Web Retrieval and Mining-程式 HW2　　鄭凱文 R08922015

0. 實作語言

Python3, with pytorch, numpy, sklearn, pandas… packages

1. MF with BCE

(實作細節可參考 model.MF, utils.train, utils.MFDataset)

Parameters:

Hidden factor d = 256

利用 PyTorch nn.Embedding，設置 User factor & Item factor

Loss function:

直接使用 PyTorch BCEWithLogitsLoss()

L2 normalization: weight_decay = 0.005

Negative sampling method:

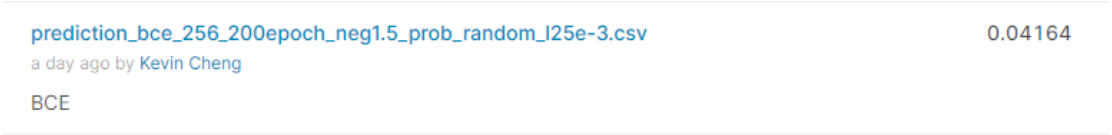
Negative sampling ratio = 1

先過濾出每個 User 沒有互動過的 Item list

根據 Negative sampling ratio，加入等倍數的 Positive item 到 Dataset 中，例如這裡 Ratio = 1 就加入所有的 Positive item、每個共一組，如果 Ratio = 2 則每個 Positive item 會在 Dataset 中出現兩次。

利用 Dataloader，每次 Load batch data 時，對於每個 Positive item 隨機挑選一個作為 Negative item，與之配成對進行訓練。

Highest MAP on Kaggle public: 0.04164



2. MF with BPR

(實作細節可參考 model.MF, utils.train, utils.MFDataset)

Parameters:

Hidden factor d = 256

利用 PyTorch nn.Embedding，設置 User factor & Item factor

Loss function: 使用自己寫的 BPR loss，對於每組 Positive pair 跟 Negative pair 各做一次 MF，再將送進 BPR Loss function。Function 嘗試過以下兩種版本：

- F.logsigmoid((pos_scores - neg_scores))

(1 - torch.sigmoid(pos_scores - neg_scores))

兩者基本上同樣概念，Sigmoid 的最大值就是 1，只有在 Positive 的分數遠大於 Negative，其值才會趨近 1，只是一個利用 Log(1) = 0、一個則是直

接用 1 去減的概念，而我認為後者比較直觀，因此採用後者。

L2 normalization: weight_decay = 0.005

Negative sampling method:

同 BCE version, Negative sampling ratio = 3

Highest MAP on Kaggle public: 0.05094

bpr_dim_256_trainNeg_3.0_epo120.csv 0.05094

4 hours ago by Kevin Cheng

Experiment BPR ratio=3.0

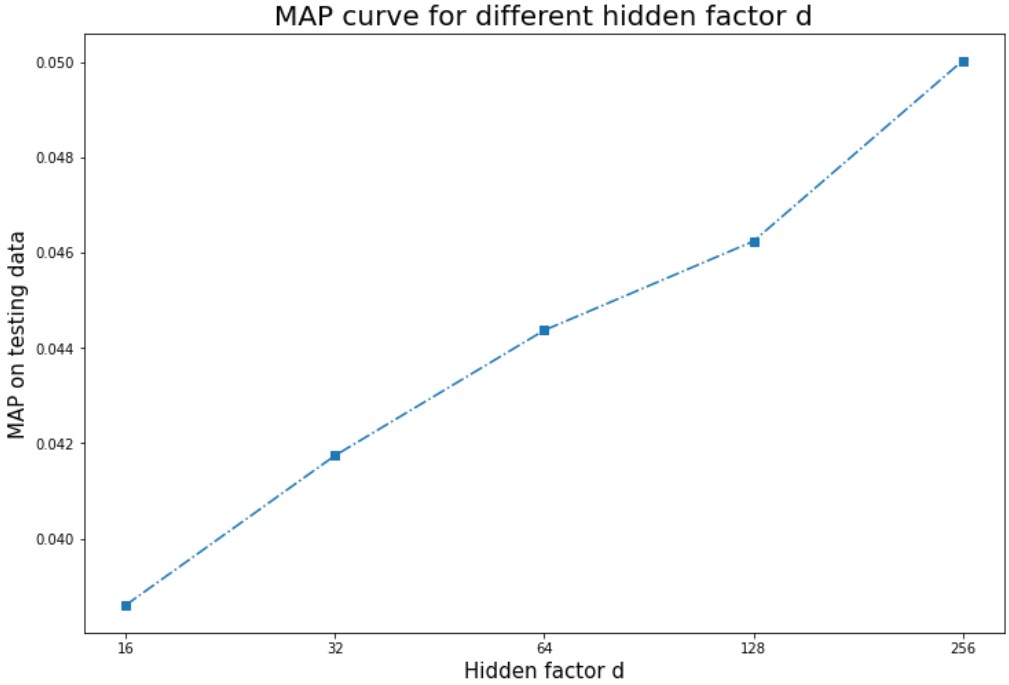
3. Compare MF with BCE and MR with BPR

從 MAP 可以看到，同樣的設定下，BPR 能比 BCE 高出將近 1%，顯示出將其視為 Ranking task、表現會比視為 Classification task 來得好，原因我認為主要在 Classification 時，所有沒被互動過的 Negative items，全部視為 0、一致往 Value 0 接近，但事實上有些 Negative 可能是潛在的 Positive items，他們的分數不該視為一致。

