Deep Learning

Logistic Regression

이진분류, Logistic regression, sigmoid,
mean square error, binary cross-entropy,
Logistic regression 코드 예제, 당뇨병진단, 유방암 진단

Yoonjoong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

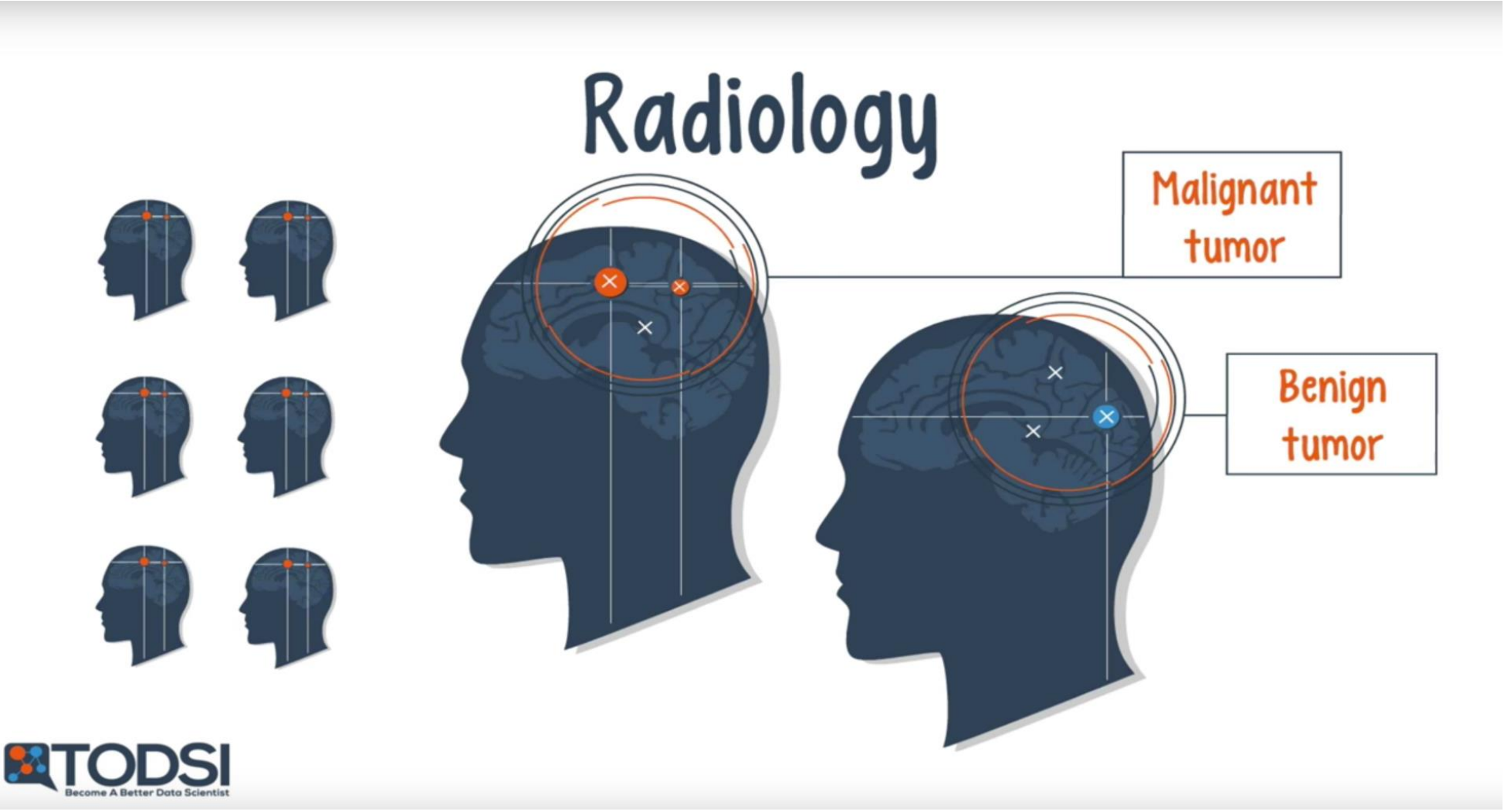
Content.

