2020.06.02 Zhi-Hong

了解,接下來會處理各集成效果

W\_V = workday\_validation

W\_NV = workday\_Non-validation

H\_V = Non-workday\_validation

H\_NV = Non-workday\_Non-validation

modeA = 一起正規化 再切工作日非工作日

modeB = 先切工作日非工作日,個別正規化

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ModeA: W\_V | ModeA:W\_NV | ModeB: W V | ModeB:W\_NV |
| ModeA:H\_V | MAPE = 3.77%  RMSE = 12.67 | MAPE = 3.86%  RMSE = 13.26 | MAPE = 3.87%  RMSE = 13.22 | MAPE = 3.92%  RMSE = 14.66 |
| ModeA:H\_NV | MAPE = 3.91%  RMSE = 12.64 | MAPE = 3.99%  RMSE = 13.22 | MAPE = 4.01%  RMSE = 13.19 | MAPE = 4.06%  RMSE = 14.64 |
| ModeB:H\_V | MAPE = 3.95%  RMSE = 12.76 | MAPE = 4.03%  RMSE = 13.33 | MAPE = 4.05%  RMSE = 13.30 | MAPE = 4.10%  RMSE = 14.73 |
| ModeB:H\_NV | MAPE = 3.94%  RMSE = 12.71 | MAPE = 4.02%  RMSE = 13.29 | MAPE = 4.04%  RMSE = 13.26 | MAPE = 4.09%  RMSE = 14.69 |

* 從以上得知,A:W\_V配A:H\_V的效果是最好的
* 在A:W的模式下,有切驗證比較好,在B:H的模式下反而沒有比較突出
* 其原因可能是因為在B:H\_NV有出現一組loss跟其他自己類型的model差不多的情況,但是卻單獨MAPE到10%左右,其被拉下整體績效.

2020.05.31 Hsu

除了分別計算Workday及non-workday個別的績效之外，整體(包含workday及non-workday)的績效也需要計算。由於workday及non-workday兩邊的天數不一樣，不可直接將兩個數值加總除以二。

2020.05.29 Zhi-Hong

我將兩邊模式的input跟model固定 約有16組的ensembles可以比對

* Mode A:
  + 一起正規化(除Training最大值)再拆Workday / Non-Workday
  + 再各別分成兩種模式:有切validation(5%)跟沒切validation
* Mode B:
  + 先拆Workday / Non-Workday dataset,再各別正規畫
  + 再各別分成兩種模式:有切validation(5%)跟沒切validation
* Model Structure:
  + CuDNNLSTM\*3 (unit = 256 too)
  + Add BatchNormalization
  + Hidden layer = 196 , relu
  + Dropout = 0.3
  + Output layer = 96 , relu
  + Batch size = 4
  + Early\_stopping = 55
  + ReduceLearningRate :
    - Factor = 0.7
    - Patience = 5
    - Min\_lr = 0.0001
* Input Feature:
  + Past two day of same week\_update of per 15mins of 'measure', 'Period\_transform', 'isHoliday', 'Temp\_H,
  + Today of per 15mins of 'Period\_transform', 'week\_update','isHoliday','dayOfYear', 'Temp\_H'
* Output:
  + Today of per 15mins of measure.
* Function:
  1. 針對modeA跟B,沒有切validation\_data的模式
     + 一次產生10組model
     + 針對其min(loss),排除異常可能練壞的model再重新產生直到補足10組
  2. 針對modeA跟B,有切validation\_data的模式
     + 每次執行都亂數切5% validation data ,總共10組
     + 針對其最低的val\_loss model的validation data組合,再練出10組
     + 針對其min(val\_loss),排除異常可能練壞的model再重新產生直到補足10組
  3. 一共會有各10組的
     + 模式A 有切validaiton的workday
       - Ensemble: MAPE 3.59 / RMSE 16.21
     + 模式A 沒有切validaiton的workday
       - Ensemble: MAPE 3.36 / RMSE 14.10
     + 模式A 有切validaiton的Non-workday
       - Ensemble: MAPE 5.11 / RMSE 8.83
     + 模式A 沒有切validaiton的Non-workday
       - Ensemble: MAPE 4.34 / RMSE 8.96
     + 模式B 有切validaiton的workday
       - Ensemble: MAPE 3.52 / RMSE 16.70
     + 模式B 沒有切validaiton的workday
       - Ensemble: MAPE 3.62 / RMSE 14.83
     + 模式B 有切validaiton的Non-workday
       - Ensemble: MAPE 5.05 / RMSE 9.13
     + 模式B 沒有切validaiton的Non-workday
       - Ensemble: MAPE 5.28 / RMSE 9.32
  4. 將以上8種各種ensemble 並排列組合搭配 共計16組左右 可得其MAPE RMSE

