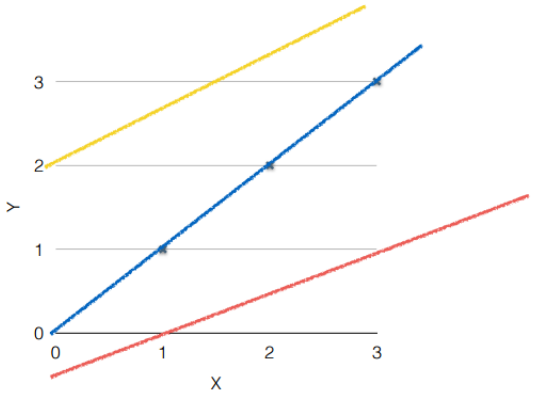
[Linear regression(단일변수)]

(Linear) Hypothesis : H(x) = W x + b

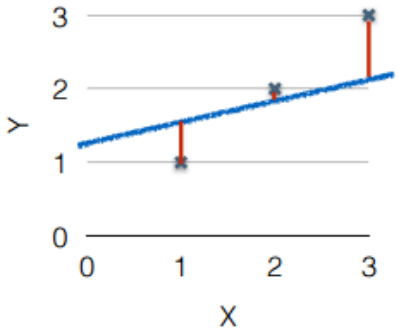
Which hypothesis is better?



minimize cost (W, b)

비용(Cost) 또는 손실(loss) : 예측 값 - 실제 값

$$H(x) - y$$



비용(Cost) 또는 손실(loss) 평균

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

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

H(x) = W x + b 의 비용(Cost) 함수

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

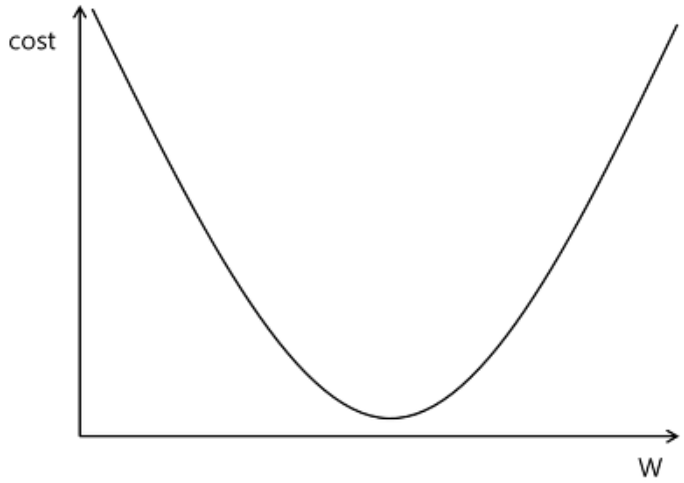
1) 비용(Cost)/손실(loss) 함수

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

2) 비용 최소화를 위한 기법 : 경사하강법

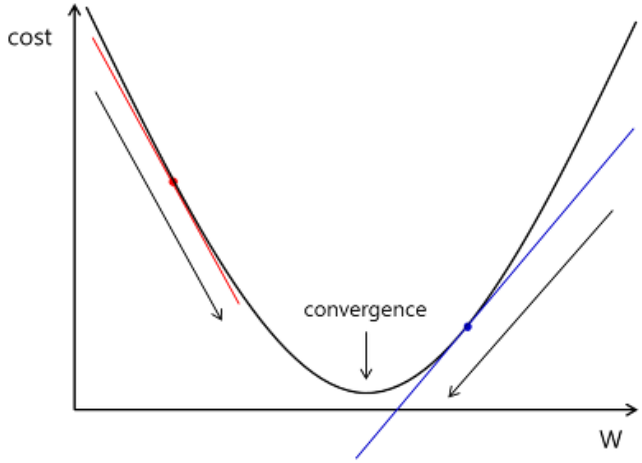
[경사하강법(Gradient descent algorithm) 이란?]

