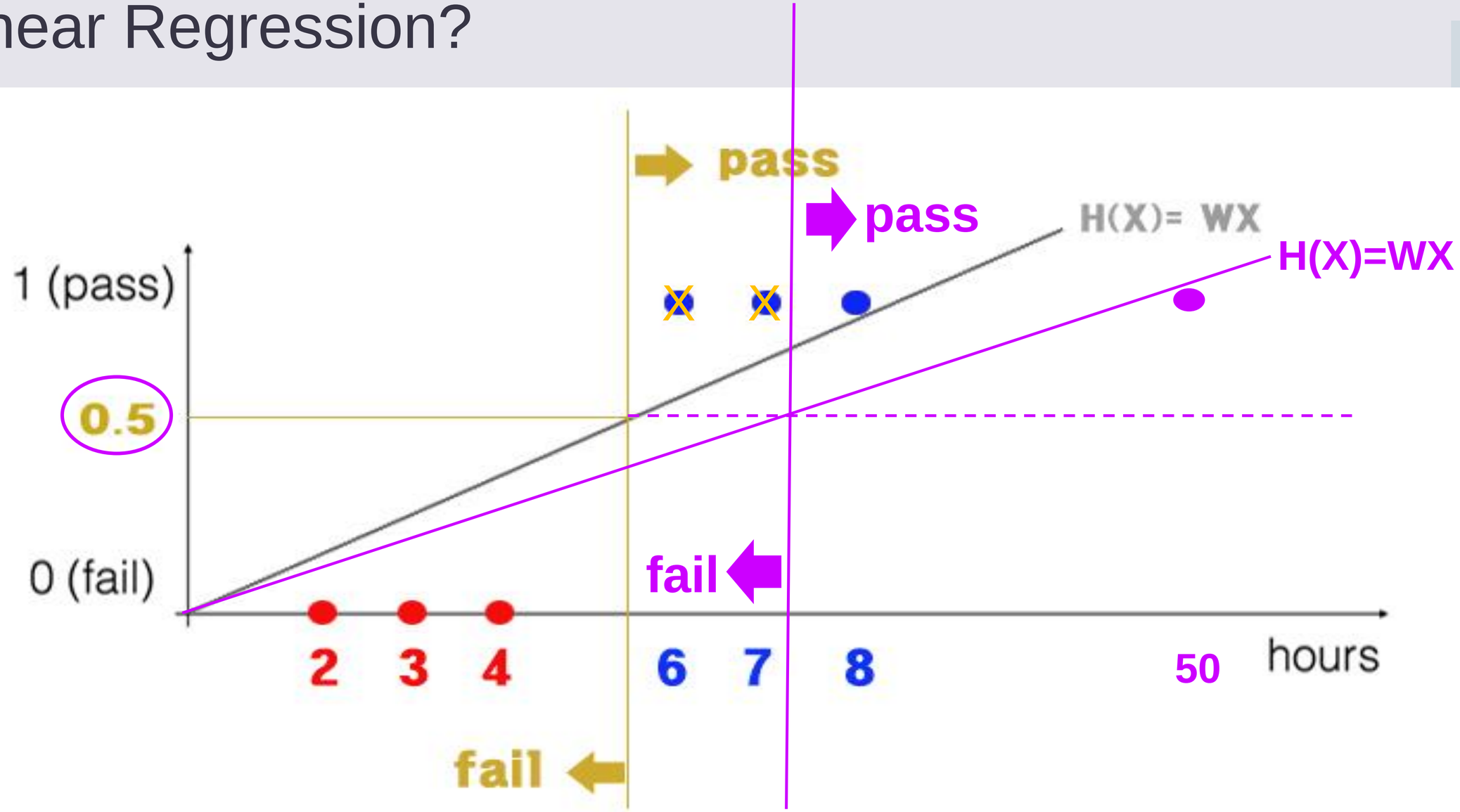
Pass(1)/Fail(0) based on study hours



Linear Regression?



Linear Regression?