2020.05.26 Zhi-Hong

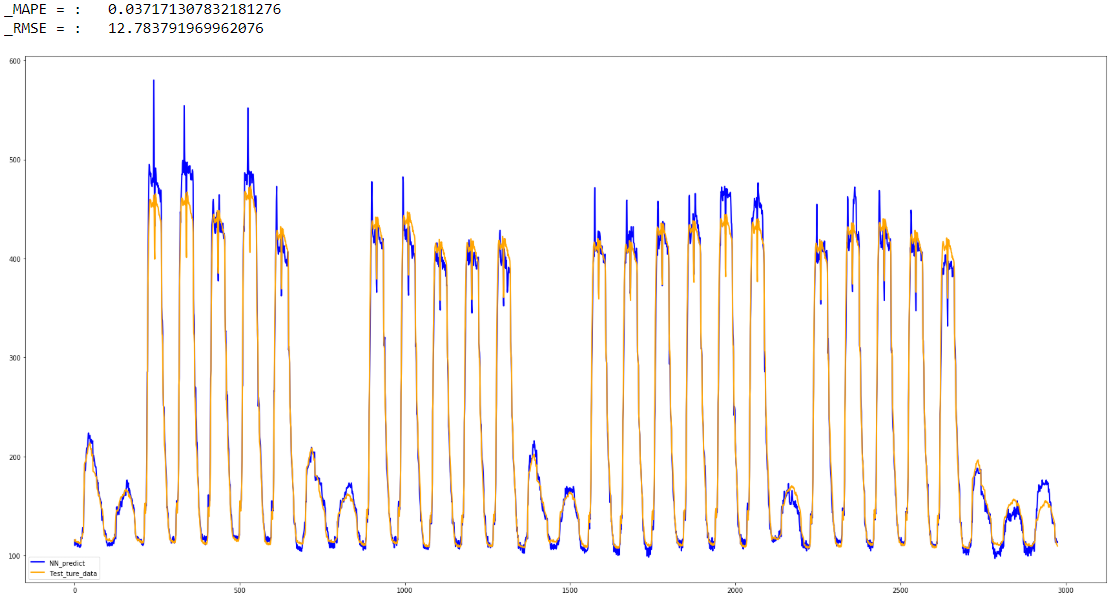
這次統整一下non-working day的部分

* Anyway , To split any validation\_dataset can’t let validation dataset represent original Non-Working day dataset of distribution.
* The original non-working datasets are too few.
* The input feature too many that the model can’t effective train with past LSTM model structure on my GPU , if I add additional some hidden layers or hidden units, the GPU will be crush.
* If don’t split any validation dataset , the model will dependent on early stopping of loss target, wherefore, it’s important to set early stopping of patience , which control the model loss to prevent model overfitting or underfitting.
* The best MAPE is between 0.0095 and 0.0085 within range of loss, if the model of loss beyond or lower than the in short range of loss, the prediction of MAPE will be garbage.
* If I set the batch\_size 8 , the model can’t effective train , must be setting 4 .
* As stated before, my LSTM model structure is:
* Input Feature:
  + Past two day of same week\_update of per 15mins of 'measure', 'Period\_transform', 'isHoliday', 'Temp\_H,
  + Today of per 15mins of 'Period\_transform', 'week\_update','isHoliday','dayOfYear', 'Temp\_H'
* Output:
  + Today of per 15mins of measure.
* Normalization:
  + Non-Working day\_train\_dataset divide by Non-Working day\_train\_dataset of per feature of max value
  + Non-Working day\_test\_dataset divide by Non-Working day\_train\_dataset of per feature of max value

|  |  |
| --- | --- |
| Model structure: | Result: |

The most models of mape is between 4.3%~5.5% , and the emsemble mape is 4.35%

To combine non-workday result and the past workday\_ensemble result , the mape is 3.71%

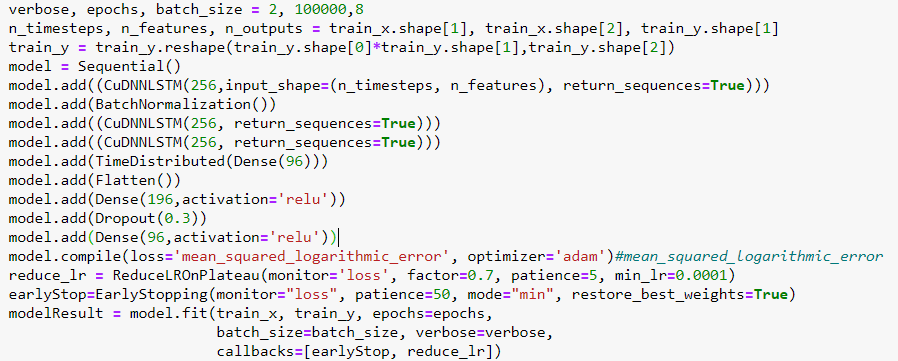
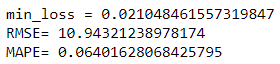


接下來將會跑[先拆開在各別正規畫的workday]這部分的績效,由於本次是將之前(先正規劃在拆開的workday)結合這次(先拆開各別正規畫的holiday)的績效,input不同,所以我這邊在做看看說如果按照後者的模式 看看會不會在更下降

2020.05.24 Zhi-Hong

這幾天再研究如何處理non-working day走經的現象

這裡稍微統整一下改變方式

* 將dayOfYear\_tramsform改為dayOfYear
  + 將input任何dayOfYear\_tramsform改為dayOfYear
  + dayOfYear of normalization divide by 365
  + 這裡不切割validation\_data
  + 架構維持不變
  + 績效:
  + 
  + 有夠遭 繼續研究
* 砍掉幾個變數
  + 將過往相同天的dayOfYear,'Hour', 'Minute', week\_update砍掉
  + 這裡不切割validation\_data
  + 架構小調 不然GPU太容易死機 太當
  + 
  + 績效:
  + 
  + 還不夠好

2020.05.21 Zhi-Hong

本篇針對國泰用電:工作日與非工作日實驗1的正規化做調整

實驗1的部分是先全體正規劃再拆工作日與非工作日的training/test dataset

這裡改為先拆工作日與非工作日的training/test dataset,在各別正規化

Workday dataset and non-working day dataset:

* Input:
  + Past two day of same week\_update of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
  + Today of per 15mins of 'Period\_transform', 'week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H'
* Output:
  + Today of per 15mins of measure.
* Normalization:
  + Workday dataset:
    - To split workday dataset -> workday\_train\_dataset and workday\_test\_dataset
    - Workday\_train\_dataset divide by workday\_train\_dataset of per feature of max value
    - Workday\_test\_dataset divide by workday\_train\_dataset of per feature of max value
  + Workday validation dataset:
    - To run 10 times with the experiment 1 of model structure and per times random split validation dataset 5% from workday\_train dataset, and then , choicing validation\_dataset of model of min(val\_loss) to facilitate subsequent testing.
  + Non-Working day dataset:
    - To split Non-Working day dataset -> Non-Working day\_train\_dataset and Non-Working day \_test\_dataset
    - Non-Working day \_train\_dataset divide by Non-Working day \_train\_dataset of per feature of max value
    - Non-Working day \_test\_dataset divide by Non-Working day \_train\_dataset of per feature of max value
  + Non-Working day validation dataset:
    - To run 10 times with the experiment 1 of model structure and per times random split validation dataset 5% from Non-Working day \_train dataset, and then , choicing validation\_dataset of model of min(val\_loss) to facilitate subsequent testing.

測試部分:

1. Work\_day:

由於work\_day\_dataset的部分跟國泰實驗一沒兩樣 所以就稍微不贅述

|  |  |
| --- | --- |
|  | 績效跟實驗一的沒兩樣 |

1. non-working day:

|  |  |
| --- | --- |
|  | 不管怎麼切 validation\_loss都不如預期 雖然validation\_loss有下去  但mape還是沒有變好的趨勢  跑了好幾次 幾乎都卡在這8%-10%這區間 |
| 本次實驗的結果:    過去實驗一的結果: | 上面為這次8.7%左右的 可以發現他的效果不如下面那張圖 |

不切 non-working day\_validation\_dataset去作測試

|  |  |
| --- | --- |
|  | 這邊不切validation出來做training,其predict出來的績效比不切還爛 |
|  | 這裡將batch\_size調成4  有比較好點 其績效只能說還是很浮動 |

由於以上將batch\_size = 4 的時候其有夠lag

所以想說看能不能換個方法能達到差不多的泛化能力而又比以上結構精簡點

Input不變,結構上改變,

有切validation data 5%

|  |  |
| --- | --- |
|  | 降低hidden layer層數 擴增一層LSTM  其績效    還是很糟  但好像不管怎樣batchsize=4的時候比較好 |

