期末報告 7108029229 徐宛萱

**研究背景**

隨著觀測儀器以及氣象科學模型的進步，許多數值得以被詳細記錄。近年氣候漸趨極端，在暖化及其他環境破壞加劇的情況下，可以將人工智慧應用於預測天氣變化，鑒於其發展逐漸應用在許多領域，在氣象科學方面，如何將這些數據被更精準的處理並應用，來使得災難或是多變的氣候更加可以被掌握，減少對人類或是資產的損害是值得研究的。

此次作業是有關空汙預測，空氣中的懸浮微粒不只會危害呼吸道，也可能造成癌症、心血管疾病。這些懸浮微粒會經由鼻、咽及喉進入人體，不同粒徑大小的懸浮微粒，可能會導致人體器官不同的危害，其中最難預防的即為PM2.5，世界衛生組織（WHO）將空氣污染列為主要環境致癌物，引發肺癌風險更勝二手菸。最危險的，是可堆積在人肺深處的細懸浮微粒PＭ2.5。PM2.5 對健康的影響極大，因為可以穿透一般口罩，提高例如呼吸道疾病、癌症、新生兒低體重、心血管疾病等等的風險。

**資料集背景**

綠色和平組織和北京大學發布的《危險的呼吸——ＰＭ2.5的健康危害和經濟損失評估研究》指出，2012年在北京、上海和廣州、西安四大城市，因ＰＭ2.5污染而早死者，超過八千五百人。哈爾濱ＰＭ2.5數值每立方公尺突破一千微克，中小學因此停課兩天，被喻為「空氣末日」。中國環保部也統計，大陸七十四個城市，2013年十月，平均每兩天就有一天空污超標。資料集來源為北京的美國大使館2010一月一日至2014十二月三十一，五年間所檢測的天氣以及空氣汙染指數。

**資料集描述**

<https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data>

涵蓋以下欄位:

行數(No)、年(year)、月(month)、日(day)、小時(hour)、PM2.5濃度(pm2.5)、露點(DEWP)、溫度(TEMP)、大氣壓(PRES)、風向(cbwd)、風速(lws)、累積雪量(ls)、累積雨量(lr)

**資料處理步驟**

1. 檢查資料集是否有空值，並對其進行處理

在最一開始的24小時PM2.5的值皆為NA，刪除此部分資料並對其它時刻少量的預設值用Pandas中的fillna進行填充，另外整合日期資料，使其作為Pandas索引(index)，並將原始資料中的”No”列重新命名。

**程式碼:**

from pandas import read\_csv

from datetime import datetime

# load data

def parse(x):

return datetime.strptime(x, '%Y %m %d %H')

dataset = read\_csv('D:Downloads/PRSA\_data\_2010.1.1-2014.12.31.csv', parse\_dates = [['year', 'month', 'day', 'hour']], index\_col=0, date\_parser=parse)

dataset.drop('No', axis=1, inplace=True)

# manually specify column names

dataset.columns = ['pollution', 'dew', 'temp', 'press', 'wnd\_dir', 'wnd\_spd', 'snow', 'rain']

dataset.index.name = 'date'

# mark all NA values with 0

dataset['pollution'].fillna(0, inplace=True)

# drop the first 24 hours

dataset = dataset[24:]



處理後的資料儲存在”pollution.csv”

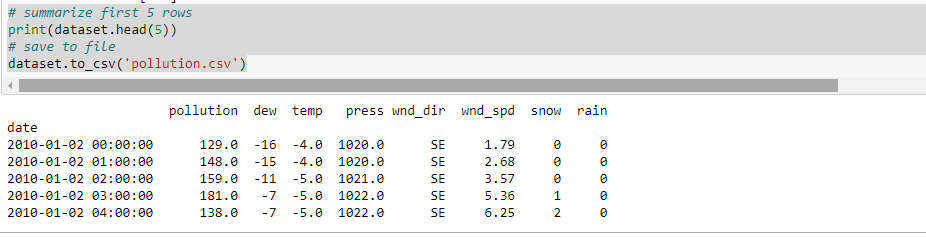
**程式碼:**

# summarize first 5 rows

print(dataset.head(5))

# save to file

dataset.to\_csv('pollution.csv')



