

kaggle competition report

Team name: Mozart - KKboxer

數據處理

label_train_source, label_train_target, label_test_source:

- session_id
- song_id
- unix_played_at
- play_status
- login_type
- listening_order
- played_at_hour: 使用unix_played_at轉換, 取出目前的小時數
- unix_played_at_next: 下一個歌曲撥放的時間
- played_time: 歌曲播放時間, $\text{unix_played_at_next} - \text{unix_played_at}$
- played: 0, 歌曲聆聽低於30秒; 1, 歌曲聆聽大於等於30秒; 0.5, 資料當中的最後一首歌曲, 不知道聆聽的時間是多久

製造出played column是為了將聆聽低於30秒以下的歌曲去除, 避免訓練的時候將無用的資訊學習進去。

- rating (positive): $\text{played_time} / \text{song_length}$
- rating (negative): 0
- 對每一個session, 產生出5個negative sample (songs)

因為訓練數據本身是implicit data, 為了要使用FM一類的模型, 必須要將implicit data轉換成explicit data, 我們使用以上的方式產生rating。

參考以下的資料:

[Collaborative Filtering based Recommender Systems for Implicit Feedback Data - Sumit's Diary \(reachsumit.com\)](#)

meta_song以及其他歌曲資料:

- song_id
- artist_id
- song_length
- album_id
- language_id
- album_month
- composer_id, lyricist_id, producer_id, genre_id


有些欄位會是空數值, 例如說album_id, album_year...等, 一些合理的推論是:

- 這些欄位當中，只要其中一個欄位不是空的，那這首的辨識度就會很高(例如 album_id, language_id)。我們將完全都沒有任何資訊的歌曲刪除，剩下約99萬首歌曲。
- 空值：我們可以想像一首年代久遠歌曲，其創作者以及演出年份未知，只知道演出的語言。我們可以另外為空值設定一個專門的類別。
- 異常值：一些歌曲的 album_year 是未來日期，我們將他視同空值處理，將他設定成一個特殊的數字(例如0)，讓模型學習這個特徵。

模型訓練

baseline - random

比賽評分計算方法中，coverage 的分數占20%，所以我們可以使用隨機的方式產生完全不重複的歌曲進行推薦，拉高coverage的分數。

	result-5-random.csv	0.06941	0.06943	<input type="checkbox"/>
	Complete · Chris_data_science · 6d ago · It's also a baseline submission wi...			

matrix factorization - ALS (Alternating Least Square)

比較暴力一點的作法是把每一個session當做一個user，輸入模型進行訓練。但是在矩陣分解的算法中，模型無法預測新的user。所以我們就將測試資料label_test_source合併到訓練資料當中。

矩陣分解的算法只能使用user- item的interaction，不能結合其他特徵，也就是我們只需要準備一個user - item的矩陣r，每一個entry， r_{ui} 代表一個用戶對一個物品的互動次數。

這個演算法將原本的矩陣分解改為以下的形式：

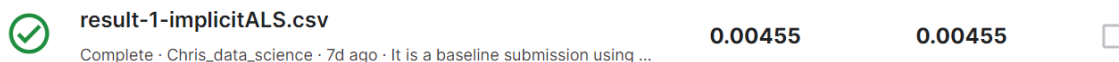
$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

$$c_{ui} = 1 + \alpha r_{ui}$$

將原本implicit data的評分矩陣r，產生出p - preference, c - confidence, 並優化以下式子：

$$\min_{y_*, y_*} \sum_{u, i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

