Week 02

Classification

#5 Review/ Regression with Pytorch

#6 Binary / Multi-Label Classification

#7 Multi-Label Classification with Pytorch

2021. 07. 20. TUE BOAZ 분석 D조 17기 곽윤경

Classification

데이터가 입력됐을 때 지도학습을 통해 미리 학습된 레이블 중 하나 또는 여러 개의 레이블로 예측하는 것

	supervised	unsupervised	reinforcement
discrete	classification	clustering	Discrete Action space agent
continuous	regression	Dimensionality reduction	Continuous Action space agent

Classification

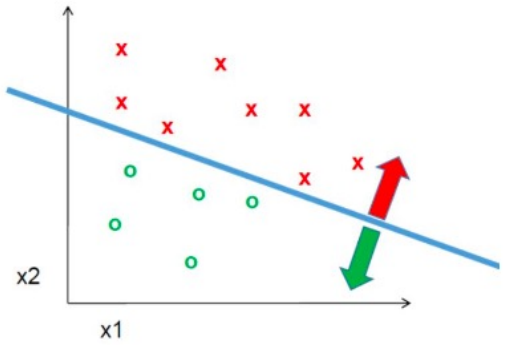
Binary Classification

둘 중 하나의 값으로 분류하는 경우

Example

(예, 아니오), (남자, 여자)
정상메일인가? 스팸메일인가?
합격인가? 불합격인가?

"**로지스틱 회귀**"

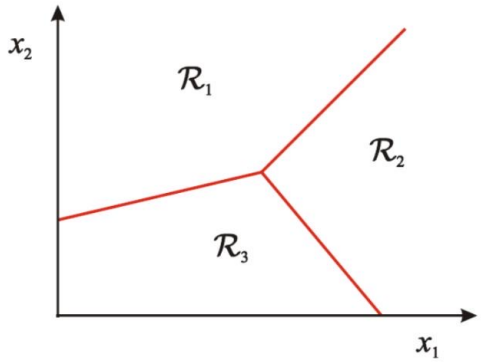


Multinomial Classfication

두 개 이상의 클래스 중에서
하나의 값으로 분류하는 경우

Example

어떤 사진이 개?고양이?책상?사람?
어떤 책이 만화?과학?학습지?



Logistic Regression

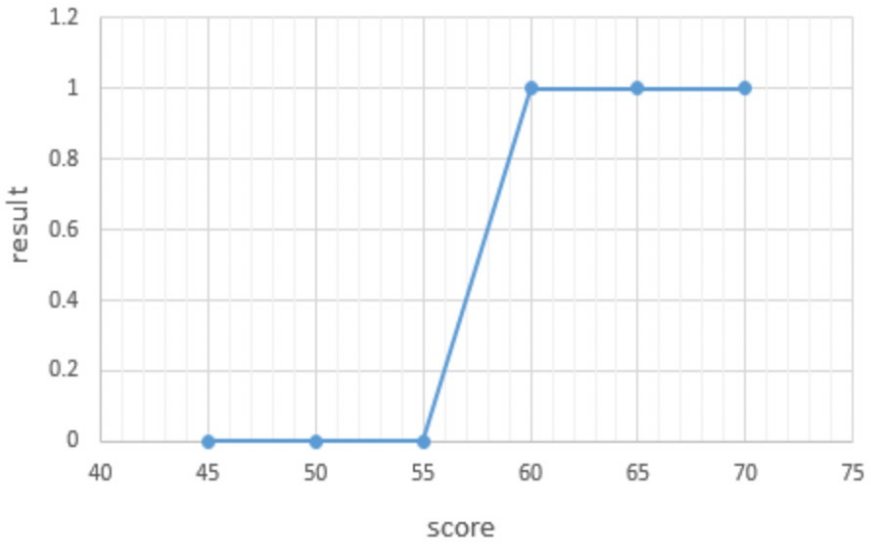
입력 X에 대해 결과 Y가 0~1 사이 값을 출력한다.
값이 0에 가까우면 a로 판단하고, 값이 1에 가까우면 b로 판단한다.

