Deep Learning

Logistic Regression(binary classification)

Yoon Joong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

1. Binary classification
    1. examples
    2. 이진분류를 위한 logistic regression
    3. Logistic(sigmoid) function
    4. Loss function
        1. Mean square function
        2. Cross-entropy function
2. Examples
    1. 2-class logistic regression
    2. 2-class logistic regression with L1 and L2 regularization
    3. Diabetes classification

1. Binary classification

- Multivariable Regression 데이터 생성

- 모델 생성
  - Multivariable linear regression
    - H=XW+B
  - Loss: mean square error
  - Optimizer : adam

- 학습
  - W,B?
  - Fit

- 검증

```
# Linear Regression
#data
X=np.array([[10,5],
            [9,5],
            [3,2]])
Y=np.array([[90],
            [80],
            [50]])

#linear regression model 생성
model=Sequential()
model.add(Dense(1,
    activation=‘linear’,     #Linear regression
    input_dim=2))
model.compile(
    loss=‘mse’,              #mean square error
    optimizer= ' adam’)      #gradient descent optimizer
#학습
model.fit(X,Y,epochs=2000,verbose=1)

#검증
p=model.predict(np.array([[90, 90, 90]]))   # 검증예측값계산 [[178.51509]]
print(p)
```

X1 (hour)	x2 (attendance)	Y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

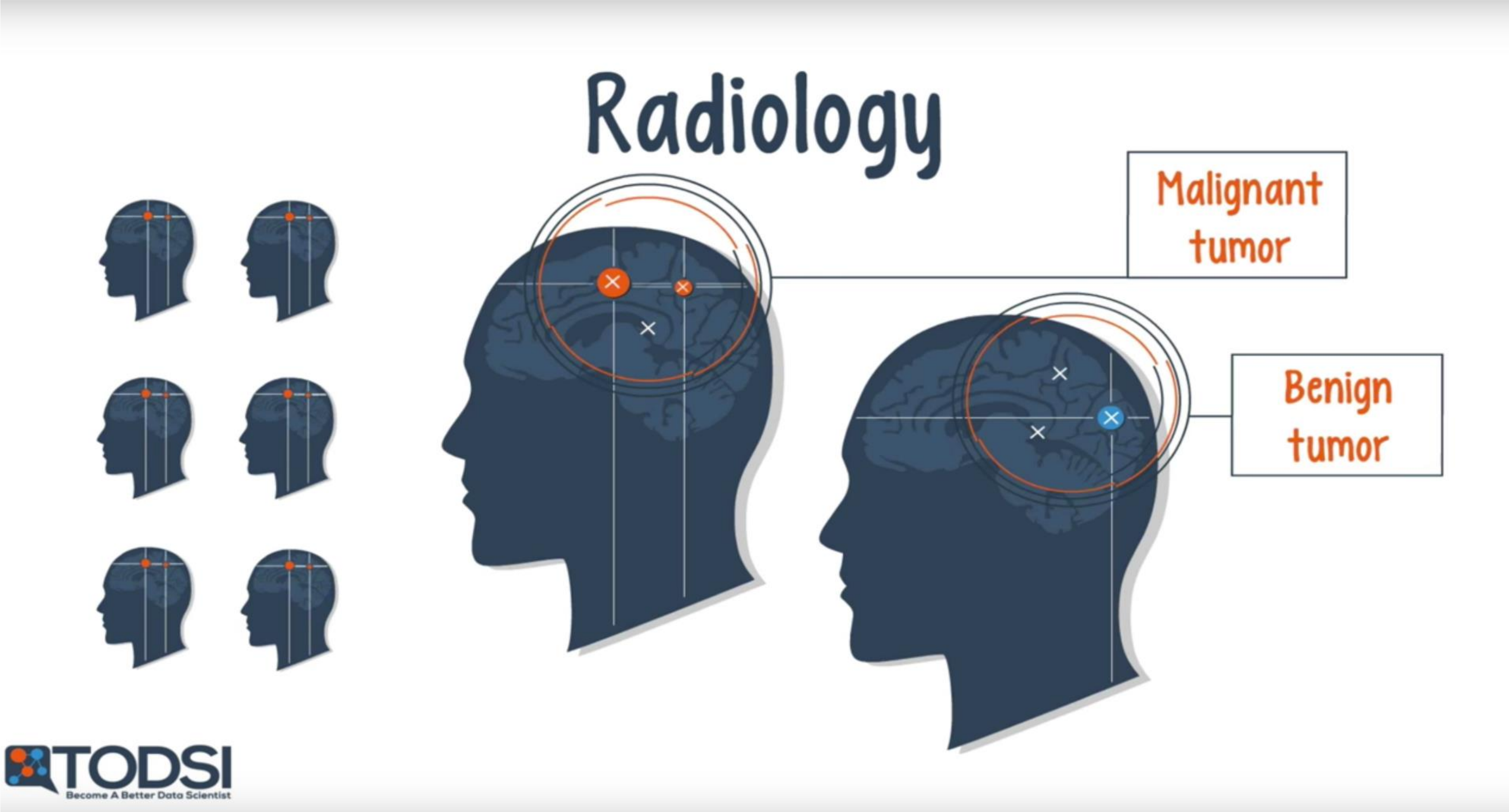
예측 값이 실수가 아니라 2인 경우 ?

1. Binary Classification

- Binary Classification
  - Spam Email Detection: Spam or Ham
  - Facebook feed: show or hide
    - Like pattern => timeline
  - Credit Card Fraudulent Transaction detection: legitimate/fraud

