ADL Final Team 64

我們實作的 code 在：https://github.com/Kaiserouo/ADL22-Final

Abstract

我們使用hahow2012年度一整年（1/1-12/31）的用戶資料，建構課程推薦系統模型。嘗試了 Two-tower, Behavior sequence transformer（BST）, Collaborative Filtering 以及 Neural Collaborative Filtering 四種深度學習模型。最終發現除了在 seen course 的task（已知使用者購買紀錄，預測購買課程）上 BST 有更好的表現以外，其餘的 task 都是 Rule-based 的方法表現更佳。

Introduction

近二十年來，機器學習被廣泛用於推薦系統，早期著名的例子是2006年的netflix 1 million dollar prize contest。

在推薦系統裡，有兩種經典的設計風格，Content-based recommendation 和 Collaborative Filtering。

在Content-based裡，我們會針對每一個用戶，設計一個predictor，將該用戶過去買過的商品encode，將商品的encodings和對應的評價來當訓練資料，訓練完畢後可以預測該用戶對每個商品的評價該方法的效果是對於相似的商品我們可以找出相似的商品來推薦用戶，缺點是難以找到好的商品encoding方式以及無法使用來自其他用戶對同商品的評價。

在Collaborative Filtering裡，想法是找出喜好相似的用戶，並使用他們喜歡的商品來推薦用戶。一個經典的方法是矩陣分解 (matrix factorization)：我們會創建一張矩陣存放用戶對商品的評價當作輸入，每個entry對應一個用戶對一個商品的評價，在該方法裡我們會將每個用戶和每個商品用一個同維度的向量表示，並以用戶與商品的內積代表我們預測該用戶對此商品的評價。Collaborative Filerting可以很好地根據所有用戶對商品的評價來推薦某個用戶，缺點是新用戶或是新商品沒有購買資料無法有效推薦他喜歡的商品（Cold Start Problem）。

現代的推薦系統裡並不會侷限於上述兩種的設計，可能會結合兩者（hybrid approach），例如netflix曾設計比較用戶間的搜尋與觀看紀錄(Collaborative Filerting)，以及根據用戶評價高的電影來推薦相似的電影(Content based)。

在這份work裡，我們使用Hahow 2012年裡用戶購買的課程紀錄以及所有的課程和用戶資料當作訓練資料，設計推薦系統來推薦用戶課程以及他可能會感興趣的領域(subgroup)，並使用map@50 (Top 50 Mean Average Precision)當作evaluation criterion。

我們嘗試五種方法來設計推薦系統，從經典的collaborive filtering (CF)，到結合神經網路的neural collaborative filtering (NCF) 和 two tower model，以及使用近年很熱門的transformer

架構的behavior sequence transformer (BST)，最後是以統計角度來實作的rule-based方法。

Related work

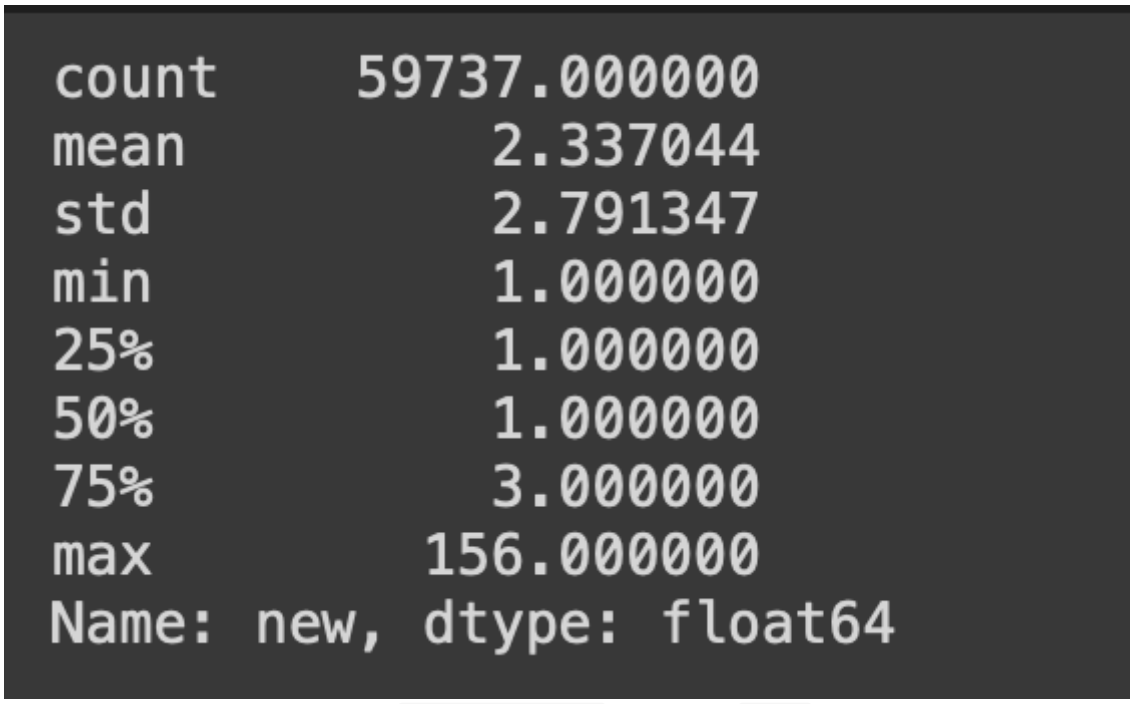
- Two-tower model
  - 在過去的文章提到可以用某種類似 encoder 的模型去對 user / item feature vector 去做 encoding，然後用某種 similarity (e.g. dot product, cosine similarity) 當成分數。
  - https://www.linkedin.com/pulse/personalized-recommendations-iv-two-tower-models-gaurav-chakravorty/
  - 我們只有有買下的 user-item pair 而沒有太多「某 user 不喜歡某 item」的資訊，直接訓練可能會變成全部預測最高分數。過去訓練 matrix factorization 的 loss function 裡，weighted mean square error 可以透過一個加權項 w 來使用沒有買過的 negative sampled user-item pair。
    - https://developers.google.com/machine-learning/recommendation/collaborative/matrix

- Behaviour sequence transformer(BST)
  - 參考自阿里巴巴開發的商品推薦模型（Chen et al., 2019）將使用者的購買紀錄視為是一個有潛在規律的sequence。借用了NLP當中transformer處理句子的想法，把商品的資訊按照購買順序接起來做embedding，用self-attention機制去學一個sequence的資訊，也就是用transformer去學商品之間的dependency
    - 希望找到一個function S，預測使用者u行為序列的最後一個位置，即下一個購買行為可能是traget item v 的機率：
    - $S(u) = \{v_1, v_2, v_3, \ldots, v_n\}$
    - reference:Chen, Q., Zhao, H., Li, W., Huang, P., & Ou, W. (2019, August). Behavior sequence transformer for e-commerce recommendation in alibaba. In Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (pp. 1-4).
    - sample code: https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/structured_data/ipynb/movielens_recommendations_transformers.ipynb

