**Math6373 Homework 3 -- An application of MLP AutoEncoders**

Team Member: Mei Ni; Hsien-Hao Hsu; Yingyu Huang

Contribution:

Mei Ni ([plumnini.xj@gmail.com](mailto:plumnini.xj@gmail.com)): Working on the programing of part 1 and 2

Hsien-Hao Hsu ([sam10030179@gmail.com](mailto:sam10030179@gmail.com)): Working on the report designing and wording

Yingyu Huang ([yingyu010365@gmail.com](mailto:yingyu010365@gmail.com)): Working on the programing of part 3

**1. data information**

**1.1 data source**

The close price of 50 selected stocks from 01/02/2014 to 12/31/2017. Data source is Yahoo Finance. Totally 1007 non-NA cases are used. The price of Amazon is the target stock that is needed to predict by itself and the other 49 stocks.

**1.2 data normalization**

* Compute the moving mean of each stock with the formula below.

Sj(t) is the close price of stock j at time t, avSj(t) = [Sj(1) + ... + Sj(t) ] /t

* Then get a new data set with 1007 cases, each case has 50 features.

**1.3 training set and test set**

* Reshape the first five rows of the normalized data into a vector with dimension as (1, 250)
* Repeat this procedure for all the cases in data set with the stride equal to 1
* Then get 1002 vectors. Combine these vectors into a matrix X with the dimension as (1002, 250)
* Choose the close price of AMAZON as the response vector Y values from 01/08/2014 to 12/31/2017, the dimension is (1002, 1)
* Combine X and Y as DATA
* Select 900 rows of DATA randomly as the training set, the other 102 rows as test set and validation set, and form the Xtrain, Xtest, Ytrain and Ytest for the autoencoder fitting

**1.4 The goal of this work**

* Develop an autoencoder based on MLP to decrease the dimension of the input data and at the same time retain at least 90% of the variance of it.
* Develop an automatic prediction based on MLP which can predict the close price of AMAZON by the dimension-reduced data.

**2. AutoEncoder by MLP**

**2.1 the size of hidden layer**

By PCA, the smallest r<=250, which makes the ratio of explanation variance of the first r principal components to be 90%, is 6. So, the initial size of hidden layer is 6.

**2.2 create the architecture of MLP**

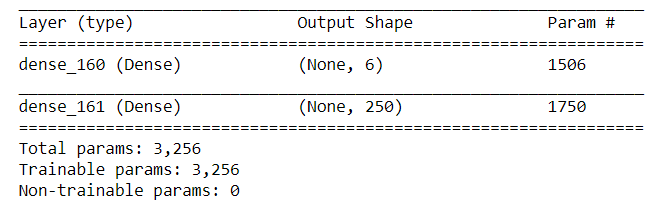
Layer 1: input layer, 250 neurons

Layer 2: hidden layer, 6 neurons, response function is RELU

Layer 3: output layer, 250 neurons, response function is RELU

Number of unknown parameters is 3256

Ratio = 900\*250/3256 ≈ 69



**2.3 the option of MLP for AutoEncoder**

* **the learning algorithm**
  + The learning algorithm is Gradient Descent
  + Initial learning rate is 0.05, decay rate is =1e-7
* **the loss functions**
  + Mean square error
* **the initialization of W**
  + initial weight is random value of 'glorot\_uniform'
  + initial bias is 10 for layer 2 and 5 for layer 3
* **Batch learning**
  + Batch size = 16
  + Epoch = 2000

**2.4 Best model and performance of MLP**

**2.4.1 Model parameters tuning**

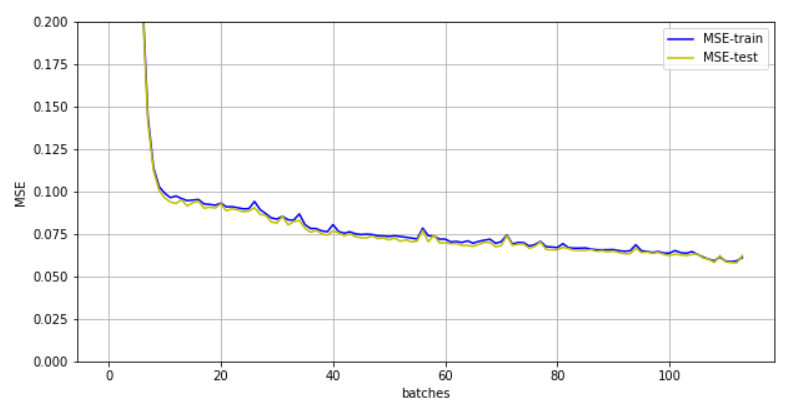
* Tuning learning rate in [0.1, 0.05, 0.01] 🡪 0.05 is the best
* Tuning batch size in [16, 32, 64] 🡪 16 is the best
* Tuning h in [5, 6, 7, 8] 🡪 6 is the best

