代码说明：

Matlab ：

首选输入原始的信号，然后用emd对信号进行分解，得到不同周期的信号。以life\_satisfaction为准

life\_satisfaction = [

6.216210365 , 6.211949591 , 6.198868247 ,6.252538057 ,6.315385905...

6.256154899 , 6.187330907 ,6.220029451 , 6.271091513, 6.308437102...

6.343245858, 6.379117972, 6.379060427 ,6.488314988 ,6.475900071...

6.439487199, 6.428687935 ,6.358960553, 6.358133187 , 6.392539253...

6.370954396, 6.39373659 , 6.416875787 , 6.408049612 ,6.409385659...

6.422427454, 6.421092227, 6.433477968, 6.43654003, 6.373620445...

6.332920204, 6.337736004, 6.387097274, 6.392753319, 6.428181465...

6.397039853, 6.39135193 , 6.419184022, 6.460422477, 6.360431719...

6.333665896, 6.217446585, 6.304412691, 6.327005177, 6.336293833...

6.338430537];

imf=emd(life\_satisfaction); %进行emd信号分解

imf1 = imf(1,:);

imf2 = imf(2,:);

imf3 = imf(3,:);

imf4 = imf(4,:);

res = imf(5,:);

将分解的信号与额外获取的特征放在同一个excel表里面。 ( new\_life\_data.csv )

Python ：

用于lstm模型来进行时间序列预测，lstm模型的优点是能很好解决时间序列的长短依赖关系。

﻿#!/usr/bin/env python2

# -\*- coding: utf-8 -\*-

"""

Created on Fri Nov 27 22:19:23 2020

@author: ninofeng

"""

import numpy as np

from keras.layers.core import Dense, Activation, Dropout

from keras.layers import LSTM

from keras.models import Sequential, load\_model

from keras.callbacks import Callback

import keras.backend.tensorflow\_backend as KTF

import tensorflow as tf

import pandas as pd

import os

import keras.callbacks

import matplotlib.pyplot as plt

import os

os.environ["KMP\_DUPLICATE\_LIB\_OK"]="TRUE"

def NormalizeMult(data):

'''

归一化 适用于单维和多维

返回归一化后的数据和最大最小值

'''

normalize = np.arange(2\*data.shape[1],dtype='float64')

normalize = normalize.reshape(data.shape[1],2)

for i in range(0,data.shape[1]):

list = data[:,i]

listlow,listhigh = np.percentile(list, [0, 100])

normalize[i,0] = listlow

normalize[i,1] = listhigh

delta = listhigh - listlow

if delta != 0:

for j in range(0,data.shape[0]):

data[j,i] = (data[j,i] - listlow)/delta

return data, normalize

def create\_dataset(data,n\_predictions,n\_next):

'''

对数据进行处理

'''

dim = data.shape[1]

train\_X, train\_Y = [], []

for i in range(data.shape[0]-n\_predictions-n\_next-1):

a = data[i:(i+n\_predictions),:]

train\_X.append(a)

tempb = data[(i+n\_predictions):(i+n\_predictions+n\_next),:]

b = []

for j in range(len(tempb)):

for k in range(dim):

b.append(tempb[j,k])

train\_Y.append(b)

train\_X = np.array(train\_X,dtype='float64')

train\_Y = np.array(train\_Y,dtype='float64')

return train\_X, train\_Y

def trainModel(train\_X, train\_Y):

'''

trainX，trainY: 训练LSTM模型所需要的数据

'''

model = Sequential()

model.add(LSTM(

18,

input\_shape=(train\_X.shape[1], train\_X.shape[2]),

return\_sequences=True))

model.add(Dropout(0.3))

model.add(LSTM(

18,

return\_sequences=False))

model.add(Dropout(0.3))

model.add(Dense(

train\_Y.shape[1]))

model.add(Activation("relu"))

model.compile(loss='mse', optimizer='adam')

model.fit(train\_X, train\_Y, epochs=100, batch\_size=4, verbose=1)

return model

def reshape\_y\_hat(y\_hat,dim):

re\_y = []

i = 0

while i < len(y\_hat):

tmp = []

for j in range(dim):

tmp.append(y\_hat[i+j])

i = i + dim

re\_y.append(tmp)

re\_y = np.array(re\_y,dtype='float64')

return re\_y

#多维反归一化

def FNormalizeMult(data,normalize):

data = np.array(data,dtype='float64')

