Recommendation with the Right Slice: Speeding Up Collaborative Filtering with Factorization Machines

Babak Loni[1], Martha Larson[1], Alexandros Karatzoglou[2], Alan Hanjalic[1]
[1]Delft University of Technology, Delft, Netherlands
[2]Telefonica Research, Barcelona, Spain
{b.loni, m.a.larson}@tudelft.nl, alexk@tid.es, a.hanjalic@tudelft.nl

ABSTRACT

We propose an alternative way to efficiently exploit rating data for collaborative filtering with Factorization Machines (FMs). Our approach partitions user-item matrix into 'slices' which are mutually exclusive with respect to items. The training phase makes direct use of the slice of interest (*target* slice), while incorporating information from other slices indirectly. FMs represent user-item interactions as feature vectors, and they offer the advantage of easy incorporation of complementary information. We exploit this advantage to integrate information from other *auxiliary* slices. We demonstrate, using experiments on two benchmark datasets, that improved performance can be achieved, while the time complexity of training can be reduced significantly.

1. INTRODUCTION

In this paper, we investigate the idea that the 'right' data, rather than all data, should be exploited to build an efficient recommender system. We introduce an approach called 'Slice and Train' that trains a model on the right slice of a dataset (a 'sensible' subset containing the current items that need to be recommended) and exploits the information in the rest of the dataset indirectly. This approach is particularly interesting in scenarios in which recommendations are needed only for a particular subset of the overall item set. An example is an e-commerce website that does not generate recommendations for out-of-season or discontinued products. The obvious advantage of this approach is that models are trained on a much smaller dataset (only the data in the slice), leading to shorter training times.

The 'Slice and Train' approach also has, however, a less expected advantage, namely, that it offers highly competitive performance with conventional approaches that make use of the whole dataset, and in some cases even improves the performance. This advantage means that it is helpful to apply 'Slice and Train' even in cases in which predictions are needed for all items in the dataset, by training a series of separate models, one for each slice.

Our approach consists of two steps: first the data is partitioned into a set of slices containing mutually exclusive items. The slices can be formed by grouping items based on their properties (e.g., the category of item), availability, item), or by more advanced slicing methods such as clustering. In the second step the model is trained using the samples in the slice of interest, i.e., *target* slice, while other slices are indirectly being exploited as *auxiliary* slices. To efficiently exploit information from auxiliary slices our approach trains the recommender model using Factorization Machines [4].

RecSys 2015 Poster Proceedings, September 16-20, 2015, Austria, Vienna.

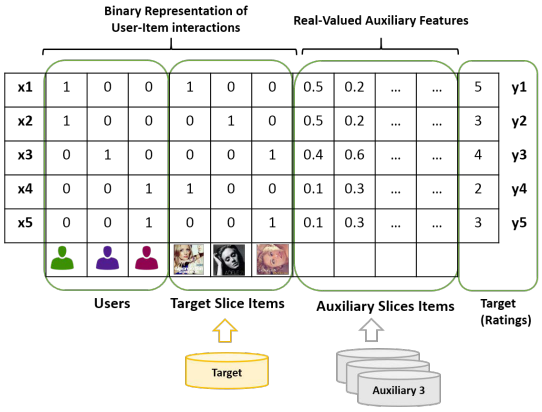

Figure 1: Feature construction in the 'Slice and Train' method.

Factorization Machines (FMs) generate recommendations by working with vector representations of user-item data. A benefit of these representations is that they can easily be extended with additional features. Such extensions are usually used to leverage additional information [5], or addition domains [3], to improve collaborative filtering. Here, the 'additional' information is actually drawn from different slices within the same dataset.

2. THE SLICE AND TRAIN METHOD

The 'Slice and Train' method is implemented with Factorization Machines [4]. FMs are general factorization models which can be easily adopted for different scenarios without requiring to adopt specific models and learning algorithms. In a rating prediction scenario with FMs, user-item interactions are represented by a feature vector $\mathbf{x}$ and the rating is taken as the output y. By learning the model, the rating y can be predicted for unknown user-item interactions. In other words, if user u rated item i the feature vector $\mathbf{x}$ can be represented by its sparse binary representation as $\mathbf{x}(u,i) = \{(u,1),(i,1)\}$, where non-zero elements correspond to user u and item i.

The 'Slice and Train' method creates feature vectors $\mathbf{x}$ only for the ratings in the target slice (i.e., the slice of interest). The rating information from the other slices is exploited indirectly by extending the feature vectors that are created for the target slice. Using this method, the accuracy of the recommender is preserved, while the training time is significantly reduced, since the number of samples in the target slice is lower than in the original dataset.

Figure 1 shows how the feature vectors in the 'Slice and Train' method are constructed. The feature vectors have a binary part, which reflects the corresponding user and item of a rating, and a

real-valued auxiliary part, which is constructed by using the rating information of auxiliary slices.