1. 이진분류

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

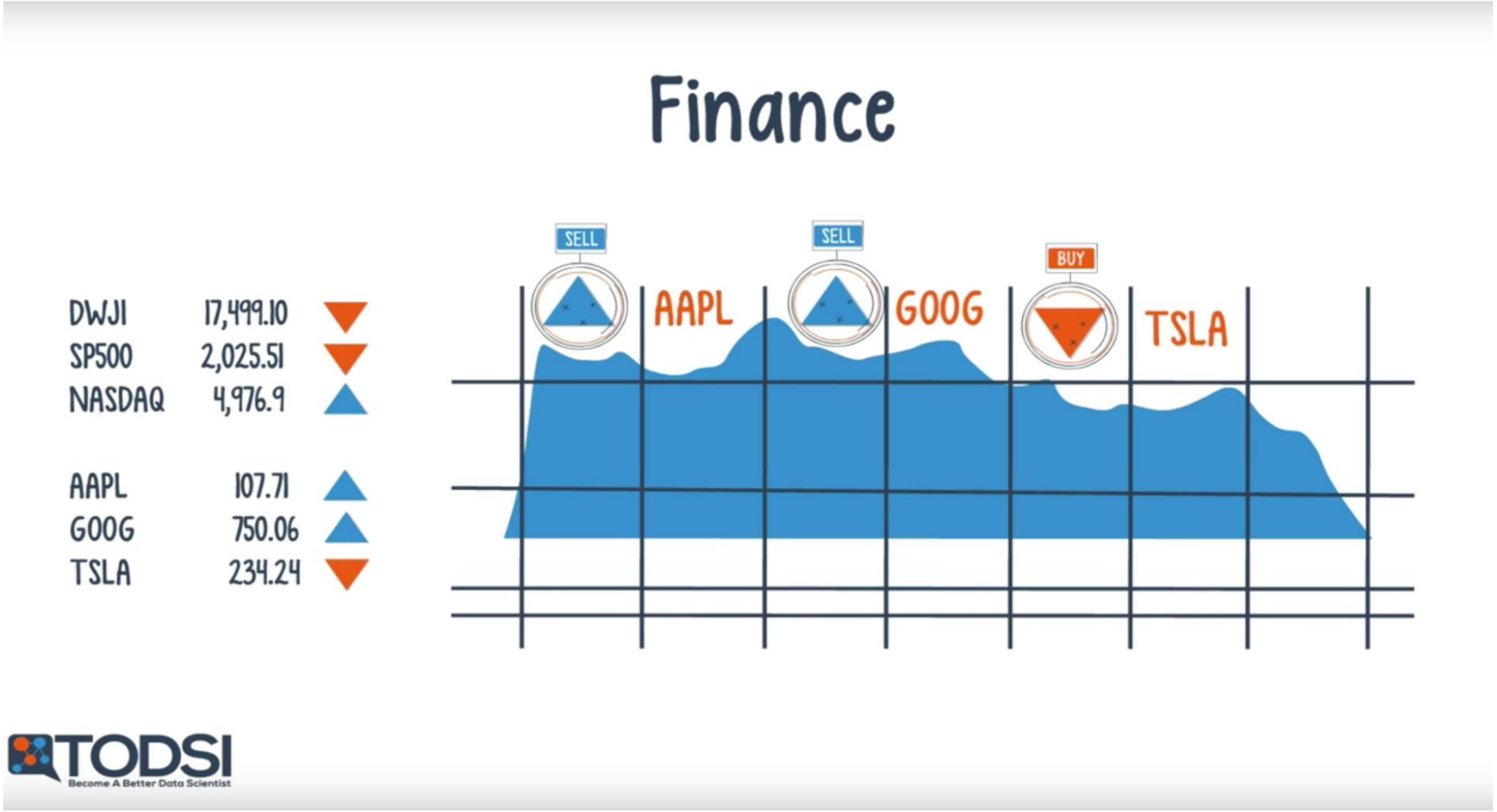
- 0,1 Encoding
  - Spam Email Detection
    - Spam(1) or Ham(0)
  - Facebook feed
    - show(1) or hide(0)
  - Credit Card Fraudulent Transaction detection
    legitimate(1)/fraud(0)

1.1 이진분류의 예



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

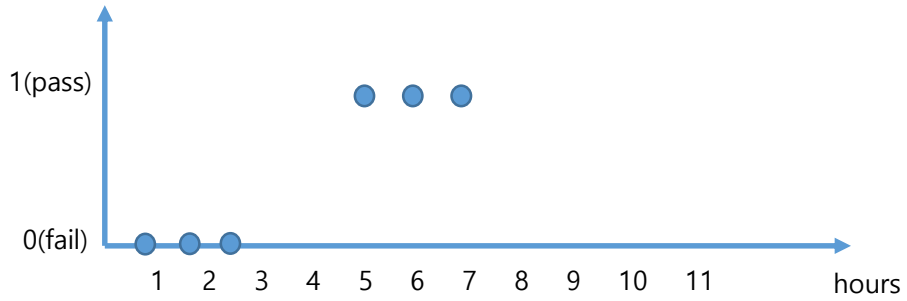
1.1 이진분류의 예



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

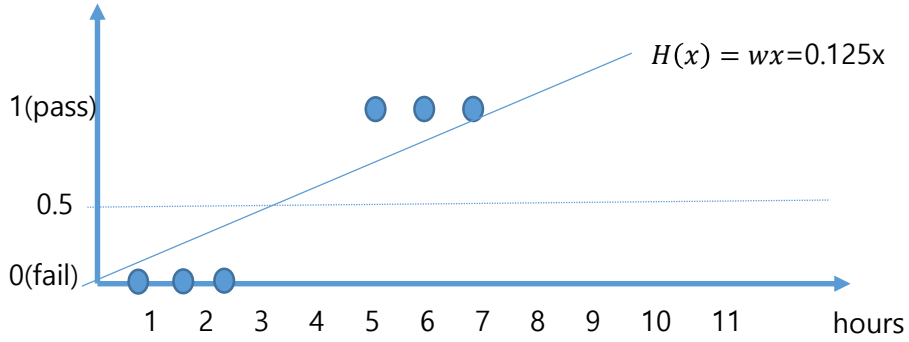
1.2 로지스텍 회귀(Logistic Regression)

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



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

  - 선형회귀모델($h(x) = wx + b$)로 분류가능한가?

