CS 5304 Final Project - Part 1: Medical Content Recommendation System

Jingwen Bi (jb2548), Danqi Mao (dm786), Xuanrui Zhang (xz572)

April 2018

1 Introduction

With a looming shortage of nearly one hundred thousand Physicians in the U.S. alone and an explosion of medical research, Doctors face the impossible task of staying current in their fields. Current tools don't source content from across medical journals or have a terrible user experience. We're building a recommendation system for Physicians that provides the latest research relevant to their practice in real-time through a simple mobile app.

2 Questions

Question 2.1 *Given a group of users and their liked articles, among other articles they have not yet liked, which one is likely to be liked by them? Which one is more likely to be liked? Which one is less likely to be liked?*

3 Data and Source

Because of the copyright issue for the medical journals, we cannot obtain the access and the related features to each single articles one user has read. Also, since the user information is inconsistent among different journal websites, it is hard for us to collect data this way. Our idea of collecting data is based on the "browsing history" and "like articles" from our app usage, which is similar to the format of data from **CiteULike**, which includes some medical content. The CiteULike database is constantly updating and it takes time to scrawl enough medical data (more than 5000 users for example.) Thus, in order to get the expected amount of raw data in the format we want, we decide to use all types of articles from `http://www.citeulike.org/all` as our ***medical articles***. For next steps, we will collect true medical article information, reapply our model to this data and to improve it, but the general idea of implementation should be the same.

4 Dataset Description

4.1 Raw Data

The raw data contains 5,551 users and the liked articles for each user. We use the **ImplicitDataset** function to reconstruct them into the tuples of (*user i*,*article j*), which represent that *user i* has liked the article j.

4.2 Validation, Test and Train Data

Validation Data: For each user, we randomly chose one liked article by the user and take them as the validation data. Then, we have 5,551 tuples (*user*, *article*) in the validation data.

Test Data: After deleting the validation data from the raw data. We randomly chose one liked article for each user and take them as the test data. It is possible that for users that have liked only one article before, it has no test data.

Train Data: After deleting validation and test data from the raw data, We take the rest as our train data.

5 Baseline Model Description

5.1 Why BPR?

From the paper *BPR: Bayesian Personalized Ranking from Implicit Feedback*, we learnt that there are many popular learning algorithms for recommendation systems. For example, K-nearest neighbor (KNN) collaborative filtering is the most popular method, as well as Matrix Factorization also has become more and more popular, and more methods. However, according to the authors, Bayesian personalized ranking method outperforms all other methods. This is because unlike other methods, which "optimize to predict if an item is selected by a user or not", BPR directly optimizes the parameters for the ranking. [1]
Openrec follows the thought of the paper, implemented BPR in the program.

5.2 BPR explained

5.2.1 Notation and Preprocess

Let $\boldsymbol{U}$ be the set of all users and $\boldsymbol{I}$ the set of all items(books). The algorithm will provide each user with personalized total ranking $>_u \subset \boldsymbol{I}^2$.
Properties $>_u$ must satisfy among total order:

$$\forall i,j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i \qquad \textit{(totality)}$$
$$\forall i,j \in I : i >_u j \wedge j >_u i \Rightarrow i = j \qquad \textit{(antisymmetry)}$$
$$\forall i,j,k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \qquad \textit{(transitivity)}$$

For convenience we also define:

$$I_u^+ := \{i \in I : (u,i) \in S\}$$
$$U_i^+ := \{u \in U : (u,i) \in S\}$$

Figure 1: Properties of $>_u$

Denote training data D_s :

$$D_S := \{(u,i,j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

For $(u,i,j) \in D_s$, it means that user u is assumed to like i more than j. When $>_u$ is antisymmetric, the negative cases will tread as implicitly.

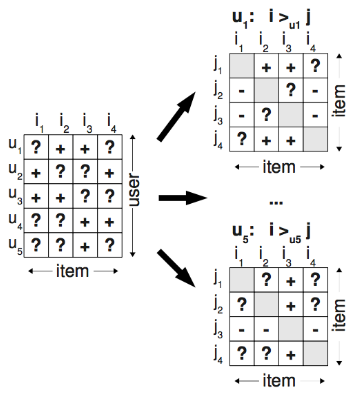


Figure 2: The left side is observed data set. The right side shows the BPR approach which building user's pairwise comparison between a pair of items. Plus(+) on box(i,j) means user prefer item i over j; Minus(-) means opposite as Plus(+).

For any user, if all value of pair (i,j) are missing, it will treat it as missing; if one number in pair (i,j) is known, it treat the known item rank higher than unknown; if all value in (i,j) are known, it will treat it as missing. This is the criterion behind Figure2.

