Machine Learning: Report of Final Project

Due on June 25, 2019 at 12:00

陳熙 R07922151[1], 蔣藝荃 R07944046[2], 張蓉蓉 R07922152[1]
[1] 國立臺灣大學資訊工程研究所
[2] 國立臺灣大學資訊網路與多媒體研究所
{r07922151,r07944046,r07922152}@ntu.edu.tw

一、前言

在這個 project 中，我們研究了影響分形布朗運動軌跡結果的三個參數。這個問題是一個回歸問題，其目標是估計 alpha、mesh size 和 penetration rate 這三個參數，訓練資料是由軌跡經過常用處理方法得到的 47500 筆 10000 維特徵矩陣。測試資料為 2500 筆。而對於結果的評估，track 1 采用 Weighted Mean Absolute Error(WMAE) 和 track 2 采用 Normalized Absolute Error(NAE) 這兩個標準。

針對這個問題，我們采用了幾種機器學習常用的演算法 XGBoost [1]、LightGBM [2]、Random Forest [3] 和 Neural Networks [4] 等，其中在 Neural Networks 的部分我們還用到了十分流行的 Convolutional Neural Networks(CNN) [4] 的方法，例如 Inception-v4 [5]。在 track 1 的 public scoreboard 上分數 43.96，排名 11；private scoreboard 上分數 43.90，排名 7。在 track 2 的 public scoreboard 上分數 0.40，排名 3；private scoreboard 上分數 0.41，排名 4。

二、方法與討論

2.1 Random Forest

我們利用 sklearn [6] 中的 RandomForestRegressor 這個函數來實現 Random Forest。我們固定 criterion 為平均絕對誤差 (MAE)，迭代次數 (n_estimators) 為 10 ，以 10 為單位逐步增加最大深度來進行訓練。從表 1可看出最大深度對模型的影響：隨著深度的增加，兩個 track 的 training error 都有很明顯的下降，說明模型可以更精細地 fit 訓練資料，但是當深度大於 20 後，在 track 1 上的 public score 幾乎沒有改變，可見將最大深度設定為 20 已足夠。sklearn 所提供的 MultiOutputRegressor 可搭配 random forest 函數，為三個 output 分別訓練獨立的模型，我們利用了這個方法進行了以下實驗。

從表 2可見，提高最大深度可以讓兩個 track 的 public score 有明顯的下降。與表 2 相比，三個獨立模型可以有效降低 track 1 的 public score，說明三個 output 沒有明顯的關聯性；但是這種方式卻提高了 track 2 的 public score，我們認為這很有可能與 track 2 所使用的 evaluation (NAE) 有關，某個獨立模型在平均數值較小的 output 上出現了 overfitting 的現象，也許 overfitting 所產生的誤差在 WMAE 的衡量上並不明顯，但是在 normalize 之後，這些誤差會被放大。

表 1: 使用不同 max_depth 參數下的 Random Forest 性能

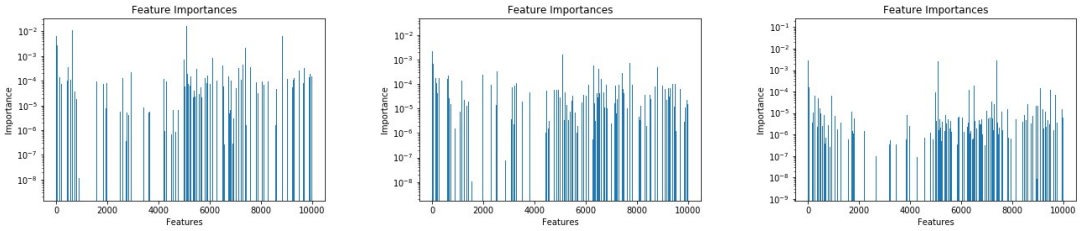
n_estimator	max_depth	track 1		track 2	
		training error	public score	training error	public score
10	10	95.9842	100.698796	2.2912	2.240516
10	20	52.3798	91.922593	1.2467	2.339864
10	30	38.0436	90.408317	0.8234	2.046537

表 2: 使用 sklearn.MultiOutputRegressor 在不同 max_depth 參數下的 Random Forest 性能

n_estimator	max_depth	track 1		track 2	
		training error	public score	training error	public score
10	10	90.1712	96.912555	2.2593	3.322853
10	20	54.3144	85.853665	1.1492	2.764412

2.2 使用 Random Forest 的特徵選擇

sklearn 的 random forest 通過統計每個 feature 能降低「不純度」的多寡來決定其重要度。在一個決策樹中，越靠近樹根的節點所用到的 feature 越重要。我們從多次實驗中為三個 output 分別整理出特徵重要性。