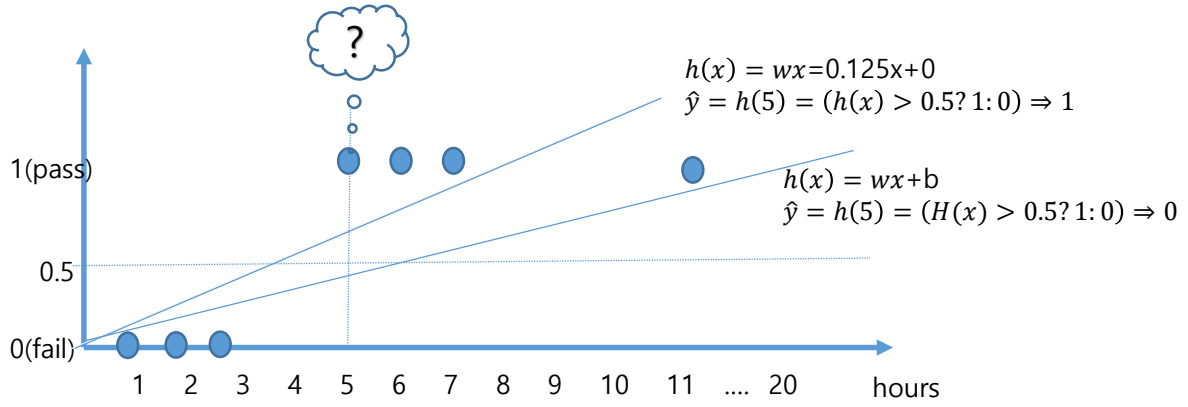


x	y
1	0
2	0
3	0
5	1
6	1
7	1

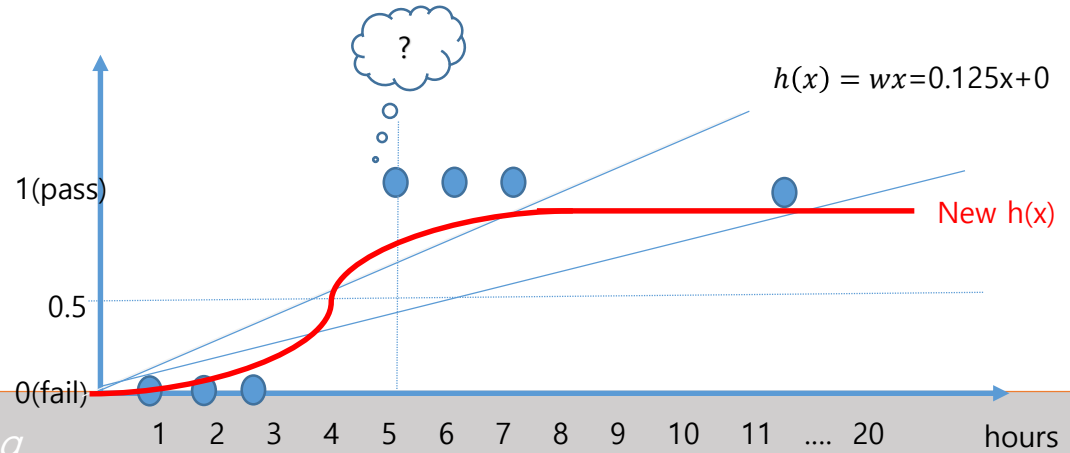
1.2 로지스텍 회귀(cont.)

- 선형회귀모델($h(x) = wx + b$)로 이진분류가능한가 ?

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



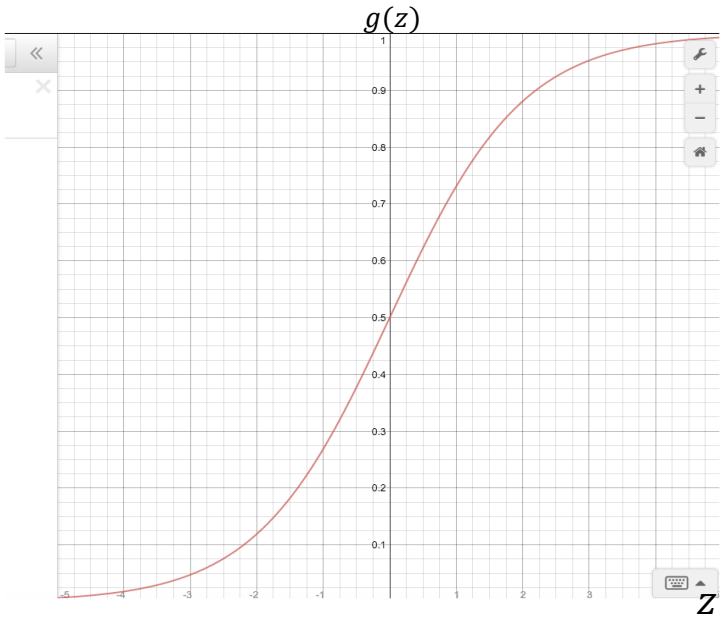
Other Hypothesis function is required



$h(5) = (H(x) > 0.5?\, 1 : 0) = 1$을 만족하는
새로운 예측(가설, $activation$) 함수가 필요

1.2 로지스텍 회귀(cont.)

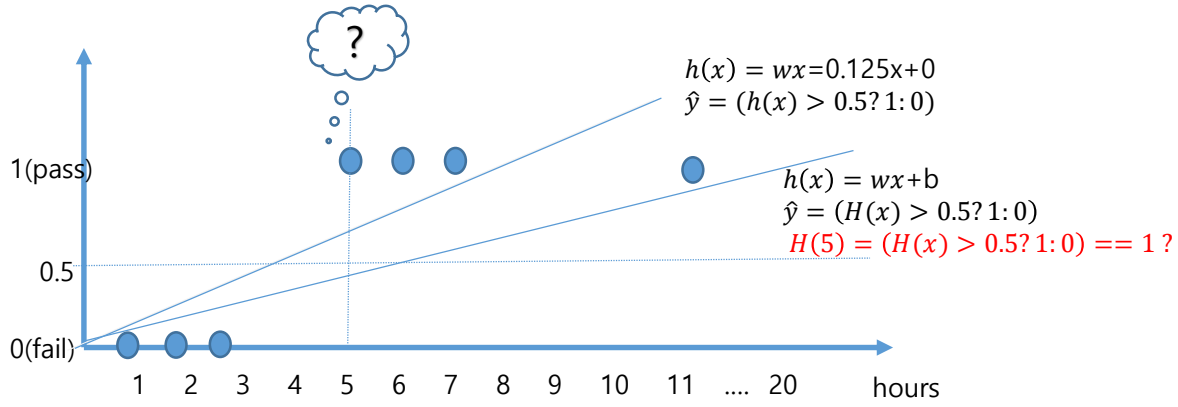
- logistic function
  - sigmoid function.
  - sigmoid :
    Curved in two directions,
    like the letter "S",
    or the Greek ς (sigma).
  - $g(z) = \frac{1}{1+e^{-z}}$
- linear hypothesis (linear regression)
  - $h_l(x) = wx + b$
- logistic hypothesis (logistic regression)
  - $h(x) = g(h_l(x))$
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



1.2 로지스텍 회귀(cont.)

