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

x_data = [[1, 2, 1],
        [1, 3, 2],
        [1, 3, 4],
        [1, 5, 5],
        [1, 7, 5],
        [1, 2, 5],
        [1, 6, 6],
        [1, 7, 7]]
y_data = [[0, 0, 1],
        [0, 0, 1],
        [0, 0, 1],
        [0, 1, 0],
        [0, 1, 0],
        [0, 1, 0],
        [1, 0, 0],
        [1, 0, 0]]
```

training datasets

```
# Evaluation our model using this test dataset
x_test = [[2, 1, 1],
        [3, 1, 2],
        [3, 3, 4]]
y_test = [[0, 0, 1],
        [0, 0, 1],
        [0, 0, 1]]
```

test datasets

```
# try different learning_rate
# learning_rate = 65535  # ? it works too hahaha
learning_rate = 0.1
# learning_rate = 1e-10  # small learning rate won't work either

tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=3, input_dim=3, activation='softmax'))
tf.model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.SGD(lr=learning_rate),
metrics=['accuracy'])

tf.model.fit(x_data, y_data, epochs=1000)
```

↳ training data set으로 학습을 진행

```
# predict
print("Prediction: ", tf.model.predict_classes(x_test))

# Calculate the accuracy
print("Accuracy: ", tf.model.evaluate(x_test, y_test)[1])
```

↳ test data set으로 정확도를 측정함

```
import tensorflow as tf
import numpy as np
import tensorflow as tf
import numpy as np

xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])

x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=1, input_dim=4))
tf.model.add(tf.keras.layers.Activation('linear'))
tf.model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(lr=1e-5))
tf.model.summary()

history = tf.model.fit(x_data, y_data, epochs=100)

print(history.history['loss']) # loss == nan
```

값이 지나치게 커서 함수가 한
쪽으로 치우치게 되는 현상이 발생
: 입력값 정규화가 필요

*NaN : ∞ (발산)

↳ 학습 포기

```
def min_max_scaler(data):
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    # noise term prevents the zero division
    return numerator / (denominator + 1e-7)
```

* MinMaxScaler ( )

: 최소값=0 , 최대값=1 로 놓고
그 사이 값들을 normalize 함

→ 입력값을 Normalize 하기 위해 사용함

```
xy = np.array(
    [
        [828.659973, 833.450012, 908100, 828.349976, 831.659973],
        [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
        [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
        [816, 820.958984, 1008100, 815.48999, 819.23999],
        [819.359985, 823, 1188100, 818.469971, 818.97998],
        [819, 823, 1198100, 816, 820.450012],
        [811.700012, 815.25, 1098100, 809.780029, 813.669983],
        [809.51001, 816.659973, 1398100, 804.539978, 809.559998],
    ]
)

# very important. It does not work without it.
xy = min_max_scaler(xy)
print(xy)

'''
[[0.99999999 0.99999999 0.         1.         1.        ]
 [0.70548491 0.70439552 1.         0.71881782 0.83755791]
 [0.54412549 0.50274824 0.57608696 0.606468   0.6606331 ]
 [0.33890353 0.31368023 0.10869565 0.45989134 0.43800918]
 [0.51436    0.42582389 0.30434783 0.58504805 0.42624401]
 [0.49556179 0.42582389 0.31521739 0.48131134 0.49276137]
 [0.11436064 0.         0.20652174 0.22007776 0.18597238]
 [0.         0.07747099 0.5326087  0.         0.        ]]
'''

x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=1, input_dim=4))
tf.model.add(tf.keras.layers.Activation('linear'))
tf.model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(lr=1e-5))
tf.model.summary()

history = tf.model.fit(x_data, y_data, epochs=1000)

predictions = tf.model.predict(x_data)
score = tf.model.evaluate(x_data, y_data)

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