1. 對每列資料進行繪圖

整理後的資料格式更加適合處理，載入”pollution.csv”對除了風速以外的七個變數類別繪製五年變化圖。

**程式碼:**

from pandas import read\_csv

from matplotlib import pyplot

# load dataset

dataset = read\_csv('pollution.csv', header=0, index\_col=0)

values = dataset.values

# specify columns to plot

groups = [0, 1, 2, 3, 5, 6, 7]

i = 1

# plot each column

pyplot.figure()

for group in groups:

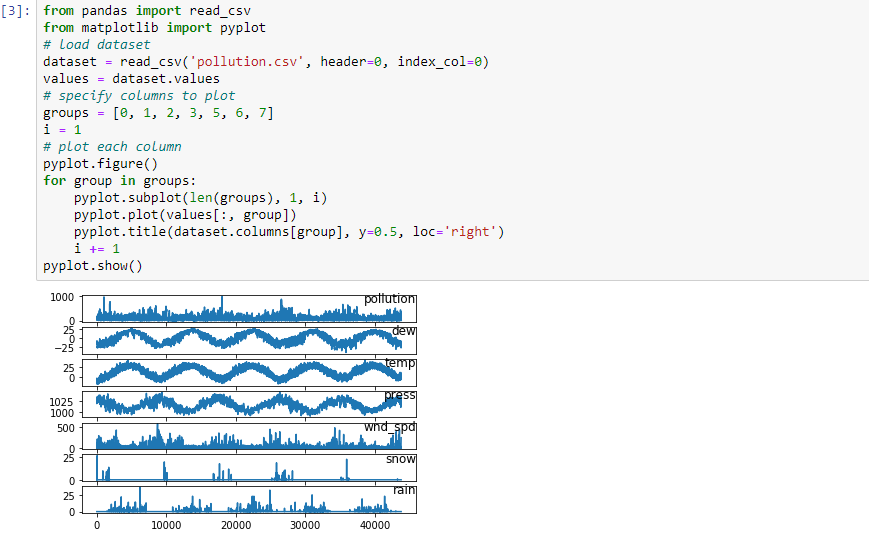
pyplot.subplot(len(groups), 1, i)

pyplot.plot(values[:, group])

pyplot.title(dataset.columns[group], y=0.5, loc='right')

i += 1

pyplot.show()



1. 採用LSTM模型

選擇LSTM對資料進行預測，其是一種遞歸神經網路(RNN)，相較RNN其解決了梯度消失的問題，原因在於多了gate，針對模型所獲得的訊息流做過濾。在LSTM當中總共構建三個gate來控制訊息流，分別為輸入門、遺忘門以及輸出門。

* 輸入門i(t) : 決定當前有多少訊息可以放入memory cell c(t)。
* 遺忘門f(t) : 決定上一層的memory cell c(t) 中的訊息可以累積多少到當前的memory cell c(t)。
* 輸出門 o(t) : 當前時刻的memory cell c(t)有多少可以流入當年隱藏狀態h(t)中。

先對資料進行處理，包括對資料集轉換為有監督學習問題和歸一化變數(涵輸入輸出)，來實現拖過前一個時刻(t-1)的污染資料和天氣條件是測當前時刻(t)的汙染，另外亦可利用過去24小時的汙染資料和天氣條件預測當前時刻的汙染或是預測下一個時刻（t+1）可能的天氣條件。首先載入“pollution.csv”檔案，利用sklearn的預處理模組對類別特徵“風向”進行編碼，也可以對該特徵進行one-hot編碼。再將所有的特徵進行歸一化處理，然後將資料集轉化為有監督學習問題，同時將需要預測的當前時刻（t）的天氣條件特徵移除。

**程式碼:**

from math import sqrt

from numpy import concatenate

from matplotlib import pyplot

from pandas import read\_csv

from pandas import DataFrame

from pandas import concat

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean\_squared\_error

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.callbacks import EarlyStopping

# convert series to supervised learning

def series\_to\_supervised(data, n\_in=1, n\_out=1, dropnan=True):

n\_vars = 1 if type(data) is list else data.shape[1]

df = DataFrame(data)

cols, names = list(), list()

