DS-GA 1004 Goodreads Recommender System

Man Jin (mj1637), Mufeng Gao (mg5381), Mu Li (ml4844)

May 10, 2020

1 Introduction

In this project, we build up a matrix factorization based recommender system for explicit feedback using the Alternating Least Squares (ALS) algorithm. The dataset we use is Goodreads datasets[1][2], which contains $\sim$229m records of user-book interaction data along with meta-data of books. With $\sim$10% of data, our model achieves Precision@500 of 0.00293 and Mean Average Precision of 0.00045 on the testing data. We also implement a single-machine recommendation algorithm using LightFM[3], and explore the ALS learned representation for books using additional genre information through data visualization.

2 Dataset

The Goodreads datasets[1][2] contains 228,648,342 records of user-book interactions, which includes 876,145 distinct users and 2,360,650 distinct books. In addition, the datasets also contain meta-data of books such as genres and author's information. For the explicit feedback recommender system, we mainly focus on `user_id`, `book_id` and `ratings` of the User-Book Interactions Data in the datasets. Data description can be found in Appendix A. We use all interaction records without any filtering based on `is_read` and `is_reviewed` attributes. We also keep all interactions where rating is 0 (not provided) as we believe these interactions provide some information about users' preferences, and these 0 ratings have minor impact on our evaluation metrics of the recommender system.

3 Data Subsampling, Filtering and Splitting

Firstly, as there are over 200 million records in the original interaction dataset, we subsample the data and gradually increase the data sizes used for fitting the models. To keep the integrity of the interactions for each user, we downsample on user ID, and extract all interactions associated with each user sample. The sample sizes we choose are 1%, 5% and 10% over 876,145 users in the entire interaction dataset.

Secondly, since Alternating Least Squares (ALS) is a matrix factorization based algorithm, so it could be hard to learn an accurate representation when we have very few non-zero entries for a user/item in the utility matrix. And in practice, the cold-start problem can be solved with additional meta-data before we collect more data on a user/item. To learn better representations of users and items, we apply 2 separate filters on items and users sequentially, filtering out books with no more than 5 ratings and users with no more than 30 interactions.

Lastly, we randomly split all distinct users in the subsample into training, validation and test sets along with all their interactions by setting the proportion of distinct users in each set to be 3: 1: 1. The ALS model without a cold-start strategy cannot deal with new users or items at prediction time. Therefore, for users in the validation/test set, we move half of their interactions to the training set. We then remove records with books not observed during training from the validation and test sets. Statistics of three subsample after data preprocessing can be found in Appendix B.

4 ALS Recommender System

4.1 ALS Introduction

Recommender systems aim to estimate the blank entries in the utility matrix, where each entry (i, j) represents the rating of i^{th} user for j^{th} item. The utility matrix is sparse as most of the feedback is unknown, and one method to recover the utility matrix is Alternating Least Squares (ALS). ALS estimates the blank entries of the utility matrix by factorizing the utility matrix into the product of two matrices U and V, and approximates U and V iteratively using least squares.

4.2 Evaluation Metric

4.2.1 Evaluation Metric Setup

Suppose we have M users $\{u_0, \ldots, u_{M-1}\}$, and for each user u_i, there are N_i ground truth labels $D_i = \{d_0, \ldots, d_{N_i-1}\}$, and Q_i recommendations in decreasing relevance $R_i = \{r_0, \ldots, r_{Q_i-1}\}$. Define an indicator variable $rel_{D_i}(r)$, where $rel_{D_i}(r) = \begin{cases} 1 & \text{if r} \in D_i \\ 0 & \text{otherwise} \end{cases}$

$$\text{Mean Average Precision} = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{N_i} \sum_{j=0}^{Q_i-1} \frac{rel_{D_i}(R_i(j))}{j+1}$$

