Collaborative Filtering for Problems on SPOJ using Spatio-Temporal Approach

Shashank Sonkar
ssonkar@iitk.ac.in
Dept. of Computer Science and Engineering
Indian Institute of Technology, Kanpur

1st Oct, 2012

Abstract

In this project, we aim to develop a recommender engine for the users of the Sphere Online Judge (SPOJ) which has a large collection of programming problems. Our engine would suggest specific problems to a user catering to his problem requirements, current intellect and skill. We suggest a collaborative filtering approach exploiting both the spatial as well as the temporal factors which determine a user's choice of problems.

1 Introduction

Enthusiasts of online competitive programming love to solve problems involving logic, mathematics and algorithms. There are several websites (eg: SPOJ, Topcoder, Codechef, UVAOnline Judge, POJ etc.) which host such problems and where users submit their solutions in form of computer programs. Of them, the polish website Sphere Online Judge - SPOJ (www.spoj.pl), with over 100,000 registered users from over 2500 academic institutions around the world provides an excellent platform for students and professional programmers to practice problem solving.

With over 10,000 problems of varying difficulty based on numerous concepts ranging over Mathematics and computer algorithms, it serves good for people preparing for prestigious competitions such as ACM-ICPC, Google CodeJam, Facebook Hacker Cup and many more.

During his learning and practice, a user may either want to solve problems based on a certain concept or algorithmic paradigm or, he may just want to pick some random problems and solve. As the number of problems is large, it is often not very easy for him to search for relevant problems quickly. Problems vary in difficulty as well as the concepts involved. There comes the role of a problem recommender which can suggest problems on the topics a user may want to solve with appropriate difficulty level according to the user's past performance.

An important aspect to consider is that a programmer's taste for problems and their difficulty level changes over time. With time and enough practice, he learns, improves and needs problems sufficient enough to challenge his intellect. Thus, we find that predicting the problems which a user may choose next to solve depends on factors which are static such as the problems, their difficulty level, the concepts and algorithmic paradigms involved; as well as the factors which are dynamic such as the user's current intellectual level and his skill at writing programs. To the authors' current knowledge, there is no such system in this domain which gauges a user's dynamic performance and suggests problems accordingly.

The above discussion suggests that a user's choice of a problem to solve depends not only on spatial factors, but is also dependent upon temporal factors.

1.1 Problem Statement

There are thousands of users on SPOJ and the total number of problems avalaible on the judge is more than 10,000. It's very difficult for a user to eaily find a problem of a particular type of programming paradigm or genre which may interest the user to attempt it. Since, there are no options available on SPOJ to search the problems depending on the algorithmic paradigm, complexity, hardness, being biased to a certain programming language, its becomes a very difficult task to find a problem worthy of user's interest.

In this project, we are trying to handle the same problem. We aim to recommend problems to a user which he/she might be interested to solve, depending on user's

submission history and recent attempted problems. Our solution to above mentioned problem is not specific to SPOJ and can be deployed for any other online programming platform (like UVa Online Judge, POJ, etc). We chose SPOJ over others due to large dataset available publicly.

The input data to our algorithm will be the users submission history with the time stamps.

1.2 Related Work

Vikhyat Korrapati, a student of IIT Mandi developed a recommender engine for problems on SPOJ and Codechef earlier this year (2012). URL: `http://vikhyat.net/projects/problem_recommendations/` It uses item-item collaborative filtering approach[1] to measure similarity between any pair of problems and generates recommendations accordingly. This model is a static version of our proposed approach. Problems are rendered similar based on the common users who have solved them. For a user, problems similar to those which he has solved are given as recommendations to him. This approach doesn't include the temporal factors such as a user's current performance and recent taste of problems.

2 Approach

2.1 Spatial Approach

2.1.1 Basic Model

The basic model for colloborative filtering models the rating of item i by user u as

$$r^{(ui)} = (p^u)^T q^i + \epsilon^{(ui)}$$

where p^u and q^i are k-dimensional user and item factors respectivelly, and $\epsilon^{(ui)}$ is observation error. Estimation of these factors is obtained by use of regularizers to avoid overfitting. Generally, L_2 norm is used as regularizer.

