Machine Learning Final Report

-- Outbrain Click Prediction

NTU_b02901031_大雞雞

b02901045 毛學涵 b02901031 陳緯哲

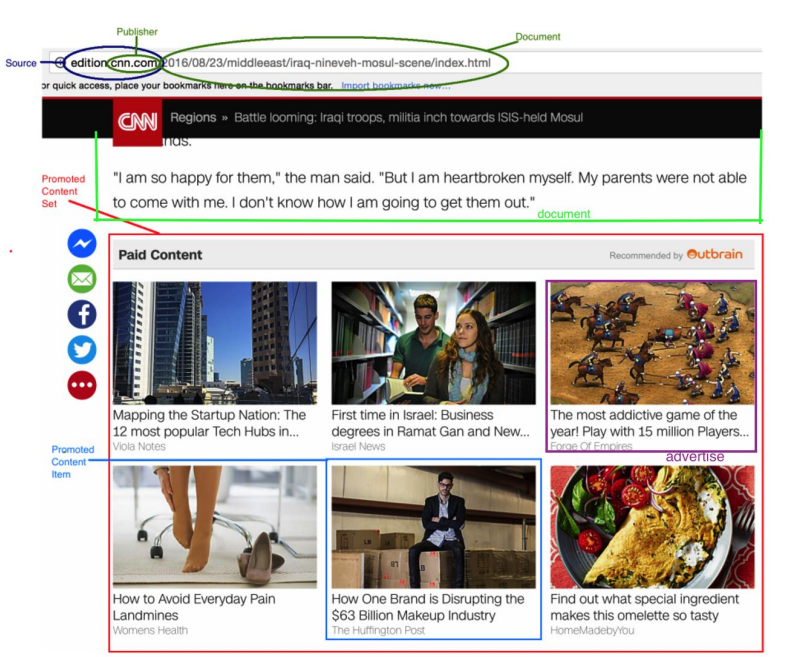
b02901126 張大方 b02901120 羅志軒

Introduction

Click-through rate (CTR) prediction是一個衡量廣告流量的關鍵指標，在machine learning領域是非常熱門的問題，隨著上網人數增加，其價值性也逐漸被Google、Outbrain等大型公司重視，本次期末報告將利用Outbrain網站提供的網頁資訊、廣告資訊和點擊數據等，預測某個頁面中的所有廣告被點擊的機率，並按照機率將廣告排序。

Data

這次的data 量相當大，總共大概100G左右，並且有許多不同data類型，做CTR 問題時，了解data 和分析data 也是最重要的一環，以下列出本次project我們使用的data：



clicks_train.csv

display_id：頁面的id，每個頁面包含一個document和多個廣告

ad_id：每個display對應到的廣告的id

clicked：這個頁面下所對應的廣告是否被點擊

promoted_content.csv

ad_id：廣告的id，每個廣告點擊進去會包含一篇document

document_id：廣告包含的內容

campaign_id：不清楚是什麼意思，但因為是廣告的feature，所以應該需要列入考慮

advertiser_id：廣告的出版商

events.csv (包含train set和test set的資訊)