我們使用implicit這個library來進行訓練，結果如下：



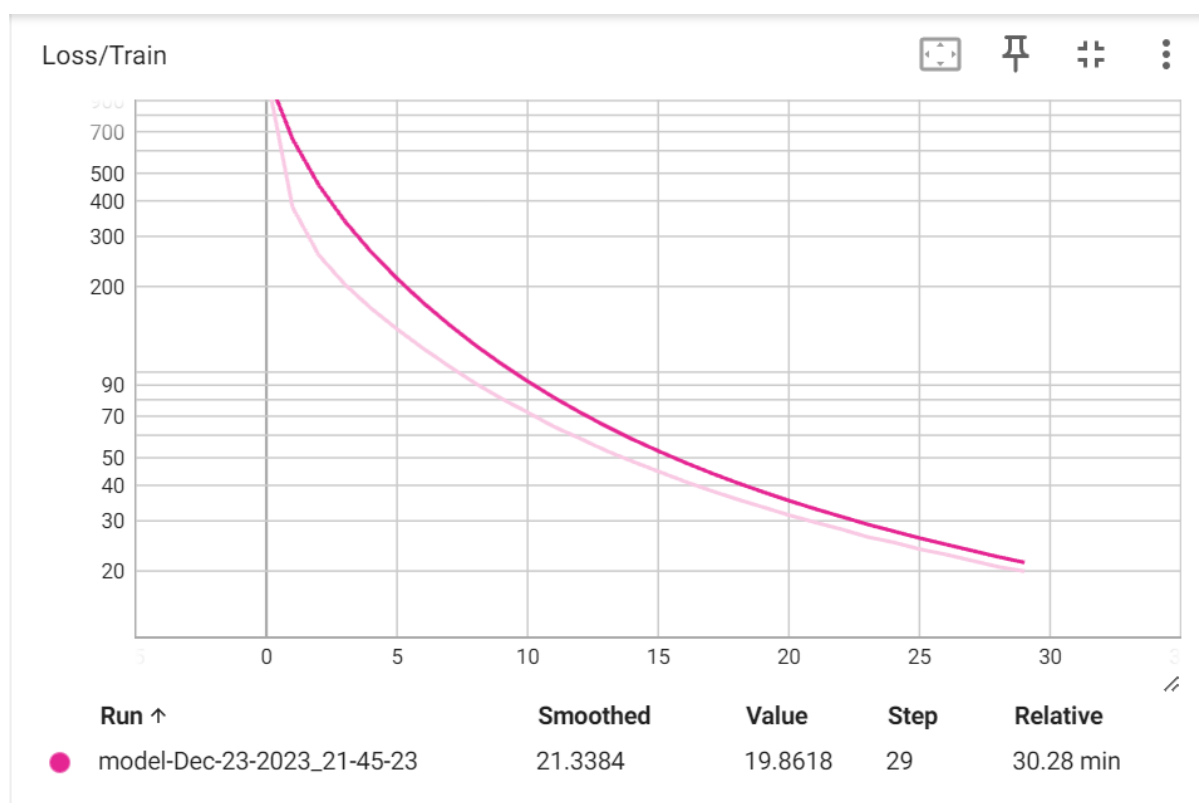
這個算法沒有考慮推薦結果的ranking，分數反而比random還要低。

BPR (Bayesian Personalized Ranking)

在先前的矩陣分解算法當中，我們將沒有互動的user - item設為最低的rating，在一個session中沒聽過，不代表他接下來不會聽(因為一個session的訓練資料也才20首歌曲)；而且，這樣子的優化方式沒有考慮推薦出來的物品的排序(因為算法直接對評分優化)。

BPR這個演算法是針對implicit data所發明的算法，他是利用貝氏機率，pairwise loss的方法來對推薦排名做優化，可以推測pairwise loss比起pointwise loss優化的效果還要好。

我們使用recbole library進行訓練，結果如下：



利用tensorboard將loss曲線可視化，可以看到還有優化的空間。

epoch = 10

✓	result-6-recboleBPR.csv	0.08439	0.08589	<input type="checkbox"/>
	Complete · Chris_data_science · 4d ago · test recbole BPR.			

epoch = 30

✓	result-9-recboleBPR-30th.csv	0.1032	0.1048	<input type="checkbox"/>
	Complete · Chris_data_science · 2d ago · 30 epoch for BPR			

