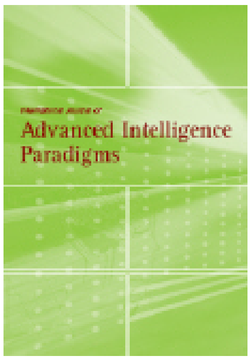


International Journal of Advanced Intelligence Paradigms

ISSN online: 1755-0394 - ISSN print: 1755-0386
https://www.inderscience.com/ijaip

An exploratory data analysis on rating data using recommender system algorithms

N Lakshmipathi Anantha, Bhanu Prakash Battula

DOI: 10.1504/IJAIP.2024.10062356

Article History:

Received:	21 April 2018
Last revised:	22 May 2018
Accepted:	28 May 2018
Published online:	22 February 2024

An exploratory data analysis on rating data using recommender system algorithms

N Lakshmipathi Anantha*

Acharya Nagarjuna University,
Nagarjuna Nagar, Nambur, Guntur,
Andhra Pradesh 522510, India
and
Department of Information Technology,
VFSTR (Deemed to be University),
Vadlamudi, Andhra Pradesh, India
Email: anlakshmipathi@gmail.com
*Corresponding author

Bhanu Prakash Battula

Department of Computer Science and Engineering,
Thirumala Engineering College,
Jonnalagadda, Narasaraopet, Andhra Pradesh, India
Email: prakash.battula33@gmail.com

Abstract: Day to day, the uploading of data into the world wide web and e-commerce directed the development of recommender systems. Recommender system filters the information based on the user's interest. Nowadays, recommender systems are being used in every domain. The advantage of a recommender system is that it makes searching easy. Recommender systems are classified into content-based filtering, collaborative filtering and hybrid approach. In this paper, we analysed the performance of item similarity, matrix factorisation and popular recommender algorithms and evaluated with precision-recall and root mean square error metrics.

Keywords: recommender systems; collaborative filtering; matrix factorisation; evaluation metrics.

Reference to this paper should be made as follows: Anantha, N.L. and Battula, B.P. (2024) 'An exploratory data analysis on rating data using recommender system algorithms', *Int. J. Advanced Intelligence Paradigms*, Vol. 27, No. 1, pp.43–60.

Biographical notes: N Lakshmipathi Anantha is a research scholar in Acharya Nagarjuna University. Currently, he is working as an Assistant Professor in the Department of IT, VFSTR University, Vadlamudi, India. He has good number of publications in reputed journals.

Bhanu Prakash Battula is working as a Professor and Head of the Department of CSE in Thirumala Engineering College, Jonnalagadda (V), India. He has good number of publications in various reputed journals.

1 Introduction

Recommender systems are playing a vital role now a day. The reason is overwhelming data is uploaded into web. Searching for something from web is very big task. Human tendency is easy way of searching. The recommender system is the only solution. Recommender System makes search easy by getting similar products or items based on the user search key word. For example, I am searching a word ‘action movies’ in the YouTube, the recommender system displays all the action movies. If we don’t have YouTube with recommender systems it will display all the movies include action, comedy and etc movies. Recommender systems filter the data based on the context of the user. If we search a key word in the search engine it will not worry about the context it simply gets all the pages which contains the keyword which we have entered. Recommender systems are being used in e-commerce, social media, movie review, news, music, video, tourism areas. The recommender system is classified into collaborative filtering approach, content-based filtering approach and hybrid approaches.

2 Related work

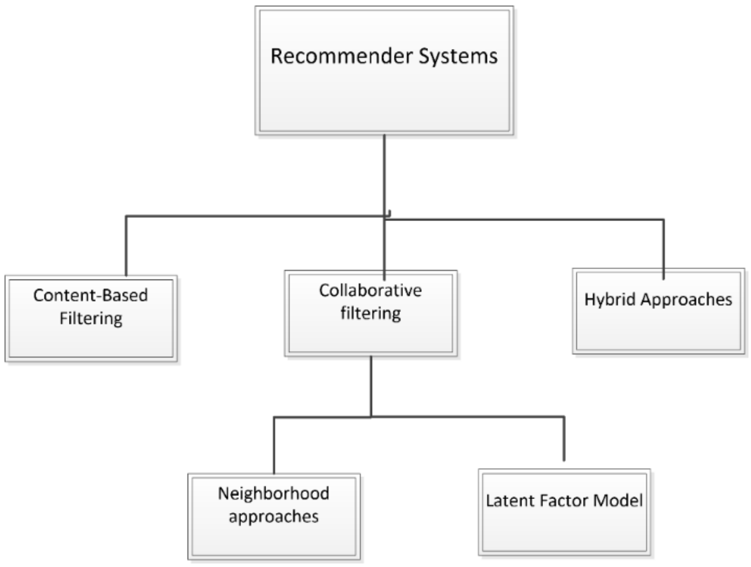
Recommender systems are broadly classified into three categories. They are content-based filtering, collaborative filtering and hybrid approaches. Content-based filtering (Van Meteren and Van Someren, 2000; Vanetti et al., 2010) recommends similar items based on the similarity. In content-based filtering user gives the recommendation based on the features of the item which the user liked the item or purchased that product. For example, a user search for Pepsi, then the content-based recommender system recommends Coca-Cola. The advantages of content-based filtering are it can recommend products for the new users with good accuracy. Content-based filtering methods rely on products metadata. That is, they require rich description of products. The only dis-advantage with content-based filtering is repetition of products. New Dude and LIBRA are implemented using content-based filtering systems.

Collaborative filtering is first coined by Tapestry developers. Collaborative filtering methods mainly run on constructing the user-product/item matrix. This matrix contains the feedback or rating given by the user on specific product/item. Then it identifies the similar users based on the feedback or ratings given by the user on the products/items. Collaborative filtering means if user X likes action movies, comedy and romantic movies and another user Y likes action movies and comedy movies they collaborative filtering recommends romantic movie to the user Y. Collaborative filtering It is broadly classified into two categories. One is neighbourhood-based models and latent factor models (Koren et al., 2009). Neighbourhood-based models concentrate on finding the relationship between user to user and product to product. The advantage of collaborative filtering is it can recommend products/items without knowing the features of the products. Disadvantages of collaborative filtering are cold start problem, data sparsity problem, etc.