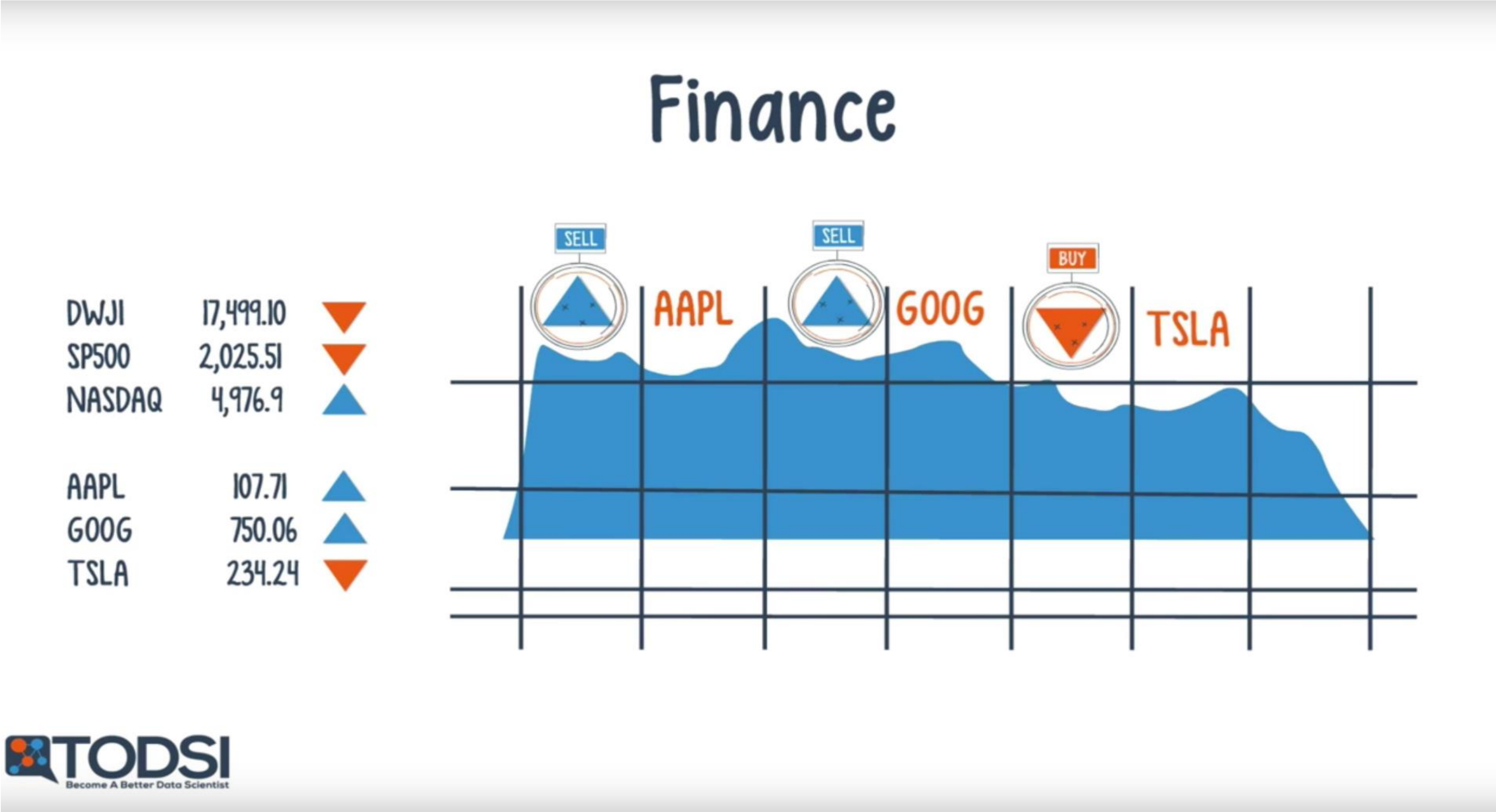
- 0,1 Encoding
  - Spam Email Detection: Spam(1) or Ham(0)
  - Facebook feed: show(1) or hide(0)
    - Like pattern => timeline
  - Credit Card Fraudulent Transaction detection: legitimate(1)/fraud(0)

1.1 Examples of logistic classifier



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

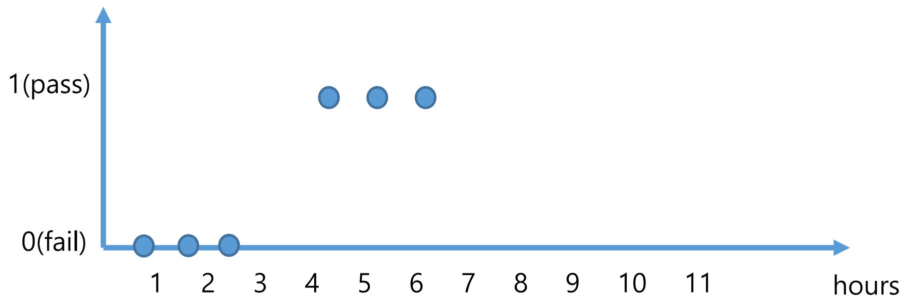
1.1 Examples of logistic classifier



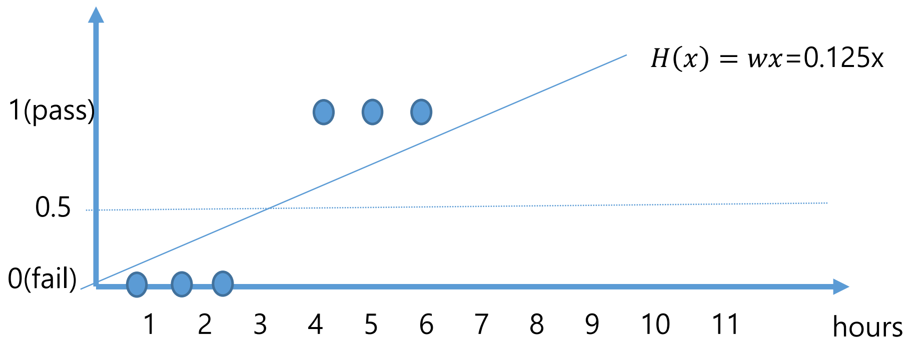
https://www.youtube.com/watch?v=BmkA1ZsG2P4

1.2 이진분류를 위한 logistic regression

- Pass(1)/Fail(0) based on study hours from passing or failing
  - 학습시간과 Pass와 Fail의 산점도



  - 선형회귀모델($H(x) = wx + b$) 로 분류가능한가?