* 國泰用電:工作日與非工作日實驗1 – LSTM系列

(先整體Normalization再拆Train/test)

(1月至11月為Training dataset ,各別feature除以最大值)

(12月為Tese dataset 除以training dataset各別feature最大值)

2020.05.21 Zhi-Hong

目前針對上次整合工作日以及非工作日的績效做個紀錄

根據5/14工作日以及上次的非工作日做一次統整 做出完整12月的評估

其績效為:

* + MAPE: 0.04005018883428556
  + RMSE: 13.16633498954641

非工作日部分有將驗證資料急切固定筆數以及index出來

* 驗證資料集共取6筆,依序為 non-working day\_training dataset的第5,20,35,50,65,80筆
* non-working day的feature跟5/13號的紀錄一致不變
* 架構的部分有稍微更改

|  |  |
| --- | --- |
|  | 考慮到dataset過少,因此將batch\_size調低  然後LR的factor改為0.7 |
| Holiday Ensumble | 這個是根據上面的架構所跑的10次結果  因為batch\_size調低  所以我這台很常出現GPU掛掉的情況  但還是勉強練得出來  Val\_loss基本上都挑低於0.004以下的  其集成為5.2%  直接平均5.7% |

以下做出本次結論:

1. 非工作日的dataset 太少 不好練
2. Validation\_data由於受到整體dataset所以受限他的整體代表性,且這次是手動切割,所以不太能完整的代表整體dataset的分布使model能夠有效訓練
3. 由於整個12月的總體MAPE 4%左右,有一半敗在非工作日的部分,目前在看有什麼辦法可以使非工作日達到4%左右,目前都位於5%~8%,且不好練

2020.05.14 Zhi-Hong

回老師 這組lr的設定是來自學姊 我這邊還沒對此API做任何參數改變的實驗.

2020.05.14

* ReduceLRonPlateau中的factor = 0.001, min\_lr = 0.001。如果initial\_lr是1的話，一次的reduce learning rate調整(lr\*0.001)就達到min\_lr，後續此API就沒有作用了，對吧？

2020.05.13

目前實驗結果紀錄個:

另外回報目前我跟學姊的input差異

目前我跟學姊的input皆為一樣

目前先將week\_update中1~5 為工作日 6~7為非工作日拆分

為工作日找好model

Input:

* Past two day of same week\_update of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H'

Output:

* Today of per 15mins of measure.

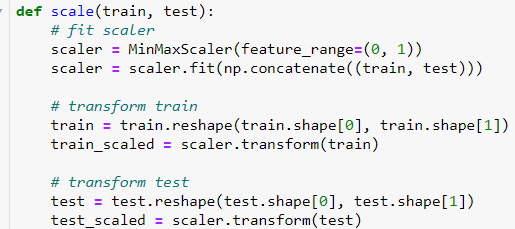
但Normalization則有些不同 目前正在討論當中

* 我的normalization為 all dataset feature value 除已training dataset feature value 最大值(Period\_transform 除外)



學姊的則是

* 將training and test 先合併 在將training and test 個別轉換



兩邊所得出來的值不同，所以這邊學姊在更正成我這邊的正規化並在使用我的model再重複實驗。

|  |  |
| --- | --- |
| 我的(那個2為Period\_transform 沒有正規劃到) | 學姊的(0過多 我懷疑是因為權重不管怎麼乘都是0 只能依靠bias) |
|  |  |

回歸正題：

我這邊有做基本的相關實驗, 並把這六日到今日的過程紀錄一下

|  |  |
| --- | --- |
| Model 結構 | 評估 |
|  | Test1:  這邊看之前paper表示雙向的具有未來學習能力,所以我認為雙向具有較好的泛化能力,因此我這邊嘗試用Bi-LSTM做做看  其績效為:    不太滿意 繼續嘗試下個實驗 |
|  | Test2:  那這邊則是，因為這是初期還在想說有沒有更好的模式，所以稍為大改，且資料量過少，同時參考學姊的model並沒有畫分validation dataset出來，所以我嘗試看看直接依靠loss會不會比較好。  其績效為:    不太滿意 繼續嘗試下個實驗 |
|  | Test3:  根據Test2的model結構,我就先嘗試拔掉雙向的框架,改成一般的LSTM.  其績效為:    恩 稍為進4了 繼續try |
|  | Test4:  這架構原本是參考學姊的架構,則是將LSTM層換成GRU層,學姊的架構會在下一個測試  其績效為:    有夠拉基 |
|  | Test5:  這是學姊原生的架構,我就拿來跑跑看  其績效為:    整組走經 |
|  | Test6:  這組根據Test4,只不過是將loss改成mse\_log  其績效為:    沒救,比lstm還糟糕 |
|  | Test7:  根據Test1,且看到keras有CuDnn的LSTM,想說先改用且使模型複雜點看能到什麼樣程度,所以試著改更雜  batchnormalization  疊加隱藏層  在增加點batch\_size  其績效為:    好 我錯了 |
|  | Test8:  我懷疑其績效不好是因為Test7沒有使用validation\_split去做驗證,所以造成overfitting,那我這邊切了驗證出來  其績效為:    出來了 但是後面重複執行則變化大 介於3-4之間, |
|  | Test9:  這裡我只將test8的loss改成mse\_log  其績效為:    出來了 但是後面重複執行則變化大 介於3-4之間, |
|  | Test10  我把Test9的batchnormalization拔掉跑跑看  但是跑3次 其中GPU2次執行出現GPU sync failed  其績效為: |
|  | 這裡將Test9 做集成  我這邊只挑val\_loss低於0.002以下的  其發現說他合出來mape為3.4%左右,相當不錯  但後面越想越不對勁,因為當我注意到val\_loss低於0.002以下時他的mape不錯,尤其資料量過少,validation\_dataset顯得更重要 因此才會在今天下午問老師您有關validation的問題. |

經過以上反覆測試,我認為Test9績效變好是因為幾點

1. 其泛化能力足夠
2. Validation\_split有挑到夠好驗證的validation dataset
3. Batchnormalization將第一層LSTM output做第一次正規化 其後續運算調整相當
4. 不放Batchnormalization我的硬體容易GPU sync failed,我預估是因為值太大不易調整
5. 目前來看 GRU效果沒有比較好
6. 雙向的沒有比單向得來的好
7. 太淺層的其效果真的有夠糟 不易收束
8. model.add(TimeDistributed(Dense(96)))我有試過改成128 192 其效果沒有比較好 但記錄刷掉了
9. 因為以上大部分我都只取1次(扣掉TEST9) 之後可能做實驗會固定validation\_dataset 不然之後可能會有誤判的情況 這次可能僅供參考