Matrix factorisation methods are realisations of Latent factor models. Recommender systems takes different types of input data (Koren et al., 2009). They are explicit and implicit data. Explicit data contains user, product/item information and the rating given by the user on the product. In case of e-commerce this explicit data could be a sparse matrix. The reason is e-commerce website contains millions of products. But a user can purchase and can give rating or gives feedback to some of the products only. Matrix factorisation can handle implicit data also. Implicit data contains only user and product/item information and in addition to this information it indirectly takes mouse movements, browsing history, etc.

Examples of collaborative filtering systems are Ringo (Shardanand and Maes, 1995) is a user-based collaborative filtering makes recommendations of music albums and artists. Amazon e-commerce is also an example for collaborative filtering system. Grouplens (Konstan et al., 1997) is collaborative filtering system which recommends Usenet news. hybrid recommender algorithms (Adomavicius and Zhang, 2012) are combination of different recommendation algorithms. They are weighted hybridisation, switching hybridisation, cascade hybridisation, mixed hybridisation, feature-augmentation and meta-level.

Figure 1 Recommender systems



3 Description of datasets

Now a day so many data sets are available for analysis. In this paper five rating datasets are taken to analyse the performance of recommender algorithms. The data sets are Amazon, Book crossings, Escorts, Jester and Movielens rating data sets. All these rating data sets have three attributes only. The first attribute is user_id, second attribute is movie_id/item_id/book_id, etc., the third attribute is ratings of movie_id/item_id/book_id given by the respective user. Amazon rating dataset contains 1,048,574 records. This data contains 24,093 users, 452,728 items and the rating distribution is between 1 to 5. The data set is sparse. Book crossings rating dataset has 433,652 records. This dataset contains 77,802 users, 185,955 are books and the rating distribution is between 1 to 10. The data set is sparse data. Escorts rating dataset has 50,632 records. This dataset has 10,106 users, 6,624 items and the rating distribution is between –1 to 1. This is dense data set. Jester rating dataset has 1,048,574 records. This dataset has 14,534 users, 100 are items and the rating distribution is between –10 to 10. This is dense data set. Movielens rating dataset is most popular dataset for the researchers who do research on Recommendation Systems. Every researcher uses this Movielens rating dataset for analysis. This dataset is having 1,048,574 records. This dataset contains 7,636 users, 9,693 items and the rating distribution is between 1 to 5. This is dense data set. We can get this datasets from different websites. We downloaded this dataset from konect (konect.uni-koblenz.de/) website.

4 Recommender algorithms

In this paper, we are doing experimental analysis on different datasets with item similarity recommender, factorisation recommender, ranking factorisation recommender and popular recommender with the help Turi recommender tool (https://turi.com/).

Item similarity recommender identifies similar user and as well as similar items based on some similarity metrics. Those metrics are cosine similarity, Pearson correlation coefficient and Jaccard. These are most popular similarity metrics.

Cosine similarity between two products is calculated as.

$$CS(i, j) = \frac{\sum_{u \in U_{ij}} r_{U_i} r_{U_j}}{\sqrt{u \in U_i^{\mu_{U_i}^2}} \sqrt{\sum_{u \in U_j} \mu_{U_j}^2}} \tag{1}$$

where *Ui* is product *i*, is rated by the user group, and *Uij* is the set of users rated products *i* and *j* are rated by user group.

The main problem with Cosine similarity measure is that it does not consider the differences in the mean and variance of the ratings made to items *i* and *j*.

Another widespread measure to compute similarity is Pearson correlation similarity:

$$PS(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - r_i)(r_{ui} - r_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - r_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{ui} - r_j)^2}} \tag{2}$$

where U_i is product *i*, is rated by the user group, and U_{ij} is the set of users rated products *i* and *j* are rated by user group.

The main problem with Cosine similarity measure is that it does not consider the differences in the mean and variance of the ratings made to items *i* and *j*. Pearson similarity measure gets affected with the mean and variance of the rating.

Cosine and Pearson similarity measures are well suited for explicit datasets. As these datasets depends on feedback of the user or rating of the user.

Jaccard similarity is used to measure the similarity between two set of elements. The Jaccard similarity between two items is computed as

$$JS(i, j) = \frac{|U_i \ ? \ U_j|}{U_i \ \text{J} \ U_j} \tag{3}$$

where U_i is the set of users rated an item *I* similarly U_j is the set of users rated an item j.

Jaccard is a good choice when the dataset is implicit data. Implicit datasets contain only users and products data only. It does not contain any information regarding ratings given by user on the product or feedback given by the user on the product.

Under latent factor models, matrix factorisation is a successful model and accurate model when compared to the remaining collaborative filtering algorithms. This is proved from the 1million challenge on Movielens dataset. Matrix factorisation techniques learn latent factors using item and user only but factorisation machine (Rendle, 2012) learns latent factors using user -item, side features, biases and pairwise combinations. So, with factorisation machine one can represent complex relationships. Both factorisation recommender and ranking factorisation recommender are trained using different solvers. They are stochastic gradient descent (Bottou, 2010), alternating least squares (ALS), adaptive gradient descendent (AdaGrad) and implicit ALS (iALS). If we don't provide any solver to factorisation recommender and ranking factorisation recommender, both consider stochastic gradient descendent (SGD) as default solver.

The difference between factorisation recommender and ranking factorisation recommender is, factorisation recommender considers user-item and their respective rating for prediction while in ranking factorisation recommender it considers only user-item for prediction. By default, ranking factorisation recommender considers only explicit data means it considers only user-item for prediction. We have an option for ranking factorisation recommender to work on implicit data also means it can consider rating data also along with user-item for better prediction.