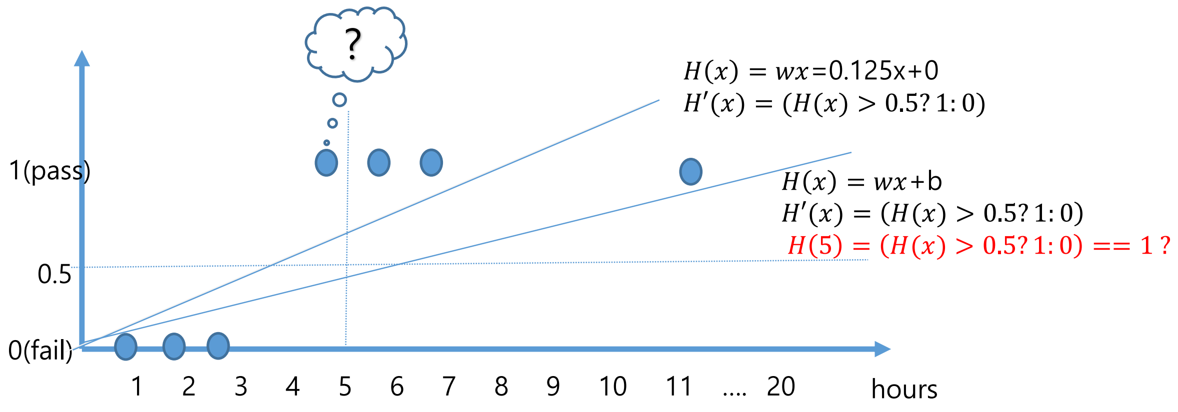


x	y
1	0
2	0
3	0
5	1
6	1
7	1

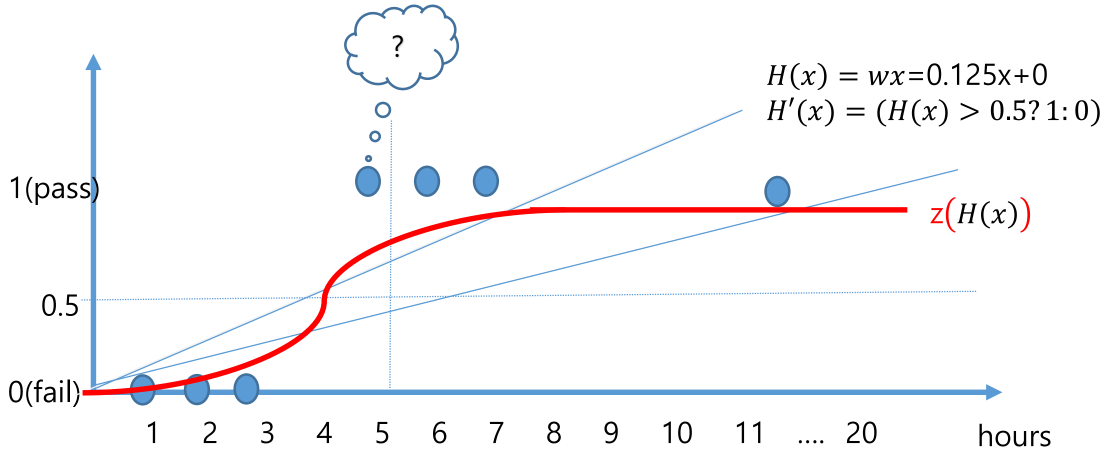
1.2 이진분류를 위한 logistic regression

- 선형회귀모델($H(x) = wx + b$)로 분류가능한가?

x	y
1	0
2	0
3	0
5	1
6	1
7	1



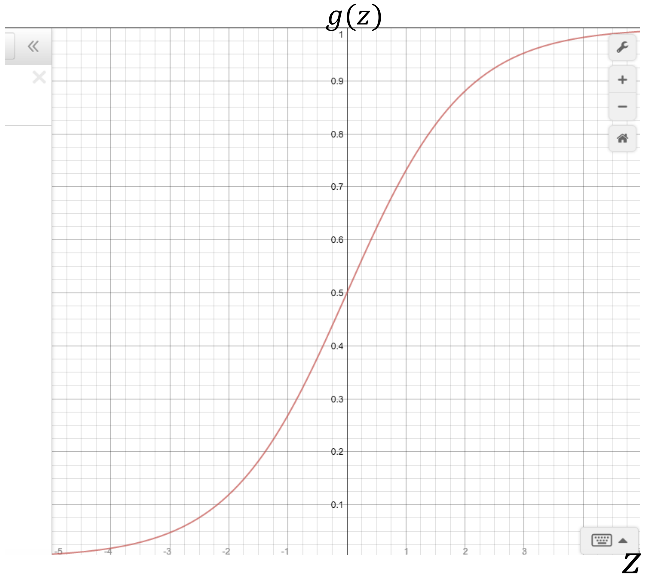
Other Hypothesis function is required



$H(5) = (H(x) > 0.5?\,1{:}\,0) = 1$
새로운 예측(가설, $activation$) 함수 필요

1.3 Logistic Hypothesis - logistic function

- logistic function
  - sigmoid function.
  - sigmoid :
    Curved in two directions,
    like the letter "S",
    or the Greek ς (sigma).
  - $g(z) = \frac{1}{1+e^{-z}}$



- linear hypothesis (linear regression)
  - $H_L(x) = wx + b$
- logistic hypothesis (logistic regression)
  - $H(x) = g(H_L(x))$

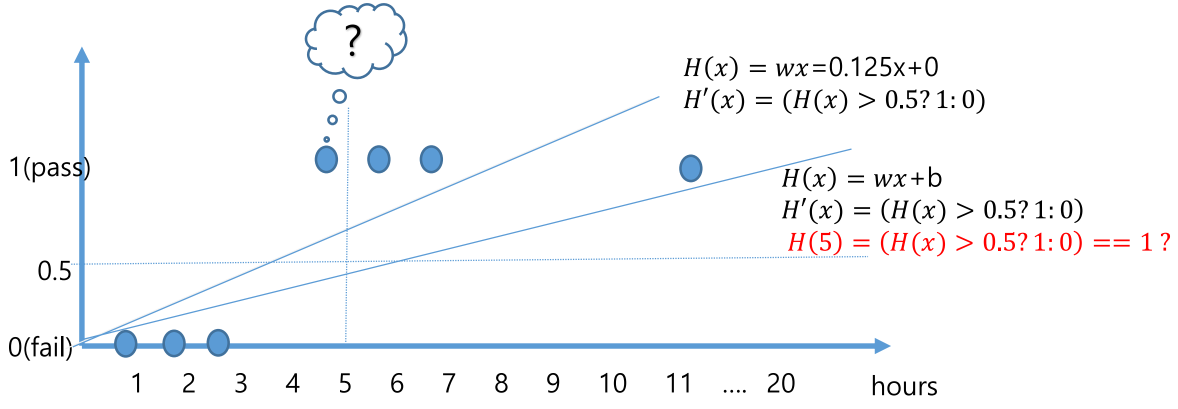
    $= g(wx + b)$

    $= \frac{1}{1-e^{-(wx+b)}}$

1.3 Logistic Hypothesis - logistic function

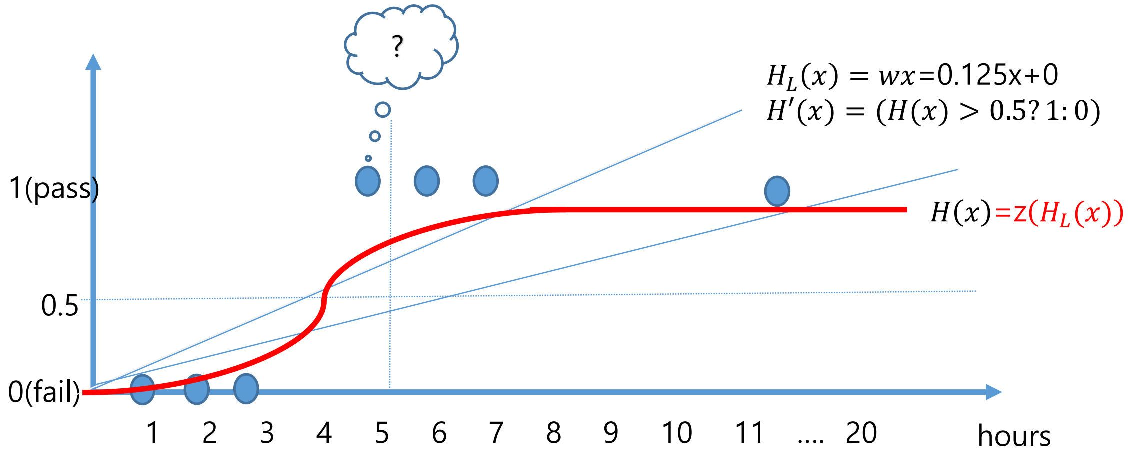
- 로지스틱회귀모델$(H(x) = z(H_L(x)) = z(wx + b))$로 분류 가능하다.