- Collaborative Filtering (CF)
  - Matrix Factorization Based
    - Singular Value Decomposition
    - Alternating Least Square
    - Bayesian Personalized Ranking
    - Logistic Matrix Factorization

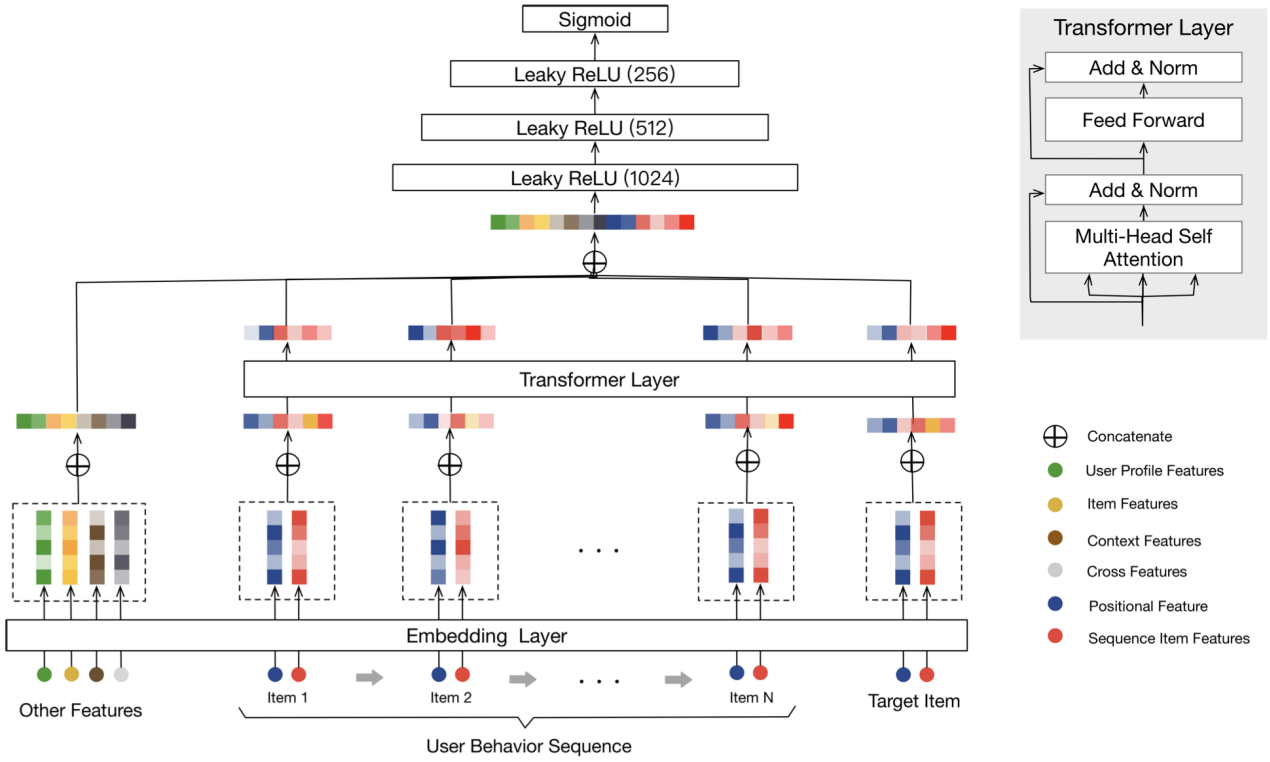
Approach

Behaviour sequence transformer(BST)