Popular recommender creates a model that makes recommendations using item popularity. When no target column is provided, the popularity is determined by the number of observations involving each item. When a target is provided, popularity is computed using the item's mean target value. When the target column contains ratings, for example, the model computes the mean rating for each item and uses this to rank items for recommendations. The popularity model ranks an item according to its overall popularity. The popularity recommender is simple and fast and provides a reasonable baseline. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

Item similarity recommender gets similar items and similar users. Popular recommender algorithm gives the popular items/movies/books and popular users. Item similarity recommender model scores items based on how likely they predict the user will rate them highly, but the absolute values of the predicted scores may not match up with the actual ratings a user would give the item.

The ranking factorisation recommender predicts items that are both similar to the items in a user's dataset and, if rating information is provided, it recommends the items those rated highly by the user. It tends to predict ratings with less accuracy than the non-ranking factorisation recommender, but it tends to do much better at choosing items that a user would rate highly. This is because it also penalises the predicted rating of items that are significantly different from the items a user has interacted with. In other words, it only predicts a high rating for user-item pairs in which it predicts a high rating and is confident in that prediction. Furthermore, this model works particularly well when the target ratings are binary.

The factorisation recommender and ranking factorisation algorithms uses SGD, AdaGrad, ALS and implicit ALS (iALS) optimisers.

SGD is one of the batch optimisation methods. SGD is a method for minimising (or maximising) cost function (or loss function). In general SGD is applied to the optimisation of a problem using the cost function as a measure of that optimisation. SGD (Anastasiu et al., 2016) iterates on selecting the ratings randomly and updates the gradients. SGD has two advantages over other gradient descendent algorithms. The first advantage is, it computes and updates the parameter value by considering the part of the training data set which in turn reduces the variance in the parameter value. The second advantage is SGD lets new data can be inserted in online environment.

AdaGrad (Anastasiu et al., 2016) is another optimisation method and it is very useful to handle sparse data sets. Now a day major e-commerce sites have sparse data sets only. The reason is e-commerce sites are having millions of products/items as well as millions of users but the products purchased by each user is very small/less when compared to the range of products available in the e-commerce sites. To handle these situations AdaGrad is very much useful. AdaGrad performs well on sparse rating datasets. AdaGrad performs larger updates for infrequent and smaller updates for frequent parameters.

ALS (Anastasiu et al., 2016) is a method that alternates between two matrices such as $D = UP$ where D is data. Essentially it guesses U to estimate P and then alternates back and forth until U and P stop changing. By fixing one or the other it becomes a simple least squares solution. The advantages of ALS are it well suited for Parallelisation and it performs well when we have large dense data set. We apply ALS on implicit data is iALS (Hu et al., 2008).

5 Evaluation metrics

Recommender systems have ample number of metrics to check the quality of recommender algorithms. Mean absolute error (Goldberg et al., 2001), root mean absolute error (Cotter and Smyth, 2000), precision-recall curve (Drosou and Pitoura, 2010) metrics are statistical accuracy metrics.

MAE is the measure of deviation of user's actual value and predicted value. It is formulated as follows:

$$MAE = \frac{1}{N}\sum_{u,i}\left|P_{ui} - r_u\right| \tag{4}$$

where P_{ui} is the predicted rating on item i by the user u, i is original rating and N is the total ratings on the item set. The MAE is minimum means, the prediction of ratings of the recommender engine accurate. Also, the root mean square error (RMSE) is given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}\left(P_{u,i} - r_{u,i}\right)2} \tag{5}$$

The minimum RMSE means, prediction of ratings by the recommender engine is accurate.

Precision is the fraction of good products recommended to total recommended products and recall defined as the fraction of good products recommended those are part of the set of all useful products recommended. They are computed as

$$\text{Precision} : \frac{good\ \ products\ recommended}{total\ recommended\ \ products} \tag{6}$$

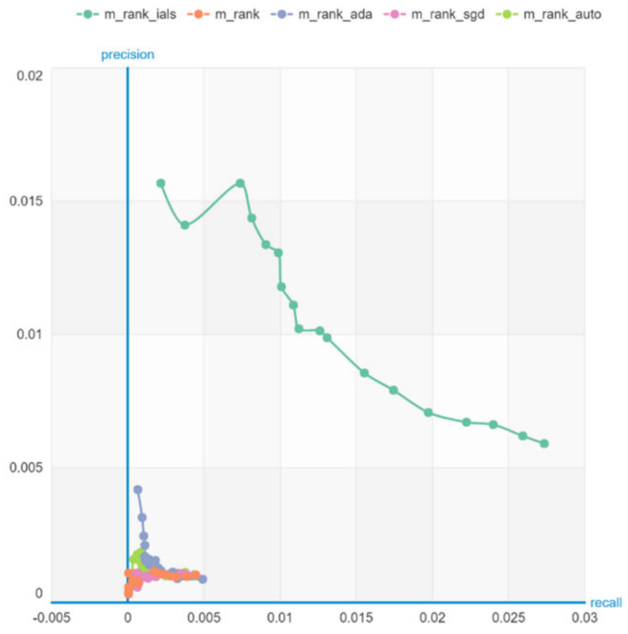
$$\text{Recall} : \frac{good\ \ products\ recommended}{all\ useful\ \ products\ recommended} \tag{7}$$

6 Experimental analysis

In this paper, we are analysing item similarity recommender, popular recommender and matrix factorisation approaches on five datasets. The performance of the algorithms is explained with precision – recall metric.