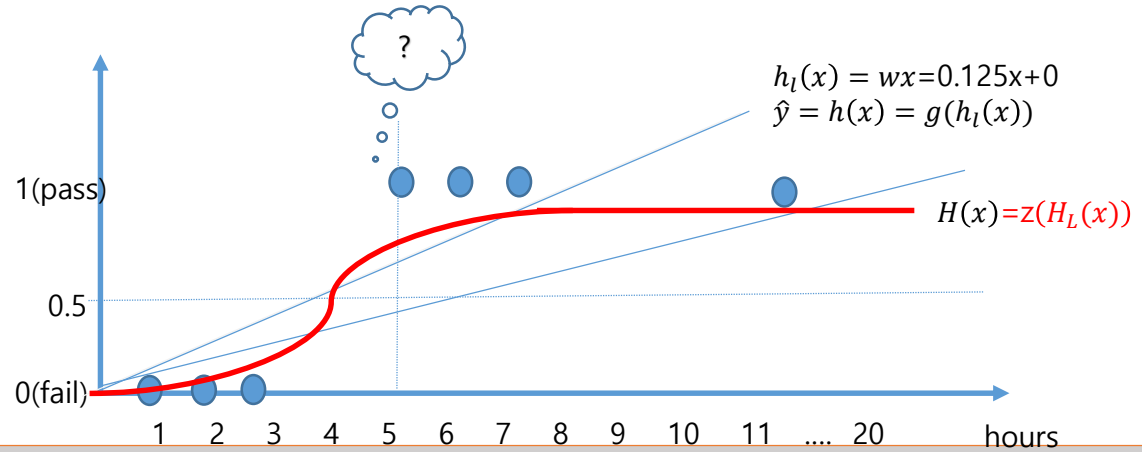
- 로지스틱회귀모델($\mathrm{h}(x) = z(\mathrm{h}_1(x)) = z(wx + b)$)로 분류 가능하다.