What cost(W) looks like?



cost를 줄이기 위해 변경되는 W의 파라미터의 상관관계를 그래프로 나타낸다면,
cost의 값이 최소가 되는 것은 W의 값이 가운데로 수렴하게 된다는 것.
(편의상 추가적으로 더하는 항인 바이어스의 값은 제외)

기울기가 0인 곳을 찾는 것이 경사하강법

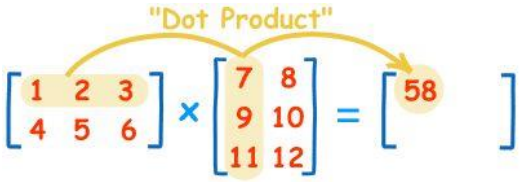


[(Multi-variable) linear regression]

H(x1,x2,x3) = (x1w1) + (x2w2) + (x3w3)

$$(x_1 \quad x_2 \quad x_3) \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = (x_1 w_1 + x_2 w_2 + x_3 w_3) \qquad H(X) = XW$$

행렬곱 예)



국어 영어 수학

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

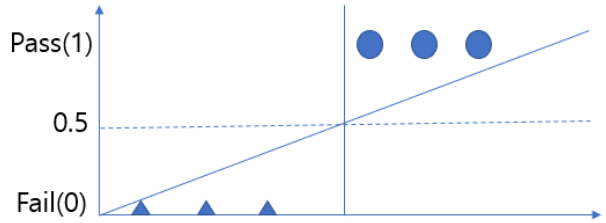
[5, 3] [3, 1] [5, 1]

Logistic Regression : 이진 분류 문제일 때 사용

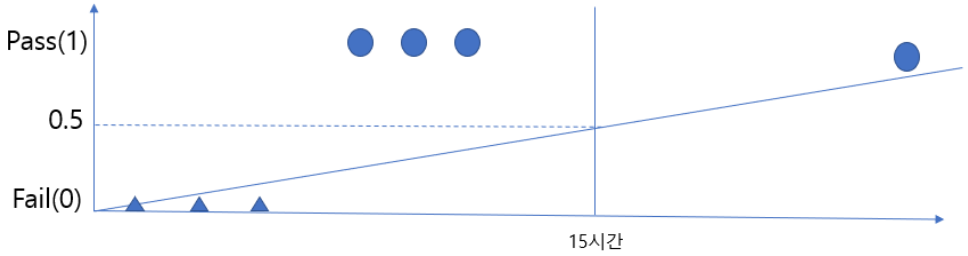
- sigmoid 함수, 새로운 cost 함수

< 선형 회귀로 분류 문제를 해결할 때의 문제점 >

예) 5시간 공부한 학생 - Pass (1)
7시간 공부한 학생 - Pass (1)
8시간 공부한 학생 - Pass (1)
1시간 공부한 학생 - Fail (0)
2시간 공부한 학생 - Fail (0)
4시간 공부한 학생 - Fail (0)

