Many advanced recommender models are implemented using matrix factorization algorithms. Experiments show that the quality of their performance depends significantly on the selected hyperparameters. Analysis of the effectiveness of using various methods for solving this problem of optimizing hyperparameters was made. It has shown that the use of classical Bayesian optimization which treats the model as a «black box» remains the standard solution. However, the models based on matrix factorization have a number of characteristic features. Their use makes it possible to introduce changes in the optimization process leading to a decrease in the time required to find the sought points without losing quality.

Modification of the Gaussian process core which is used as a surrogate model for the loss function when performing the Bayesian optimization was proposed. The described modification at first iterations increases the variance of the values predicted by the Gaussian process over a given region of the hyperparameter space. In some cases, this makes it possible to obtain more information about the real form of the investigated loss function in less time.

Experiments were carried out using well-known data sets for recommender systems. Total optimization time when applying the modification was reduced by 16 % (or 263 seconds) at best and remained the same at worst (less than 1-second difference). In this case, the expected error of the recommender model did not change (the absolute difference in values is two orders of magnitude lower than the value of error reduction in the optimization process). Thus, the use of the proposed modification contributes to finding a better set of hyperparameters in less time without loss of quality

Keywords: Bayesian optimization, Gaussian process, covariance function, matrix factorization, recommender systems

UDC 004.852
DOI: 10.15587/1729-4061.2021.239124

DEVELOPMENT OF A HYPERPARAMETER OPTIMIZATION METHOD FOR RECOMMENDER MODELS BASED ON MATRIX FACTORIZATION

Alexander Nechaev
Corresponding author
Postgraduate Student*
E-mail: dapqa@yandex.ru
Vasily Meltsov
PhD, Associate Professor*
Dmitry Strabykin
Doctor of Technical Sciences, Professor*
*Department of Computer Science
Vyatka State University
Moskovskaya str., 36,
Kirov, Russian Federation, 610000

Received date 23.08.2021
Accepted date 04.10.2021
Published date 29.10.2021

How to Cite: *Nechaev, A., Meltsov, V., Strabykin, D. (2021). Development of a hyperparameter optimization method for matrix factorization based recommender models. Eastern-European Journal of Enterprise Technologies, 5 (4 (113)), 45–54. doi: https://doi.org/10.15587/1729-4061.2021.239124*

1. Introduction

Recommender systems (RS) are software systems that solve the problem of predicting user interest in objects from a certain subject area [1]. Depending on the context, objects and users can mean different entities. For example, products and customers in e-commerce, content and its consumers in media services, actions and situations in decision-making support systems, prescriptions and patients in medical systems. This makes it possible to use common mechanisms for effective solutions to problems of completely different directionalities.

One of the most widespread approaches to constructing an RS is based on the use of collaborative filtering methods. With this approach, interest predictions are calculated on the assumption that similar users rate the same items in the same way. Depending on the chosen method of determining the proximity of users, various implementations of recommender models are created. When the task of forming a list of recommendations is reduced to the task of predicting the numerical assessment (rating) of objects, advanced results demonstrate latent-factor models. Such models associate each user and object with a vector of latent factors, i.e. implicit features that characterize entities and patterns of their interaction. The learning process consists in calculating the sought vectors. Since the input data usually are a matrix of existing ratings explicitly or implicitly given by users to the objects with which they interacted, the latent factors can be calculated using matrix factorization algorithms. Despite the simplicity of the described idea and the steady development of new methods of generating recommendations, various versions of collaborative filtering models based on matrix factorization algorithms remain in demand. They demonstrate advanced results in solving the rating prediction problem outperforming more sophisticated machine learning methods in many cases [2, 3].

An important problem that arises when training models with matrix factorization algorithms is a choice of hyperparameters of such algorithms. The values used have a direct impact on the quality of rating predictions by the final model. At the same time, their optimal values can change over time as the amount of data for training grows and initial patterns of interaction between users and objects are transformed into new ones. Consequently, the resource-demanding optimization process is repeatedly performed at the stages of developing a recommender model and teaching it new data during operation. Reducing overall optimization time

without sacrificing quality will make it possible to fasten experimentation and actualization of production models. This substantiates the relevance of developing new effective methods for optimizing the hyperparameters of the matrix factorization algorithms.

