BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **16**, No 6
Special issue with selection of extended papers from 6th International Conference on Logistic, Informatics and Service Science LISS'2016

Sofia • 2016 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2016-0092

Application of Improved Recommendation System Based on Spark Platform in Big Data Analysis

Li Xie, Wenbo Zhou, Yaosen Li

Institute of Disaster Prevention, Yanjiao Town 065201, China
Email: xieli@cidp.edu.cn

Abstract**:** *In the era of big data, people have to face information filtration problem. For those cases when users do not or cannot express their demands clearly, recommender system can analyse user's information more proactive and intelligent to filter out something users want. This property makes recommender system play a very important role in the field of e-commerce, social network and so on. The collaborative filtering recommendation algorithm based on Alternating Least Squares* (*ALS*) *is one of common algorithms using matrix factorization technique of recommendation system. In this paper, we design the parallel implementation process of the recommendation algorithm based on Spark platform and the related technology research of recommendation systems. Because of the shortcomings of the recommendation algorithm based on ALS model, a new loss function is designed. Before the model is trained, the similarity information of users and items is fused. The experimental results show that the performance of the proposed algorithm is better than that of algorithm based on ALS.*

Keywords**:** *Spark, recommendation system, collaborative filtering, alternating least squares.*

1. Introduction

With the popularity of the Internet and the rapid growth of the number of Internet users, the information on the Internet presents explosive growth. Although the mass of information can meet the information needs of Internet users, a serious challenge of processing information has to be handled. Users can not search for the needed information from the vast amount of available information quickly and accurately. Under this background, recommendation systems arise. Recommendation system can learn the user's interest and behavior patterns through collection and analysis of user's information, which helps to recommend services for the particular user.

Due to the user's difference and personalized recommendation information, the recommendation system has much better performance than the traditional search engine. An excellent recommendation system not only predicts the users' preferences to enhance their experience accurately, but also allows enterprises benefit quite a lot. According to VentureBeat statistics, Amazon's recommendation system can provide 35% of the sales of goods [1]. Collaborative Filtering (CF) is one of the most successful techniques to construct a recommendation system in the real world. It can forecast the preference of other unknown users and provide personalized recommendation by analysing the existing partial users' preferences. It has been widely used in commercial websites including Amazon, Netflix, Hulu, eBay, Taobao, etc.

Wang and Zhao [2] proposes the Online Multi-Task learning algorithm based on CF (OMTCF), which can improve the recommendation accuracy effectively. Koren [3] proposes an algorithm in which the time information is considered as one of the users' characteristics. This algorithm is called TimeSVD++, and can solve the problem of time drift. Ling, Yang and King [4] proposes SGD-RMF/DA-RMF Algorithm, which can solve the dynamic changes of users. Jamali and Ester [5] proposes the TrustWalker Algorithm based on random walk model, which is a good way to deal with the problem of interest.

The recommendation algorithm includes collaborative filtering, nearest neighbour clustering, content based recommendation, Bayesian network, association rules, and so on. Collaborative filtering can be divided into memory-based and model-based filtering. The collaborative filtering based on memory is used to calculate the historical information of the existing users in the system and the nearest neighbour of the target user. Then it uses the nearest neighbour to predict the degree of preference of the target user to the item. The collaborative filtering algorithm based on model is used to train the prediction model, which is used to predict the model.

Online learning algorithm is fast, simple and based less on statistical hypothesis. The first order online learning algorithm is proposed by Crammer et al., and is called Passive-Aggressive. Recently, researchers have proposed second order online learning algorithm to improve the effect of online learning through learning confidence information. Dredze, Crammer and Pereira [6] proposed the Confidence-Weight (CW) learning algorithm by maintaining the Gauss distribution; it is applied to control parameter update size and orientation. Other second order online learning algorithms are Adaptive Regularization Of Weight (AROW) [7], New Adaptive Regularization Of Weight (NAROW) [8], Soft Confidence Weighted (SCW) and so on [9]. These algorithms are used for classification firstly; most of the online collaborative filtering algorithms, such as gradient descent method, mean value and so on, are also used by the first order optimization method. They ignore the second order informational. Lu, Hoi and Wang [10] propose Confidence Weighted Online Collaborative Filtering (CWOCF) Algorithm. It combines second order collaborative filtering and online learning. Although the AROW learning rules in the CWOCF Algorithm have better performance, its update rules are still strong, which may over-fit in some cases. The noise data processing by AROW still needs to be improved, and the computational cost can be reduced.

