Summer tutorial HW2　　r01922164 李揚

實作:

我是用 Stochastic gradient descent。大致上跟投影片講得一樣

而我存 sparse array 的方法很簡單，我定義一個資料結構 Data

他裡面包含 item id, user id, time ,rating。用這種方法來存所有的 rating。

基本的 model 的 parameters 我這樣設定

Latent class = 10

Regularization term = 0

Learning rate =0.000005

額外的

User bias, item bias。並且加上 smoothing

Smoothing 部分 (後面最後一部分會提及)

User 0.1

Item 5

除此之外，我加了以下的東西

1. User, item preference

   我這部分是先把每個 user 還有每個 item 評分(或被評分)的 rating 算出來，然後再算他與平均的差距，在 training 的時候扣掉(等於說去除偏見)，在 predict 的時候再加回來。

   等於說在 training 前對每個 rating 做這件事

   Rating- user preference- item bias + 2*average rating。

   訓練好 model(p,q)後，predict 時把他加回來

   Predict+ user preference + item bias – 2*average rating。

   這平均來說會讓我的 RMSE 再進步 1。

   而我在 random p q 起始值時

   用

   Sqrt(average/# Latent class)+my_rand_method

   For example :

   sqrt(ave_rating/HF)+0.001*((float)((rand()*i*243245+j*3432134*rand()+3)%RAND_MAX)/RAND_MAX)-0.0005

   使得 p q 相乘都在平均附近，這樣比較不會收斂到不好的 local optima。

   那為什麼沒加上 temporal bias 呢?

   因為在這個 case 裡，temporal bias 沒有很好。

   一開始我其實是有加的，理由也很簡單，雖然投影片說照著時間順序去 train 會比較好，但那是在 validation set 的時間全部比 training set 晚的情形才會成

立，我仔細看一下這次的資料，發現並沒有這個性質。所以我可以想見，我照著時間拿 training set 的 data 做 SGD 是很沒根據的。
但我還是想要有看到 temporal bias 的效果，於是我就像上面一樣做做了一個 temporal bias。只是很不幸的，效果沒有很好。不但沒有比原本的結果好，把時間切的越細(分的越多短)，結果反而更差。
於是我去看了一下 data set 找原因，
我去把 training set 和 validation set 拿出來找原因，發現如果是同一個使用者的情形之下，他在各個時間的評分是沒有差很多的。而且在同一個時段的各個使用者的相關性也沒有很一定，甚至還有一個更慘的情形，就是 validation set 在某時段的 user，在 training set 這個時段沒出現過。**以上種種都導致在這個 data set 之下，temporal bias 不是一個很好的參考。**

除此之外，我有特別對 item_bias，和 user_bias 做過研究。
(以下結果是取 training set 去算好後，再用 validation set 計算 RMSE)

如果我不做任何 training 只用 user_bias 的話，RMSE 會是 29.9376
如果我不做任何 training 只用 item_bias 的話，RMSE 會是 34.9273
如果我不做任何 training 只用 average_rating 的話，RMSE 會是 38.2996
如果我不做任何 training 但 user_bias 和 item_bias 雙劍合璧的話，RMSE 會是 31.6548　(rating 數目加權平均的話)
一樣是雙劍合璧，但我現在不用 rating 的數目加權平均，只單純相加除以二，RMSE 會是 29.8842
一樣是雙劍合璧，但我是相加後在減掉 average rating，RMSE 會是 29.2217。(如果用後面我提到的 smoothing 方法還可以更好)
可以看得出來，在這組 data set 裡面，user_bias 就可以猜得很準了!!!!
雙劍合璧減平均也是很不錯的，加權平均反而很差。其實這道理也很簡單，因為本身 item_bias 就不太準確，且平均每個 item 的 rating 數是 user 的 1.66 倍。
所以結果就沒這麼好。
也可以觀察到一個現象，user_bias 的影響遠比 item_bias 的大，且可以看到加進 item_bias 只不過比只用平均好 3.3723。
也就是說在現實生活中，再這麼好聽的歌，在一個嚴苛的人眼裡評低分的機率還是很大。在這麼難聽的歌，在一個心地良好的人眼裡評分還是不會太差。
(這點在報告後面(smoothing 部分)還會做更深入的分析)

2. Retrain
我也有做 retrain 的動作，充分利用 data。
我蠻不確定 retrain 在您給的 test set 結果會不會比較好。所以我兩組都傳上去，希望到時候可以再告訴我結果。

心得:
我後來仔細思考，其實 Collaborative filtering 在做的事就是互相把對方當作 feature。
原本我沒有的 feature 就拿我 rate 過的物件(對物件來說就是被 rate)來當作我的 feature。
而 MF 在做的事就是在做分類!!!! 利用他的 latent class，把我的使用者和物品分到各個不同的 class。
假設有三個 latent class，
我就是等於說把原本有 item 數個 feature 的 users，把這些 feature 對應成三個，在物理意義來講，較像是把物品分成三種 type，而現在 p 矩陣所做的事，就是把各個 user 對這三個分群的物品的喜好程度當作 feature。
反之亦然。

