ML2018SPRING HW6

B05902042 林瑋毅

1.請比較有無normalize的差別。並說明如何normalize.

Normalize 的方法：計算所有 training data 中 rating 的平均與標準差，並將 training data 中的 rating 統一減去平均並除以標準差，再拿去訓練；預測時則先乘以標準差再加上平均。 (平均 = 3.5817120860388076、標準差 = 1.116898281728514)

有 Normalize：private / public = 0.85777 / 0.86388
無 Normalize：private / public = 0.87714 / 0.88569

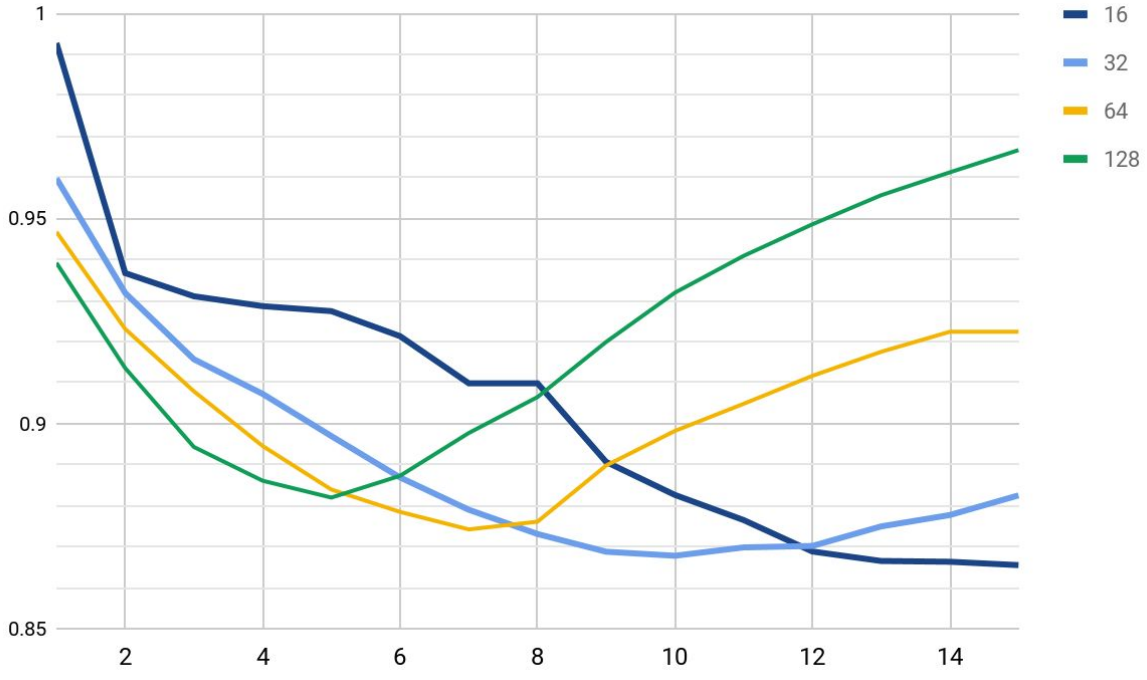
可知資料經 normalize 後，對 optimizer 來說，數值上比較穩定，所以會有比較好的結果。

此題的 embedding 層的維度皆為 64，並且使用 xavier uniform (Pytorch 預設參數) 來初始化權重，不使用 bias 項，其中 optimizer 為 Adam(lr=1e-3, weight_decay=2e-5)，batch 大小為 128，並根據 validation 的結果選擇 epoch 數：有 Normalize 為 5，無 Normalize 為 7。

PS 此題有使用 numpy.clip 將預測的數值限制到區間 [1,5]。

2.比較不同的embedding dimension的結果。

嘗試 16, 32, 64, 128 維的 embedding dimension，下圖為不同維度的 learning curve ，縱軸是 validation set 上的 root mean square error



可知只要維度太大都會導致 overfit，若不 normalize 資料，最佳維度為 16。

此題的 embedding 層使用 xavier uniform (Pytorch 預設參數) 來初始化權重，不使用 bias 項，其中 optimizer 為 Adam(lr=1e-3, weight_decay=1e-5)，batch 大小為 128，不對資料做 normalize。