# input sequence (t-n, ... t-1)

for i in range(n\_in, 0, -1):

cols.append(df.shift(i))

names += [('var%d(t-%d)' % (j+1, i)) for j in range(n\_vars)]

# forecast sequence (t, t+1, ... t+n)

for i in range(0, n\_out):

cols.append(df.shift(-i))

if i == 0:

names += [('var%d(t)' % (j+1)) for j in range(n\_vars)]

else:

names += [('var%d(t+%d)' % (j+1, i)) for j in range(n\_vars)]

# put it all together

agg = concat(cols, axis=1)

agg.columns = names

# drop rows with NaN values

if dropnan:

agg.dropna(inplace=True)

return agg

# load dataset

dataset = read\_csv('pollution.csv', header=0, index\_col=0)

values = dataset.values

# integer encode direction

encoder = LabelEncoder()

values[:,4] = encoder.fit\_transform(values[:,4])

# ensure all data is float

values = values.astype('float32')

# normalize features

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled = scaler.fit\_transform(values)

# frame as supervised learning

reframed = series\_to\_supervised(scaled, 1, 1)

# drop columns we don't want to predict

reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)

print(reframed.head())

被轉化後的資料集包括8個輸入變數（輸入特徵）和1個輸出變數（當前時刻t的空氣汙染值，標籤）

　　資料集的處理比較簡單，還有很多的方式可以嘗試，一些可以嘗試的方向包括：

　　1. 對“風向”特徵啞編碼；

　　2. 加入季節特徵；

　　3. 時間步長超過1。

　　其中，上述第三種方式對於處理時間序列問題的LSTM可能是重要的。

4.構建模型

將處理後的資料集劃分為訓練集和測試集。為了加速模型的訓練，僅利用第一年資料進行訓練，然後利用剩下的4年進行評估。

　　下面的程式碼將資料集進行劃分，然後將訓練集和測試集劃分為輸入和輸出變數，最終將輸入（X）改造為LSTM的輸入格式，即[samples,timesteps,features]。

執行程式碼列印訓練集和測試集的輸入輸出格式。

**程式碼:**

# split into train and test sets

values = reframed.values

n\_train\_hours = 365 \* 24

train = values[:n\_train\_hours, :]

test = values[n\_train\_hours:, :]

# split into input and outputs

train\_X, train\_y = train[:, :-1], train[:, -1]

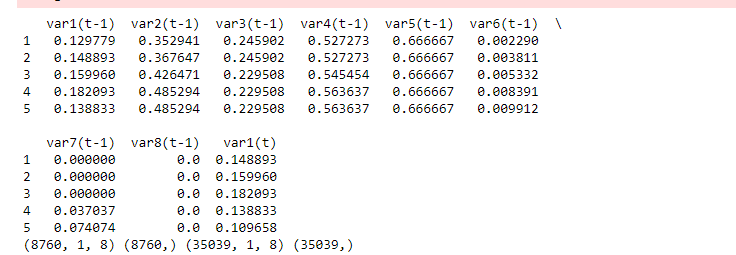
test\_X, test\_y = test[:, :-1], test[:, -1]

# reshape input to be 3D [samples, timesteps, features]

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

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

print(train\_X.shape, train\_y.shape, test\_X.shape, test\_y.shape)



5.搭建LSTM模型

LSTM模型中，隱藏層有50個神經元，輸出層1個神經元（迴歸問題），輸入變數是一個時間步（t-1）的特徵，損失函式採用Mean Absolute Error(MAE)，優化演算法採用Adam，模型採用50個epochs並且每個batch的大小為72

在fit()函式中設定validation\_data引數，記錄訓練集和測試集的損失，並在完成訓練和測試後繪製損失圖。

加入EarlyStopping來使訓練有效率。

**程式碼:**

# design network

model = Sequential()

model.add(LSTM(50, input\_shape=(train\_X.shape[1], train\_X.shape[2])))

model.add(Dense(1))

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

# fit network

early\_stopping = EarlyStopping(monitor='val\_loss', patience=50, verbose=2)

history = model.fit(train\_X, train\_y, epochs=50, batch\_size=72, validation\_data=(test\_X, test\_y), verbose=2, shuffle=False callbacks=[early\_stopping])

# plot history

pyplot.plot(history.history['loss'], label='train')

pyplot.plot(history.history['val\_loss'], label='test')

pyplot.legend()

pyplot.show()

