R05631036 于允中
R06922077 鄭宇軒
R06631035 王凱陞

I. Pre-process

在資料預先處理部分，我們先針對 User-ID、ISBN 由 1 號開始，各自做了依序的編碼，重複的號碼視作相同編碼，即將 userID 轉換為 1-77805 之編碼，而 ISBN 轉換為 1-185973 之編碼。依照編碼依序將 book_ratings_train.csv 和 book_ratings_test.csv 轉換為新的檔案格式，分別為 new_trainData.csv 和 new_testData.csv，輸出 userID 和 bookID 作為 model 的 input feature。

除此之外，為了提取使用者和書籍的 feature，我們也另外將 users.csv 和 books.csv 內的資訊進行轉換，將我們需要使用到的幾個 feature 連同對應的編碼和原先的 ID 輸出成 corresponding_user_info.csv 和 corresponding_book_info.csv，以下表為兩者的格式和轉換方法:

A. corresponding_user_info.csv

在 user 部份我們所使用的 feature 有 age、location。Age 部份我們使用 users.csv 內提供的資訊，部分年齡資料有做修正，因平均年齡在 35 歲，故我們將小於 8 歲、大於 88 歲和年齡資料未填的使用者皆設定為 35 歲。

Location 的部分針對國家做處理，先計算其分佈，且取分佈有超過 0.04 的國家做排序處理。未超過 0.04 分布的國家和 0.016453%沒有填寫國家的使用者我們將其 location 設定為 other。如此處理後，我們將 location 分為七個類別，分別代表七個國家，其分佈如下:

Location	Percentage
USA	0.501015
Canada	0.077667
UK	0.066543
Germany	0.061225
Spain	0.047544
Australia	0.042262
Italy	0.040418
Other	0.1643

將重新編碼過的資料依序排列，並依序對應其原始 userID、年齡、所處國家位置和原始 userID 在 user.csv 中的 index，最後輸出成新的檔案格式 corresponding_user_info.csv 如下表所示。

new_userID	userID	age	location	users_csv_index
資料排序 (1-77805)	原編碼	年齡	重新編碼的地區碼	該資料對應 users.csv 的 index

B. corresponding_book_info.csv

在 books 部份我們使用的 feature 有出版年代(year)和重新編碼的書籍分類(category)，因為平均年分在 1993.689(不取整數)，若是書籍年分未訂(未在 book.csv 中出現)，則將其設定為 1993.689。將各 feature 整理後如 user_info 般依序排列輸出成新的檔案格式 corresponding_book_info.csv，格式如下表所示。

R05631036 于允中
R06922077 鄭宇軒
R06631035 王凱陞

new_bookID	ISBN	year	category	original_idx
資料排序 (1-185973)	國際書碼	出版年代	重新編碼的書籍分類	該資料對應 books.csv 的 index

此部分比較特別的是 category 的部分，我們運用了 tf-idf 的做法進行書籍的分類，實際方法如下:

利用 tf-idf 值從 book description 中找出具有代表性的字將每本書轉換成向量，其中向量的元素表示出現過幾次對應的代表字。
例如：假設具有代表性的字是 science, art, bag，統計每本書出現次數如下

	Science	Art	Bag
Book 1	5	2	7
Book 2	1	8	3

則 Book 1 = (5, 2, 7)而 Book 2 = (1, 8, 3)
我們所找到具有代表性的字是['author', 'beauti', 'book', 'come', 'famili', 'help', 'includ', 'just', 'life', 'like', 'live', 'love', 'make', 'man', 'new', 'onli', 'power', 'stori', 'time', 'way', 'woman', 'work', 'world', 'year', 'young']，有些字像是不完整的單字是因為還經過 stemming 處理，將不同型態但是同一字源的字視為相同的字。

