IR HW2 Report

B05902010 資工四 張頌平

Q1 : Describe your MF with BCE (e.g. parameters, loss function, negative sample method and MAP score on Kaggle public scoreboard)

Ans:
我實作的方法是，在每一個batch當中，對於當中的user都會有相同數量的positive & negative的item，利用其user和不同item的embedding的內積去計算BCE loss，這部分pytorch有套件幫助我們實作計算loss的過程，我們只需要將positive和negative items的target設定為1和0，並且跟embedding內積的結果一起塞進函式 ***torch.nn.BCEWithLogits*** 即可， 詳細結果如下:

Settings:
negative sample ratio: 1:5
hidden size: 64
optimizer: SGD
initial learning rate: 1
batch size: 64
weight decay: 0
scheduler method: Multiply 0.2 if valid loss not improved every 3 epochs
train/test ratio: 0.9: 0.1
training epoch: 50
embedding normalize method: orthogonal
Kaggle public score: 0.04793

Q2 : Describe your MF with BPR (e.g. parameters, loss function, negative sample method and MAP score on Kaggle public scoreboard)

Ans:
我實作的方法是，在每一個batch當中，對於當中的user都會有相同數量的positive & negative的item，利用其user和不同item的embedding去計算BPR loss，BPR loss算法如同作業投影片，將positive的item embedding和user embedding內積減去negative的item embedding和user embedding的內積，並經過sigmoid和log後加總，注意是因為我們希望要minimize我們的loss，因此我們會將上面結果取負號，最後利用pytorch 的套件幫助我們更新模型的參數，詳細設定和結果如下:

Settings:
negative sample ratio: 1:5
hidden size: 64
optimizer: SGD
initial learning rate: 1e-2
batch size: 64
weight decay: 1e-4
scheduler method: Multiply 0.2 if valid loss not improved every 3 epochs
train/test ratio: 0.9: 0.1
training epoch: 50
embedding normalize method: orthogonal
Kaggle public score: 0.05057

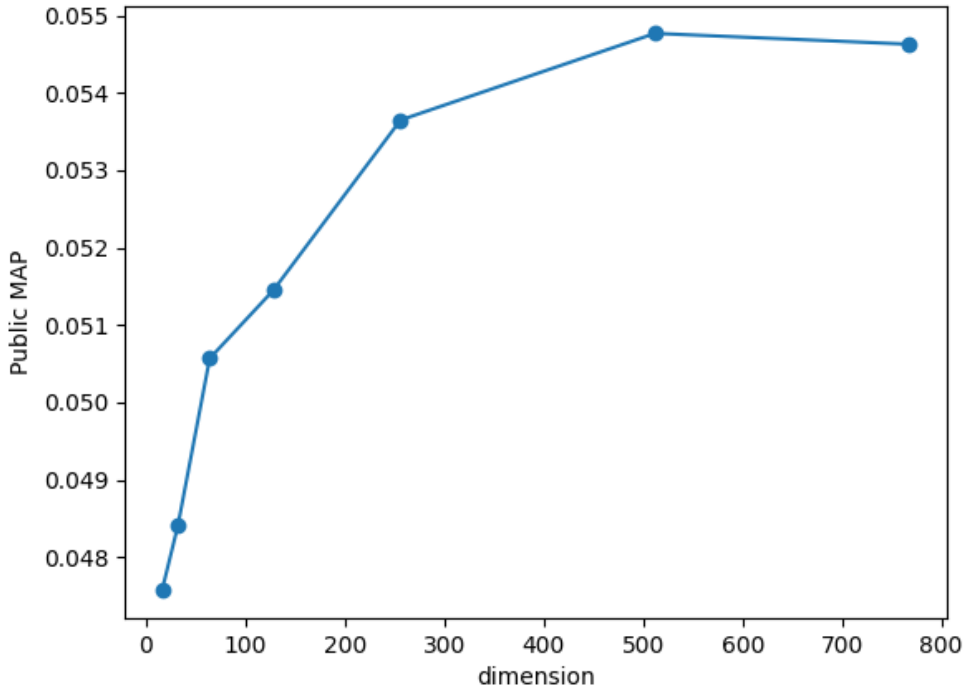
Q3 : Compare your results of Q1 and Q2. Do you think the BPR loss benefits the performance? If do, write some reasons of why BPR works well; If not, write some reasons of why BPR fails.

Ans:
我覺得BPR相較於BCE來得更好，當然在kaggle上面的成績有所進步是其中一點，但主要的原因我覺得在於我們的這個task是ranking的task，BPR的目標是希望能夠讓negative的item和positive的item相差越遠越好，而BCE則是希望能讓positive越接近1，且negative越接近0，兩者最大的差異在於BPR有考慮到positive和negative之間的關係，但BCE並沒有，因爲我們最後的衡量指標並不只是要預測這個物品是否有被瀏覽，而是要對其預測結果做ranking，因此BPR loss 會比BCE loss更加適合這樣的任務。

Q4 : Plot the MAP curve on testing data(Kaggle) for hidden factors d = 16, 32, 64, 128 and describe your finding.

Ans:

Figure:



dimension \ MAP & loss	Public MAP	Validation loss
16	0.04757	0.1743
32	0.04841	0.1678
64	0.05057	0.1616
128	0.05145	0.1581
256	0.05365	0.1551
512	***0.05477***	***0.1537***
768	0.05463	0.1540

Findings:
原本僅有要比較到dimension=128的情況，不過因為分數跟validation loss都有持續變好，因此我便持續往更高dimension進行實驗，經過上述的圖表可以發現，在我固定的參數設置下，dim=512為最好的結果，這樣的結果可以解讀成item和user的embedding呈現在比較小的dimension時，因為參數量不夠，可能導致embedding無法表達其完整的含義，因此在最後的分數上還沒有辦法達到最好，而到dim過大時，可能會導致model提早開始overfitting，也會造成performance上的些微降低。

(Bonus 10%) Q5 : Change the ratio between positive and negative pairs, compare the results and discuss your finding.

Ans:
以下為不同negative sample ratio在kaggle上面的 public & private 的MAP分數

	Public MAP	Private MAP	Average MAP
1:1	0.04984	0.04892	0.04938
1:2	0.05235	0.05066	0.05150
1:3	0.05402	0.05079	0.05240
1:5	0.05477	***0.05267***	***0.05372***
1:10	***0.05505***	0.05046	0.05275
1:15	0.05482	0.05253	0.05367

Findings:
negative sample ratio在訓練時最主要的差別是收斂的epoch數量，1:1在我預測的learning rate和scheduling的方法大概要70epoch左右才會收斂，而1:15則在第10個epoch左右就收斂，其實這也很合理，因為畢竟比例越小時，看過negative的sample就越少，因此要花更多epoch才能收斂也很正常。而我覺得造成上面分數的差異可能也跟看過negative sample的數量有關係，像是1:1, 1:2, 1:3的實驗，雖然我都已經等到它們的validation loss沒有再往下降才停止，但其實他們看過的negative sample其實相較於比較大的ratio還是比較少的，如果再繼續train下去或許可以達到更好的成績。

其實在kaggle deadline以前我還沒有進行negative sample ratio的實驗，最後我選擇1:5的原因是因為看了一些paper，發現他們大部分都使用1:5的ratio，結果還滿幸運的，在我設置的參數中，綜合了public和private分數，1:5的分數真的為全部當中最高的！對我來說，本來滿不清楚 negative sample ratio 到底對於performance 會有多少影響，在嘗試和參考別人的paper，以及自己實際做了這些實驗比較後，覺得獲得滿大的學習和收穫。