PS 此題有使用 numpy.clip 將預測的數值限制到區間 [1,5]。

3.比較有無bias的結果。

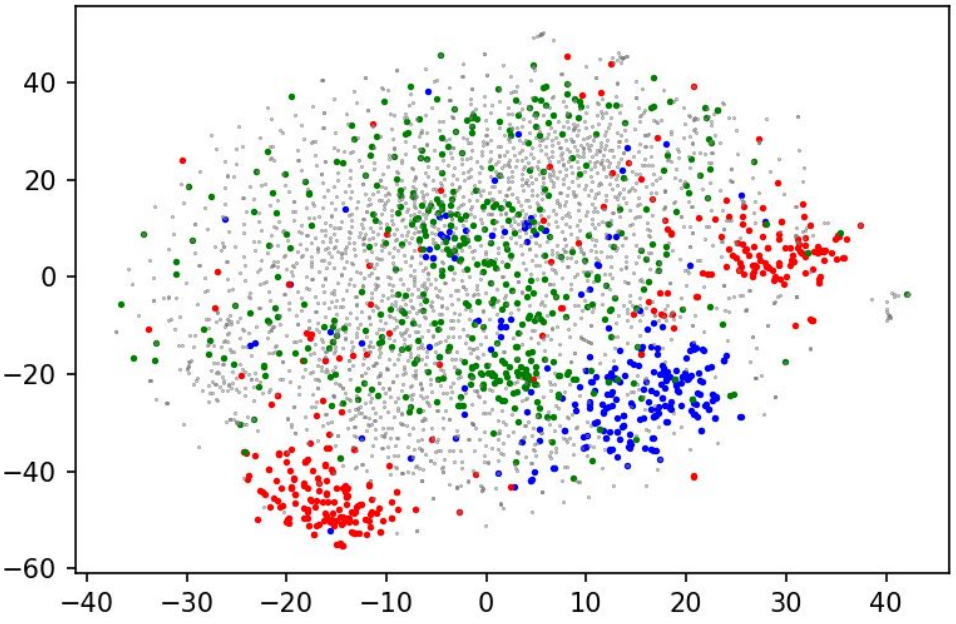
有 bias：private / public = 0.86288 / 0.86884
無 bias：private / public = 0.86684 / 0.87399

發現加入 bias 可以些微地提高準確率，bias 項可以讓 model 進一步捕捉各 user 打分的 bias 和大家對各 movie 評分時的 bias，所以會有較好的結果。

此題的 embedding 層的維度皆為 16，並且使用 xavier uniform (Pytorch 預設參數) 來初始化權重，不使用 bias 項，其中 optimizer 為 Adam(lr=1e-3, weight_decay=1e-5)，batch 大小為 128，並根據 validation 的結果選擇 epoch 數：有 bias 為 13 ，無 bias 為 16，不對資料做 normalize。

4.請試著將movie的embedding用tsne降維後，將movie category當作label來作圖。

上圖即將 movie 在 embedding 後用 tsne 降維的結果，其中藍點是 Animation 或 Children's、紅點是 Horror、綠點是 Romance、灰點則是不屬於這三類的其他電影，可見相同類別的電影在 latent space 上距離較近，不同類別的電影則距離較遠。



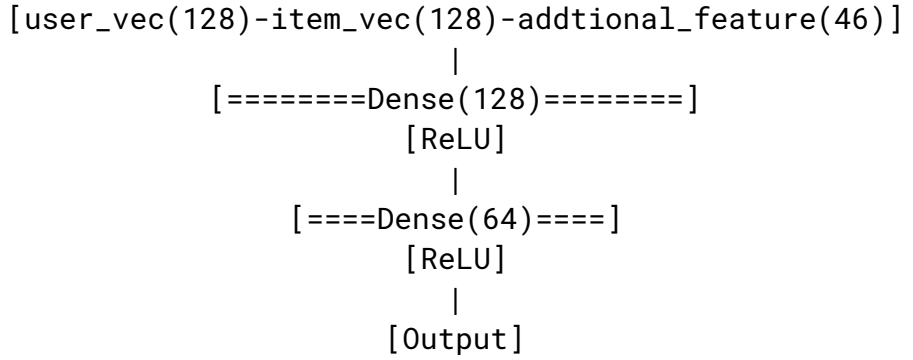
5.試著使用除了rating以外的feature, 並說明你的作法和結果，結果好壞不會影響評分。

首先，使用的 user features 包含性別、年齡、職業， movie features 包含類別。分別將以上的 feature 做 one-hot encoding，形成一個 (user_id, movie_id) 的 additional feature。

模型如下：

```
DenseNet(
        (user_embedding): Embedding(6041, 128)
        (item_embedding): Embedding(3953, 128)
        (fc): Sequential(
                (0): Linear(in_features=304, out_features=128)
                (1): ReLU()
                (2): Linear(in_features=128, out_features=64)
                (3): ReLU()
                (4): Linear(in_features=64, out_features=1)
        )
)
```

請注意，得到 user_embedding 和 item_embedding 的向量之後，再將他們以及前面所述的 addtional feature 連接起來，再經過三層的 DNN，最後輸出結果，如下所示



訓練時使用 xavier uniform (Pytorch 預設參數) 來初始化 embedding 層權重，其中 optimizer 為 Adam(lr=1e-3, weight_decay=1e-5)，batch 大小為128，並根據 validation 的結果選擇 epoch 數：5，有對資料做 normalize。

模型表現如下：

private / public = 0.85363 / 0.86145