Linear regression

- We know Y is 0 or 1

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

X = [0.1, 1.2, 1.5, 1.9] , W=0.5 , b = 0

0 < H(x) < 1

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

X = [100] , W=0.5 , b = 0

H(x) = 50

H(x) > 1

Logistic Hypothesis

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

Z

g (Z)

0~1

z can give values large than 1 or less than 0

Sigmoid

$$g(z) = \frac{1}{(1+e^{-z})}$$

Wx

$$g(z) = \frac{1}{(1+e^{-z})}$$

~~$H(x) = Wx$~~

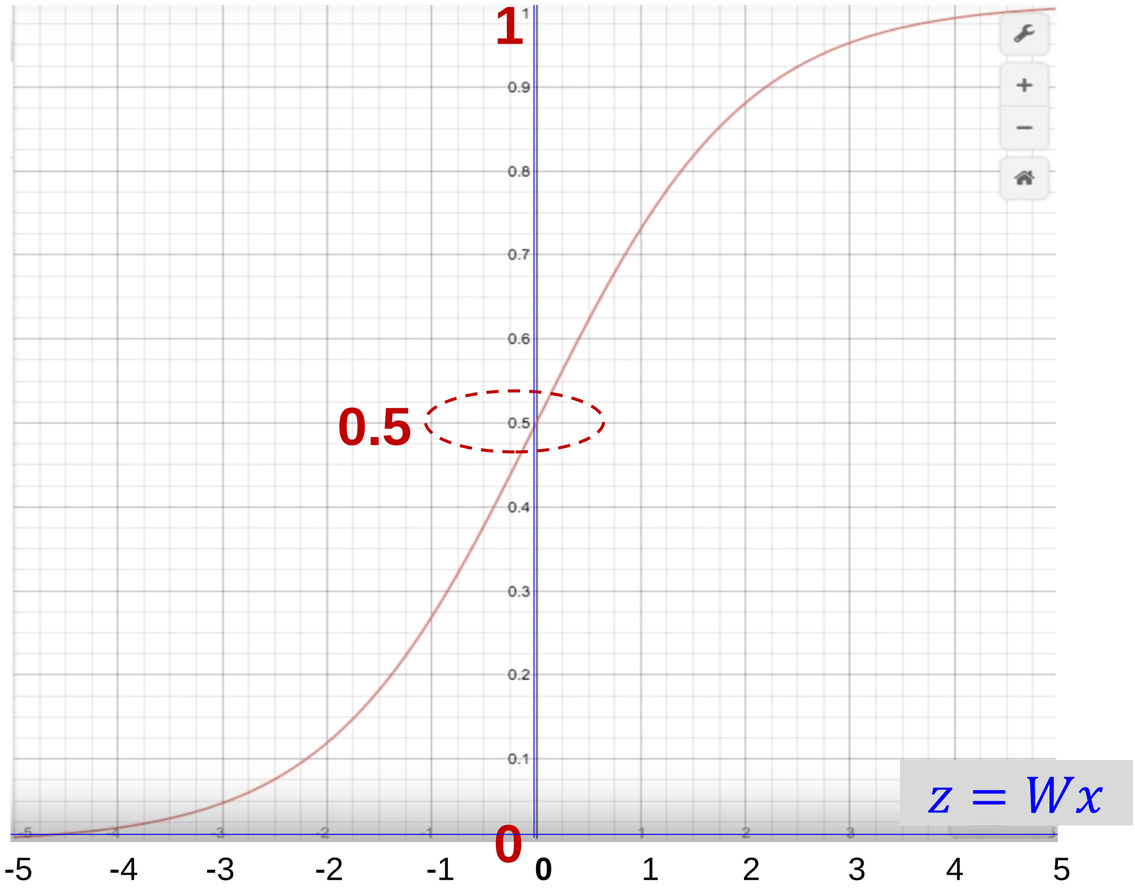
$z = Wx$

$\boldsymbol{H(x)} = g(z)$

- logistic function or Sigmoid function

Sigmoid : Curved in two directions, like the letter “S”

+							-
z	-z	exp(-z)	g(z)	g(z)	exp(-z)	-z	z
				0.5	1	0	0
1	-1	0.36788	0.73106	0.26894	2.71828	1	-1
2	-2	0.13534	0.88080	0.11920	7.38906	2	-2
3	-3	0.04979	0.95257	0.04743	20.08554	3	-3
4	-4	0.01832	0.98201	0.01799	54.59815	4	-4
5	-5	0.00674	0.99331	0.00669	148.41316	5	-5