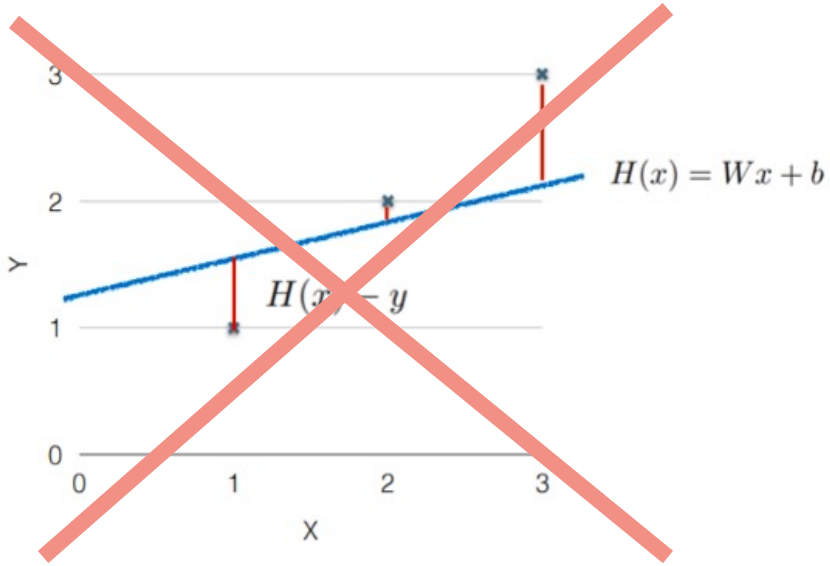
Y=0.21
79% 확률로 고양이!

Y=0.83
83% 확률로 강아지!

