機器學習技法 期末專案

Machine Learning Techniques Final Project

廣告點擊預測

Click through rate prediction of advertisement

組別: Deep Sleeping

組員:

R04943174 王嚴徵

B02902127 李旻倫

R05522625 王威翔

我們這組進行專案的方式為每個人分別去嘗試一種有興趣的模型，主要為 (1) XGBoost (加上小部分 Random Forest) (2) Logistic Regression (3) Multi-Layer Perceptron。截止日(6/19)當天 Judge 系統上 Track 1 表現最佳的模型為 Logistic Regression，分數為 **Public : 1.071908 / Private : 1.080638**；Track 2 為 XGBoost，分數為 **Public: 0.126734 / Private: 0.129061**。Track 1 錯誤率相當高的原因是我們一開始弄錯了輸出格式導致。後續一週進行調整過後，Track 1 以及 Track 2 表現最佳的模型皆為 Logistic Regression，Track 1 分數為 **Public : 0.145706 / Private : 0.146343**；Track 2 分數為 **Public : 0.138421 / Private : 0.140240**，皆有進步。以下會先就各模型的實作與參數調整過程進行敘述並針對模型表現結果進行討論分析，最後會將總結與討論。

Model I . Multi-Layer Perceptron

首先，抽取訓練資料 feature 的方式為 ***One-hot advertisement + 1-Day-per 8 hours time step :*** 將總共 623 筆廣告(包含只有出現極少次的廣告)以 one-hot 方式作成 623 個維度的廣告 feature；將 15 天的訓練集資料以 1 天為單位加上 8 個小時為一個區段的方式，將一天分成三個時段，形成了共 15*(24/8) = 45 個維度的時間 feature，後續使用 TensorFlow 套件進行訓練，原因為可以一次只訓練一部份資料集，訓練好的模型亦可以自由存取，加上視覺化工具 TensorBoard，整體使用起來相當方便。模型參數的選擇如下列所示

1. 初始值:因特徵矩陣為 1/0，故權重之初始值必須較小，且須避免全 1 或 0 以免造成部份權重梯度為零，故設定平均值為 0、標準差 0.01 的常態分佈。
2. 隱藏層數量:為了得到較佳的非線性擬合效果與收斂性，層數固定為兩層
3. 輸出層節點數量: 1。
4. Activation function: tf.nn.sigmoid
5. Cost function: tf.nn.l2_loss
6. Optimizer: tf.train.AdamOptimizer
7. Learning rate:用 validation 找

Learning rate	$10^{-0.5}$	10^{-1}	$10^{-1.5}$	10^{-2}
Average cost	0.024	0.038	0.06	0.082

利用此模型所獲得的訓練結果並不是很理想，無論參數如何去調整(每層神經元數目、Learning rate、換 Activation function…等)，Ein 與 Eval 皆沒有辦法壓到 20%以下，Eval 也和 Ein 有所差距。Judge 系統上的 Public score 與 Private score 只有 0.046613 與 0.047360 的分數。主要的原因可能是該模型相當複雜，我們對於參數選擇還不夠透徹理解所導致，因此後續決定開始使用表現穩定的 Gradient Boosting 模型以及較簡單直觀的 Logistic regression 模型。

Model II. XGBoost

本模型同樣使用 ***One-hot advertisement + 1-Day-per 8 hours time step*** 的特徵進行訓練(使用 XGBoost 以及 Scikit-Learn 套件)。由於訓練集資料相當大，因此在針對 XGBoost 模型進行做參數選擇時，僅使用第 0 天的數據作為訓練集資料，並將訓練集資料以隨機選取的方式抽取 30%的數據作為 validation 使用。初始參數設定如**表一**所示:選用 *binary:logistic* 是因為可以得到一個介於 0~1 的值，可直接用於 track 1；之後可以再選一個適當的 *threshold* 將獲得的機率值轉為 0/1 分類預測結果用於 track 2；將 *subsample* 設為 0.8 確保每棵樹使用的訓練集資料不太一樣，增加樹的多元性，也可以避免模型在訓練集資料上 overfitting 的太嚴重；*eta* 選得比較高(>0.3)的是擔心在工作站上訓練全部資料所花的時間會太長(後來有試過，設得太小不符合訓練效益)；評分基準選用 *root mean square error*，因為比較熟悉與常見，感覺適合大多數的模型。