Python (math.exp)

```
import math
z_plus=[]
for i in range (1,21) :  # 리스트에 1~20 저장
    z_plus.append(i)
print("z_plus", z_plus)

z_minus=[]
for j in range (-1,-21,-1) :  # 리스트에 -1 ~ -20 저장
    z_minus.append(j)
print("z_minus", z_minus)
```

```
exp_plus=[]
exp_minus=[]
for k in range (0,20) :  # k는 0~19 (20) 번 반복
    exp_plus.append(math.exp(z_plus[k]))
    exp_minus.append(math.exp(z_minus[k]))
print("\n1~20까지 exp 함수 출력 결과 ")
print(exp_plus)
print("\n-1 ~ -20까지 exp 함수 출력 결과 ")
print(exp_minus)
```

```
g_plus=[]
g_minus=[]

for z in range (0,20) :  #0~20 인덱스
    g_plus.append(1/(1+exp_plus[z]))
    g_minus.append(1/(1+exp_minus[z]) )

print("\n1~20까지 g(z) 함수 출력 결과 ")
print(g_plus)
print("\n-1 ~ -20까지 g(z) 함수 출력 결과 ")
print(g_minus)
print("\n")

for z in range (0,20) :  #0~20 인덱스
    print ('z : ',-(z_minus[z]), '-z : ',z_minus[z] , '\texp :', exp_minus[z] , '\tg(z) :', g_minus[z] ,  )
    print ('z : ',-(z_plus[z]), '-z : ',z_plus[z] , '\texp :', exp_plus[z] , '\tg(z) :', g_plus[z] ,  )
    print ("="*50)
```

Python (math.exp)

z_plus [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
z_minus [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16, -17, -18, -19, -20]

1~20까지 exp 함수 출력 결과
[2.718281828459045, 7.38905609893065, 20.085536923187668, 54.598150033144236, 148.4131591025766, 403.4287934927351, 1096.6331584284585, 2980.9579870417283, 8103.083927575384, 22026.465794806718, 59874.14171519782, 162754.79141900392, 442413.3920089205, 1202604.2841647768, 3269017.3724721107, 8886110.520507872, 24154952.7535753, 65659969.13733051, 178482300.96318725, 485165195.4097903]

-1 ~ -20까지 exp 함수 출력 결과
[0.36787944117144233, 0.1353352832366127, 0.049787068367863944, 0.01831563888873418, 0.006737946999085467, 0.0024787521766663585, 0.000911881965545162, 0.0003354626279025118 5, 0.00012340980408667956, 4.539929762484854e-05, 1.670170079024566e-05, 6.14421235332821e-06, 2.2603294069810542e-06, 8.315287191035679e-07, 3.059023205018258e-07, 1.125351747192591 2e-07, 4.139937718785167e-08, 1.522997974471263e-08, 5.602796437537268e-09, 2.061153622438558e-09]

1~20까지 g(z) 함수 출력 결과
[0.2689414213699951, 0.11920292202211755, 0.04742587317756678, 0.01798620996209156, 0.006692850924284855 4, 0.0024726231566347743, 0.0009110511944006454, 0.0003353501304664781, 0.00012339457598623172, 4.5397868702434395e-05, 1.670142184809518e-05, 6.144174602214718e-06, 2.260324297903574 6e-06, 8.315280276641321e-07, 3.059022692562 47e-07, 1.12535162055095e-07, 4.1399375473943306e-08, 1.522997951276035e-08, 5.602796406145941e-09, 2.061153618190203 7e-09]

-1 ~ -20까지 g(z) 함수 출력 결과
[0.7310585786300049, 0.8807970779778823, 0.9525741268224334, 0.9820137900379085, 0.9933071490757153, 0.9975273768433653, 0.9990889488055994, 0.9996646498695336, 0.9998766054240137, 0.9999546021312976, 0.999983298578152, 0.9999938558253978, 0.999997739675702, 0.9999991684719722, 0.999999694097773, 0.9999998874648379, 0.9999999586006244, 0.9999999847700205, 0.9999999943972036, 0.9999999979388463]

Python (math.exp)

```
z :  1 -z :  -1      exp : 0.36787944117144233    g(z) : 0.7310585786300049
z :  -1 -z :  1      exp : 2.718281828459045      g(z) : 0.2689414213699951
==================================================
z :  2 -z :  -2      exp : 0.1353352832366127     g(z) : 0.8807970779778823
z :  -2 -z :  2      exp : 7.38905609893065       g(z) : 0.11920292202211755
==================================================
z :  3 -z :  -3      exp : 0.049787068367863944   g(z) : 0.9525741268224334
z :  -3 -z :  3      exp : 20.085536923187668     g(z) : 0.04742587317756678
==================================================
z :  4 -z :  -4      exp : 0.01831563888873418    g(z) : 0.9820137900379085
z :  -4 -z :  4      exp : 54.598150033144236     g(z) : 0.01798620996209156
==================================================
z :  5 -z :  -5      exp : 0.006737946999085467   g(z) : 0.9933071490757153
z :  -5 -z :  5      exp : 148.4131591025766      g(z) : 0.006692850924284855
==================================================
z :  6 -z :  -6      exp : 0.0024787521766663585  g(z) : 0.9975273768433653
z :  -6 -z :  6      exp : 403.4287934927351      g(z) : 0.0024726231566347743
==================================================
z :  7 -z :  -7      exp : 0.0009118819655545162  g(z) : 0.9990889488055994
z :  -7 -z :  7      exp : 1096.6331584284585     g(z) : 0.0009110511944006454
==================================================
z :  8 -z :  -8      exp : 0.00033546262790251185 g(z) : 0.9996646498695336
z :  -8 -z :  8      exp : 2980.9579870417283     g(z) : 0.0003353501304664781
==================================================
z :  9 -z :  -9      exp : 0.00012340980408667956 g(z) : 0.9998766054240137
z :  -9 -z :  9      exp : 8103.083927575384      g(z) : 0.00012339457598623172
==================================================
z :  10 -z :  -10    exp : 4.5399929762484854e-05 g(z) : 0.9999546021312976
z :  -10 -z :  10    exp : 22026.465794806718     g(z) : 4.5397868702434395e-05
==================================================
```

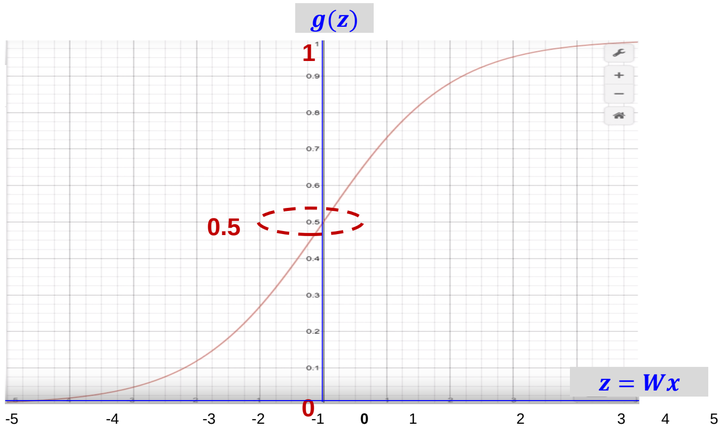
```
z :  11 -z :  -11    exp : 1.670170079024566e-05  g(z) : 0.999983298578152
z :  -11 -z :  11    exp : 59874.14171519782      g(z) : 1.670142184809518e-05
==================================================
z :  12 -z :  -12    exp : 6.14421235332821e-06   g(z) : 0.9999938558253978
z :  -12 -z :  12    exp : 162754.79141900392     g(z) : 6.144174602214718e-06
==================================================
z :  13 -z :  -13    exp : 2.2603294069810542e-06 g(z) : 0.999997739675702
z :  -13 -z :  13    exp : 442413.3920089205      g(z) : 2.2603242979035746e-06
==================================================
z :  14 -z :  -14    exp : 8.315287191035679e-07  g(z) : 0.9999991684719722
z :  -14 -z :  14    exp : 1202604.2841647768     g(z) : 8.315280276641321e-07
==================================================
z :  15 -z :  -15    exp : 3.059023205018258e-07  g(z) : 0.999999694097773
z :  -15 -z :  15    exp : 3269017.3724721107     g(z) : 3.059022269256247e-07
==================================================
z :  16 -z :  -16    exp : 1.1253517471925912e-07 g(z) : 0.9999998874648379
z :  -16 -z :  16    exp : 8886110.520507872      g(z) : 1.12535162055095e-07
==================================================
z :  17 -z :  -17    exp : 4.139937718785167e-08  g(z) : 0.9999999586006244
z :  -17 -z :  17    exp : 24154952.7535753       g(z) : 4.1399375473943306e-08
==================================================
z :  18 -z :  -18    exp : 1.522997974471263e-08  g(z) : 0.9999999847700205
z :  -18 -z :  18    exp : 65659969.13733051      g(z) : 1.522997951276035e-08
==================================================
z :  19 -z :  -19    exp : 5.602796437537268e-09  g(z) : 0.9999999943972036
z :  -19 -z :  19    exp : 178482300.96318725     g(z) : 5.602796406145941e-09
==================================================
z :  20 -z :  -20    exp : 2.061153622438558e-09  g(z) : 0.9999999979388463
z :  -20 -z :  20    exp : 485165195.4097903      g(z) : 2.0611536181902037e-09
==================================================
```

