

```
# Lab 7 Learning rate and Evaluation
import tensorflow as tf
```

```
learning_rate = 0.001
```

```
batch_size = 100
```

```
training_epochs = 15
```

```
nb_classes = 10 → 0~9 : 10개
```

```
mnist = tf.keras.datasets.mnist
```

→ 손으로 쓰인 이미지들을 처리하는 데이터셋

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
# normalizing data
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

→ MNIST 데이터셋을 인터넷을 통해 가져옴

(60000, 28, 28)
[0] [1] [2]

원래 데이터를 변경하지 않고 반환됨

결과를 사용하여 데이터의 shape을 변경해줌

```
# change data shape
```

```
print(x_train.shape) # (60000, 28, 28)
```

```
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1] * x_train.shape[2])
```

```
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1] * x_test.shape[2])
```

→ (60000, 28x28)

→ x_train 에는 총 60000개의 28x28 크기의 이미지가 담겨있음

```
# change result to one-hot encoding
```

```
# in tf1, one_hot= True in read_data_sets("MNIST_data/", one_hot=True)
```

```
# took care of it, but here we need to manually convert them
```

```
y_train = tf.keras.utils.to_categorical(y_train, 10)
```

```
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

→ x_train의 60000 개에 대한 값(0~9)이 담겨있는 레이블 데이터셋

```
# # Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
```

```
# array([0, 2, 1, 2, 0])
```

```
# 'to_categorical' converts this into a matrix with as many columns as there are classes. The number of rows
```

```
# stays the same. to_categorical(labels)
```

```
# array([[ 1.,  0.,  0.],
```

```
#       [ 0.,  0.,  1.],
```

```
#       [ 0.,  1.,  0.],
```

```
#       [ 0.,  0.,  1.],
```

```
#       [ 1.,  0.,  0.]], dtype=float32)
```

0~9

28x28

```
tf.model = tf.keras.Sequential()
```

```
tf.model.add(tf.keras.layers.Dense(units=10, input_dim=784, activation='softmax'))
```

```
tf.model.compile(loss='categorical_crossentropy', optimizer=tf.optimizers.Adam(0.001),
```

```
metrics=['accuracy'])
```

```
tf.model.summary()
```

```
history = tf.model.fit(x_train, y_train, batch_size=batch_size, epochs=training_epochs)
```

```
predictions = tf.model.predict(x_test)
```

```
print('Prediction: \n', predictions)
```

```
x_train
```

```
score = tf.model.evaluate(x_train, y_train)
```

```
print('Accuracy: ', score[1])
```