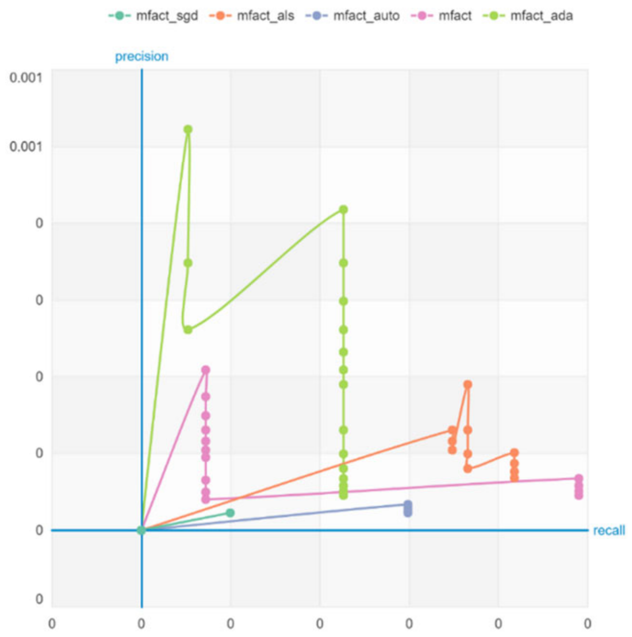
On Amazon rating dataset, we conducted experiments using ranking factorisation, factorisation, item similarity recommender and popular recommender algorithms. In ranking factorisation algorithms, we compared the results with the AdaGrad, iALS, ALS, SGD optimisers. iALS is performing well when compared to the remaining optimisers. The precision-recall curve tells that, the dataset growing large the value of precision and recall value getting decreased and become stable. On explicit datasets iALS performs well. After iALS, AdaGrad is doing well. The reason is AdaGrad learning rate is good on sparse datasets when compared to dense datasets. As the Amazon rating dataset is sparse this algorithm is doing well. Ranking factorisation using implicit ALS is too random. Figure 2(a) depicts the comparison of all the algorithms performance.

Figure 2 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Item similarity recommender performance (see online version for colours)

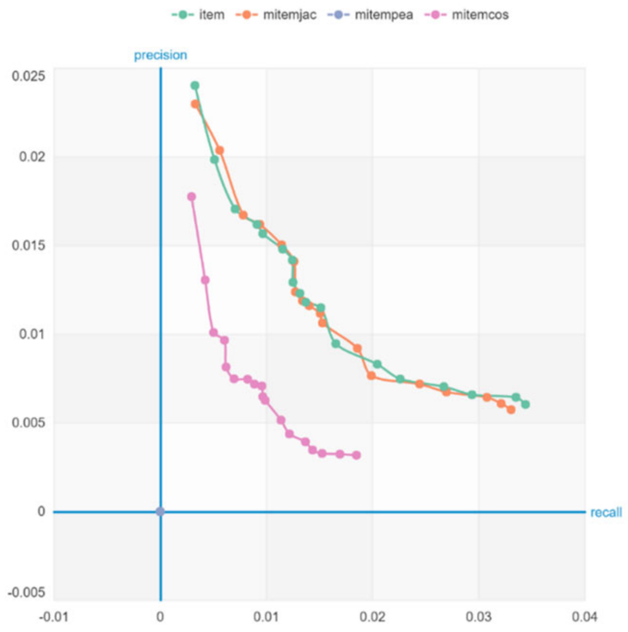


(a)

Figure 2 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Item similarity recommender performance (continued) (see online version for colours)



(b)



(c)

AdaGrad, ALS, SGD optimisers are used in Factorisation approach. From these optimisers, AdaGrad optimiser is performing well when compared to the remaining optimisers. After AdaGrad optimiser ALS is working better when compared to the remaining optimisers.

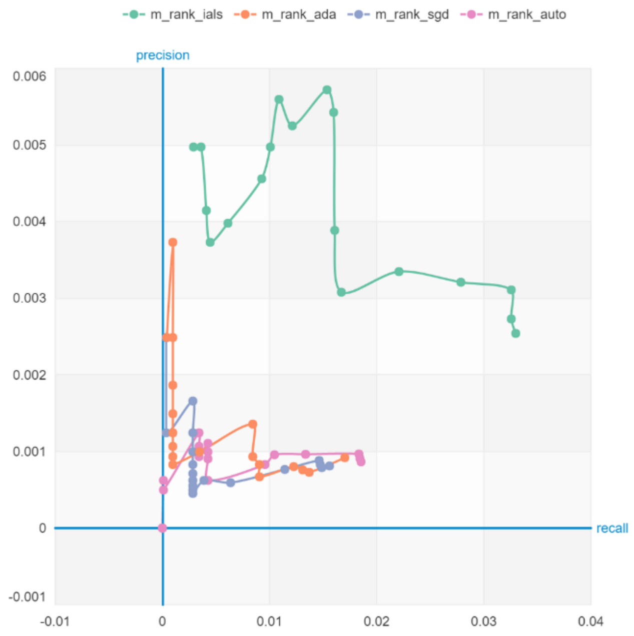
The reason is the learning rate of AdaGrad is good on sparse datasets compared to dense datasets. As the Amazon rating dataset is sparse this algorithm is doing well. Figure 2(b) depicts the comparison of all the algorithms performance.

Cosine similarity, Jaccard and Pearson similarity measures are used in item similarity algorithms. In these three metrics Jaccard similarity measure is performing well. The reason is the learning rate is good on sparse datasets and it is not good on dense datasets. Figure 2(c) gives comparison of all the metrics used in item similarity algorithms.

In popular recommender algorithms, we have taken base line algorithms. The Popularity Recommender algorithm is simple and fast and provides a reasonable baseline. In the Amazon dataset, we have provided the ratings also. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

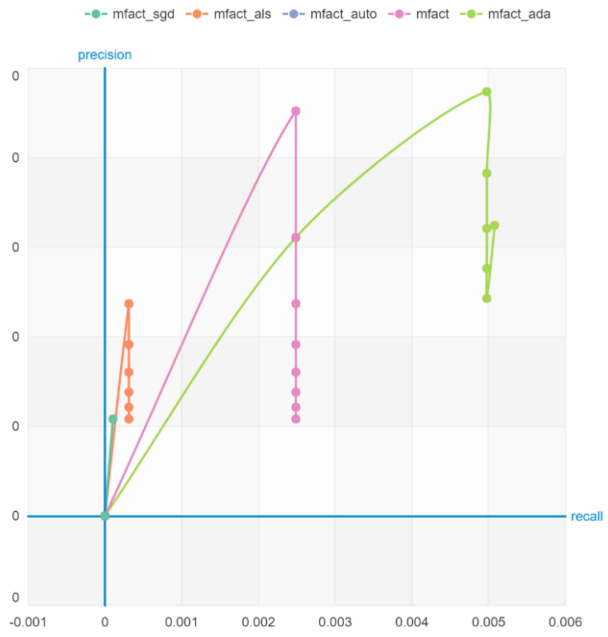
On Bookcrossing rating dataset, we conducted analysis with recommender algorithms. In ranking factorisation algorithms, we compared the results