**2.4.2 Early stopping technique**

Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. So, we use the early stopping method in Keras to stop the learning when the chosen performance measure stops improving. In our case, the monitor is the mean square error of the validation data. When the MSE of validation data is monitored to stop decreasing in 200 epochs, the early stop will be triggered and learning stops, and at the same time save the best model.

**2.4.3 Plot MSE(AutoTrain) and MSE(AutoTest) versus the number of batches**

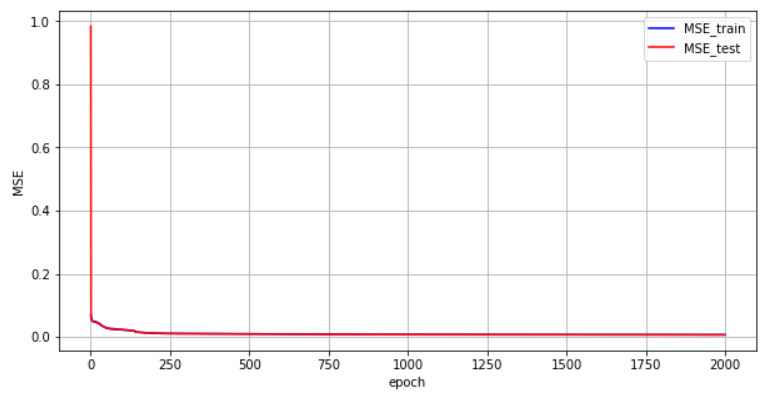
* The performance of best model is **MSE-train = 0.0036, MSE-test = 0.0038**
* Plot the MSE-train and MSE-test on the first 2 epoch. The MSE of training set and test set keep decreasing. They drop down to a very low level at the first ten batches, then decrease slowly. And the two curves are close to each other.



(To see the detail of the curve, cut off the first 7 batches.)

**2.4.4** **Plot MSE(AutoTrain) and MSE(AutoTest) versus epochs**

* The two curves close to each other, so, the blue line is covered by the red line.
* It means the model is robust.



**2.5 extract the state of hidden layer on training set and test set**

Extract the state of hidden layer on training set and test set to Ztrain and Ztest by the best model, which is the input data for the prediction MLP below. The shape of Ztrain is 900 \* 6, of Ztest is 102 \* 6.

**3 Predict the close price of AMAZON by MLP**

**3.1 the size of hidden layer**

Let the size of hidden layer be H. Then the ratio is 900\*1/(6 \* H + H \*1 + H +1) and should be larger than 1. Solving this unequal equation, we can get that H should be less than 112. So, the initial H of MLP is set as 112.

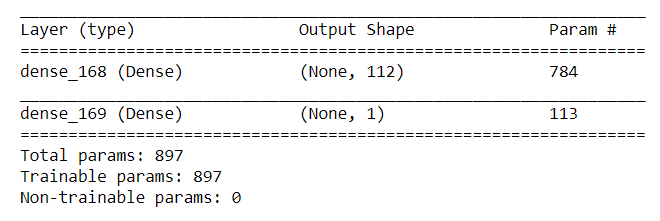
**3.2 create the architecture of MLP**

Layer 1: input layer, 6 neurons

Layer 2: hidden layer, 112 neurons, response function is RELU

Layer 3: output layer, 1 neuron, response function is RELU

Number of unknown parameters is 897



**3.3 the option of MLP**

* **the learning algorithm**
  + The learning algorithm is Gradient Descent
  + Initial learning rate is 0.00001, decay rate is =1e-7
* **the loss functions**
  + Mean square error
* **the initialization of W**
  + initial weight is random value of 'glorot\_uniform'
  + initial bias is 5 for layer 2 and 40 for layer 3
* **Batch learning**
  + Batch size = 16
  + Epoch = 10000

