Uniwersytet Ekonomiczny w Katowicach

Studia Ekonomiczne. Zeszyty Naukowe
Uniwersytetu Ekonomicznego w Katowicach
ISSN 2083-8611 Nr 342 · 2017

Informatyka i Ekonometria 11

Maryna Nehrey

National University of Life and Environmental Science of Ukraine
Department of Economic Cybernetics
marina.nehrey@nubip.edu.ua

Taras Hnot

National University of Life and Environmental Science of Ukraine
Department of Economic Cybernetics
tarashnot@gmail.com

USING RECOMMENDATION APPROACHES FOR RATINGS MATRIXES IN ONLINE MARKETING

Summary: The main objective of the study is detecting of advantages and disadvantages of different algorithms which are used when building recommender system. Recommender systems became so popular because of active development of online marketing and increase of sales through the Internet. Development and implementation of a strategy for recommending products cause effective use of resources and dynamic sales of the company. Recommender systems are one of the most effective tools: systems, which are built using memory-based algorithms, and systems with model-based algorithms. The best performance was shown by Matrix Factorization techniques with Stochastic Gradient Descend. When selecting a recommender system it is advisable to consider the purpose of use, product features, specifications and availability of customer data on their preferences. The use of one of the described recommender system will improve the efficiency of the product marketing.

Keywords: recommender systems, memory-based recommender systems, model-based recommender systems, matrix factorization.

JEL Classification: C10, C18, C22, C60, C88.

Introduction

Nowadays, it is not enough to advertise products on different Internet resources. Now we all live and work in the epoch of Big Data. It means that management of data is very cheap in comparison with knowledge which could be extracted from it in case of right processing. It is very important to use modern methods of marketing strategies. In the late 90th data scientists of one of the biggest Internet stores of books – Amazon, developed method, which gave an ability to recommend books to clients based on associations between them. This

led to incredible results – sales increased 100 times in comparison with recommendations which were given by critics. Today third of sales Amazon makes using recommender systems, there are many such examples in the world. And all of them show that target advertisement is more efficient than general one.

There are many different strategies of Internet-marketing. Simple ones are focused on advertising new or popular products. They don't take into account target audience. There are also strategies which advertise products based on mathematical calculations, so their purpose is to predict product, which most likely would be sold to client. Such models could be simple, for example, they could propose products with the highest demand. On the other hand, they could be mathematically complicated and built using machine learning and artificial intelligence frameworks. Such models are models of machine learning class of models – the basis to build recommender systems.

From the scientific view, recommender systems – subclass of information filtering systems, which build sorted and ranked list of objects, which could be interesting for user. To build such system we could use information about the user, her history in the environment (for example, history of purchases), information about the product, etc. In addition, recommender systems compare data of the same type from different people and calculate list of recommendations for specific user.

Such diversity in recommender algorithms evokes questions: "Which one is the best?", "Which one is better for mine problem?", "Which one is more accurate?". Of course, there are no general answer for all these questions. Different recommender algorithms could show different accuracy in different situations. They also are diverse in training time and tune complexity.

This article is focused on a comparison of few recommender algorithms of different types; it contains the comparison of their accuracy, training time and tune complexity.

It also contains links to developed R libraries and R Markdown HTML file with all code available. So, it is possible to reproduce results and repeat the experiment on other datasets.

1. Related works

Comparison of recommender systems has been the subject of many works. Ricci, Rokach and Shapira [2015] were focused on general idea and challenges of Recommender Systems. Gorakala and Usuelli [2015] studied the main steps of Building a Recommendation system. The study by Breese, Heckerman and

Kadie [1998] compared few different models. It was focused on a comparison of different types of similarity which could be used to build collaborative filtering system (Cosine similarity and Pearson correlation). This work compares as memory-based as model-based recommenders (clustering approach, Bayesian networks). All experiments were run on three different datasets.

Huang, Zeng and Chen [2007], in their article compared user-based CF, item-based CF, SVD Approximation and few other model-based techniques. They used classification evaluation measures (precision, recall, F1-measure and rank score) in their work, as a basis for comparison and included few remarks about computation complexity.

Lee, Sun and Lebanon [2012] included in their study more than 10 different models and variations of approaches. They evaluated them in few dimensions (accuracy dependency on a number of users, items, the sparsity of matrix, computation complexity). All work was done on Netflix dataset.

