Joint Features Regression for Cold-Start Recommendation on VideoLectures.Net

Gokhan Capan, Ozgur Yilmazel

Department of Computer Engineering
Anadolu University, Eskişehir, Turkey
{gcapan, oyilmazel}@anadolu.edu.tr

Abstract. Recommender systems are popular information filtering systems used in various domains. Cold-start problem is a key challenge in a recommender system. In *new-item/existing-user* case of the cold-start problem, which is recommendation of a recently-arrived item to a user with historical data, finding links between existing items with recently-arrived items is critical. Using VideoLectures.net Cold-Start Recommendation Challenge data, this paper includes a linear regression model to predict future co-viewing count between an existing item and a recently-arrived, not-yet-viewed item.

1 Introduction

A recommender system produces user-specific subsets of a global item set, in which users are expected to be interested. These recommendations are computed by predicting user-item scores that are not known yet, generally based on:

- Implicit or explicit ratings of similar users on items that user has not rated yet.
- Content similarity between items that user has rated high and has not rated yet.

According to the algorithms listed above, a recommender system may either use a collaborative, content based, or hybrid approach to produce recommendations.

Although collaborative filtering systems are very successful when there are sufficient historical user-item data available, they cannot produce recommendations in any of the cold-start cases, which are recommending new items (an item that nobody has rated yet) to users, or recommending items to new users (a user with no historical ratings data).

Recommending existing items is not in this paper's scope, we will focus on new-item cases.

Finding *existing item-new item* links using *existing item-existing item* links provide scores to be used to discover next high-rated item after high-rating an existing item, which is an important signal for recommendation. Our method represents two items as one vector of a joint feature set, which is constructed by computing relationships between some of their content features (title, categories, authors, … in videolectures.net domain). Then we apply linear regression to predict which item would be consumed after current item most probably. This is a ranked list of candidate successor items for each existing item.

The rest of the paper is organized as follows: Section 2 is the related work, Section 3 describes the methods we have used and experiments we have done, Section 4 concludes the paper.

2 Related Work

There have been many collaborative filtering algorithms studied. These algortihms produce recommendations using:

- Neighborhood based methods [1, 2, 3]
- Latent Factor Models [4, 5, 6]
- Matrix Factorization based methods [7]

Neighborhood methods are the traditional collaborative filtering models, which used to be the dominant collaborative filtering method before matrix factorization techniques. Typically, a neighborhood based collaborative filtering algorithm finds nearest neighbors of items or users, according to historical rating data [8]. Using the neighborhood, the algorithm tries to predict unknown user-item ratings.

Latent factor models are based on representing both users and items in the same feature space, latent factors. Latent Semantic Models for Collaborative Filtering [9] is an example.

Matrix factorization for performing collaborative filtering (may be included in latent factor models) is based on factorizing ratings data, which is a user-item matrix. Singular Value Decomposition based recommenders are the popular factorization based methods [7].

The pure collaborative methods for recommender systems are not able to solve cold-start problems for new-item case. Generally, hybrid models are used to solve cold-start problems [10, 11]. There are some other approaches that fill the user-item matrix by generating ratings (with a bot, for example) [12].

Menon and Elkan's method for Dyadic Prediction (Recommendation and link prediction are examples of dyadic prediction) [13] introduces a log-linear model for discovering latent factors, where a dyad may be a user-item pair (recommendation), or either item-item or user-user pairs (link predicition). Their approach takes side information (different from just unique identifiers) into account, thus providing a solution to cold start problem.

Chu and Park's bilinear regression method for recommendation on dynamic content [14] also benefits from the static features of users (gender, for example) and items (bag of words, category, ...). Their method let them recommend very recent items to users, showing that it may be considered as a solution to cold start problem for the new item case.

Again, Park and Chu's pairwise preference regression method specifically addresses the cold start problem [15]. Their method represents a joint feature space for user/item pairs via outer products.

Park and Chu's pairwise preference regression method is the inspiring method for our solution for predicting links between existing items and new items. However, there are differences. Our solution focuses on constructing a joint feature space for item-item pairs. This feature space is not bilinear. Instead, it is constructed using relationships between two items on several content features. A transformed feature may be either numerical or categorical. For example, cosine similarity between titles of two items is a candidate numerical joint feature (title relationship) for the final dataset, whereas the language relationship (the language code if they are in the same language) is a categorical joint feature.