x	y
1	0
2	0
3	0
5	1
6	1
7	1



Linear regression model

```
model=Sequential()
model.add(Dense(1,
            activation='linear',
            input_dim=1))
```



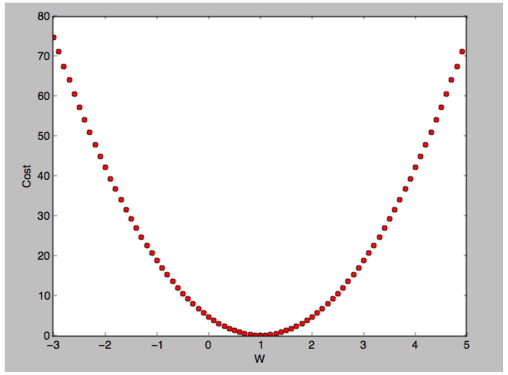
Logistic regression model

```
model=Sequential()
model.add(Dense(1,
            activation='sigmoid',
            input_dim=1))
```

1.4 Loss(cost) function of logistic regression model

- Loss function of Linear regression
  - $H(x) = wx + b \quad for\ \{x_i, y_i\}, i = 1..m$
  - loss function of mean square error
    - $L(\{x_i\},\{y_i\}|w,b)$=L(X,Y)

$$= \frac{1}{m}\sum_{i=1}^{m}(H(x_I) - y_i)^2$$

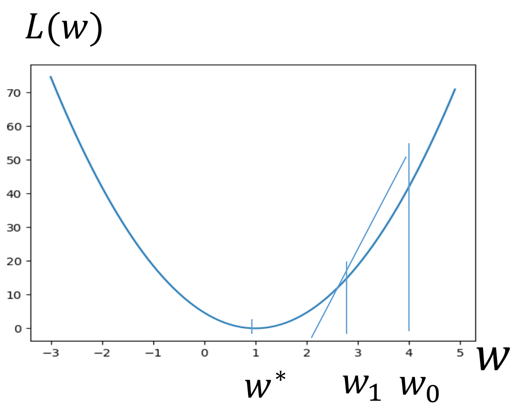


1.4.1 Mean square function for loss function

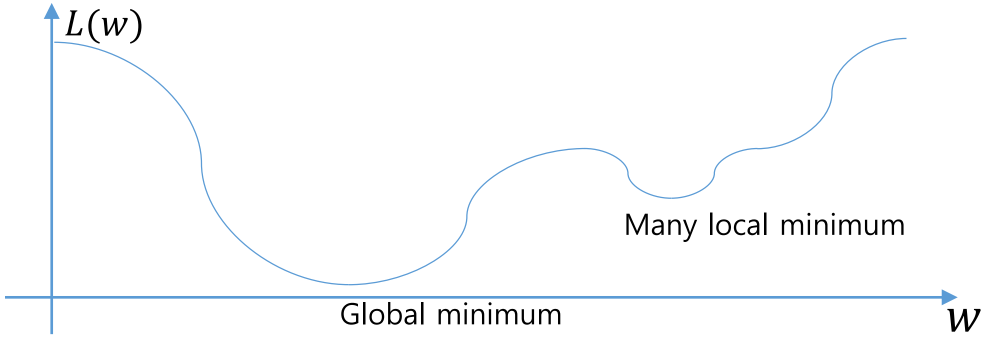
- Mean square function for linear regression loss
  - $L(w,b) = \frac{1}{m}\sum_{i=1}^{m}(H_L(x_i) - y_i)^2$
  - $H_L(x) = wx + b$
  -

- Mean square function for logistic regression
  - $L(w,b) = \frac{1}{m}\sum_{i=1}^{m}(H_S(x_i) - y_i)^2$
  - $H_s(x) = z(H_L(x)) = \frac{1}{1+e^{-(wx+b)}}$

Train : gradient descent algorithm is ok



Train : gradient descent algorithm ? X

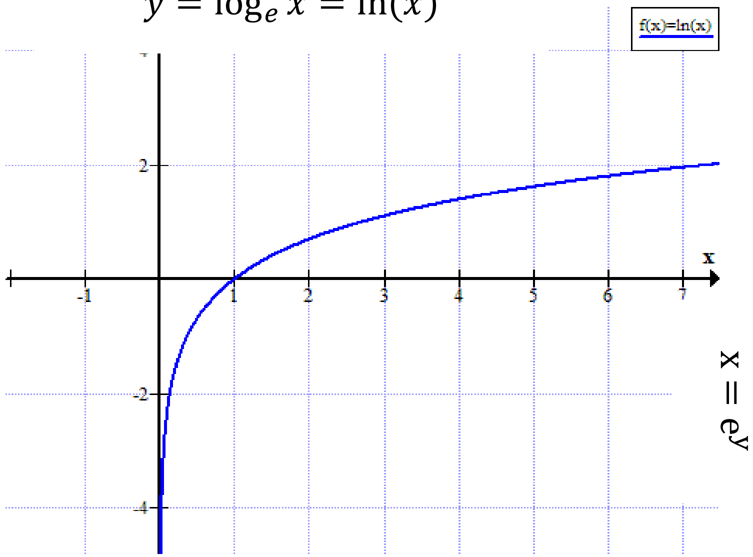


Other loss function ?

1.4.2 Cross-entropy function for logistic classifier

- Cross-entropy function
  - $ce(\bar{y}, y) = \begin{cases} -\log(\bar{y}) & : y = 1 \\ -\log(1-\bar{y}) & : y = 0 \end{cases}$
    - $\bar{y} : [-\infty, \infty]$
    - $y \in \{0,1\}$
- Loss function for logistic classifier
  - $L(w, b) = \frac{1}{N} \sum_{i=1}^{N} ce(\bar{y}, y_i)$
  - $\bar{y} = H_S(x) = z(H_L(x_i)) = \frac{1}{1+e^{-(wx+b)}}$
  - $H_L(x) = wx + b$

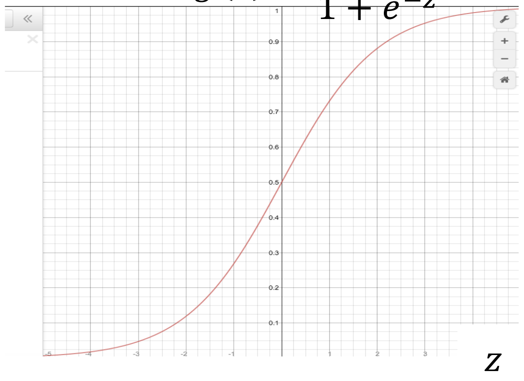
$e^y = x$
$y = \log_e x = \ln(x)$
$e \approx 2.71828183$: Euler number