Figure 3 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (see online version for colours)

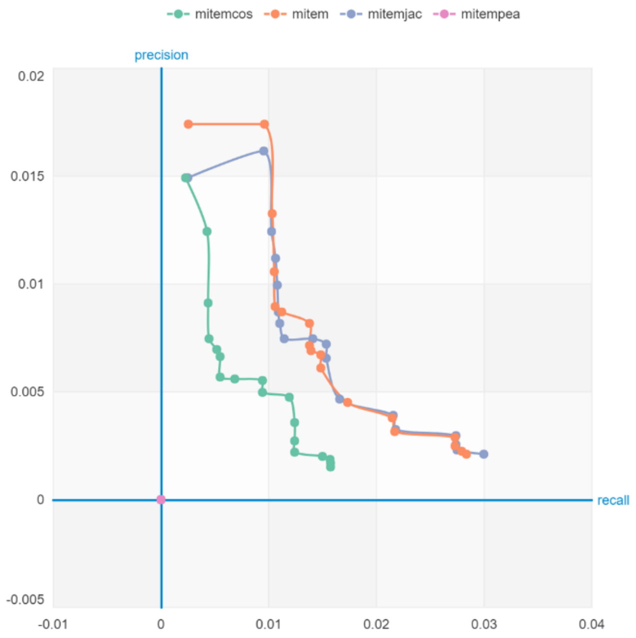


(a)

Figure 3 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (continued) (see online version for colours)



(b)



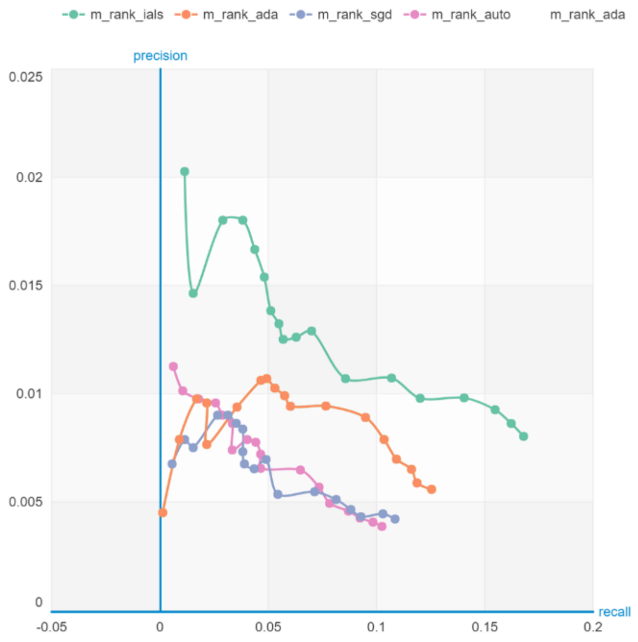
(c)

On Bookcrossing rating dataset, we conducted analysis with Recommender Algorithms. In ranking factorisation algorithms, we compared the results with AdaGrad, iALS, ALS, SGD optimisers. iALS is performing well when compared to the remaining optimisers. The reason is that the dataset we have provided is explicit. On explicit datasets iALS performs well. If we observe the diagram we come to know all the remaining performing same. After iALS, AdaGrad is going well. The reason is learning is good on sparse datasets and it is not good on dense datasets. As the Amazon rating dataset is sparse this algorithm is doing well. Ranking factorisation using Implicit ALS is too random. Figure 3(a) gives individual performance of the algorithms and comparison of all the algorithms performance given.

In factorisation algorithms, we have taken AdaGrad, ALS, SGD optimisers. In these optimisers, AdaGrad optimiser is performing well when compared to the remaining optimisers. After AdaGrad optimiser ALS is working better when compared to the remaining optimisers. In the diagram, the complete dataset is loading the performance of the algorithm also increasing for iALS and the learning rate is good on sparse datasets compared to dense datasets. As the Bookcrossing rating dataset is sparse this algorithm is doing well. Figure 3(b) gives the comparison of all the algorithms performance.

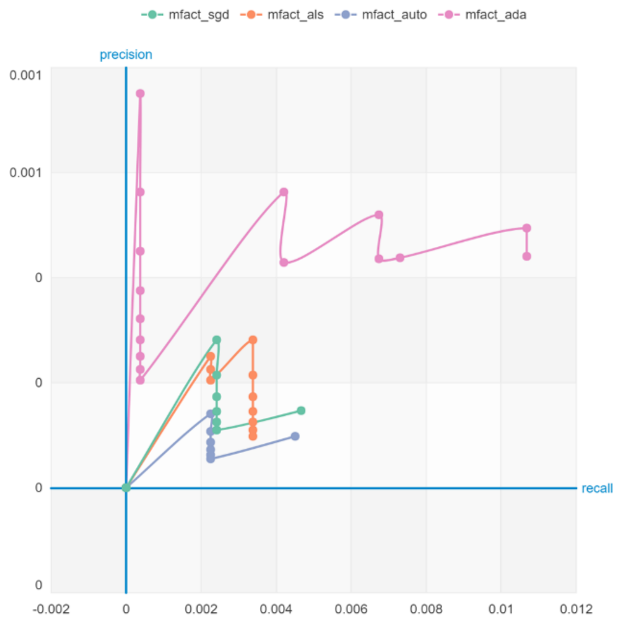
Jaccard similarity is prediction is good when compared to the remaining similarity metrics. The reason is the learning is good on sparse datasets and it is not good on dense datasets. Figure 3(c) gives comparison of all the algorithms performance.