Binary Classification

Score (X)	result (Y)
45	불합격
50	불합격
55	불합격
60	합격
65	합격
70	합격





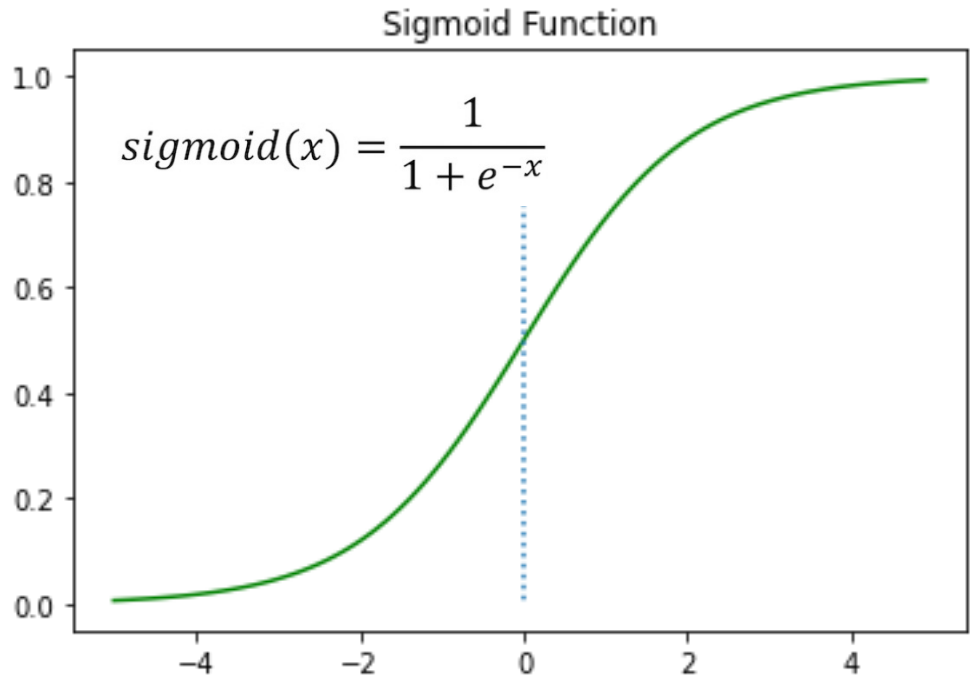
Hypothesis

Sigmoid Function

$$H(X) = \frac{1}{1 + e^{-(Wx+b)}} = sigmoid(Wx + b) = \sigma(Wx + b)$$

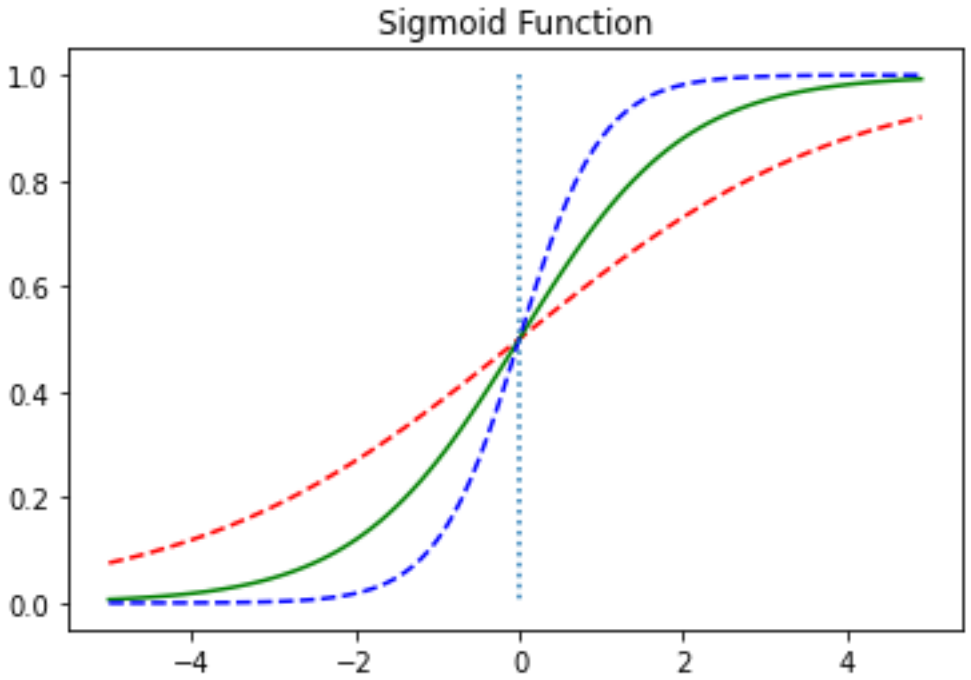
```
[1]  %matplotlib inline
     import numpy as np # 넘파이 사용
     import matplotlib.pyplot as plt # 맷플롯립 사용
```

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)

