Local Sparse Linear Model Ensemble for Top-N Recommendation

Ziwei Fan* Xia Ning*

Abstract

In this paper, we develop local sparse linear model ensemble to tackle both the data sparsity and the user/item heterogeneity issues for top-N recommendation. In specific, we learn multiple local sparse linear models for all the users and items in the system. These models are then combined in various ways to produce top-N recommendations. We develop various ways to select training data for each local model, and different methods to combine local models/results. Our experiments demonstrate at 18.4% improvement from such ensemble models particularly on sparse datasets.

1 Introduction

Top-N Recommender Systems (RS) have been widely used in E-commerce applications. However, two typical issues still challenge the current top-N RS development: 1). data sparsity, when there are not sufficient data to train a good RS model, and 2). user/item heterogeneity, when a global model (e.g., the popular matrix factorization models) trained from all the users/items fail for certain users/items. Existing methods that tackle the first issue include factorized models [5] and implicit feedback based models [4], etc. Emerging methods dealing with the second issue include the most recent local model based approaches [6, 2].

In this paper, we develop local sparse linear model ensemble to tackle both the data sparsity and the user/item heterogeneity issues. We learn multiple local Sparse LInear Models (`SLIM`) [7] for all the users and items in the system. These models are then combined in various ways to produce top-N recommendations. `SLIM` is strong in learning relations among items, while localizing `SLIM` with respective to certain users and items better reveal localized item relations among a certain group of users. By combining multiple `SLIM` models, signals from multiple models are aggregated so as to enable better results for sparse data. Our experiments over datasets of different sparsity levels demonstrate the superior performance of the model ensemble method.

2 Related Work

2.1 Sparse Linear Method for top-N Recommendation Ning and Karypis proposed a state-of-the-art Sparse LInear Method (`SLIM`) for top-N recommendation [7]. In `SLIM`, the user u's preference over an item i is modeled as a linear aggregation over the items that the user purchased before, that is,

$$\tilde{r}_{u,i} = R(u,\cdot)W(\cdot,i), \tag{2.1}$$

where $\tilde{r}_{u,i}$ is the estimated user preference of user u on item i, $R(u,\cdot)$ is the user preference over other items, and $W(\cdot,i)$ is coefficient with respect to item i. To solve for W, `SLIM` solves the following optimization problem,

$$\begin{aligned} \min_{W} \quad & \frac{1}{2}\|R - RW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_{\ell_1} \\ \text{s.t.} \quad & W \geq 0, \mathrm{diag}(W) = 0. \end{aligned} \tag{2.2}$$

2.2 Local Low-Rank Matrix Approximation Lee *et al.* [6] developed a Local Low-Rank Matrix Approximation (`LLORMA`) method. `LLORMA` first randomly selects a set of K anchor pairs $\{(u_k^*, i_k^*)\}$ ($k = 1, \cdots, K$). With respect to each anchor pair (u_k^*, i_k^*), a local model is learned using the training data $\{(u,i)\}$ that are selected based on user-item kernel values $\mathcal{K}((u,i),(u_k^*,i_k^*))$. The local models are low-rank matrix factorization models, that is,

$$\tilde{r}_{u,i}^k = {\mathbf{u}_u^k}^\mathsf{T}\mathbf{v}_i^k, \tag{2.3}$$

where $\mathbf{u}_u^k$ and $\mathbf{v}_i^k$ are the latent factors for user u and item i from the k-th model, respectively. The global model prediction $\tilde{\tilde{r}}_{u,i}$ of user u's preference over item i is a weighted combination of the predictions from multiple local models as follows:

$$\tilde{\tilde{r}}_{u,i} = \sum_{k=1}^{K} \frac{\mathcal{K}((u,i),(u_k^*,i_k^*))}{\sum_{k'=1}^{K}\mathcal{K}((u,i),(u_{k'}^*,i_{k'}^*))}\tilde{r}_{u,i}^k \tag{2.4}$$

2.3 Combining Local Models for Recommendation The idea of combining local models for recommendation has attracted increasing attention recently. For example, Xu *et al.* [8] first cluster users and items into user-item groups. Within each user-item group, collaborative filtering s applied to generate recommendations. Items with the highest recommendation scores are recommended. Beutel *et al.* [1] first cluster users and items, respectively. Then they fit models on the user-item clusters using some mean values. The residuals from this clustering-mean fitting process are forwarded to the next iteration of same processes. Christakopoulou and

*Indiana University-Purdue University Indianapolis, {zifan@umail.iu.edu, xning@iupui.edu}

Karypis [2] combine local models and a global model. The user membership in each local model is re-assigned after each iteration of model updates. Local models and the global model are then further updated to adapt to the new user assignment.

