Web Retrieval and Mining 2021 Spring Programming HW2

r09922104 余友竹

1. MF with BCE

1.1 Method

1.1.1 Loss Function

我的BCE模型參考講義的公式，先通過sigmoid function，再計算binary cross entropy，這在pytorch中有well-implement的function，叫做`torch.nn.BCEWithLogitsLoss`。

1.1.2 Positive / Negative sampling

在positive/negative pairs的選擇上，我的做法是將不在一開始給定的item_list中的items全部標為negative items，並且使用uniform random sampling的方式選取。positive / negative 的比例與實驗結果會在bonus part做詳細的說明。

1.1.3 Validation

validation的切法，我根據講義的提示 (testing 的 positive數量約是trianing的 11%)，將training data中，每個user的item list切出11%的item當作validation set。實務上選取hyper-parameters時，我的作法是觀察validation set中，MAP的表現作為標準來選取。

值得注意的地方是，我的validation set只負責幫助評估hyper-parameters的好壞，以及挑選適當的early-stopping時機，實際上傳的檔案仍會用完整的training set做訓練。

1.2 Parameters

參數的部分，我的模型提供了幾個hyper-parameters供調整

- seed (s): 提供亂數模型固定的random seed, 方便reproduction
- learning_rate (lr): 決定更新gradient的大小
- num_epoch (e): 決定訓練的次數
- batch_size (bs): 決定一個mini-batch的大小
- weight_decay (w): 決定regularization term的權重，避免overfitting
- hidden_dim (d): 決定Matrix Factorization中，latent space的維度
- num_neg (nn): 決定positive / negative pairs的比例

在觀察validation set的表現後，我選取了以下的參數

```
-s 1126 -lr 0.001 -e 91 -bs 1024 -w 0.001 -d 512 -nn 1
```

根據這套參數得到的public / private score是：

PUBLIC	PRIVATE
0.05724	0.05165

值得注意的點是：調整num_neg的數字，可以客製化調整positive / negative pairs的比例，有關這部分的實驗，同樣會在bonus part列出。

2. MF with BPR

2.1 Method

2.1.1 Loss Function

同樣參考講義，我定義BPR的loss function的公式為：

$$\text{BPRLoss} = \sum_{(u,i,j)\in D} \ln \sigma(u^T i - u^T j)$$

值得注意的是，原始公式中，有加上一個跟權重大小有關的regularization term，但因為在套件中，我們可以輕易地使用weight-decay達到這個目的，因此也不需要特別在loss function中加上這個term。

2.1.2 Dataset Sampling

在dataset的部分，按照BPR原始論文 (https://arxiv.org/ftp/arxiv/papers/1205/1205.2618.pdf) 的建議，我一次sample一組 (u, i, j)，其中：

- u: user
- i: u的其中一個 positive item
- j: random sample一個不在item_list的item，作為negative item。

此外，為了實現原論文提掉的sampling with replacement的效果，我另外設置一個參數：duplicate (d)，複製了同一個 (u, i) pair d 次，而每個pair 獨立 sample 不同的 j

這麼做的好處是：在同一個batch中，有機會出現同一組(user, positve_item)，只是對應到不同（也可能相同）的negative_item，就能達到 sampling with replacement 的效果。

2.1.3 Validation

在BPR的實驗中，validation的設置基本上跟BCE大同小異，詳細內容可以參考 section 1.1.3。

2.2 Parameters

在我的BPR模型中，參數跟BCE基本上是一樣的，唯一多的一個parameter即是上述提到的：

- duplicate (d): 複製同一個(user, positive_item) 多次，以達到sampling with replacement的效果。

此外，因為調整 positive / negative 的比例只會對BCE模型產生影響（在BPR中，這個操作跟提高epoch數量基本是相通的，因此不需額外設置），所以在BPR模型中，我不會使用`num_neg (nn)` 這個參數。

在觀察了validation set的表現後，我選取了以下的參數

```
-s 1126 -lr 0.001 -e 14 -bs 1024 -dup 10 -w 0.001 -d 512
```

根據這套參數得到的public / private score是：

PUBLIC	PRIVATE
0.05687	0.05235

3. Comparison between BCE & BPR

從 section 1.2, 以及 section 1.3 附上的結果來看，至少在我的實驗中，兩種方法的表現幾乎是不相上下的，BCE在public上略好一點，而BPR在private上則是略高一點，從我實驗的結果來看，BPR with duplication的收斂速度較快，在validation set上的表現，可以在約3~5個epoch就收斂到合理的MAP scores。因此，我認為，整體來講，使用BPR還是佔有優勢。