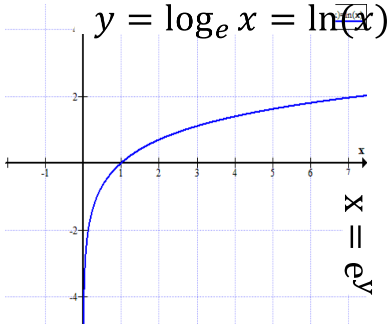


1.4.2 Cross-entropy function for logistic classifier

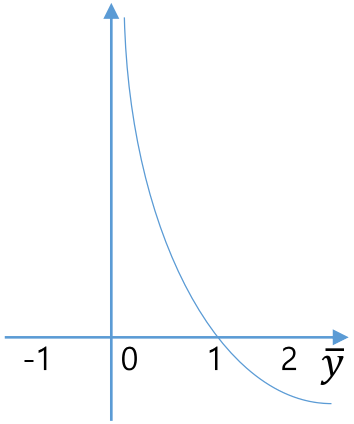
- Cross-entropy function
  - $ce(\bar{y}, y) = \begin{cases} -\log(\bar{y}) & : y = 1 \\ -\log(1-\bar{y}) & : y = 0 \end{cases}$
    - $\bar{y}$: [0,1]
    - $y \in \{0,1\}$
- Loss function for logistic classifier
  - $L(w,b) = \frac{1}{N}\sum_{i=1}^{N} ce(\bar{y}, y_i)$
  - $\bar{y} = H_S(x_i) = z(H_L(x_i)) = \frac{1}{1+e^{-(wx+b)}}$
  - $H_L(x) = wx + b$

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

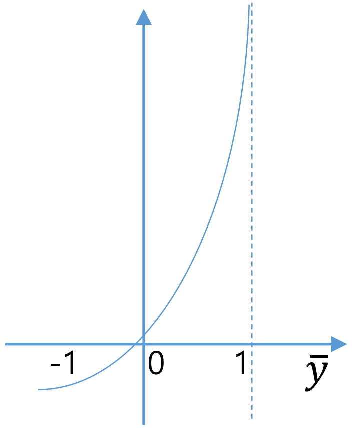




$y = 1$
$c(\bar{y}, y) = -\log_e(\bar{y})$



$y = 0$
$c(\bar{y}, y) = -\log_e(1-\bar{y})$

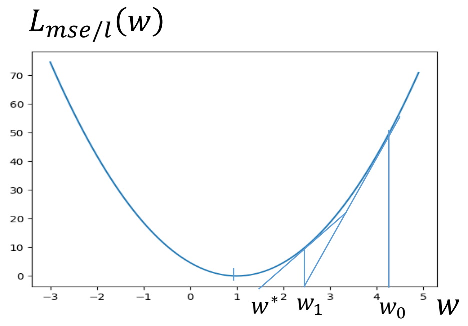


$L(\{x_i\},\{y_i\}) = \begin{cases} \Rightarrow 0, \bar{y} \Rightarrow 1, y = 1 \\ \Rightarrow \infty, \bar{y} \Rightarrow 0, y = 1 \end{cases}$

$L(\{x_i\},\{y_i\}) = \begin{cases} \Rightarrow \infty, \bar{y} \Rightarrow 1, y = 0 \\ \Rightarrow 0, \bar{y} \Rightarrow 0, y = 0 \end{cases}$

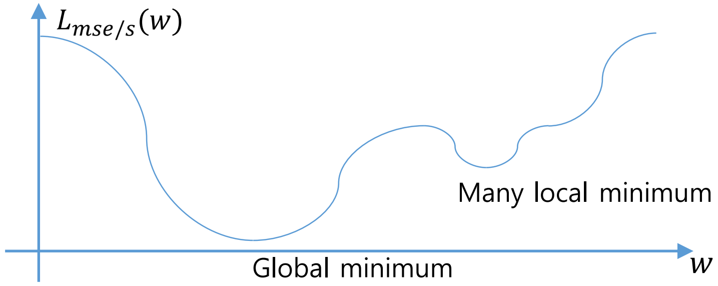
1.4.2 Cross-entropy function for logistic classifier

- Mean square function for linear regression loss
  - $L_{mse/l}(w,b|X,Y) = \frac{1}{m}\sum_{i=1}^{m}(H_L(x_i) - y_i)^2$
  - $H_L(x) = wx + b$



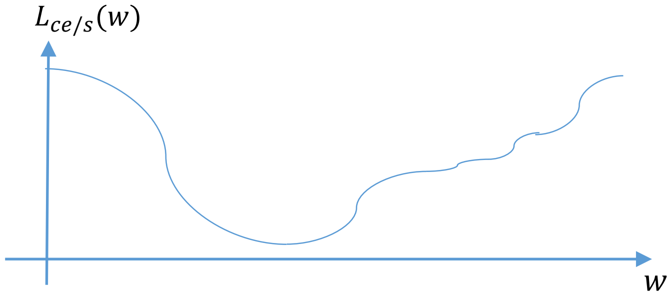
Train : gradient descent algorithm is ok

- Mean square function for logistic regression
  - $L_{mse/s}(w,b|X,Y) = \frac{1}{m}\sum_{i=1}^{m}(H_S(x_i) - y_i)^2$
  - $H_s(x) = z\big(H_L(x)\big) = \frac{1}{1+e^{-(wx+b)}}$



Train : gradient descent algorithm ? X

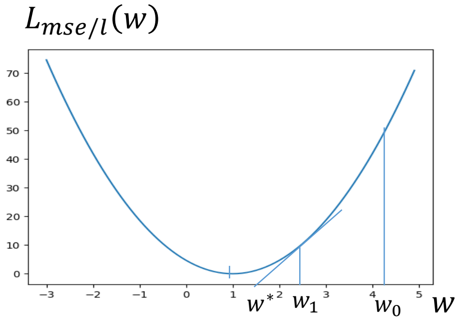
- Cross entropy function for logistic regression
  - $L_{ce/s}(w,b) = \frac{1}{N}\sum_{i=1}^{N} ce(\bar{y}, y_i)$
  - $\bar{y} = H_S(x_i) = z(H_L(x_i)) = \frac{1}{1+e^{-(wx+b)}}$
  - $H_L(x) = wx + b$



gradient descent algorithm is applicable

1.4.2 Cross-entropy function for logistic classifier

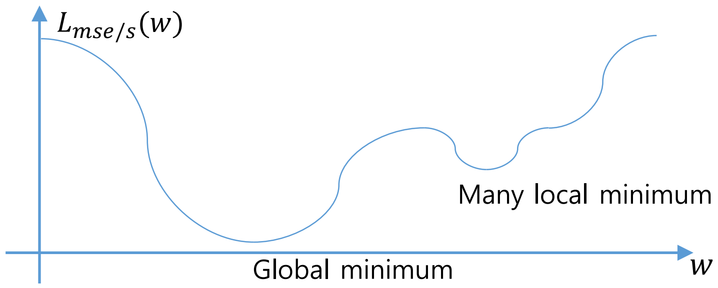
- Mean square function for linear regression loss
  - $L_{mse/l}(w,b|X,Y) = \frac{1}{m}\sum_{i=1}^{m}(H_L(x_i) - y_i)^2$
  - $H_L(x) = wx + b$