(a) alpha (b) mesh size (c) penetration rate

圖 1: Random Forest 得到的對 alpha、mesh size 和 penetration rate 的特徵重要性

從圖 1可以看出，不同 output 的特徵重要度有一些差異，因此我們在三組數據中對每個特徵做平均，得到每個特徵的平均重要度，然後根據平均重要程度對所有個特徵做排序，得到「累積重要度曲線圖」。

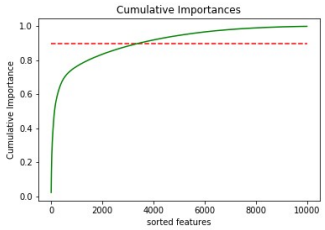


圖 2: Random Forest 得到的累積重要度曲線

圖 2中綠色曲線為「累積重要度曲線」，紅色虛線為 90% 重要度的 threshold。我們選擇保留 90% 的重要度，一共篩選出 3752 個重要特徵。這 3752 個重要特徵將在後續實驗中發揮 dimension reduction 的作用。

Random forest 有良好的可解釋性，因此可以用來統計不同特徵的重要程度，挑選出的重要特徵在我們其他模型上發揮了巨大作用，可見利用 random forest 挑選特徵是比較合理的。訓練時間較長 (至少一小時)；提高 n_estimator 或最大深度都會明顯增加記憶體的用量，程式會因為 Out of Memory(OOM) 錯誤而中斷。

2.3 Multiple Layer Perceptron(MLP)

我們利用 keras [7] 建構了三個三層 MLP [8] (neurons: 512, 128, 3)，讓每個 MLP 獨立預測一個 output。我們發現只要小心地調整一些重要的參數，就可以在 Track 2 中得到 0.48 的 public score。以下為重要的調參細節：

- Dimension Reduction: 只用上文所提到的 3752 個重要特徵作為 input，降低了第一層網路的參數數目。

- Input Normalization: 利用 sklearn 的 StandardScaler 函數對輸入值做縮放。
- Loss Function: 在 track 1 上，我們使用 MAE，並為每個 output 設定權重 (300, 1, 200); 在 track 2 上，我們使用 MAPE。
- Validation data: 固定分割出 20% 的資料作為 validation data。
- Saving the Best Model: 保存能讓 validation loss 最低的模型。
- Batch Size: 經過實驗我們發現過小的 batch 會讓 validation loss 無法持續下降，將 batch size 設定為 1024 或 512 最合適。

經過以上細節調整，MLP 模型在 track 1 上的 public score 可以從 141 上升到 55，在 track 2 上 public score 可以從 0.99 上升到 0.48。此外，MLP 的 validation loss 幾乎與 public score 相等，可以通過 validation loss 直接估計 public score，這說明 testing data 與 trainig data 的資料分佈應該相差不大。

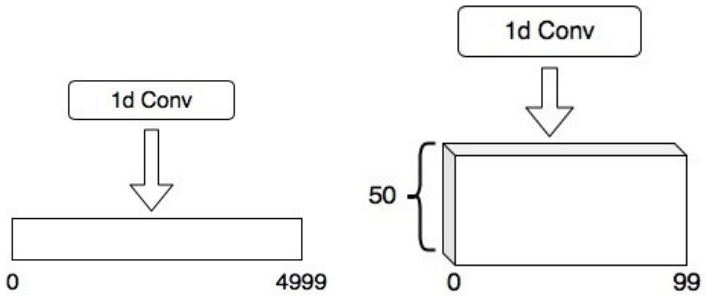
表 3: 使用 MLP 搭配 tricks 的性能提升

	public score of track 1	public score of track 2
Before Tuning	141.348746	0.993674
After Tuning	55.339174	0.480024

MLP 的優勢在於可用 GPU 運算，能在一分鐘之內完成訓練，記憶體佔用量很小，能得到很好的 public score。但是模型複雜度不夠高，缺乏可擴展性，可解釋性較低。

2.4 Naïve CNN

我們根據原始資料的物理意義，將 10000 維度分成前 5000 維 (input 1) 與後 5000 維 (input 2)，並且以連續的 100 個維度為一組，將後 5000 維分割成 50 個 100 維度的向量，再把 50 個維度疊在一起，形成一個深度為 50 的 1×100 特徵圖。這種堆疊的方式可以提升卷積層的網路深度，從而提高 CNN 的模型複雜度。由於兩個 input 代表不同的物理意義，我們用兩組不同的 kernels 分別對這兩塊輸入做卷積，卷積的結果輸入 Dense layer 做 regression。如圖 3所示。