Vozalis, Markos and Margaritis [2009] compared user-based collaborative filtering and item-based with a non-personalized approach for recommendations. They did a set of experiments, playing with different tuning parameters and showed that classical collaborative filtering (user-based) shows the best results. Also, it was mentioned that non-personalized approach did unexpectedly well.

Hauger, Tso and Schmidt-Thieme [2008] were focused on new items problem. They have shown that few products' attributes could help to overcome a problem of new-items and user-bias. They did comparative analysis using three algorithms, which work only with rating matrixes, and one, which takes into account attributes information and showed that this last works better.

Even all these works give a good vision of advantages and disadvantages of different recommender algorithms, we decided to make our own study. It is more focused on tuning problem of different recommenders and compares tuned models with the best parameters.

2. Dataset

In the study, we have used 1M MovieLens dataset. MovieLens is a project of GroupLens Research [www 1], a research lab in the Department of Computer Science and Engineering at the University of Minnesota, since 1997. This project was focused on gathering research data on personalized recommendations. MovieLens is a recommender system and virtual community website that recommends movies for its users to watch, based on their film preferences using collaborative filtering. Reproducible code of 1M MovieLens dataset [www 2].

1M MovieLens dataset contains approximately 1 million ratings of 6040 movies from 3706 users. Ratings variate from 1 to 5. General statistics of this dataset could be observed in the Figure 1 (grey lines show median values). The level of rating matrix sparsity is 0.045.

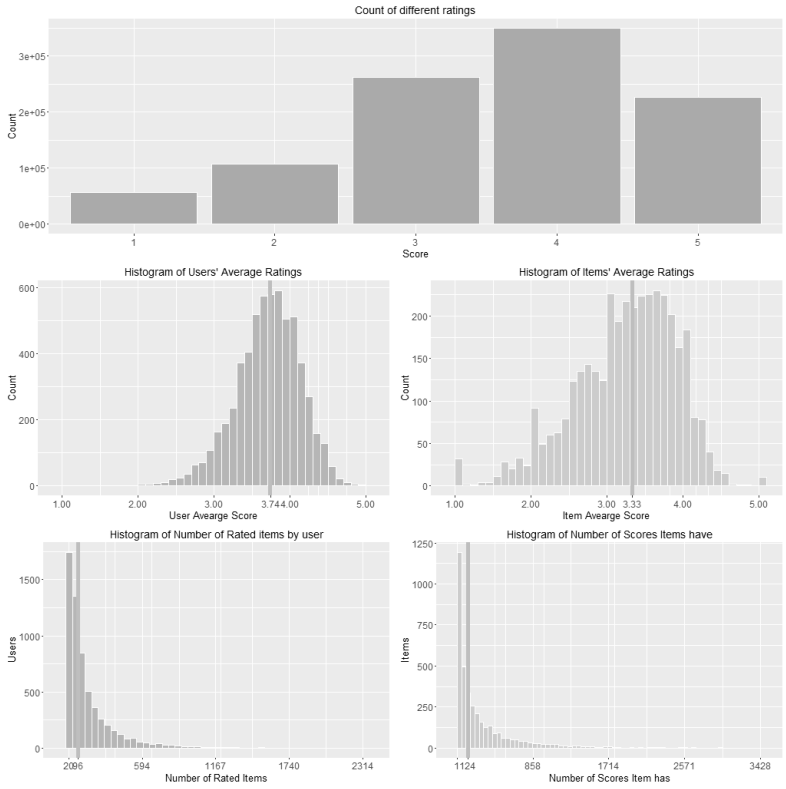

Figure 1. Overview of 1M MovieLens dataset

Source: Own research.

3. Evaluation of recommenders

There are different evaluation metrics of recommender systems: mean absolute error (MAE), root of the mean square error (RMSE), precision, recall, F1 score. Also, recommender systems could be evaluated based on training time, tuning complexity, etc. Different metrics lead to different results. In our work, we focus on RMSE:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{u,i} \left(p_{u,i} - r_{u,i} \right)^2},$$

$p_{u,i}$ – *prediction for user u and item i*,

n – *number of available ratings.*

Each algorithm was evaluated 10 times for different train/test splits in order to achieve more stable results.

4. Types of recommender models

There are two main categories of collaborative filtering algorithms: memory-based and model-based methods [Lee, Sun, Lebanon, 2012]. Memory-based methods simply memorize all ratings and make recommendations based on the relation between user-item and rest of the matrix. In model-based methods, predicting parametrized model firstly is needed to be fit based on rating matrix and then recommendations are issued based on a fitted model.