2.1.2 Matrix Factorization

Let R be a $N \times M$ user-item matrix (N users and M items). $R^{(ui)}$ represents the rating of item i by user u. Considering the basic model of collabrative filtering, matrix factorization is used to find rank-k approximation of R.

$$R \approx PQ^T \quad (P\epsilon\mathbb{R}^{N\times k}, Q\epsilon\mathbb{R}^{k\times M})$$

where P is user-factor matrix and Q is item-factor matrix. Rows of P and Q are user-factors and item-factors respectively.

2.2 Temporal Approach

Temporal approach takes into considertion the change in ratings by user over the time for user factor estimation. Time is an important factor for dyanamic modelling of user and item factors. For simplicity, we assume item factors to be time-invatiant.

2.2.1 Dynamic Model

We assume that item factors are known beforehand (by item features from other source, or estimated by static Matrix factorization). We model user factor for each user as random walk with gaussian noise.

$$p_t^u = p_{t-1}^u + w_t^u$$

where p_t^u is user factor for user u at time interval t. w is guassian process noise. The observation can be modeled as:

$$r_t^u = H_t^u p_t^u + v_t^u \tag{1}$$

where r_t^u is vector of ratings from user u in time interval t and H_t^u is matrix composed of corresponding row of Q(item factors), v_t^u is gaussian observation noise.

2.2.2 Kalman filtering approach

Based on the dynamic model, the user factor can be dynamically and efficiently estimated using Kalman filtering. Kalman filtering(KF) sequentially takes the ratings $\{r_0^u, ..., r_{t-1}^u, r_t^u\}$ as observations and return state (user factor) at time t, denoted by $\hat{p}_t^u$ and associated variance, denoted by σ_t^u. In each KF step at time t, the estimate $(\hat{p}_t^u, \sigma_t^u)$ from previous step is updated by incorporating new observation r_t^u as:

$$(\hat{p}_t^u, \sigma_t^u) = KFupdate((\hat{p}_{t-1}^u, \sigma_{t-1}^u, r_t^u))$$

Since, equation (1) is linear (since factors are assumed to be time-invariant), Kalman filtering can be efficiently applied.

3 Algorithm

We need to apply static MF to obtain the intial user factors as well as item factors. Then, we would use iteratively apply method based on Kalman filtering to estimate the changing user facrtors.

3.1 Pseudo-code for Spartial temporal filtering

Input: R_0, the ratings before time 0, and the sequence of ratings $\{R_1, \ldots, R_T\}$.
Output: User factors $\{p_t^1, \ldots, p_t^N\}$
step 0: Initialization
Initialize user factors and obtain item factors through a static matrix factorization over R_0
step 1: Individual Kalman filtering
for each u, $(\hat{p}_t^u, \sigma_t^u) = KFupdate((\hat{p}_{t-1}^u, \sigma_{t-1}^u, r_t^u))$
step 2: Update the item factors
This step is omitted if factors are considered time-invariant.
step 3:
$t = t + 1$, go to step 1.

4 Our Progress

We have been able to extract the SPOJ user database by following steps:

1. Extraction of usernames of the top 5000 users from SPOJ ranking pages.

2. Extraction of complete submissoin history for those 5000 users.

We have been able to extract data from different sources, which will enable us classify the given problem on SPOJ with respect to programming paradigm(s).

References

Links:

1. Linden, G., Smith, B., York, J.: Amazon.Com Recommendations: Item-To-Item Collaborative Filtering. IEEE Internet Computing 7(1), 7680

2. Lu, Z., Agarwal, D., Dhillon, I.S.: A spatio-temporal approach to collaborative filtering. In: RecSys 2009, pp. 1320 (2009)

Data Sources:

1. SPOJ
   `www.spoj.pl`

2. Problem Classifier
   `http://problemclassifier.appspot.com/`

3. VNOI Classifier
   `http://vnoi.info/index.php?option=com_voj&task=classify&site=spoj&sort=1&limit=15000&limitstart=0`

4. SMILITUDE Classifier
   `http://smilitude.wikispaces.com`