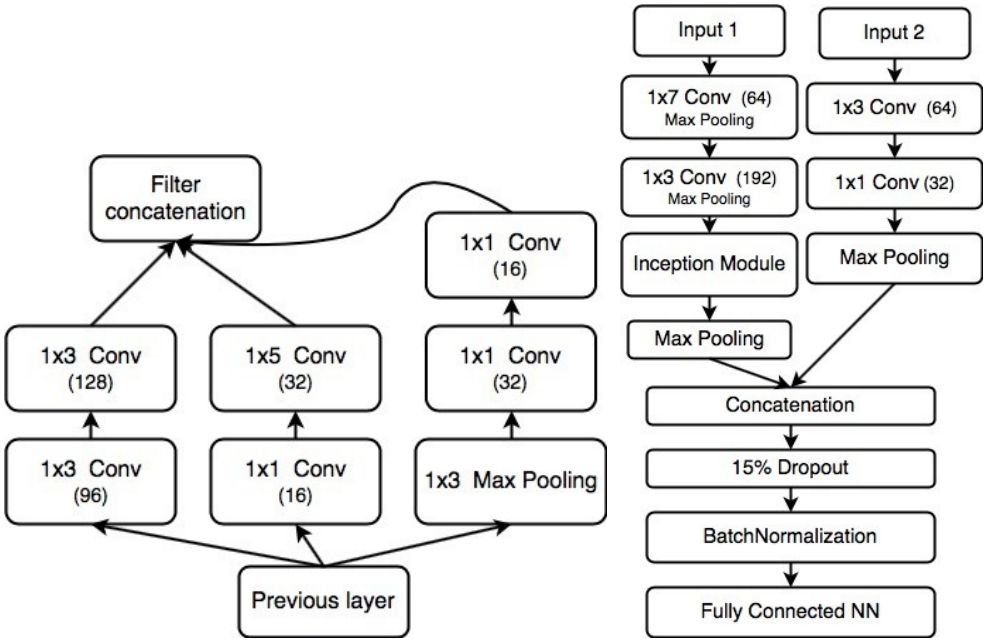


(a) input 1 (b) input 2

圖 3: Naïve CNN 中對輸入資料的處理方式

2.5 Inception module

我們參考 GoogLeNet [9] 模型，搭建了一個 Inception 模組，如圖 4a所示。Inception 模組是一個 sparse 網路結構，卻可以得到 dense data，他內部卷積層大致平行分佈，用大小不同的 kernel 同時對輸入做處理，可以獲取輸入層的不同大小的特徵，又能達到高效的運算。此模組會將前一層的輸出分成三個分支，分別用 1×3 Conv，1×5 Conv 和 MaxPooling 做處理，最後再用 filter concatenation 得到一個非常「厚」的 feature map。由於前一層的輸出通常很「厚」，



(a) Inception Module (b) Inception CNN

圖 4: Inception Module 和 Inception CNN 的模型結構

Inception 模組會在 1×5 Conv 前加上深度為 16 的 1×1 Conv 對前一層的輸出先做降維，在不改變 input 大小的前提下減少 1×5 Conv 的參數。

2.6 Inception CNN

我們將 Inception 模組整合到網路中 (如圖 4b所示)。由於 input 2 長度只有 100 維，不需要用多個平行的 kernels 來處理，因此我們只用 Inception 模組處理 input 1。模型細節與解釋如下：

- 1×7 Conv (64) and 1×3 Conv (192): input 1(size: 1×5000) 的長度太長，厚度太「薄」，不適合之間丟給 Inception，因此我們先用這兩個卷積層和池化層對 input 1 做處理，從而可以得到長度較小，厚度較「厚」的特徵圖。
- 1×1 Conv (32): 來源於 Network in Network (NIN) 的概念，深度為 32 的 1×1 卷積層可以對上一層深度為 64 的 1×3 卷積層做降維，同時可以對前一個卷積層的輸出做非線性轉換，具有提升模型複雜度的作用。
- Dropout: 在沒加入 dropout 前，training loss 會遠低於 validation loss，為儘量減輕 overfitting 的現象，我們謹慎地加入較小的 dropout，因為較大的 dropout 會影響後面的 regression。在 track 2 上，我們使用 MAPE。
- Batch Normalization(BN): 兩個不同的 input 分別經過不同結構的卷積層，所產生的 Internal Features 也許在數值範圍上有較大差異，因此我們在 concatenation 層後加入 BN。

表 4: Naïve CNN 和 Inception CNN 的性能比較

	public score of track 1	public score of track 2
Naïve CNN	52.034407	0.874545
Inception CNN	46.275181	0.406665

Inception CNN 可在 GPU 上運算，能在 10 分鐘之內完成訓練，有很高的模型複雜度，可以做到高效的特徵轉換，可擴張性很高，能得到較好的 public score。但是相比於 MLP，需要更多的記憶體，可解釋性較低。