The order of predicted confidence is taken into consideration while computing MAP score, where penalty for highly relevant documents is higher. Moreover, MAP takes average over all users, and for each user, the weighted indicator variable is normalized by the cardinality of the ground truth set. Therefore, it is applicable when the cardinality of the ground truth set is less than k (500 in our project), where the precision at 500 score can be biased. Thus, we choose MAP as our metric for hyper-parameter tuning.

$$\text{Precision at k} = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{k} \sum_{j=0}^{\min(Q_i,k)-1} rel_{D_i}(R_i(j))$$

To the contrary of MAP, Precision at k ignores the ranking of recommendations, and it only measures how many of the first k recommended items are correct. Note that the sum of indicator variables are always normalized by k, even when the number of true labels is less than k. This can affect the evaluation of performance in our project, because we evaluate models with Precision at 500, but the majority of users have less than 500 ratings ≥ 3 in the validation set, which means the cardinality of the ground truth labels set for these users would be less than 500. Always dividing by k (500 in this case) could underestimate the model performance.

4.2.2 Model Evaluation Details

In practice, it is less likely for a user to buy a book more than once. Therefore, the main goal of our recommender system is to predict books that users would prefer and haven't read before. For each user, we define the ground truth labels set to be all books in the user's validation/test interactions that have ratings ≥ 3, assuming that ratings given by a typical user are not biased towards 0 or 5.

Furthermore, we want to evaluate the ALS model based on the predicted top 500 items for each user. For each user, we start with the top 700 recommended books with descending score(confidence) generated by the ALS model, remove books that have already appeared in the training interactions of that user, and generate predictions with the top 500 of the remaining books. This helps reduce the bias of ranking scores as books that a user highly rated in the training set won't appear in the ground truth labels set defined above. We then compute the two ranking metrics based on the $\sim$500 predictions for each user. Note that some active users have more training interactions than others. We start with the top 700 predicted books to keep the number of recommendations around 500 without spending too much time filtering out training interactions. It turns out that the average number of predictions for each user is 496, which means the precision at 500 is still informative.

4.3 Hyper-parameter Tuning and Test Results

We perform grid search on 2 hyper-parameters, the size of latent factors (rank of matrix U and V) and the regularization penalty. We train the ALS model on the training set, and select the best hyperparameter settings based on MAP score on the validation set. We also computed precision at 500 score on the validation set for the comparison with LightFM models later. Full hyper-parameter tuning results are attached in the Appendix C. In general, as rank increases, the scores increase from level of 10^{-5} to 10^{-3}. For different regularization parameters with the same rank, the best ones are around 0.15.

The test performances of models with the best parameter setting for each subsample is shown in Table 1. The performance of baseline model that always recommends the top 500 books based on average ratings (after filtering on count of ratings ≥ 20) given by training users without any customization is also reported in the table.

Subsample Size	**ALS Parameters**	**ALS MAP**	**ALS Precision@500**	**Baseline MAP**	**Baseline Precision@500**
1%	Rank: 30; Reg: 0.15	3.230×10^{-3}	0.00904	0.00576	0.0165
5%	Rank: 30; Reg: 0.1	4.036×10^{-5}	0.00248	0.000799	0.00739
10%	Rank: 30; Reg: 0.2	4.552×10^{-4}	0.00293	0.000635	0.00600

Table 1: ALS and baseline model performance on test set

5 Extension

5.1 Comparison to LightFM Model

In this extension, we train LightFM[3] recommender models on a single machine and compare both the training efficiency and prediction accuracy with the ALS model from Section 4.

LightFM[3] is a hybrid recommender model that can incorporate item/user meta-data into the matrix factorization algorithms to deal with cold-start scenarios. And when user/item features are not provided, as in our case, LightFM reduces to a traditional matrix factorization method. However, LightFM still differs from ALS in the following aspects. Firstly, it transforms ratings into implicit feedback. The developer believes that the implicit feedback works better in practice, as the explicit feedback treats the absence of ratings as absence of information, but ignores the important fact that a book is not rated because it is not chosen in the first place. Secondly, LightFM uses Weighted Approximate-Rank Pairwise (WARP) loss[4] that is useful for implicit feedback especially when optimising the top of the recommendation list (precision@k) is desired.