小實驗:
然後我有做一個實驗，如果我一開始就把我的起始點，精確地設在我加上 bias 的點會怎樣。意外的發現那是 local optima(在那點跑 SGD 是不會動的)。可見這個 data set 真的受 preference 的影響很大!
單純的 preference model 解出的答案就是 MF 裡的一個解。
還有一點就是，一開始做一些小小的 random 是很重要的，若當純設在 bias 的那點很有可能卡住。
也可以想見，扣掉 preference 的 model 並不是很複雜。在我的實驗結果中，有六個 Latent class 就已經很好了，而且 regularization term 並不是很必要，加太大反而會使得 training set 和 validation set 都 fit 的很差。
做到 20 個 latent class 的 model 很容易 overfit(training set 非常好但 validation set 非常差)，要非常小心去選 regularization term，且就算選對，也沒有比 10 個 Latent class 的結果好。所以我推論扣掉 preference 的資料已經不是很複雜了。

我犯過什麼錯:
我自己覺得這些錯誤對第一次寫 MF 的我來說是很難發現的，所以寫在這裡警惕自己。

1. 自取其辱(predict 不可能的值)
   在 predict 的時候，其實沒有任何保證，p 跟 q 乘出來的那個內積，會在哪個值之間。而要是很不幸的，我的 model 覺得某人特別喜歡某物，就 predict 一個遠超過 100 的值，或我的 model 覺得某人特別不喜歡某次，就 predict 一個遠低於 0 的值，這會對我的 model 的 RMSE 有非常大的影響。
   所以如果 prediction 超過 100 應該就設成 100 了，在大即使這個使用者真的很喜歡這個物品(rating=100)，還是有 error，反之亦然。
2. smoothing

這是我在做 user, item preference(bias)時犯下的錯誤，再算 user, item 的 preference(bias)的時候，有很多人沒有評分，也有很多物品沒有評分。一開始我覺得反正這些沒有評分的我之後再 training 也沒有用到，所以就給他隨便一個值(我那時是給 0)，但這其實是非常不好的事。雖然這對我訓練的 model 沒有影響，但在 predict 的時候就造成了非常大的影響，在 validation set (held-out data)會有一些我之前在 training set 所沒有看到的 user id 和 item id，這時候我就等於在 predict 這些東西的時候，無端給它減去兩倍的 average rating，這造成我的 RMSE 一直無法突破 28。所以後來我再算 preference 對沒有的 user id 和 item id 我都會給他 average rating。其實這也很合理，對於一個之前沒有出現的人(或物品)，我給他大家 rating 的平均(或每個物品被評分的平均)。
後來我仔細想想還可以做一點點簡單的 smoothing，也發現 RMSE 有因此稍微好一點。
我的 smoothing 方法如下:
使用者的偏好 * 使用者的 rating 數 + 0.1*平均 rating
除上 使用者 rating 數加 0.1
物品我在測試的時候發現他對我整體系統的幫助較小，於是我做了較大的 smoothing 3(相當於把比較不可靠的 bias 減小效用)，這樣也可以得到些微的進步。
然後我也把 smoothing 後的結果做不做 MF 的測試，(user_bias+item_bias 減平均)，去算 RMSE，發現可以到 28.863

我做這樣的 smoothing 其實是有理由的，因為前面我有提到，在這組 data set 裡面，item_bias 不是這麼有效力的，主要還是要看到 rate 它的是哪個 user。所以可以發現，user 的 rating 比 item 的 rating 重要很多。
但即使如此，我的 item 只要夠多的話，他一樣可以很重要。於是我把 item 那邊做了一個很重的 smoothing，他必須要有很多 rating(要很多人說它好或差，我的 model 才真的覺得差，不希望被偏激的使用者影響)，才可以把 item_bias 漸漸拉回拉到他真正的 item_bias。平均每個物品會有 38.333 個 rating，所以我覺得 5 的 smoothing 並不過分。
然後我也把 smoothing 後的結果做不做 MF 的測試，(user_bias+item_bias 減平均)，去算 RMSE，發現可以到 28.863
而在 training set 的 RMSE 更是低到 27.823
所以我其實在 train 之前的起始點是不錯的。可惜經過 MF 一長串的訓練後的進步只有 2.3 左右，反倒沒有比不加 bias 和 smoothing 好多少。(在 validation set 上 RMSE=26.5 左右)。