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))
```



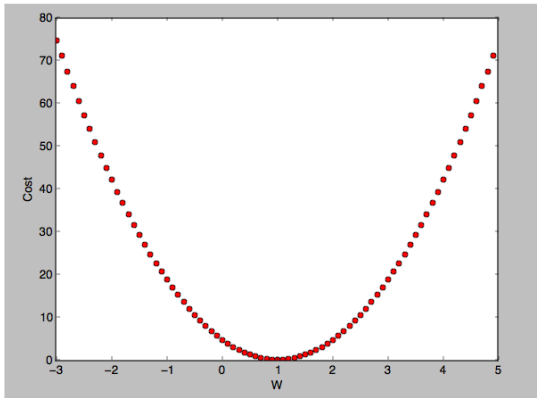
Logistic regression model

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

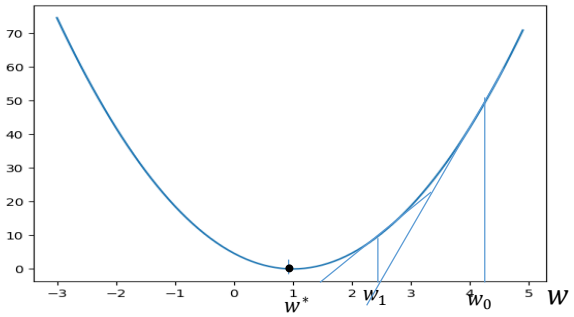
1.2 로지스텍 회귀(cont.)

학습은 가능한가?

- Linear Regression => Logistic Regression
  - 모델함수
    - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
      - $\hat{y}_l = wx + b,$ $[-\infty \sim +\infty]$
  - **손실함수(mean square error) ?**
    - $loss(X, Y) = \frac{1}{N}\sum(\hat{y}_i - y_i)^2$
  - **학습 ?**
    - $w^*, b^* = argmin_{w,b}(loss(w, b)|X, Y))$
    - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
    - $R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$, $\text{mse} = \frac{1}{N}\sum(\hat{y}_i - y_i)^2$

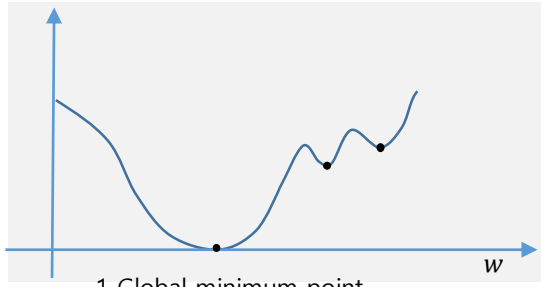


$loss_{mse}(w)$ in Linear Regression



1 Global minimum point
No Local minimum points
GDA : good

$loss_{mse}(w)$ in Logistic Regression



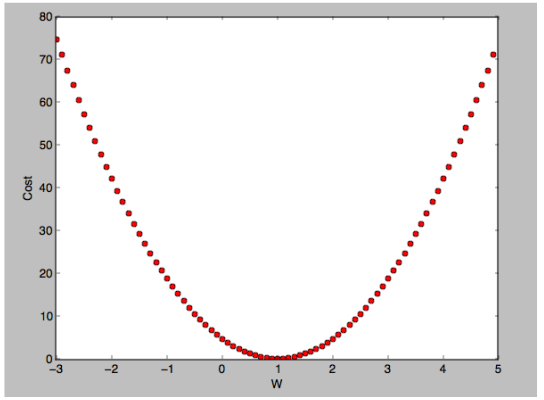
1 Global minimum point
some Local minimum points
GDA : difficult

*Local minima*가 안생기는
다른 손실함수가 필요한다.

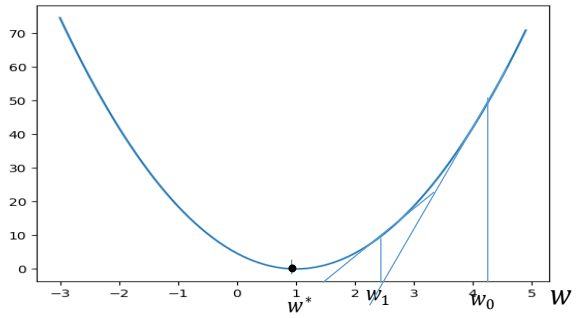
1.2 로지스텍 회귀(cont.)

학습은 가능한가?

- Logistic Regression 선형회귀 분석
  - 모델함수
    - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
      - $\hat{y}_l = wx + b,$ $[-\infty \sim +\infty]$
  - **손실함수(mean square error) ?**
    - $loss(X, Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
  - **학습 ?**
    - $w^*, b^* = argmin_{w,b}(loss(w, b)|X, Y))$
    - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
    - acc = $\frac{\text{정인시 샘플수}}{\text{평가샘플수}}$

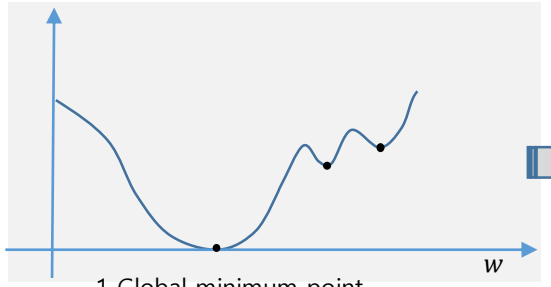


$loss_{mse}(w)$ in Linear Regression



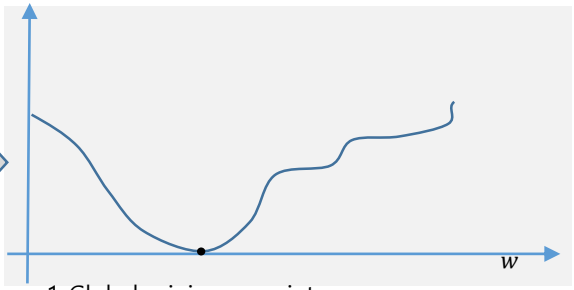
1 Global minimum point
No Local minimum points
GDA : good

$loss_{mse}(w)$ in Logistic Regression



1 Global minimum point
some Local minimum points
GDA : difficult

$loss_{ce}(w)$ in Logistic Regression

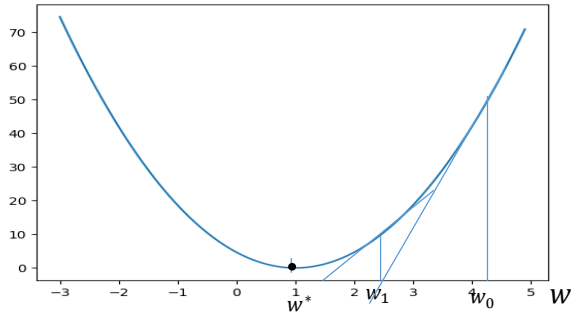


1 Global minimum point
No Local minimum points
GDA : good

1.2 로지스텍 회귀(cont.)

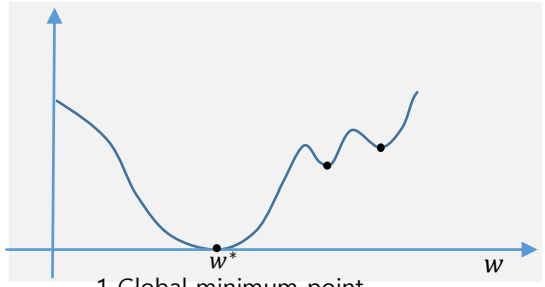
- Linear Regression
  - 모델
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$
  - 손실함수 (mean square error)
    - $loss(X,Y) = \frac{1}{N}\sum(y_i - \hat{y}_i)^2$
  - 학습
    - Gradient Descent Algorithm(GDA)
- Logistic Regression
  - 모델
    - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수 (binary cross entropy)
    - $loss(X,Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
  - 학습
    - Gradient Descent Algorithm

$los_{\mathrm{mse}}(w)$ in Linear Regression



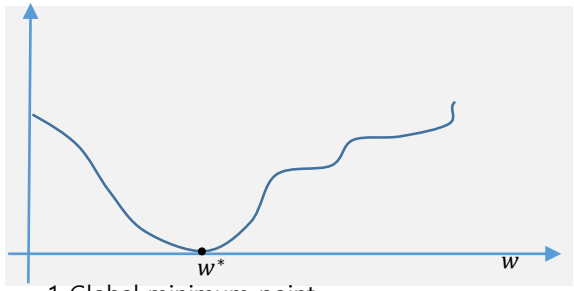
1 Global minimum point
No Local minimum points
GDA : good

$loss_{mse}(w)$ in Logistic Regression



1 Global minimum point
some Local minimum points
GDA : difficult

$loss_{ce}(w)$ in Logistic Regression

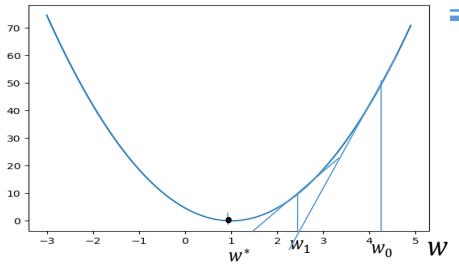


1 Global minimum point
No Local minimum points
GDA : good

1.2 로지스텍 회귀(cont.)

- Linear Regression
  - 모델
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$
  - 손실함수 (mean square error)
    - $loss(X,Y) = \frac{1}{N}\sum(y_i - \hat{y}_i)^2$
  - 학습
    - Gradient Descent Algorithm(GDA)
- Logistic Regression
  - 모델
    - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수 (binary cross entropy)
    - $loss(X,Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
  - 학습
    - Gradient Descent Algorithm

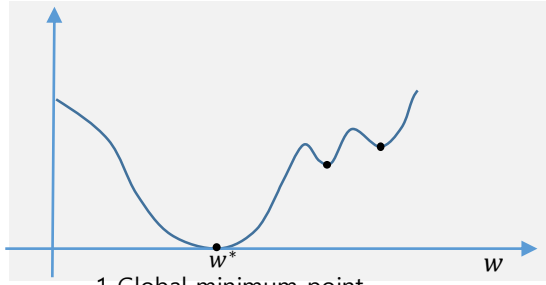
$loss_{mse}(w)$ in Linear Regression



1 Global minimum point
No Local minimum points
GDA : good

