

Web Mining programming HW2

R09944019

Describe MF

由於 BCE 跟 BPR 都是基於一個 Scoring 的 Matrix Factorization Model, 因此先描述這個基本的 scoring model 的作法。

- Latex matrix: 透過 “torch.nn.embedding” 建立兩個 latex matrix 分別作為 user embedding 與 item embedding, 一開始的是取 hiden factor 為 32, 並對 matrix 做 normalize 避免 update parameter 爆掉的問題
- Split dataset: 因為本次作業只有提供 training data, 因此依照作業的提示, 把整個 training data 按照 9:1 的比例分成 training data 與 testing data, testing data 用來做 validation 提供調整 hyper parameter 的依據
- Training:
 - 一開始的 hyper parameter 是用 batch-size= 1024, lr= 0.01, lambda= 0.001, loss_function= MSE, optimizier=SGD 下去 train 得到的結果會最好, 不過這部份的數據在使用 BCE 與 BPR 作為 loss 時差異很大, 因此更細部的 hyper parameter 調整會在 BCE 與 BPR 的部份描述
 - 先建好 utility matrix (interaction), 把有 label 的設為 1, 沒 label 過的設為 0, 訓練時, 僅取出設為 1 的 element 對應的 latex matrix 的 row 與 col 出來內積, 內積得到的結果即為預測值。這部份雖然說僅取出設為 1 的 element, 但用在 BCE, BPR 時, 因為同時需要 positive 與 negative item score 因此也會選設為 0 的 element 對應的 latex matrix 的 row 與 col 出來計算。

Q1: Describe MF with BCE

基於上述的 scoring model, BCE 的概念是把 user label 過的資料作為 positve item , 反之則為 negative item, 並且因為在 BCE 中僅有 0/1, 因此會分別盡量讓 predict 1 的 val 越大, predict 0 的 val 越小越好, 因此 loss function 為 :

$$(-1) * [\sum \log(\sigma(pos))) + \sum \log(1 - \sigma(neg)))]$$

第一層 sigmoid 是 prediction 出來的值介於 0 ~ 1 之間 , 當 prediction 的是 positive item 時, 會希望值越大越好 , 因此 sigmoid 值越接近 1 越好 , 當值越接近 1 取 log 時, 就負的越少。反之, negative item 的部份用 1 - sigmoid(pred) 是希望讓值越小越好 , 當 1 - val 時, 反而會讓值變大 , 藉此得到負比較少的結果。最後再乘上 (-1) , 使得 loss function 越少時, 代表預測的越準。接著再以往 loss function 越小的方向更新

- Sample data: 在先前建好的 utility matrix 中取出值為 1 的 item 作為 postive item, 取出值為 0 的 item 作為 negative item
- Sample rate: 在 BCE 中, 我所測到最佳的 positive : negative 大約是 1 : 15, 並且每次 training 都做 repair 去增加他的隨機性

- 利用 scoring model 去 predict 對應 positive item 與 negative item 的並套入上述的 BCE loss function 後，透過 SGD 做 optimization，其中的設置 lr = 0.01, lambda = 0.0001
- Hyper parameter: lr=0.01, batch-size=4096, epoch=20, lamb=0.0001, positive/negative rate=1:15, hiden_factor=128

另外一開始 epoch 都是設在 30 附近，但後來發現大概到 20 左右的時候 training loss 還在下降，但 map 却開始大幅度下降，猜測試是 overfitting 已經發生，因此採用未 overfitting 的結果即 epoch 20 得到最佳的預測結果。

Q2: Describe MF with BPR

基於上述的 Scoring model, BPR 的概念一樣是把 user label 過的資料作為 positive item, 反之則為 negative item, 與 BCE 不同的是, BPR 是採用 item ranking 的概念, 即 positive item 的 score 應該要大於 negative 的 score, 因此希望 predict 出兩者間的差距越大越好，為此 loss function 為：

$$(-1) * \left[\sum \log(\sigma(pos - neg))) \right]$$

當兩者差距越大時, sigmoid 越接近 1, log 值也越負的越少，因此乘上 (-1) 時數字也會正的比較少，也就是說兩者差距越大時，loss 越小，預測的越正確。