# make a prediction

yhat = model.predict(test\_X)

test\_X = test\_X.reshape((test\_X.shape[0], test\_X.shape[2]))

# invert scaling for forecast

inv\_yhat = concatenate((yhat, test\_X[:, 1:]), axis=1)

inv\_yhat = scaler.inverse\_transform(inv\_yhat)

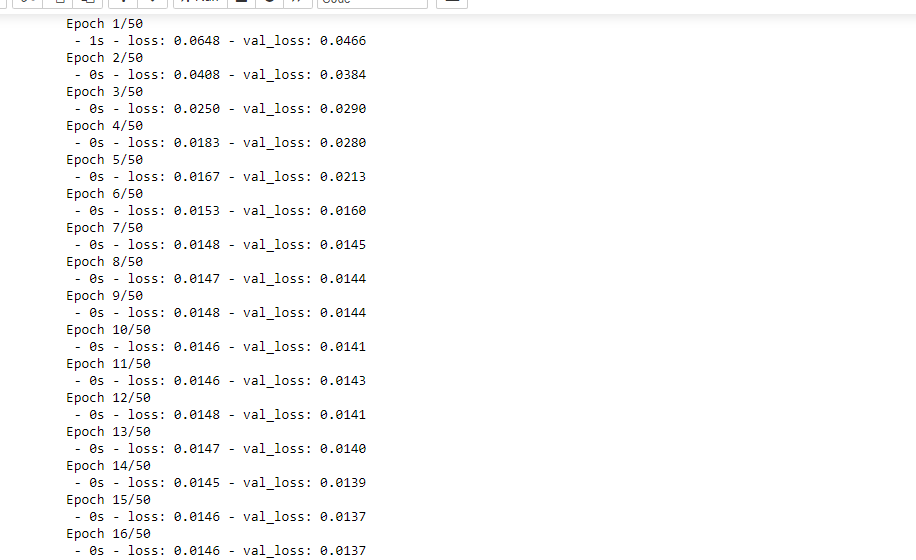
inv\_yhat = inv\_yhat[:,0]

# invert scaling for actual

test\_y = test\_y.reshape((len(test\_y), 1))

inv\_y = concatenate((test\_y, test\_X[:, 1:]), axis=1)

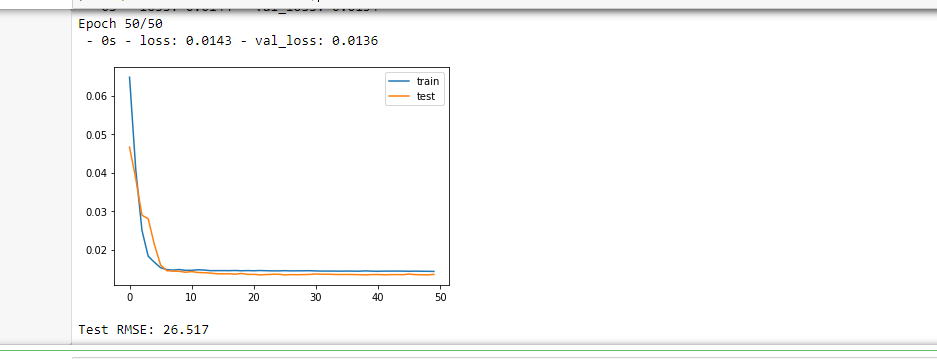
inv\_y = scaler.inverse\_transform(inv\_y)

inv\_y = inv\_y[:,0] 