Two LightFM models are trained. LFM_1 uses (rating + 1) as weights for implicit feedback, where books with higher ratings are penalized more if misclassified. And LFM_2 assigns equal weights for every interaction. Both LFM models are trained on a single machine with a single thread. Grid search is performed on three parameters, where `no_components` (equivalent to rank in ALS) takes value in [10, 20, 30] and `user_alpha`/`item_alpha` (similar to regParam in ALS) takes value in $[0, 10^{-5}, 5 \times 10^{-5}]$. The data and ground truth label definition used in this section is the same as the previous ALS section, and parameters are all chosen to be identical to the ALS settings for the purpose of comparison. While LFM_1 has a better test precision@500 than LFM_2 in some cases, they have identical test precision@500 in most cases and their training times are similar. The training times and test precision@500 results of the best ALS and LFM_1 models after grid search on different data sizes is shown in Table 2. Note that the training time here only refers to the model fitting time. We do not report evaluation time here because our evaluation process is more complicated, and our implementation efficiency is not comparable with that of LightFM.

	Precision@500		**Training Time (s)**		**Best Parameters**	
	ALS	LFM_1	ALS	LFM_1	ALS	LFM_1
1%	0.00904	0.04447	104.61	75.59	Rank: 30 Reg: 0.15	Rank: 30 Reg: 5×10^{-5}
5%	0.00248	0.04638	172.63	534.69	Rank: 30 Reg: 0.1	Rank: 30 Reg: 10^{-5}
10%	0.00293	0.04644	384.65	1126.43	Rank: 30 Reg: 0.2	Rank: 30 Reg: 10^{-5}

Table 2: Performance comparison between ALS and LightFM

We can see that the prediction accuracy of the LightFM model improves as the training data increases. And in all scenarios, LightFM models have a much better precision@500 on the test data, though they generally do take longer time to train on large dataset with a single machine and a single thread. This shows that in practice, optimizing the WARP loss with implicit feedback can potentially perform much better than optimizing RMSE with explicit feedback in terms of prediction accuracy. Training efficiency of models like LightFM can be greatly improved with more threads.

5.2 Exploration

In this extension, we use UMAP to visualize ALS's learned representation of books using additional meta-data of book genres. The Goodreads Dataset has a very fuzzy version of book genres dataset in book's meta-data, as genre tags are extracted from users' popular shelves by a simple keyword matching process. In this fuzzy genre dataset, each book is matched with multiple genres with counts of the matches.

We first match each book to the genre that has the highest counts in the fuzzy genre dataset, and extract their representation from the best ALS models trained on 1%, 5% and 10% subsample sizes (all have rank 30). Note that some books have vector representation of all 0's from the ALS model as these books receive 0 ratings in all training interaction records, so we filter these books out for better visualization. We then apply the UMAP algorithm to reduce the 30-dimensional book representations into 2 dimensions, and color these data points based on their genre information. The resulting plots are really similar regardless of training data size, so only plots for latent representations learned from ALS trained with 10% subsample is shown in Figure 1.

