Collaborative Filtering for Course Recommendations

Camille Bustalinio

Abstract—A recommendation system was built using a hybrid model composed of memory-based and model-based recommendations of Learning Management System courses for Orange Silicon Valley. Memory-based recommendations used previously taken courses of each user to recommend future courses, ranked using the course hierarchy. Model-based recommendations compared different collaborative filtering algorithms from Surprise, a collaborative filtering library in Python. SVD was used to predict pseudo-ratings, hyperparameters were fine-tuned using FCP in evaluating residuals.

Keywords—Collaborative Filtering, Recommendation System, SVD, Ordinal Rating, FCP

I. INTRODUCTION

Orange Silicon Valley has a learning management system for employee training. Different courses on different subject areas are available for employees in different locations. With employees of the same role having different backgrounds, the objective was to standardize knowledge. A personalized course recommendation was the proposed solution. Courses that an employee was most likely to complete were given preference.

Courses history for all employees in the company was used to recommend courses to 500 employees part of the project.

II. THEORY

Collaborative Filtering. Collaborative Filtering uses past consumed items by different users to provide recommendations for future consumption. In a typical CF scenario, there is a rating m*n matrix which includes a list of m users and a list of n items and lots of ratings. Items represented any kind of products. A rating $r_{u,i}$ means how the user *u* likes the item *i*. It is supposed that users mainly interested in high ratings. The key step of CF is to extrapolate unknown ratings [1].
User-based collaborative recommendation aims to calculate some similarity metric between all pairs of users and then predict a particular user u's rating for an item i by collecting and processing the ratings of u's "neighborhood" (all the other users with high similarity as compared to *u*) for *i*.

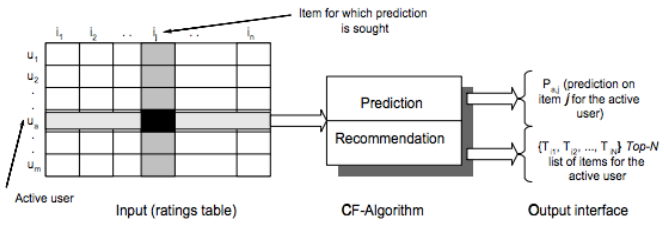

Fig.1 Collaborative filtering expresses the data as a matrix of user-item ratings

SVD. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. For example, when the products are movies, factors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined dimensions such as depth of character development or "quirkiness"; or completely uninterpretable dimensions. [2]

Accordingly, each item *i* is associated with a vector $q_i \in \mathrm{R}^f$, and each user *u* is associated with a vector $p_u \in \mathrm{R}^f$. For a given item *i*, the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For a given user *u*, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors [2]
The resulting dot product, $^1 q^T_i p_u$, captures the interaction between user *u* and item *i*—i.e., the overall interest of the user in characteristics of the item. The final rating is created by also adding in the aforementioned baseline predictors that depend only on the user or item. Thus, a rating is predicted by the rule:

$$\hat{r}_{ui} = \mu + b_i + b_u + q^T_i p_u.$$

In order to learn the model parameters (b_u, b_i, p_u and q_i) we minimize the regularized squared error [2]

$$\min_{b_*, q_*, p_*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

The *SVD++* (SVDpp in Surprise library) algorithm is an extension of SVD, taking into account implicit ratings.[3]

The prediction is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

In using SVD++, the numeric value of the rating is replaced by an indicator variable. Hence, ratings are treated as nominal instead of ordinal variables, and rank is omitted.

Evaluation Metrics. For evaluation metrics, popular metrics used to measure the performance of recommendation systems where a user gives an item a rating are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). RMSE and MAE are defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}.$$

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

RMSE and MAE can be quite detached from the ultimate goal of evaluating item ranking experience, since a perfectly ranked solution can score arbitrarily badly on numeric scale out of bounds, or just very close to each other. [4]
The RMSE and MAE metrics have another issue: they assume numerical rating values. First, they cannot express rating scales which vary among different users. Second, they cannot be applied in cases where ratings are ordinal.

Thus, besides using RMSE we also employ a ranking-oriented metric which is free of the aforementioned issues. Given a test set $\hat{R}$, we define the number of concordant pairs for user u by counting those ranked correctly by rating predictor $\hat{r}_u$.

$$n^u_c = |\{(i,j) \mid \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} > r_{uj}\}|$$

Discordant pairs n^u_d for user u are counted similarly:

$$n^u_d = |\{(i,j) \mid \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} < r_{uj}\}|$$

Summing over all users:

$$n_c = \Sigma_u n^u_c \ , \qquad n_d = \Sigma_u n^u_d$$

The quality metric used measures the proportion of well ranked items pairs, denoted by FCP (for Fraction of Concordant Pairs)

$$FCP = \frac{n_c}{n_c + n_d}$$

III. METHODOLOGY

Data preparation, model training, and output testing were implemented in Jupyter Notebooks. These are available in the project's GitHub repository.