5.2.2 Optimization Criterion

The optimization criterion of BPR was derived from the maximum posterior estimator for optimal personalized ranking. Let Θ be the parameter of any model class. (e.g. Matrix factorization)

$$p(\Theta | >_u) \propto p(>_u | \Theta)\, p(\Theta)$$

The posterior distribution given above is proportion to prior distribution multiply by likelihood. For each user, the probability that they prefer item i to item j is denote:

$$p(i >_u j | \Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

where σ is the sigmoid function. $\hat{x}_{uij}(\Theta)$ contains information between user u, item i and j, which calculated from any model with parameter Θ. Let's assume the prior distribution$p(\Theta)$ follows normal distribution with zero mean and variance/covariance matrix $\lambda_\Theta I$. Then, the log-likelihood function of the posterior distribution is:

$$\begin{aligned}
\text{BPR-O}{\scriptstyle\text{PT}} :&= \ln p(\Theta | >_u) \\
&= \ln p(>_u | \Theta)\, p(\Theta) \\
&= \ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij})\, p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}$$

where λ_Θ is the regularization parameters to prevent overfitting.

5.2.3 Learning Algorithm

The algorithm wants to maximum the posterior estimator for optimal personalized ranking, which is same as finding the Θ that maximized the log-likelihood function of posterior distribution. It also means finding the Θ which minimized the loss function (negative log-likelihood function). Since computing the full gradient in each step is time consuming and not feasible in large data set, stochastic gradient descent with bootstrapping was being used in BPR. Here is the learning algorithm:

```
1: procedure LearnBPR($D_S, \Theta$)
2:     initialize $\Theta$
3:     repeat
4:         draw $(u, i, j)$ from $D_S$
5:         $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial\Theta}\hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
6:     until convergence
7:     return $\hat{\Theta}$
8: end procedure
```

For standard collaborative filtering models, their prediction is $\hat{x}_{ul}$ for every user-item pairs. Since BPR method has three parameters (u, i, j), then $\hat{x}_{uij}$ in BRP method has been defined as $\hat{x}_{ui} - \hat{x}_{uj}$.

Next, we can apply any standard collaborative filtering models to predict $\hat{x}_{ul}$. Then, the algorithm will regress on the difference of two predictions($\hat{x}_{uij}$) in order to find Θ which minimized the loss function. The evaluation method of BPR model is using AUC(Area under ROC Curve), which defined as:

$$\text{AUC}(u) := \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)$$

where δ is the Heaviside function(Unit step function).

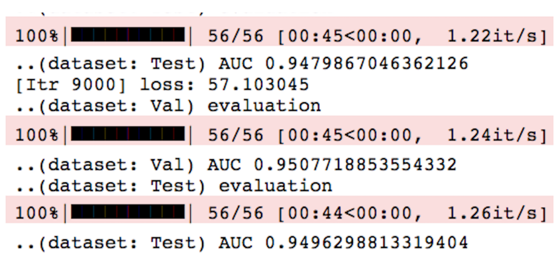
6 Our baseline model and the results

6.1 Our baseline model

Our code for implementing the model, scrawling data from CiteULike, and some saved data on different medical topics can be found in `https://github.com/xuanruizhang0203/Qual`. You can find the baseline model on general CiteULike data in the file ***Model _1.ipynb***. All the general CitiULike data we used in the baseline model can be found in the github under the folder **ctrsr_datasets**.

6.2 The results

The program takes about an hour to run. The final results can be found in the very bottom of the file ***Model _1.ipynb***. Our results shows the loss and AUC values. The interpretation of the AUC, according to wikipedia, is ”the area under the curve (often referred to as simply the AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming ’positive’ ranks higher than ’negative’)”.[4]
For your convenience, the screen shot version of the results are shown down below.



```
100%|██████████| 56/56 [00:45<00:00,  1.22it/s]
..(dataset: Test) AUC 0.9479867046362126
[Itr 9000] loss: 57.103045
..(dataset: Val) evaluation
100%|██████████| 56/56 [00:45<00:00,  1.24it/s]
..(dataset: Val) AUC 0.9507718853554332
..(dataset: Test) evaluation
100%|██████████| 56/56 [00:44<00:00,  1.26it/s]
..(dataset: Test) AUC 0.9496298813319404
```

References

[1] Steen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. *BPR: Bayesian Personalized Ranking from Implicit Feedback.*

[2] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. *OpenRec: A Modular Framework for Extensible and Adaptable Recommendation Algorithms.*

[3] OpenRec: `https://github.com/ylongqi/openrec`.

[4] https://en.wikipedia.org/wiki/Receiver_operating_characteristicArea_under_the_curve