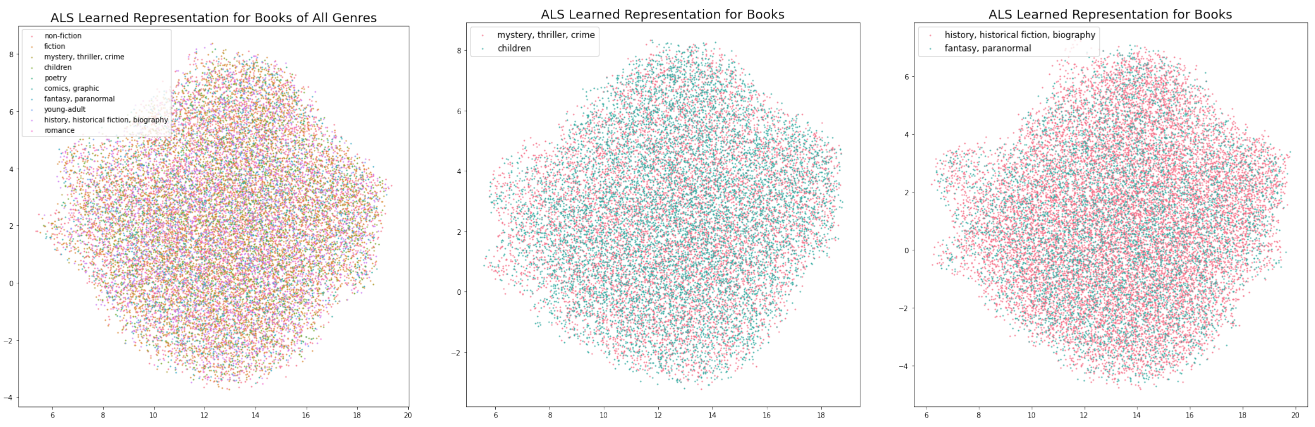

Figure 1: ALS learned representation for books trained on 10% subsample. *Left:* All genres. *Middle:* Genre `Mystery, Thriller, Crime` and `Children`. *Right:* Genre `History, Historical Fiction, Biography` and `Fantasy, Paranormal`.

We can see that books of different genres don't have explicit patterns in the 2D plot. Even for genre pairs like `Mystery, Thriller, Crime` and `Children` that are really different are not separated in 2 dimensions. Reasons may be that genre information is not significant in the latent representation of books, genre information is not explicit in 2-dimension, or our ALS model is not good enough to learn an accurate representation of the genre information.

6 Discussion

There are several limitations of our ALS model. Firstly, the memory allocated for our programming jobs is constrained by NYU Dumbo Hadoop cluster. We are faced with the trade-off between the amount of training data and the accuracy of hyper-parameter grid search. Since the utility matrix is very sparse, the model is possible to perform better when trained with more data. And based on our experiments, the size of latent factors cannot be greater than 30 when fitting 10% data on Dumbo cluster. But the score increases in general when we increase the rank parameter, as we represent the utility matrix in a higher dimensional subspace. Therefore, the model performance may also be improved with a more extensive hyper-parameter grid search. However, with the constraint on Dumbo, we suspect that we are not able to achieve these two aspects simultaneously. Secondly, the MAP and precision@500 scores of ALS are slightly lower than those of our baseline non-customized models, and we can also infer from Section 5.2 that our ALS model fails to capture hidden item features like genre information accurately. ALS can potentially learn better representations of users and items by training on data after a stricter filtering than our current one in Section 3. Users and items without enough interactions can be treated separately using cold-start strategies in practice. Thirdly, considering the performance of ALS and LightFM, we may also be able to improve ALS performance by building up an implicit recommender system and incorporating more attributes into weights.

7 Contributions

- Man Jin: data preprocessing, ALS implementation, LightFM extension, exploration extension genre matching and visualization, report writing
- Mufeng Gao: data preprocessing, baseline model implementation, ALS implementation and evaluation, exploration extension visualization, report writing
- Mu Li: data preprocessing, exploration extension visualization

References

[1] M. Wan and J. J. McAuley, "Item recommendation on monotonic behavior chains," in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, Eds., ACM, 2018, pp. 86–94. DOI: `10.1145/3240323.3240369`. [Online]. Available: `https://doi.org/10.1145/3240323.3240369`.

[2] M. Wan, R. Misra, N. Nakashole, and J. J. McAuley, "Fine-grained spoiler detection from large-scale review corpora," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds., Association for Computational Linguistics, 2019, pp. 2605–2610. DOI: `10.18653/v1/p19-1248`. [Online]. Available: `https://doi.org/10.18653/v1/p19-1248`.