Sigmoid

+							-
z	-z	exp	g(z)	g(z)	exp	-z	z
				0.5	1	0	0
1	-1	0.36788	0.73106	0.26894	2.72	1	-1
2	-2	0.13534	0.88080	0.11920	7.39	2	-2
3	-3	0.04979	0.95257	0.04743	20.09	3	-3
4	-4	0.01832	0.98201	0.01799	54.60	4	-4
5	-5	0.00674	0.99331	0.00669	148.41	5	-5
6	-6	0.00248	0.99753	0.00247	403.43	6	-6
7	-7	0.00091	0.99909	0.00091	1096.63	7	-7
8	-8	0.00034	0.99966	0.00034	2980.96	8	-8
9	-9	0.00012	0.99988	0.00012	8103.08	9	-9
10	-10	0.00005	0.99995	0.00005	22026.47	10	-10
11	-11	0.00002	0.99998	0.00002	59874.14	11	-11
12	-12	0.00001	0.99999	0.00001	162754.79	12	-12
13	-13	0.00000	1.00000	0.00000	442413.39	13	-13
14	-14	0.00000	1.00000	0.00000	1202604.28	14	-14
15	-15	0.00000	1.00000	0.00000	3269017.37	15	-15
16	-16	0.00000	1.00000	0.00000	8886110.52	16	-16
17	-17	0.00000	1.00000	0.00000	24154952.75	17	-17
18	-18	0.00000	1.00000	0.00000	65659969.14	18	-18
19	-19	0.00000	1.00000	0.00000	178482300.96	19	-19
20	-20	0.00000	1.00000	0.00000	485165195.41	20	-20
21	-21	0.00000	1.00000	0.00000	1318815734.48	21	-21
22	-22	0.00000	1.00000	0.00000	3584912846.13	22	-22
23	-23	0.00000	1.00000	0.00000	9744803446.25	23	-23
24	-24	0.00000	1.00000	0.00000	26489122129.84	24	-24
25	-25	0.00000	1.00000	0.00000	72004899337.39	25	-25

0~1

$g(Z)$



Logistic Classification Hypothesis

$$H(x) = Wx$$

$$z = Wx$$

$$z = Wx$$

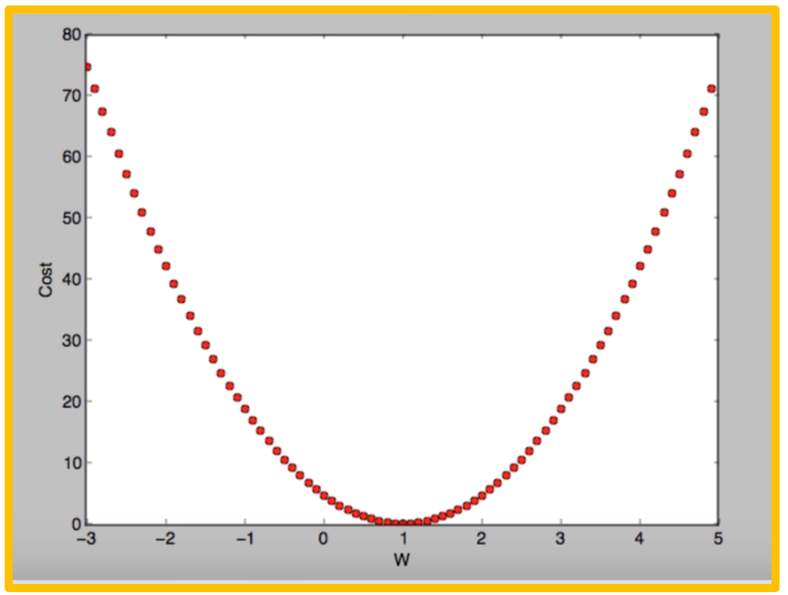
$$H(x) = g(z)$$

$$g(z) = \frac{1}{(1 + e^{-z})}$$

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

Cost

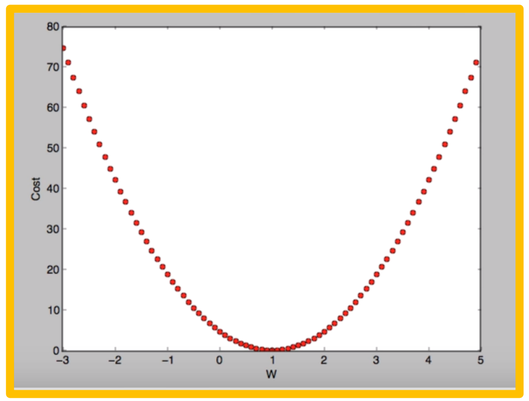
$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$ when $H(x) = Wx + b$