In this paper, we design the parallel implementation process of the recommendation algorithm based on the Spark platform and on the related technology research of recommendation system. The function of the cluster nodes and the distribution of the task are analysed in detail after the algorithm is submitted. Secondly, parallel implementation of the recommendation algorithms based on Spark platform is described here; it includes collaborative filtering based on users, collaborative filtering based on terms and recommendation algorithm based on ALS model. We make a detailed implementation of the parallel processing. Finally, a detailed analysis of the implementation process of the Spark storage algorithm is carried out. In this paper we also present a new loss function which is designed because of the shortcomings of the recommendation algorithm based on ALS model. Before the model is trained, the similarity information of users and items is fused. The experimental results show that the performance of Spark is better than that of Hadoop in the parallel implementation of the recommendation algorithms. Compared to the optimization scheme of the recommendation algorithm proposed by ALS, our algorithm has better performance than algorithm based on ALS.

2. Results and discussion

2.1. The introduction of Spark

In recent years, the large data computing platform called Spark has been widely concerned due to the popularity of big data. Spark is an open source parallel computing framework of BerkeleyAMP lab, which is similar to the Hadoop MapReduce. It is a kind of fast and universal data analysis engine based on memory. Compared with Hadoop MapReduce, Spark' speed of iterative calculation is faster. At present, a lot of sub projects have been derived in respect to the development of large data calculation process based on the Spark platform. Berkeley University regards entire ecosystem of the Spark as Berkeley Data Analysis Stack (BDAS), which is shown in Fig. 1. On the basis of the core framework of Spark, it mainly provides four categories of computing framework including Streaming Spark, Graphx, MLbase, and SparkSQL. Streaming Spark mainly supports the stream computing. Graphx is a parallel graph computing framework. MLlib mainly supports the underlying distributed machine learning and machine learning capabilities. SparkSQL is a query engine that supports SQL query and analysis of structured data. Because of these sub projects, Spark provides a more high-level, more extensive computing model. We introduce each sub item in detail from the four parties to have a better understanding of Spark.

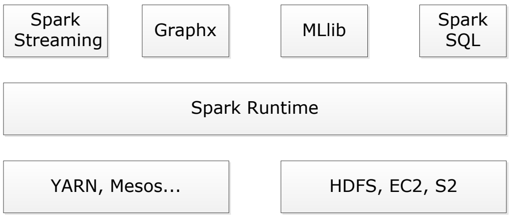

Fig. 1. The Berkeley data analysis stack

1) **Streaming Spark.** The main function of Streaming Spark is that the flow of data is accumulated to RDD based on the length of time. Then the RDD is processed in batch, so that it can achieve the function of large scale data processing. Due to this kind of working mode, its output is larger than that of the current mainstream framework. At the same time, it also provides multiple APIs used in the calculation of the stream data.

2) **Graphx.** Graphx is based on the BSP model, which provides an interface similar to Google's Pregel. It provides large-scale global synchronization of graph computation. Spark's advantage is particularly evident in the case of many iterative times, because it is based on the memory. GraphX splits the graph into a number of sub graphs firstly, and then computes them based on these sub graphs. It can be carried out in the calculation of the iterative operation and to achieve the task of parallelization.

3) **MLlib.** Spark is a computing framework based on memory. If the machine memory is relatively large, the RDD data will be all in memory. In this way, the gradient descent, EM algorithm and other machine learning iterative algorithms are able to run very efficiently. MLlib project is a machine learning solution in Spark environment. At present, the machine learning algorithms realized by MLlib mainly include support vector machine, decision tree, naive Bayes, K.Means, singular value decomposition, and so on. Developers can use MLlib without having a professional machine learning knowledge.

4) **SparkSQL.** In Hadoop system, Hive is a data warehouse query tool that uses HiveSql to query data in HDFS or HBase. SparkSQL is a tool similar to Hive. Due to the architecture in the Spark environment, SparkSQL has higher and faster efficiency than Hive. SparkSQL can read the local files and also can read the HDFS files; the file type usually includes the text type or Parquet file type. It can be used to build a data source RDD in the program; the RDD is converted to the warehouse table to deal with data set by a distributed SQL-like language.