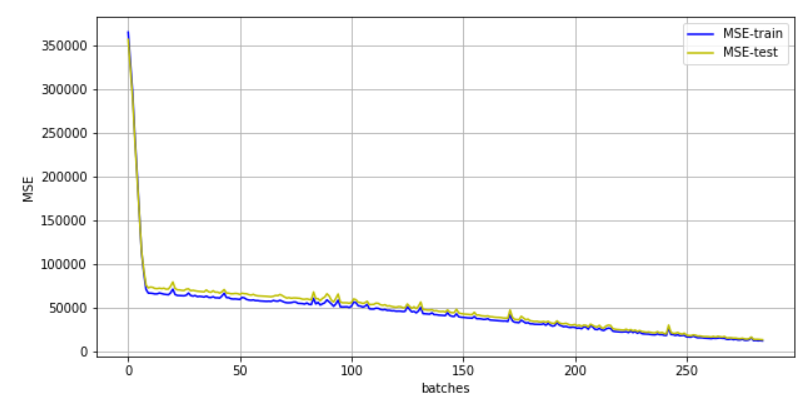
**3.4 Best model and performance of MLP**

**3.4.1 model tuning**

* Tuning learning rate in [0.0001, 0.00005, 0.00001,0.000001] 🡪 0.00001 is the best
* Tuning batch size in [16, 32, 64] 🡪 16 is the best
* Tuning h in [20, 40, 60, 80, 100, 112] 🡪 60 is the best
* Ratio = 900\*1 / 481 ≈ 1.87

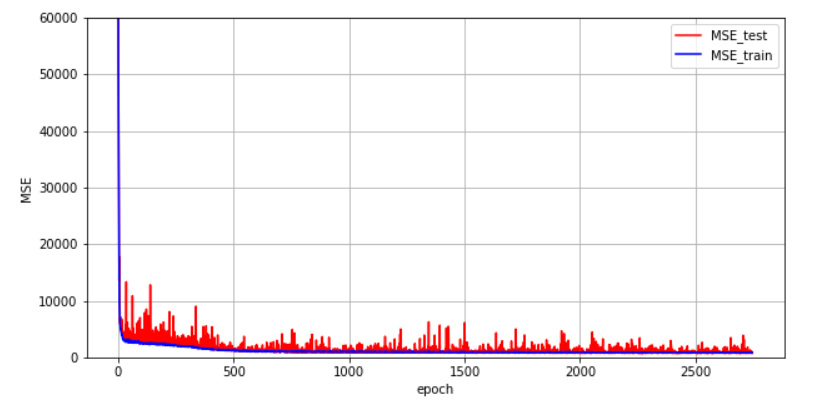
**3.4.2 Plot MSE(Htrain) and MSE(Htest) versus the number of batches**

* The required number of epochs is 10000 but stopped at 2747.
* The performance of best model is: **MSE-train = 770, MSE-test = 669**
* Plot the MSE-train and MSE-test on the batches of the first 5 epochs.



**3.4.3 Plot MSE(Htrain) and MSE(Htest) versus the number of epochs**

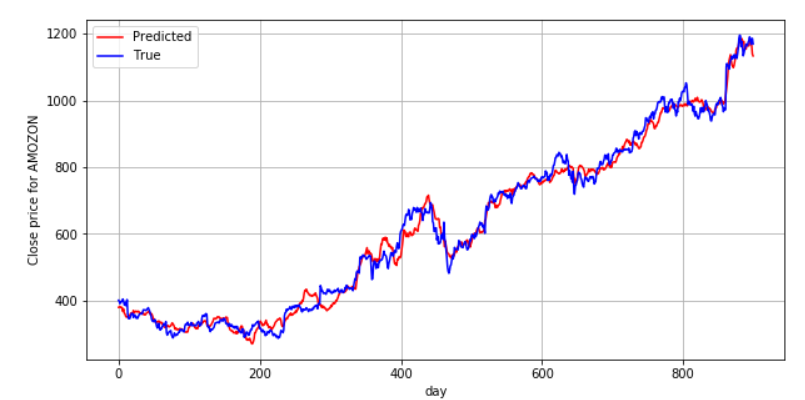
* The mean square error of training set and test set has a sharp dropping down at the first 10 epochs, then keep decreasing slowly.
* The mean square error of test set oscillates seriously at the beginning, then the amplitude shrinks along with the training procedure. The reason may be that the batch size is small and some of the fitting on batches have large bias.
* The unstableness of the performance of test set shows that this model may has poor robust and the accuracy of the prediction may have a large variability.