plt.plot(x, y, 'g')
plt.plot([0,0],[1.0,0.0], ':') # 가운데 점선 추가
plt.title('Sigmoid Function')
plt.show()
```



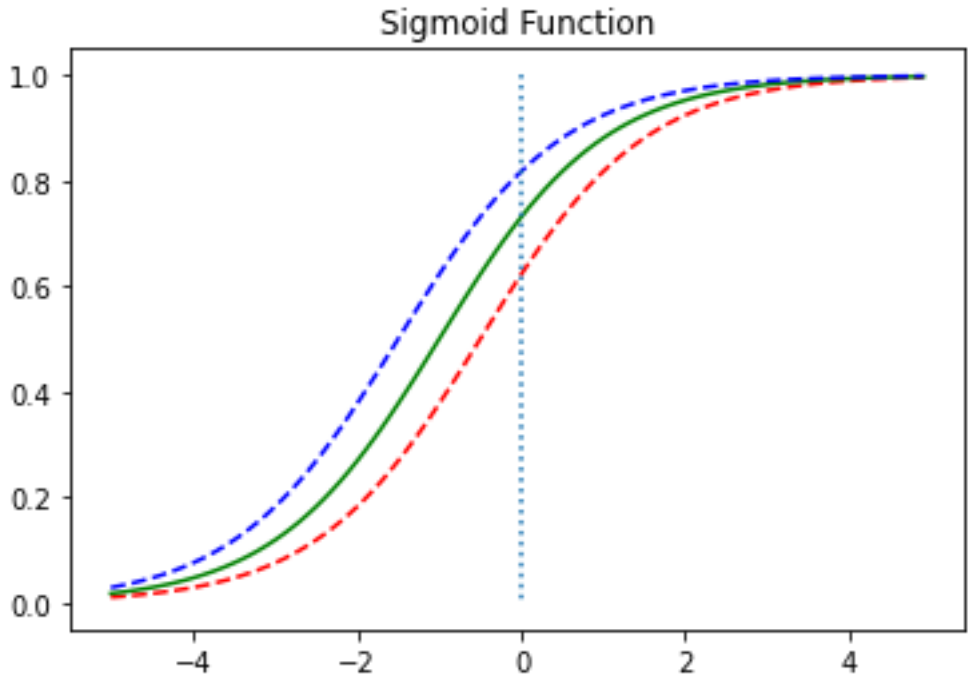
Sigmoid Function

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
x = np.arange(-5.0, 5.0, 0.1)
y1 = sigmoid(0.5*x)
y2 = sigmoid(x)
y3 = sigmoid(2*x)

plt.plot(x, y1, 'r', linestyle='--') # W의 값이 0.5일때
plt.plot(x, y2, 'g') # W의 값이 1일때
plt.plot(x, y3, 'b', linestyle='--') # W의 값이 2일때
plt.plot([0,0],[1.0,0.0], ':') # 가운데 점선 추가
plt.title('Sigmoid Function')
plt.show()
```



Sigmoid Function

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
x = np.arange(-5.0, 5.0, 0.1)
y1 = sigmoid(x+0.5)
y2 = sigmoid(x+1)
y3 = sigmoid(x+1.5)

plt.plot(x, y1, 'r', linestyle='--') # x + 0.5
plt.plot(x, y2, 'g') # x + 1
plt.plot(x, y3, 'b', linestyle='--') # x + 1.5
plt.plot([0,0],[1.0,0.0], ':') # 가운데 점선 추가
plt.title('Sigmoid Function')
plt.show()
```



Cost Function

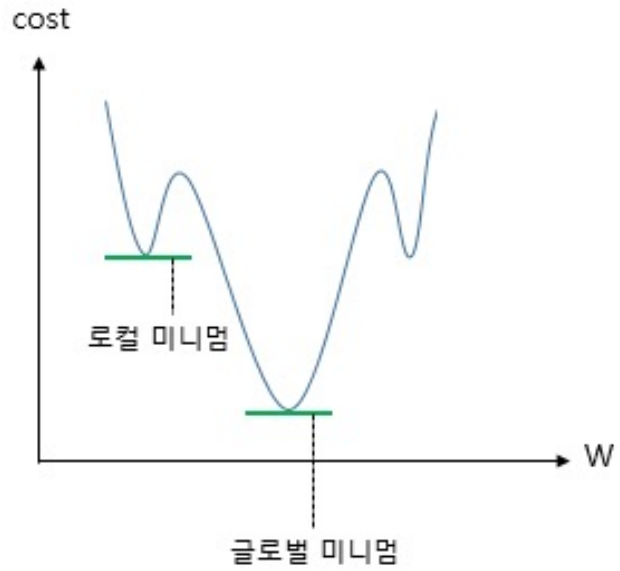
학습 현재 가지고 있는 데이터를 통해 최적의 W를 찾아내는 과정

Linear regression

랜덤한 W를 최초 부여한 후, 경사하강법으로 평균제곱오차가 가장 작은 W를 찾는다.

Logistic regression

경사하강법으로 최적의 W를 찾아내지만, 비용함수는 크로스 엔트로피를 사용한다.



Cross Entropy

모델의 예측값과 실제값 확률의 차이

예측값과 실제값의 차이를 가장 작게하는 최적의 W를 구하는 방법

$$H(p,q) = -\sum_{c=1}^{C} p(c)\log q(c)$$

p(c): 실제 데이터 분포, q(c): 모델의 예측값 분포

불일치

실제 데이터 [0,1]
모델 예측값 [1,0]

$$[0,1] \times \begin{bmatrix} \log 1 \\ \log 0 \end{bmatrix} = \infty$$

일 치

실제 데이터 [0,1]
모델 예측값 [0,1]

$$[0,1] \times \begin{bmatrix} \log 0 \\ \log 1 \end{bmatrix} = 0$$

Binary Classification