A. Data Preparation

Available dataset was in the form of two tables: a course catalog, and a table of courses previously taken by the employees [STEP]. The course catalog had challenges: the catalog had a list of courses that occurred multiple times, as it was based on reports from irregular timeframes. Also, the same courses may have different course titles and course numbers. The variety in course numbers within the same course was attributed to their new iterations or versions. The dataset included URLs to the course summary: the pages were scraped for the text strings on course title, description, audience, duration, and objectives. TF-IDF was used to vectorize data from each URL, and pairwise cosine similarities were calculated to gauge if different URLs corresponded to different courses.

Aside from duplicity in course numbers, courses also followed a hierarchy: Solution Area >> Curriculum >> Series >> Course. Exploratory Data Analysis techniques were used to confirm if a course belongs to exactly one branch of this hierarchy. This would also indicate uniqueness of courses per course number.

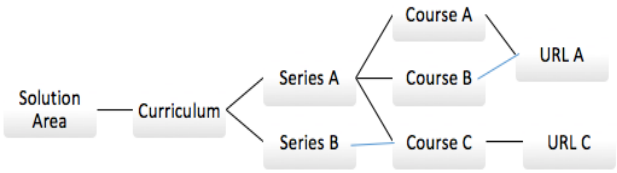

Fig 2. Course hierarchy within course catalog. Note that Course A and Course B are the same course, with different course names. Also, Course C belongs to Series A and Series B simultaneously

In the STEP table, courses that were taken had an indicator for completeness. Completed courses were given a rating of 5, while courses taken in progress were given a rating of 1. By default, courses not taken by each user had rating of 0. It was assumed. The ordinal pseudo-rating is an indicator of desirability in predicting courses: The objective is to prioritize courses that have been completed over courses that have not been completed. The pseudo-ratings used the ends of the rating scale to maximize the range in between courses when predicting rating.

B. Generating Recommendations

A recommendation was defined as a list of 20 courses not previously taken for each of the 500 employees included in the project.

Rule-based: With the hierarchy of the courses, a rule was created: Courses that would result to finishing the series should be recommended first. That is: if 2 series, A and B, have 5 courses each, and employee Z has completed 4 courses in A, and 3 courses in B, the last course in A would have higher priority in the recommendation, over the two courses in B. After the distinct courses were mapped under their hierarchies, a lookup table with unique series-course pairs was created, and from this, a Cartesian product of all the users. From this exploded table, the percentage of finished courses for each series was calculated as the recommendation score.

Model-based: Collaborative Filtering
Python's Surprise library was used for collaborative filtering. Different algorithms were used: SVDpp and SVD. Gridsearch was implemented across the different algorithms to fine-tune the parameters. Training was conducted on STEP records for all employees, including the 500 employees included in the pilot. RMSE, MAE and FCP were used to evaluate prediction residuals. Since the recommendation decision would be based on the ranking of the predicted rating, and not the actual prediction, the model with the highest FCP was selected, though not the model with the lowest RMSE or MAE. The model was able to predict a pseudo-rating score, which was used as an indicator of course preference. Ratings were ranked within each group of courses per employee.

Cold-start: A list with the top 20 courses with the most completion was generated. Theoretically, this would have been the default recommendation for a new user with no previous record of the courses. However, for this project, there was a fixed list of 500 users with records of previously-taken courses. Hence, a cold-start solution was unnecessary.

B. Model validation. During model training, FCP was the main metric used in comparing model performance. FCP as a metric treats the pseudo-ratings as ordinal numbers, instead of numeric (which is the case in using RMSE and MAE). Using RMSE and MAE both use the actual difference of the predicted from the actual. In the case of our data, the pseudo-ratings are indicators of preference, and has no numeric evaluation, hence are ordinal variables. Using RMSE and MAE would indicate validity of interpreting numeric values. Also, RMSE and MAE are sensitive to extreme values. In encoding pseudo-ratings, extreme ends of the scale were used (1 and 5) to maximize the scale, and highlight the differences in the ratings.