```
model=Sequential()
model.add(Dense(1,          #출력의 수
    activation='linear',    #linear regression
    input_dim=2))           #입력 특징의수
model.compile(
    loss='mse',             #손실함수 mean square error
    optimizer='sgd')        #gradient descent optimizer
```

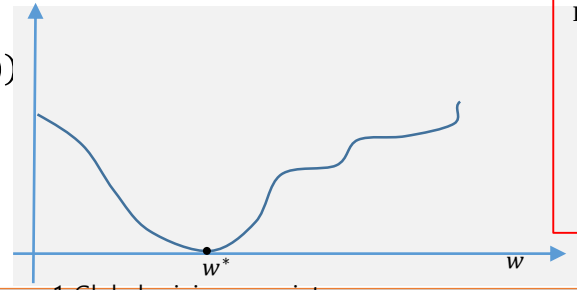
$loss_{mse}(w)$ in Logistic Regression



1 Global minimum point
some Local minimum points
GDA : difficult

```
model=Sequential()
model.add(Dense(1,          #출력의 수
    activation='sigmoid',   #logistic regression
    input_dim=2))           #입력특징의 수
model.compile(
    loss='mse',             #손실함수 mean square error
    optimizer='sgd')        #gradient descent optimizer
```

$loss_{ce}(w)$ in Logistic Regression



1 Global minimum point
No Local minimum points
GDA : good

```
model=Sequential()
model.add(Dense(1,          #출력의 수
    activation='sigmoid',   #logistic regression
    input_dim=2))           #입력특징의 수
model.compile(
    loss='binary_crossentropy', #손실함수 cross entropy
    optimizer='sgd')        #gradient descent optimizer
```

1.2 로지스텍 회귀(cont.)

- Linear Regression 선형회귀 분석
  - 모델,선형예측함수 , 실수 예측
    - $\hat{y} = h(\mathrm{x}) = wx + b \qquad [-\infty \sim +\infty]$
  - 손실함수(mean square error)
    - $loss(\mathrm{X},\mathrm{Y}) = \frac{1}{N}\sum(\hat{y}_i - y_i)^2$
  - 학습
    - $w^*, b^* = argmin_{w,b}(loss(w,b)|X,Y))$
    - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = w^*x + b^*$
  - 평가지수
    - $mse = \frac{1}{N}\sum(\hat{y}_i - y_i)^2$

```
#Linear Regression
#data 생성
    X=np.array([[1], [2], [3], [5], [6], [7]])
    y=np.array([[50],[55],[60],[90], [93], [95]])

    #모델 생성
    model=Sequential()
    model.add(                      #add Dense layer
        Dense(                      #set Dense layer structure
            units=1,                #1, 출력항목 수
            input_dim=X.shape[1],#1,입력 특성 수
            activation='linear'))   # linear function(regression)
    model.summary()

    #학습방법설정(compile)
    model.compile(                  #set compiler-process
            loss='mse',             #loss function
            optimizer='sgd',        #stochastic gradient descent training
            metrics=['mse'])        #evaluation index

    #학습
    hist=model.fit(                 #set train-process
            X,y,                        #학습데이터셋
            epochs=2000,                #반복학습횟수
            verbose=1)                  #학습과정 출력 모드

    #모델 평가
    print('h([[5],[8]]): ',model.predict(np.array([[5],[8]]) ))
    #h([[5],[8]]):  [[ 82.44047 ] [108.262146]]

    print(X,":",model.predict(X))
    #[[1] [2] [3] [5] [6] [7]] :
    #[[48.007107] [56.615417] [65.223724] [82.440346] [91.04865 ] [99.65697 ]]

    print(model.evaluate(X,y))
    #loss acc = [19.41866111755371, 19.41866111755371]
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
```

X1 (hour)	Y (score)
1	50
2	55
3	60
5	90
6	93
7	95

예측 값이 실수가 아니라 2인 경우 ?

1.2 로지스텍 회귀(cont.)

- Logistic Regression 분석 :이진분류
  - 모델, 예측함수
    - $\hat{y} =$ h(x) $= g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ [0 ~1]
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수 (binary cross entropy)
    - $loss(X,Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
  - 학습
    - $w^*, b^* = argmin_{w,b}(loss(w,b)|X,Y))$
    - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = w^*x + b^*$
  - 평가지수
    - Acc=$\frac{\text{정인식 샘플수}}{\text{전체평가샘플수}}$

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

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
```

```
#데이터셋 생성
X=np.array([[1], [2], [3], [5], [6], [7], [11]])
y=np.array([[0],[0],[0],[1], [1], [1], [1]])

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

#학습방법설정(compile)
model.compile(                      #set compiler-process
        loss='binary_crossentropy',    # 손실함수
        optimizer='sgd',            #stochastic gradient descent training
        metrics=['accuracy'])           #evaluation index

#학습
hist=model.fit(                     #set train-process
        X,y,                        #학습데이터셋
        epochs=1000,                    #반복학습횟수
        verbose=0)                  #학습과정 출력 모드

#모델 평가
print('h([[5],[8]]): ',model.predict(np.array([[5],[8]]) ))
#h([[5],[8]]):  [[0.78487945]  [0.9704663 ]]


print(model.predict(X))
print(np.round(model.predict(X)))
#[[ 1] [ 2] [ 3] [ 5] [ 6] [ 7] [11]] :
#[[0.16298041]  [0.2883153 ] [0.45736802] [0.78487945] [0.88359964] [0.9404536 ] [0.9966323 ]]
#[[0.] [0.] [0.] [1.] [1.] [1.] [1.]]

print(model.evaluate(X,y))
#loss acc = [0.2228706330060959, 1.0]
```