```
[1]  from keras.models import Sequential
     from keras.layers import Dense, Activation
     import numpy as np
```

```
[6]  model= Sequential()
     #입력 1개를 받아 출력 1개를 리턴하는 선형 회귀 레이어 생성
     model.add(Dense(input_dim=1, units=1))
     #선형 회귀의 출력값을 시그모이드에 연결
     model.add(Activation('sigmoid'))
     #크로스 엔트로피를 비용함수로 설정해 경사하강법으로 학습
     model.compile(loss='binary_crossentropy',optimizer='sgd',metrics=['binary_accuracy'])
```

모델 학습을 위한 데이터 생성

```
[7]  x= np.array([-2,-1.5,-1,1.25,1.62,2])
     y= np.array([0,0,0,1,1,1])
```

모델 학습 진행(300번의 반복학습으로 최적의 W,b 찾기)

```
[8]  model.fit(x,y,epochs=300, verbose=0)
```

```
     <keras.callbacks.History at 0x7f4260bf5050>
```

Binary Classification

예측값 확인

```
[9]  model.predict([-2,-1.5,-1,1.25,1.62,2])
```

```
array([[0.02349776],
       [0.05720928],
       [0.13271242],
       [0.90763205],
       [0.9511779 ],
       [0.9752164 ]], dtype=float32)
```

```
[ ]  최적의 W,b는 get_weights() 함수로 확인
```

```
[10] model.layers[0].get_weights()
```

```
[array([[1.8498867]], dtype=float32), array([-0.02729927], dtype=float32)]
```

Multinomial Classification

One- Hot Encoding

데이터가 어느 클래스에 속하는지 이진수로 매핑하는 방법

고양이	강아지	사람	앵무새
0	0	1	0

고양이	강아지	사람	앵무새
1	0	0	0

고양이	강아지	사람	앵무새
0	1	0	0

[0,0,0,1] ?

Hypothesis

Softmax Function

k차원의 벡터를 입력받아 각 클래스에 대한 확률을 추정하는 함수

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad for\ i = 1, 2, \ldots k$$

$$softmax(z) = [\frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_2}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_3}}{\sum_{j=1}^{3} e^{z_j}}] = [p_1, p_2, p_3] = \hat{y} = \text{예측값}$$

Multinomial Classification

```
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils import to_categorical
from keras.datasets import mnist
```

```
Using TensorFlow backend.
```

MNIST 손글씨 데이터를 다운로드 받아서 변수에 저장합니다.

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

손글씨 데이터(X_train, X_test)가 가로 28픽셀, 세로 28픽셀로 구성된 것을 확인할 수 있습니다.
학습에 사용될 X_train은 총 60000개의 데이터, 테스트에 사용될 X_test는 총 10000개의 데이터가 있습니다.

```
print("train data (count, row, column) : " + str(X_train.shape) )
print("test data  (count, row, column) : " + str(X_test.shape) )
```

```
train data (count, row, column) : (60000, 28, 28)
test data  (count, row, column) : (10000, 28, 28)
```

Multinomial Classification

```
print(X_train[0])
```

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
```

모델 학습 시작에 앞서, 데이터를 정규화합니다.
정규화는 입력값을 0부터 1의 값으로 변경하게 됩니다.
정규화된 입력값은 경사하강법으로 모델 학습 시, 보다 쉽고 빠르게 최적의 W,B를 찾는 데 도움을 줍니다.