Cost function

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

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



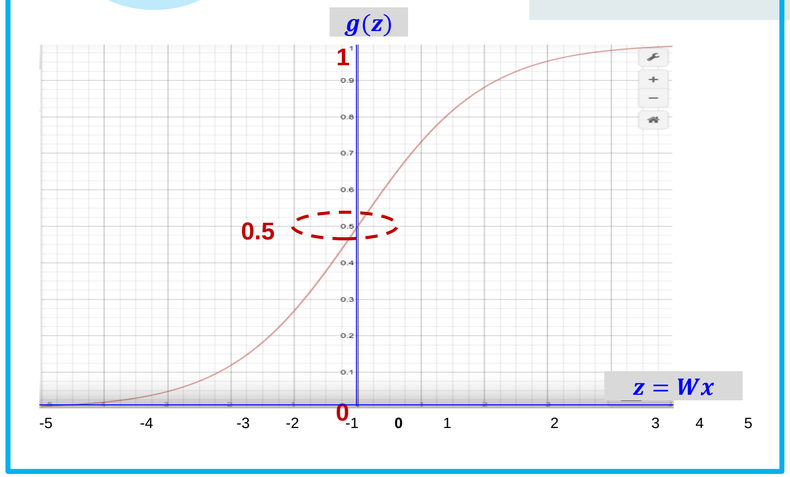
$$g(z) = \frac{1}{(1+e^{-z})}$$

Wx

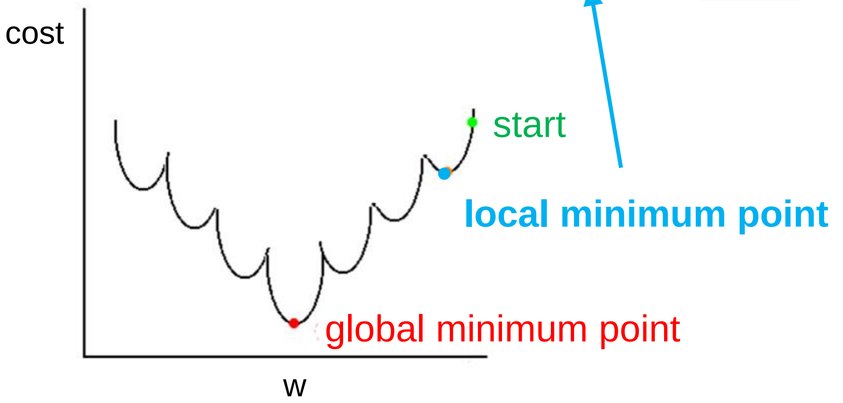
$$H(x) = \frac{1}{1+e^{-W^T X}}$$

Gradient decent algorithm

$H(x)$



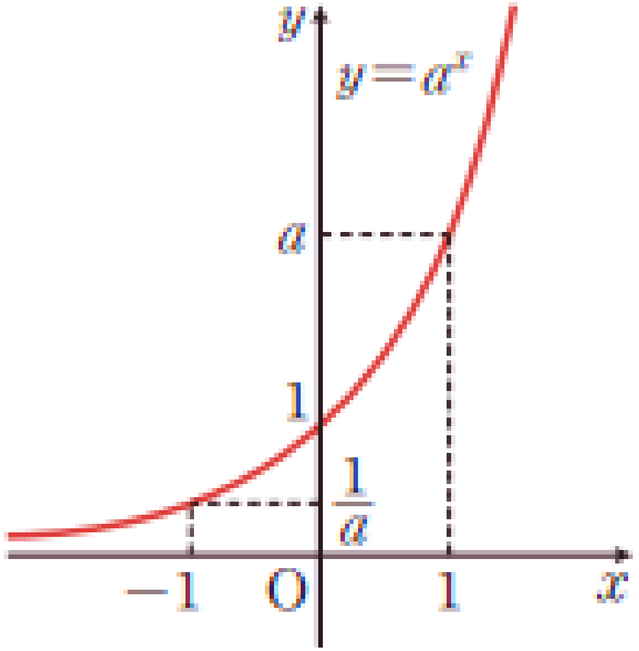
sigmoid ?