2020.05.01 Zhi-Hong

紀錄目前結果

目前最好情況: 過去兩周同天 掛reduce\_LR ensemble = 4.02

實驗一: 過去三周掛同天reduce\_lr

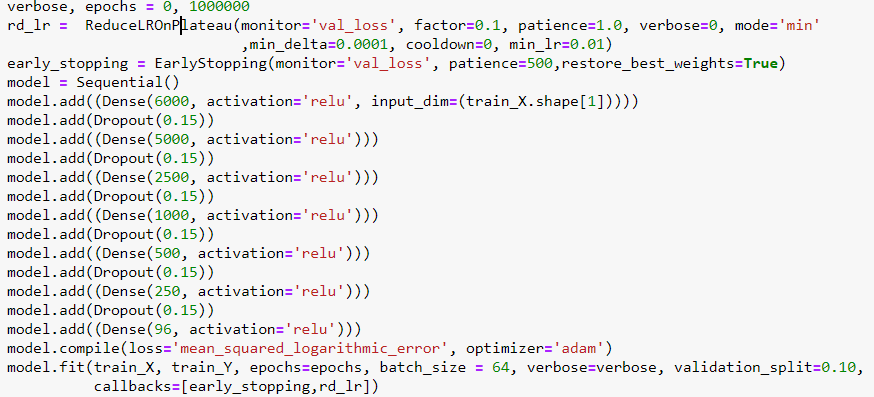
Input :

* Day-7, Day-14, Day-21 of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.11% | 4.38% | 4.37% | 4.00% | 4.08% | 4.39% | 4.43% | 3.93% | 4.29% | 4.14% |
| RMSE | 13.615 | 12.801 | 13.264 | 13.168 | 13.190 | 13.846 | 14.566 | 12.109 | 13.775 | 13.594 |

集成平均MAPE:4.05%

集成平均RMSE:12.99

直接平均MAPE:4.21%

直接平均RMSE:13.39

實驗二:過去兩周同天 不掛reduce\_LR

Input :

* Day-7, Day-14 of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.24% | 4.38% | 4.86% | 4.59% | 4.49% | 3.99% | 4.31% | 4.12% | 4.21% | 4.35% |
| RMSE | 13.695 | 14.197 | 15.874 | 14.292 | 13.903 | 12.975 | 13.061 | 13.697 | 13.152 | 14.190 |

集成平均MAPE:4.06%

集成平均RMSE:12.99

直接平均MAPE:4.35%

直接平均RMSE:13.904

實驗三:過去兩周同天 掛reduce\_LR

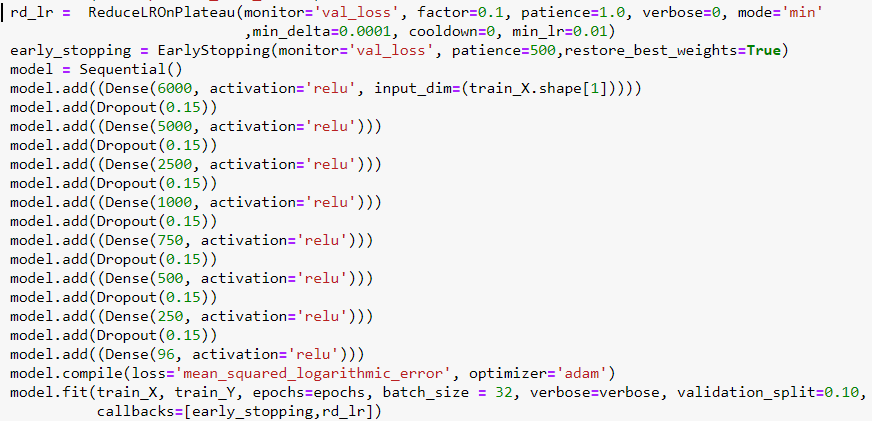
Input :

* Day-7, Day-14 of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.54% | 4.12% | 4.13% | 4.51% | 4.31% | 3.89% | 4.77% | 4.21% | 4.94% | 4.93% |
| RMSE | 14.201 | 13.521 | 14.079 | 13.954 | 13.185 | 13.176 | 14.523 | 13.862 | 15.079 | 13.975 |

集成平均MAPE:4.02%

集成平均RMSE:13.00

直接平均MAPE:4.43%

直接平均RMSE:13.95

實驗四:過去兩周同天+昨日 不掛reduce\_LR

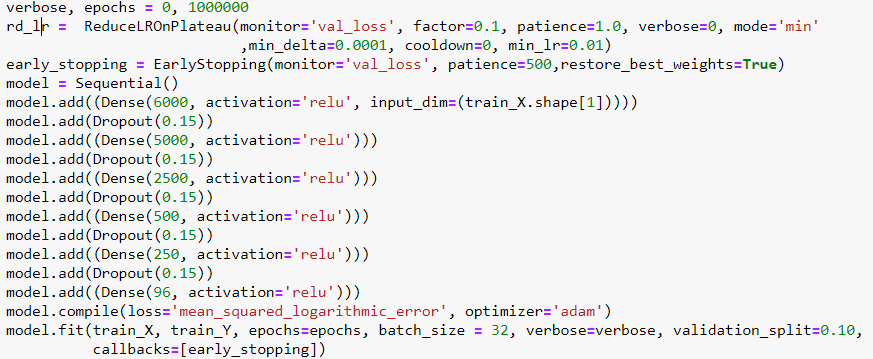
Input :

* Day-7, Day-14 and yesterday of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.14% | 4.41% | 4.38% | 4.90% | 4.55% | 4.19% | 4.46% | 4.39% | 4.24% | 4.61% |
| RMSE | 14.076 | 14.823 | 15.209 | 17.772 | 14.433 | 13.392 | 14.873 | 14.199 | 13.048 | 13.968 |

集成平均MAPE:4.12%

集成平均RMSE:13.47

直接平均MAPE:4.43%

直接平均RMSE:14.57

實驗四:過去兩周同天+前三日 不掛reduce\_LR

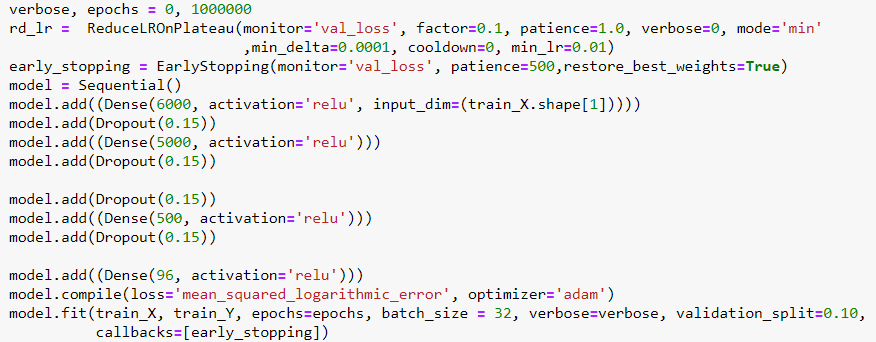
Input :