表一、xgboost 模型基本參數選擇

booster	***objective***	***subsample***	***colsample_bytree***	***Eta***	***Eval_metric***	***threshold***
gbtree	binary:logistic	0.8	0.8	0.3	rmse	0.25

首先針對 *max_depth* (D)，範圍 3~10 (間距 1)，以及 *scale_pos_weight* (權重, W)，範圍 3~7 (間距 1)兩項參數進行選擇。前者目的為找出強度適合且不會 overfitting 的小樹，後者為平衡訓練集的資料的不均勻性(全部訓練集資料 class 1: ~ 3%, class 0: ~ 97%)，為的是避免在 0 資料集上 overfitting 的太嚴重。**表二**為參數的訓練資料的部分結果 Class 1 (%)表示模型在訓練集以及 validation 集上將資料分類為 1 的比例。

表二、各參數在第 0 天資料上的訓練結果，Class 1(%)表示模型分別在訓練集以及 Validation 集上預測為 1 的資料比例。

(D, W)	**Ein(%)**	**Eval(%)**	**Class 1 (%)**	**(D, W)**	**Ein(%)**	**Eval(%)**	**Class 1 (%)**
(3, 3)	3.75	3.76	0.716 , 0.726	**(6, 3)**	3.73	3.95	0.917 , 0.951
(3, 5)	7.20	7.15	4.771 , 4.726	**(6, 5)**	7.03	7.27	4.885 , 4.817
(3, 7)	20.0	19.9	19.15 , 19.16	**(6, 7)**	21.8	22.1	21.47 , 21.49
(4, 3)	3.74	3.75	0.765 , 0.729	**(7, 3)**	3.66	3.98	1.029 , 0.976
(4, 5)	7.23	7.27	4.903 , 4.870	**(7, 5)**	6.76	7.10	4.733 , 4.666
(4, 7)	22.4	22.4	21.83 , 21.81	**(7, 7)**	21.4	21.9	21.17 , 21.15
(5, 3)	3.75	3.84	0.880 , 0.813	**(8, 3)**	3.56	3.99	1.058 , 0.986
(5, 5)	7.14	7.21	4.886 , 4.788	**(8, 5)**	6.50	7.00	4.581 , 4.491
(5, 7)	19.8	19.9	19.12 , 19.13	**(8, 7)**	21.1	21.6	20.89 , 20.87

由**表二**可以發現 Eval 僅稍高於 Ein，主要原因可能來自兩者的資料分布仍然很相近或是 subsample/*colsample_bytree* 參數的設定使得 overfitting 程度不明顯。由**表二**可以發現權重設在 5 時會有比較好的表現，權重設 7 會讓模型將 1 預測地太多，導致 Ein/Eval 變得相當大，權重設為 3 雖然可以將 Ein/Eval 做得

很好，但是由 Class 1(%)中可以發現其預測 1 的結果比例很低(< 1%)，遠低於測試資料集上 Class 1 的比例，所以可以推測 Ein/Eval 低的原因來自於模型傾向預測 0，表示很有可能已經在 0 的資料上 overfitting 了。另外當權重設為 5 時，樹的最大深度變大時 Ein/Eval 會同時下降，但是兩者的差距越來越大，表示我們的模型越變越強但同時也有 overfitting 的疑慮，因此我們不考慮最大深度>8 的情形，由過去經驗覺得 D 設為 3 跟 4 有點不夠，加上這兩項參數的 Ein/Eval 稍微比 D = 5, 7 大了一些，所以我們最後還是選擇了 D 為 5 與 7，W 為 5 的模型，利用全部 15 天的訓練資料集進行訓練，訓練結果如**表三**所示。

表三、xgboost 模型訓練結果。*Y (5, 5) 為除去時間 feature，使用與(5, 5)相同參數的訓練結果。(Track1 的部分是拿 Track2 最好的 model X **去掉權重設置**，重 train 一遍得到的，因訓練時間太長，所以只做 X 模型。)

	Track 1		Track 2	
(D, W)	Public score	Private score	Public score	Private
X (5, 5)	0.147472	0.148000	0.126734	0.129061
***Y** (5, 5)			0.120648	0.122037
Z (5, 7)			0.117718	0.119580

由上圖的訓練結果可以發現，X 模型擁有最好的訓練結果(使用全資料集訓練完的 Ein 與 Eval 皆≈3%)，而 Z 模型(Ein 與 Eval 皆≈5%)則可能因為樹太深而導致了 overfitting 的結果，所以使其在 public/private score 上都略遜於 X，甚至比去除掉時間 feature 但使用與 X 一樣參數的 Y 模型差。而由 X 與 Y 的結果也知道時間特性對於模型訓練也有很重要的影響(事前有針對訓練集資料做過探索，得知資料對於週間週末以及早午晚皆呈現不同的特性)，因此加入了時間序列的 feature 使得分數上升了 0.006 左右，最後將 X 模型去掉權重的設置後在 Track 1 的表現也不錯，去掉權重的原因為 Track 1 使用了機率方式作為評分標準，加上權重使 Class balanced 之後反而會使結果偏離分布，導致表現較差。

附錄:** 另外也有嘗試 Random Forest 演算法，參數為 max_depth : 5、class weight {1: 75, 0: 3}、n_estimator : 100 (怕設太大會跑很久)，一樣使用 ***One-hot advertisement + 1-Day-per 8 hours time step feature 進行訓練，但由於 Day 0 的 class 1 比例與整體 data 的 class 1 比例可能沒有很接近，所以即使在較小的測試集上，用上述參數，可以做到 Ein≈3%、OOB_score≈97%、Class 1 預測比例≈4%。但實際使用全部的資料集訓練出來的結果 Ein 竟接近 20%，OOB_score 也剩下不到 80%；public 跟 private score 也只能做到 0.097627 與 0.098656，低於 xgboost 的結果。但事實上權重調好應該可以提升表現不少。但實作的經驗是，最佳的權重真的不太容易從小的訓練集中(only Day0)獲得，而 Random Forest 感覺上是對權重特別敏感的。而相較於 xgboost 還有 subsample 與 colsample_bytree(只選取部分 feature 來 train)等防止 overfitting 的方式，這次 project 做起來的想法是 xgboost 是一個非常強也非常方便的模型。

Model III. Logistic Regression

本模型一開始先使用只有擷取使用者特徵(User feature only)的方式，用 Logistic regression 進行訓練，針對 Track 1 做簡單的測試，結果如表四所示。由於我們一開始對 Track 1 繳交格式的理解錯誤，只對 half day8 的進行預測，導致表現分數相當糟糕(但事後發現原先 score 在 1.07 左右的模型，經過修正後應該都可以達到 0.15 左右以下的分數，不過我們沒有再針對這個比較簡單的 Feature 重 train 一遍)，但我們仍然可以從以下的測試結果理解到在 Track 1 中，針對參數 C 的調整對模型表現影響不太大根據我的想法這跟 regression 有很大的關係，因為增加 penalty 的係數只是使得每個點對回歸曲線的 error 等比例放大縮小，離回歸曲線越遠的點影響力較大，至於為何 C=1 最好呢?我想是因為剛好 C=1 時的 error function 最接近 Track1 的 evaluation function。

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

如果刻意在訓練過程加入 balanced 參數，反而會使結果變得較差，主要的原因是 Track 1 需要的是一個機率值，而原先數據集中，0 非常的多，所以將數據 balanced 後會大幅增加預測機率值，導致結果變差。另外推測有可能是 evaluation function 的關係，因為 logistic regression 和 Track 1 的 evaluation function 相近，得出的最佳解也相似，因此調整 data 的 balance 會破壞原本的分佈，得出另外一種分佈下的最佳解，以至於在 judge system 下表現不好。因此我們的 Model II, III 在做 Track1 的時候都不會加入 balanced 影響，後針對 Logistic regression 模型的改進也只透過 Feature 改變的方式，而不再針對參數另外進行調整。

表四、只有使用者特徵與不同 C 值的 Logistic Regression 結果

註：Eval 和 score 的巨大差異是因為測資錯誤

User feature only	Ein	Eval	Public score	Private score
C = 1	0.124575	0.145692	1.072432	1.081220
C = 1 ,balanced	0.155773	0.175854	1.186871	1.195794
C = 0.001	0.124786	0.145812	1.072434	1.072434
C = 100	0.125113	0.146012	1.072645	1.083055

表五、使用不同 Feature 的 Logistic Regression 結果

Track 1	Public score	Private score
User feature only	1.072432	1.081220
Advertise+User feature	1.071908	1.080638
Time+Advertise+User feature	1.071824	1.080535

在修正測資錯誤之後，就只有針對 Time + Advertise + User feature 的去做改善了，因為其表現最佳。後來使用的 Time feature 為 : 在 1-Day-per 1 hour time step 下有 168 維，亦即一星期有七天，一天有 24 小時，共 168 維，而我將測資共兩星期的 data'摺疊'成一星期，意即 day_0 的 time feature 會跟 day_7 一樣，用這星期二預測下星期二的用意，同理 1-Day-per 0.5 hour time step 下有 336 維。訓練結果如**表五**所示。再來使用最佳結果 1-Day-per 0.5 hour time step 來做 Track 2，用機率最高前 x %當作 1，後面設為 0。因為在分析 Track 2 的 evaluation function 後，覺得傾向猜 1 會表現較好，而訓練資料集的 Class 1 大約佔 3~4%，於是將 x%從 3%開始調，可以得到最佳的解在 10%附近，訓練結果如**表六**所示。

表五、使用不同 Time step 的 Logistic Regression 結果。

Track 1	Public score	Private score
1-Day-per 1 hour time step	0.145706	0.146343
1-Day-per 0.5 hour time step	0.145714	0.146353

表六、1-Day-per 0.5 hour time step + Logistic Regression 在 Track 2 的訓練結果。

Track 2	Public score	Private score
3%	0.112000	0.115559
6%	0.132413	0.134377
10%	0.138421	0.140240
12%	0.137598	0.140132

做完 Logistic regression 的想法是訓練速度不是很快，但有 stochastic gradient descent 的選項可以加快一點速度，使用這個模型的最重要原因，就是其 error function 很接近 Track1 的 evaluation function，所以可以確保在訓練的同時也可以將 Track 1 做好，另外就是後來發現調整參數影響不大，或許也是個優點，因此後來可以專注的在 Feature 抽取上做改進，最後做出來的結果也很不錯，顯示簡單的模型也可以做得還不錯。

Conclusion

由上述三個模型的結果可以發現，Feature 的選擇對於模型訓練相當的重要，但是針對廣告與時間使用簡單的 one-hot feature，就可以有不錯的結果，而如果將 Feature 分得越來越仔細精確的話 (Ex:有 time feature vs. No time feature、Time feature 以 2 小時切 vs. 以 8 小時切)，對整個模型表現的提升也是相當有幫助的。模型選擇的部分，Multi-Layer Perceptron 模型感覺上相對複雜，需要調整的參數也比較多，也不太容易了解，所以如果沒有對模型有透徹理解的話，可能不適合輕易地使用，否則針對模型較差的表現，要進行改善與討論時並不容易。

相比之下，XGBoost 可以動的參數也很多，但先將部分的參數設一個固定合理值(*subsample*、*colsample_bytree*、*Learning rate*)，就可以非常容易針對其他參數進行調整 Ex: *Max_depth*、*scale_pos_weight*，最後也僅需調整 *Max_depth*、*scale_pos_weight* 兩個參數後就有相當不錯的表現，又不太會有 overfitting 的問題，CP 值感覺很高；而 Random Forest 則是權重對模型預測的影響很大，但最佳權重的設置又不太容易，所以使用起來較不方便。使用 XGBoost 的時候為了使訓練較快較有效率，我們先選擇了較大的 Learning rate (0.3)以及設定 early stopping 在 50，或許某種程度上有犧牲了一些模型的精確性，我們預期 XGBoost 如果使用更精細的時間 Feature (2 hours step instead of 8 hours)，加上較保守的訓練方式(較小的 learning 以及寬鬆的 early stopping 條件)，應該在 Track 2 上可以有與 Logistic regression 相近的表現。

Logistic Regression 是針對 Track 1 測試最理想的模型，因為 Track 1 評分的方式和，所以在訓練模型時降低了 error 也某種程度上表示在 Track 1 測試上也做好了，這是我們覺得在 Track 1 中，使用相同的 Feature，但表現優於強大的 XGBoost 的原因。由於感覺透過參數調整隊表現影響不大，所以後續主要針對時間 Feature 進行改善，竟使 Track 2 表現大量進步到 Public score 與 Private score 分別為 0.138421、0.140240，(當下並無再用 XGBoost 對這個 feature 重新進行訓練)。除了再一次證明好的 feature 可以有效增進模型表現之外，也開始思考有時候使用一些相對熟悉單純的模型，其表現不一定會比較強、較 fancy 的模型差，值得在未來做相關研究的時候，多多留意這個簡單卻不容易發現的道理。

Contribution

王嚴徵 : XGBoost + Random Forest、報告統整

李伯倫 : Logistic Regression

王威翔 : Multi-Layer Perceptron