display_id：頁面的id

uuid：進入這個頁面的使用者id

timestamp：使用者進入這個頁面的時間（從第一筆data的時間開始以毫秒計算）

platform：使用者進入頁面所使用的裝置

geo_location：使用者網路連線地點

documents_topics/categories.csv

document_id：每個document的id

topic/category_id：每個document對應的標題和種類

confidence_level：通常一個document的標題和種類有多種可能，confidence_level給予每種可能對應的機率

documents_meta.csv

document_id：每個document的id

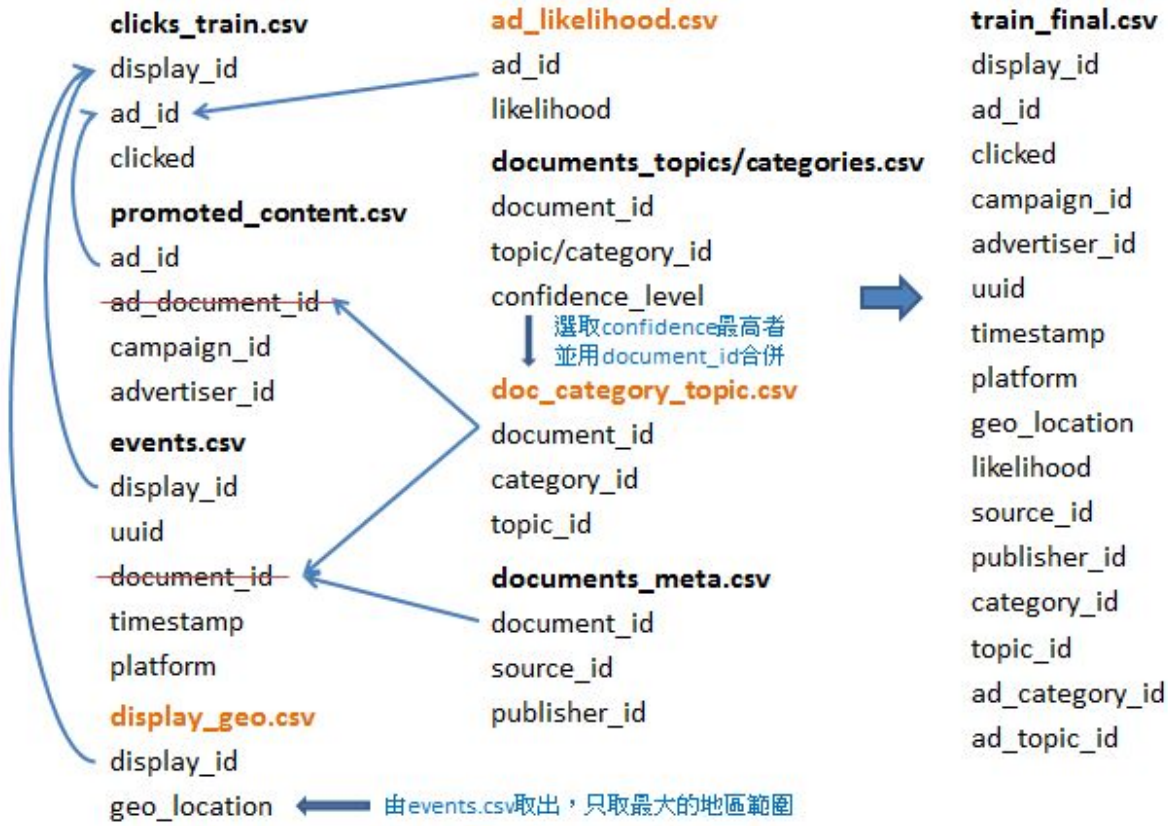
source_id：document的網站來源

publisher_id：document的作者

Preprocessing/Feature Engineering

首先利用pandas將我們需要的所有features merge在一起，如下圖。

其中ad_document_id及document_id在merge完後就從feature中抽掉，因為我們認為相較於id，topic與category更能表現文章的特性。



觀察merge完的資料，有些欄位會是NaN，我們將這些欄位以0取代；uuid、geo_location利用個別字元的ascii相連後轉換為整數形式。

最後train的時候，display_id、ad_id、timestamp並沒有被使用，理由是display_id及timestamp都是根據瀏覽的順序遞增，與文章特性無關，ad_id則是和document_id一樣，由topic及category代表；clicked用來作為label。

Training set and validation set

training set 及 validation set 我們在聽過top 10%的報告後，我們仿照他們的方法抓取clicks_train最後兩天的data，因為此種data最接近testing set 的時間點。其中，training set 和 validation set 的比率約為6:1