2. Literature review and problem statement

When analyzing and developing approaches to RS design, it is important to take into account the fact that their ultimate task consists in listing the recommendation users. Systems based on matrix factorization solve this problem by sorting objects according to the calculated ratings. It is argued in [4] that this approach leads to the following problem. Even minor fluctuations in the rating prediction error entail significant changes in the quality of the samples formed by the recommender system.

Experimental testing of latent-factor models confirms the existence of the indicated problem. For example, the study [5] investigates the operation of the SVD model on the well-known Movielens 100k data set [6]. This set contains the ratings given by users to the movies they have watched. According to the presented results, the models trained using different regularization constants generate different sets of recommendations. In this case, the prediction error changes accordingly. However, the influence of the selected values of hyperparameters was considered in [5] only for isolated cases.

A much more detailed study of this dependence for the SVD model on the Movielens 100k dataset was given in [7]. The presented results show that the studied error can vary significantly (often by more than 10 %) depending on the chosen learning rate, the number of factors and constants of regularization. A large number of tests have been performed and detailed graphs were presented that reflect the dependence under consideration. Nevertheless, in order to develop methods that improve the process of optimizing hyperparameters, it is important to consider the features of functioning of other models as well and use datasets of different sizes.

Broader studies of these features of the models based on matrix factorization were carried out in [8] by the authors of the current study. The models trained by the SVD and SVD++ algorithms were tested on larger datasets using different values of the number of factors and regularization constants. Despite the use of a broader set of algorithms and data sets, the dependence of the prediction error on the selected hyperparameters remained within similar limits and was subject to the same characteristic features. Preservation of this dependence is observed with further model complications as well.

Similar results are presented in [9] where a significantly improved version of SVD was described. In addition to the classical interactions of latent factors, additional user perceptions, as well as joint attractiveness and unattractiveness of objects, were introduced in the model. Prediction accuracy was assessed by ranking metrics. The use of a complicated algorithm and other metrics also did not change the considered negative dependences. Scatter of values of the loss function of the model at different regularization constants remained within the same limits (more than 10 % of the minimum) and the shape of graphs of this dependence had the same characteristic features as in the studies discussed above [7, 8].

Thus, the results presented in [5, 7–9] confirm the importance of optimization of hyperparameters in the RS design and substantiate the relevance of the studies carried out by the authors. The quality of the rating predictions calculated by the models based on matrix factorization depends significantly on the chosen regularization constants and the number of factors. This, in turn, has a significant impact on the quality of the generated recommendations.

In the studies considered above [4, 5, 7–9], general methods used in other subject areas are used to optimize hyperparameters. Much attention is paid to Bayesian optimization which is actually a standard solution for a wide range of similar problems. For the latent-factor models, the effectiveness of this approach is limited for the following reasons. Bayesian optimization takes into account the visited points of the hyperparameter space creating a probabilistic model of the sought loss function on their basis [10]. The investigated function is considered as a «black box»: it is assumed that there can be any number of its extrema, and the cost of testing any different points is the same. This prerequisite force one to obtain all the necessary information empirically. At the same time, some of the necessary information is a priori for the models based on matrix factorization. This allows one to perform optimization more efficiently finding a set of hyperparameters of the same quality in a shorter time. Reducing wasted time without sacrificing quality is essential for both industrial and research purposes. This makes it possible to carry out the necessary numerical experiments and update the models faster, and, possibly, find more advantageous sets of hyperparameters in a limited time. The use of general-purpose methods (such as classical Bayesian optimization) does not enable the use of a priori information, and, accordingly, obtain the necessary speed of the RS development cycle and teaching the RS to new data.