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

```
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.01176471 0.07058824 0.07058824 0.07058824 0.49411765 0.53333336
  0.6862745  0.10196079 0.6509804  1.         0.96862745 0.49803922
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.11764706 0.14117648 0.36862746 0.6039216
  0.6666667  0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
  0.88235295 0.6745098  0.99215686 0.9490196  0.7647059  0.2509804
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.19215687 0.93333334 0.99215686 0.99215686 0.99215686
  0.99215686 0.99215686 0.99215686 0.99215686 0.99215686 0.9843137
  0.3647059  0.32156864 0.32156864 0.21960784 0.15294118 0.
  0.         0.         0.         0.        ]
```

Multinomial Classification

y_train은 총 6만개, y_test는 총 1만개의 숫자를 가지고 있습니다.

```
print("train target (count) : " + str(y_train.shape) )
print("test target  (count) : " + str(y_test.shape) )
```

```
train target (count) : (60000,)
test target  (count) : (10000,)
```

아래의 코드를 실행하여, y_train과 y_test에서 샘플로 숫자를 출력해봅니다.

```
print("sample from train : " + str(y_train[0]) )
print("sample from test : " + str(y_test[0]) )
```

```
sample from train : 5
sample from test : 7
```

이번 실습에서는 28*28 픽셀의 지역적인 정보를 사용하지 않고, 단순히 정규화된 입력값만을 가지고,
숫자 분류를 할 것이기 때문에, 행과 열의 구분 없이, 단순히 784 길이의 배열로 데이터를 단순화시킵니다.

```
input_dim = 784 #28*28
X_train = X_train.reshape(60000, input_dim)
X_test = X_test.reshape(10000, input_dim)
```

Multinomial Classification

학습 시, y값과의 cross entropy를 측정해야하므로, 아래의 코드를 실행하여 y를 one hot encoding으로 변환시켜줍니다.

```
num_classes = 10
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
```

아래 코드를 실행하여, 5였던 값이, one hot encoding으로 변환되어,
클래스 갯수만큼의 길이를 갖는 벡터로 변경이 되었고, 5에 해당되는 인덱스의 값만 1인 것을 확인할 수 있습니다.

```
print(y_train[0])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

케라스의 Sequential()을 사용하여 간단하게 소프트맥스를 구현할 수 있습니다.
총 784개(28*28)의 입력을 받아서, 10개의 시그모이드 값을 출력하는 모델을 아래의 코드를 실행하여 구현합니다.

```
model = Sequential()
model.add(Dense(input_dim=input_dim, units = 10, activation='softmax'))
```

모델의 학습을 진행합니다.
10개의 클래스로 분류할 것이기 때문에, categorical_crossentropy를 비용함수로 사용한 경사하강법으로 최적의 W와 biases를 학습합니다.

```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=2048, epochs=100, verbose=0)
```

Multinomial Classification

```
score = model.evaluate(X_test, y_test)
print('Test accuracy:', score[1])
```

```
10000/10000 [==============================] - 1s 50us/step
Test accuracy: 0.8927
```

아래의 코드를 실행하여, 소프트맥스 모델의 구조를 쉽게 시각화 할 수 있습니다.
총 10개의 로지스틱회귀가 있고, 각 로지스틱회귀는 784개의 weight와 1개의 bias를 갖고 있기 때문에,
총 7850 (785*10)개의 Param이 있는 것을 보실 수 있습니다.

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 10)                7850      
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

첫번째 레이어에 존재하는 w1, w2,...,w784, b1, b2,..., b10은 아래의 명령어로 확인하실 수 있습니다.

```
model.layers[0].weights
```

```
[<tf.Variable 'dense_1/kernel:0' shape=(784, 10) dtype=float32_ref>,
 <tf.Variable 'dense_1/bias:0' shape=(10,) dtype=float32_ref>]
```

Thank you!

참 고 자 료

나의 첫 머신러닝/딥러닝, 위키북스, 허민석
https://wikidocs.net/22881
https://www.youtube.com/watch?v=RvIf-POuZ4Y