6.對模型進行評估

需要將預測結果和部分測試集資料組合然後進行比例反轉（invert the scaling），也需要將測試集上的預期值也進行比例轉換。

通過以上處理之後，再結合RMSE（均方根誤差）計算損失。



**使用其它方法:**

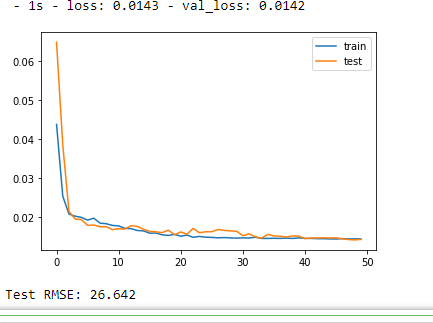
1. 將輸入改為一次三小時

在呼叫serises\_to\_supervised()將以下程式碼加入

# specify the number of lag hours

n\_hours = 3

n\_features = 8



實驗結果RMSE並沒有降低

1. 優化器換成adagrad並調整各項參數

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Batch size | 100 | 120 | 120 | 110 | 50 | 30 |
| patience | 50 | 50 | 100 | 200 | 30 | 300 |
| N\_hour | 2 | 2 | 2 | 4 | 2 | 5 |
| N\_feather | 10 | 10 | 10 | 10 | 10 | 8 |
| epoch | 50 | 50 | 50 | 50 | 50 | 50 |
| RMSE | 23.853 | 24.660 | 24.320 | 24.185 | 24.073 | 26.535 |

各項參數說明:

* patience：意即可以容忍多少epoch都沒有再提升準確度，如果patience設的大，最终得到的準確度要略低於模型可以達到的最高準確度；如果patience設的小，模型很可能在前期擺動，還在做全域搜索的階段就停止了，準確度一般很差。
* batch size : 適當增加可以提高記憶體利用率，使得梯度下降方向準確度增加，訓練時震動幅度減小
* epoch: 隨著epoch數量增加，神經網路中的權重的更新次數也在增加，曲線從欠擬合變得更加擬合。
* n\_hours, n\_feathers: 用於指定輸入以及輸出列，以n\_hours\*n\_feathers作為前n小時所有特性的obs輸入。

1. 優化器換成AdaDelta並調整各項參數

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Batch size | 100 | 100 | 100 | 200 | 500 | 350 |
| patience | 50 | 200 | 500 | 100 | 100 | 100 |
| N\_hour | 2 | 2 | 2 | 2 | 2 | 2 |
| N\_feather | 10 | 10 | 10 | 10 | 10 | 10 |
| epoch | 50 | 50 | 50 | 40 | 50 | 50 |
| RMSE | 25.585 | 25.894 | 25.541 | 25.666 | 27.168 | 27.168 |

1. 將優化器換成AdaDelta，並將於adagrad跑出誤差最低之參數組合進行實驗
2. 由使用adagrad優化器的實驗過程發現增加patience至一定程度可以減少誤差值，由第二次增加至200
3. 再將patience值增加至500，誤差僅小小減少，因此可能需要藉由調整其它參數才能達到更加減少誤差的效果
4. 調整Batch size，誤差稍微增加
5. 再將Batch size增加至500，誤差大幅增加
6. 將Batch size調小誤差上升

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 8 | 9 | 10 | 11 | 12 |
| Batch size | 300 | 200 | 200 | 50 | 50 | 50 |
| patience | 100 | 400 | 400 | 100 | 100 | 100 |
| N\_hour | 2 | 2 | 2 | 2 | 2 | 2 |
| N\_feather | 10 | 10 | 12 | 12 | 12 | 12 |
| epoch | 50 | 50 | 50 | 50 | 100 | 1000 |
| RMSE | 25.878 | 25.523 | 16.788 | 14.420 | 15.480 | 22.233 |

1. 將Batch size調整至300，誤差仍然比200時還大
2. 將Batch size保留仍為200，將patience值調大至400，誤差稍微降低

調整輸入的小時及特徵

(9)分別將N\_hour設為2，N\_feather設為12，誤差大幅降低至16.788

(10)調整Batch size以及patience，誤差值下降至14.420

(11)根據此進行訓練，epoch調整至100，誤差值增加至15.480

(12)將epoch調整至1000，搭配early stoppping，訓練至118次時停止，誤差上升至22.233

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Batch size | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| patience | 400 | 400 | 400 | 400 | 400 | 400 | 400 | 400 |
| N\_hour | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| N\_feather | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| epoch | 50 | 1000 | 100 | 200 | 300 | 250 | 150 | 200 |
| RMSE | 9.603 | 9.803 | 8.432 | 8.223 | 9.126 | 8.862 | 14.106 | 7.980 |

(13)由於先前實驗將patience調大有助於降低誤差，因此將patience調成400，且將小時及特徵分別切成2及12亦有助於增加預測準確度，另外調整Batch size為100，發現誤差有所降低。