X1 (hour)	Y (pass/fail)
1	0
2	0
3	0
5	1
6	1
7	1
11	1

2 이진분류 예제

- 이진분류 코드 예제

- 당뇨병 진단 예제
  - 8특징 셈플

- 유방암 진단
  - 위스콘신 유방암 데이터셋
  - 30특징 샘플

2.1 이진분류 코드 예제

- Logistic Regression 예제
  - Dataset 생성
    - X,Y
  - 모델생성
    - Logistic Regression 함수
      - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
      - $\hat{y}_l = h_l(x) = wx + b$ $[-\infty \sim +\infty]$
  - 학습방법설정
    - 손실함수
      - $loss(X,Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
    - 학습기(방법) 설정
      - $w^*, b^* = argmin_{w,b}(loss(w,b)|X,Y))$
      - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = g(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
    - accuracy $= \frac{1}{N}\sum(\hat{y}_i = y_i)$

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

```
y = np.arra
[[0],
 [0],
 [0],
 [1],
 [1],
 [1]])
```

```
#dataset 생성
X  =np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]])     #학습데이터셋 X
Y  =np.array([[0],[0],[0],[1], [1], [1]])                          #학습데이터셋 Y
X_val=np.array([[1, 1], [2, 2], [4, 4], [5, 3]])                   #검증데이터셋 X
Y_val=np.array([[0],[0],[1], [1] ])                                #검증데이터셋 Y

#모델생성
model=Sequential()
model.add(                          #모델의 한 layer 추가
   Dense(                           #Dense layer의 구조
        units=1,                    #1, 출력 항의 수
        input_dim=X.shape[1],       #2, 입력 특징의 수
        activation= ' sigmoid'))    #logistic regression
model.summary()                     #model의 구조 출력

#학습(compile) 방법설정
sgd=optimizers.SGD(lr=0.9)          #sgd optimizer wirh learning rate =0.9
model.compile(                      #set compiler-process
        loss='binary_crossentropy',  #손실함수 cross-entropy
        optimizer=sgd,              #학습기 stochastic gradient descent training
        metrics=['accuracy'])       #평가지수

#학습
hist=model.fit(                     #set train-process
        X,y,                        #학습데이터셋
        epochs=2000,                #학습반복횟수
        verbose=1,                  #학습과정 출력 모드
        validation_data=(X_val, y_val)) #검증요 데이터셋
#평가
y_hat       = model.predict(X)                  #샘플 X의 예측 값 계산 ŷ = h(x)
Acc         = model.evaluate(X, Y, verbose=0)[1] #학습데이터셋 X,Y의 성능(예측정확도) 계산
Acc_max     = np.max(hist.history['val_accuracy']) #학습 중 최대 정확도
```

2.1 이진분류 코드 예제(cont.)

- Logistic Regression 예제
  - Dataset 생성
    - X,Y
  - 모델생성
    - Logistic Regression 함수
      - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
      - $\hat{y}_l = h_l(x) = wx + b$ $[-\infty \sim +\infty]$
  - 학습방법설정
    - 손실함수
      - $loss(X,Y) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}_i) + (1-y)\log(1-\hat{y}_i))$
    - 학습기(방법) 설정
      - $w^*, b^* = argmin_{w,b}(loss(w,b)|X,Y))$
      - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = g(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
    - accuracy $= \frac{1}{N}\sum(\hat{y}_i = y_i)$

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

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

#모델생성
model=Sequential()
model.add(                                                           #
    Dense(                                                           #
        units=1,                                                     #
        input_dim=X.shape[1],                                        #
        activation= ' sigmoid’))                                     #logistic regression
model.summary()                                                      #model의 구조 출력

#학습(compile) 방법설정
sgd=optimizers.SGD(lr=0.9)          #sgd optimizer wirh learning rate =0.9
model.compile(                      #set compiler-process
        loss='binary_crossentropy’, #손실함수 cross-entropy
        optimizer=sgd,              #학습기 stochastic gradient descent training
        metrics=['accuracy’])       #평가지수

#학습
hist=model.fit(                     #set train-process
        X,y,                        #학습데이터셋
        epochs=2000,                #학습반복횟수
        verbose=1,                  #학습과정 출력 모드
        validation_data=(X_val, y_val)) #검증요 데이터셋
#평가
y_hat    = model.predict(X)                    #샘플 X의 예측 값 계산 ŷ = h(x)
Acc      = model.evaluate(X, y, verbose=0)[1] #학습데이터셋 X,Y의 성능(예측정확도) 계산
Acc_max  = np.max(hist.history[‘val_accuracy’]) #학습 중 최대 정확도
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

2.1 이진분류 코드 예제(cont.)

```
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
```

```
#dataset 생성
X  =np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]])  #학습데이터셋 X
Y  =np.array([[0],[0],[0],[1], [1], [1]])                        #학습데이터셋 Y
X_val=np.array([[1, 1], [2, 2], [4, 4], [5, 3]])                 #검증데이터셋 X
Y_val=np.array([[0],[0],[1], [1] ])                              #검증데이터셋 Y
```

```
#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('acc_val:',
      model.evaluate(X_val, y_val, verbose=0)[1])  #acc_val: 1.0     cf, linreg.score(X_val, y_val,) in machine laerning
print('mean(round(y_hat)==y_val):',
      np.mean(np.round(y_hat)==y_val))             #mean(round(y_hat)==y_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
```