* Day-7, Day-14 and past 3 day of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.04% | 4.20% | 4.39% | 4.44% | 4.39% | 4.05% | 4.47% | 4.41% | 4.33% | 4.63% |
| RMSE | 12.964 | 12.612 | 13.551 | 13.216 | 12.762 | 12.719 | 13.246 | 13.097 | 13.381 | 13.546 |

集成平均MAPE:4.13%

集成平均RMSE:12.61

直接平均MAPE:4.33%

直接平均RMSE:13.10

2020.04.23 Zhi-Hong

先記錄一下結果目前結果

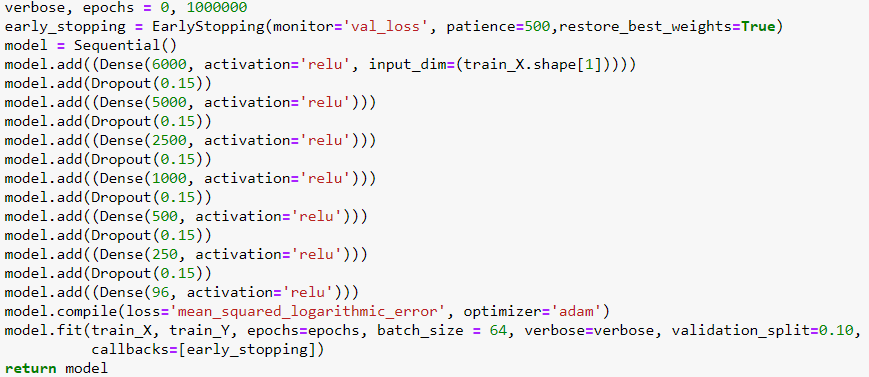
Input :

* Day-7, Day-14, Day-21 of per 15mins of 'measure', 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'
* Today of per 15mins of 'Period\_transform', week\_update','isHoliday','dayOfYear\_transform', 'Temp\_H', 'Hour', 'Minute'

Output:

* Today of per 15mins of measure.

Model Structure:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| MAPE | 4.16% | 4.05% | 3.84% | 4.36% | 4.22% | 4.46% | 4.20% | 4.57% | 4.56% | 3.97% |
| RMSE | 13.412 | 12.675 | 12.655 | 13.536 | 13.788 | 14.314 | 13.758 | 15.635 | 14.478 | 13.237 |

|  |  |
| --- | --- |
| 集成平均MAPE:4.06%  集成平均RMSE:13.237  直接平均MAPE:4.24%  直接平均RMSE:13.310 |  |

下面為個別增加架構而導致反彈的因素:

|  |  |  |  |
| --- | --- | --- | --- |
| MAPE | RMSE | 反彈原因 | 架構圖 |
| 10.37% | 38.531 | Output前多加一層hidden layer |  |
| 5.22% | 15.812 | 加入batchnormalization |  |
| 6.01% | 29.805 | 加入regulartion L2 |  |
| 18.98% | 29.287 | 加入regulartion L1\_L2 |  |
| 14.012% | 48.409 | 加入regulartion L1 |  |

2020.03.27 Zhi-Hong

阿災 好問題 不知道其他人有沒有放batch normalization

另外記錄一下今天的成果

Input :

* D-1 to D-7 per 15mins measure ,temp , hour, minute, day of year, period, week ,holiday
* D per 15mins temp, day of year, period, week ,holiday

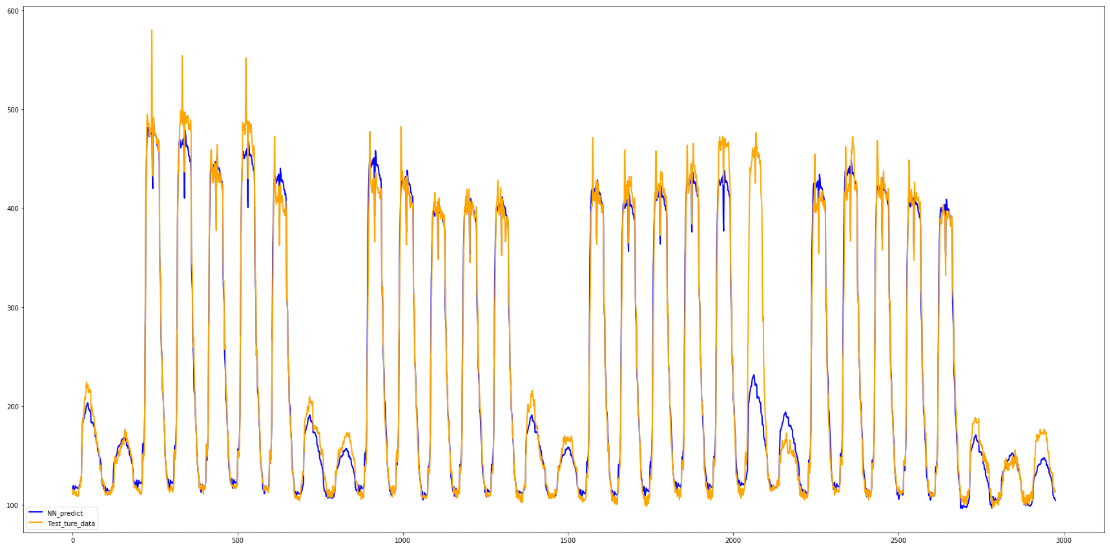
Output:

* D per 15mins measure

另外今天12/22我講反了 其實是沒問題的 橘色為真實值 藍色為預測值 拍謝

Mape: 5.34%

Rmse: 30.728



2020.03.27 Hsu

根據你的實驗結果，batch normalization是可以提升績效。不知其他的人，是否也有相同的結論？

2020.03.25 Zhi-Hong

工業用電收尾

懶人包 : 有正規化情況下 使用batchnormalization搭配adam and MSE會比較好(平均約落在1.8 1.9%左右 最低點有衝到1.71%)

由於batch normalization + MSE 整體效果會比較好

試試看扣掉batch normalization 發現他的績效變爛 整體落在2.2%左右

但是如果用batch normalization + hyperas 我這邊讓他疊代50次記憶體會不足(16G)

我這邊最高限制就是跑40次

我將相關檔案放入我的資料夾中

Code : \108 老師與治紘\Programs\Forecast\_7Days\END.ipynb

Excel : \108 老師與治紘\Results\forecast\_7\_days\_result\_ELEC\hyperas\_and\_10times\_model\_result.xlsx

