YouBike Prediction Report

* 資料處理

根據報告中的要求處理遺失資料，例如11月25日11點25分的資料遺失，就去檢查26分有沒有資料，沒有的話就再往下找27分，以此類推；如果是23點59分資料遺失，就要從11月26日的0點0分開始檢查。

下面三個模型使用的訓練資料時間範圍均為助教提供的資料中10月2號一直到12月3號；由於10月11號跟10月15號的晚上資料遺失太多，所以三個模型訓練的時候並未使用這兩天的資料。驗證資料為12月11號到12月17號，測試資料為12月18號到12月24號。

* 模型介紹

1. 平均值模型 Mean Value Model(MV model)

這個模型的訓練方式就跟其名稱一樣。對於各個站點，我將訓練資料以星期分類，然後對每個時刻的值都做平均值，得到1440個代表個時刻的平均值。舉例來說，對於捷運公館站3號出口，假如分類好後星期一的資料有10天，因此一天中的1440分鐘，每分鐘就有10筆數據，每分鐘各自對自己的10筆數據取平均，得到的值就是我們之後拿來預測這個站點，在星期一的每分鐘的值。但是為了考慮到這次的evaluation metric是帶有權重的，所以我是取加權平均，權重就是根據evaluation metric中的：

部分作為加權。

1. 梯度類型模型 Gradient Descent Type Model(GD model)

這個模型是以前20分鐘(包含當前分鐘)的站點資料，來預測下一週跟當前相同分鐘的腳踏車數量，例如要預測公館站3號出口在11月26號11點26分的腳踏車數量，輸入資料就會是11月19號11點07分到11點26分的資料。模型更新的方式是使用題目給的evaluation metric作為loss function，使用scipy.optimize.minimize，method根據ChatGPT的建議以及個人的測試後使用L-BFGS-B，以用於帶絕對值的loss function，參數的部分均按照scipy內建預設，這裡只有選擇使用L-BFGS-B作為method而已。預測的時候，我讓0點0分到0點18分的預測數值跟0點19分的預測數值一樣。

1. 1-D CNN神經網路模型 1-D CNN model(CNN model)

這個模型使用了經常用於時間序列預測的1維CNN神經網路來進行預測。這裡以前1440分鐘(包含當前分鐘)的站點資料，來預測下一週跟當前相同分鐘的腳踏車數量，例如要預測公館站3號出口在11月26號11點26分的腳踏車數量，輸入資料就會是11月18號11點27分到11月19號11點26分的資料。

建構模型採用了torch的模組，定義自己的卷積神經網路，下面是訓練時有關的參數以及配置：

* + - 架構使用了兩層一維卷積，一層平均池化層和一層機率為0.5的dropout層。
    - optimizer使用Adam，1.126E-4的learning rate，和係數1.126E-6的L2 regularizer；同時optimizer搭配learning rate scheduler，採用StepLR，每經過4個epoch就將learning rate乘以0.1。
    - batch size為64，並且訓練10個epoch。
    - loss function一樣採用error metric。

下面是forward時的傳遞路徑：

|  |  |  |
| --- | --- | --- |
| layer | parameters | input |
| Conv1d\_1(nn.Conv1d) | in\_channels=1, out\_channels=32, kernel\_size=20,stride=1 | 1x1440 |
| ReLU(nn.ReLU) | | |
| Conv1d\_2(nn.Conv1d) | in\_channels=32, out\_channels=64,kernel\_size=20,stride=1 | 32x1421 |
| ReLU(nn.ReLU) | | |
| AvgP\_1(nn.AvgPool1d) | kernel\_size=20, stride=20 | 64x1402 |
| flat(nn.Flatten) | start\_dim=1,end\_dim=-1 | 64x70 |
| Drop1(nn.Dropout) | p=0.5 | 64x70 |
| dense(nn.Linear) | in\_features=4480,out\_features=1 | 4480 |
| ReLU(nn.ReLU) | | |

* 預測成績

預測時均為預測每天的0點0分到23點59分，沒有以20分鐘為間隔。

|  |  |  |  |
| --- | --- | --- | --- |
|  | MV model | GD model | CNN model |
| 平均值 | 0.30704 | 0.35262 | 0.33152 |
| 標準差 | 0.10809 | 0.13656 | 0.12976 |
| 平均值 | 0.42588 | 0.51080 | 0.46772 |
| 標準差 | 0.22705 | 0.28022 | 0.21245 |