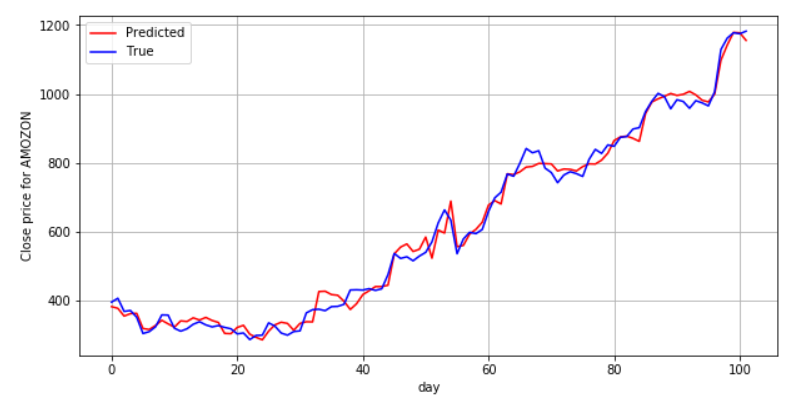
**3.4.4 plot the true price and the predicted price of AMAZON vs the date**

* Predict the close price of AMAZON on training set and test set by the best MLP model
* **The Mean Relative Errors of training set is 0.0407, of test set is 0.0411**
* From the plot curves below, the predicted close price is close to the true one, but not good enough. Maybe the prediction MLP need to be tuned farther.

True price of AMAZON and the predicted price vs the date on training set



True price of AMAZON and the predicted price vs the date on test set



Code:

**# part 1: data preparation**

import pandas as pd

import numpy as np

dataY = pd.DataFrame(pd.read\_csv('AMZN.csv')[['Date','Close']])

dataY.columns = ['Date', 'AMZN']

data1 = pd.DataFrame(pd.read\_csv('AAL.csv')[['Date','Close']])

data1.columns = ['Date', 'X1']

data2 = pd.DataFrame(pd.read\_csv('AAPL.csv')[['Date','Close']])

data2.columns = ['Date', 'X2']

data3 = pd.DataFrame(pd.read\_csv('AMD.csv')[['Date','Close']])

data3.columns = ['Date', 'X3']

data4 = pd.DataFrame(pd.read\_csv('AMRN.csv')[['Date','Close']])

data4.columns = ['Date', 'X4']

data5 = pd.DataFrame(pd.read\_csv('AMZN.csv')[['Date','Close']])

data5.columns = ['Date', 'X5']

data6 = pd.DataFrame(pd.read\_csv('APA.csv')[['Date','Close']])

data6.columns = ['Date', 'X6']

data7 = pd.DataFrame(pd.read\_csv('AUY.csv')[['Date','Close']])

data7.columns = ['Date', 'X7']

data8 = pd.DataFrame(pd.read\_csv('BA.csv')[['Date','Close']])

data8.columns = ['Date', 'X8']

data9 = pd.DataFrame(pd.read\_csv('BAC.csv')[['Date','Close']])

data9.columns = ['Date', 'X9']

data10 = pd.DataFrame(pd.read\_csv('BBD.csv')[['Date','Close']])

data10.columns = ['Date', 'X10']

data11 = pd.DataFrame(pd.read\_csv('C.csv')[['Date','Close']])

data11.columns = ['Date', 'X11']

data12 = pd.DataFrame(pd.read\_csv('CCL.csv')[['Date','Close']])

data12.columns = ['Date', 'X12']

data13 = pd.DataFrame(pd.read\_csv('CMCSA.csv')[['Date','Close']])

data13.columns = ['Date', 'X13']

data14 = pd.DataFrame(pd.read\_csv('CZR.csv')[['Date','Close']])

data14.columns = ['Date', 'X14']

data15 = pd.DataFrame(pd.read\_csv('DAL.csv')[['Date','Close']])

data15.columns = ['Date', 'X15']

data16 = pd.DataFrame(pd.read\_csv('DVN.csv')[['Date','Close']])

data16.columns = ['Date', 'X16']

data17 = pd.DataFrame(pd.read\_csv('EQT.csv')[['Date','Close']])

data17.columns = ['Date', 'X17']

data18 = pd.DataFrame(pd.read\_csv('ET.csv')[['Date','Close']])

data18.columns = ['Date', 'X18']

data19 = pd.DataFrame(pd.read\_csv('F.csv')[['Date','Close']])

data19.columns = ['Date', 'X19']

data20 = pd.DataFrame(pd.read\_csv('FB.csv')[['Date','Close']])

data20.columns = ['Date', 'X20']

data21 = pd.DataFrame(pd.read\_csv('FCX.csv')[['Date','Close']])

data21.columns = ['Date', 'X21']

data22 = pd.DataFrame(pd.read\_csv('GE.csv')[['Date','Close']])

data22.columns = ['Date', 'X22']

data23 = pd.DataFrame(pd.read\_csv('GILD.csv')[['Date','Close']])

data23.columns = ['Date', 'X23']

data24 = pd.DataFrame(pd.read\_csv('GOLD.csv')[['Date','Close']])

data24.columns = ['Date', 'X24']

data25 = pd.DataFrame(pd.read\_csv('HAL.csv')[['Date','Close']])

data25.columns = ['Date', 'X25']

data26 = pd.DataFrame(pd.read\_csv('INO.csv')[['Date','Close']])

data26.columns = ['Date', 'X26']

data27 = pd.DataFrame(pd.read\_csv('ITUB.csv')[['Date','Close']])

data27.columns = ['Date', 'X27']

data28 = pd.DataFrame(pd.read\_csv('IVZ.csv')[['Date','Close']])

data28.columns = ['Date', 'X28']

data29 = pd.DataFrame(pd.read\_csv('JPM.csv')[['Date','Close']])

data29.columns = ['Date', 'X29']

data30 = pd.DataFrame(pd.read\_csv('KGC.csv')[['Date','Close']])

data30.columns = ['Date', 'X30']

data31 = pd.DataFrame(pd.read\_csv('KMI.csv')[['Date','Close']])

data31.columns = ['Date', 'X31']

data32 = pd.DataFrame(pd.read\_csv('MGM.csv')[['Date','Close']])

data32.columns = ['Date', 'X32']

data33 = pd.DataFrame(pd.read\_csv('MRO.csv')[['Date','Close']])

data33.columns = ['Date', 'X33']

data34 = pd.DataFrame(pd.read\_csv('MSFT.csv')[['Date','Close']])

data34.columns = ['Date', 'X34']

data35 = pd.DataFrame(pd.read\_csv('MU.csv')[['Date','Close']])

data35.columns = ['Date', 'X35']

data36 = pd.DataFrame(pd.read\_csv('NBL.csv')[['Date','Close']])

data36.columns = ['Date', 'X36']

data37 = pd.DataFrame(pd.read\_csv('NCLH.csv')[['Date','Close']])

data37.columns = ['Date', 'X37']

data38 = pd.DataFrame(pd.read\_csv('OXY.csv')[['Date','Close']])

data38.columns = ['Date', 'X38']

data39 = pd.DataFrame(pd.read\_csv('BRK-B.csv')[['Date','Close']])

data39.columns = ['Date', 'X39']

data40 = pd.DataFrame(pd.read\_csv('PFE.csv')[['Date','Close']])

data40.columns = ['Date', 'X40']

data41 = pd.DataFrame(pd.read\_csv('RF.csv')[['Date','Close']])

data41.columns = ['Date', 'X41']

data42 = pd.DataFrame(pd.read\_csv('SIRI.csv')[['Date','Close']])

data42.columns = ['Date', 'X42']

data43 = pd.DataFrame(pd.read\_csv('SLB.csv')[['Date','Close']])

data43.columns = ['Date', 'X43']

data44 = pd.DataFrame(pd.read\_csv('T.csv')[['Date','Close']])

data44.columns = ['Date', 'X44']

data45 = pd.DataFrame(pd.read\_csv('TWTR.csv')[['Date','Close']])

data45.columns = ['Date', 'X45']

data46 = pd.DataFrame(pd.read\_csv('UAL.csv')[['Date','Close']])

data46.columns = ['Date', 'X46']

data47 = pd.DataFrame(pd.read\_csv('VALE.csv')[['Date','Close']])

data47.columns = ['Date', 'X47']

data48 = pd.DataFrame(pd.read\_csv('VER.csv')[['Date','Close']])

data48.columns = ['Date', 'X48']

data49 = pd.DataFrame(pd.read\_csv('WFC.csv')[['Date','Close']])

data49.columns = ['Date', 'X49']

data50 = pd.DataFrame(pd.read\_csv('XOM.csv')[['Date','Close']])

data50.columns = ['Date', 'X50']

data = pd.merge(data1, data2, how='outer', on='Date')

data = pd.merge(data, data3, how='outer', on='Date')

data = pd.merge(data, data4, how='outer', on='Date')

data = pd.merge(data, data5, how='outer', on='Date')

data = pd.merge(data, data6, how='outer', on='Date')

data = pd.merge(data, data7, how='outer', on='Date')

data = pd.merge(data, data8, how='outer', on='Date')

data = pd.merge(data, data9, how='outer', on='Date')

data = pd.merge(data, data10, how='outer', on='Date')

data = pd.merge(data, data11, how='outer', on='Date')

data = pd.merge(data, data12, how='outer', on='Date')

data = pd.merge(data, data13, how='outer', on='Date')

data = pd.merge(data, data14, how='outer', on='Date')

data = pd.merge(data, data15, how='outer', on='Date')

data = pd.merge(data, data16, how='outer', on='Date')

data = pd.merge(data, data17, how='outer', on='Date')

data = pd.merge(data, data18, how='outer', on='Date')

data = pd.merge(data, data19, how='outer', on='Date')

data = pd.merge(data, data20, how='outer', on='Date')

data = pd.merge(data, data21, how='outer', on='Date')

data = pd.merge(data, data22, how='outer', on='Date')

data = pd.merge(data, data23, how='outer', on='Date')

data = pd.merge(data, data24, how='outer', on='Date')

data = pd.merge(data, data25, how='outer', on='Date')

data = pd.merge(data, data26, how='outer', on='Date')

data = pd.merge(data, data27, how='outer', on='Date')

data = pd.merge(data, data28, how='outer', on='Date')

data = pd.merge(data, data29, how='outer', on='Date')

data = pd.merge(data, data30, how='outer', on='Date')

data = pd.merge(data, data31, how='outer', on='Date')

data = pd.merge(data, data32, how='outer', on='Date')

data = pd.merge(data, data33, how='outer', on='Date')

data = pd.merge(data, data34, how='outer', on='Date')

data = pd.merge(data, data35, how='outer', on='Date')

data = pd.merge(data, data36, how='outer', on='Date')

data = pd.merge(data, data37, how='outer', on='Date')

data = pd.merge(data, data38, how='outer', on='Date')

data = pd.merge(data, data39, how='outer', on='Date')

data = pd.merge(data, data40, how='outer', on='Date')

data = pd.merge(data, data41, how='outer', on='Date')

data = pd.merge(data, data42, how='outer', on='Date')

data = pd.merge(data, data43, how='outer', on='Date')

data = pd.merge(data, data44, how='outer', on='Date')

data = pd.merge(data, data45, how='outer', on='Date')

data = pd.merge(data, data46, how='outer', on='Date')

data = pd.merge(data, data47, how='outer', on='Date')

data = pd.merge(data, data48, how='outer', on='Date')

data = pd.merge(data, data49, how='outer', on='Date')

data = pd.merge(data, data50, how='outer', on='Date')

data.isna().sum()

data.shape

**# calculate the moving ave of data**

data\_ave = np.empty(shape=[1007,50])

for stock in range(1,51) :

for day in range(1007):

data\_ave[day,stock-1]= float(data.iloc[day,stock])/float(data.iloc[0:(day+1),stock].sum())\*(day+1)

**# extract the input data by 50\*5 window**

data\_input = np.empty(shape=[1002,250])

for day in range(1002):

data\_input[day,]=data\_ave[day:day+5,:].reshape((1,250))

data\_input\_y = np.matrix(dataY.iloc[5:1007,1]).T

data\_input\_y

date\_input = np.matrix(dataY.iloc[5:1007,0]).T

date\_input.shape

data\_input = np.concatenate((data\_input, data\_input\_y,date\_input),axis=1)

**# split training set and test set**

from sklearn.model\_selection import train\_test\_split

second\_size = 102/1002

train, test = train\_test\_split(data\_input, test\_size=second\_size,random\_state=123)

train=pd.DataFrame(train)

test=pd.DataFrame(test)

train=train.sort\_values(by=[251])

test=test.sort\_values(by=[251])

train.iloc[:,0:251].to\_csv('train.csv')

test.iloc[:,0:251].to\_csv('test.csv')

Xtrain = np.matrix(train.iloc[:,0:250])

**# calculate the h by PCA**

G = np.corrcoef(Xtrain.T)

# pca to determine h

from numpy import linalg as la

eigenv=la.eig(G)

eigenValues=eigenv[0]

eigenValues=np.sort(eigenValues)[::-1]

RAT=[]

eigen\_sum=0

for i in range(250):

eigen\_sum=eigen\_sum+eigenValues[i]

ratio=eigen\_sum/sum(eigenValues)

RAT.append(ratio)