The most popular two memory-based methods are user-based and item--based collaborative filtering. These methods are an example of neighboring--based methods, which refer to ratings of similar users or items and make recommendations based on the weighed sum of nearest users/items ratings. User-based CF method is built based on assumption that if two users have similar ratings on some items, they will have similar ratings on the remaining items. The same for item-based CF, but with item perspective.

Model-based methods, on the other hand, build parametrized models and recommend items with the highest rank, returned by model. For example, Slope One method learns a set of simple predictors (one for each pair of two items) with a just constant variable. Therefore, this variable represents average difference between ratings of two items. Using this method, fast computation and reasonable accuracy could be easily achieved. Another example of this class of methods – it's SVD Approximation. In this approach, the ranking matrix is decomposed based on Singular Value Decomposition and then reconstructed keeping only first most significance entities. This gives an ability to predict missing values of the ranking matrix.

5. Recommender algorithms

In this study, we compared 6 different algorithms which are listed in Table 1.

Table 1. Recommender Algorithms used in comparative study

Category	Subcategory	Algorithm
Memory-Based	Baseline	Most Popular (Item Average)
	Similarity Based	User-Based Collaborative Filtering
		Item-Based Collaborative Filtering
Model-Based	Linear Regression	Slope One
	Matrix Factorization	Matrix Factorization with Gradient Descend
		SVD Approximation

Source: Own research.

5.1. Data preprocessing

Each collaborative filtering method works with rating matrix. Formally, we have a set of users $U = \{u_1, u_2, ..., u_m\}$ and a set of items $I = \{i_1, i_2, ..., i_n\}$. Rating matrix is represented by $R = \{r_{jk}\}, j \in \overline{1,m}, k \in \overline{1,n}.$ Each row of matrix R represents user and columns represent items.

Matrix R is likely to have user rating bias which could decrease performance of some algorithms. This bias should be removed by normalizing rating matrix before applying recommender. This step could be treated as preprocessing step in building recommender system. In every rating data, there are users, who consistently rates items with high or low scores. Normalization could be performed by extracting average rating of each user from all her known ratings:

$$R_{norm} = \{r_{jk}\}_{norm} = \{r_{jk} - \bar{r}_j\}, \bar{r}_j = \frac{\sum_{k \in A_j} r_{jk}}{|A_j|}, \ j \in \overline{1,m}.$$

$$A_j - set\ of\ indexes\ of\ available\ ratings\ for\ user\ j.$$

There are also a set of another normalization techniques. We could remove item rating bias by subtracting averages of items' available ratings or apply user and item normalization simultaneously. We could also apply Z-Score normalization which also takes variance of ratings into account.

In the following experiments, we have normalized data by removing user rating bias.

5.2. Most popular

Most popular (item average) approach computes average rating for each item based on available ratings and predicts each unknown rating as average for item [Hahsler, 2011]. As a result, missed ratings for each item will be the same for each user.

Algorithm:

1. Calculate average rating for each item:

$$\overline{r_k} = \frac{\sum_{j \in B_k} r_{jk}}{|B_k|}, \; k \in \overline{1,n.}$$

B_k – *set of indexes of available ratings for item* k .

2. Predict missed ratings in R as average for item:

$$r_{jk} = \overline{r_k}, \; r_{jk} = \; ? \; (\textit{missed ratings}) .$$

5.3. User-based collaborative filtering

User-based CF forms predictions based on aggregated ratings from the closest users (nearest neighbors). Nearest neighbors are defined based on similarity between users which is calculated using available ratings. It is important to understand, that this method works under assumption that users with similar ratings will rate items similarly.

There are many different similarity measures which are used for training such recommender. The most popular for collaborative filtering are Pearson correlation and Cosine similarity.

Algorithm:

For user $u \in \overline{1,m}$:

1. Calculate similarity between user u and all other users. For this could be used any preferred similarity measure.
2. Select top n users with the highest similarity to users u.
3. Calculate predictions for unknown ratings for user u as average of available ratings from n closest users or as weighed (on similarity distance) ratings of n closest users.

To find the best value of n, separate validation set or cross-validation could be used.

In our experiment, we have used Cosine similarity and selected *n* as 50 based on cross-validation. Results for different number of *n* are shown in the Figure 2. As experiment were run multiple times (to achieve more stable results), all estimates will be shown using boxplots.