3 Methods

We developed the model ensemble over local sparse linear models for top-N recommendation. Following the idea from Lee [6], a set of anchor pairs is first selected. With respect to each of the anchor pairs, a local `SLIM` model is trained. The local models are then ensembled via combining their results or combining the models directly.

3.1 Anchor Pair Selection We first randomly select a user u^* out of m users as the anchor user. Then from u^*, we randomly select an item i^* that u^* has purchased. This item will be the anchor item. The user-item pair (u^*, i^*) will be the anchor pair that each local model will be built with respect to.

3.2 Training Data Selection for Local Models Training data for each local model with respect to each anchor pair (u^*, i^*) are selected according to: 1). user and item similarities, and 2). user and item popularities.

3.2.1 Similarity-based Training Data Selection In this method, we use a radial basis function (RBF) kernel to measure the similarity between a user-item pair (u', i') and the anchor pair (u^*, i^*), and apply two different similarity-based schemes to select training data. The RBF kernel over the two pairs is defined as follows [6]:

$$\mathcal{K}_{ui}((u', i'), (u^*, i^*)) = \mathcal{K}_u(u', u^*) \times \mathcal{K}_i(i', i^*) \tag{3.5}$$

where both $\mathcal{K}_u$ and $\mathcal{K}_i$ are RBF kernels:

$$\mathcal{K}_u(u', u^*) = \exp(-\gamma \|R(u', \cdot) - R(u^*, \cdot)\|^2), \tag{3.6}$$

$$\mathcal{K}_i(i', i^*) = \exp(-\gamma \|R(\cdot, i') - R(\cdot, i^*)\|^2). \tag{3.7}$$

In Equation 3.6 and Equation 3.7, $R(u, \cdot)$ ($R(\cdot, i)$) is purchase profile of by user u (of item i). Since both $\mathcal{K}_u$ and $\mathcal{K}_i$ are valid kernels, $\mathcal{K}_{ui}$ is also a valid kernel. The definition in Equation 3.6 and Equation 3.7 follow the idea in user-based and item-based Collaborative Filtering (CF), respectively, that calculates user and item similarities directly from user-item matrix R.

Given the similarities, all the user-item pairs are weighted by their similarities with the anchor pair (u^*, i^*) as follows,

$$R^{\mathcal{K}_{ui}}_{u^*, i^*}(u', i') = \mathcal{K}_{ui}((u', i'), (u^*, i^*))R(u', i'), \tag{3.8}$$

and used for local model training. This data selection method is referred to as $\mathtt{S}_{\mathcal{K}_{ui}}$.

3.2.2 Popularity-based Training Data Selection In this method, we select the user-item pairs such that the selected users/items have similar popularities as the anchor user/item. The user popularity is defined as the number of items that the user has purchased, and the item popularity is defined as the number of users who have purchased the item. For each anchor pair (u^*, i^*), we first select $\alpha\%$ of all the users who have closest but lower or higher popularity than u^*, respectively. From the selected users, we select $\alpha\%$ of all the items that have closest but lower or higher popularity than i^*, respectively. The interactions between the selected users and items will be used as training data. This training data selection method is referred to as $\mathtt{S}_{\mathcal{P}_{ui}}$.

3.3 Model Combination and Recommendation Generation After training a `SLIM` model on each of K selected training datasets with respect to K anchor pairs, we ensemble the model results or the models themselves in order to produce recommendations.

3.3.1 Model Result Aggregation Each local model first produces a recommendation list. Each of items that has ever appeared in any of the K recommendation lists is then scored. These items are re-ranked using the scores into a new ranked list and the top-N items in the new list will be recommended. This result-aggregation based method is referred to as `MRA`.

To score the items, we use the following two approaches. The first one is the Borda [3] approach, which scores each item using the sum of their ranking positions from all the recommendation lists. The Borda scoring approach is denoted as $\mathtt{C}_\mathtt{r}$.

The second scoring approach is to use the weighted sum of recommendation scores from all the recommendation lists. In specific, the score of a user u on an item i, denoted as $\tilde{r}_{u,i}$, is calculated using Equation 2.4 as in `LLORMA`. This scoring approach is denoted as $\mathtt{C}_\mathtt{s}$.

3.3.2 Linear `SLIM` Model Ensemble Instead of combining recommendation results from local models, we can also combine models directly. In the case of `SLIM`, the coefficient matrices from local models are linearly combined as follows:

$$W_{i,j} = \frac{1}{|\{k | W^k_{i,j} \neq 0, k = 1, \cdots, K\}|} \sum_{k=1}^{K} W^k_{i,j}, \tag{3.9}$$

where W^k is the k-th model (coefficient matrix) with respect to the k-th anchor pair, $\{W_{i,j} | W^k_{i,j} \neq 0, k = 1, \cdots, K\}$ is the set of coefficients in which $W^k_{i,j} \neq 0$, $|\cdot|$ is the cardinality of a set. We use W^e to produce recommendations as in Equation 2.1.

Note that only a certain portion of W^k that corre-

sponds to the selected training items can have non-zero values. Thus, the linear combination of multiple W^k's resembles using multiple small plates to approximate a manifold. Thus, it may represent non-linear relations among items. This method is referred to as LSME.

4 Materials

4.1 Datasets We evaluated the different methods on a benchmark dataset ML100K[1], and its sparsified versions. From the original dataset (referred to as ML100K-1), we generated three sparsified datasets (referred to as ML100K-2, ML100K-3 and ML100K-4, respectively). The first sparsified dataset MK100K-2 is generated by randomly selecting 50% of purchases from ML100K-1, The second/third sparsified dataset ML100K-3/ML100K-4 is generated by randomly selecting 50% of purchases from ML100K-2/ML100K-3. Table 1 represents the dataset summaries.

Table 1: Datasets Used in Evaluation

dataset	#users	#items	#ratings	rsize	csize	density
ML100K-1	943	1,682	100,000	106.05	59.45	6.30%
ML100K-2	943	1,682	49,760	52.77	29.58	3.14%
ML100K-3	943	1,682	24,647	26.14	14.65	1.55%
ML100K-4	943	1,682	12,086	12.82	7.19	0.76%

Columns of "#users", "#items" and "#ratings" represent the number of users, items and ratings in the datasets, respectively. Columns of "rsize" and "csize" represent the average number of ratings for each user and each item, respectively. Column of "density" represents the density of each dataset (i.e., density = #ratings/(#users × #items)).

4.2 Evaluation Methodology and Metrics We applied 5-time Leave-One-Out cross validation (LOOCV) to evaluate the performance of different methods. In each run, one of the purchases of each user is randomly selected into the testing set, and the remaining purchases are used in the training set. For each user, a size-N ($N = 10$ by default) ranked list of items is recommended from the ensemble model trained using the training set. The evaluation is performed by comparing the recommendations for each user and the left-out item of that user in the testing set. We use Hit Rate (HR) and the Average Reciprocal Hit-Rank (ARHR) [7] as the evaluation metrics. HR is defined as the rate of correctly recommended items, that is,

$$\text{HR} = \frac{\#\text{hits}}{\#\text{users}}, \tag{4.10}$$

where #users is the total number of users in the testing set, and #hits is the number of users who have their testing items correctly recommended (i.e., hit). ARHR is a weighted version of HR defined as follows:

$$\text{ARHR} = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i} \tag{4.11}$$

where if an item of a user is hit, p_i is the position of the item in the ranked recommendation list. Higher HR and ARHR values indicate better performance.

5 Experimental Results

5.1 Overall Performance Table 2 presents the best performance of the methods on the four datasets. SLIM significantly outperforms other methods on ML100K-1 in HR. However, when the datasets become sparser, LSME and MRA significantly outperform SLIM. In specific, LSME with $\mathtt{S}_{\mathcal{P}_{ui}}$ as the training data selection method outperforms SLIM on ML100K-2 at 1.89%, and on ML100K-3 at 2.97%. MRA with $\mathtt{S}_{\mathcal{K}_{ui}}$ and $\mathtt{C_s}$ outperforms SLIM on ML100K-2 at 1.26%, on ML100K-3 at 14.9% and on MK100K-4 at 18.4%. This demonstrates that the ensembled based methods are superior in learning from sparser datasets for top-N recommendation.

Among the four methods, MRA-$\mathtt{S}_{\mathcal{K}_{ui}}$-$\mathtt{C_r}$ has the worst performance overall but MRA-$\mathtt{S}_{\mathcal{K}_{ui}}$-$\mathtt{C_s}$ has the best. The difference may stem from the recommendation result scoring and combination scheme $\mathtt{C_r}$ and $\mathtt{C_s}$. The $\mathtt{C_r}$ method scores recommendations based on their positions in multiple ranked lists, and thus treats the multiple local models and their recommendations equally. The $\mathtt{C_s}$ scores recommendations using a weighted sum of their respective recommendation scores from local models, and therefore is able to differentiate local models based on their recommendation qualities.