# Identify the smallest j such that RATj ≥ 90%

j=0

for i in range(250):

if RAT[i]>=0.90:

j=i

break

j

**# Part 2: auto encoder MLP**

import tensorflow as tf

import numpy as np

import pandas as pd

# load data

train = pd.read\_csv("train.csv", dtype='float32')

test = pd.read\_csv("test.csv", dtype='float32')

train.shape

Xtrain = np.array(train.iloc[:,1:251])

Ytrain = np.array(train.iloc[:,251])

Xtest = np.array(test.iloc[:,1:251])

Ytest = np.array(test.iloc[:,251])

Xtest.shape, Ytest.shape

Xtrain.shape, Ytrain.shape

# set the MLP architecture

import tensorflow.keras as keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.initializers import Constant

h = 6

model = Sequential()

model.add(Dense(h, activation='relu', input\_dim=250, bias\_initializer=Constant(value=10)))

model.add(Dense(250, activation='relu', bias\_initializer=Constant(value=5)))

model.summary()

# compile the model

from tensorflow.keras import optimizers, losses

model.compile(optimizer=optimizers.SGD(learning\_rate=0.05, decay=1e-7), loss='mean\_squared\_error')

# callback for evaluation results by batch and early stopping monitor

from tensorflow.keras import callbacks

class MyHistory(callbacks.Callback):

def on\_train\_begin(self, logs={}):

self.MSEtrain = []

self.MSEtest = []

def on\_batch\_end(self, batch, logs={}):

self.MSEtrain.append(self.model.evaluate(Xtrain,Xtrain,verbose = 0))

self.MSEtest.append(self.model.evaluate(Xtest,Xtest,verbose = 0))

MyMonitor = MyHistory()

es = callbacks.EarlyStopping(monitor='val\_loss', mode='min', verbose=1, patience=1000, restore\_best\_weights=True)

# save the best model

filepath = 'BestModel.h5'

checkpointer = callbacks.ModelCheckpoint(filepath=filepath, monitor='val\_loss', save\_best\_only=True, mode='auto', save\_freq='epoch')

# learning on training set and validate on test set

Monitor = model.fit(Xtrain, Xtrain, epochs=2000, batch\_size=16, callbacks = [MyMonitor, es,checkpointer], validation\_data = (Xtest, Xtest), verbose = 2)

# MSE of training set and test set of the best model

from tensorflow.keras.models import load\_model

bestmodel = load\_model('BestModel.h5')

Xtrani\_MSE = bestmodel.evaluate(Xtrain,Xtrain)

Xtest\_MSE = bestmodel.evaluate(Xtest,Xtest)

print(Xtrani\_MSE, Xtest\_MSE)

# Plot MSE(AutoTrain) and MSE(AutoTest) versus the number of batches

MyMonitor.MSEtrain

MSE\_train = np.array(MyMonitor.MSEtrain)

MSE\_test = np.array(MyMonitor.MSEtest)

import matplotlib.pyplot as plt

fig=plt.figure(figsize=(10,5))

axb=fig.add\_subplot(1,1,1)

axb.grid()

axb.plot(range(len(MSE\_train)),MSE\_train,c='b')

axb.plot(range(len(MSE\_train)),MSE\_test,c='y')

axb.set\_xlabel('batches')

axb.set\_ylabel('MSE')

axb.set\_ylim(0,0.2)

plt.legend(['MSE-train','MSE-test'],loc='best')

plt.show

# plot loss and val\_loss of all epochs

loss\_tr = Monitor.history['loss']

loss\_te = Monitor.history['val\_loss']

fig=plt.figure(figsize=(10,5))

ax=fig.add\_subplot(1,1,1)

ax.grid()

ax.plot(range(len(loss\_tr)),loss\_te,c='b')

ax.plot(range(len(loss\_tr)),loss\_tr,c='r')

ax.set\_xlabel('epoch')

ax.set\_ylabel('MSE')

#ax.set\_ylim(0,6000)

plt.legend(['MSE\_train','MSE\_test'],loc='best')

plt.show

# Compute Compressed Inputs

Htrain = model.layers[0](Xtrain).numpy()

Htest = model.layers[0](Xtest).numpy()

#np.savetxt('Htrain.csv',Htrain)

#np.savetxt('Htest.csv',Htest)

Htrain

Ytrain = np.reshape(Ytrain,(900,1))

Ytest = np.reshape(Ytest,(102,1))

#np.savetxt('Ytrain.csv',Ytrain)

#np.savetxt('Ytest.csv',Ytest)