The features of the models based on matrix factorization which make it possible to improve the optimization process were mentioned in [7–9] and are as follows. First, optimal values of the regularization constants and the number of factors correlate with each other rather weakly. In order to obtain more information about the influence of other parameters on the model error, it may be advantageous to fix values of a part of the studied parameters in a certain part of the space. Since a weak dependence does exist, the fixation does not have to be strict. It is preferable to temporarily narrow the boundaries of the studied values. Secondly, the speed of training and testing the model directly depends on the number of factors used. For common matrix factorization algorithms, the upper estimate of complexity has a linear dependence on the number of factors since the most frequently performed operation is the elementwise multiplication of vectors. Thus, the desired advantage can be obtained by simultaneously optimizing the number of factors and the regularization constants used (and, possibly, a small number of other hyperparameters). These features can be used either by modifying the known general-purpose optimization methods or developing special methods for the latent-factor models. The following examples are existing examples of the application of these two approaches.

Most modifications of general-purpose methods are associated with solving the problem of optimizing the number of factors. This problem is reduced to the problem of estimating the cost of training and testing the model which is quite common among a wide range of machine learning algorithms. Its theoretical solution consisting in changing the function of point extraction is proposed in [11]. The described modification includes modeling the cost of calculating the loss function. At each iteration of the search, not only the value of the objective function is memorized but also the amount of resources spent on its calculation. Due to this, when choosing

the next point, one can take into account not only the expected improvement but also the estimated cost making a choice with accounting for the available resources. For example, the implementation of such an idea in practice is presented in [12]. It describes the entropy-based point extraction function and the method used in conjunction with it to select the reference point. Due to their use, the optimization process takes into account the cost of testing and gives priority to more resource-intensive calculations at the beginning of optimization. The application of the presented method makes it possible to find a more advantageous set of hyperparameters in a limited time for a model taken as a «black box» but does not enable solving the problems of the current study. The total optimization time required to generate a model with the same prediction accuracy as classical Bayesian optimization is not necessarily reduced. Information about the testing cost is formed in this case as a result of a series of experiments. However, it is known a priori for latent-factorial models. The ability to analyze a much smaller number of factors to find a neighborhood of the optimal values of the regularization constants is also not used. In addition, the idea of prioritizing more resource-intensive computations contradicts theoretical conclusions about features of the considered loss functions.

The second approach using the features of models based on matrix factorization is based on the development of special optimization methods. A solution to the problem of finding values of the regularization constants is the main subject of such studies. The study [13] is an illustrative example. It proposes the «λOpt» method which consists in modifying the regularization coefficients during the model training. The application of this method makes it possible to obtain values of the constants that are acceptable in terms of efficiency at the end of the training stage. Duration of this stage increases but the total amount of the computational resources spent is reduced in comparison with the implementation of full optimization of hyperparameters by classical methods. Nevertheless, the sought number of factors remains undetermined which makes it necessary to perform its optimization separately. Since there is a weak correlation between the considered values of hyperparameters, the final efficiency of the regularization constants chosen at the first stage is not guaranteed.

Thus, the problem of accelerating the search for values of the regularization constants together with the number of factors cannot be solved even by using special well-known methods. The results of this analysis allow us to assert that it is relevant to conduct studies on the development of a new optimization method for the models based on matrix factorization. The main purpose of its application in scientific and applied fields consists in reducing the total time of optimizing the hyperparameters without losing the quality of the generated recommendations.

3. The aim and objectives of the study

The presented study objective is to develop a special method for optimizing the hyperparameters of matrix factorization algorithms used to train recommender models. The sought method should provide the ability to perform less resource-intensive computations at the beginning of optimization. Due to this, the total time of searching for optimal values of the regularization constants and the number of factors can be reduced without loss of quality compared to the classical Bayesian optimization.

To achieve the objective, it was necessary to solve the following tasks:

– identify the existing drawbacks of using classical Bayesian optimization and formulate requirements for the special optimization method being developed;

– theoretically describe the sought-for special method that meets the formed requirements by modifying one or more components of classical Bayesian optimization;

– programmatically implement the described method and repeatedly test it using latent-factor models trained on known datasets for the recommender systems having obtained statistically significant results.