2.2. Design idea of Spark

The idea of Spark is to design a kind of new fault-tolerant method in order to reduce the I/O overhead of network and disk. In order to achieve this goal, a new data format RDD is born. RDD is a read-only data block, which can be obtained by reading the data from the storage system or by the operation of other RDD (including Transformation and Action). The read-only property of the RDD data indicates that if an RDD data block has to be operated, the result is a new RDD. In this case, the same variable is used to represent the RDD before and after the transformation. The data inside RDD is not the real data, but is information of some metadata. In the computer system, there is a noun called lineage, which is used to indicate the transformation of this relationship before and after the transformation. Through the lineage, the entire results of calculation no longer need to be stored in the HDFS. If a node has an error, it is only necessary to re-calculate the lineage relationship, which can be used for fault tolerance. Due to this design idea, Spark can build a one-stop solution strategy.

1) Based on the core of Spark, a variety of computing methods are provided to make an efficient data pipeline. Compared to the MapReduce, the Spark provides much more complex query operations, in addition to the simple operation "map" and "reduce", which includes streaming computing, machine learning, graph computation and so on. Therefore, users can use any of these features.

2) It is supported by multiple languages. Spark official website announces that three kinds of interface languages are supported, which includes Scala, Python, Java. Developers are allowed to choose their own language according to their own interests. At the same time, Spark provides its own shell, which is more convenient for users to improve its ease to interact with Spark.

3) It can be compatible with a variety of underlying storage systems. Spark can run on any of the data sources in Hadoop, such as HDFS, Hive, Hbase, and so on. This feature allows the developers and users to migrate from the original system to the Spark system easily.

2.3. Comparison between Spark and MapReduce

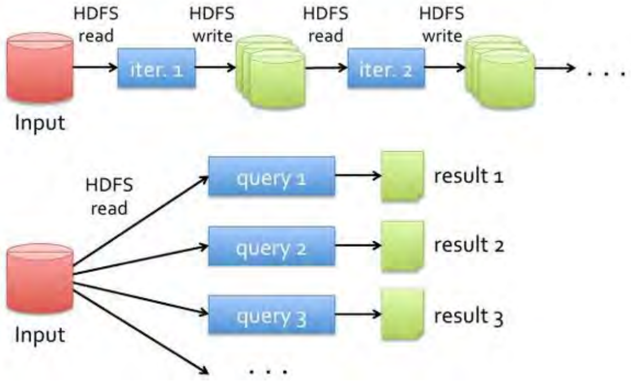

Fig. 2. Operation process of MapReduce

Spark and MapReduce are big data computing frameworks. The difference between them is mainly in the following two aspects:

1) On Fig. 2, the Shuffle process between the Mapper output and the Reducer input in MapReduce requires frequent reading and writing disks. This process is very slow; it is an important bottleneck restricting the performance of the MapReduce framework. Spark is a computing system based on memory; the data exchange in each of the RDDs is carried out mostly in memory, as is shown on

Fig. 3. In the area of machine learning, it requires frequent iterations. Spark officials claim that the performance of Spark is 100 times more than that of MapReduce.

2) In the programming model, the MapReduce framework provides only two kinds of operators (map and reduce). However, Spark provides up to several dozens of operators, which can be divided into two categories (Transformations and Actions). There are map, filter, flatMap, sample, groupByKey, reduceByKey and other Transformations operations. The actions operations include count, collect, reduce, take and so on. Because these high-level APIs achieve a lot of data operations, for the Spark program can be very simple to achieve a very powerful function.

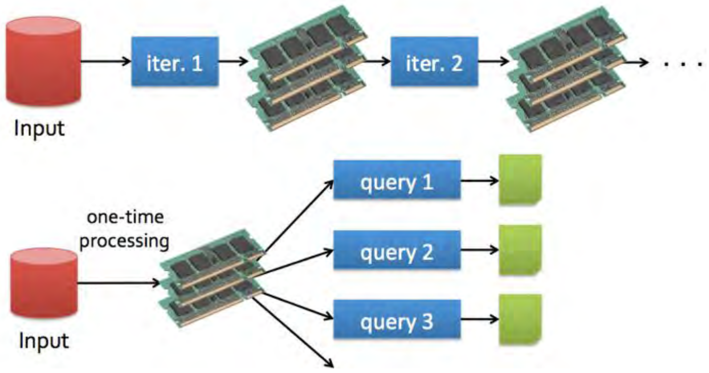

Fig. 3. Operation process of Spark

2.4. Introduction of the recommendation system

Personalized recommendation engine is meant to do information filtering; it tries to find the items that are interesting for users from a large number of items. These items can be of any type, such as movies, music, books, websites and news. The score of one item for the user reflects the degree of user's interest. The recommendation system can predict the item that user has never used before, and can give a score for each item. Then the recommendation engine can recommend the highest scoring items to the users according to these scores.