- data preprocess
  - raw data
    - user profile： 使用者年齡、興趣，性別
    - item： 商品資訊如名稱
    - rating: 特定商品使用者的評價、購買時間，因為我們沒有使用者購買課程的時間，所以因此改用原始資料的順序，假定使用者購買課程的時間間隔都是等距的。使用者有購買的課程，我們將其評分編為1。
  - 過半數的使用者都只有買過一套課程，但模型至少需要一個長度為二的序列，第一個位置代表過往的行為，第二個位置代表未來的行為。
  - 
    ```
    count    59737.000000
    mean         2.337044
    std          2.791347
    min          1.000000
    25%          1.000000
    50%          1.000000
    75%          3.000000
    max        156.000000
    Name: new, dtype: float64
    ```
    故而我們創一個課程名稱為`hahow線上課程`，類別為`教育`，給定所有使用者都購買過這套課程，也是假定既然是hahow的使用者應該會對該套課程內容有興趣
  - 訓練資料內容依序如下，各個欄位之間我們以`|`串連：
    1. 使用者編號：我們將原本的字串id重新編為int型態
    2. 使用者購課紀錄：統一長度為2
    3. 評分： 統一為1 1代表使用者對序列的第一、二個課程評分為1
    4. 職業：缺失資料編為 other
    5. 性別：缺失資料編為 X
    6. 興趣: 各個之間以`_`間隔
  - 以下方為例，該位使用者依序購買過的課程編號為：
    `608, 624, 624, 678`

3|608	624|1.0	1.0|other|投資理財_理財_攝影_影像創作_投資理財_投資觀念_藝術
3|624	78|1.0	1.0|other|投資理財_理財_攝影_影像創作_投資理財_投資觀念_藝術

- model architecture

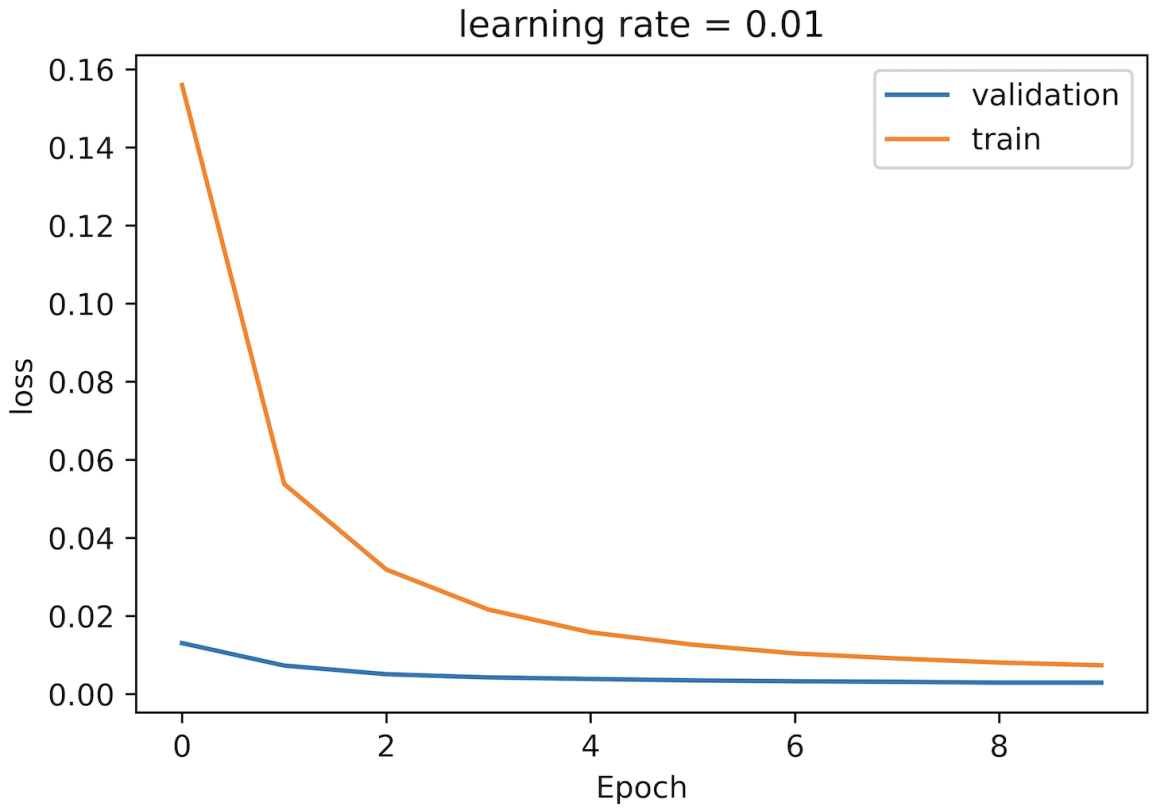


(chen et al., 2019)

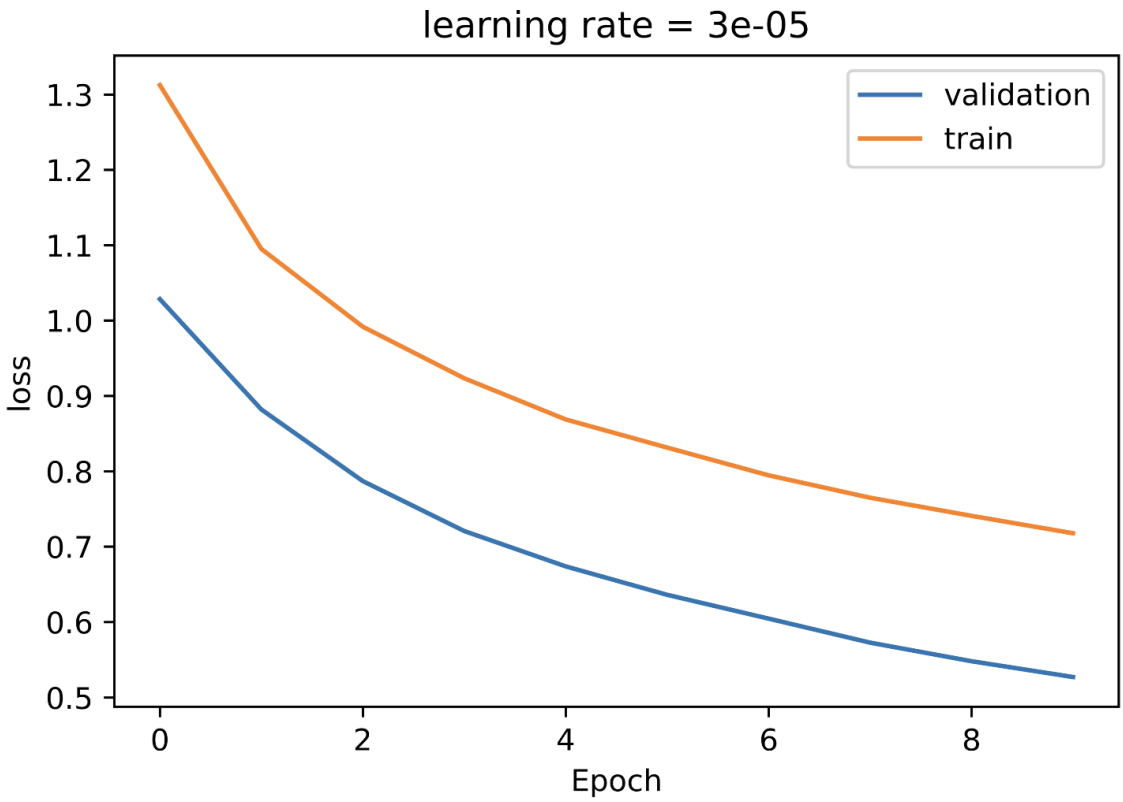
  - embedding layer
  - transformer
  - MLP layers
  - loss: mean-square error
    $\frac{1}{n}\sum_{i=i}^{N}(y-\hat{y})^2$
    - y為1
    - $\hat{y}$為預測出來為1的機率
    - n為訓練資料筆數
  - output: 使用者購買 target item 的機率
    - seen course: 總共有728個課程，我們分別預測使用者會購買這728個課程的機率。最後用機率高低為課程排序
    - seen subgroup: 把預測出來的課程轉換成為其相對應的subgroup

- experiment
  - 我們首先使用sample code當中預測的參數，再針對各個項目去做不同的嘗試:
    - batch size = 256
    - learning rate = 0.01
    - dropout rate = 0.1

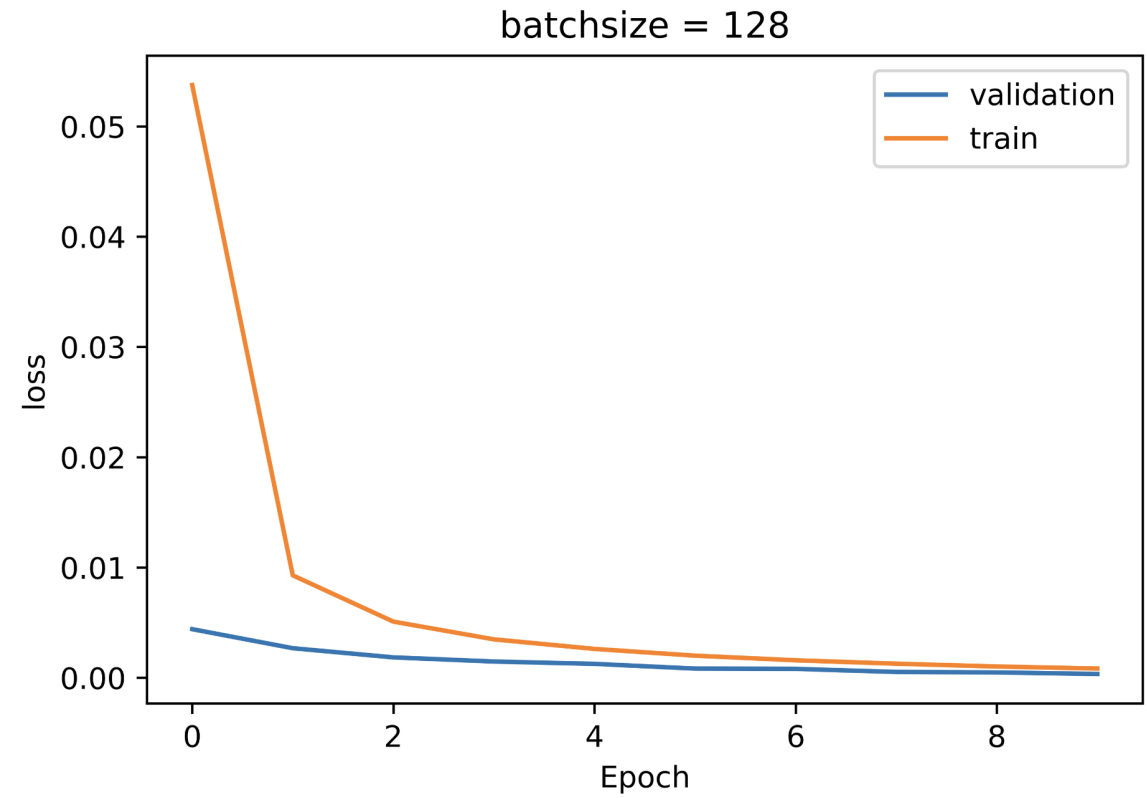
- number of epoch = 10



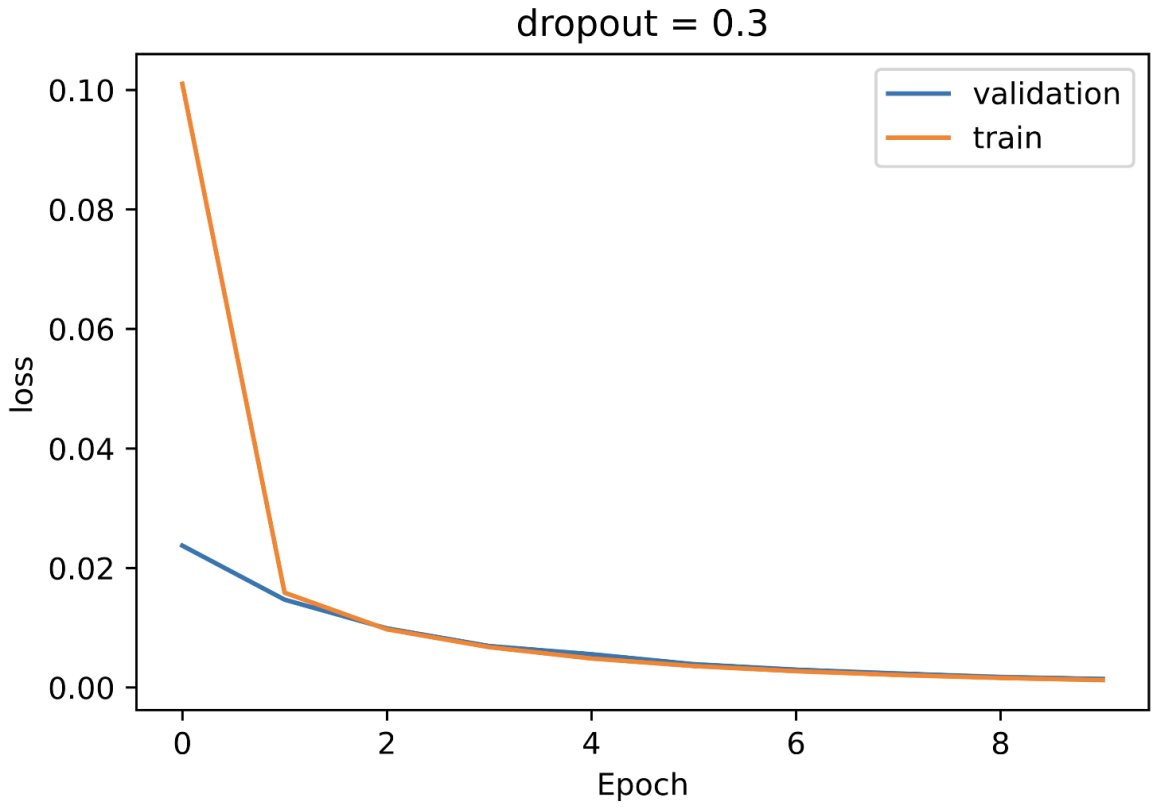
原始參數



調整 learning rate 為 3e-5



調整 batch size 為 128



調整 dropout rate 為 3e-5

- performance
  - 最終原始sample code 預設的參數表現最好
    - seen course: 0.08492
    - seen subgroup: 0.1984

- discussion
    - limit
        - loss/eval設計：訓練資料當中只有評分為1的資料（使用者購買過該課程），而沒有評分為0的資料（使用者不喜歡該課程），因為沒有購買過，不代表使用者不喜歡。而這導致模型最後都傾向預測使用者會購買target課程，loss趨近於零，但事實上模型還有進步空間。
        - missing data: BST 使用者的購買行為可能是跨平台的。例如曾經在 coursera 上購買、觀看其他的課程，之後才到hahow上面購課，但我們只有 hahow 上的資訊，我們的行為序列可能不完整。
        - 序列過短：多數使用者購買課程數量太少，導致我們序列長度只能設定為2，可能也是基於線上課程的購買，本來就不會太頻繁的緣故
        - 只適用於有使用紀錄的使用者，沒辦法針對新的使用者單純只看 user profile 就做課程推薦，只適用在 seen course 上面

Two-Tower Model

我們定義：

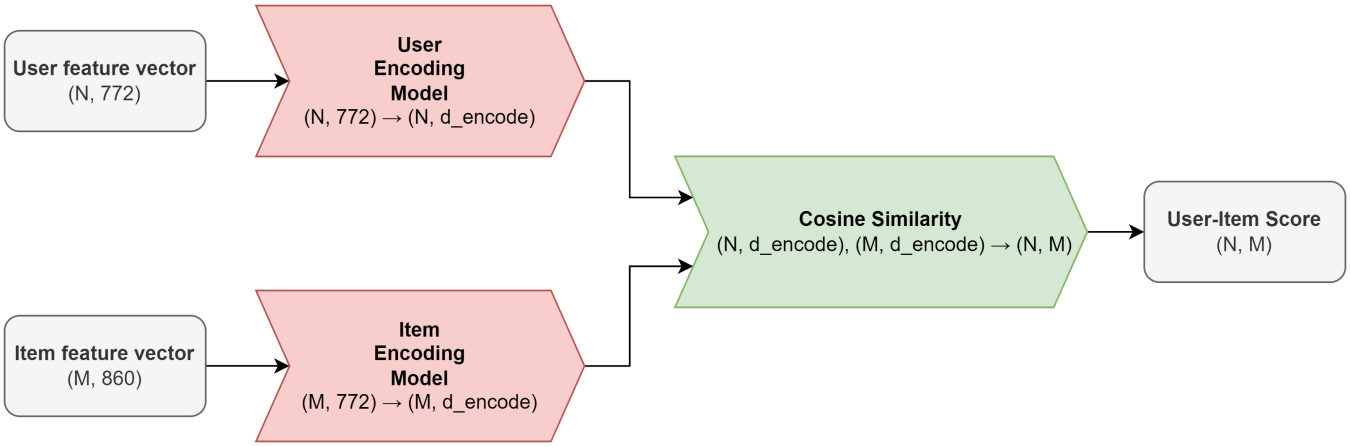
- `concat()`：串接 tensor。
- `onehot()`：把分類做成 one-hot tensor。
- `embed()`：把字串送進 BERT 裡，拿 hidden layer 當作他的 embedding。
    - 用的是 hfl/chinese-roberta-wwm-ext。
    - 在 256 word 截斷，用 BERT 的第一個字的 output.last_hidden_state。

那我們先把 user / item feature 變成向量形式：