平均值跟標準差為112個站，在七天預測的平均值中的平均值跟標準差。

* 模型比較

從跟的平均值來比較的話，GD model皆不及MV model跟CNN model；而MV model跟CNN model則是MV model均低於CNN model；

標準差的部分，GD model一樣有著較大的值；MV model跟CNN model則是MV model測試資料些微高於CNN model，驗證資料則低於CNN model。

綜合兩點來看，GD model的表現均遜於另外兩個model。MV model跟CNN model則是MV model表現較好，並且從標準差的部分可以看出兩者預測的變異性都較GD model低。

接著從實作層面以及訓練時間的部分來看，本次的平台為AMD 5900X，32G DDR4記憶體，NVDIA RTX3080以及PCIE gen4 SSD。

首先CNN model訓練時間最久，並且只有他有使用GPU訓練，需要49分鐘36.1秒，還有著較高的建構複雜度，以及要求更多的預先知識；接著是GD model，需要29分鐘17.8秒，建構上的複雜度不低，也需要有一定的知識；最後是MV model，訓練時間最短，只需要1分鐘42.6秒，建構上的複雜度最低，而且知識門檻也非常的低，並且因為其根本上就是在算平均值，具有非常高度的可解釋性。

容量大小方面，CNN model儲存所需的大小最大，需要大約20MB左右的空間，MV model則是其次，大約8MB，GD model最少，只要2MB。此處都是利用python的pickle模組做資料儲存。

預測所需資源的部分，根據運算時間為基準的話，MV最少，只需要5秒左右，GD model其次，大約15秒左右，CNN模型最長，需要近1分鐘。

至於日後更新方面，MV model具有較大的隨時更新彈性，而CNN model需要對網路的內部層做修改，需要一點技術，GD model則是可能要考慮改採用帶有SGD單一資料用來更新的性質的method。

* 推薦選擇

經過上述的評估，我這裡鄭重的推薦MV model做為日後預測的模型，訓練快，容易理解跟解釋，預測效果佳且快速，易於更新，甚至在低開發程度的地區，如果沒有電子設備的環境下也可以直接人工實作。以上的種種優點，使得我認為MV model是最佳的選擇。

如果一定要說缺點的話，那就是一般人如果知道我們採用了MV model可能會覺得他也可以來做資料科學或機器學習…。

* 給公司的建議

在整理資料的過程中，其實不難看出貴公司會在凌晨進行車的調動，然而這成了我們團隊這次的些許困擾：

首先，每次調動的時間點沒有固定，這可能難以使我們的模型學會調動的時機，降低預測的準確性；再來每次車子調動的量也沒有固定的範圍，可能有時站點調整後的車輛很少，有時卻很多，這對於我們這次的evaluation metric有著極大的影響，因為我們的模型有可能是因為這些突然與往常不同的調動，導致預測失準，而恰好該時候的量是很少或是很滿的，使得錯誤量偏高。

或者我們團隊可以嘗試使用更多或更深入的資料處理，讓模型學會公司的相關調動行為，近一步可以推測出調動的時間點等等資訊，以應對這個問題；如有這個需求，未來我們會嘗試開發相對應的模型。

* Reference

GD model：

1. scipy.optimize.minimize 的使用方法

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>

2. GPT的建議

<https://chat.openai.com/share/5869d231-40a1-407f-a464-12fb1e466feb>

CNN model

1. 建構時是詳細再詳細的閱讀torch官網的document。

2. 認知到CNN可以處理時間序列問題：

<https://bc165870081.medium.com/%E6%99%82%E9%96%93%E5%BA%8F%E5%88%97%E7%9A%84ai%E4%BB%8B%E7%B4%B9-ff250cfc2ff9>

<https://blog.csdn.net/weixin_38346042/article/details/121742025>

<https://www.twblogs.net/a/5e905b7dbd9eee342090101e>

<https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/>

雖然以上都是tensorflow的，但後來有查找pytorch官網和一些網站，了解如何用pytorch建構1D-CNN。

* 團隊分工

此為一人團隊，如有不足之處，敬請見諒。