3 Methods and Experimental Study

3.1 VideoLectures.Net Data

VideoLectures.Net[1] is a repository for video lectures from scientists in various events. The algorithm is applied to the dataset provided by VideoLectures.Net Recommendation Challenge [16]. Listed below are some properties of the dataset:

1. **authors**: Contains data on authors registered on VideoLectures.Net
   attributes: id, name, gender, email, homepage
2. **author_lectures**: Information on which author authored what lecture (There is a many-to-many relationship between authors and lectures)
   attributes: author_id, lecture_id
3. **categories**: Information on categories in scientific taxonomy.
   attributes: id, name, parent_id
4. **categories_lectures**: Information on what lecture is categorized under which category. (There is a many-to-many relationship between categories and lectures
   attributes: category_id, lecture_id
5. **lectures_train**: contains a subset of lectures with publication date prior to 1.7.2009
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description, slide_titles, views
6. **lectures_test**: contains a subset of lectures published after 1.7.2009
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description, slide_titlesll
7. **pairs**: records about a pair of lectures viewed together with at least two distinct cookie identified browsers.
   attributes: lecture1_id, lecture2_id, frequency
8. **events**: contains information on events, which are basically a set of lectures grouped together.
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description
9. **task1_query**: Contains lecture ids from the subset of lectures_train, for which a ranked list of 30 recommended lectures from lectrues_test is expected.
   attributes: id

3.2 Methods and Experiments

To estimate the model that predicts co-viewing counts of *existing video-new video* pairs, we first transform the features of video pairs into one joint feature space. We find relationships between two videos on several content features, and use these relationships as resulting features. Attributes of the transformed data are:

- **type**: If two videos are in the same type (lecture, keynote, ...) the common type, 0 otherwise. This feature is categorical with 16 distinct values
- **language**: If two videos are in the same language the language code, 0 otherwise. This feature is categorical with 10 distinct values.

[1]http://www.videolectures.net

- **parent**: The parent property is hierarchical, that is, parents of videos (the events) also have parents. We have detected all parents of two lectures to the deepest level.
  The value of this feature is simply the jaccard similarity of the resulting sets.
- **title**: After the indexing process, total idf (inverse document frequency) of the common terms is the value for this feature. The reason we chose inverse document frequencies over term frequencies (or simply 0 or 1) is to make significant words more discriminative.
- **categories**: In the dataset, categories are also defined in a hierarchy. We have created category indexes for lectures to the deepest level. Because the top categories are very common for all videos, we applied the same strategy as we did while computing the value for the title feature. The total idf of the common categories of two lectures is the resulting value.
- **authors**: To increase the effect of 'same author's different lectures' if she has fewer lectures in the web site, we applied the total idf strategy again, for authors feature.
- **description**: We computed the value in the same way as we did for the title.
- **co-viewing count**: Score is the target value of the predictors defined so far, which is frequency of lecture pairs viewed together.

Each example in the transformed data representing a video pair contains a set of features, and a target variable (co-viewing count). The task of estimating a function to predict future co-viewing counts is a supervised learning, specifically regression problem. We have used linear regression with the simplest estimator; ordinary least squares. Linear regression with an intercept term estimates a function in the following format:

$$y = ß_0 + ß_1x_1 + ß_2x_2 + ... + ß_kx_k \quad (1)$$

where y is the target variable to be predicted, x_i's are regressor variables (features), and $ß_i$'s are the parameters to be learned. We want to estimate the parameters so that the resulting function minimizes the error, which is the difference between the observed (actual) and estimated y values. Ordinary least squares is the approach of minimizing the sum of squared errors. Finally, one can apply the estimated regression function to an arbitrary video pair to predict the number of times the videos will be watched together. Before applying the function, she need to transform the video-pair into our joint features space.

We used R programming language [17] to estimate the linear model. Because language and type attributes are categorical and have 10 and 16 distinct values, respectively; the resulting size of dimensions is 31, while there are 363880 training example.

To measure the relative importance of regressor variables on predicting the co-viewing count of two videos, we have used the approach Grömping has introduced [18]. Using the metric *last*, we have compared each regressor variable's contribution to accuracy when all other regressors are available. The top five regressor variables are shown in Table 1.

Table 1. Top 5 relatively important regressors

Feature	Importance
title	0.0065
categories	0.0043
parent	0.0040
description	0.0025
authors	0.0022

Finally, we have assigned values of common term frequencies to features where we have used inverse document frequencies previously, and run the same algorithm. The final evaluation results will show that using term frequencies decreases the prediction accuracy.

3.3 Evaluation

To pre-evaluate the model we have estimated, we applied 3-fold cross-validation to data. The cross validated standard error of estimate, which is the square root of the mean of MSE's of 3 folds, is computed as 23.4.

We have also analyzed the residuals, which is the difference between the actual value and estimated value of an observation, when 10% of the observations were used to test the model we have estimated from the remaining 90%. The quartiles of residuals may give an idea about its distribution. Table 2 shows the quartiles with minimum and maximum values of residuals we have computed, Figure 1 is a boxplot of quartiles with a range of 20. The quartiles show that 50% of residuals are between -4.14 and -0.27.