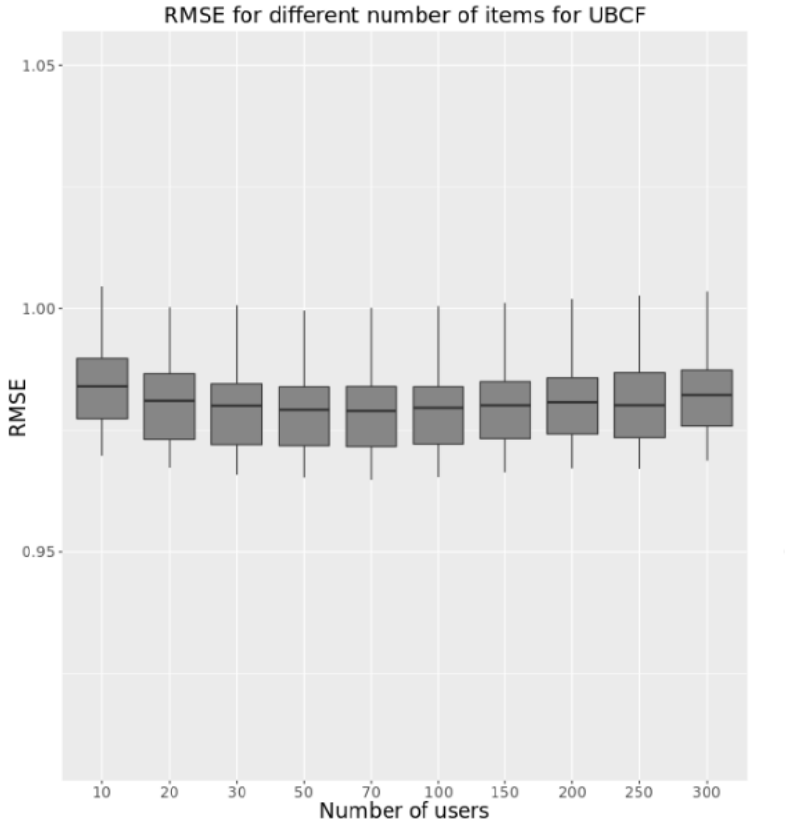

Figure 2. Cross-validation RMSE for different number of users

Source: Own research.

Even we have selected *n* as 50 as the best-estimated value, we could see that $n = 20$ is also good enough.

5.4. Item-based Collaborative Filtering

Item-based CF approach is very similar to user-based. But in this one, similarity, is computed between items, not users. Assumption is that users will prefer items similar to other items they like.

Algorithm:

1. Calculate similarity matrix between all items based on available users' ratings. For this could be used any preferred similarity measure.

2. For user $u \in \overline{1,m}$:

2.1. Store only n closest items to each item.

2.2. Calculate predicted rating for each item based on available ratings of user u by weighting available ratings of user on similarities.

As with user-based CF, we have used Cosine similarity and selected n as 350 based on cross-validation. Results for different number of n are shown in the Figure 3.

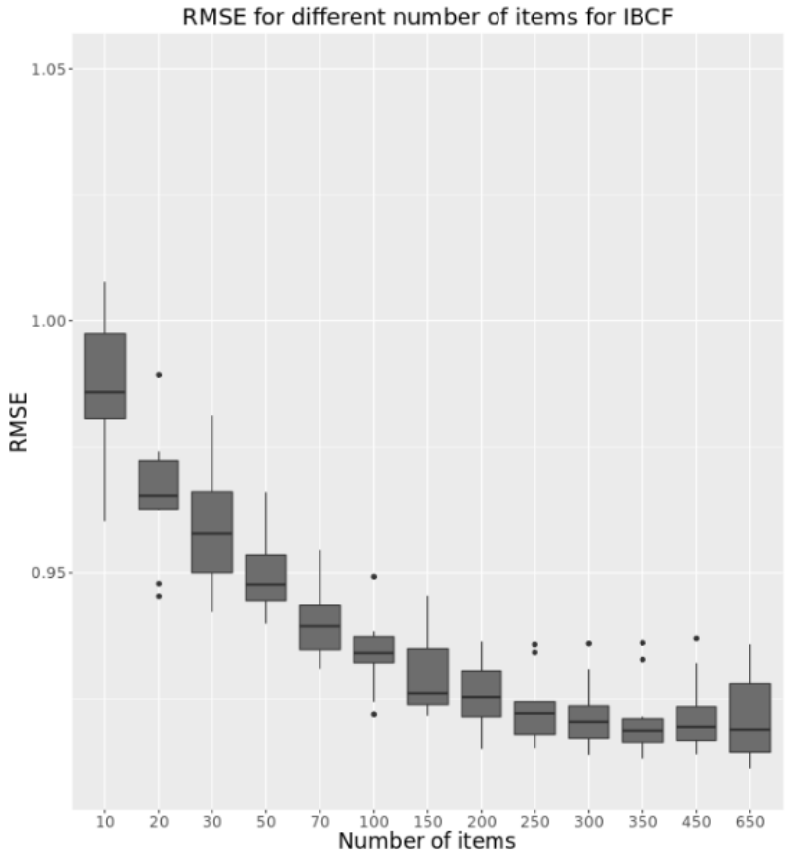

Figure 3. Cross-validation RMSE for different number of items

Source: Own research.