```
user_feature = concat(
    onehot(gender),
    embed("|".join(occupation_title, interests, recreation_name))
)    # shape (N, 772), N: number of user in a batch
```

```
item_feature = concat(
    log(price + 1),
    onehot(sub_groups),    # all existing subgroups have value 1
    embed("|".join(
        course_name, sub_groups, topics, will_learn, required_tools,
        recommended_background, target_group,
        "|".join(chapter_item_name)   # all chapter's name of this course is included
    ))
)    # shape (M, 860), M: number of items in a batch
```

我們可以定義我們的 model：我們對處理過的 feature vector 全部做 autoencode 成同樣的維度 `d_encode`，並且定義一個 user-item pair 的定義為此 user / item 被 encode 完後的向量的 cosine similarity。



Experiments

我們使用的 Train dataset 中含有所有 train.csv 的 user-item pair (約 13 萬個)，以及隨機抽取一樣數量，沒有出現在 train.csv 的 user-item pair 當作 negative sample，共計約 26 萬個 (user, item, label) pair。

Loss function 將會用 Weighted mean square error 當作 loss function，也就是：

- 有買過的 user-item pair，label 為 1，loss 為 $(\text{label} - \text{score})^2$。
- 沒買過的 user-item pair，label 為 -1，loss 為 $w * (\text{label} - \text{score})^2$。

我們沒有 fine-tune BERT，只把 pretrained BERT 當成一種 word embedding。

至於 subgroup 預測，我們選擇用此模型預測完 course 之後，取前 50 名然後對 subgroup 統計並根據出現數量排名。另外，我們也會稍微利用交易紀錄，對於在 train.csv 有紀錄的 seen user，我們會再統計的時候再次把這些 course 加入計算。

我們的兩個 encoding model 使用 DNN：

```
optimizer = AdamW (lr = 1e-3)
w = 0.2
epoch = 50
batch size = 512

model = TwoTowelModel(
  (user_model): EncodingModel(
    (model): Sequential(
      (0): Linear(in_features=772, out_features=500, bias=True)
      (1): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (2): Dropout(p=0.1, inplace=False)
      (3): LeakyReLU(negative_slope=0.01)
      (4): Linear(in_features=500, out_features=500, bias=True)
```