[3] M. Kula, "Metadata embeddings for user and item cold-start recommendations," *CoRR*, vol. abs/1507.08439, 2015. arXiv: `1507.08439`. [Online]. Available: `http://arxiv.org/abs/1507.08439`.

[4] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2011.

A Interactions Data Description

Attribute	Brief Explain	Type	values
user_id	User ID	integer	-
book_id	Book ID	integer	-
is_read	If a user has read a certain book	integer	{0(No), 1(Yes)}
rating	Rating of a user to a certain book	integer	{0(Rating not provided), 1, 2, 3, 4, 5}
is_reviewed	If a user has posted any review for a certain book	integer	{0(No), 1(Yes)}

Table 3: Goodreads User-Book Interactions Data Description

B Statistics of Subsamples

Dataset	Records	Users	Items
1% Training	1183603	6212	59222
1% Validation	144384	1228	40927
1% Test	144579	1233	40403
5% Training	7756774	32185	259999
5% Validation	927544	6333	185505
5% Test	955689	6350	187801
10% Training	16690309	65237	445588
10% Validation	2009637	12836	326356
10% Test	2045642	12870	331729

Table 4: Statistics of subsamples after data preprocessing

C Full ALS Hyperparameter Tuning Performance

Subsample Size	**Rank**	**Reg Param**	**MAP**	**Precision@500**
1%	10	0.001	4.800×10^{-5}	4.765×10^{-4}
		0.01	1.179×10^{-4}	6.447×10^{-4}
		0.1	1.880×10^{-4}	1.682×10^{-3}
		1	1.458×10^{-4}	6.051×10^{-4}
1%	20	0.001	1.443×10^{-4}	1.215×10^{-3}
		0.01	2.723×10^{-4}	1.675×10^{-3}
		0.1	9.246×10^{-4}	4.643×10^{-3}
		1	1.392×10^{-4}	5.804×10^{-4}
$\underline{1\%}$	$\underline{30}$	0.05	1.336×10^{-3}	6.620×10^{-3}
		0.1	2.327×10^{-3}	8.440×10^{-3}
		$\underline{0.15}$	$\underline{3.230\times10^{-3}}$	$\underline{9.150\times10^{-3}}$
		1	1.409×10^{-4}	5.952×10^{-4}
5%	10	0.001	5.513×10^{-6}	6.732×10^{-5}
		0.01	5.762×10^{-6}	9.072×10^{-5}
		0.1	2.882×10^{-5}	2.023×10^{-4}
		1	7.756×10^{-5}	1.199×10^{-4}
5%	20	0.05	1.187×10^{-5}	1.163×10^{-4}
		0.15	6.802×10^{-5}	3.683×10^{-4}
$\underline{5\%}$	$\underline{30}$	0.001	1.730×10^{-5}	3.093×10^{-4}
		0.01	3.485×10^{-5}	4.914×10^{-4}
		$\underline{0.1}$	$\underline{5.149\times10^{-4}}$	$\underline{2.347\times10^{-3}}$
		1	4.036×10^{-5}	1.192×10^{-4}
10%	10	0.01	3.962×10^{-6}	4.283×10^{-5}
		0.1	1.183×10^{-5}	1.016×10^{-4}
10%	20	0.01	8.246×10^{-6}	1.112×10^{-4}
		0.1	6.570×10^{-5}	5.368×10^{-4}
		1	9.605×10^{-6}	5.894×10^{-5}
$\underline{10\%}$	$\underline{30}$	0.1	4.252×10^{-4}	1.557×10^{-3}
		$\underline{0.2}$	$\underline{4.503\times10^{-4}}$	$\underline{2.050\times10^{-3}}$
		0.25	3.008×10^{-4}	1.404×10^{-3}
		1	1.856×10^{-5}	6.704×10^{-5}

Table 5: Full ALS hyperparameter tuning performance on validation set