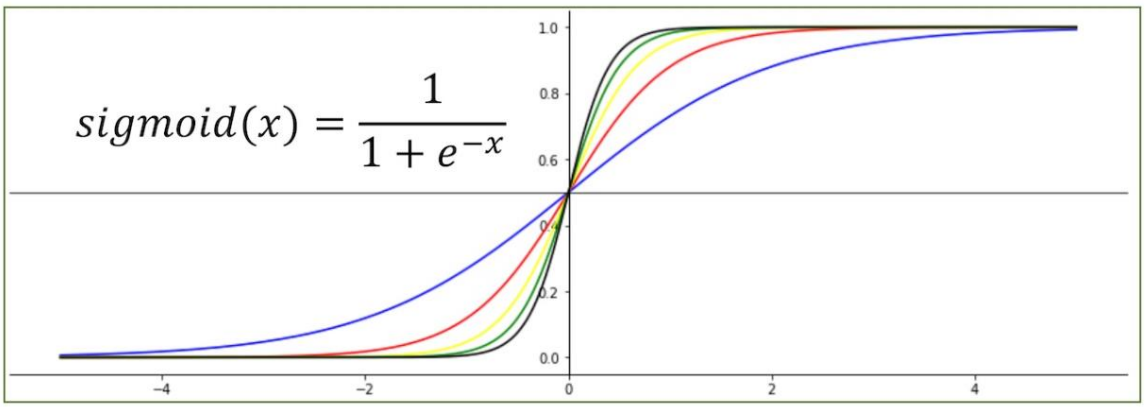


만약, 50 시간 공부한 학생이 있으면 ?



1) binary classification 의 결과는 0과 1이 필요한데,

H(x) = Wx + b 는 1보다 훨씬 큰 수가 나타날 수 있음.
H(x) 를 0~1 사이의 값으로 변경해주는 함수. –> sigmoid

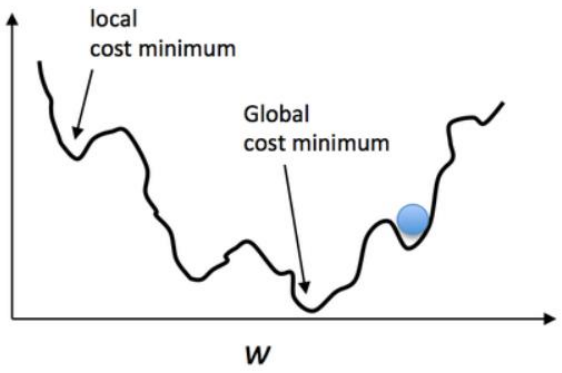


그래서, H(x)를 sigmoid 함수로 한번 더 계산. (H(x)를 0~1 사이의 값으로 변경)

$$\frac{1}{1+e^{-H(x)}}$$

sigmoid 함수의 특징은, X값이 아무리 크거나 작더라도 그 결과값은 최대가 1, 최소가 0이 됩니다.

2) sigmoid 함수를 통과한 값의 Cost 는 다음과 같이 구불구불한 모습으로 경사 하강법으로 최소 Cost 값을 찾을 때 문제가 발생할 수 있습니다. gradient descent algorithm 을 적용해도 우리가 필요로 하는 최종 값인 Global cost minimum 을 구하지 않고, local cost minimum 까지만 구할 수 있습니다.

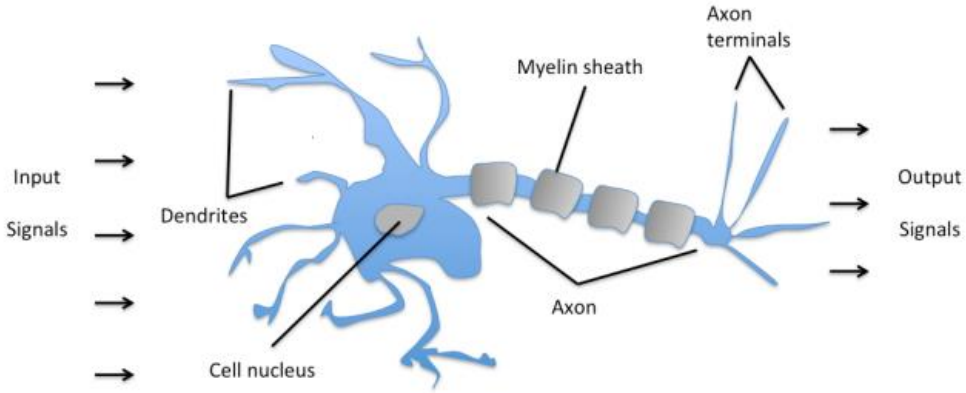


이것을 해결하기 위해서 cost 함수를 아래와 같이 바꾸어야 합니다. 새로운 cost 함수

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

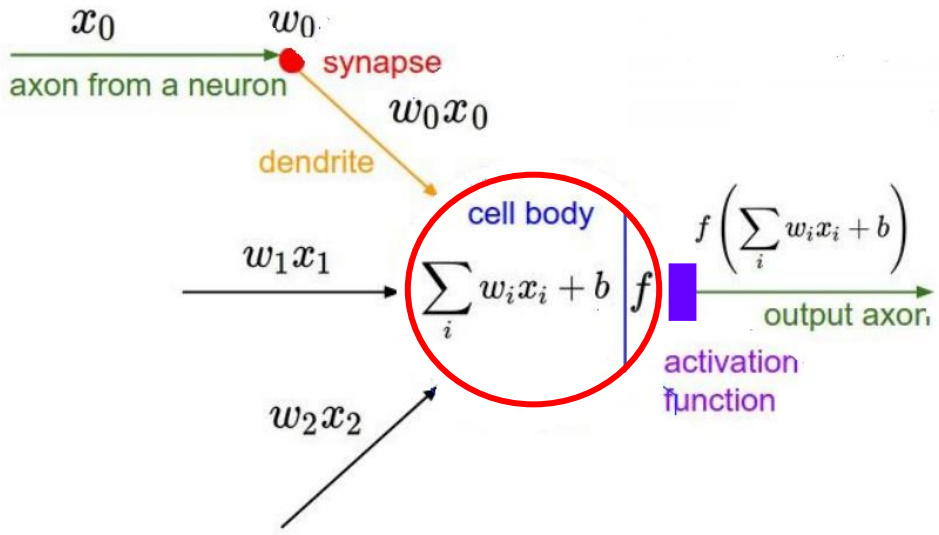
[신경망 구현]

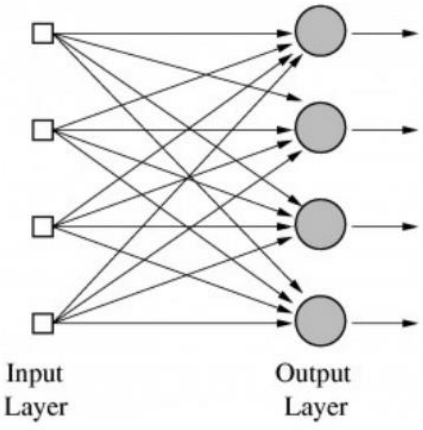
사람의 신경망



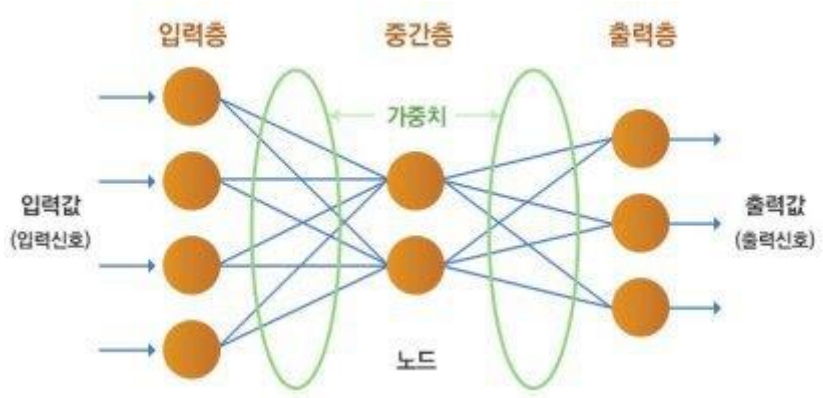
Schematic of a biological neuron.

Neural Network (인공 신경망)





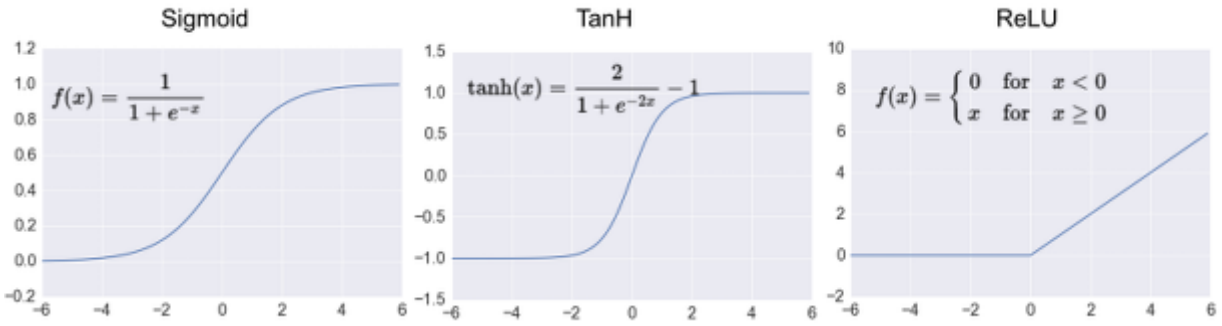
신경망의 구성



입력값(X)에 가중치(W)를 곱하고 편향(b)를 더한 뒤 활성함수 (Sigmode, ReLU 등)를 거쳐 결과값 y를 만들어내는 것이 인공 신경망의 기본입니다. 원하는 y값을 만들어내기 위해 W와 b 값을 변경해가면서 적절한 값을 찾아내는 최적화 과정을 학습 또는 훈련이라고 합니다.

Y = Sigmoid(WX + b)

뉴런은 입력이 누적되어 어떤 수준으로 커진 경우에만 출력을 합니다. 즉 입력 값이 어떤 **분계점(threshold)**에 도달해야 출력이 발생합니다. 따라서 뉴런은 미세한 잡음 신호 따위는 전달하지 않고 의미가 있는 신호만 전달하게 된다는 점에서도 직관적으로 이해할 수 있다. 이처럼 입력 신호를 받아 특정 분계점을 넘어서는 경우에 출력 신호를 생성해 주는 함수를 활성화 함수(activation function)라고 합니다. 활성화 함수의 종류는 매우 다양한데 그 중 Sigmoid (Logistic 함수라고도 불림) 는 단순하면서도 전통적으로 많이 쓰여 인공 신경망을 처음 공부할 때는 최고다. 현업에서 많이 쓰는 활성화 함수는 **ReLU**함수다.



뉴런은 한 개의 입력이 아닌, 여러 개의 입력을 받습니다. 값들을 모두 더해서 분계점을 넘으면 작동하게 되는데, 다른 입력 값이 작아도 하나만 커도 작동하게 됩니다. 이러한 생물학적 뉴런을 인공적으로 모델화하려면 뉴런을 여러 계층에 걸쳐 위치시키고, 각각의 뉴런은 직전 계층과 직후 계층에 있는 모든 뉴런들과 상호 연결되어 있는 식으로 표현하면 됩니다. 신경망 층이 둘 이상으로 구성된 딥 러닝이라고 합니다.각각의 인공 뉴런을 노드라고 부릅니다. 그리고 학습을 한다는 것은 연결의 강도를 조절해 나간다는 것입니다. 이 연결을 신경망에서는 가중치라고 부릅니다.

[Keras]

Keras는 TensorFlow와 theano의 High level API입니다.

Keras 설치 on Windows

1) Anaconda Prompt에서 다음 명령어를 실행합니다. Tensorflow 가 설치되어 있는 환경에서 설치합니다.

```
# Tensorflow 설치
>>>(base)C:> pip install tensorflow

# Keras 설치
>>>(base)C:> pip install keras
```

k_ex01.py Keras로 간단한 지도학습 머신러닝을 수행해봅니다. (선형 회귀)

```
import keras
import numpy

# 학습 데이터
x = numpy.array([0, 1, 2, 3, 4])
y = x * 2 + 1

# Keras 모델 시작
model = keras.models.Sequential()
# 인공지능 계층을 하나 추가 ( 입력 노드 하나와 가중치 하나)
model.add(keras.layers.Dense(1, input_shape=(1,)))
# SGD - 확률적 경사하강법,  mse - 평균 제곱 오차 cost function
model.compile('SGD', 'mse')

# 지도 학습  # 1000 에포크(학습 횟수), verbose 는 학습 진행 사항 보시여부, 0은 표지하지 않음
model.fit(x[:2], y[:2], epochs=1000, verbose=0)

print('Targets:', y[2:])  # 실제 결과
print('Predictions:', model.predict(x[2:]).flatten()) # 예측 결과
```

(출력 결과)

Targets: [5 7 9]

Predictions: [4.9625897 6.935519 8.908449]

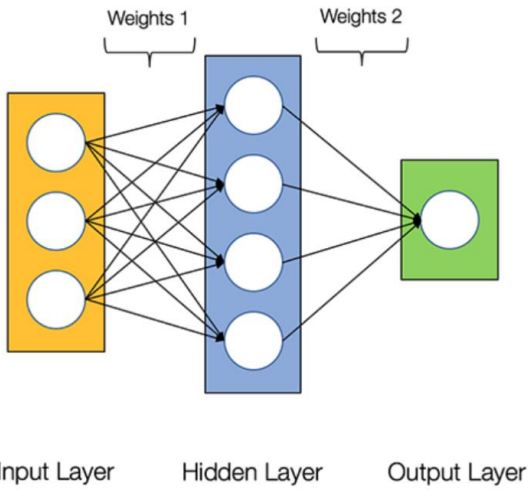
keras.layers.Dense 가 신경망 층을 하나 생성한다.

keras.layers.Dense(1 , input_shape=(,1)) ←- 출력특성 수 1 개, 입력 특성 수 : 1 개

ANN (Artificial Neural Network)

ANN은 넓은 의미로 인공신경망을 총칭하는 용어. 또는 얕은 신경망으로 구분

ANN 구조



회귀 ANN

아나콘다 콘솔에서 sklearn 설치합니다. (sklearn.preprocessing 의 MinMaxScalar를 사용하기 위하여)
(base) C:₩Windows₩system32>conda activate mlenv
(mlenv) C:₩Windows₩system32>pip install **sklearn**

데이터셋 Boston Housing에는 총 506건의 보스턴의 집 값과 관련된 13가지 정보가 담겨 있습니다.
학습 데이터셋과 평가 데이터셋으로 나눈 후, **13가지 정보로 집 값을 예측이** 가능하도록 학습합니다.
평가 데이터셋으로 집값 예측이 정확한지 평가합니다.

k_ex02.py

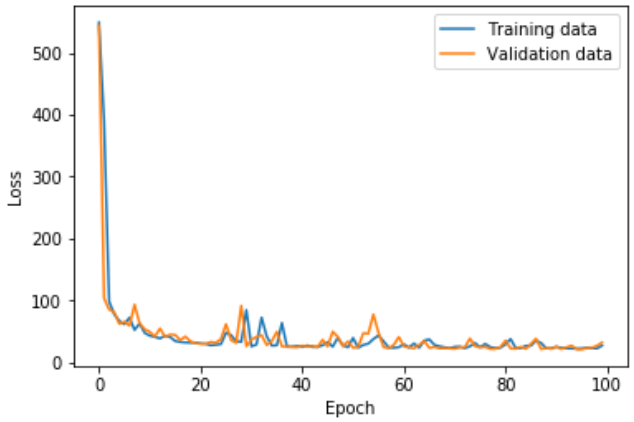
```
################################################
# Modeling
################################################

from tensorflow.python.keras import layers, models
from sklearn import preprocessing

def ANN_seq_func(Nin, Nh, Nout):
    model = models.Sequential()
    """ Keras 모델 시작 """

    model.add(layers.Dense(Nh, activation='relu', input_shape=(Nin,)))
    """입력 계층 노드 수 Nin 개,  은닉 계층의 노드 수 Nh 개, 활성함수는 relu  """

    model.add(layers.Dense(Nout, activation='relu'))
    """출력 노드 수 Nout 개,  활성함수는  relu) """

    model.compile(loss='mse',  optimizer='sgd' )
    """ cost함수 - mse - 평균 제곱 오차  최적화  알고리즘 -SGD(확률적 경사하강법)
    """
```

(계속)

```
##############################################
# Data  학습과 평가용 데이터 불러오기
##############################################
from keras import datasets

def Data_func():
    (X_train, y_train), (X_test, y_test) = datasets.boston_housing.load_data()
    scaler = preprocessing.MinMaxScaler()  # 데이터 정규화
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    return (X_train, y_train), (X_test, y_test)

##############################################
# Plotting  결과 그래프 구현
##############################################
import matplotlib.pyplot as plt


##############################################
# Main  회귀 ANN 학습 및 성능 분석
##############################################
def main():
    Nin = 13
    Nh = 5
    Nout = 1

    model = ANN_seq_func(Nin, Nh, Nout)
    (X_train, y_train), (X_test, y_test) = Data_func()

    history = model.fit(X_train, y_train, epochs=100, batch_size=100, \
                        validation_split=0.2, verbose=2)

    performace_test = model.evaluate(X_test, y_test, batch_size=100)
    print('\nTest Loss -> {:.2f}'.format(performace_test))

    history = history.history

    """Cost/Loss  변화 추이 그래프"""
    plt.plot(history['loss'])
    plt.plot(history['val_loss'])
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Training', 'Verification'], loc=0)
    plt.show()


if __name__ == '__main__':
    main()
```

(출력결과)

```
Epoch 1/100
 - 0s - loss: 474.0749 - val_loss: 105.4658
Epoch 99/100
 - 0s - loss: 22.1986 - val_loss: 26.6115
Epoch 100/100
 - 0s - loss: 26.8277 - val_loss: 31.8782
102/102 [==============================] - 0s 0us/step

Test Loss -> 30.13
```



(코드 상세 설명)

def ANN_seq_func(Nin, Nh, Nout): # 코드 설명

```
    model.add(layers.Dense(Nh, activation='relu', input_shape=(Nin,)))
    """입력 계층 노드 수 Nin 개,  은닉 계층의 노드 수 Nh 개, 활성함수는 relu  """

    model.add(layers.Dense(Nout, activation='relu'))
    """출력 노드 수 Nout 개,  활성함수는  relu) """

    model.compile(loss='mse',  optimizer='sgd' )
    """ cost 함수 - mse – 평균 제곱 오차(mean squar error)
        최적화  알고리즘 -SGD(확률적 경사하강법)
    """
```

def Data_func(): 코드 설명

```
    (X_train, y_train), (X_test, y_test) = datasets.boston_housing.load_data()
                총 506건의 보스턴 집 값과 관련된 13가지 정보.
                학습 데이터와 테스트 데이터로 읽어옵니다
    scaler = preprocessing.MinMaxScaler()  # 데이터 정규화. (0~1 사이의 값으로)
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    return (X_train, y_train), (X_test, y_test)
```

def main(): : 코드 설명

```
        Nin = 13     #입력층 13개
        Nh = 5       #은닉층 5개
        Nout = 1     #출력층 1개

        model = ANN_seq_func (Nin, Nh, Nout)
        (X_train, y_train), (X_test, y_test) = Data_func()
```

```
history = model.fit(X_train, y_train, epochs=100, batch_size=100, ₩
                    validation_split=0.2, verbose=2)
                    학습 데이터를 100개씩 학습, 총 데이터셋 학습을 15번 반복(epochs=15).
                    검증 데이터셋 비율은 0.2 사용
performace_test = model.evaluate(X_test, y_test, batch_size=100)
          평가 데이터를 100개씩 평가

print('₩nTest Loss -> {:.2f}'.format(performace_test))

plot_loss(history) # Cost 변화 그래프
plt.show()
```