To understand how the auxiliary features are built, assume that the dataset is divided into m slices $\{S_1, \ldots, S_m\}$. Let us also assume that the items that are rated by user u in slice S_j is represented by $s_j(u)$. By extending the feature vector $\mathbf{x}(u,i)$ with auxiliary features, we can represent it with the following sparse representation:

$$\mathbf{x}(u,i) = \{\underbrace{(u,1),(i,1)}_{\text{target slice}}, \underbrace{\mathbf{z}_2(u),\ldots,\mathbf{z}_m(u)}_{\text{auxiliary slices}}\} \quad (1)$$

where $\mathbf{z}_j(u)$ is sparse representation of auxiliary features from slice j and is defined as:

$$\mathbf{z}_j(u) = \{(l, \phi_j(u,l)) : l \in s_j(u)\} \quad (2)$$

where $\phi_j(u,l)$ is a normalization function that defines the value of the auxiliary features. We define ϕ_j based on the ratings that user u gave to items in slice j and normalize it based on the average value of user ratings as follows:

$$\phi_j(u,l) = \frac{r_j(u,l) - \bar{r}(u)}{r_{max} - r_{min}} + 1 \quad (3)$$

where $r_j(u,l)$ indicates the rating of user u to item l in slice j, $\bar{r}(u)$ indicates the average value of user ratings, and r_{max} and r_{min} indicate the maximum and minimum possible values for rating items.

3. DATASET AND EXPERIMENTS

In this work we tested our method on two benchmark datasets of MovieLens 1M dataset[1] and Amazon reviews dataset [1]. The Amazon dataset contains product ratings in four different groups of items namely books, music CDs, DVDs and Video tapes. We use these four groups as natural slices that exist in this dataset. For the MovieLens dataset we build slices by clustering movies based on their genres using a k-means clustering algorithm. Various numbers of clusters (i.e., slices) can be made for this dataset. Via exploratory experiments we found that two or three slices perform well.

In order to test our approach, the data in target slices are divided into 75% training and 25% test data. For every experiment 10% of training data is only used as validation data to tune the hyper-parameters. Factorization Machines can be trained using three different learning methods [4]: Stochastic Gradient Decent (SGD), Alternating Least Square (ALS) and Markov Chain Monte Carlo (MCMC). In this paper we only report the results using MCMC learning method due to space limitation and since it usually perform better than the other two methods.

The experiments are implemented using WrapRec [2] and LibFM [4] open source toolkits. Furthermore, we compare the performance of FM with Matrix Factorization (MF) to ensure our FM-based solution is competitive with other state-of-the-art methods. We used an implementation of MF in MyMediaLite[2] toolkit.

Table 1 presents the performance of our sliced training method compared with the situation that the no slicing is done. The experiments are evaluated using RMSE and MAE evaluation metrics. The reported results are the averaged metrics over all slices when each of them is considered as the target slice. The four different setups that are compared are the followings:

- **FM-ALL**: FM applied on the complete dataset.
- **MF-ALL**: MF applied on the complete dataset.
- **FM-SLICE**: FM applied on independent slices. No auxiliary features are used for this setup.

[1]http://grouplens.org/datasets/movielens/

[2]http://www.mymedialite.net

Table 1: The performance of the proposed 'Slice and Train' method compared to other experimental setups.

Eval. Metric	RMSE		MAE	
Dataset	**Amazon**	**ML**	**Amazon**	**ML**
FM-ALL	1.1610	0.8894	0.9533	0.8387
MF-ALL	1.2927	0.8939	1.014	0.8411
FM-SLICE	1.2330	0.8974	0.9915	0.8427
FM-SLICE-AUX	1.0539	0.8644	0.9017	0.8260

- **FM-SLICE-AUX**: FM applied to slices with feature vectors extended by auxiliary features derived form auxiliary slices.

The first two lines of Table 1 report results when all the data are used. The effectiveness of FM as our model is confirmed when we compare FM-ALL baseline with MF-ALL, the Matrix-Factorization-based method. Comparing the remaining lines yields to some interesting insights. Not surprisingly, when we train the model only on a target slice without using any auxiliary information (FM-SLICE) the performance of the model drops. This drop can be attributed to the smaller number of samples that is being used when training. However, when the auxiliary feature are exploited with the 'Slice and Train' method (FM-SLICE-AUX) the performance becomes even better than the situation where the complete data is being used (FM-ALL). We attribute this improvement to the items in the slices being more homogeneous than the dataset as a whole. However, relying on homogeneity alone does not lead to the best performance (FM-SLICE). Instead the best performance is achieved when slicing is combined with auxiliary features. Furthermore, the time-complexity of training is significantly reduced when the model is trained on a target slice. In our setup the training time of one slice including auxiliary features compared to the training time of complete dataset falls from 7431 ms to 1605 ms for Amazon dataset and from 14569 ms to 6574 ms for the MovieLens dataset.

4. CONCLUSION

In this paper, we presented a brief overview of the 'Slice and Train' method, which trains a recommender model on a sensible subset of items (target slice) using Factorization Machines, and exploits other information indirectly. This sort of targeted training yields improvements in both performance and complexity.

Acknowledgment

This work is supported by funding from EU FP7 project under grant agreements no. 610594 (CrowdRec)

5. REFERENCES

[1] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), 2007.

[2] B. Loni and A. Said. Wraprec: An easy extension of recommender system libraries. In *Proceedings of 8th ACM International Conference of Recommender Systems*, 2014.

[3] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *Proceedings of the 36th European Conference on Information Retrieval*, ECIR '14, 2014.

[4] S. Rendle. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.*, 3(3), May 2012.

[5] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11. ACM, 2011.