Train : gradient descent algorithm is ok

```
#model 정의
model=Sequential()
model.add(Dense(1,
          activation='linear' , #logistic regression
          input_dim=2))
Model.compile(
          loss='mse',    #mean square error
          optimizer= ' adam') #gradient descent optimizer
```

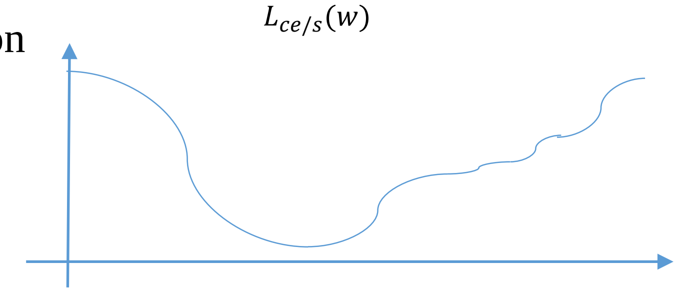
- Mean square function for logistic regression
  - $L_{mse/s}(w,b|X,Y) = \frac{1}{m}\sum_{i=1}^{m}(H_S(x_i) - y_i)^2$
  - $H_s(x) = z(H_L(x)) = \frac{1}{1+e^{-(wx+b)}}$



Train : gradient descent algorithm ? X

```
#model 정의
model=Sequential()
model.add(Dense(1,
          activation='sigmoid' , #logistic regression
          input_dim=2))
Model.compile(
          loss='mse',    #mean square error
          optimizer= ' adam') #gradient descent optimizer
```

- Cross entropy function for logistic regression
  - $L_{ce/s}(w,b) = \frac{1}{N}\sum_{i=1}^{N} ce(\bar{y}, y_i)$
  - $\bar{y} = H_S(x_i) = z(H_L(x_i)) = \frac{1}{1+e^{-(wx+b)}}$
  - $H_L(x) = wx + b$



gradient descent algorithm is applicable

```
#model 정의
model=Sequential()
model.add(Dense(1,
       activation='sigmoid' , #logistic regression
       input_dim=2))
Model.compile(
       loss='binary_crossentropy', #cross entropy
       optimizer= ' adam')    #gradient descent optimizer
```

Example 1. an example of Logistic Regression (binary classifier)

- Data 준비
  - Train data and evluation data
- Logistic regression model 정의
  - activation= sigmoid
  - loss = binary crossentropy
- 학습
  - model.fit
- 검증
  - model.predict

```
#linear model
X      =np.array([[1.1, 2.3], [2.0, 3.6]]) ;Y    =np.array([[10,25]])
X_val=np.array([[1.3, 1.8]]);        ;       y_val=np.array([[10]])
model=Sequential()
model.add(Dense(1,
         activation='linear' , #logistic regression
        input_dim=2))
Model.compile(
        loss='mse',    #mean square error
        optimizer= ' adam') #gradient descent optimizer
p=model.predict(np.array([[1.1, 1.7]]))#[[11.33]]
```

```
#logistic mode
X    =np.array([[1, 2], [2, 3]]) ;Y    =np.array([[0],[1]])
X_val=np.array([[1, 1]]);         ; y_val=np.array([[0]])
model=Sequential()
model.add(Dense(1,
      activation='sigmoid' , #logistic regression
      input_dim=2))
model.compile(
      loss='binary_crossentropy', #cross entropy
      optimizer= ' adam')          #gradient descent optimizer
y_hat     =model.predict(X)          #[[0.3] [0.4]]
Acc       =model.evaluate(X_val, y_val, verbose=0)[1] #0.7 evaluate
accuracy of validation dataset
```

Example 1. an example of Logistic Regression (binary classifier)

- Dataset 생성
  - Train data

x1	x2	y
1	2	0
2	3	0
3	1	0
4	3	1
5	3	1
6	2	1

```
x = np.array(
[[1, 2],
 [2, 3],
 [3, 1],
 [4, 3],
 [5, 3],
 [6, 2]])
```

```
y = np.array(
[[0],
 [0],
 [0],
 [1],
  [1],
  [1]])
```

  - Validation data

x1	x2	y
1	1	0
2	2	0
5	5	1
6	6	1

```
x_val=np.array(
[[1, 1],
 [2, 2],
 [4, 4],
 [5, 5]])
```

```
y_val=np.array(
[[0],
 [0],
 [1],
 [1]])
```

```
x=np.array([[1, 2], [2, 3], [3, 1], [4, 3],  [5, 3], [6, 2]])
y=np.array([[0],    [0],    [0],    [1],     [1],    [1]])
x_val=np.array([[1, 1], [2, 2], [4, 4],  [5, 5]])
y_val=np.array([[0],    [0],    [1],     [1]  ])
```

Example 1. an example of Logistic Regression (binary classifier)

- Data
  - $X = \{X_i\}$, $Y = \{Y_i\}$
- Hypothesis function (predict function)
  - $\bar{Y} = H(X) = g(XW + b) = \frac{1}{1+e^{-(WX+b)}}$
- Loss function: cross-entropy function
  - $Loss(\{X_i, Y_i\}) = -\frac{1}{n}\sum_{i=1}^{n} ce(H(X_i), Y_i)$

    $= -\frac{1}{n}\sum_{i=1}^{n} Y_i log\big(H(X_i)\big) + (1 - Y_i)\log\big(1 - H(X_i)\big)$
- Train
  - *Minimize W, b*

    $$W^*, b^* = \underset{W,b}{\text{argmin}}\, \text{Loss}(W, b)$$
  - Gradient Descent Algorithm

    initialize $w_0,\ b_0$

    $$w_{n+1} = w_n - \alpha \frac{\partial}{\partial w} cost(w, b)\Big]_{w=w_n, b=b_n}$$