This plot shows us a little different behavior, than we saw for user-based CF. First of all, it is possible to achieve much better result with increasing number of nearest items. Moreover, the best value is much higher in comparison with user-based CF.

Item-based CF gives an ability to achieve lower RMSE on test set than user-based CF, what makes it more suitable for given dataset.

5.5. Slope One

Slope One was introduced by Lemire and Maclachlan [2005]. This algorithm is one of the simplest way to perform collaborative filtering based on items' similarity. This makes it very easy to implement and use, and accuracy of this algorithm equals to the accuracy of more complicated and resource-intensive algorithms.

Algorithm:

For item $i \in \overline{1,n}$:

1. Calculate average difference in ratings for item *i* and all another items $k \in \overline{1,n},\ k \neq i$:

$$Diff(i,k) = \frac{\sum_{j \in U_{ik}} r_{ji} - r_{jk}}{|U_{ik}|},$$

$i \in \overline{1,n},$

$k \in \overline{1,n}, k \neq i,$

U_{ik} – *indexes of users, who have available ratings for items i and k*.

2. Predict unknown rating for user *j* and item *k*, based on weighed differences in ratings of all known pairs of items:

$$r_{jk} = \sum_{p \in \overline{1,n},\ p \neq k} \frac{\left(r_{jp} + Diff(p,k)\right) \cdot |U_{kp}|}{\sum_{p \in \overline{1,n},\ p \neq k} |U_{kp}|}, r_{jk} = ?(\textit{missed rating}).$$

5.6. Matrix Factorization with Gradient Descend

Matrix Factorization is a popular technique to solve recommender system problem. The main idea is to approximate the matrix $R_{m\,x\,n}$ by the product of two matrixes of lower dimension: $P_{k\,x\,m}$ and $Q_{k\,x\,n}$.

Matrix P represents latent factors of users. So, each k-elements column of matrix P represents each user. Each k-elements column of matrix Q represents each item. So, to find rating for item i by user u we simply need to compute two vectors:

$$r_{ui} = P'[,u] \cdot Q[,i] .$$

Algorithm:

For item $i \in \overline{1,n}$:

1. Calculate matrixes P and Q, by minimizing the following function using stochastic gradient descend:

$$\min_{P,Q} \sum_{(u,i)\in R} \left(\left(r_{u,i} - p_u' \cdot q_i \right)^2 + \lambda \left(\left\| p_u \right\|^2 + \left\| q_i \right\|^2 \right) \right)$$

$r_{u,i}$ – *observed rating*,

λ – *penalty value to avoid overfitting*.

2. Predict ratings by multiplying matrixes P and Q.

This method has 3 parameters which should be tuned: number of factors (k), step of gradient descend and penalty on huge values. Its result highly depends on number of factors. Small number causes underfitting, high value – overfitting. Figure 4 shows test error for values in range from 1 to 20.