Model Description

Likelihood

使用 click_train.csv 裡的ad_id, 計算每個ad_id 被點擊的likelihood。

$$w = \frac{c + 12m}{12+N}$$

c 是指這個廣告在data裡被點擊幾次，m是整個training data中平均一個廣告會被點擊機率，N是指這個廣告出現在training data中的次數，若單使用 c/N 可以簡單計算出每個廣告目前被點擊的機率，而加上 12*m 可以看做每個廣告目前被點擊的機率和平均一個廣告被點擊機率的 weighted-sum，用於增加 likelihood 預測的一般性。

Decision Tree

decision tree是一個用於處理分類問題的結構，原理是從資料中抽出最具分類價值的feature當作內部節點，並將以此節點分類的資料產生對應的分支，重複以上步驟直到滿足終結條件。

Fieldaware factorization machines (FFMs)

factorization machine（FM）是一個廣泛被使用於解決稀疏數據下feature組合分類問題的模型，其基本概念是在線性回歸（linear regression）模型中加上兩兩feature間的相關性，最終完成數據的分類(ex: clicked or not)，以下為 FM模型的方程式：

$$y(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

其中 x_i 是第 i 個 feature， v_i 代表第 i 個 feature所產生的 latent vector，$\langle , \rangle$ 則代表向量內積，$\langle v_i, v_j \rangle$ 可以視為 x_i 和 x_j 相關性，是整個FM的核心概念，x_i 和所有其他的 x_j 組合都可以用來更新 v_i 的參數，很大程度避免稀疏數據造成的影響，此外，透過調整每個 feature的 latent vector大小 k也可以有效控制數據量大小，降低模型的參數量。FFMs則是一個以factorization machine爲基礎加入 field概念的升級版模型，其概念是將 feature的 type事先分類 (如：天氣和日期為不同的 type)，這裡稱之為field，訓練過程中，在尋找每個 feature 和其他 feature的關聯性時，會依照對方 field的的種類，使用不同 latent vector來訓練，FFMs方程式如下：

$$y(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_{i,f_j}, v_{j,f_i} \rangle x_i x_j$$

f_i 為第 i 個 feature所屬的 field。

在使用 FFMs前，我們必需將原本的 training feature 轉換成 "field_id: feat_id:value" 的形式，field_id為資料所屬 field的 id，feat_id為資料本身的category id，value為資料的值，以下以這次 project 的資料為範例：

clicked	display_id	ad_id	uuid	likelihood
1	1	42337	cb8c55702	0.167

	Field_id	Feat_id	Value
display_id	1	1	1
ad_id	2	42337	1
uuid	3	101700003256(由uuid轉換)	1
likelihood	4	1	0.167

Ouput format：

0(clicked) 1:1:1 2:42337:1 3:101700003256:1 4:1:0.167

Leakage

使用者在點擊廣告後，會進入一個頁面，而這個頁面呈現的文章，可以對應到一個document_id，因此我們可以藉由document_id 對應到一個ad_id。藉由以上關係，我們可以推斷使用者如果看過這個document，也就代表使用者曾經點擊過某個廣告。利用這樣的關係，我們可以從page_views 這個檔案找出使用者點擊過哪些document_id，再對應回相對應的廣告。

在testing set 裡，我們可以找到少部分的資料可以應用這樣的關係去判斷這筆資料是哪一個廣告被點擊，這筆資料就叫做leakage。換句話說，這筆資料算是某種“答案”。當然，擁有此種關係的data非常少，因為只有少部分的事件被記錄下來。但是在

LB score 上也有不小的影響。藉由我們的實驗，這筆data可以提升每個model 約0.01~0.02的LB score。

Experiments and Discussion

A.Decision Tree

Feature :

display_id	*campaign_id*	*source_id*	*ad_id*	*advertiser_id*	*ad_publisher_id*	*likelihood*	*category_id*	*uuid*
	v	*v*		*v*	*v*		*v*	*v*
topic_id	*ad_category_id*	*ad_topic_id*	*clicked*	*publisher_id*	*platform*	*geo_location*	*ad_source_id*	
v	*v*	*v*	*v*	*v*	*v*	*v*	*v*	

在decision tree 的實驗中，我們取了13個feature(不包含click)去train 我們的model,
在decision tree 的實驗中，我們將decision tree 預測出的機率和 likelihood 做一個weighting。我們做了不同weight 的調整。

$$probability = w_{likelihood} * likelihood + w_{tree} * P_{tree}$$

**best score:* $w_{likelihood}$ *:0.9* w_{tree} *: 0.2 val = 0.6177 LB score = 0.6122*

	$w_{likelihood}$:0.8 w_{tree}:0.2	$w_{likelihood}$:0.6 w_{tree}:0.4	$w_{likelihood}$:0.4 w_{tree}:0.6	$w_{likelihood}$:0.2 w_{tree}:0.8
validation	0.6139	0.5967	0.5965	0.5965
LB score	0.6084	0.5894	0.5894	0.5897

*without leakage

從分數可以看出decision tree的確能夠做出合理的預測。我們並沒有把likelihood 當作feature的原因是，decision tree 是藉由entropy的減少去learn feature的優先順序，因此沒有辦法有效利用 分數的feature，而是藉由weighting的方式增加score。

B. FFM

trainging set (features):

t_ffm (= t2_ffm + diplay_id + ad_id):

display_id	campaign_id	source_id	ad_id	advertiser_id	ad_publisher_id	likelihood	category_id	uuid
v	v		v	v		v	v	v
topic_id	ad_category_id	ad_topic_id	clicked	publisher_id	platform	geo_location	ad_source_id	
v	v	v	v		v	v		

t2_ffm:

display_id	campaign_id	source_id	ad_id	advertiser_id	ad_publisher_id	likelihood	category_id	uuid
	v			v		v	v	v
topic_id	ad_category_id	ad_topic_id	clicked	publisher_id	platform	geo_location	ad_source_id	
v	v	v	v		v	v		

t_expr_ffm:(= t1_ffm + source_id + publisher_id)

display_id	campaign_id	source_id	ad_id	advertiser_id	ad_publisher_id	likelihood	category_id	uuid
v	v	v	v	v		v	v	
topic_id	ad_category_id	ad_topic_id	clicked	publisher_id	platform	geo_location	ad_source_id	
v	v	v	v	v	v			

t_expr2_ffm:(= t_expr_ffm + uuid + geo_location)

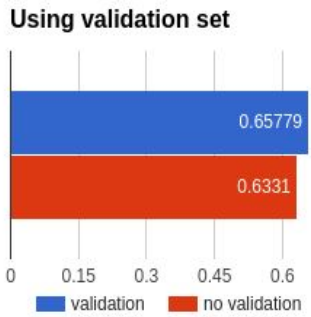
display_id	campaign_id	source_id	ad_id	advertiser_id	ad_publisher_id	likelihood	category_id	uuid
v	v	v	v	v		v	v	v
topic_id	ad_category_id	ad_topic_id	clicked	publisher_id	platform	geo_location	ad_source_id	
v	v	v	v	v	v	v		

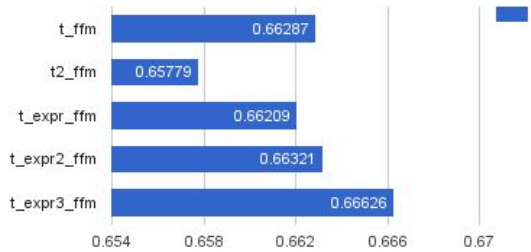
t_expr3_ffm:(= t_expr2_ffm + ad_publisher_id + ad_source_id)

display_id	campaign_id	source_id	ad_id	advertiser_id	ad_publisher_id	likelihood	category_id	uuid
v	v	v	v	v	v	v	v	v
topic_id	ad_category_id	ad_topic_id	clicked	publisher_id	platform	geo_location	ad_source_id	
v	v	v	v	v	v	v	v	

1. **Compare the results of using validation set or not**

使用t2_ffm當作training set，比較同筆資料在有無使用validation set下，結果好壞的差異。(**圖一**)可看出有使用validation set可以避免overfitting，得到較好結果。