而 Ranking 的作法則是將目標換成盡量拉大 Positive 與 Negative 的距離，因此雖然皆朝著降低分數的方向，但移動距離取決於 Item 間的交互影響，這種方法下不僅能維持 Positive pair 的高分，也更能分辨出 Negative items 間的分數高低，達到我們想要的 Ranking 結果。

4. MAP curve for different hidden factor

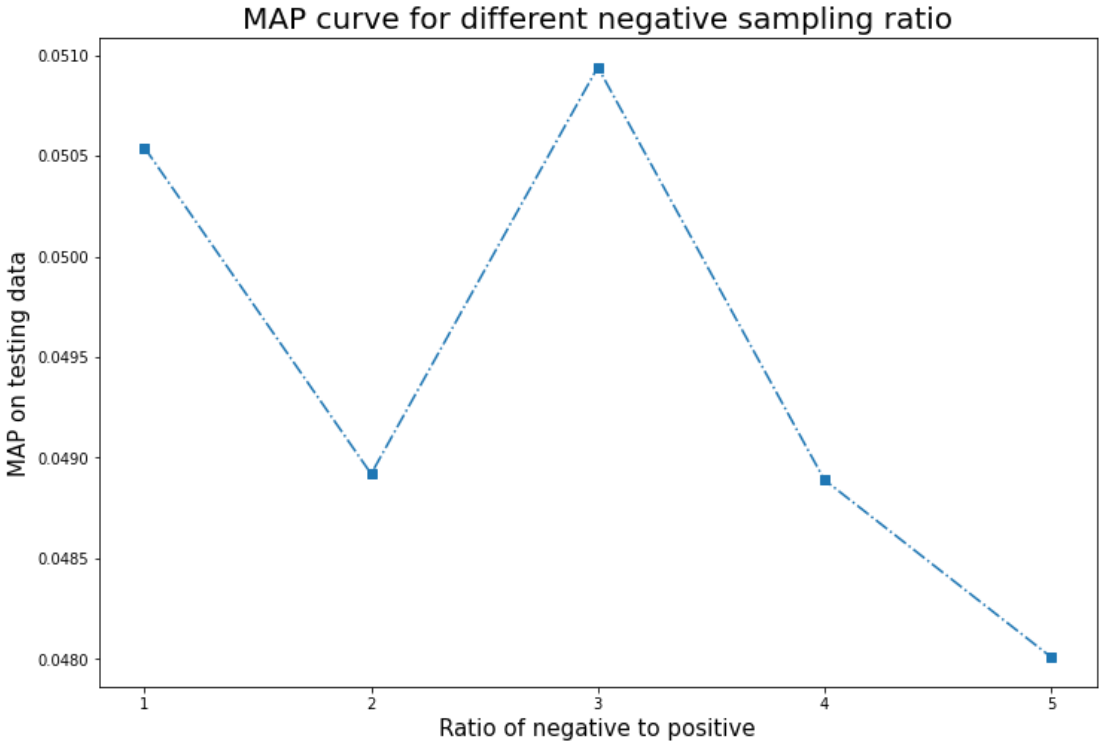
這裡一共嘗試 d = 16, 32, 64, 128, 256 等五種不同的 Hidden factor，使用在 Kaggle 上得分最高的參數設定 (BPR loss, weight_decay=5e-3, negative sampling ratio=1.0)，於 Kaggle public 上的得分。結果如下圖：



從圖可以清楚看到，Hidden factor size 愈大，MAP 也愈高，原因是較大的 Hidden factor 代表能保留較多維的資訊，因此更能體現完整的 Data。

5. Bonus – Influence of negative sampling ratio

這裡一共嘗試 Ratio = 1, 2, 3 等共三種 Negative sampling ratio，與第 4 題一樣使用 Kaggle public 最高分數下的參數設定。結果如下圖：



由圖可知，Ratio = 3 時最高、Ratio = 5 最低，在 Ratio 1~3 其實分數有所震盪，而且差距非常小，從趨勢來看則隨著 Ratio 慢慢調大，分數有下滑的傾向，我認為是因為 BPR 是透過拉開與 Positive 的距離，以此讓 Negative 間也有鑑別度，如果一次 Sample 的 Negative item 過多，則意味著愈多 Item 被視為等同相關度，自然對於鑑別上會有影響。

除了分數上的高低，訓練過程中當 Ratio 愈高，也較快收斂，這點符合預期，因為一個 Epoch 訓練的次數較多，但整體花費時間卻也差不多，因此我認為在本次實驗中，不須採取過大的 Negative sampling ratio。

6. Reference

https://blog.fastforwardlabs.com/2018/04/10/pytorch-for-recommenders-101.html

https://www.ethanrosenthal.com/2017/06/20/matrix-factorization-in-pytorch/