Figure 4 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (see online version for colours)

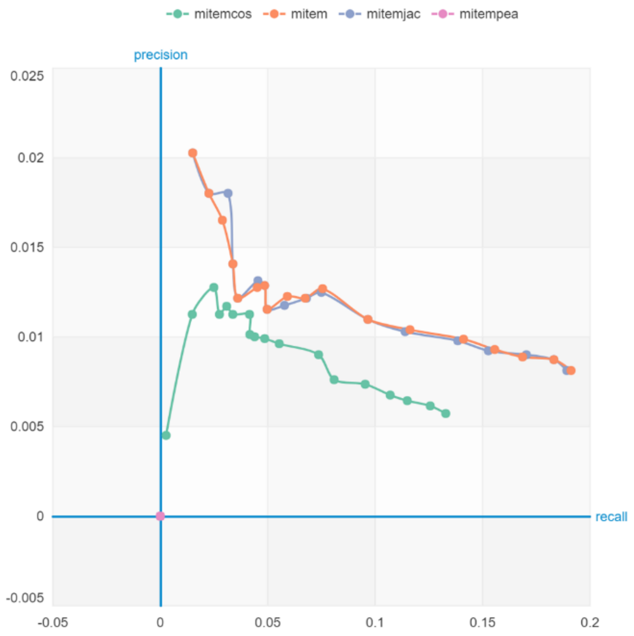


(a)

Figure 4 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (continued) (see online version for colours)



(b)



(c)

The popularity recommender is simple and fast and provides a reasonable baseline. In the Bookcrossing dataset, we have provided the ratings also. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

Now on Escorts rating data set we conducted experiments. In ranking factorisation algorithms, we have taken AdaGrad, iALS, ALS, SGD optimisers. AdaGrad is performing well when compared to the remaining optimisers. The reason is that the dataset we have provided is explicit. On explicit datasets AdaGrad performs well. If we observe the diagram we come to know all the remaining performing same. After AdaGrad, iALS is going well.

As the Escort rating dataset is sparse this algorithm is doing well. Ranking factorisation using iALS is too random. Figure 4 give individual performance of the algorithms and comparison of all the algorithms performance given.

AdaGrad, ALS, SGD optimisers are used in factorisation algorithms. In these optimisers, AdaGrad optimiser is performing well when compared to the remaining optimisers. The reason is the learning is good on sparse datasets and it is not good on dense datasets. As the Escort rating dataset is sparse this algorithm is doing well. Jaccard similarity metric is performing well. Pearson similarity metric is not performing well. The reason is the learning is good on sparse datasets and it is not good on dense datasets.

As the Escorts rating dataset is sparse this algorithm is doing well. The following diagram gives comparison of all the algorithms performance.

In popular recommender algorithms, we have taken base line algorithms. The popularity recommender is simple and fast and provides a reasonable baseline. In the Amazon dataset, we have provided the ratings also. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

On Jester rating data set we conducted analysis. In ranking factorisation algorithms, we have taken AdaGrad, iALS, ALS, SGD optimisers. AdaGrad is performing well when compared to the remaining optimisers.

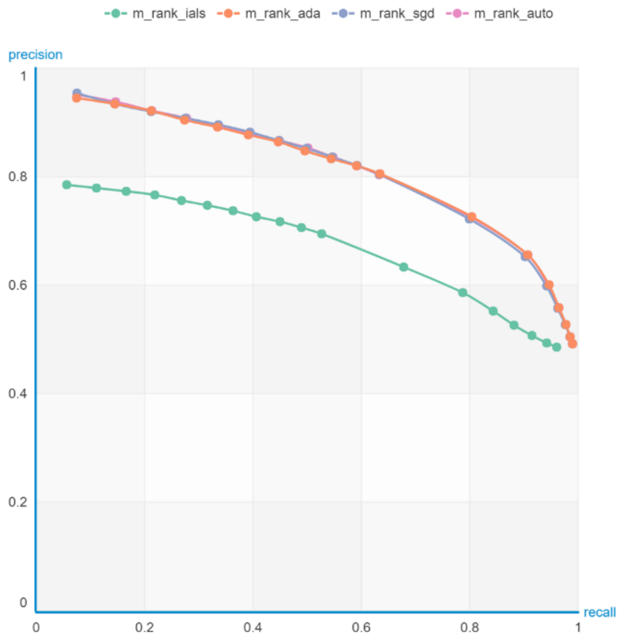
The reason is that the dataset we have provided is explicit. On explicit datasets AdaGrad performs well. If we observe the diagram we come to know all the remaining performing same. After AdaGrad, iALS is going well. Ranking factorisation using iALS is too random. Figure 5(a) gives individual performance of the algorithms and comparison of all the algorithms performance given. Now on Jester rating data set we conducted experimental analysis. In factorisation algorithms, we have taken AdaGrad, ALS, SGD optimisers.

In these optimisers, SGD is performing well when compared to the remaining optimisers. After SGD, AdaGrad is working better when compared to the remaining optimisers. Figure 5(b) gives the comparison of all the algorithms performance.

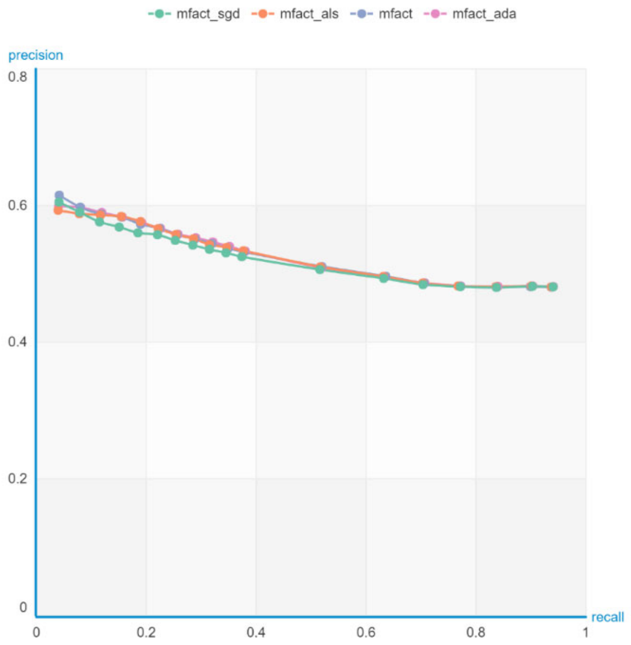
In item similarity recommender algorithms, Jaccard similarity metric is performing well. Pearson similarity metric is not performing well. The reason is the learning is good on sparse datasets and it is not good on dense datasets.