2020.03.15 Hsu

現在看起來，peakload預測的誤差大部分落在1.5%左右，感覺這是較合理的預測結果。

2020.03.15 Zhi-Hong

先來總結一下”peakload”的部分

Output = D to D+6 peakload

Input為:

* D-1 to D-7
  + day of year(冬夏至轉換)
  + holiday
  + week
  + peakload
  + measure
  + temp
* D to D+6
  + day of year(冬夏至轉換)
  + holiday
  + week
  + temp

我這邊用hyperas 跑一次 mape為1.32% , Rmse為495.52 , min\_val\_loss = 0.0003



而10次結果為下圖

由左而右為 model最低的valloss / mape / RMSE



只要val\_loss落在0.001以下的 代表說他的MAPE不錯 有學到那個區間

我是用log\_mse , 所以valloss的對比很明顯

只要valloss>0.01 他的mape就不好

可以看到val\_loss在15跟45的時候他的績效奇差無比,且不一定說都保證valloss很低(包含hyperas 他沒中就是沒中 怎麼強求都沒用 他頂多確保它比較好的結果出現的機率比沒用來得高而已 不然就不會出現星期五我peakload mape 落在15%)

Hyperas還是可以用的 但我不知道其他model的情況 在MLPNN來說的確不錯

用hyperas的valloss < 10次不管哪一次的valloss

另外我將這次peakload的相關資料新增在我的檔案底下了

如果要看code 檔案在210 巨資成員\108 老師與治紘\Programs\Forecast\_D\_to\_D6\_peakload

包含輸出結果

那我這邊Peakload先告一段落 之後跑measure的部分 由於NN沒有時間步長 比其他model而言我這邊前處理的部分比較麻煩 那我到時候會丟初版的code上去 我會用English annotation , hyperas沒辦法打中文就這點麻煩而已, 我會開GPU加速整個運行時間 hyperas太慢

2020.03.08 Zhi-Hong

Upload algorithms.docx this report to \108 老師與治紘\Algorithms\forecast\_7\_days\_ELEC\_ANN+hyperas , 參考之前的algorithms做一些修改,特別增加的地方有用紅字另外標.

2020.03.07 Zhi-Hong

新增model 8 – 將特殊日改為用自己的特殊日 與學姊的進行比較 而early stopping調為100

特殊日包含:

* 過年期間2019/02/02 - 2019/02/10
* 228年假 2019/02/28 - 2019/03/03
* 清明期間2019/04/04 - 2019/04/07
* 勞動假期2019/05/01 - 2019/05/01
* 6月上旬 2019/06/07 - 2019/06/09
* 9月中旬 2019/09/13 - 2019/09/15

替換過程如下

* 由於model predict 一次產出7天,所以過年期間會切割為02/02-02/08以及02/09-02/10 兩個時段,將02/02作為index,input為D-1toD-7相關feature得到02/02-02/08七天prediction value,直接替換掉原先的prediction value
* 而02/09-02/10兩天期間則是將02/09作為index, input為D-1toD-7相關feature得到02/09-02/15七天預測值,只取02/09 and 02/10 兩天的predition value,替換掉原先的預測值
* 剩下的以此類推

10次平均結果為 (原先尚未替代)1.86>(學姊的特殊日替代調)1.82>(自己的model替代掉)1.78

已將結果放入 report報告內 以及update excel中

另外將根據這次結果嘗試放入batch normalization layer

2020.03.01 Zhi-Hong

已將Model2 /model7重跑

將peakload改為prediction value

將特殊日替換成學姊的值

已經更正report.docx

另外model 8 / model9的結果在report下面 對應之前老師所說沒看到的部份

2020.02.25 Zhi-Hong

以新增另外2個model在紀錄中

結論而言 放dropout不會以較好 在這例子當中

2020.02.21 Zhi-Hong

已經將7種結果詳細的紀錄在report/dataset/program/result

簡單來說

單看平均結果來說

model2(mape 1.99)> model7(mape 2.00)>model1(mape 2.15)>model6 (mape 4.35)>model4(mape 4.45)>model3 (mape 10.33) > model5 (mape 11.12)

有用hyperas的結構一定優於沒用hyperas的

正規化minmaxscaler(ALL dataset) > feature / max(training feature) > minmaxscaler(training of dataset)

有使用到未來資訊的>沒有使用未來資訊(D to D+N of feature)

2020.02.20 Zhi-Hong

給我一點時間 從昨日半夜開始寫model 現在要跑model

我實驗會跑以下假設 都會記錄

1. 最一開始仿照學姊的dataset與model架構(layer層數與unit…等)跑NN
   * 相關資訊:dataset feature and normalization from 學姊
     + D-1 to D-7 measure
     + D to D+6 of temp , Peakload , day of year(1-365) , week , holiday
     + Normalization : Feature minmaxscaler(all dataset)←學姊的正規化
2. 使用hyperas跑上面的東西 改架構而已
3. 用過去資料去預測未來7天 ,架構採用第一個
   * 相關資訊:dataset feature and normalization
     + D-1 to D-7 of measure ,temp, day of year(1-365),week,holiday, Peakload
     + Normalization: Feature minmaxscaler(all dataset)
4. 拿第3組搭配hyperas
5. 拿第3組,架構採用第一個,正規化採用Feature minmaxscaler(train dataset)
6. 拿第5組搭配hyperas 改架構而已
7. 一樣跑hyperas 但有點不一樣
   * 相關資訊:dataset feature and normalization
     + D-1 to D-7 of measure ,temp, day of year(冬夏至),week,holiday, Peakload
     + D to D+6 of day of year(冬夏至) , week , holiday , Peakload , temp
     + Normalization: Feature / max(training dataset of feature)

2020.02.19 Hsu

做一個實驗：使用和湘婷的最佳績效的模型相同的輸入參數設定，但是你使用ANN模型。這樣的實驗可以比較在相同的輸入、相同的條件之下，ANN及LSTM哪個模型的預測能力較好。

2020.12.19 Zhi-Hong 07.40PM

好的

另外 這是剛剛將day of year 轉成1-365的情況

Mape更差

|  |
| --- |
|  |
|  |
|  |

2020.02.19

* Loss及metrics的設定有試試別的。Accuracy通常用在分類的問題。Loss可以回頭試試看MSE or MAE，先前是否是因沒有正規化的問題？
* 湘婷在model.compile中沒有加metrics的選項，也可以跑。要試試看嗎？