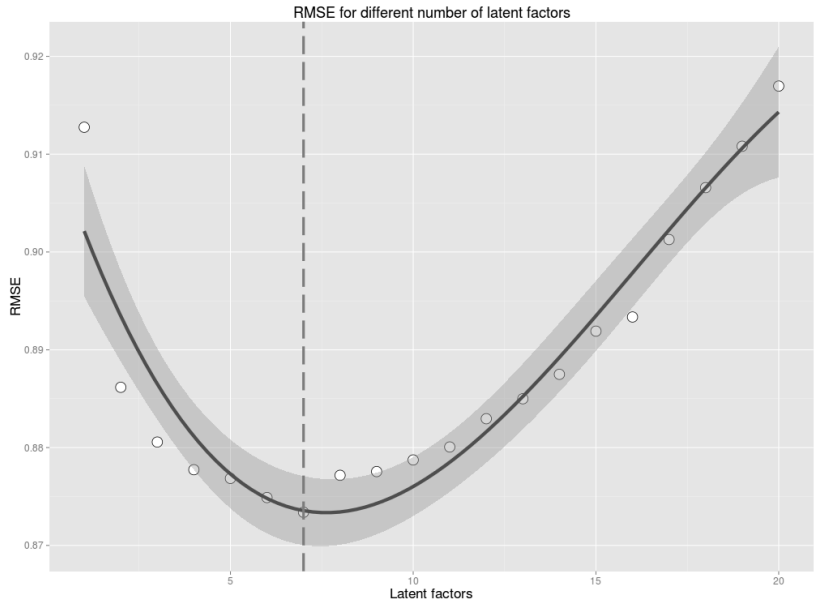

Figure 4. Test error for different number of latent factor for matrix factorization with stochastic gradient descend

Source: Own research.

5.7. SVD Approximation

Singular Value Decomposition (SVD) is based on a well-known matrix factorization method which takes an $m \times n$ matrix R and decomposes it as follows:

$$R = U \cdot S \cdot V^T .$$

After decomposition matrix R could be reconstructed back keeping only first r most significance entities. This gives an ability to predict missing values of ranking matrix [Lemire, Maclachlan, 2005].

Algorithm:

1. Replace all missing values with items' averages:

$$r_{jk} = r_{jk} - \overline{r_k}, \overline{r_k} = \frac{\sum_{j \in A_k} r_{jk}}{|A_k|}, k \in \overline{1,n}.$$

$$A_k - set\ of\ indexes\ of\ available\ ratings\ for\ item\ k .$$

2. Normalize matrix by subtracting users' averages (calculated based on initial rating matrix, not filled-in).
3. Perform Singular Value Decomposition of R:

$$R = U \cdot S \cdot V^T .$$

4. Keeping only first r rows of matrix U, r rows and r columns of matrix S and r columns of matrix V, reconstruct matrix R:

$$R_{predicted} = U[1:r,] \cdot S[1:r,1:r] \cdot V^T[,1:r] ,$$

r denotes number of latent factors of decomposition and best value of this parameter could be found using cross-validation or separate ratings for validation. For our dataset the smallest value of RMSE on validation set corresponds to $r = 24$. But, as for UBCF, we could decrease this number to 10-15, and this cost us almost no performance lose. Also, in the Figure 5, we could see that first 24 principal components explains 17% of variability, what is enough for the prediction. Using more components will cause overfitting.

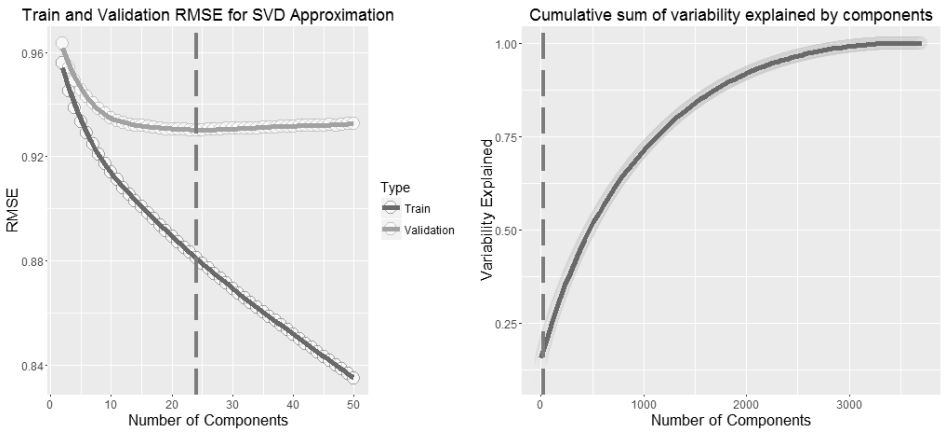

Figure 5. Train and validation RMSE for SVD approximation algorithm for different number of latent factors

Source: Own research.

6. Comparison of recommenders

Figure 6 shows RMSE estimates for different algorithms.