Table 2. Quartiles of residuals

Minimum	1st quartile	2nd quartile (median)	3rd quartile	Maximum
-151	-4.14	-2.22	-0.27	3170

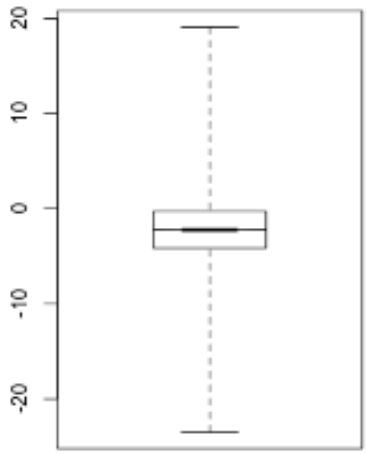

Fig. 1. Boxplot of quartiles of residuals

Goal of VideoLectures.Net Recommender System Challenge Task 1 is finding a ranked list of potential next lectures for each lecture in *task1_query* data, by retrieving a ranked sublist from the test set of lectures. The challenge can be considered as an information retrieval problem, and they have defined an R-precision variants of precision at K and MAP, which are standard evaluation measures used in information retrieval [19].

To find a ranked list of recommended lectures for a given lecture in *task1_query* data, we have paired the lecture with each possible test lecture from *lectures_test* data, computed predicted values, ranked them according to the scores we have computed, and submitted the top ranked 30 candidate lectures. The evaluation score is computed as 0.2492. The experiments also show that choosing term frequencies instead of inverse document frequencies decreases the predicting performance. In this case, the evaluation score is computed as 0.2266. The final evaluation score, 0.2492, can be considered high, ranked 7th among 1656 submitted solutions from 62 active teams of 303 registered teams with 346 members.

4 Conclusion

In this paper, we have described our solution to predict item-to-item link scores (co-viewing counts, in this case) to solve cold-start problem in recommender systems using VideoLectures.Net Recommender System Challenge data. We have used ordinary least squares linear regression to predict scores. The results show that the method of defining joint features and applying regression on transformed data provides us simple and accurate results in relatively small dimensions.

However, we have only tried ordinary least squares linear regression, which may not be the best model for the problem; as a future work other regression methods, especially the non-linear models may be applied to the problem. In addition, more features may be defined, and an efficient feature selection method may be applied to data. We also left this process as a future work.

References

1. Shardanand, U. and Maes, P.: Social Information Filtering: Algorithms of automating “word of mouth”, Proceedings of the SIGCHI Conference on Human Factors in Computer Systems. (1995)
2. Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J.: Item-based Collaborative Filtering Recommender Algorithms, WWW, pp. 285-295 (2001)
3. Herlocker, J. L., Konstan, J. A., Borhcers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering, Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (2002)
4. Billsus, D., Pazzani, M. J.: Learning Collaborative Information Filters, Proceedings of the 15th International Conference on Machine Learning, pp. 46-54. (1998)
5. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm, Information Retrieval, vol. 4 issue 2, pp. 133-151 (2001)
6. Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J.: Application of dimensionality reduction in recommender systems – a case study, ACM WebKDD Workshop. (2000)
7. Sarwar, B. M., Karypis, G., Konstan, J., Riedl, J.: Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. Fifth International Conference on Computer and Information Technology. (2002)
8. Koren, Y. and Bell, R.: Advances in Collaborative Filtering, Recommender Systems Handbook, pp. 145-186. Springer (2011)
9. Hoffman, T.: Latent Semantic Models for Collaborative Filtering. ACM Transactions on Information Systems, vol. 22 issue 1, pp. 89-115. ACM (2004)
10. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M.: Combining content-based and collaborative filters in an online newspaper. ACM SIGIR Workshop on Recommender Systems. (1999)
11. Melville, P., Mooney, R., Nagarajan, R.: Content-boosted Collaborative Filtering for Improved Recommendations. Proc. of the National Conf. on Artificial Intelligence, pp. 187-192. AAAI (2002)
12. Park, S. T., Pennock, D. M., Madani, O., Good, N., and DeCoste, D.: Naïve filterbots for robust cold-start recommendations., Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2006)
13. Menon, A. K., Elkan, C.: Dyadic Prediction Using a Latent Feature Log-Linear Model. CoRR (2010)
14. Chu, W., Park, S.: Personalized Recommendation on Dynamic Content Using Predictive Bilinear Models. Proceedings of the 18th International Conference of World Wide Web. ACM (2009)
15. Chu, W., Park, S.: Pairwise Preference Regression for Cold-Start Recommendation. Proceedings of the Third ACM Conference on Recommender Systems. ACM (2009)
16. Antulov-Fantulin, N., Bošnjak, M., Šmuc, T., Jermol, M., Žnidaršič, M., Grčar, M., Keše, P., Lavrač, N.: ECML/PKDD 2011 - Discovery challenge: "VideoLectures.Net Recommender System Challenge", http://tunedit.org/challenge/VLNetChallenge
17. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, http://www.R-project.org. (2011)
18. Grömping, U: Relative Importance for Linear Regression in R: The Package relaimpo. Journal of Statistical Software, vol. 17, pp. 1-27. (2006)
19. Buckley, C., Voorhees, E. M.: Evaluating evaluation measure stability. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (2000)