**# part 3: prediction MLP**

# set the MLP architecture

k = 60 # ration = 900\*1 / number of weight >=1 --> k < = 112

mlp = Sequential()

mlp.add(Dense(k, activation='relu', input\_dim=h, bias\_initializer=Constant(value=5)))

mlp.add(Dense(1, activation='relu', bias\_initializer=Constant(value=40)))

mlp.summary()

# compile model

mlp.compile(optimizer=optimizers.SGD(learning\_rate=0.00001, decay=1e-7), loss='mean\_squared\_error')

# callback for evaluation results by batch and early stopping monitor

class mlpMyHistory(callbacks.Callback):

def on\_train\_begin(self, logs={}):

self.MSEtrain = []

self.MSEtest = []

def on\_batch\_end(self, batch, logs={}):

self.MSEtrain.append(self.model.evaluate(Htrain,Ytrain,verbose = 0))

self.MSEtest.append(self.model.evaluate(Htest,Ytest,verbose = 0))

mlpMyMonitor1 = mlpMyHistory()

es1 = callbacks.EarlyStopping(monitor='val\_loss', mode='min', verbose=1, patience=200, restore\_best\_weights=True)

# save the best model

filepath = 'BestModel1.h5'

checkpointer = callbacks.ModelCheckpoint(filepath=filepath, monitor='val\_loss', save\_best\_only=True, mode='auto', save\_freq='epoch')

# learning

mlpMonitor = mlp.fit(Htrain, Ytrain, epochs=2000, batch\_size=16, callbacks = [mlpMyMonitor1, es1,checkpointer], validation\_data = (Htest, Ytest), verbose = 1)

# the MSE of training set and test set of the best model

bestmodel2 = load\_model('BestModel1.h5')

Htrani\_MSE = bestmodel2.evaluate(Htrain,Ytrain)

Htest\_MSE = bestmodel2.evaluate(Htest,Ytest)

print(Htrani\_MSE, Htest\_MSE)

# plot loss and val\_loss of all epochs

loss\_tr = mlpMonitor.history['loss']

loss\_te = mlpMonitor.history['val\_loss']

fig=plt.figure(figsize=(10,5))

ax=fig.add\_subplot(1,1,1)

ax.grid()

ax.plot(range(len(loss\_tr)),loss\_te,c='b')

ax.plot(range(len(loss\_tr)),loss\_tr,c='r')

ax.set\_xlabel('epoch')

ax.set\_ylabel('MSE')

ax.set\_ylim(0,6000)

plt.legend(['MSE\_train','MSE\_test'],loc='best')

plt.show

# plot the predicted Y and true Y on training set

Ypred = mlp.predict(Htrain)

import matplotlib.pyplot as plt

fig=plt.figure(figsize=(10,5))

ax=fig.add\_subplot(1,1,1)

ax.grid()

ax.plot(range(len(Ypred)),Ypred,c='r')

ax.plot(range(len(Ypred)),Ytrain,c='b')

ax.set\_xlabel('day')

ax.set\_ylabel('Close price for AMOZONE')

#ax.set\_yticks([0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1])

plt.legend(['Predicted','True'],loc='best')

plt.show

# plot the predicted Y and true Y on training set

Ypredt = mlp.predict(Htest)

import matplotlib.pyplot as plt

fig=plt.figure(figsize=(10,5))

ax=fig.add\_subplot(1,1,1)

ax.grid()

ax.plot(range(len(Ypredt)),Ypredt,c='r')

ax.plot(range(len(Ypredt)),Ytest,c='b')

ax.set\_xlabel('day')

ax.set\_ylabel('Close price for AMOZONE')

#ax.set\_yticks([0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1])

plt.legend(['Predicted','True'],loc='best')

plt.show

# calculate the relative error of training set and test set

np.average(np.absolute((Ypred-Ytrain))/Ypred)

np.average(np.absolute((Ypredt-Ytest))/Ypredt)

# plot the MSE by batches

MSE\_train = np.array(mlpMyMonitor1.MSEtrain)

MSE\_test = np.array(mlpMyMonitor1.MSEtest)

fig=plt.figure(figsize=(10,5))

axb=fig.add\_subplot(1,1,1)

axb.grid()

axb.plot(range(len(MSE\_train)),MSE\_train,c='b')

axb.plot(range(len(MSE\_train)),MSE\_test,c='y')

axb.set\_xlabel('batches')

axb.set\_ylabel('MSE')

plt.legend(['MSE-train','MSE-test'],loc='best')

plt.show