(14)以1000 epoch搭配early stoppping進行訓練，於807 epoch時停止，誤差不減反升。

(15)將epoch調整成訓練100次，誤差降低。

(16) )將epoch調整成訓練200次，誤差降低。

(17) )將epoch調整成訓練300次，誤差回到9點多。

(18)再將epoch往上調整，誤差有減少，可是仍然比200時高。

(19) 將epoch調整成訓練150次，預測準度大幅降低許多。

(20)重新對第十六次所設定之參數進行實驗，誤差為目前之最低。

4.以實驗較佳之參數組合搭配不同優化器測試，並分別先進行50 epoch的訓練

(1)參數值分別為以下:

Batch size=100

Patience=400

N\_hour=2

N\_feather=12

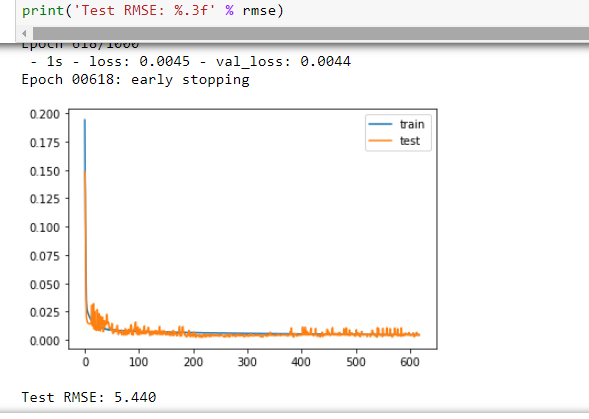
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 優化器 | Nadam | Adamax | RMSprop | SGD |
| RMSE | 9.921 | 15.424 | 8.962 | 40.234 |

(2)以以上優化器跑出之最小誤差進行實驗

以同樣的參數並選擇RMSprop，藉由調整epoch次數嘗試降低RMSE

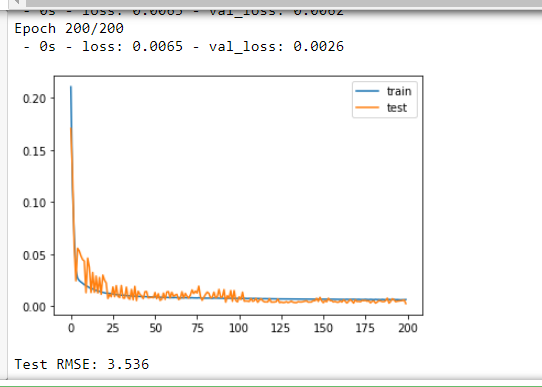
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| epoch | 100 | 1000 (搭配early stopping) | 200 | 300 |
| RMSE | 6.391 | 5.440 | 3.536 | 6.391 |

訓練100 epoch時結果的誤差稍微降低，因此搭配early stopping設定訓練1000 epoch，在第618 epoch時停止(下圖所示)，誤差降低相當少。



由於並不是epoch 調整的越多次就會誤差越小，因此改為由200 epoch進行實驗(如下圖)，RMSE降低至目前為止最低。

當再往上調整訓練300 epoch 時，RMSE上升，推測較佳預測之epoch 次數在約略200上下。



5.以實驗較佳之參數組合搭配不同activation測試，並分別先進行50 epoch的訓練

(1)使用激活函數的原因:

若不使用激活函數，每一層的輸出都會是上層輸入的線性函數，無論神經網路內有多少層，輸出輸入都會是線性的組合，使用激活函數可以在神經元內等入非線性因素，使神經網路可以逼近任何非線性函數，這樣神經網路可以應用到更多非線性的模型，在運用上也比較符合實際有多複雜變數的現況。

(2)常見的激活函數有以下:

* Sigmoid: 又稱為Logistic函數，用於隱藏神經元的輸出，可以把一個實數映射到(0,1)的區間，可以用來做二分類。適合用於前向傳播，可以壓縮數據並保證數據幅度不會有問題，可是容易出現梯度消失問題，且收斂較為緩慢。
* Tanh: 又稱雙切正切函數，取值範圍在[-1,1]，與sigmoid的區別為tanh的0是均值的，在實際應用上會比sigmoid更好。
* ReLU: 當輸入x<0時輸出為0，當x>0時輸出為x。優點為SGD算法的收斂速度比sigmoid、tanh更快，且梯度不會飽合，因此解決了梯度消失的問題，計算複雜度也低，不用進行指數運算，很適合用於後向傳播。缺點就是會有Dead ReLU Problem(神經元壞死現象)，有些神經元不會被激活到，讓相對應的參數不會被更新。
* Softmax: 其主要用於多類分類。
* Elu: 是”指數線性單元”，能通過正值的標識來避免梯度消失的問題。Elu有ReLU的所有優點，且不會有Dead ReLU的問題，輸出的均值也接近0。
* Softplus: 其為sigmoid的原函式，可以將其看成ReLU較平滑的版本，其對全部資料進行非線性對映，是一種不飽合非線性函式，收斂速度較ReLU慢，但因變化較為平緩，更接近生物學的啟用特性，也解決了sigmoid函式的假飽合現象，易於網路訓練，泛化效能也提高，在表現效能方面更優於ReLU及sigmoid，但是神經網路的學習速度並沒有加速。

(3)各激活函數實驗之成果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| activation | relu | tanh | softmax | elu | selu | softplus |
| RMSE | 10.233 | 19.049 | 538.698 | 18.332 | 10.730 | 51.073 |

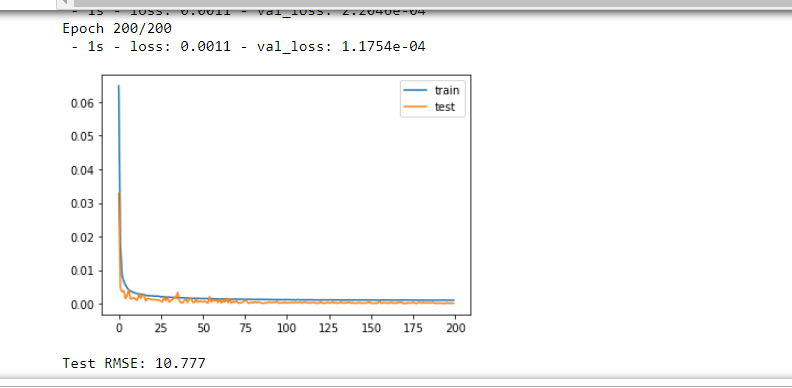
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| activation | linear | softsign | hard\_sigmoid | sigmoid | exponential |
| RMSE | 24.676 | 87.973 | 7.674 | 56.644 | 52.534 |

6.增加Dropout層

(1)為了避免過度擬合的情況，在訓練時忽略一部份特徵檢測器(意即讓一半隱藏層節點值變成0)，可以減少隱藏節點間的交互作用，在向前傳播時可以使模型泛化能力更強，就不會太依賴某些區域性的特徵。

(2)儘管實驗結果並沒有過度擬合的情況，但仍以目前實驗較佳之參數組合增加兩層dropout層來進行測試。

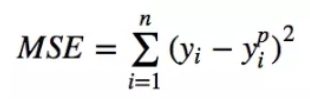
實驗結果如下圖，增加dropout層使得誤差加大。



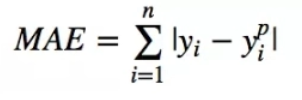
7.調整損失函數

(1)預計採用進行實驗之損失函數介紹:

* Mse: 是常用的回歸損失函數，是求預測值與真實值間距離的平方和，公式如下圖。



* Mae: 另一種用於回歸模型的損失函數，是目標值和預測值之差的絕對值之和，其只衡量預測值誤差的平均模長，不考慮方向，取值範圍從0到正無窮。公式如下圖。



(2)分別以有兩層dropout以及沒有dropout層來比較實驗結果

參數採用目前實驗較佳之組合，並訓練50epoch:

|  |  |  |
| --- | --- | --- |
|  | 兩層dropout | 沒有dropout層 |
| 採用mae之RMSE | 17.895 | 8.842 |
| 採用mse之RMSE | 13.232 | 8.625 |

藉由以上實驗，目前使用mse相較mae，RMSE較小。

8.選取特定欄位並使用不同input實驗

(1)更改train、test reshape的input(如下圖):