The data source of the recommendation engine includes the metadata of articles or the content, the basic information of the user, and the preference of the user to the item. Users' preference information is divided into explicit feedback and implicit feedback. Explicit feedback includes a user's rating of items, comments, or tags. Implicit feedback is the data generated by the user when visiting websites, such as users' browsing, item collection or page residence time. The explicit user behaviour can accurately reflect the user's preferences, but an additional cost has to be paid. The implicit user behaviour can also reflect the user's preferences, but it may not be very accurate, because there is a lot of noise in the data. The recommendation engine uses some of the data from the data source according to the different recommendation mechanisms, which to analyse the user's interest directly or build a certain rule model.

2.5. Collaborative filtering recommendation algorithm

Collaborative Filtering (CF) algorithm is the most classical recommendation algorithm among several commonly used recommendation algorithms. The core idea is to use group wisdom to do recommendation operation. Collaborative filtering is divided into collaborative filtering based on User (User-CF) and collaborative filtering based on Item (Item-CF). User-CF is used to find similar users according to the behaviour record firstly, and then to do recommendation operation according to the similarity of users. There are many ways to calculate the similarity, and the Pearson Product-Moment correlation coefficient is used to calculate the score,

$$\text{(1)} \qquad \mathrm{sim}(i,j)=\frac{\sum_{p\in P}(r_{i,p}-\overline{r_a})(r_{j,p}-\overline{r_b})}{\sqrt{\sum_{p\in P}(r_{i,p}-\overline{r_a})^2}\sqrt{\sum_{p\in P}(r_{j,p}-\overline{r_b})^2}},$$

where $U=\{u_1,u_2,...,u_m\}$ is the user set, $I=\{i_1,i_2,...,i_n\}$ is the item set, R is $m\times n$ evaluation matrix; $\mathrm{sim}(i,\,j)$ is similarity between user i and user j; $r_{i,p}$ and $r_{j,p}$ are the scores of item p by user i and user j; $\overline{r_a}$ and $\overline{r_b}$ are the average scores of items by user i and user j.

For Top-*N* recommendation, the most simple similarity calculation is to use Jaccard formula:

$$\text{(2)} \qquad \mathrm{sim}(i,\,j)=\frac{|N(i)\cap N(j)|}{|N(i)\cup N(j)|},$$

where $N(i)$ and $N(j)$ indicate the item that user i and user j have used before. The formula in the molecule is the number of the items that user i and user j used before. The denominator in the formula is the number of the items that user i or user j used before.

CF-Item is used to find similar item by the behaviour firstly, and then recommend similar items to users according to the users' choice. The similarity of the goods is calculated according to the user's behaviour information, rather than the information itself. The similarity calculation of CF-Item can also be calculated by the Pearson Product-Moment correlation coefficient shown in (1), or can be calculated by the cosine similarity

$$\text{(3)} \qquad \mathrm{sim}(u,\,v)=\frac{\overline{u}\cdot\overline{v}}{|\overline{u}||\overline{v}|},$$

where $\overline{u}$ and $\overline{v}$ represent the score vector of item *u* and item *v*, respectively. There are many ways to calculate the similarity. In addition to the introduced above, there are European distance, Manhattan distance, similarity calculation based on the hash method, and so on. Each method has its advantages and disadvantages, for example Euclidean distance is suitable to calculate the dimension of small vector similarity, and cosine theorem is suitable for the larger dimension of the similarity calculations.

2.6. Collaborative filtering algorithm based on ALS

Collaborative filtering recommendation algorithm based on matrix decomposition model mainly includes SVD and ALS. In the computational step of SVD, the matrix R' is obtained from the user rating matrix R with the weighted average, and then the matrix R' is decomposed by using mathematical SVD. In [11] is proposed a new SVD++ model. However, the computational complexity of such method is very high; it is difficult to apply it to a real recommendation system [12]. Zhou, Wilkinson and Schreiber [13] proposes collaborative filtering algorithm based on the Alternating Least Squares (ALS), which is a powerful matrix decomposition algorithm. It can be very good to extend to the distributed computing and solve the data sparse problem. The following describes the principle of collaborative filtering algorithm based on ALS.