使用BPR模型佔有優勢的原因，我認為在於，BPR模型的最佳化公式，是針對每個 item 的 relative scores 去做最佳化的。也就是，在每個(u, i, j) term中，一個positive item i得到的分數，一定要大過 negative item j 得到的分數。在這樣的前提下，我們有考慮不同的item之間的排序，也因此，較符合這次的任務（計算map score)。

此外，在原始論文中有提到，我們很難去enumerate所有可能的positve / negative pair，因此，sampling的數量可以直接的影響表現結果，這也是我為什麼要新增duplicate 的參數，以增加sampling不同 negative item的機率。

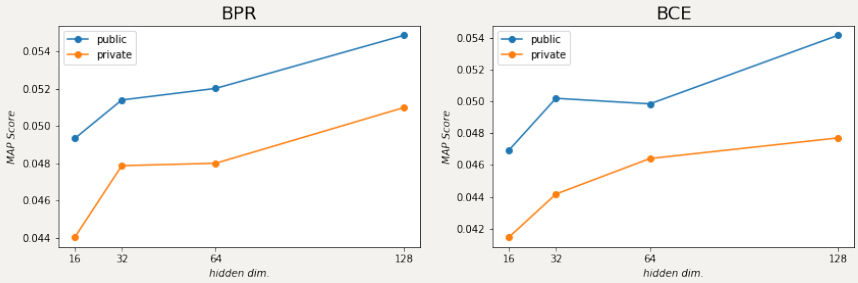
根據這個結果，我最後挑選了不同iteration下的BPR模型，將這幾個模型預測的scores作加權，得到一個簡單的 ensemble 模型，這個表現得到的public / private score是：

PUBLIC	PRIVATE
0.05748	0.05318

可以發現，使用BPR模型，並且加上非常簡單的ensemble機制 (uniformly blending)，即可讓public / private scores 皆有所提升。

4. MAP curce with different hidden factors

附表是我嘗試使用不同的hidden factor進行matrix factorization得到的結果。



可以看到，較高的hidden factor，不論不論使用BPR, BCE，在public / private 上，表現都有明顯上升的趨勢。

這可以從模型的複雜度上做解釋。理論上，越高的hidden factor可以描繪出越複雜的關係，卻也有機會造成overfitting的現象。由於我們的item總數為3260，在這個數量下，無論hidden factor 是選擇16, 32, 64, 128，應該都不至於到overfitting的等級。根據這個推測，我最後的模型採用的hidden factor是512，在前幾個section中也可以看到，最後的表現會優於這四個選擇。

5. (Bonus) Different ratios between positive and negative pairs

在 Section 2.2 中有提到，當我們使用BPR模型時，調整positve / negative pairs的比例，達到的效果跟提高epoch數量基本是相通的，不需額外設置。因此，在這個章節中，我只會探討different rations between positive and negative pairs 對 BCE 模型所帶來的影響。

以下實驗只會調整 num_neg (nn) 的數字，其餘的參數皆與 Section 1.2 提到的相同。

NUM_NEG	PUBLIC	PRIVATE
1	0.05724	0.05165
3	0.05699	0.05179
5	0.05642	0.05113
7	0.05326	0.05031
9	0.04906	0.04750

從表中可以看到，在 num_neg 為 1, 3時，兩者的表現相差無幾，甚至在private上的表現又更好了一點。但當我提高比例時，無論在public / private 上的表現都略為降低

這可以從很多面向去解釋，一方面我發現，調動 loss 的數值範圍，會對結果產生巨大的影響，而經過實驗，我發現不同的參數，對應到的最佳數值範圍，皆有所不同。因此，在固定其他控制參數的情況下，對於不同的ratio，會有所不公。

此外，當我們調整了 positive / negative 的比例後，這樣的任務似乎變成另一個machine learning的經典問題：imbalance learning。主要在探討，label 分布不均的情況下，如何在學習策略上做調整，以達到更好的表現。

顯而易見的，如果我們沒有對imbalance的dataset做特殊處理，當我們顯著提高兩者的比例差距時，對應到的結果自然會變糟。