2020.02.18 Hsu

如果相同參數設定之下，每次結果都不相同的話，就必須跑多次，然後取平均值。

🡪 以將實驗結果放入report

2020.02.18 Zhi-Hong

我剛試了 不過不是1~365 而是冬夏至轉換 冬至為1 夏至最大 整體比較好點 但沒突破 我放在skype 我要再多跑幾次確認確認

2020.02.18 Hsu

你的Day of year是1~1096？湘婷的Day of year是用1~365，使用這樣的結果比沒有用Day of year的結果還好。你要試試看嗎？

2020.02.18 Hsu

請將最優的結果以紅色字體顯目呈現。

2020.02.18 Hsu

這幾天的實驗很有參考價值。請將這幾天的實驗結果彙整記錄在下表。並設定”標題”，以利在功能視窗內直接點選，方便日後瀏覽。

預測未來七天工業用電實驗結果 - ANN

|  |  |  |  |
| --- | --- | --- | --- |
| **實驗設計(參數設定)** | **Output time** | **MAPE** | **RMSE** |
| 備註:  use max(training data) normalization  no have day of year  Model structure: | 2-1  3-2  1-7 | 5.84%  6.24%  5.74% | 23.81  25.37  23.56 |
| 備註:  use minmax(training data) normalization  no have day of year  Model structure: | 2-1  3-2  1-7 | 5.60%  6.19%  5.98% | 22.46  24.66  24.44 |
| 備註:  use minmax(All dataset) normalization  no have day of year  Model structure: | 2-1  3-2  1-7 | 5.60%  5.91%  5.49% | 22.47  23.81  22.64 |
| 備註:  No use normalization  have day of year (1-1096)  Model structure: | 2-1  3-2  1-7 | 6.24  6.22  6.64 | 25.37  25.59  27.19 |

2020.02.18 Zhi-Hong

已新增使用feature / max(training data feature) 正規化(不含day of year)的report於指定資料夾,含excel / coding / model / result / report

結論:結論而言跟之前一樣,MAPE沒比以前好,約差0.25%左右

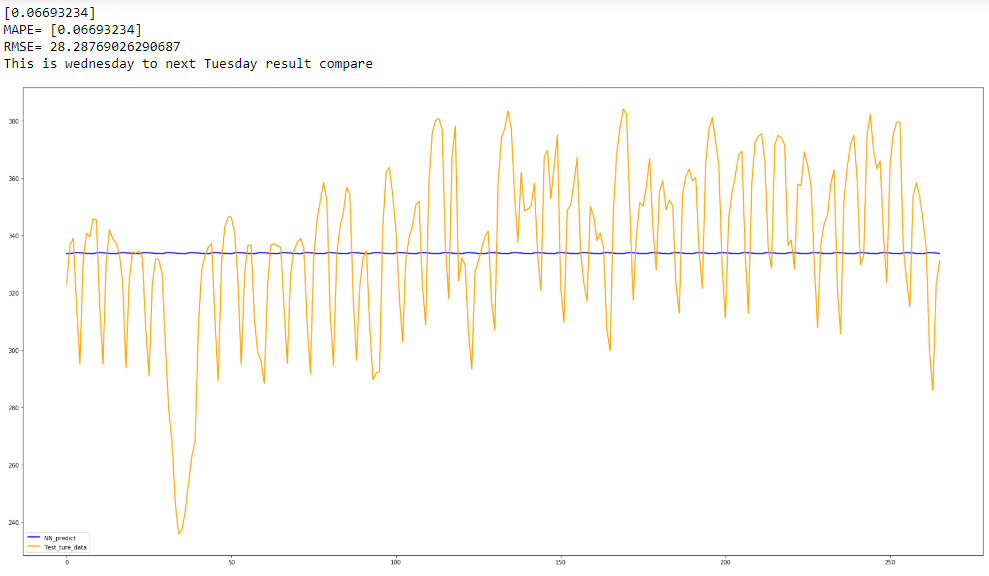
2020.02.17 Zhi-Hong

我原本在skype上提到說要跑跑看sigmoid

不跑還好 跑了超lag且吃超多資源 掛機一整天當了四五次

而且最後輸出變成一直線 直接梯度消失

結論:hyperas本身夠吃資源 sigmoid下去直接超lag



2020.02.16 Hsu

沒有歸零應該是不合理。不過，轉換成1~365是否是個好轉換，也不確定。因為年底的那幾天，和隔年年初的那幾天，天氣及景氣應該都是差不多。但是Day of year的值卻差很大。此問題，老師在Skype上面有提到。

2020.02.16 Hsu

Day of year是使用Excel裡面的date\_change嗎？該值一直增加，元旦沒有歸零。

🡪好像是直接連過去耶 我忘記那時候怎麼做的 不過那時候我沒有用到excel 所以要讓他每年歸0喔

2020.02.16 Zhi-Hong

* 目前的輸入變數有

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X of Day of year | X of week | X+1 of week | X+2 of week | X+3 of week |
| X+4 of week | X+5 of week | X+6 of week | X-1 of week | X-2 of week |
| X-3 of week | X-4 of week | X-5 of week | X-6 of week | X-7 of week |
| X of holiday | X+1 of holiday | X+2 of holiday | X+3 of holiday | X+4 of holiday |
| X+5 of holiday | X+6 of holiday | X-1 of holiday | X-2 of holiday | X-3 of holiday |
| X-4 of holiday | X-5 of holiday | X-6 of holiday | X-7 of holiday | X-1 of measure |
| X-2 of measure | X-3 of measure | X-4 of measure | X-5 of measure | X-6 of measure |
| X-7 of measure |  |  |  |  |

這樣應該算是有區分哪天是周一哪天是周日吧 總共36個變數

2020.02.16 Hsu

* 輸入時，有區分哪一天是周一，哪一天是周日嗎？
* 目前輸入總共有多少個變數(number of variables or number of input neurons)？

2020.02.16 Zhi-Hong

沒有正規劃

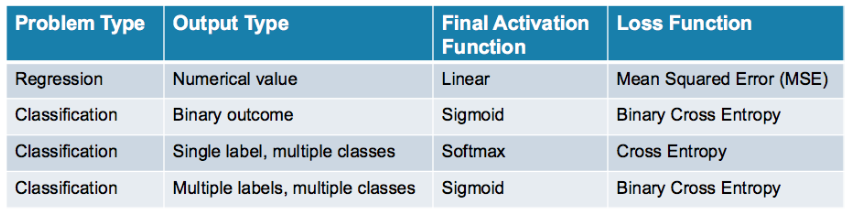
給我點時間

然後我會換成linear activation try try看

包含正規劃

2020.02.16 Hsu

* 我還是覺得很怪異。以周一到周日為例，7個用電輸入對7個輸出，第七個的用電值都很小，應該不難學到此特徵。
* 輸入資料有無正規化？使用何種正規化？根據教科書建議，數值預測的最後輸出層直接使用linear activation，不知結果和ReLU有差異否？要不要試試看？