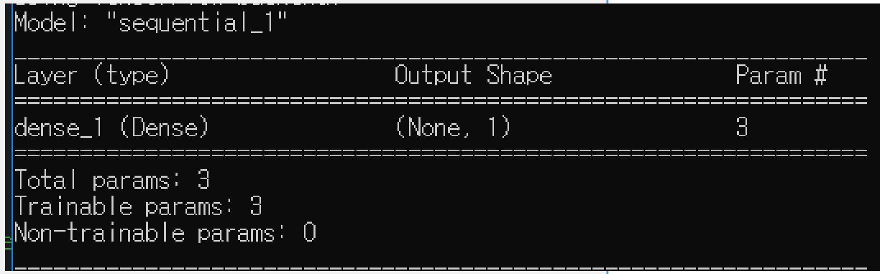
    $$b_{n+1} = b_n - \alpha \frac{\partial}{\partial b} cost(w, b)\Big]_{w=w_n, b=b_n}$$
- Estimation
  - Predict $\bar{y}_i = H(x_i)$
  - Accuracy$=\frac{1}{m}\sum_{i=1}^{m}(y_i == \bar{y}_i > 0.5 ? 1 : 0)$
  - 학습 중 최대 정확도

```
#creatre dataset
X      =np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]])
Y      =np.array([[0],[0],[0],[1], [1], [1]])
X_val=np.array([[1, 1], [2, 2], [4, 4], [5, 3]])
y_val=np.array([[0],[0],[1], [1] ])

#create model
model=Sequential()
model.add(                          #add layer
      Dense(                        #set Dense layer structure
            units=1,                #output no
            input_dim=X.shape[1],#input element no
            activation='sigmoid'))  #logistic function(regression)
model.summary()

#set compile(train) process
sgd=optimizers.SGD(lr=0.9)          #sgd optimizer wirh learning rate =0.9
model.compile(                      #set compiler-process
      loss='binary_crossentropy',   #loss function
      optimizer=sgd,                #stocahstic grdient descent training
      metrics=['accuracy'])         #evaluation index

#train model
hist=model.fit(                     #set train-process
      X,y,                          #train dataset
      epochs=2000,                  #iteration no
      verbose=1,                    #학습과정 출력 모드
      validation_data=(X_val, y_val)) #평가용 dataset
#evaluate model
Y_hat       =model.predict(X)       #predict with hypothesis function
Acc         =model.evaluate(X, y, verbose=0)[1] #evaluate accuracy of validation dataset
Acc_max     =np.max(hist.history['val_accuracy']) #학습 중 최대 정확도
```



Example 1. an example of Logistic Regression

```
from keras.models import Sequential
from keras.layers import Dense
import keras.optimizers as optimizers
import tensorflow as tf
import matplotlib.pylab as plt
import numpy as np
import os
```

```
#evaluate model
print(model.predict(np.array([[1,1]])) )          #[[1 1]] =>[[0.22243975]]
print(model.predict(np.array([X_val[0]])) )       #[[1 1]] =>[[0.22243975]]

print(X_val)                                      #X_val     :[[1 1] [2 2] [4 4] [5 3]]
print(y_val)                                      #y_val     :[[0] [0] [1] [1]]
y_hat=model.predict(X_val)
print('Y_hat:',y_hat)                             #Y_hat     : [[0.22243977] [0.3137669 ] [0.53874916] [0.82207584]]
print('round(y_hat):',np.round(y_hat))            #round(y_hat): [[0.] [0.] [1.] [1.]]
print('mean(round(y_hat)==y_val):',
      np.mean(np.round(y_hat)==y_val))            #mean(round(y_hat)==y_val): 1.0
print('acc_val:',
      model.evaluate(X_val, y_val, verbose=0)[1]) #acc_val: 1.0

print('loss,acc   :',
      model.evaluate(X, y, verbose=0))            #loss,acc   : [0.33072206377983093, 0.8333333134651184]
print('loss,acc val:',
      model.evaluate(X_val, y_val, verbose=0)) #loss,acc val: [0.3210359811782837, 1.0]

print(model.layers[0].get_weights())              #[array([[ 0.9520406], [-0.4421141]], dtype=float32), array([-1.857736], dtype=float32)]
print('acc_max   :',  np.max(hist.history['val_accuracy'])) #acc_max : 1.0
```

Example 1. an example of Logistic Regressio

```
from keras.models import Sequential
from keras.layers import Dense
import keras.optimizers as optimizers
import tensorflow as tf
import matplotlib.pylab as plt
import numpy as np
import os
```

```
#evaluate model
print(model.predict(np.array([[1,1]])) )          #[[1 1]] =>[[0.22243975]]
print(model.predict(np.array([X_val[0]])) )       #[[1 1]] =>[[0.22243975]]

print(X_val)                                      #X_val
print(y_val)                                      #y_val
y_hat=model.predict(X_val)
print('Y_hat:',y_hat)                             #Y_hat
print('round(y_hat):',np.round(y_hat))            #round(
print('mean(round(y_hat)==y_val):',
      np.mean(np.round(y_hat)==y_val))            #mean(
print('acc_val:',
      model.evaluate(X_val, y_val, verbose=0)[1]) #a

print('loss,acc   :',
      model.evaluate(X, y, verbose=0))            #loss,ac
print('loss,acc val:',
      model.evaluate(X_val, y_val, verbose=0)) #loss

print(model.layers[0].get_weights())#[array([[ 0.9520
```

```
Train on 6 samples, validate on 4 samples
Epoch 1/2000
6/6 [==============================] - 0s 48ms/step - loss: 0.9668 - accuracy: 0.5000 - val_loss: 1.2026 - val_accuracy: 0.5000
Epoch 2/2000
6/6 [==============================] - 0s 2ms/step - loss: 1.5434 - accuracy: 0.5000 - val_loss: 0.6789 - val_accuracy: 0.7500
Epoch 3/2000
6/6 [==============================] - 0s 1ms/step - loss: 0.5122 - accuracy: 0.8333 - val_loss: 0.5418 - val_accuracy: 0.7500
Epoch 1999/2000
6/6 [==============================] - 0s 831us/step - loss: 0.0114 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 2000/2000
6/6 [==============================] - 0s 831us/step - loss: 0.0114 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 1.0000
[[1.1566788e-05]]
[[1.1566788e-05]]
[[1 1]
 [2 2]
 [4 4]
 [5 3]]
[[0]
 [0]
 [1]
 [1]]
Y_hat: [[1.1563301e-05]
 [3.1127334e-03]
 [9.9562448e-01]
 [9.9939990e-01]]
round(y_hat): [[0.]
 [0.]
 [1.]
 [1.]]
mean(round(y_hat)==y_val): 1.0
acc_val: 1.0
loss,acc   : [0.011412438936531544, 1.0]
loss,acc val: [0.002028629882261157, 1.0]
[array([[3.7944148],
       [1.8038198]], dtype=float32), array([-16.965595], dtype=float32)]
acc_max   : 1.0
Press any key to continue . . .
```

Example 2. a logistic regression example with L1 and L2 regularization

- 개념
  - Overfitting 문제를 해결하기위한 방법
  - Dense layer의 weight의학습에 제약을 가한다.

