Web Retrieval and Mining-程式HW2 鄭凱文 R08922015

1. **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:

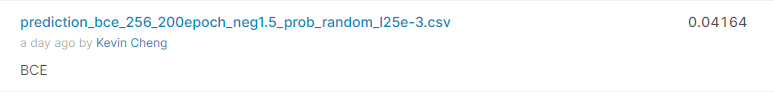
Negative sampling ratio = 1

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



1. **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))

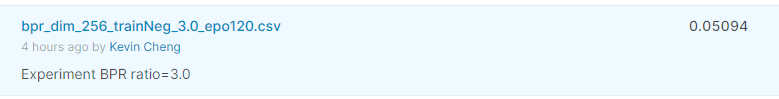
兩者基本上同樣概念，Sigmoid的最大值就是1，只有在Positive的分數遠大於Negative，其值才會趨近1，只是一個利用Log(1) = 0、一個則是直接用1去減的概念，而我認為後者比較直觀，因此採用後者。

L2 normalization: weight\_decay = 0.005

Negative sampling method:

同BCE version, Negative sampling ratio = 3

Highest MAP on Kaggle public: 0.05094



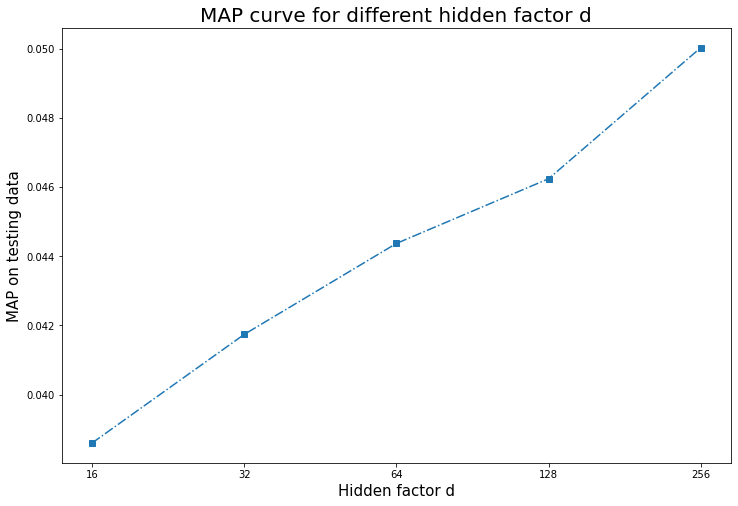
1. **Compare MF with BCE and MR with BPR**

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。

1. **MAP curve for different hidden factor**

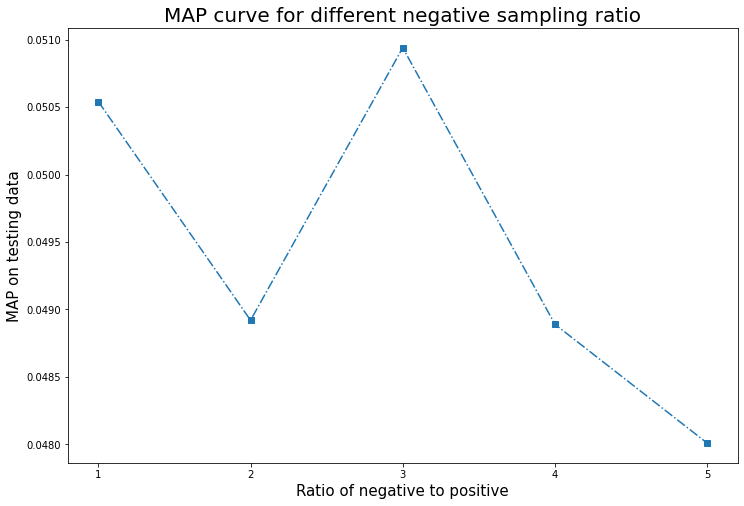
這裡一共嘗試 d = 16, 32, 64, 128, 256等五種不同的Hidden factor，使用在Kaggle上得分最高的參數設定 (BPR loss, weight\_decay=5e-3, negative sampling ratio=1.0)，於Kaggle public上的得分。結果如下圖：



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。

1. **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。

1. **Reference**

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

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