In popular recommender algorithms, we have taken base line algorithms. The popularity recommender is simple and fast and provides a reasonable baseline. In the Amazon dataset, we have provided the ratings also. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

Figure 5 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (d) Popular algorithms performance (see online version for colours)

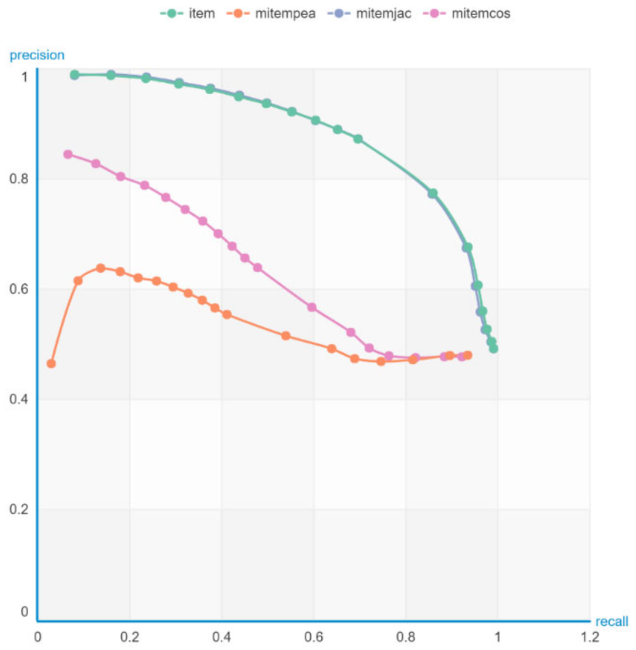


(a)

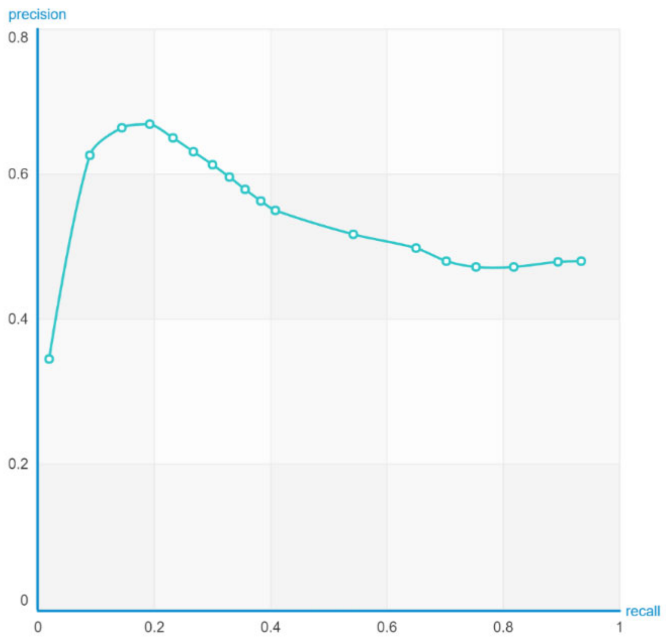


(b)

Figure 5 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (d) Popular algorithms performance (continued) (see online version for colours)

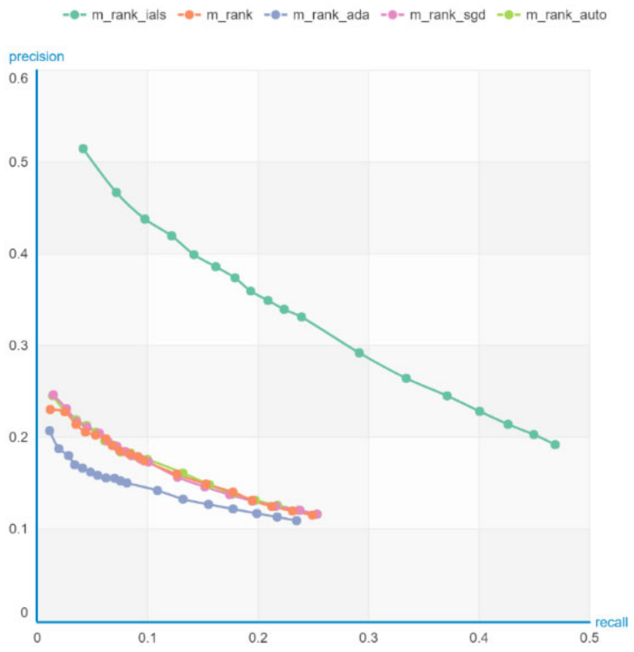


(c)

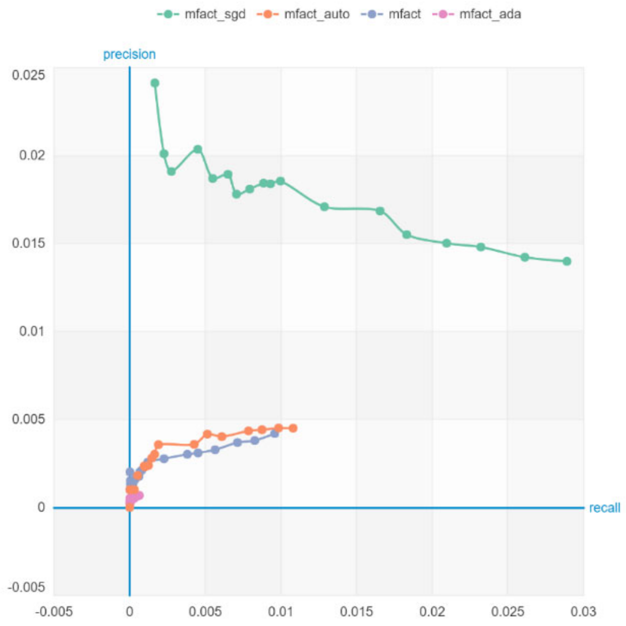


(d)