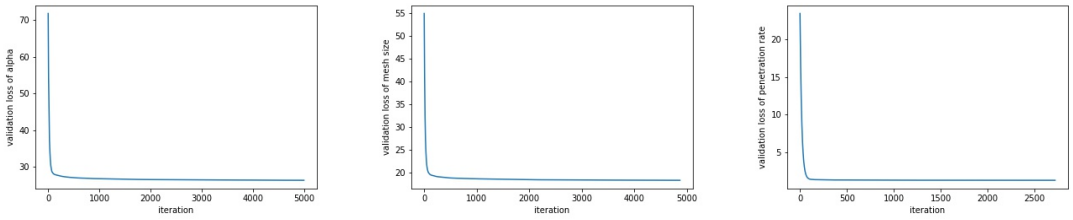
2.7 LightGBM

LightGBM 是一種使用了 gradient boosting 的 tree based 機器學習演算法。其優勢在於訓練速度很快、效率很高、使用的記憶體空間較少、準確率較高以及可以解決大規模資料的問題。

我們抽取資料中 Random Forest 預測得到的 feature 重要性排名前 3000 維的 feature。對於 track 1，我們首先將資料的預測結果（Y）乘上相對應的 weight（alpha 是 300，penetration rate 是 200），然後將資料集按照 9:1 的比例，隨機劃分為 training set 和 validation set。劃分的時候我們要注意讓 training set 和 validation set 的分佈保持一致。

對於 LightGBM，我們設置的 objective 為 L1 的 regression，leaves 數目為 200，learning rate 為 0.05，feature fraction 為 0.9，bagging fraction 為 0.9，bagging 的頻率為每 5 輪 interation。訓練參數為 5000 輪，early stopping 的輪數為 100。之後我們發現 LightGBM 有可以設置 decay learning rate 的方法，但是由於時間問題，沒有進行過多的嘗試。

由於訓練的 iteration 數目設置較大，訓練一共持續了 36 個小時。我們觀察到 validation error 在前 300 個 iteration 內快速下降。我們估計由於 learning rate 一直保持不變，導致其下降速度緩慢。如果此時將 learning rate decay 可能會獲得更好的結果。最後我們得到在 alpha 上的 weighted validation error 為 26.2611，在 mesh size 上的 weighted validation error 為 18.2526，以及在 penetration rate 上的 weighted validation error 為 1.3030。提交後得到的 public score 為 46.0958，private score 為 47.5836。



(a) alpha (b) mesh size (c) penetration rate

圖 5: 使用 XGBoost 得到的 alpha、mesh size 和 penetration rate 的 validation error

由於 LightGBM 中我們設置了較高的 bagging 頻率，導致其不容易 overfitting，其結果十分穩定。我們使用這一方法得到了單一模型的最好成績。但是由於我們 iteration 數目設置較大，訓練速度不是很快，另外，參數數目較多，使用上需要考量一些前人的經驗。

三、總結與展望

對於這個 project，我們主要采用了以上方法，包括 Random Forest、MLP、Naïve CNN、Inception、XGBoost 和 LightGBM，這些方法既有基於決策樹的方法，也有基於類神經網絡深度學習的方法，可見現代機器學習工具的強大威力。另外，我們也嘗試了一些簡單的回歸模型，例如 Linear Regression 在 track 1 的 public score 上得到了 93.48、track 2 上 3.40 的成績。我們把方法 $\mathcal{A}_i$ 的 public score 用 s_i 表示，我們最好的 track 1 記錄是由 LightGBM 和幾種 Inception CNN 上按照權重

$$w = \frac{\exp(-s_i)}{\sum_j \exp(-s_j)}$$

加權得到的。經過 ensemble，我們的 public score 得到了從 46.10 到 44.00 的提升。而 track 2 最好的記錄是由 Inception CNN 得到的。在 LightGBM 部分，我們通過調整 learning rate 還可能得到更好的結果。

四、團隊分工

陳熙主要進行 tree-based 方法的實驗，包括 XGBoost 和 LightGBM 的方法，另外還進行一些簡單回歸模型的實驗，例如 Linear Regression [10] 和 Ridged Linear Regression [10]。蔣藝荃主要進行了 CNN 方法的實驗，而張蓉蓉主要進行 Random Forest 模型的實驗以及後期模型整合以及對實驗結果進行分析和思考，指導陳熙和蔣藝荃的實驗方向，同時也參與了調整模型參數的過程。

References

[1] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[2] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (USA), pp. 3149–3157, Curran Associates Inc., 2017.

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, p. 85–117, Jan 2015.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] F. Chollet *et al.*, "Keras." https://github.com/fchollet/keras, 2015.

[8] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," tech. rep., Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[10] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data: A Short Course*. AML-Book, March 2012.