IV. Results

A. Finding distinct courses: One challenge for the dataset was having different course ID's for the same course. From EDA, multiple course numbers had exactly the same course summaries. The pairwise cosine similarities of courses within the same series have a left-skewed distribution. This indicated that even though courses belong to the same series, the summary descriptions indicate distinction between the courses. The only exception was the 60 course pairs with 90% cosine similarities. These courses had summaries that differ by less than five technical words (difference between programming language, Part 1 vs Part II of preparation materials for certification exams). From these results, we treated all courses that direct to the same URL as the same class. The exception for this was the video classes, which did not have URL information, and thus treated as different classes per course id.

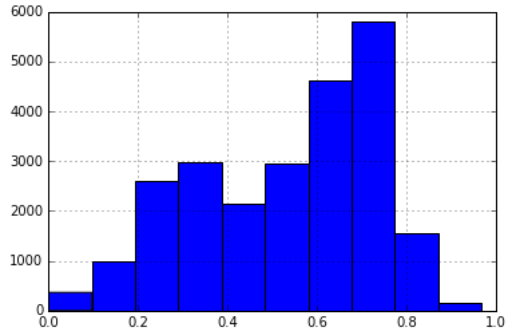

Fig.3: Distribution of the Cosine Similarity between pairs of course descriptions by URL where only 16 pairs had similarities over 90%

B. Rule-based

Based on the number of unique courses and 500 employees included in the project, there are more than 6.6million user-course combinations. After calculating the series-rating, only 675 data points had series scores greater than 0. All data points were from courses that were previously taken (completed or in-progress). Hence, the series-rating was irrelevant for creating recommendations.

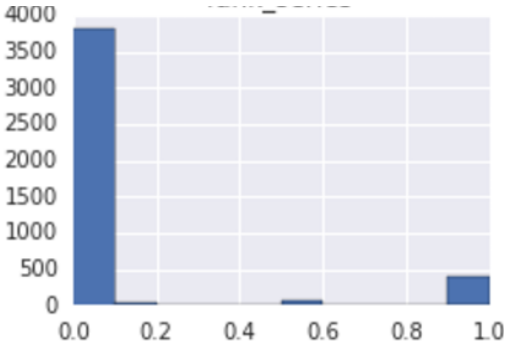

Fig. 4. Distribution of Series ratings indicate majority of courses had 0 scores, but the next bin with the highest score were series that were almost completed

C. Model-based:
Different collaborative filtering algorithms were used to come up with predicted pseudo-ratings. Matrix-factorization-based algorithms used were SVD and SVDpp. SVD and SVDpp both predict a user-item rating based on collaborative filtering. The main difference between these

algorithms is in how they treat the input data. SVD treats ratings as numeric variables, while SVDpp treats ratings as implicit indicators, regardless of value. Hence, a completed course was viewed the same way as a course in-progress.
After employing gridsearch to fine-tune model hyperparameters, the SVD model performed better than SVDpp in all accuracy metrics (MAE, RMSE, and FCP) in all folds

MAE	Fold 1	Fold 2	Fold 3	Mean
SVD	0.795	0.793	0.797	0.795
SVDpp	0.802	0.803	0.802	0.802
RMSE	**Fold 1**	**Fold 2**	**Fold 3**	**Mean**
SVD	1.135	1.132	1.138	1.135
SVDpp	1.149	1.151	1.150	1.150
FCP	**Fold 1**	**Fold 2**	**Fold 3**	**Mean**
SVD	0.775	0.731	0.742	0.749
SVDpp	0.773	0.731	0.733	0.746

Fig 5. Summary of error metrics across all folds show SVD is better than SVDpp

For actually completed courses, below is the distribution of the predicted rating based on the 2 algorithms:

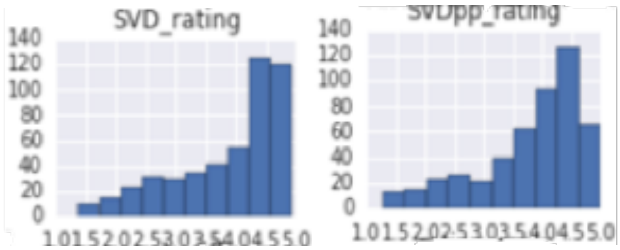

Fig. 6 Distribution of predicted ratings indicate SVD ratings that are heavier in range (4,5) than SVDpp ratings. Actual rating is 5 for all points

Both algorithms had left-skewed distributions. However, more predictions within the range of 4-5 were made with SVD algorithm versus SVDpp. Also, there were more predictions in the left tail of the distribution with SVDpp algorithm versus SVD.