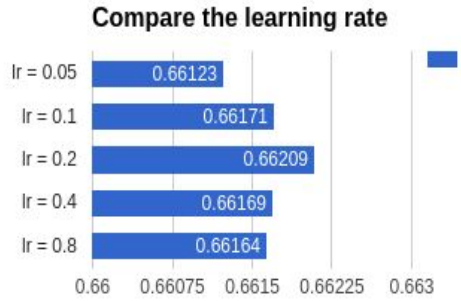
圖一　　**圖二**

2. **Compare different training set**

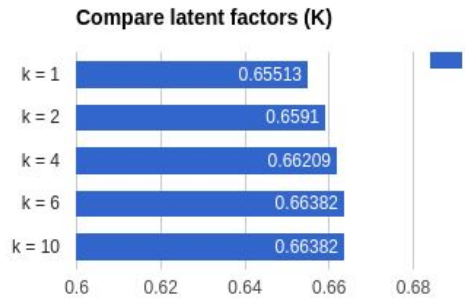
使用**不同的features**當作不同的training set，比較不同的features對於結果的影響。(**圖二**)可以看到**t_ffm**比**t2_ffm**多了兩項features**(display_id, ad_id)**，正確率從**0.65779-->0.66287**。**t_expr_ffm**比**t_ffm**少了**(uuid, geo_location)**，但是多了**(source_id, publisher_id)**，正確率些微降到**0.66209**。由此可以看出對於**features的增加與減少對於整體的正確率影響很大**，因此我們嘗試再多加入features看表現能不能更加提升。**t_expr2_ffm**比t_expr_ffm多加回**(uuid, geo_location)**，正確率提升到**0.66321**，**t_expr3_ffm**再加入**(ad_source_id, ad_publisher_id)**，提升到**0.66626**，為所有training set中表現最好的模型。

3. **Compare the results of different learning rate**

固定使用t_expr_ffm當作training set，比較在不同**learning rate**下對結果的影響。(**圖三**)可以看到在其他參數固定的情況下，**learning rate在0.2**左右時可以得到比較好的結果。



圖三



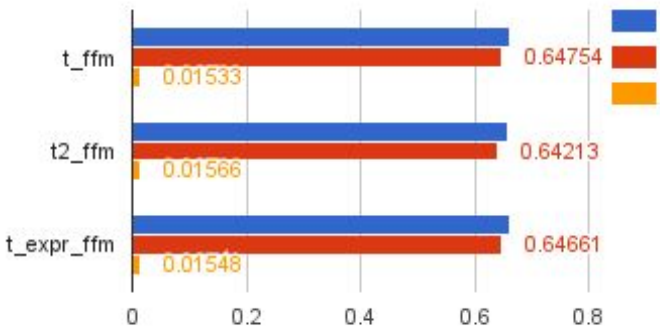
圖四

4. Compare the results of different latent factor

固定使用t_expr_ffm當作training set，比較在不同**latent facor**下對結果的影響。(**圖四**)可以看到在其他參數固定的情況下，**latent variable (k)在6以上**時可以得到比較好的結果，但是**k=10時取更多維度-->跑比較久**的情況下，**表現與k=6卻差不多**。

5. Compare leakage effect for different training sets

下圖藍色為加入leakage的效果，紅色則無。可以看出當原先表現(紅色)已經比較好時，再加入leakage則增加的正確率沒有那麼大，可以得知此方法再高正確率時飽和，並沒有辦法從learning的角度變好。黃色為增加正確率的幅度，紅愈高黃則愈低



Work Division

陳緯哲：Feature engineering, ffm model , report.

張大方：Data loading, Feature engineering, ffm experiments, report.

毛學涵：Feature engineering, decision model building, report.

羅志軒：Data anlysis, Feature engineering, report.

Reference :

FFM

https://github.com/guestwalk/libffm

http://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf

Decision tree

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html