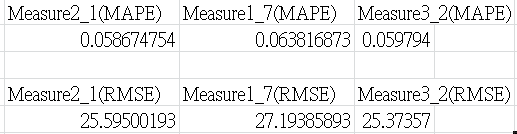


<https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8>

2020.02.16 Zhi-Hong

* 已經將model / result / code / excel / 放入指定資料夾中
* 架構的部分我覺得沒什麼問題,黃色為真實值,藍色為預測值,類神經網路的model架構圖我寫在下方,目前就只是這樣子,我覺得應該是因為說,畢竟是使用傳統ANN,其沒有所謂的時間依賴性,輸出沒辦法根據上一個輸出做關聯,且一次七個相對來說比一次預測一個還要來的不準確,才導致目前輸出結果呈現而言會是那樣

|  |  |
| --- | --- |
|  | Input |
| Hidden layer |
| Hidden layer |
| Dropout layer |
| Hidden layer |
| Dropout layer |
| Hidden layer |
| Dropout layer |
| Output layer |

* 另外Excel已經算好了,其落差有點差距,我總結一下原因
* 

|  |
| --- |
| Python計算是用浮點數計算 對應到下方的excel試算表 我拿其中一組做舉例 |
|  |

其中 abs((預測值-真實值)/真實值) 兩個計算呈現結果不一樣 以至於最終結果會有誤差

但跟python結果的順序是一致的 值不同而已

2020.02.16 Hsu

* 好的，沒問題。
* 預測結果的圖形，老師一直覺得很奇怪，為何藍色與黃色的高低起伏差那麼多？有辦法確認這是類神經網路模型的問題？還是哪裡出問題嗎？目前的所用的類神經網路的結構？
* 另外，是否可以用Excel人工驗算一下測試資料的MAPE及RMSE值？
* 請將你的程式及結果放在/108 老師與治紘的分享夾內。

2020.02.16 Zhi-Hong

我另創一個docx 屬於演算法的部分 已經想破頭並修改一下 老師請您過目一下

這是之前的紀錄

|  |  |
| --- | --- |
| 周一至周日 | 6.64 |
| 週二至下周一 | 6.22 |
| 週三至下周二 | 6.24 |

2020.02.14 Hsu

如果是 周一到周日 呢？

2020.02.14 Zhi-Hong 06:00:

我拿原本273(就原來test資料 2019-01-01 ~ 2019-09-30)以及現在的(2019-01-02~2019-09-24)預測做比對

在ANN的情況下是 [沒差多少] 用同一組訓練的model去做predict

以下比對三次結果

|  |
| --- |
| 2019-01-02~2019-09-24 6.27% (周三至下周二) |
| 2019-01-01 ~ 2019-09-30 6.33%(原周二至下周一) |

|  |
| --- |
| 2019-01-02~2019-09-24 6.27% (周三至下周二) |
| 2019-01-01 ~ 2019-09-30 6.33%(原周二至下周一) |

|  |
| --- |
| 2019-01-02~2019-09-24 6.27% (周三至下周二) |
| 2019-01-01 ~ 2019-09-30 6.33%(原周二至下周一) |

2020.02.14 Zhi-Hong 04:00:

我是周三到下周二

然後是預測七天 由X開始算 X X+1 X+2… X+6 總共7天

另外 使用hyperas跑的一定是最低的, 不然就是跟後者沒相差多少

不用額外fit 至少我現在跑出來是這樣

|  |
| --- |
| Hyperas structure |
|  |
|  |
|  |

|  |
| --- |
| Hyperas structure |
|  |
|  |
|  |

2020.02.14 Hsu

星期的切割，有試試看 ”周三到下周二” 之切割方式嗎？

2020.02.14 Zhi-Hong

我將feature擴充了一下

新增前7天每一天的week holiday狀態以及未來6天(7天？)week holiday狀態

目前來說再跑model相對比較穩定

績效也比較好一咪咪

上半段使用hyperas

下半段則是繼續fit得出結果

下半段沒比較好 會再多測試

|  |  |
| --- | --- |
|  | 6.28 |
|  | 6.66 |

2020.02.12 Hsu

比先前的文字敘述較精簡了，較容易閱讀。但下列演算法的表達，還有精進的空間，再努力一下。

2020.02.11 Zhi-Hong

Hyperas演算法 以過去7天預測未來7天的工業用電

Feature:

Day of the year

Holiday

One day ago to seven days ago measure

Week

Algorithm : To forecast next n days measures everyday with NN by hyperas

Import keras , hyperas, hyperopt

Data function

1. Create a data function
   1. Import data from excel or database
   2. Let data divide train\_data,train\_label,test\_data,test\_label
   3. Declare a new array
   4. Repeat len(data):
2. Let data(dataframe) append in an array[i:i+n]
   1. Train\_label = array
   2. Return train\_data,train\_label,test\_data,test\_label

Model setting function

1. Create a model function(train\_data,train\_label,test\_data,test\_label)
   1. Create a new model that it is a type of Sequential.
   2. Add input layer , some hidden layers , some dropout layers and output layers.
   3. Setting every layers activation function except dropout layers.
   4. Setting range to every unit parameters and dropout parameters from every hidden layers and dropout layers by hyperas
   5. Declare model.compile and setting parameters include optimizers, loss function .
   6. Declare model.fit and setting parameters include train\_data, train\_label and validation\_split,epochs
   7. Setting batchsize range into [model.fit] by hyperas
   8. Setting callback parameters like early\_stopping into [model.fit].
   9. Return {'loss': min(model.history.history['val\_loss']) , 'status': STATUS\_OK, 'model': model}

Main run

1. Declare train\_data,train\_label,test\_data,test\_label from data function.
2. Declare best\_run, best\_model from optim.minimize and setting optim.minimize parameters include model = model function , data = data function , algo=tpe.suggest , max\_evals = how many time do you want to run , trials=Trials() and notebook\_name = your jupyter notebook title name.
3. Running step 1 and 2 and will get best run and best\_model
4. Predict next n days measure everyday with test data by best model
5. Compute and print MAPE and RMSE by prediction and test\_label

Increase model

1. Create increase model function(best\_model):
   1. Declare train\_data,train\_label,test\_data,test\_label from data function
   2. Declare “change\_model” = Sequential()
   3. Declare “min\_val\_loss” = 100
   4. Repeat n times:
      1. Declare best\_model.fit and setting parameters include train\_data, train\_label validation\_split,epochs, batch\_size (from main run hyperas output), callback like early\_stopping.
      2. If (min\_val\_loss >min(model.history.history[‘val\_loss’])):
         1. Change\_model = model
         2. Min\_val\_loss = min(model.history.history[‘val\_loss’])
   5. Predict next n days measure everyday with test data by change\_model
   6. Compute and print MAPE and RMSE by predictions and test\_label
   7. Return change\_model and MAPE
2. Run Step 1 with best\_model which is from Main run.