Matrix $R=(R_{ij})^{m\times n}$ represents the rating matrix of n items by m users. We hope to find a low rank matrix X to approximate the matrix R, $X=UV^{\mathrm{T}}$, $U\in C^{m\times d}$, $V\in C^{n\times d}d$, and d represents the number of eigenvalues. In general, $d\ll r$, $r\approx\min(m,n)$, and r represents the rank of the matrix, and the loss function is as follows:

$$L(U,V)=\sum_{ij}(R_{ij}-X_{ij})^2, \quad (4)$$

where $(R_{ij}-X_{ij})^2$ is a common error term in the low rank approximation. Here we need to find a way to solve the optimization problem $\arg\min_x L(x)$, which is to minimize the loss function. Formula (4) can be re-written as follows:

$$L(U,V)=\sum_{ij}(R_{ij}-U_iV_j^{\mathrm{T}})^2. \quad (5)$$

In order to prevent over-fitting, the second order regularization term is added to the above formula:

$$L(U,V)=\sum_{ij}(R_{ij}-U_iV_j^{\mathrm{T}})^2+\lambda(\|U_i\|^2+\|V_i\|^2). \quad (6)$$

If V is known, the ridge regression can be used to predict each line of U, and vice versa. Therefore, the matrix V is fixed, then the derivative of the U_i is taken; now we can obtain the following formula for U_i:

$$U_i=R_iV_{ui}^{\mathrm{T}}(V_{ui}^{\mathrm{T}}V_{ui}+\lambda n_{ui}I)^{-1},\ i\in[1,m], \quad (7)$$

where R_i represents the score vector for the item by user i; V_{ui} represents the characteristic matrix formed by the characteristic vectors of the items evaluated by the user i; U_{ui} represents the number of items evaluated by the user i. In the same way, the matrix U is fixed, then the derivative of V is taken; we can obtain the following formula for V:

$$V_j=R_j{}^{\mathrm{T}}U_{mj}(U_{mj}^{\mathrm{T}}U_{mj}+\lambda n_{mj}I)^{-1},\ j\in[1,n], \quad (8)$$

where R_j represents the score vector for the item by user j; U_{mj} represents the characteristic matrix formed by the characteristic vectors of the items evaluated by the user j; n_{mj} represents the number of items evaluated by the user j; I is $d \times d$ unit matrix.

Based on the ALS collaborative filtering algorithm is performed, namely the formulae (4) and (5) are called alternately to update U and V. Calculation processing ends if the result of the calculation is convergence or the number of iterations reaches the maximum value. Finally, the approximation matrix X is got to recommend items for users.

When the algorithm is implemented on Spark, the original data set is stored in the distributed file system HDFS. Then, the data on the HDFS is read and transformed into a compression matrix to create RDD according to the transformed matrix data. The intermediate data U and V generated in each iteration, as well as the data set are cached to memory.

2.7. Collaborative filtering algorithm based on improved ALS

The definition of loss function is particularly important in the ALS model training method described in Section 4.1. In the processing, we find that the matrix U and matrix V can lose some information of users or items. The similarity between the users and items which is obtained by matrix U and matrix V is not similar to that between the users and items after training. So, we need to design a new loss function that can take into account the similarity between the user and the item. After the model is trained, their similarities will be same as the former similarities. The pseudo-code of specific process of the algorithm is as follows:

```
Input: user's rating matrix $R$
Output: matrix $U$ and matrix $V$
For $u$ from 1 to $N - 1$
For $v$ from $u + 1$ to $N$
   Calculate sim($u$, $v$) according to Formula (1)
End for
For $m$ from 1 to $M - 1$
For $n$ from $m + 1$ to $M$
   Calculate sim($m$, $n$) according to Formula (2)
End for
For $i$ from 1 to Iterations
     Fixed $V$
     Calculate $U$ according to Formula (7)
     Fixed $U$
     Calculate $V$ according to Formula (8)
End for
```

3. Results and discussion

3.1. Experimental environment and data

We build a total of Spark cluster that contains six virtual machines; one is the main node (master), the others are the sub-nodes (workers); they are running on the OpenStack cloud platform. The main configuration of each virtual machine is as follows: 4 virtual kernels, memory is 128G, and disk is 500G. Java version is JaVa-7-oracle, and Linusystem is the Ubulltul 12.04, Spark version is 1.0.0, Hadoop version is 2.2.0. In this experiment, we use the data set of MovieLens in the website. We randomly selected 10 thousands of the 1 million data-items as the experimental data. Randomly 80% of the data set is selected as a training set, and 20% is selected as a test set.