For courses in progress, both algorithms had predicted ratings with similar distributions

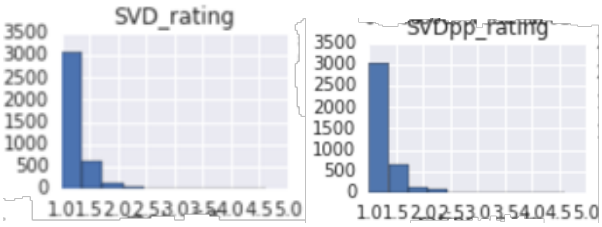

Fig. 7 Distribution of predicted ratings indicate SVD and SVDpp ratings are almost the same: heavy near the actual rating 1 for all points

For courses that were not previously taken, the top 20 courses with the highest SVP rating were collected as recommendations. For these datapoints, most of the predicted scores between 2-3.5. Still, around 14% had predicted ratings greater than 4 for both SVD and SVDpp ratings.

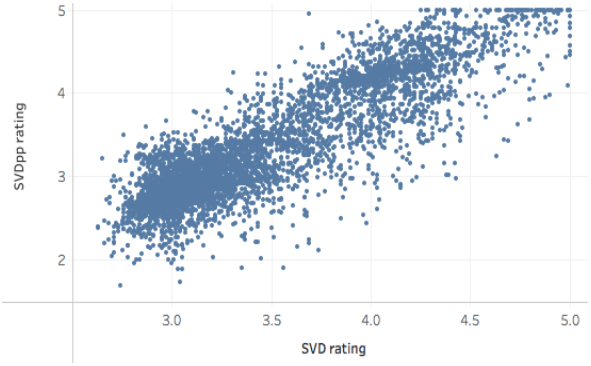

Fig. 8 Distribution of predicted ratings indicate SVD and SVDpp ratings for recommended courses. Note the range of SVD ratings (2.5,5), versus SVDpp ratings range (2,5). Correlation is 0.7

The difference in SVD and SVDpp algorithms was more obvious in score rankings.

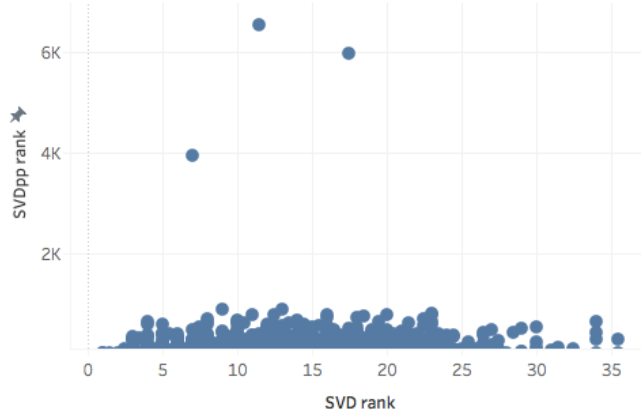

Fig. 9 Distribution of predicted ratings indicate SVD and SVDpp ranks for recommended courses. Note the range of SVD ranks (1,35), versus SVDpp ratings range (1, 6000). Correlation is 0.4. Also note outliers with SVDpp rank more than 4000

The maximum SVD rank was 35.5, but 25% of SVDpp rankings were between the range (36, 1000). Outliers were even observed, with ranks 4000 to 6000. If SVDpp rankings were used, these courses would not have been included in the recommendation.

V. Summary and recommendations

Collaborative filtering using SVD can be used in creating recommendations for courses, with the objective of standardizing the knowledge-base in the employee-cohort. With a pseudo-rating system with ordinal values, FCP is the better metric to use in evaluating models and tuning hyperparameters.

In the next iteration of this project, OrdRec method as proposed by Koren and Sill [4] can be used to generate recommendations. Ricci, Rokach, et al [6] also devoted a chapter discussing parameters on Recommender Systems in Technology Enhanced Learning. Within the project, A/B testing on the recommendations is suggested to validate recommendation validity.

Acknowledgement
We gratefully thank the faculty from Galvanize – Alessandro Gagliardi and Conor Murphy for their guidance during the development of the project. Gratitude goes to Wilson Lau of Orange Silicon Valley as well, for patiently working with us on the project.

References

[1] Min Gao, Yunqing Fu, Yixiong Chen, Feng Jiang. User Weight Model for Item-based Recommendation Systems, 2012

[2] Yehuda Koren and Robert Bell . Advances in Collaborative Filtering

[3] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. 2008. URL: http://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization Meets the Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model.pdf.

[4] Yehuda Koren and Joseph Sill. Collaborative Filtering on Ordinal User Feedback. URL: https://pdfs.semanticscholar.org/934a/729409d6fbd9894a94d4af66bd82222b5515.pdf

[5] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. 2008. URL: http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf.

[6] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. 1st edition, 2010. URL: http://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf