Final Report

Qinzhou Li
qli8@albany.edu

May, 2020

Contents

---

1 Introduction

As online recommendation systems can be harmful to students' performance[4, 5], we need to create a system that can do both recommendation and evaluation offline, which could be more valuable for students. In order to create this system, there are two major steps: 1) Based on current information about students, we need to find out a way to predict her/his future performance on unsolved questions. In this project, we implemented two off-line methods to analyse students' previous performances and questions' latent features, so that there is a way to represent the estimate score. 2)Once we collect the predicted information, an proper algorithm to recommend questions for students is essential. We do not want to recommend a question that is too easy or too hard for students, since in both ways, the student's learning will be hurt. Therefore, we come up with an approach to evaluate each question's difficulty for each student and based on the student's current performance and ranked questions, the system can provide a suitable question for the student.

2 Related Work

Rank-based tensor factorization(RBTF) Than-Nam et al.[1] proposes a student performance prediction model,and the idea of the aforementioned assumptions is the notion of "concepts": gradual learning can be viewed as gaining knowledge in course concepts; student knowledge-based similarities are based on how much they mastered each of the concepts; and problem similarities are defined on how their represented concepts are shared. They model each problem as a vector of k latent concepts, that shows the importance of each latent concept in that problem. Also, they model each student's knowledge at any time point as another vector of the same latent concepts. By assuming that student $\mathbf{u}^{th}$'s performance on question $\mathbf{i}^{th}$ at attempt time t is a result of her/his existent knowledge in the latent concepts required by the problem. Hence, they model estimated student score $\hat{y}_{u,t,i}$ as a dot product between problem's latent concept vector q_i and student's knowledge in those concepts $t_{t,u}$:

$$\hat{y}_{u,t,i} \approx t_{u,t} \cdot q_i \tag{1}$$

BKT Based Methods Pardos et al.[2] proposes a Bayesian method using similar permutation analysis techniques and knowledge tracing to determine which orderings of questions produce the most learning. Pelánek et al.[3] leverages knowledge tracing to model students' skill acquisition over time and sequence questions to students based on their mastery level and predicted performance. One limitation of knowledge tracing is the assumption of single knowledge concept for each learning content.

3 Problem Formulation

For this recommendation system, we first collect previous information of students, which is a list of performances for students. Each sub-list represents one result of student practice, which includes the information of student id, question id, attempt time and score of current question. As these data are not completed, which means that students are just at the start of the tutorial.Therefore we can analyse the data, recommend more suitable questions for each student at different attempt time, so that the final performance of students could be maximum. The main problem is how to find which question is suitable for students at a certain attempt time.

4 Proposed Method[1]

4.1 Learning Analytics and Content Analytics via Tensor Factorization

Based on the information they have, they create a tensor $\mathbf{X} \in [0,1]^{u \times t \times i}$. For each entry $x_{u,t,i}$, it represents the performance of the student u for the question i at attempt t. Their purpose is to fill out these empty entries, which means that they need to find a way to predict performances that

[1]These two methods are proposed by Siqian Zhao, Chunpai Wang and Shaghayegh Sahebi.

students have not experienced yet. In order to solve this problem, they apply tensor factorization to decompose tensor $\mathbf{X_{u,t,i}}$ to three parts: $S_{u,s}, T_{s,t,c}, Q_{c,i}$, where they add "skill latent"(s) for students and "concept latent"(c) for questions. Therefore they represent students' grades as follows:

$$\hat{x}_{u,t,i} \approx p_u \cdot T_t \cdot q_i + b_u + b_t + b_i \tag{2}$$

where

- $\mathbf{p}_u$(one row of $\mathbf{S}_{u,s}$) is the latent feature vector of student u
- $\mathbf{T}_t$ is the temporal dynamic at attempt t
- $\mathbf{q}_i$(one column of $\mathbf{Q}_{c,i}$) is the latent feature vector of question i
- $\mathbf{b}_u$ denotes the student bias for student u
- $\mathbf{b}_t$ denotes the attempt bias for attempt t
- $\mathbf{b}_i$ denotes the question bias for question i

And then they apply the logistic regression to learn parameters and get the optimal model, where they minimize the regularized squared error:

$$\begin{aligned}\mathcal{L} = \sum_{u,t,i \in \Omega_{obs}} & (x_{u,t,i} - \hat{x}_{u,t,i})^2 + \lambda_t \|S_t\|_F^2 \\ & + \lambda_b \left(\|b_u\|^2 + \|b_t\|^2 + \|b_i\|^2 \right)\end{aligned} \tag{3}$$

Finally, by comparing the estimated values with the data in the "validate set", they evaluate the model by calculating the generalization errors.