與baseline相比，可以推測這個推薦方法應該有猜中一些target。

Factorization Machine

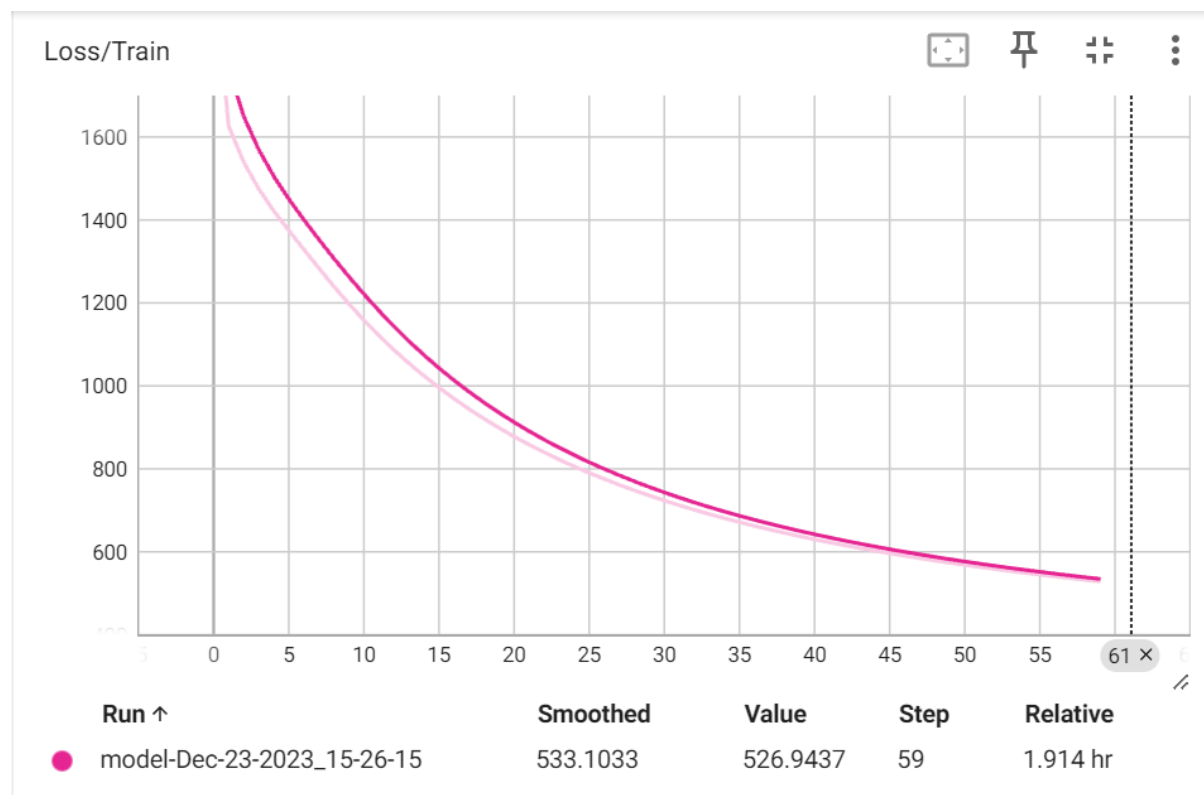
我們為了結合meta feature進行訓練，以獲得更好的效果，我們使用FM的方法結合以下feature進行訓練



- 互動數據: label_train_source, label_test_source中的所有欄位。
- 所有的meta_data: 所有欄位除了composer_id, producer_id, titletext, lyricist_id。

我們將使用以下方法產生評分(沒有實現負採樣):

- rating (positive): played_time / song_length
- rating (negative): 0

我們使用recbole的FM進行訓練，結果如下：



	result-7-recboleFM-10th.csv Complete · Chris_data_science · 3d ago · 10 epoch training for recbole FM.	0.00054	0.00047	<input type="checkbox"/>
	result-10-recboleFM-60th.csv Complete · Chris_data_science · 2d ago · 6 epoch using recbole FM.	0.0013	0.00126	<input type="checkbox"/>

結果可說是慘不忍睹。

我們從推薦結果觀察發現，推薦歌曲的總數大約為 $14萬 \times 5 = 70萬$ 首歌曲，但是我們推薦出來的結果，不重複的歌曲只有大約2000首，模型實際上沒有很好的學習到排名推薦 (因為FM本身是用pointwise的方式進行優化)

其他來不及測試的方法：

rankFM (使用rank優化的FM), session-based recommendation (ex: GRU4Rec使用互動的sequence進行訓練), Graph based recommendation (比較好處理composer, producer太多的問題)

賽後總結：

第一次參加kaggle競賽就從千萬級的數據入手，比較困難的地方在於

- 數據處理:隨便一個操作就好好幾秒，弄不好的話就會跑太久，所以我們這次練習到相當多的pandas資料處理技巧
- 數據內容:與一般的資料科學的資料集不一樣的地方是，所有的feature都被hash過，無法透過統計得到一些比較直觀的結果。
- 模型訓練:對於大學部的同學而言(我們三個成員都是)，我們比較難有機會接觸GPU等運算資源，在本地設定環境很困難；並且在一般的google colab上面處理資料 / 進行訓練，RAM絕對不夠。這一點我們花了很長的時間才解決，直到最後一個禮拜才搞定，後來是使用google colab pro才有辦法進行訓練。如果早點開始的話，說不定用其他AI的深度學習模型進行訓練會有更好的效果。