The LSME does not perform as well as MRA based methods. This may be due to the fact that when LSME combines multiple local models as in Equation 3.9, the qualities of local models and their respective significance are not considered. We will investigate this aspect in our future work.

6 Discussions and Conclusions

6.1 Computational Consideration For LSME and MRA based methods, training multiple local models is computationally expensive. However, the training process can be trivially paralleled, that is, each local model can be trained independently and in parallel with others. In addition, the training process for each local model is in principle faster than the baseline SLIM model over a same dataset. This is because each local model either has smaller values (e.g., selected by $\mathtt{S}_{\mathcal{K}_{ui}}$) or has fewer training data (e.g., selected by $\mathtt{S}_{\mathcal{P}_{ui}}$). Thus, the model training for LSME and MRA based models can be even faster than SLIM. For example, for ML100K-

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

Table 2: Performance Comparison

dataset	SLIM				LSME-$S_{\mathcal{P}_{ui}}$					
	β	λ	HR	ARHR	$\alpha(\%)$	n	β	λ	HR	ARHR
ML100K-1	10	1e-3	**0.339**	0.154	40.0	50	1e-1	5e-0	0.326	0.142
ML100K-2	20	1e-0	0.159	0.056	30.0	60	10	1e-0	**0.162**	0.057
ML100K-3	20	1e-1	0.101	0.036	40.0	50	10	1e-1	0.104	0.036
ML100K-4	25	1e-3	0.049	0.016	40.0	60	10	1e-4	0.047	0.014

dataset	MRA-$S_{\mathcal{K}_{ui}}$-C_r					MRA-$S_{\mathcal{K}_{ui}}$-C_s				
	n	β	λ	HR	ARHR	n	β	λ	HR	ARHR
ML100K-1	50	1e-1	1e-7	0.255	0.097	20	1	1e-5	0.273	0.122
ML100K-2	50	5	1e-2	0.106	0.034	80	2	1e-5	0.161	0.057
ML100K-3	50	15	1e-1	0.093	0.029	20	10	1e-2	**0.116**	0.040
ML100K-4	70	20	1e-1	0.046	0.013	5	25	1e-4	**0.058**	0.020

Columns of "β" and "λ" present the parameters for the local SLIM models. Column of "n" represents the number of local models. Column of "$\alpha(\%)$" represents the percentage of users/items selected for training. Columns of "HR" and "ARHR" present the hit rate and average reciprocal hit-rank, respectively. LSME-$S_{\mathcal{P}_{ui}}$, MRA-$S_{\mathcal{K}_{ui}}$-C_r and MRA-$S_{\mathcal{K}_{ui}}$-C_s represent the combinations of different model ensemble, training data selection and recommendation combination schemes. **Bold** numbers are the best performance in terms of HR for each dataset.

1, SLIM takes 75.47 seconds for model training, but a parallel implementation of MRA-$S_{\mathcal{K}_{ui}}$-C_s could take 26.72 seconds.

6.2 Parameter Selection In principle, each local model should have its own optimal parameters. However, this will lead to a huge set of parameters that each LSME and MRA based models need to identify. To avoid this complexity, we use same parameters for all the local models. We will investigate heuristics to identify optimal parameters for local models and thus further improve the performance of LSME and MRA.

6.3 Conclusions We developed multiple LSME and MRA based methods to build local SLIM models and ensemble local models for better top-N recommendation. To select training data for each local model, we developed $S_{\mathcal{K}_{ui}}$ and $S_{\mathcal{P}_{ui}}$ methods, which select training data based on user-item similarities and popularities with respect to anchor pairs, respective. To combine local models, we developed LSME, which combines models (coefficient matrices in local SLIM models) in a linear fashion, and MRA, which combines local model recommendations based on recommendation orders and scores, respectively. Our experiments demonstrate significant improvement from such ensemble models particularly on sparse datasets.

Acknowledgment This work was supported by the National Science Foundation IIS-1566219 and DRIVE, IUPUI.

References

[1] Alex Beutel, Amr Ahmed, and Alexander J Smola. Accams: Additive co-clustering to approximate matrices succinctly. In *Proceedings of the 24th International Conference on World Wide Web*, pages 119–129. ACM, 2015.

[2] Evangelia Christakopoulou and George Karypis. Local item-item models for top-n recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 67–74. ACM, 2016.

[3] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

[4] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Dec 2008.

[5] Santosh Kabbur, Xia Ning, and George Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 659–667, New York, NY, USA, 2013. ACM.

[6] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[7] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[8] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. ACM, 2012.