2.2 당뇨병 진단

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

- Dataset
  - 8개의 특징을 갖는 샘플과 2개의 라벨을 갖는 암 진단 데이터셋

	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

```
#데이터셋 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))
```

2.2 당뇨병 진단(cont.)

```
#모델생성
model=Sequential()
model.add(#add layer
        Dense(                          #set Dense layer structure
                units=1,                #1, 출력항의 수
                input_dim=X.shape[1],#8, 입력특징의 수
                activation='sigmoid'))  #logistic regression
model.summary()

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

#학습
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

#모델의 성능평가
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 . . .
```

2.3 유방암 진단

- 제목 : 유방암 진단
- 주제 : 위스콘신 유방암 데이터셋에 대하여 Dense 모델과 LogisticRegression 모델을 생성하고 성능을 비교 분석하라
- 내용
  - 데이터셋 생성

```
from sklearn import datasets
cancer=datasets.load_breast_cancer()   # cancer['data'] :(569, 30)
X_train, X_val, y_train, y_val=train_test_split(cancer['data'],cancer['target'],random_state=0,shuffle=True)
#train:((426, 30),(426,)),  val:((143, 30),(143,))
```

1. Dense를 이용한 로지스틱 회귀 모델의 생성 및 성능계산
   - Dense 레이어로 모델을 만들고 학습하여 평가용 데이터의 성능 acc_dense를 구하시오.

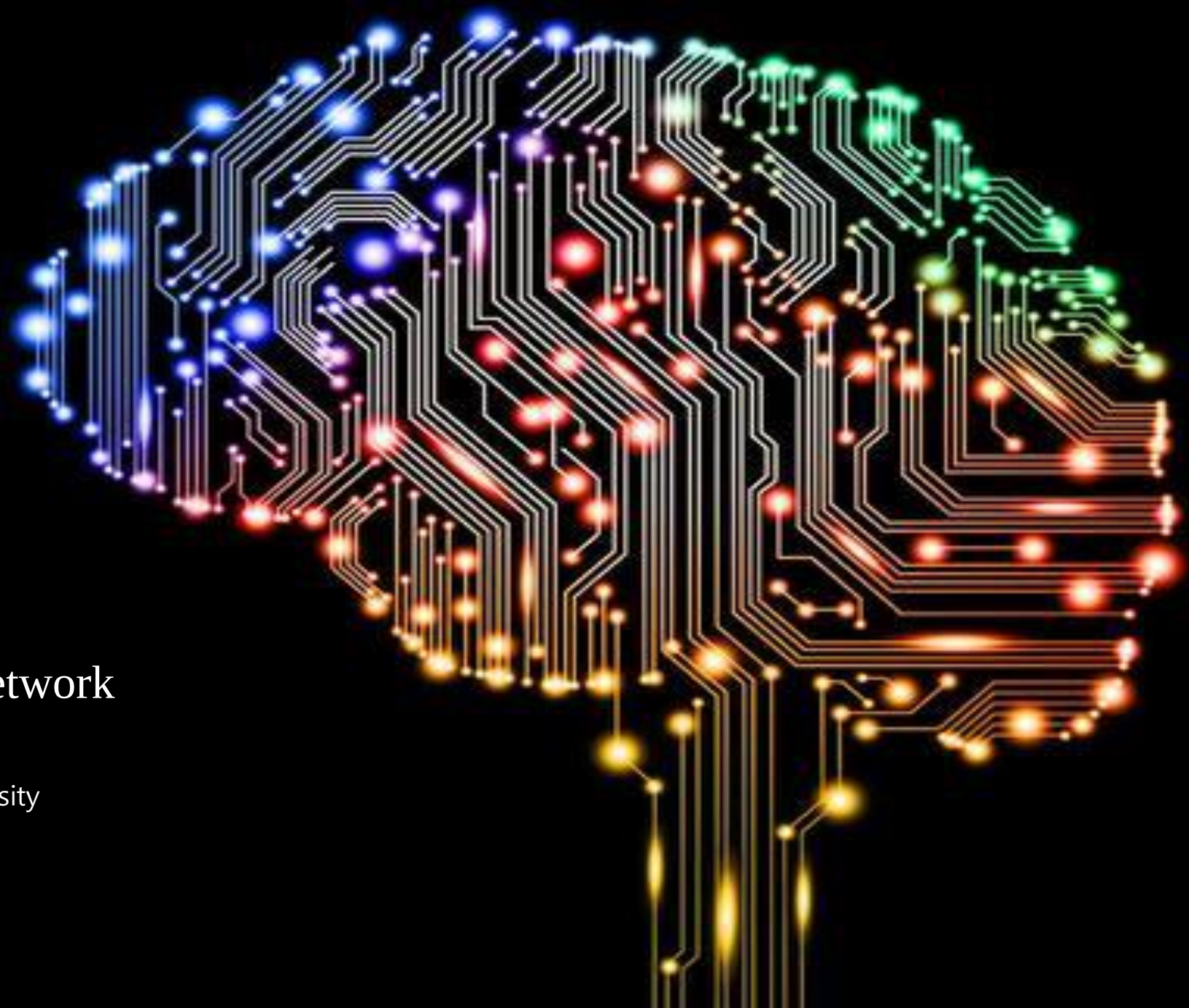
```
Dense(              #set Dense layer structure
   units=1,         #1, 출력항의 수
  input_dim=30  # 입력특징의 수
 activation='sigmoid'))    #logistic regression
```

2.3 유방암 진단(cont.)

2. LogisticRegression 을 이용한 로지스틱 회귀 모델의 생성 및 성능계산
   - Sklearn.linear_model 패키지의의 LogisticRegresion 클래스를이용하여 acc_logreg을 구하 시오.

```
from sklearn.linear_model import LogisticRegresion
logreg=LogisticRegresion().fit(X_train,y_train)
acc=logreg.score(X_train,y_train)
```

3. acc_dense와 acc_logreg의 비교분석하시오.



Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University