Figure 6 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (see online version for colours)

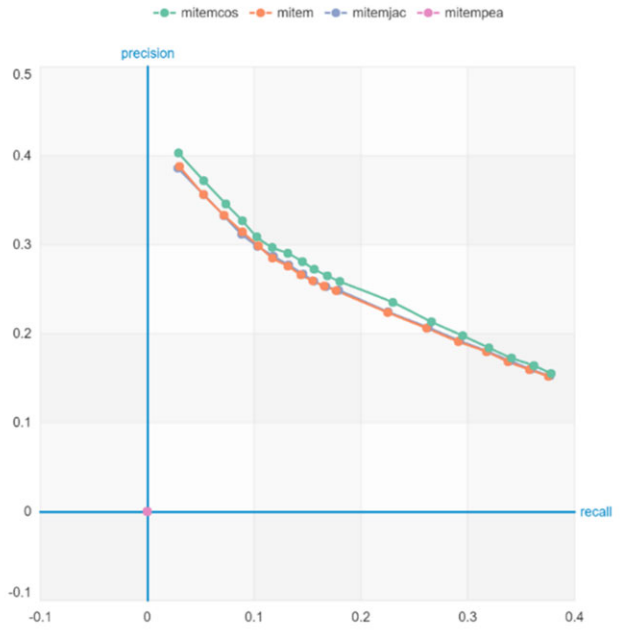


(a)



(b)

Figure 6 (a) Ranking factorisation algorithms performance (b) Factorisation algorithms performance (c) Ranking factorisation algorithms performance (continued) (see online version for colours)



(c)

In ranking factorisation algorithms, we have taken AdaGrad, iALS, ALS, SGD optimisers. iALS is performing well when compared to the remaining optimisers. The reason is that the dataset we have provided is explicit. On explicit datasets iALS performs well. If we observe the diagram we come to know all the remaining performing same. After iALS, AdaGrad is going well. The following diagram gives comparison of all the algorithms performance.

Ranking factorisation using iALS is too random. Figure 6(a) gives individual performance of the algorithms and comparison of all the algorithms performance given.

SGD, ALS and AdaGrad optimisers are used in Factorisation Algorithms. SGD is performing well on Movielens dataset. The reason is on dense datasets SGD performs well. Figure 6(b) gives the performance of different optimisers on Movielens dataset.

Cosine similarity is performing well on Movielens dataset. Figure 6(c) gives comparison of all the algorithms performance on rating dataset. In popular recommender algorithms, we have taken base line algorithms.

The popularity recommender is simple and fast and provides a reasonable baseline. In the Amazon dataset, we have provided the ratings also. It can work well when observation data is sparse. It can be used as a 'background' model for new users.

The reason is both the algorithms identifying the patterns between user-item/product. These algorithms do not consider rating/feedback given by the user. The performance of Factorisation algorithm is not good because it considers rating/feedback given by the user mainly. The reason is user gives rating/feedback for very few products/items.

7 Conclusions and future work

In this paper, we evaluated the recommender algorithms performance on Amazon, Book crossing, Escorts, Jester, Movielens datasets. Item similarity recommender is performing well on most of the datasets. After item similarity recommender, ranking factorisation recommender is performance is good. In future, we want to develop a new algorithm for top-N recommendations and innovative approach for ranking the products/items.

References

Adomavicius, G. and Zhang, J. (2012) 'Impact of data characteristics on recommender systems performance', *ACM Trans. Manage Inform. Syst.*, Vol. 3, No. 1, pp.1–6.

Anastasiu, D.C., Christakopoulou, E., Smith, S., Sharma, M. and Karypis, G. (2016) *Big Data and Recommender Systems*, DOI: 10.13140/RG.2.2.12526.00325,PP-1-26.

Bottou, L. (2010) 'Large-scale machine learning with stochastic gradient descent', in *Proceedings of COMPSTAT' 2010*, Physica-Verlag HD, pp.177–186.

Cotter, P. and Smyth, B. (2000) 'PTV: intelligent personalized TV guides', in *Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp.957–964.

Drosou, M. and Pitoura, E. (2010) 'Search result diversification', *SIGMOD Rec*, Vol. 39, No. 1, pp.41–7.

Goldberg, K., Roeder, T., Gupta, D. and Perkins, C. (2001) 'Eigen taste: a constant time collaborative filtering algorithm', *Inform Retrieval J*, Vol. 4, No. 2, pp.133–151.

Hu, Y., Koren, Y. and Volinsky, C. (2008) 'Collaborative filtering for implicit feedback datasets', in *Data Mining*, *ICDM'08*, *Eighth IEEE International Conference on*, IEEE, pp.263–272.

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedl, J. (1997) 'Applying collaborative filtering to usenet news', *Commun. ACM*, Vol. 40, No. 3, pp.77–87.

Koren, Y., Bell, R. and Volinsky, C. (2009) 'Matrix factorization techniques for recommender systems', *Computer*, Vol. 42, No. 8, pp.42–49.

Rendle, S. (2012) 'Factorization machines with LIBFM', *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 3, No. 3, p.57.

Shardanand, U. and Maes, P. (1995) 'Social information filtering: algorithms for automating 'word of mouth'', in *Proceedings of the Sigchi Conference on Human Factors in Computing Systems*, ACM Press/Addison-Wesley Publishing Co., pp.210–217.

Van Meteren, R. and Van Someren, M. (2000) 'Using content-based filtering for recommendation', in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, pp.47–56.

Vanetti, M., Binaghi, E., Carminati, B., Carullo, M. and Ferrari, E. (2010) 'Content-based filtering in on-line social networks', in *International Workshop on Privacy and Security Issues in Data Mining and Machine Learning*, pp.127–140, Springer, Berlin, Heidelberg.