- Sample data: 在先前建好的 utility matrix 中取出值為 1 的 item 作為 postive item, 取出值為 0 的 item 作為 negative item
- Sample rate: 在 BPR 中, 我所測到最佳的 positive : negative 大約是 1 : 15, 並且每次 training 都做 repair 去增加他的隨機性
- 利用 scoring model 去 predict 對應 positive item 與 negative item 的並套入上述的 BPR loss function 後，透過 SGD 做 optimization，其中的設置 lr = 0.01, lambda = 0.0001
- Hyper parameter: lr=0.01, batch-size=64, epoch=30, lamb=0.0001, positive/negative rate=1:15, hiden_factor=128

BPR 的部份，我發現有滿多情況會造成 update parameter 爆掉：

- lr 太高，使得更新的太多，update latex matrix 的 element 直接變成 nan
- hiden factor 太高，gradient 直接被 latex matrix dominate
- lamb 太高，一樣 gradient 受 latex matrix dominate

當然，hiden factor 太高與 lamb 太少又可能造成 overfitting。

Q3: BCE v.s BPR

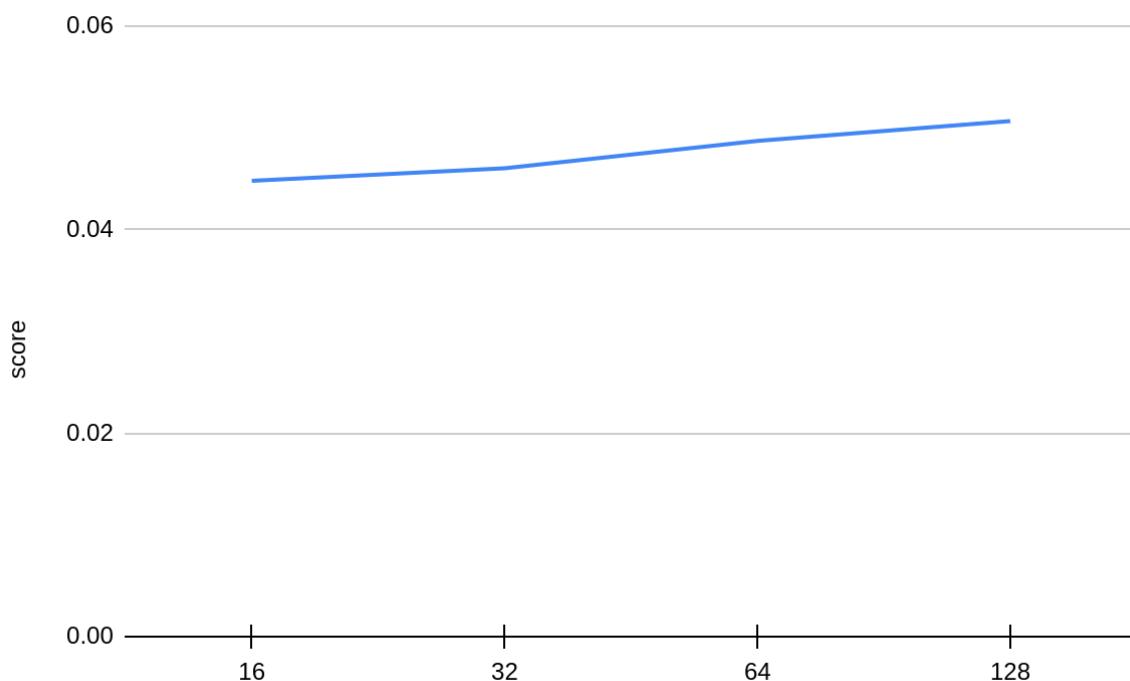
- Do you think the BPR loss benefits the performance ?
 - 以本次作業來說， No
- 本次作業中這兩種 model, 我的 BCE 的表現是大於 BPR 的，我認為 BCE 的表現應該要比 BPR 要好是正常的，原因是作業是採用 implicit interaction 的 data，即 data 只有 1 / 0 的情況，這種情況正好符合 BCE 的假設，而 BPR 雖然在 1/0 的情況也通用，但應該是要在 rating 有分程度 (0~5) 的 dataset 下，效果才