#列

for i in range(0,data.shape[1]):

listlow = normalize[i,0]

listhigh = normalize[i,1]

delta = listhigh - listlow

#行

if delta != 0:

for j in range(0,data.shape[0]):

data[j,i] = data[j,i]\*delta + listlow

return data

dataframe = pd.read\_csv('new\_life\_data.csv');

imf1 = dataframe["imf1"].tolist();

imf2 = dataframe["imf2"].tolist();

imf3 = dataframe["imf3"].tolist();

imf4 = dataframe["imf4"].tolist();

res = dataframe["res"].tolist();

holiday = dataframe["holiday"].tolist();

Unemployment\_rate = dataframe["Unemployment\_rate"].tolist()

Unemployment\_White = dataframe["Unemployment\_White"].tolist()

Unemployment\_Asian = dataframe["Unemployment\_Asian"].tolist()

Unemployment\_African = dataframe["Unemployment\_African"].tolist()

Consumer\_Price\_Index = dataframe["Consumer\_Price\_Index"].tolist()

COVID\_cases = dataframe["Covid\_cases"].tolist()

data = np.zeros(552)

data.dtype = 'float64'

data = data.reshape(46, 12);

data[:, 0] = np.array(imf1)

data[:, 1] = np.array(imf2)

data[:, 2] = np.array(imf3)

data[:, 3] = np.array(imf4)

data[:, 4] = np.array(holiday)

data[:, 5] = np.array(Unemployment\_rate)

data[:, 6] = np.array(Unemployment\_White)

data[:, 7] = np.array(Unemployment\_Asian)

data[:, 8] = np.array(Unemployment\_African)

data[:, 9] = np.array(Consumer\_Price\_Index)

data[:, 10] = np.array(COVID\_cases)

data[:, 11] = np.array(res)

print 'data shape: ', data.shape

#归一化的加入

data,normalize = NormalizeMult(data)

print data

train\_X,train\_Y = create\_dataset(data, 24, 12)

print train\_X.shape

print train\_Y.shape

model = trainModel(train\_X, train\_Y)

#print data[-12:, :].shape

test\_X = data[-24:, :]

test\_X = test\_X.reshape(1, test\_X.shape[0], test\_X.shape[1])

y\_hat = model.predict(test\_X)

#重组

y\_hat = y\_hat.reshape(y\_hat.shape[1])

y\_hat = reshape\_y\_hat(y\_hat, 12)

#反归一化

y\_hat = FNormalizeMult(y\_hat, normalize)

imf1\_pre = y\_hat[:,0];

imf2\_pre = y\_hat[:,1];

imf3\_pre = y\_hat[:,2];

imf4\_pre = y\_hat[:,3];

res\_pre = y\_hat[:, 11]

t = [i for i in range(1, 47)]

pre\_t = [i for i in range(47, 59)]

plt.plot(t , imf1, label="Train", color = 'r')

plt.plot(pre\_t , imf1\_pre, label="Test", color = 'b')

plt.show()

plt.plot(t , imf2, label="Train", color = 'r')

plt.plot(pre\_t , imf2\_pre, label="Test", color = 'b')

plt.show()

plt.plot(t , imf3, label="Train", color = 'r')

plt.plot(pre\_t , imf3\_pre, label="Test", color = 'b')

plt.show()

plt.plot(t , imf4, label="Train", color = 'r')

plt.plot(pre\_t , imf4\_pre, label="Test", color = 'b')

plt.show()

plt.plot(t , res, label="Train", color = 'r')

plt.plot(pre\_t , res\_pre, label="Test", color = 'b')

plt.show()

final\_result = []

for i in xrange(len(imf1\_pre)):

final\_result.append( imf1\_pre[i] + imf2\_pre[i] + imf3\_pre[i] + imf4\_pre[i] + res\_pre[i])

life\_satisfaction = [6.216210365 , 6.211949591 , 6.198868247 ,6.252538057 ,6.315385905,

6.256154899 , 6.187330907 ,6.220029451 , 6.271091513, 6.308437102,

6.343245858, 6.379117972, 6.379060427 ,6.488314988 ,6.475900071,

6.439487199, 6.428687935 ,6.358960553, 6.358133187 , 6.392539253,

6.370954396, 6.39373659 , 6.416875787 , 6.408049612 ,6.409385659,

6.422427454, 6.421092227, 6.433477968, 6.43654003, 6.373620445,

6.332920204, 6.337736004, 6.387097274, 6.392753319, 6.428181465,

6.397039853, 6.39135193 , 6.419184022, 6.460422477, 6.360431719,

6.333665896, 6.217446585, 6.304412691, 6.327005177, 6.336293833,

6.338430537]

plt.plot(train\_x , life\_satisfaction, label="Train", color = 'r')

plt.plot(test\_x , final\_result, label="Test", color = 'b')

plt.show()