將每本書轉換成對應的向量後，再利用 KMeans 將全部的書分類，KMeans 需要填入「期望的類別數」，經過幾次測試後選定 7 類作為最終選擇。而有些 training data 或 testing data 的書沒有出現在 books.csv 裡面，我們將這些書分在第 8 類，最後就得到所有書對應的分類了。
(參考來源: http://brandonrose.org/clustering)

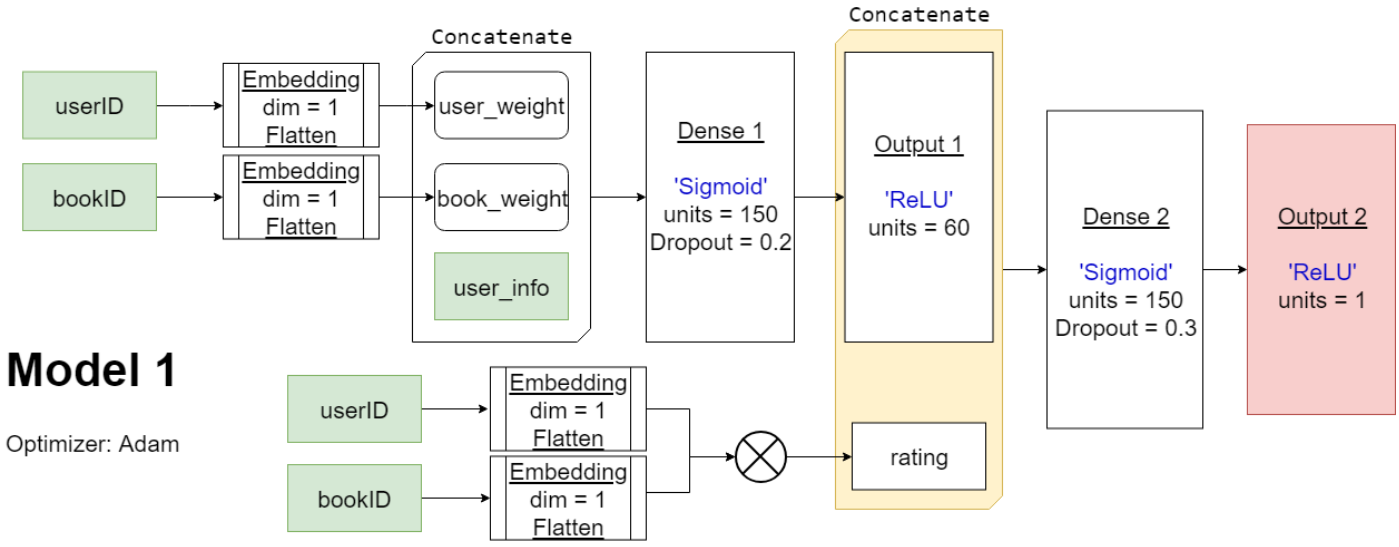
最後，根據 corresponding_user_info.csv 和 corresponding_book_info.csv，我們對於 training 和 test 資料新增了兩個 input feature: user_info 和 book_info。