3.2. Results and discussion

In this paper, we use Root Mean Square Error (RMSE) to evaluate the accuracy of the prediction. The smaller RMSE, the higher the accuracy is,

(9) $$\mathrm{RMSE} = \frac{\sqrt{\sum_{u,i \in T} (r_{ui} - \overline{r_{ui}})^2}}{|T|},$$

where r_{ui} represents the actual score of the user u on the film *i*, and $\overline{r_{ui}}$ is predicted by the recommendation algorithm.

Table 1. The results of different algorithms

Iterations	Algorithm based on ALS	Improved algorithm
20	1.1132	1.0816
40	0.9124	0.8741
60	0.8696	0.8264
80	0.7935	0.7682
100	0.7254	0.7016

In Table 1, we can find the different results of these two algorithms in different iterations. The RMSE decreases with the increase of the number of iterations. Comparing with these two different results, the root mean square of our new algorithm is smaller than that of algorithm based on ALS in different iterations. The average RMSE in the five iterations decreases by 3.7% by the proposed algorithm.

4. Conclusion

With the popularity of the Internet, it becomes very difficult for people to search the information they need from the vast amounts of information. Recommendation system can recommend related information to users intelligently through analysis of the interests and behaviours of users. The collaborative filtering recommendation algorithm based on ALS is one of most common algorithms using matrix factorization technique in recommendation systems. It combines a lot of ratings data to calculate and store characteristic matrix in the process of calculation, so it may encounter the bottleneck of computation speed, if it runs on a single node. Spark is a new kind of distributed computing platform in the big data era and has excellent computing performance. In this paper, firstly, we make research on the existing collaborative filtering algorithm based on ALS and the big data distributed computing platform of Spark. Then, we realize the shortcomings of the recommendation algorithm based on ALS model. Finally, a new loss function is designed. Before the model is trained, the similarity information of users and items is fused. The experimental results show that the performance of our algorithm is better than that of algorithm based on ALS.

References

1. Liu, J. G., T. Zhou, B. H. Wang. Research Progress of Personalized Recommendation System. – Regress in National Science, Vol. **19**, 2009, No 1, pp. 1-15.
2. Wang, J., P. Zhao. Online Multi-Task Collaborative Filtering for On-the-Fly Recommender Systems. – In: Proc. of 7th ACM Conference on Recommender Systems, New York, ACM, 2013, pp. 237-244.
3. Koren, Y. Factorization Meets the Neighbourhood: A Multifaceted Collaborative Filtering Model. – In: Proc. of 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, ACM, 2008, pp. 426-434.
4. Ling, G., H. Yang, I. King. Online Learning for Collaborative Filtering. – In: Proc. of 2012 International Joint Conference on Neural Networks, IEEE, 2012, pp. 1-8.
5. Jamali, M., M. Ester. TrustWalker: A Random Walk Model for Combining Trust-Based and Item-Based Recommendation. – In: Proc. of 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, ACM, 2009, pp. 397-406.
6. Dredze, M., K. Crammer, F. Pereira. Confidence-Weighted Linear Classification. – In: Proc. of 25th International Conference on Machine Learning, Helsinki, ACM, 2008, pp. 264-271.
7. Crammer, K., A. Kulesza, M. Dredze. Adaptive Regularization of Weight Vectors – Machine Learning, Vol. **91**, 2013, No 2, 155-187.
8. Orabona, F., K. Crammer. New Adaptive Algorithms for Online Classification. – In: Proc. of 24th Annual Conference on Neural Information Processing Systems, Vancouver, 2010, pp. 1840-1848.
9. Wang, J., P. Zhao, S. C. H. Hoi. Exact Soft Confidence-Weighted Learning. – In: Proc. of 29th International Conference on Machine Learning. Edinburgh, Scotland, 2012.
10. Lu, J., S. Hoi, J. Wang. Second Order Online Collaborative Filtering. – In: Proc. of Asian Conference on Machine Learning, 2013, pp. 325-340.
11. Pan, R., Y. Zhou, B. Cao. One-Class Collaborative Filtering. – In: Proc. of 8th IEEE International Conference Data Mining, 2008, ICDM'08, IEEE, 2008, pp. 502-511.
12. Liu, Q. Research on the Key Algorithm in Collaborative Filtering Recommendation System. Zhejiang University, 2013.
13. Zhou, Y. H., D. Wilkinson, R. Schreiber. Large Scale Parallel Collaborative Filtering for the Netflix Prize. – In: Proc. of 4th International Conference on Algorithmic Aspects in Information and Management, ShangHai, Springer, 2008, pp. 337-348.