會較顯著。而事實上，本次作業中我僅有 BCE model 通過 baseline, BPR model 的 performance 非常差，這也證明了這個推論。

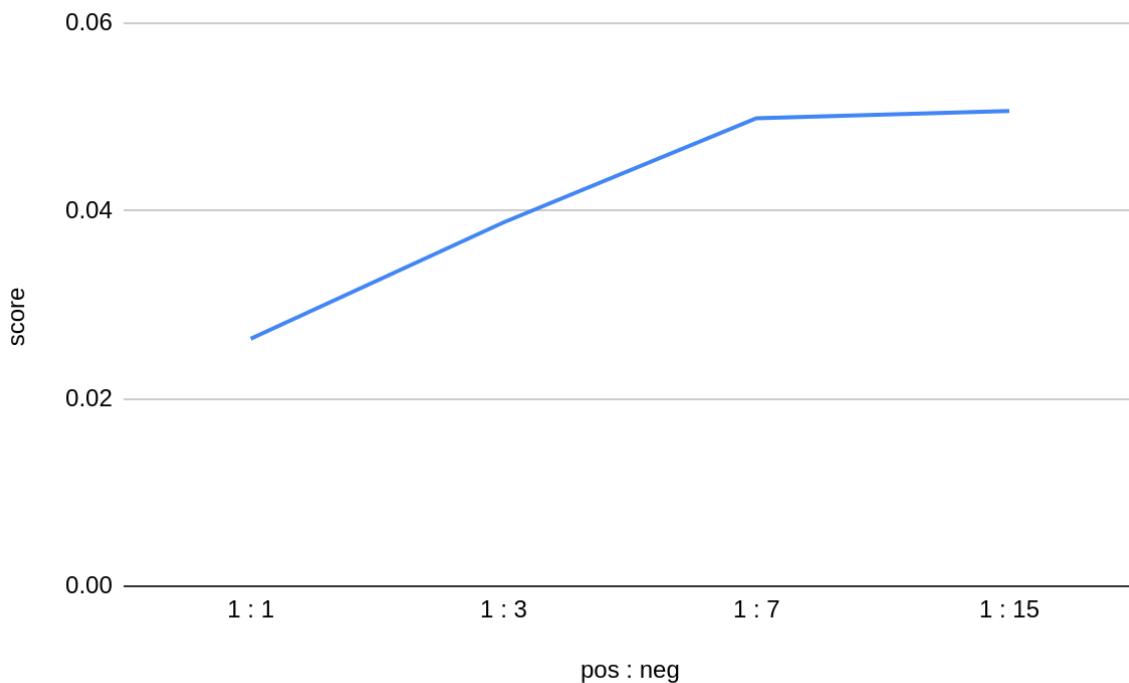
Q4: Plot the MAP curve on testing data(Kaggle) for hidden factors d=16,32,64,128

在其他 parameter 不變的情況下，hidden factor 越大，performance 理論上會先升高後下降，先升高的原因是適度的增加 latex matrix 的 dimension 時，因為維度增加運算量提升，準確度也可以隨之增加。後下降則是因為當 hidden factor 過大時，對於 train data 的準確度大大提昇，相對的就可能造成對 test data 的預估發生錯誤，即 overfitting。但在我的實驗中，下圖，performance 是隨著 factor 增加而不斷增加，原因是我為了避免 overfit 的情況發生，有先適度的 regularization 也就是調整 lamb 的值，來 penalty 過度擬和的情況。

另外，為了避免 overfitting，過度的調大 lambda 也可能造成 gradient 往錯誤的方向更新，因此如何調整 lambda 也成為預測準確的一大因素。



Q5: Change the ratio between positive and negative pairs, compare the results



可以看到當 ratio of between positive and negative pairs 從 1:1 增加到 1:15 的期間 performance 越來越好，原因是 sample 的 negative item 較多也意味著 training data 也變多，因此 performance 也越好。不過可以看到 1:1 增加到 1:7 這之間 performance 上升的非常顯著，但 1:7 到 1:15 就僅增加一點點，由此得知 1:7 其實就很夠了，1:15 使得資料量變大兩倍 latency 也增大接近 2 倍，考慮到 efficiency, 1:7 大概是最好的選擇，因為其同時兼顧 accuracy 與 efficiency。