```
      (5): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (6): Dropout(p=0.1, inplace=False)
      (7): LeakyReLU(negative_slope=0.01)
      (8): Linear(in_features=500, out_features=500, bias=True)
      (9): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (10): Dropout(p=0.1, inplace=False)
      (11): LeakyReLU(negative_slope=0.01)
      (12): Linear(in_features=500, out_features=500, bias=True)
    )
  )
  (item_model): EncodingModel(
    (model): Sequential(
      (0): Linear(in_features=860, out_features=500, bias=True)
      (1): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (2): Dropout(p=0.1, inplace=False)
      (3): LeakyReLU(negative_slope=0.01)
      (4): Linear(in_features=500, out_features=500, bias=True)
      (5): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (6): Dropout(p=0.1, inplace=False)
      (7): LeakyReLU(negative_slope=0.01)
      (8): Linear(in_features=500, out_features=500, bias=True)
      (9): BatchNorm1d(500, eps=1e-05, momentum=0.1,
             affine=True, track_running_stats=True)
      (10): Dropout(p=0.1, inplace=False)
      (11): LeakyReLU(negative_slope=0.01)
      (12): Linear(in_features=500, out_features=500, bias=True)
    )
  )
  (cos_sim): CosineSimilarity()
)
```

這個模型在 Kaggle 上的 performance 為：

- seen course: 0.0342
- unseen course: 0.05396
- seen subgroup: 0.25165
- unseen subgroup: 0.21728

Discussion

這個模型最大的好處是可以應付 seen 和 unseen 的 user 和 course，交易紀錄只用來訓練模型，在預測的時候並不需要，只需要 user 和 course 的 feature 即可以對此 user-item pair 做預測。並且因為是 cosine similarity，我們不需要每個 user-item pair 都預測一次，對於 N 個

user 和 M 個 course，只需要使用 $O(N+M)$ 次模型而不是 $O(NM)$ 次，並且矩陣相乘相對於 DNN 模型快非常多，整體上預測並不需要太多時間。

缺點是相對於 collaborative filtering，他沒有充分使用群眾力量和交易資料，在 seen user 的方面表現可能相對差。另外，我們使用其他很簡單規則的 rule-based 技巧可能還會達到更高的 performance，像是統計熱門的免費課程然後所有人都推銷一樣的東西等等。

另外，也有一些未來的發展可以考量：

- 可以多探討 Encoding model 的架構。
  - 只有試過兩種，另一種更深更寬的 DNN autoencoder 的 performance 做了 270 epoch 也趕不過上述的模型，可能有 overfit 等等的情況。
- BERT。
  - 可以一起 fine-tune，但是 6 萬個 user，過一次 BERT 要 20 分鐘。會有訓練時間考量。
  - 可以取看看其他種 BERT，我選擇 hfl/chinese-roberta-wwm-ext 的原因是在過去的作業表現不錯。
- Hyperparameters
  - d_encode, d_hidden, w, p_dropout, lr, epoch，以及negative sample的比例等等都可以再搜尋。

Collaborative Filtering

Collaborative Filtering (CF) 基本想法是相似用戶的喜好會相同，假設用戶A 買了商品 a 和 b 並給了不錯的評價，且如果用戶 B 也買了商品 a，則推薦商品 b 給 B。

Matrix Factorization

CF裡最常用的技巧就是Matrix Factorization (MF)

令 $n_u = \text{number of users}$, $n_i = \text{number of items}$
我們會有一個 $n_u \times n_i$ 的input matrix R