4.2 Proximity-based Recommendation

They use the cosine similarity to measure how much a student understand a question, a higher score means that the student is familiar with the question. However, they do not want to just recommend the questions that students are familiar with and the questions should be not too hard so well. To handle this issue, they weight the cosine similarity with the estimated value 1-x u,t,i. The higher this weight, the less likely the student is to solve the recommended problem. As a result the multiplication of this weight and the cosine similarity provides an automatic balance between knowledge gain and comprehensibility of the recommended material.They interpret this weighted value as the proximity score of question i for student u at time t before time stamp t+ 1. A higher value of proximity score of a question indicates a higher probability that target student will get larger knowledge gain

$$\hat{d}_{u,i} = (1 - \hat{x}_{u,t,i}) \cdot cosine(s_{u,t}, q_i) \tag{4}$$

Therefore, they calculate questions' difficulty rank for each student at each attempt time. And then they collect a set of k questions for this student.Ultimately, by comparing the set of questions to the current question, they determine the final recommended question.

5 Offline Evaluation on Educational Recommendation[2]

As the online method is time-consuming. They have to wait for the students' feedback and also students need to wait for the system to give the recommendation, which is harmful to students' learning performances. In here, they come up with a new method to solve the evaluation of the

[2]This method is proposed by Siqian Zhao, Chunpai Wang and Shaghayegh Sahebi.

offline recommendation system,where they borrow the idea of nDCG and design a new method called normalized discounted cumulative hit (nDCH):

$$\text{nDCH} = \frac{1}{\mathcal{Z}} \sum_{j=1}^{k} \frac{2^{\theta_{hit}} - 1}{\log_2(j+1)} \tag{5}$$

Instead of using graded relevance values, they employ the θ_{hit}. It is 1 if we recommended the correct question based on the baseline or 0 if we miss it. Moreover, as they add the propensity score,which is the probability of a question being followed to a particular order given another question and can be used to reduce selection bias,for each DCH. Because our data all comes from the logged system and we cannot use the online testing as the evaluation, it is a good approach to avoid the misleading information from the logged system(e.g., question i has the highest frequency following question j, but it does not mean that question i is the best choice for the next question of question j.) They call this metric as inverse probability weighted nDCH (IPW-nDCH).Since they have the different number of interactions for different students, they need to compute the Average IPW-nDCH scores, which simply average over all IPW-nDCH scores of all interactions.

Finally, they need to evaluate the student's performance that is affected by the recommended questions. By comparing the Average IPW-nDCH scores with the true performance of a student, they can evaluate the offline system based on the correlation between the difference of these two scores, and they employ Spearman's rankcorrelation coefficient in our experiments, which is defined

$$\rho_{\text{spearman}} = 1 - \frac{6\Sigma_i d_i^2}{|U|\left(|U|^2 - 1\right)} \tag{6}$$

where|U|is the number of test students, and d is the difference in the ranks of the i student in Average IPW-nDCH and real interaction performance score.

6 Experiment

6.1 Dataset

Loura Dataset: is a dataset from an online platform for a whole semester study. For each question, there are several test cases that students need to pass all the test cases in order to move forward. By calculating the proportion of correct test cases in each attempt, we can get the score between 0 and 1. The data contains 234 students, 65 questions, and 29273 interaction records. In Figure 1, we can see the number of students' interactions at each timestamp. Also, we can see that different students' final attempt times for a whole semester study in Figure 2. Finally, the Figure 3 shows the distribution of the number of questions solved by students and we can see that about 60 percent students only finished half of questions and quit. The maximum number of quizzes finished by one student is 65, within minimum 103 attempts. We select users who had finished 65 questions as the test users, and there are 19 test users. Meanwhile, we also test 27 students with low performance that only 30-39 questions are finished.

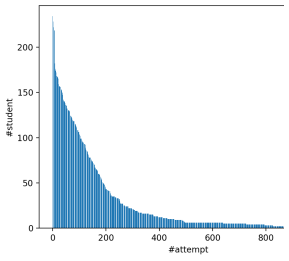

Figure 1

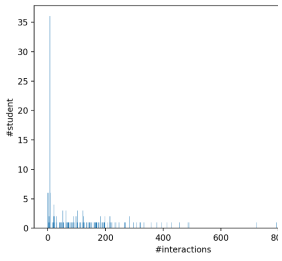

Figure 2

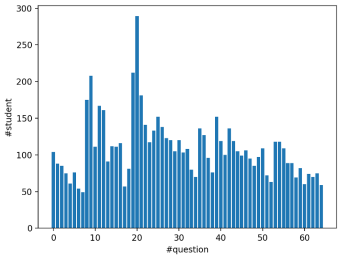

Figure 3

6.2 Experimental Result

We use two methods as simple baselines:

Random Recommendation: The random system picks 3 random questions to the target student at each time stamp. We report the average performance over 10 random runs for each experimental setting.

Popularity-based Recommendation: The system picks the top-k(3) most popular questions at each time stamp to the target student.

Method	Spearman Correlation(k = 3)
Popularity-based	0.084559
Random-based	-0.180538
Proximity–based	0.613572

The table 1 is the result of high performance students who finished 65 questions.

Method	Spearman Correlation(k = 3)
Popularity-based	0.24740
Random-based	-0.345043
Proximity–based	0.18379

The table 2 is the result of low performance students who finished 30-39 questions.

The starting attempt of the experiment is 60 and the ending attempt is 103. The minimum attempt time of our test users who finished all 65 questions is 103, so by setting the ending attempt as 103, the system can get all the information it needs. For these users who only finished 30-39 questions, we set the end attempt to 130 as it is the minimum attempt from these students records. Based on the shown results, we can see that our proposed method has a much bigger score than other two simple baselines in table 1, which indicates the effectiveness of our proposed recommendation method. However, in table 2, the test users did not have enough study records, so that the system performs badly on them.

7 Review and Analysis

As we need to calculate the cosine similarity of $s_{u,t}$ and q_i:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{\sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{a}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{b}_i)^2}} \tag{7}$$

we have to make sure that the $s_{u,t}$ and q_i are initialized properly in our code. During the experiment, we found that any initialization for matrixes $s_{u,t}$, q_i and tensor T will not affect the tensor factorization part. However, if we set our $s_{u,t}$ and q_i with equal number(Eg $s_{u,t}$=$\{1\}$ and q_i=$\{1\}$), the result of cosine similarity will be 1.(You can cancel out a and b, and the cos ($\mathbf{a}$,$\mathbf{b}$)= 1.) Therefore, for our $s_{u,t}$ and q_i, the uniform distribution is not available and it is better to use normal distribution for initialization.

8 Conclusion

In this research, we create an offline model to recommend questions for students as a tutor by tracking down students' personal learning performances. And also instead of getting feedback during the online study, which could take a long time and it's harmful to students' performance, we purpose a new way to evaluate this offline system before it is set up online.

References

[1] Thanh-Nam Doan and Shaghayegh Sahebi. Rank-based tensor factorization for student performance prediction.

[2] Zachary A Pardos and Neil T Heffernan. Determining the significance of item order in randomized problem sets. *International Working Group on Educational Data Mining*, 2009.

[3] Radek Pelánek. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3-5):313–350, 2017.

[4] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. Combining difficulty ranking with multi-armed bandits to sequence educational content. In *International Conference on Artificial Intelligence in Education*, pages 317–321. Springer, 2018.

[5] Guy Shani and Bracha Shapira. Edurank: A collaborative filtering approach to personalization in e-learning.