Exponential function vs. Log function

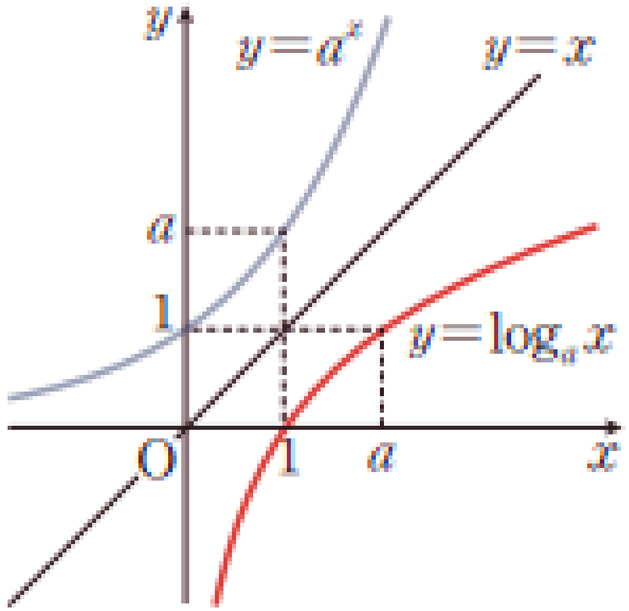
일반적으로 지수함수 $y=a^x\ (a>0,\ a\neq 1)$의 그래프는 a의 값에 따라 다음과 같다.

$a>1$



일반적으로 로그함수 $y=\log_a x\ (a>0,\ a\neq 1)$는 지수함수 $y=a^x$의 역함수이므로 이들의 그래프는 직선 $y=x$에 대하여 대칭이다.

$a>1$



New cost function for logistic classification

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

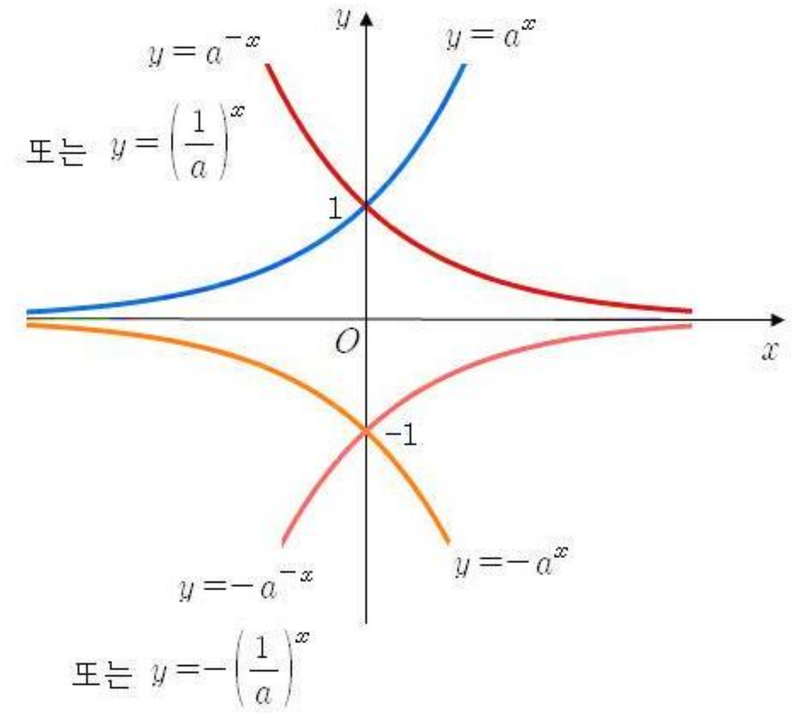
$$H(X) = \frac{1}{1+e^{-W^T X}}$$

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

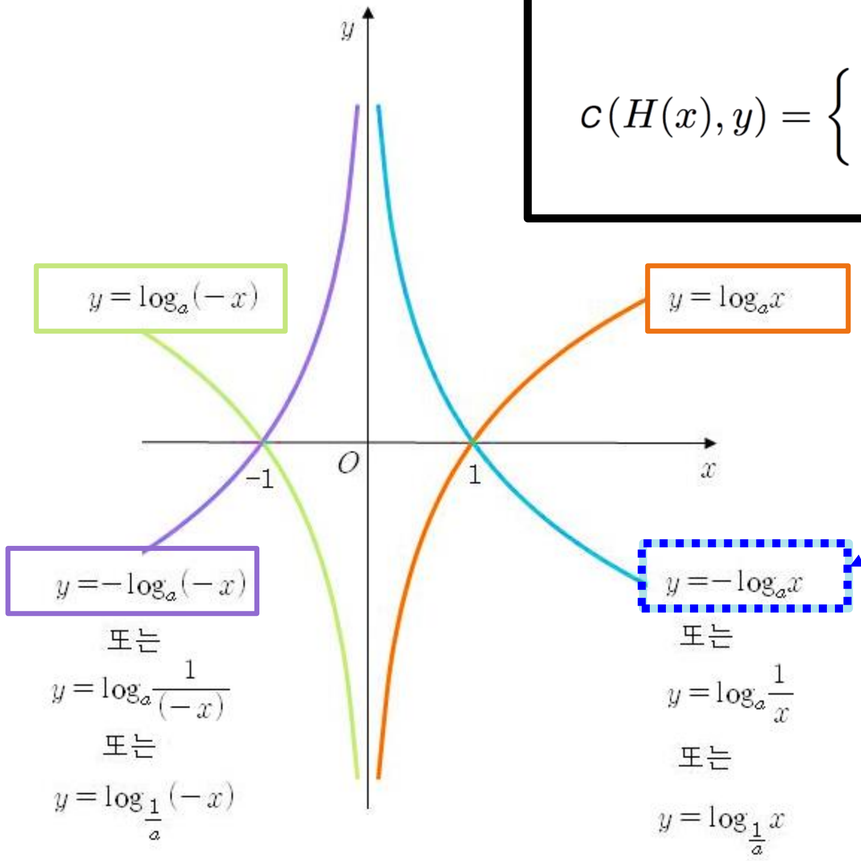
$$c(H(x), y) = \begin{cases} -log(H(x)) & y = 1 \\ -log(1 - H(x)) & y = 0 \end{cases}$$