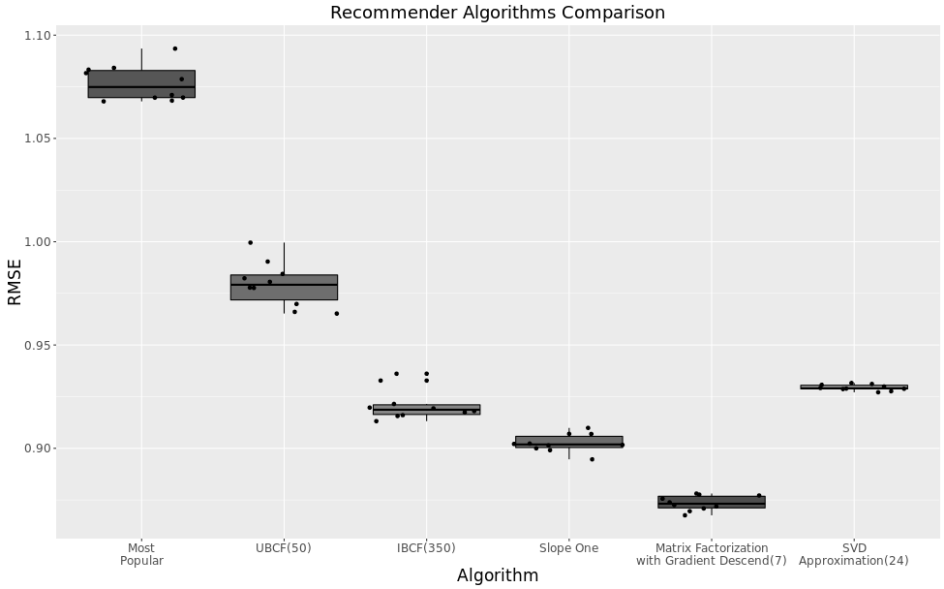

Figure 6. Comparison RMSE of recommender algorithms

Source: Own research.

From memory-based approaches Item-Based CF shows the best result. It performs much better than Most Popular and User-Based CF.

The best performance was shown by Matrix Factorization techniques with Stochastic Gradient Descend (0.873). Moreover, for this method it is enough to use only 7 factors to achieve good result. Slope One also performs well. In comparison with Matrix Factorization, it needs much less time for training.

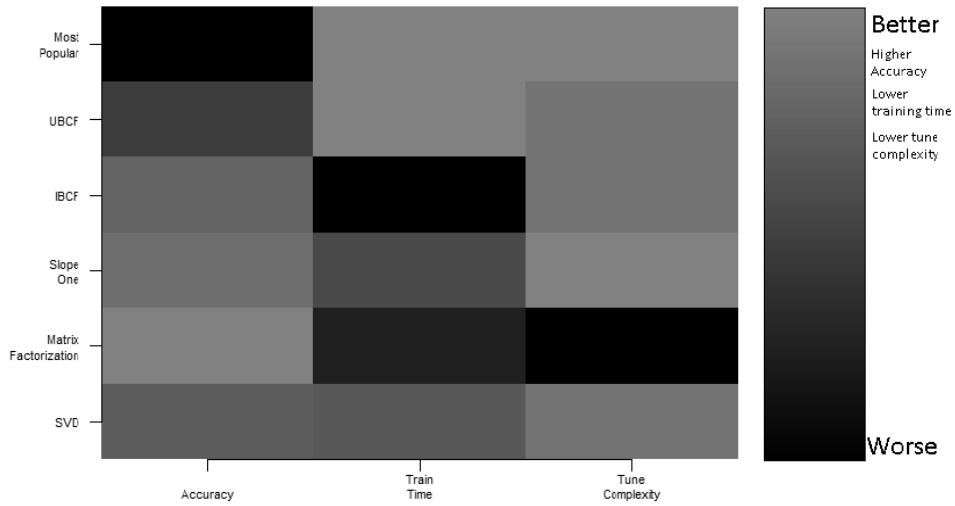

Figure 7. Comparison of recommender algorithms

Source: Own research.

In the Figure 7, we could see more full comparison of observed algorithms. Even Matrix Factorization is the best from accuracy perspective, it is very hard to tune and train it.

Conclusions

The research was focused on the comparison of few different approaches to build recommender system. "Most Popular Product" is one of the most widely used techniques for basic recommendations extraction. User-based and item--based algorithms are also a powerful way to do recommendations. Despite the fact that these algorithms were developed more than 20 years ago, they show great interpretability and could be easily used for small datasets. Factorization--based techniques like "SVD Approximation" or "Matrix Factorization with Gradient Descend" are base for state-of-the-art approaches for mining recom-

mendations in today's online world. There are a lot of different algorithms, which are related to factorization techniques, like non-negative/non-linear matrix factorization, weighted matrix factorization, etc. All of them are built based on the idea of decomposition of the matrix on two smaller one, the product of which should replicate original matrix. This class of algorithms is well on small and big datasets, shows great performance and accuracy, as was shown based on "Matrix Factorization with Gradient Descend" example in the article.