(2) 更改invert scaling for actual部分



(3)訓練50 epoch 藉由調整batch\_size進行實驗:

|  |  |  |  |
| --- | --- | --- | --- |
| batch\_size | 100 | 72 | 30 |
| RMSE | 11.992 | 6.494 | 10.731 |

由以上可知，batch\_size大誤差較大，但也非越小越好。

(4)延續以batch\_size=72 分別實驗mae以及mse 誤差

|  |  |  |
| --- | --- | --- |
| Loss function | mae | mse |
| RMSE | 13.261 | 8.435 |

在此次實驗當中，mse訓練後的誤差相較mae較小。

(5)loss function為mae 分別以不同epoch進行實驗:

|  |  |  |  |
| --- | --- | --- | --- |
| Epoch | 50 | 100 | 200 |
| RMSE | 8.435 | 9.592 | 9.824 |

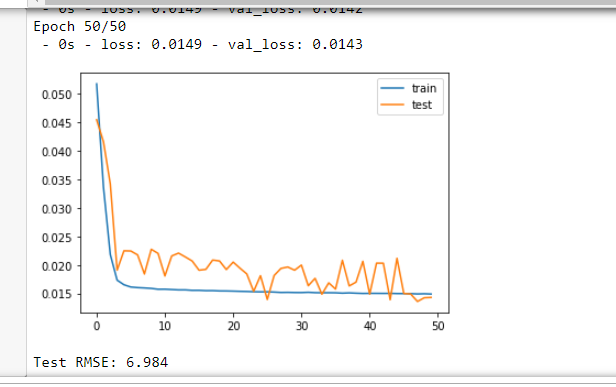
Epoch並非越多越好，可能需要更多實驗來尋找更好的epoch值。

(6) loss function為mse 分別以不同epoch進行實驗:

|  |  |  |  |
| --- | --- | --- | --- |
| Epoch | 50 | 100 | 200 |
| RMSE | 8.122 | 9.426 | 10.192 |

(7)加入learning\_rate進行實驗(如下圖)

同樣採用RMSprop優化器，batch\_size=72並訓練50 epoch



**結果分析**

目前最佳實驗結果僅將RMSE由最初之26.517調整至3.536。

其是將預測的小時切割並對特徵分類，模型部分採用LSTM，優化器選擇RMSprop，並將patience值調整為400，batch\_size調整為100，訓練200 epoch。

在實驗過程當中修改了許多參數，亦套用了不同的優化器進行實驗，目前的實驗成果還有相當大進步空間，未來若使用更適模型可能更有機會能將誤差值降低。

**結論**

在台灣十大死因中，有七大和空氣污染密切相關。

肺癌更早在二○○一年，躍居台灣癌症死亡榜之首，並經常榜上有名。每年因罹患肺、支氣管和氣管相關癌症而死亡者，達八千六百多人，三十年內成長五倍之多。另一方面，因為肺炎、氣管和支氣管炎等呼吸系統健康問題，到醫院就診的人次，也高居所有病因的第一位，約佔就診人次七成，所需診療費用超過一千億元。

由台北市的空污地圖，或許更能按圖索驥。距離台大醫院不到五公里的古亭測站，過去十年的空氣污染指標（PSI）居台北市七大測站之冠，比附近的中山、士林、松山和萬華站都高。

細懸浮微粒無所不在，除沙塵暴或河川揚塵等自然生成，到汽機車、工廠排放所產生的空氣污染物經過反應，都足以生成。

ＰＭ2.5在台灣已經不是陌生的名詞，也是各國政府關心的指標。未來若能透過對資料更精準的預測來使得人類對空汙有所防範，方能有效降低因空汙引起之相關災害或是對人體的傷害。

**參考文獻**

<https://www.cw.com.tw/article/article.action?id=5054076>

<https://blog.csdn.net/silent56_th/article/details/72845912>

<https://www.zhihu.com/question/32673260>

<https://www.itread01.com/content/1550898141.html>

<https://www.itread01.com/content/1546354994.html>

<https://kknews.cc/zh-tw/tech/ogjgm95.html>

<https://zhuanlan.zhihu.com/p/38200980>

<https://bigdatafinance.tw/index.php/news/607-5>

<https://tanet2019.nsysu.edu.tw/assets/TANET2019_thesis/B2_009.pdf>