II. Approach:

以下為我們所使用的三種主要架構方法和一些額外的輔助方法，以增加 leaderboard 的表現。

R05631036 于允中
R06922077 鄭宇軒
R06631035 王凱陞

1. Approach 1:



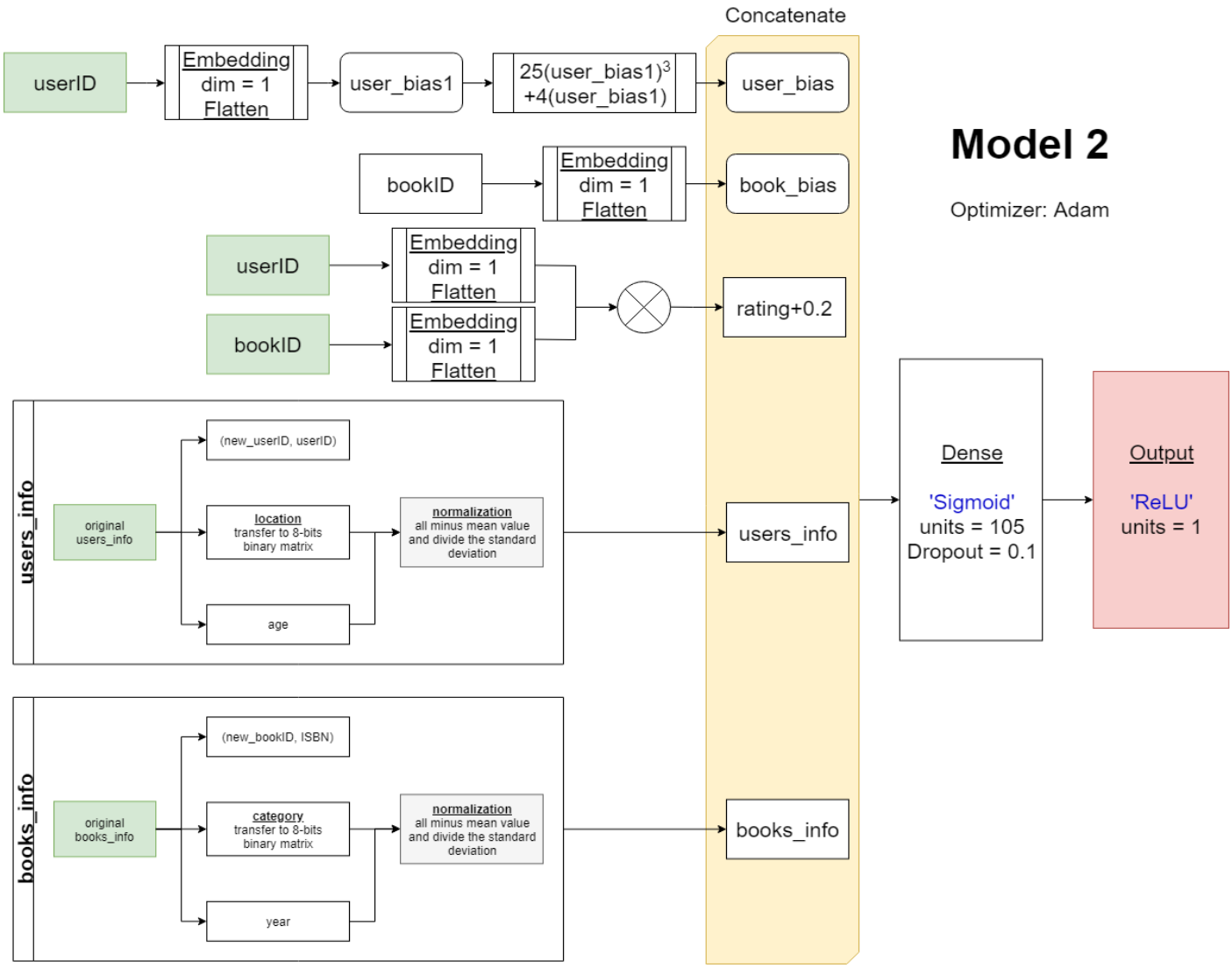
模型一架構主要是覺得 userID、bookID、user_info 和 book_info 間存在某種相關性，所以先針對此四個輸入做一層神經網路訓練，得到 output1。接著透過上課所說明的 matrix factorization 方式，將 userID 和 bookID 的 pointwise 乘積視為另外一種輸入 feature(rating，也就是 userID 和 bookID 的一種初步評分)，透過 rating 和 output1 做第二次的神經網路訓練而得到最終評分。值得一提的是經過測試之後加入 book_info 於第一個 Conatenate layer 中，反而會稍微降低 validation 和 public score 的表現，故在這個 model 中我們並沒有選擇加入 book_info。