$$R_{ui} = \text{rating of item i by user u}$$

我們在模型裡會將每個用戶以及每個商品各自encode成一個維度 k 的向量，並以用戶向量 u 與商品向量 i 的內積來當作預測的此用戶對該商品的評價

$$u^T i = \text{predicted rating of item i by the user u}$$

如果將所有用戶向量依序當作column vector排好則可以得到一個用戶矩陣 U, 同理我們也可以得到一個商品矩陣 I。將 U^T 和 I 相乘則可以得到一個$n_u \times n_i$的預測矩陣 P

$$P_{ui} = \text{predicted rating of item i by user u}$$

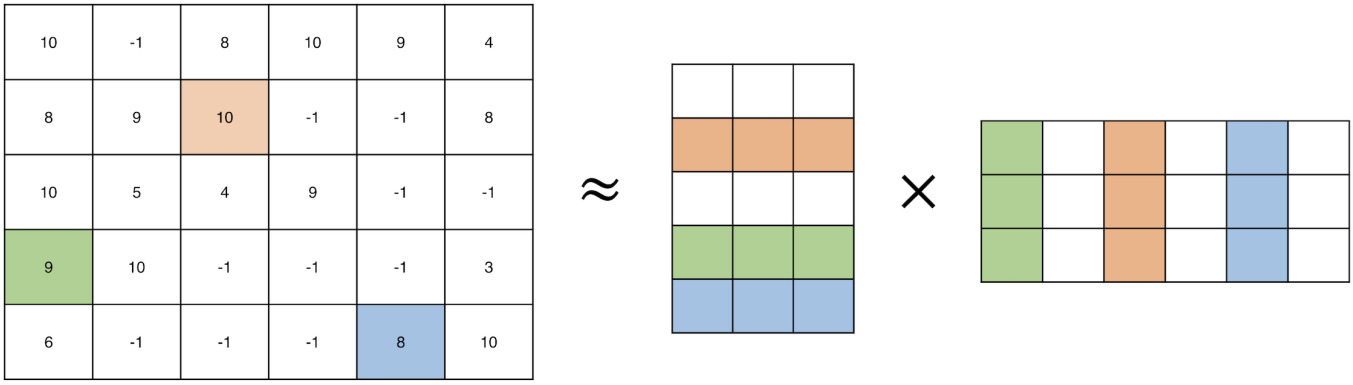


image source

Learning Algorithms

由於Hahow資料的提供的購買記錄是implicit data (即非用戶主動評價，而是用戶的一些客觀行動)，且無negative example(即不能確定一個用戶是否討厭某項商品)，傳統用於explicit data的CF方法皆不能使用(如 SVD)。

在這個部分我們使用了以下三種適用implicit data的learning algorithms來訓練模型

- Alternating Least Square (ALS)
- Bayesian Personalized Ranking (BPR)
- Logistic Matrix Factorization (LMF)

這三種learning algorithms皆屬於matrix factorization的範疇，差別在於loss function以及參數更新的設計。

Experiment

以下三個實驗以三種learning algorithms 來測試不同的 input matrix R 得到的 map@50。三個實驗裡每個algorithm的hyperparameters都是相同的

Hyperparameters

- ALS
  - factors = 1
  - epochs = 40
  - regularization factor = 0.01
- BPR
  - factors = 128
  - epochs = 50
  - regularization factor = 0.001
  - learning rate = 5e-5
- LMF

- factors = 10
- epochs = 40
- regularization factor = 0.6
- learning rate = 0.5

Exp1

首先我們需要建立輸入矩陣 R，由於Hahow的訓練集只有提供每個用戶買過的課程，最直覺的方法即是將買過的課程的 entry 設1，沒買過的設0。

$$R_{ui} = \begin{cases} 1 & \text{if user } u \text{ bought course } i \\ 0 & \text{otherwise} \end{cases}$$

algo	validation (x100)
ALS	5.37
BPR	5.24
LMF	2.1

Exp2

如果進一步思考，買過的課程在 R 裡的權重不應該都一樣，例如有些課程很熱門，那這些課程的權重應該高一點。

加上我們觀察到熱門課程主要分布於價錢低的區段，於是我們在 R 的設計上增加了一點 heuristic ：買過的課程裡，價格越低的使其權重越高。

$$R_{ui} = \begin{cases} \frac{1}{1+\text{price}(u)} & \text{if user } u \text{ bought course } i \\ 0 & \text{otherwise} \end{cases}$$

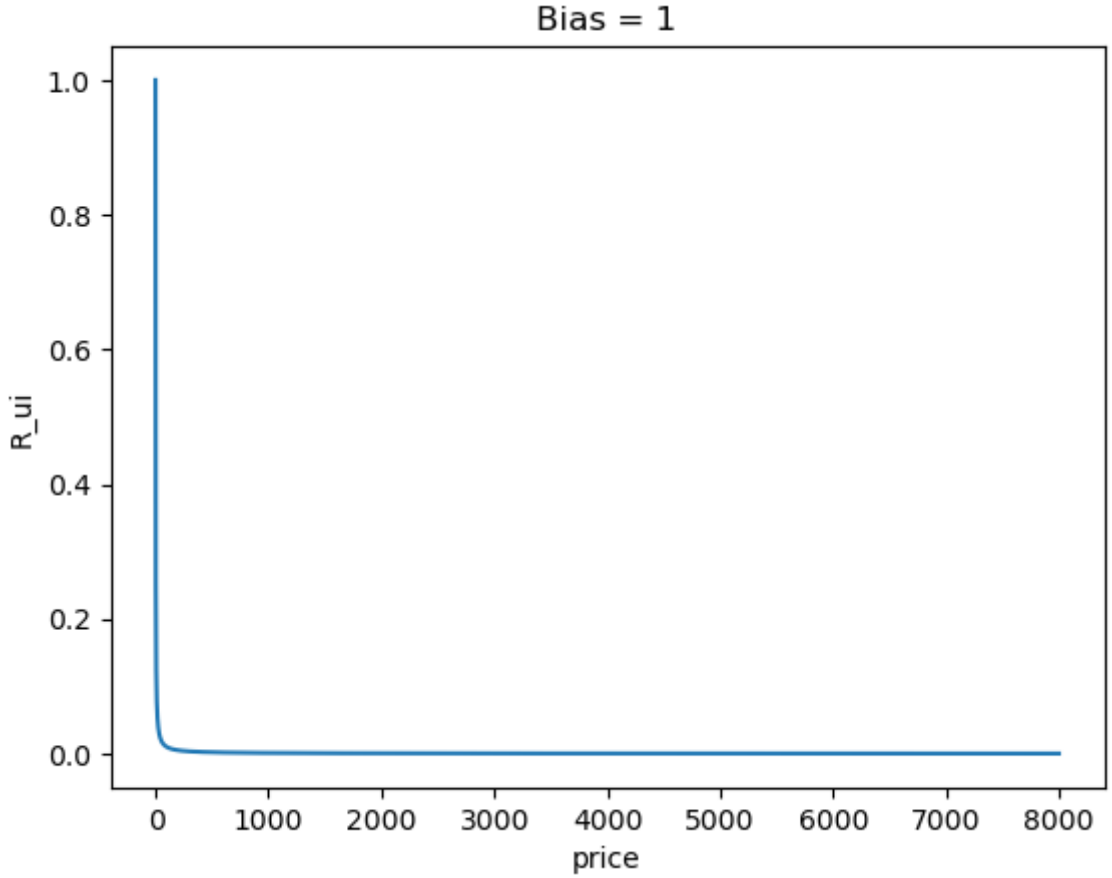
algo	validation (x100)	kaggle seen course public (x100)
ALS	5.47	3.709
BPR	5.27	3.191
LMF	4.41	3.42

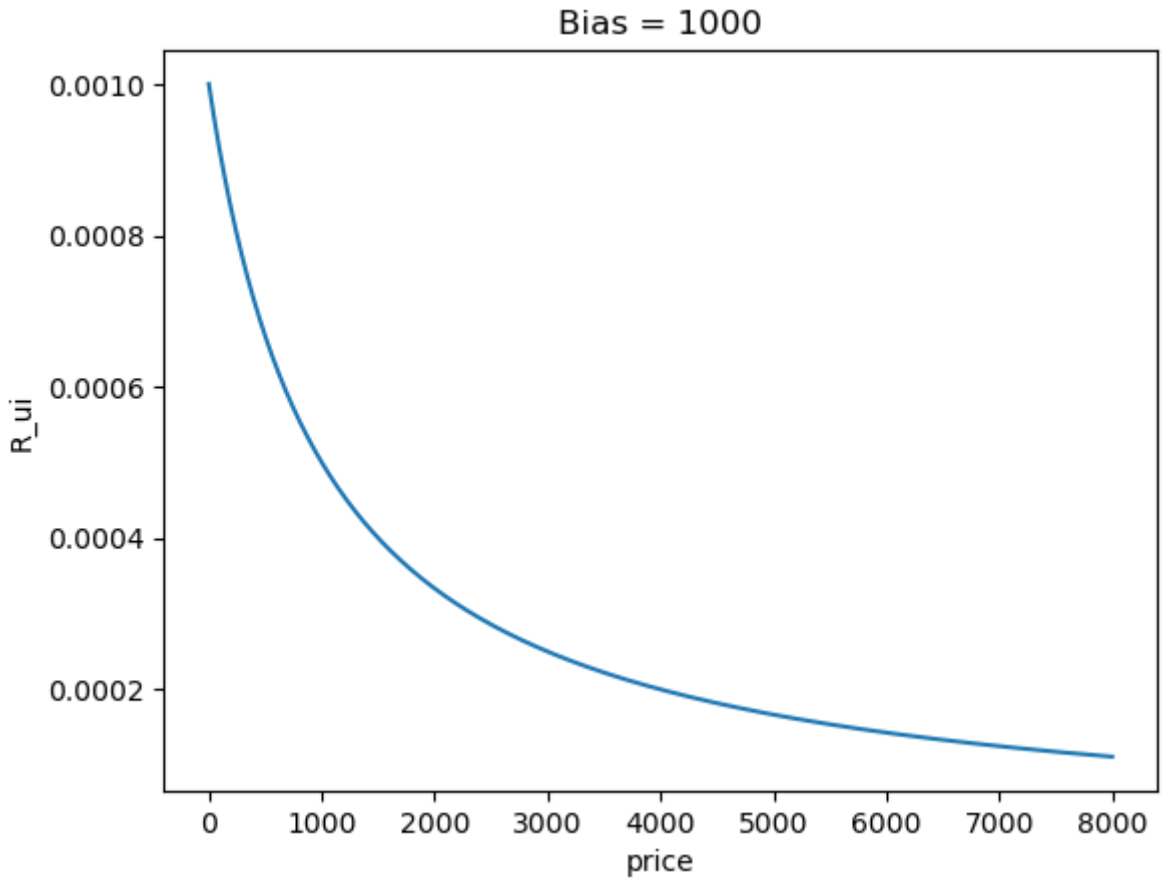
Exp3

經過一些參數調整後，發現將**Exp2**裡公式的 bias 調整為1000，可以得到更好的結果。

$$R_{ui} = \begin{cases} \frac{1}{1000+\text{price}(u)} & \text{if user } u \text{ bought course } i \\ 0 & \text{otherwise} \end{cases}$$

增加 bias 可以避免免費課程權重過大。





algo	validation (x100)	kaggle seen course public (x100)
ALS	6.52	4.339

BPR	5.27	N/A
LMF	5.37	N/A

Discussion

在這三個實驗裡，LMF 對於 表現進步幅度最大，原因是對於其演算法對於 R 裡的數字敏感度最大，在LMF的 loss function (Negative Log Likelihood) 裡他不是直接用 r_{ui} 裡當 ground truth，而是使用

$$c = \alpha r_{ui}$$

α是一個固定的常數，c 代表 confidence，我們對於該 r_{ui} 的信任程度。

根據heuristic，通常價格越低的課程消費者更願意購買，也就代表我們越信任這個課程是真的願意被消費者購買。於是加入 heuristic 後，LMF的分數得到大幅度的提升。