  - From keras.regularizers import l1_l2

  - Reg = l1_l2(l1=0.01, l2=0.01)

  - model=Sequential()
  - model.add(Dense(1,activation= ' sigmoid’ ,
    input_dim=x.shape[1],
    W_regularizer=reg))
  - model.compile(optimizer='rmsprop', loss='binary_crossentropy')

Example 2. a logistic regression example with L1 and L2 regularization

```
from keras.regularizers import l1_l2

#train and validation data
x=np.array([[1, 2], [2, 3], [3, 1], [4, 3],  [5, 3], [6, 2]])
y=np.array([[0],[0],[0],[1], [1],    [1]])
x_val=np.array([[1, 1], [2, 2], [4, 4],  [5, 5]])
y_val=np.array([[0],[0],[1], [1]  ])
print(x.shape,x); print(y.shape,y)
print(x_val.shape,x_val); print(y_val.shape,y_val)

# 2-class logistic regression with L1 and L2 regularization
model=Sequential()                    #모델 정의
reg = l1_l2(l1=0.01, l2=0.01)
model.add(Dense(1,activation= ' sigmoid’,
                input_dim=x.shape[1],
                W_regularizer=reg))
model.compile(optimizer='rmsprop', loss='binary_crossentropy')

model.fit(x,y,epochs=2000,verbose=1,
        validation_data=(x_val, y_val))

p=model.predict(x_val)                  #검증용 데이터의 예측값 계산
Print( ' accracy : {:.2f} % ' .format(   #예측값의 인식률 계산
        np.mean(np.round(p)==y_val)*100)) #accracy : 100.00 %
```

Example 3. an example of logistic examples

1) Diabetes Diagnosis(당뇨병 진단)을 위한 logistic regression(binary classifier)

- Dataset구조

	1	2	3	4	5	6	7	8	9
1	-0.88235	-0.14573	0.081967	-0.41414	0	-0.20715	-0.76687	-0.66667	1
2	-0.05882	0.839196	0.04918	0	0	-0.30551	-0.49274	-0.63333	0
3	-0.88235	-0.10553	0.081967	-0.53535	-0.77778	-0.16244	-0.924	0	1
4	0	0.376884	-0.34426	-0.29293	-0.60284	0.28465	0.887276	-0.6	0
5	-0.41177	0.165829	0.213115	0	0	-0.23696	-0.89496	-0.7	1
6	-0.64706	-0.21608	-0.18033	-0.35354	-0.79196	-0.07601	-0.85483	-0.83333	0
7	0.176471	0.155779	0	0	0	0.052161	-0.95218	-0.73333	1
8	-0.76471	0.979899	0.147541	-0.09091	0.283688	-0.09091	-0.93168	0.066667	0
9	-0.05882	0.256281	0.57377	0	0	0	-0.86849	0.1	0

- Dataset 생성

```
XY    =np.loadtxt('data/data-03-diabets.txt',dtype=float,delimiter=',')
X     =XY[:,:-1]          #X:(759, 8)
Y     =XY[:,[-1]]         #Y:(759, 1)
X,X_val,y,y_val=train_test_split(X,Y,random_state=0,shuffle=True)
#train:((569, 8),(569, 1)),  val:((190, 8),(190, 1))
```

Example 3. an example of logistic examples

```
#create model
model=Sequential()
model.add(#add layer
        Dense(                          #set Dense layer structure
                units=1,                #output no
                input_dim=X.shape[1],#input element no
                activation='sigmoid’))  #logistic function(regression)
model.summary()

#set compile(train) process
model.compile(                          #set compiler-process
        loss='binary_crossentropy’,     #loss function
        optimizer=‘sgd’,                #stocahstic grdient descent training
        metrics=['accuracy’])           #evaluation index

#train model
hist=model.fit(                 #set train-process
        X,y,                    #train dataset
        epochs=1000,            #iteration no
        verbose=1,              #학습과정 출력 모드
        validation_data=(X_val, y_val)) #평가용 dataset

#evaluate model
print('acc_train :', model.evaluate(X, y, verbose=0)[1])            #
print('acc_val  :', model.evaluate(X_val, y_val, verbose=0)[1])  #

print('acc_train_max :', np.max(hist.history['accuracy']))          #
print('acc_val_max   :',  np.max(hist.history['val_accuracy']))     #
```

```
X: (759, 8)    Y: (759, 1)
train:((569, 8),(569, 1)),  val:((190, 8),(190, 1))
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 9
=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
Train on 569 samples, validate on 190 samples
Epoch 1/1000
569/569 [==============================] - 0s 694us/step - loss: 0.8233 - accuracy: 0.3638 - val_loss: 0.8542 - val_accuracy: 0.3579
Epoch 2/1000
569/569 [==============================] - 0s 145us/step - loss: 0.7961 - accuracy: 0.4007 - val_loss: 0.8289 - val_accuracy: 0.3632
Epoch 3/1000
569/569 [==============================] - 0s 100us/step - loss: 0.7730 - accuracy: 0.4200 - val_loss: 0.8078 - val_accuracy: 0.3474
Epoch 4/1000
569/569 [==============================] - 0s 79us/step - loss: 0.7536 - accuracy: 0.4341 - val_loss: 0.7899 - val_accuracy: 0.3579
Epoch 5/1000
569/569 [==============================] - 0s 93us/step - loss: 0.7373 - accuracy: 0.4587 - val_loss: 0.7746 - val_accuracy: 0.3684
Epoch 6/1000
569/569 [==============================] - 0s 100us/step - loss: 0.7233 - accuracy: 0.4657 - val_loss: 0.7613 - val_accuracy: 0.3789
Epoch 7/1000
569/569 [==============================] - 0s 126us/step - loss: 0.7113 - accuracy: 0.5097 - val_loss: 0.7503 - val_accuracy: 0.4158
Epoch 8/1000
569/569 [==============================] - 0s 119us/step - loss: 0.7012 - accuracy: 0.5290 - val_loss: 0.7407 - val_accuracy: 0.4421
Epoch 9/1000
569/569 [==============================] - 0s 107us/step - loss: 0.6925 - accuracy: 0.5501 - val_loss: 0.7324 - val_accuracy: 0.4632
Epoch 10/1000
569/569 [==============================] - 0s 105us/step - loss: 0.6851 - accuracy: 0.5624 - val_loss: 0.7252 - val_accuracy: 0.5000
Epoch 11/1000
Epoch 998/1000
569/569 [==============================] - 0s 84us/step - loss: 0.4885 - accuracy: 0.7663 - val_loss: 0.4378 - val_accuracy: 0.8105
Epoch 999/1000
569/569 [==============================] - 0s 84us/step - loss: 0.4885 - accuracy: 0.7663 - val_loss: 0.4378 - val_accuracy: 0.8105
Epoch 1000/1000
569/569 [==============================] - 0s 78us/step - loss: 0.4885 - accuracy: 0.7663 - val_loss: 0.4378 - val_accuracy: 0.8105
acc_train : 0.76625657081604
acc_val   : 0.8105263113975525
acc_train_max : 0.7697715
acc_val_max   : 0.8105263113975525
Press any key to continue . . .
```

Example 3. an example of logistic examples

2)위스콘신 유방암 데이터셋을 이용한 logistic regression(binary classifier)

- from sklearn import datasets
- cancer=datasets.load_breast_cancer()

```
cancer=datasets.load_breast_cancer()   # cancer[‘data’] :(569, 30)
X,X_val,y,y_val=train_test_split(cancer['data'],cancer['target'],random_state=0,shuffle=True)
#train:((426, 30),(426,)),  val:((143, 30),(143,))
```

- logistic classifier를 생성하고 정확도를 계산하시오 .

```
569/569 [==============================] - 0s 79us/step - loss: 0.4910 - accuracy: 0.7610 - val_loss: 0.4353 - val_accuracy: 0.7947
```