



– Logistic Regression –

박성호 (neowizard2018@gmail.com)

Overview – Training Data (Diabetes.csv)

임신 횟수 포도당 농도 이완기 혈압 삼두근 피부두께 2시간 인슐린 체질량 당뇨 직계 가족력 나이 5년내 당뇨병 발병여부

-0.29412	0.487437	0.180328	-0.29293	0	0.00149	-0.53117	-0.03333	0
-0.88235	-0.14573	0.081967	-0.41414	0	-0.20715	-0.76687	-0.66667	1
-0.05882	0.839196	0.04918	0	0	-0.30551	-0.49274	-0.63333	0
-0.88235	-0.10553	0.081967	-0.53535	-0.77778	-0.16244	-0.924	0	1
0	0.376884	-0.34426	-0.29293	-0.60284	0.28465	0.887276	-0.6	0

759

.....

-0.88235	0.899497	-0.01639	-0.53535	1	-0.10283	-0.72673	0.266667	0
-0.17647	0.005025	0	0	0	-0.10581	-0.65329	-0.63333	0
0	0.18593	0.377049	-0.05051	-0.45627	0.365127	-0.59607	-0.66667	0
-0.17647	0.075377	0.213115	0	0	-0.11774	-0.8497	-0.66667	0

Logistic Regression Example

```
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import SGD, Adam
```

[1] 데이터 생성

```
import numpy as np

try:

    loaded_data = np.loadtxt('./diabetes.csv', delimiter=',')

    x_data = loaded_data[:, 0:-1]
    t_data = loaded_data[:, [-1]]

    print("x_data.shape = ", x_data.shape)
    print("t_data.shape = ", t_data.shape)

except Exception as err:

    print(str(err))
```

```
x_data.shape = (759, 8)
t_data.shape = (759, 1)
```

[2] 모델 구축

```
model = Sequential()

model.add(Dense(t_data.shape[1],
                input_shape=(x_data.shape[1], ), activation='sigmoid'))
```

활성화 함수
↓

[3] 모델 컴파일

```
model.compile(optimizer=SGD(learning_rate=0.01),
              loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

학습 알고리즘
↑
손실 함수
↑
측정 지표

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1)	9

Total params: 9

Trainable params: 9

Non-trainable params: 0

Logistic Regression Example

[4] 모델 학습

training data로 부터 20% 비율로
validation data 생성 후 overfitting 확인

```
hist = model.fit(x_data, t_data, epochs=500, validation_split=0.2, verbose=2)
```

Epoch 1/500

19/19 - 0s - loss: 0.6041 - accuracy: 0.6689 - val_loss: 0.6042 - val_accuracy: 0.6250

Epoch 2/500

19/19 - 0s - loss: 0.6028 - accuracy: 0.6705 - val_loss: 0.6030 - val_accuracy: 0.6250

.....

Epoch 499/500

19/19 - 0s - loss: 0.4785 - accuracy: 0.7743 - val_loss: 0.4885 - val_accuracy: 0.7500

Epoch 500/500

19/19 - 0s - loss: 0.4785 - accuracy: 0.7743 - val_loss: 0.4885 - val_accuracy: 0.7500

[7] 모델 (정확도) 평가

```
model.evaluate(x_data, t_data)
```

기본 batch_size = 32 이므로 총 24회 반복 (759 / 32 = 23.7)

24/24 [=====] - 0s 2ms/step - loss: 0.4825 - accuracy: 0.7655
[0.4825364351272583, 0.7654808759689331]

Logistic Regression Example

[5] 손실 및 정확도 추세

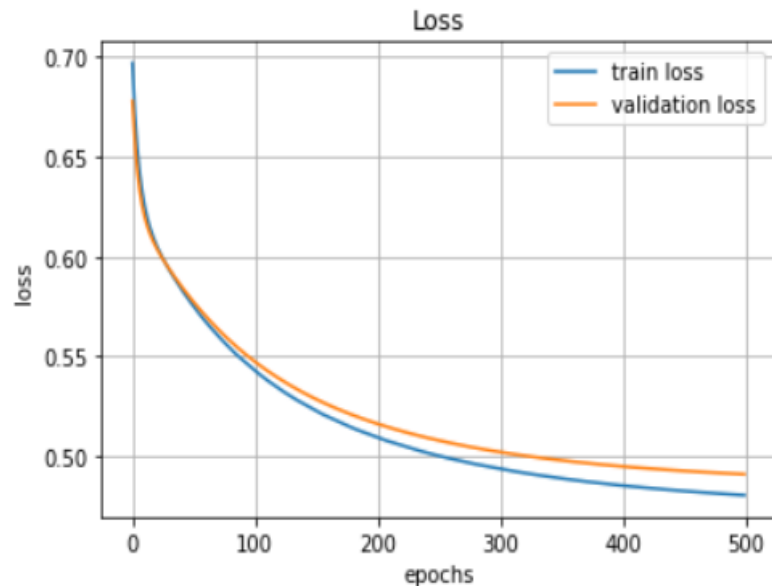
```
import matplotlib.pyplot as plt

plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['loss'], label='train loss')
plt.plot(hist.history['val_loss'], label='validation loss')

plt.legend(loc='best')

plt.show()
```



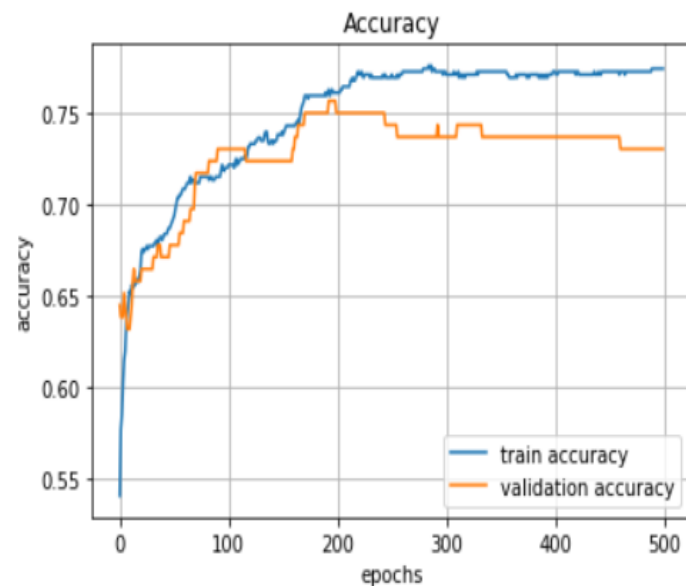
```
import matplotlib.pyplot as plt

plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(hist.history['accuracy'], label='train accuracy')
plt.plot(hist.history['val_accuracy'], label='validation accuracy')

plt.legend(loc='best')

plt.show()
```



[예제 1] multi variable linear regression (Sequential / Functional)

입력			정답
1	2	0	-4
5	4	3	4
1	2	-1	-6
3	1	0	3
2	4	2	-4
4	1	2	9
-1	3	2	-7
4	3	3	5
0	2	6	6
2	2	1	0
1	-2	-2	4
0	1	3	3
1	1	3	5
0	1	4	5
2	3	3	1

[Q1] training data 입력과 정답에는 어떤 관계가 있는가 ?

[Q2] TensorFlow 이용하여, training data를 학습한 후에,
loss trend 출력하고 다음의 test data 결과를 예측하시오

(5, 5, 0) , (2, 3, 1) , (-1, 0, -1) , (10, 5, 2) , (4, -1, -2)

Sequential API 와 Functional API 이용하여 Keras 모델을 구축하시오



입력층 노드: 4개,

은닉층1 노드: 8개, activation function = 'relu'

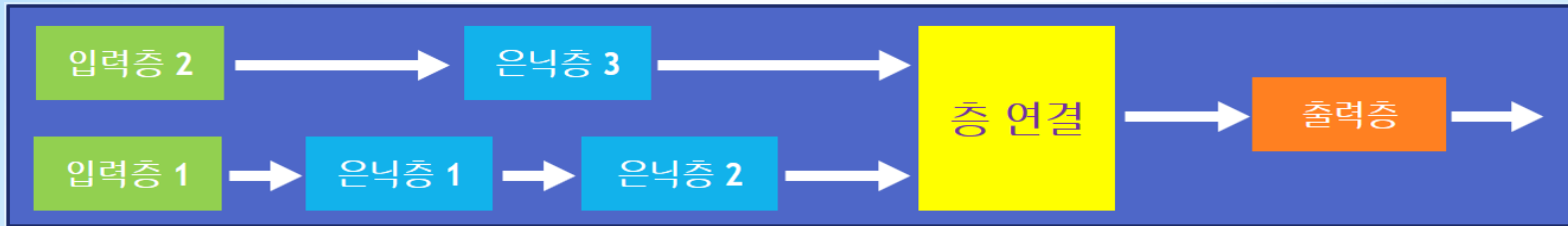
은닉층2 노드: 16개, activation function = 'relu'

은닉층3 노드: 32개, activation function = 'relu'

출력층 노드: 10개, activation function = 'softmax'

`model.summary()` 이용하여 shape, total parameter 확인

Functional API 이용하여 모델을 구축하시오 (<https://youtu.be/KugOEdth3iE>)



입력층1 노드: 4개,

은닉층1 노드: 8개, activation function = 'relu'

은닉층2 노드: 16개, activation function = 'relu'

입력층2 노드: 8개,

은닉층3 노드: 8개, activation function = 'relu'

출력층 노드: 10개, activation function = 'softmax'

층연결 층은 `tensorflow.keras.layers.concatenate([A, B])` 형식으로 이용함

`model.summary()` 이용하여 shape, total parameter 확인