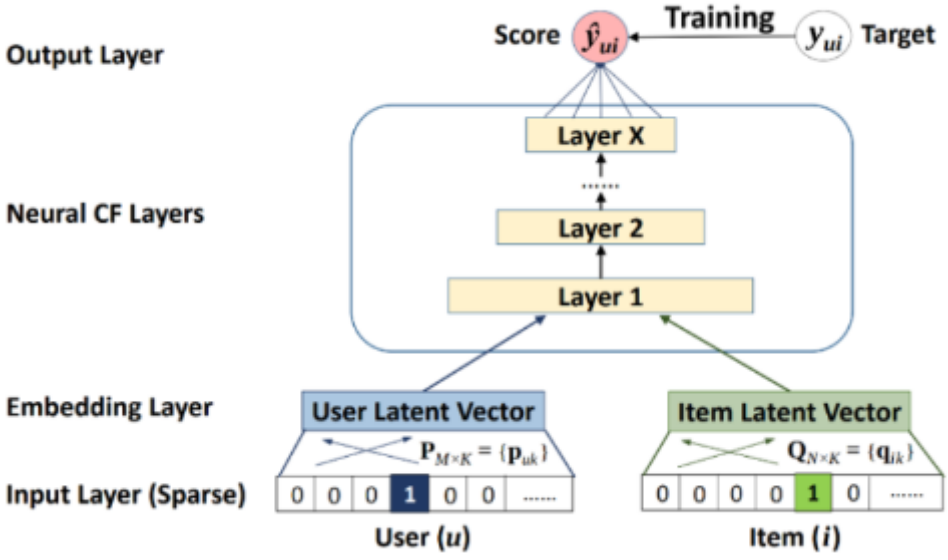
最後，從這些實驗我們也可以看出對於 implicit data，單純使用 CF 無法充分地學習到 user-user (or user-item) similarity。要得到更好的 performance，我們可以

- 使用 explicit data
- 增加經驗法則進 model 裡
- 使用更多商品和用戶的資訊，結合 neural network 來學習 user/item encodings

Neural Collaborative Filtering

Neural collaborative filtering (NCF) 是以collaborative filtering的概念為基礎衍生來的推薦系統模型。不同於傳統的collaborative filtering將user-item矩陣做分解的方法去學習，NCF通常會用簡單的類神經網路將user-item的相對資訊學起來，透過幾層的網路架構甚至可以去實現非線性的推薦，因此我們也打算試試看這樣的架構。

Model Architecture



Experiment

從上面的架構圖可以得知這樣的模型只知道user跟item的id的情況下，很難去有很好的表現，因此我們試驗了兩種不同的rating function來看看是否考慮其他資訊時會有較好的表現。
Type1: 只考慮使用者有無購買該課程

$$R_{ui} = \begin{cases} 1, & \text{if user u bought i'th course} \\ 0, & \text{otherwise} \end{cases}$$

Type2: 將購買次數高於1000的課程也列入考慮，並對於原本就購買的課程考慮其購買次數

$$R_{ui} = log_{10}(1 + freq(course_i))$$

Rating Function	Validation (*100)	Kaggle (*100)	Embedding Dimension	iterations (epochs)	learning rate
Type 1	0.65	0.87	32	5	0.001
Type 2	6.82	4.26	32	5	0.001

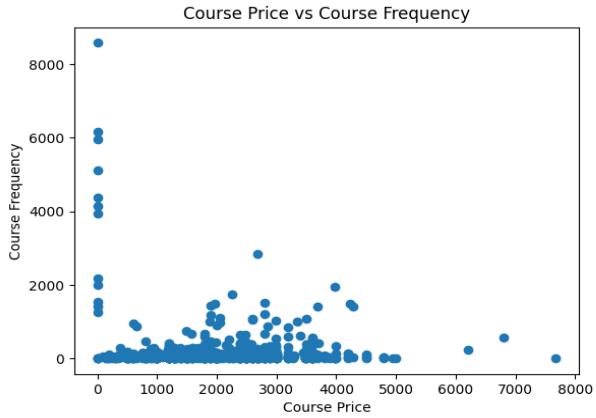
Discussion

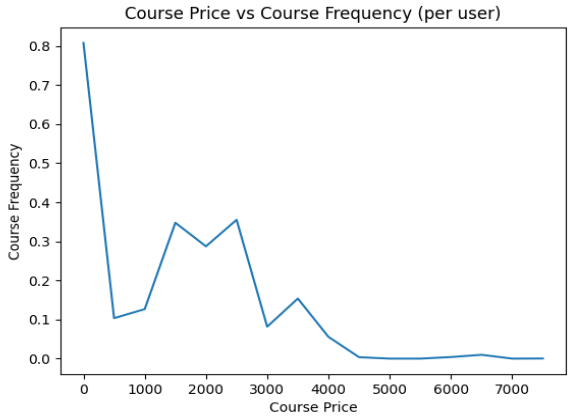
1. 這樣的架構非常好實作，也可以很好的學會user-item矩陣的關係，但是在實際要推薦課程時，單純只靠學好在訓練集的user-item矩陣是很難彈性地去推薦課程給使用者。
2. 因為NCF只考慮了user跟item的id，並硬把對應評分學起來，因此無法透過更深層的資訊去推薦課程，也就無法處理Unseen的情況。
3. 從表現上可以看到Type 2的表現很明顯比type 1來的更加出色，這也同時告訴我們只靠id是完全沒辦法在實務上有好表現的，我們只是單純多考慮了購買次數高的課程，表現就比原本好了十幾倍，因此或許之後能考慮更多資訊達到更好的表現。

Rule-based Recommendation

從之前Neural collaborative filtering和collaborative filtering的實驗中可以發現價格對於使用者購買意願有非常大的影響，因此我們將價格跟購買頻率做分析畫出以下兩張圖。從左邊那張圖可以明顯看出免費課程是非常熱門的，從右邊那張更可以發現購買次數大致上是隨著價格增加而減少的，尤其是價格為0~500這個區間時，幾乎每個使用者都至少會買這樣一門課程。

因此我們決定只推薦價格低及熱門的課程給使用者當作我們的baseline。





Experiment

Task	Validation (*100)	Kaggle (*100)
Seen Course	9.90	7.11
Seen Topic	17.42	20.06
Unseen Course	6.24	6.01
Unseen Topic	15.78	17.54

Discussion

1. 可以看到當考慮了這些資訊後，整體的表現甚至比大部分甚麼都沒考慮的深度學習或是傳統機器學習的方法來的更好，因此我們認為後續如果可以將這樣的資訊考慮進去這些比較近期的方法，可能可以讓這些方法的表現有大幅的進步。
2. 這樣個結果也進一步跟我們說在解決這些問題之前，如果是從raw data開始處理，應該要做好資料分析，資料處理再開始處理，這樣才能知道甚麼特徵是對表現是最有影響的。

Conclusion

在課程預測的任務上，以rule-based的方法取得了最好的成果，一則代表在訓練前我們需要先對訓練資料有足夠的了解，二則在未來模型的建構上，我們可以嘗試整合課程價錢的資訊到模型當中，或許能對模型表現有大幅增進。

Work Distribution

- B08902068 黃政穎
  - Two-tower model: search & implementation
- B08305056 劉羽芯
  - Behavior sequence Transformers: search & implementation
- R11922139 許洸誠
  - Collaborative Filtering: search & implementation
- R11922025
  - Neural collaborative filtering: search & implementation
  - Rule-based recommendation: search & implementation