The analysis in this article was performed on 1M MovieLens dataset, where we have rates in the range from 1 to 5. But this does not conclude that received results are only applicable for data of the same nature. In online retail, there are two most popular data types: transactions and rating data. But transactions data could be transformed to rating matrix as well, by counting product numbers bought by some customer and normalizing this score with log or in some similar way. So, results received in this paper could be applied for different situations, not only with explicit rating data. What is more important, research results applicable for situations where sparsity of rating matrix is approximately the same as in test dataset (4,5%). Sparsity plays important role and algorithms could treat themselves differently with smaller or larger level of sparsity.

Recommendations play important role in today's online business. It is not only the way to find what to show to customer, expecting that she will click on it, but a way to do complex analysis of products and customers, detect patterns in customers behaviors, find way to sell some products and, what is important, invest money in targeted advertisement, understanding possible profit of it.

References

Breese J.S., Heckerman D., Kadie C. (1998), *Empirical Analysis of Predictive Algorithms for Collaborative Filtering* [in:] Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., pp. 43-52.

Gorakala S.K., Usuelli M. (2015), *Building a Recommendation System with R*, Packt Publishing Ltd.

Hahsler M. (2011), *Recommenderlab: A Framework for Developing and Testing Recommendation Algorithms*, Southern Methodist University.

Hauger S., Tso K.H., Schmidt-Thieme L. (2008), *Comparison of Recommender System Algorithms Focusing on the New-Item and User-bias Problem* [in:] Ch. Preisach, H. Burkhardt, L. Schmidt-Thieme, R. Decker (eds.), *Data Analysis, Machine Learning and Applications*, Springer, Berlin–Heidelberg, pp. 525-532.

Huang Z., Zeng D., Chen H. (2007), *A Comparison of Collaborative-Filtering Recommendation Algorithms for e-Commerce*, "IEEE Intelligent Systems", No. 22(5), pp. 68-78.

Lee J., Sun M., Lebanon G. (2012), *A Comparative Study of Collaborative Filtering Algorithms*, arXiv preprint, arXiv:1205.3193.

Lemire D., Maclachlan A. (2005), *Slope One Predictors for Online Rating-Based Collaborative Filtering* [in:] Proceedings of the 2005 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, pp. 471-475.

Ricci F., Rokach L., Shapira B. (2015), *Recommender Systems: Introduction and Challenges* [in:] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (eds.), *Recommender Systems Handbook*, Springer, US, pp. 1-34.

Vozalis M., Markos A., Margaritis K. (2009), *Evaluation of Standard SVD-based Techniques for Collaborative Filtering* [in:] Proceedings the 9th Hellenic European Research on Computer Mathematics and its Applications.

Zhang S., Wang W., Ford J., Makedon F., Pearlman J. (2005), *Using Singular Value Decomposition Approximation for Collaborative Filtering* [in:] "E-Commerce Technology", July, CEC 2005, Seventh IEEE International Conference, pp. 257-264.

[www 1] MovieLens project, https://movielens.org/ (access: 2016).

[www 2] Reproducible code for research: https://rpubs.com/tarashnot/recommender_comparison (access: 2016).

ZASTOSOWANIE ZALECEŃ REKOMENDACJI DO OCENY MACIERZY W MARKETINGU ONLINE

Streszczenie: Głównym celem badania jest wykrycie zalet i wad różnych algorytmów wykorzystywanych podczas budowania systemu rekomendacji. Systemy rekomendujące stały się tak popularne ze względu na aktywny rozwój marketingu internetowego i wzrost sprzedaży za pośrednictwem Internetu. Opracowanie i wdrożenie strategii rekomendowania produktów powoduje efektywne wykorzystanie zasobów firmy i dynamiczną sprzedaż. Systemy rekomendujące są jednym z najbardziej efektywnych narzędzi – systemów, które są zbudowane przy użyciu algorytmów opartych na pamięci i systemów z algorytmami opartymi na modelach. Najlepszą wydajność pokazały techniki Matrix Factorization ze Stochastic Gradient Descend. Wybierając system rekomendujący, należy wziąć pod uwagę cel używania, cechy produktu, specyfikacje i dostępność danych klienta według ich preferencji. Korzystanie z jednego z opisanych systemów rekomendujących poprawi efektywność marketingu produktów.

Słowa kluczowe: systemy rekomendujące, systemy rekomendujące oparte na pamięci, modele rekomendujące oparte na modelach, faktoryzacja macierzowa.