Second Programming Exam.
Deep Learning, 2022

姓名：吳佳展　　學號：110318517

```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import LSTM, Dropout, Dense, Activation, GRU
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping


def Get_data():
	df_train = pd.read_csv('predict2_trn.csv')
	df_test = pd.read_csv('predict2_tst.csv')
	X_train, y_train = df_train.iloc[:,:12], df_train.iloc[:,-4:]
	X_test = df_test.iloc[:,:12]
	return X_train, y_train, X_test

def Normalize():
	df_train = pd.read_csv('predict2_trn.csv')
	scaler = MinMaxScaler(feature_range=(0,1))
	scaled = scaler.fit_transform(df_train)
	X_scaled, y_scaled = scaled[:,:12], scaled[:,-4:]
	return scaler, X_scaled, y_scaled

def Build_Model(X_train, y_train):
	# The GRU architecture
	regressorGRU = Sequential()
	# First GRU layer with Dropout regularisation
	regressorGRU.add(GRU(units=50, return_sequences=True, 
input_shape=(X_train.shape[1],1)))
	regressorGRU.add(Dropout(0.2))
	# Second GRU layer
	regressorGRU.add(GRU(units=50, return_sequences=True, 
input_shape=(X_train.shape[1],1)))
	regressorGRU.add(Dropout(0.2))
	# Third GRU layer
```

```
    regressorGRU.add(GRU(units=50))
    regressorGRU.add(Dropout(0.2))
    # The output layer
    regressorGRU.add(Dense(y_train.shape[1]))
    regressorGRU.summary()
    # Compiling the RNN
    # regressorGRU.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9,
nesterov=False),loss='mean_squared_error', metrics=['mse'])
    regressorGRU.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mse'])
    return regressorGRU


def Train_Model(X, y):
    re_X = X.reshape(X.shape[0], X.shape[1],1)
    file1 = 'Q3/predict2.h5'#儲存訓練的最小誤差
    checkpoint = ModelCheckpoint(file1, monitor='val_loss', verbose=2,
save_best_only = True, mode='min')
    earlyStopping = EarlyStopping(monitor='val_loss', patience=50, verbose=2,
mode='auto')
    callbacks_list= [checkpoint,earlyStopping]
    gru_model = Build_Model(re_X, y)
    history = gru_model.fit(re_X, y, epochs=80, batch_size=35,
callbacks=callbacks_list, validation_split=0.1)
    # savemodel
    gru_model.save(file1)
    return history.history ['loss'], history.history ['val_loss']

def Predict(model, X_test):
    temp_x = np.concatenate((X_test.iloc[:,:].values, X_test.iloc[:,-4:].values),axis=1)
    temp_x = scaler.fit_transform(temp_x)
    y_hat = model.predict(temp_x[:,:12])
    # inverse
    all_test = np.concatenate((temp_x[:,:12], y_hat),axis=1)
    actual_test = scaler.inverse_transform(all_test)
    return y_hat, actual_test

def plot_loss(x, y):
    plt.figure(figsize=(12,6))
    plt.plot(x)
    plt.plot(y)
    plt.title('GRU Model loss')
    plt.ylabel('MSE')
    plt.xlabel('epoch')
    plt.legend(['Train_loss', 'Val_loss'], loc='upper right')

def Export_csv(data):
    cols = []
    for i in range(-11,5,1):
        if i <0 :
            cols.append(f't-{i*(-1)}')
        elif i==0:
            cols.append('t')
        else:
            cols.append(f't+{i}')

    df_fn = pd.DataFrame(data)
    df_fn.to_csv('Q3/predict2_answer.csv', index=None, header=cols)

# Get data
X_train, y_train, X_test = Get_data()

# data normalize
scaler, X_scaled, y_scaled = Normalize()

# train model
loss, val_loss = Train_Model(X_scaled, y_scaled)

# plot loss
plot_loss(loss, val_loss)

# load model
pre_model= load_model('Q3/predict2.h5')

# predict value
y_hat, actual = Predict(pre_model, X_test)
```

```
# export csv
Export_csv = Export_csv(actual)
```