此模型中可以選擇的參數有: embedding dimension、各 Dense 層的 activation function 和 dimenstion、output1 的 activation function 和 dimenstion 以及兩個 Dense 層間的 Dropout 比例。

i. embedding dimension: 從 0~30 進行測試，發現越小越好，故選 1
ii. Dense 層的 activation function: 在"tanh"、"sigmoid"、"Relu"、"linear"間進行測試，得出最佳組合。
iii. Dense 層的 dimenstion: 從2^0~2^8間的數字進行微調比較，發現在 150 的時候會有較好表現
iv. output1 的 activation function 和 dimenstion: 與 Dense 層的測試相同
v. Dense 層間的 Dropout 比例: 比較從 0.1~0.5 間的小數值，得到最佳比例組合。我們將 Dropout 設定為 0.2 除了測試後得到效果較佳以外，我們也認為這樣的設定可以保留足夠的 user_info 內 feature nodes 的權重，故在測試期間，我們所選擇的設定都為 Dense 1 的 Dropout 會比 Dense 2 小一些

R05631036 于允中
R06922077 鄭宇軒
R06631035 王凱陞

2. Approach 2



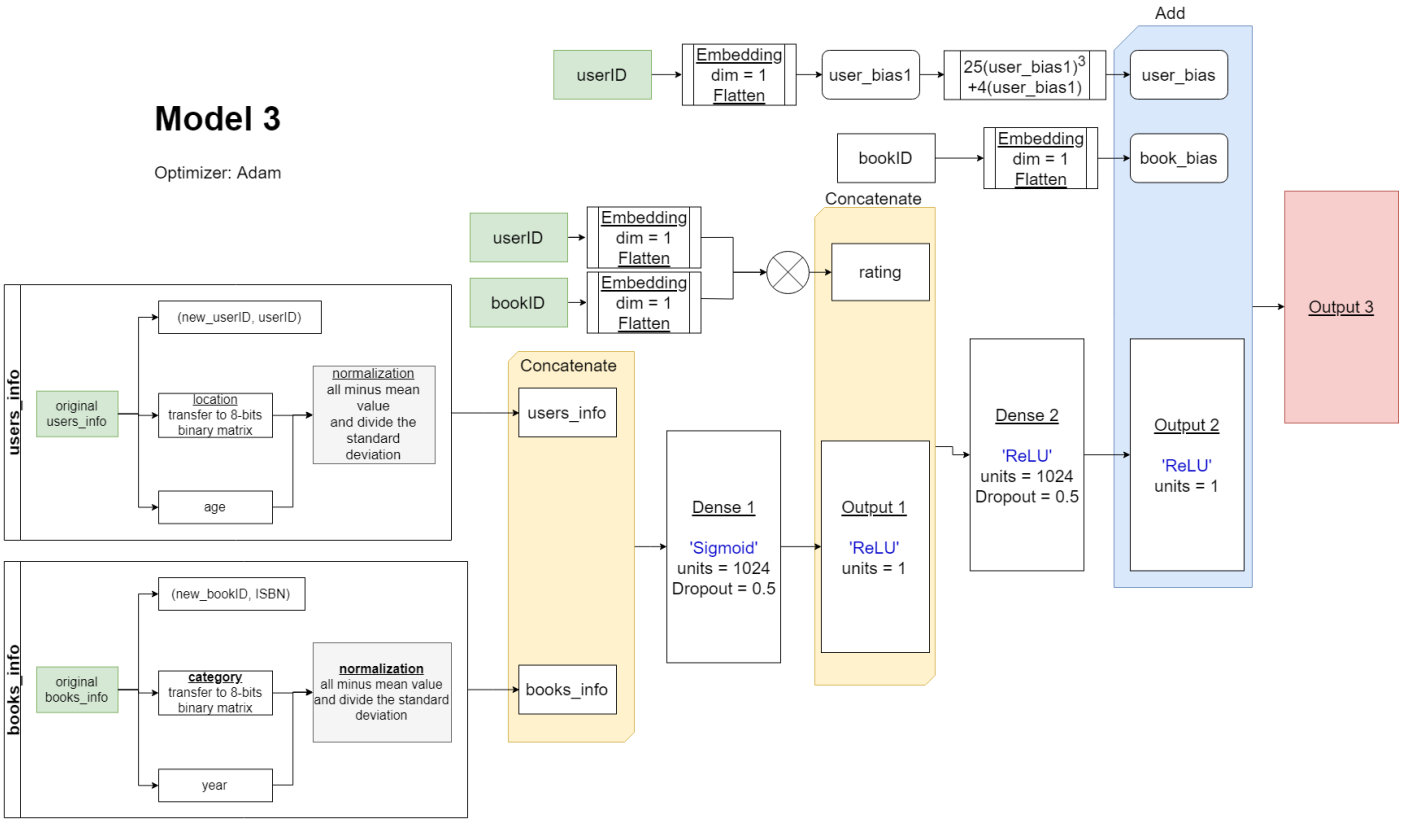
模型二把所有已有的 feature、matrix factorization、IDs 和 information 丟入一層隱藏層中，取資料中的 10%作為 validation set 做驗證，在利用 validation set 去針對每一個 feature 作係數和截距調整。此模型中可以選擇的參數和模型一中的大同小異，決定參數的步驟也差不多，故就不再重複撰述。