New cost function for logistic classification

▶ 지수함수



▶ 로그함수



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

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

Understanding cost function

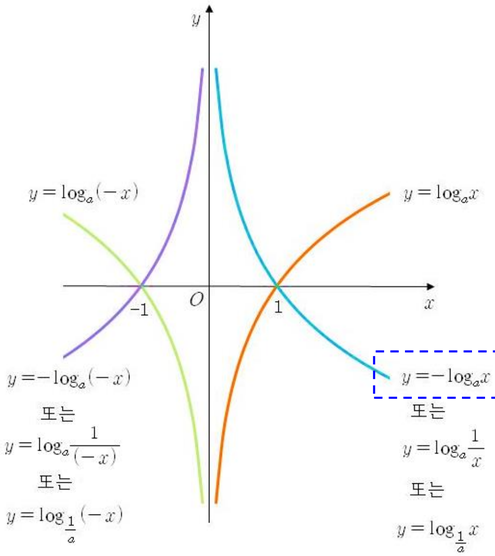
1

$$g(z) = \frac{1}{(1+e^{-z})}$$

$$z = Wx$$

$$H(x) = g(z)$$

$H(x)$ = **1 or 0**



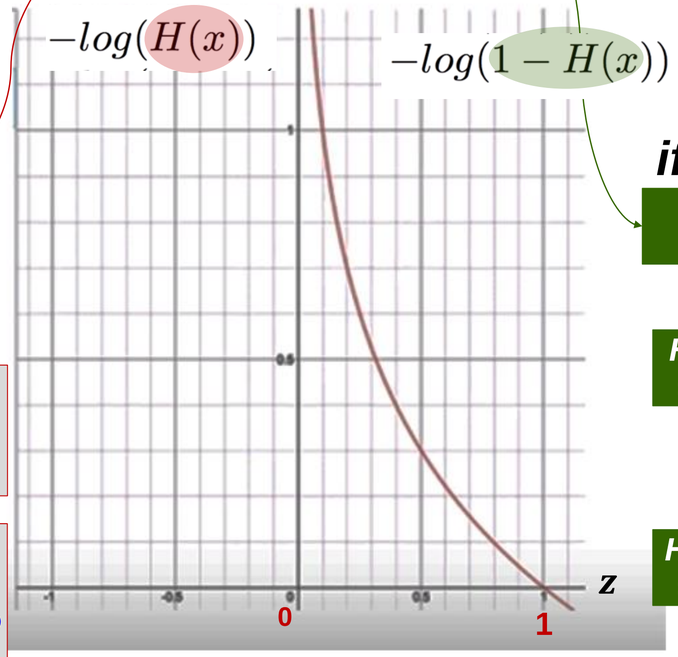
$-log(H(x))$

2

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

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

cost



if

y=1

H(x) = 1

-log(H(x))
= -log(1)
= 0

H(x) = 1
cost = 0

H(x) =0

-log(H(x))
= -log(0)
= ∞

H(x) = 0
cost = ∞

if

y=0

H(x) =1

-log(1-H(x))
= -log(1-1)
= -log(0)
= ∞

H(x) = 1
cost = ∞

H(x) =0

-log(1-H(x))
= -log(1-0)
= -log(1)
= 0

H(x) = 0
cost = 0

Cost function

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

if y=0

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

if y=1

$C = -1 * log(1-H(x))$

$C = -log(H(x))$

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

y=0 이면 생략 가능

y=1 이면 생략 가능

Minimize cost - Gradient decent algorithm

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

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

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

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

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

Gradient decent algorithm

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

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

```
# 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)
```