在圖中 normalize 部分是針對 book 和 user 的輸入資訊做歸一化。其作法以 user 資料舉例是在 users_info 中，先將 pre-processing 的 location 資料作 one-hot encoding，再和 age 一起作 normalization。normalization 方法是將所有資料中的同一 feature 值都減去其平均值，並除上標準差，這樣做平均值就會為 0 且標準差為 1，這樣做將 age 和 one-hot encoding(location)的權重達到近乎一致，避免 age 的值因為過大而將 one-hot encoding(location)的結果吃掉。同理，books_info 的部分也採取和 users_info 類似的作法，先將 category 進行 one-hot encoding，並和 year 一起作 normalization。

在 feature 係數調整上，我們嘗試不同係數組合在 validation set 上的進步幅度，首先在 user_bias 上進行雙倍，發現有不錯的結果，進而測試次方，在多方嘗試後測得最佳係數。最終我們將 user_bias 的部分調整成為圖上所示之多項式，rating 的部分所加上的常數項 0.2，而 book_bias 的部分則是都沒有調整即會有最好的結果。有趣的是這項特性似乎在其他兩個 Model 上不太適用。

R05631036 于允中
R06922077 鄭宇軒
R06631035 王凱陞

3. Approach 3



模型三覺得 user_info 和 book_info 間存在某種相關性，所以先將 book_info 和 user_info 先丟入一層隱藏層中，得到 output1，後與 matrix factorization 的結果 rating 結合後進行再作一層的訓練，最後與原始的 userID 和 bookID 組合成為最後評分。

此模型的架構是依據前兩個模型所得到的結果進行進一步的優化，我們認為針對 users_info 和 books_info，在模型一中作的並不完整，但將 users 和 books 的一些特性資料作第一層的訓練確實得到了不錯的效果，所以在 book_info 和 user_info 部分先做第一層的訓練。

而在第二次訓練時之所以設定這層的 activation function 為 ReLU，除了因為使用其他 activation function 的效果較差外，另一方面是想作線性組合的計算，放大這部分大於 0 的權重。

此模型同樣取資料中的 10%作為 validation set 做驗證，其中其他可以選擇的參數和模型一中的大同小異，決定參數的步驟也差不多，故就不再重複撰述。

4. Additional Approaches

i. Bagging:

一開始我們將資料作 bootstrapping，將資料量放大七倍，測試結果有稍微變好，但是卻反而將原先我們所測試出來的最佳 epoch 數降低，甚至出現 epoch=1 時，就達到最好的情形，我們認為這部分在 tune 參數上會降低很多自由度和準確性，所以最後改為下面的做法。

ii. Shuffling and Voting:

因上述的 bootstrapping 問題，我們改利用 NN 本身的所帶有的隨機性質，將資料作隨機得 shuffle 300 次，並將這 300 次的結果進行投票，此做法可以保留原本參數調整上的自由度和準確性，且亦可利用 NN 本身在訓練上的隨機性，得到類似 bagging 的效果，經測試確實在 leaderboard 上有不錯的效果。

iii. 神奇的 0.4 參數:

針對 track 1，需要上傳的結果必須是 integer，我們透過改動了原先四捨五入的作法，將資料設定為只要小數點第一位超過 0.4 即無條件進位，藉此我們也發現可以提升預測的準確度。

iv. Implicit 資料:

利用已經 train 好的 Model，對 implicit data 作預測，為維護資料的正確性，只留下和整數相差 ±0.01的資料，並將這些資料和 training data 再作一次訓練，得到最終結果，但是經測試之後此方法對於最後結果並沒有有效得提升，甚至有些降低，所以再最後並未採用。

III. Comparison

根據神經網路架構理論，我們共組合了三種神經網路模型，在前一章有詳細的參數比較和介紹。三者模型差異主要是由於我們認為那些 feature 間會有相應關係而進行多層的神經網路訓練。三個模型的表現結果如下表所示

Model	Batch size	Epoch	Training stage		Validation stage		Test stage	
-	-	-	MAE	MAPE	MAE	MAPE	MAE	MAPE
Model1	512	3	1.235	20.8	1.257	22.26	1.234	22.46
Model2	1126	7	1.11	17.448	1.246	21.831	1.210	21.889
Model3	1126	15	1.0604	17.95	1.2457	21.86	1.208	21.833

根據不同模型我們選擇表現最好的 batch size 和 epoch 的組合，並查看其表現結果。兩種 performance index 分別為 MAE 和 MAPE，其中 MAPE 的單位為%。Test 資料是將預測結果上傳到 scoreboard 並以 public score 作為指標。由結果我們可得知，模型三在訓練和驗證時 MAE 的表現皆最佳，在 test 時也表現最好；模型三對 MAPE 的表現在訓練和驗證時皆弱於模型二，但相差不大，反而在 test 時表現比模型二較好。綜合上述所說，本應以模型三作為 track1 和 track2 的最終模型，但由於模型二在 track1 中做完 bagging 後 test 表現會優於模型三，也就是 scoreboard 上的最終結果，而模型三在做完 bagging 後表現沒有變好，故最終結果是在 track1 中使用 bagging 完的模型二，在 track2 中使用模型 3。

針對效率部分，Model 1 是最快但是效果沒有其他二者好，在未加入 bagging 前，Model 2 的 training 過程需要 16 秒，Model 3 卻需要 140 秒，在考慮 testing 表現的情況下，運算效率上明顯 Model 2 效能較好，若是在需要即時預測資料的情境，Model 2 會是更好的選擇。

IV. Conclusion

根據三種架構的神經網路模型進行 MAE 和 MAPE 的測試，在尚未 bagging 的條件下，單就結果模型三的表現最好，即是說明 book_info 和 user_info 間可以透過訓練找出隱藏的特性，進而加強訓練出來的結果。而模型二因單純將 book_info、user_info、user_bias、book_bias 和初步 rating 進行訓練，而可能忽略掉了部分特性，但相較換來的是運算速度上的提升。而模型一則可能是因為 user_info、user_bias、book_bias 間的關係不明確，所以以第一層訓練的結果運行第二次訓練效果反而的沒有模型二和三好。或許嘗試將不同的 input feature 間的組合先進行訓練，在 merge 後進行第二次訓練可能還會有更不一樣地的效果。