4. Materials and methods used in the study

The problem of the recommender system was considered as the problem of predicting ratings. To solve it, the models trained by matrix factorization algorithms of the SVD family were used.

The problem of optimizing hyperparameters was posed as the problem of finding optimal values of the number of factors and constants of regularization for the algorithms under consideration. The classical method of Bayesian optimization was used as a basic one. The special optimization method was developed on the basis of the basic one. It was proposed to analyze its limitations and modify it to provide the desired changes in the optimization process.

It was proposed to evaluate the effectiveness of the developed method by conducting experiments on known datasets for recommender systems. The experiments should compare the total optimization time and quality of rating the predictions generated by the final models. The results obtained using classical Bayesian optimization and the developed special method were subject to comparison. To achieve the set objective, the time spent on optimization should be reduced without losing the quality of rating predictions.

Further in the current section, a formal description of the mathematical apparatus used is given and the experimenting procedure is described in detail.

4. 1. The problem of the recommender system and its solution using the matrix factorization algorithms

A formal statement of the problem for the recommender system in the considered variant looks as follows. A set of users U and a set of objects I are given. As a result of interactions, some users have rated some objects. This has made it possible to form the set $K=\{r_{ui}\in\mathbb{R},|u\in U\wedge i\in I\}$, where r_{ui} is the rating assigned by the user u to the object i. The task of the RS consists in calculating the set of unknown ratings $\{\hat{r}_{ui}\in\mathbb{R}|u\in U\wedge i\in I\wedge r_{ui}\notin K\}$. More generally, we can say that the recommender model calculates an $\hat{\mathbf{R}}\in\mathbb{R}^{U\times I}$ matrix containing predictions of $\hat{r}_{ui}$ ratings for all possible pairs of users and objects.

The idea of using matrix factorization algorithms to implement the collaborative filtering approach is as follows. The sought rating prediction matrix can be roughly represented as a product of two others: $\hat{\mathbf{R}}\approx\mathbf{PQ}$, $\mathbf{P}\in\mathbb{R}^{U\times f}$, $\mathbf{Q}\in\mathbb{R}^{f\times I}$. Then the matrix rows $\mathbf{P}$ are nothing but vectors of latent factors $\mathbf{p}_u\in\mathbb{R}^f$ of users and columns $\mathbf{Q}$ are vectors of latent factors of objects $\mathbf{q}_i\in\mathbb{R}^f$, respectively. Individual rating prediction is calculated as follows:

$$\hat{r}_{ui}=\mathbf{p}_u\mathbf{q}_i^T. \quad (1)$$

It is also important to remember that the rating is influenced not only by interaction between users and objects but also by their individual characteristics. The latter ones are capable, for example, of making permanent adjustments to the assessments given by one of the users. Biases are used to account for these effects. The b_{ui} bias of the r_{ui} rating can be given as:

$$b_{ui} = \mu_K + b_u + b_i, \quad (2)$$

where μ_K is the average rating in the entire dataset and b_u and b_i are the observed biases of ratings of an individual user and an object, respectively.

Rating prediction taking into account the latent factors and biases is used in the well-known SVD matrix factorization algorithm:

$$\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}_i^T + \mu_K + b_u + b_i. \quad (3)$$

The model training consists in finding the values $\mathbf{p}_u$, q_i, b_u, b_i for all users $u \in U$ and objects $i \in I$. The regularized squared error is minimized for training:

$$\sum_{r_{ui} \in K} \left[\begin{array}{l} \left(r_{ui} - \mathbf{p}_u \mathbf{q}_i^T - \mu_K - b_u - b_i\right)^2 + \\ +\lambda\left(b_i^2 + b_u^2 + \|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2\right) \end{array} \right], \quad (4)$$

where λ is a regularization constant. The loss function can be minimized using standard techniques such as SGD or ALS.

The SVD++ algorithm improves the accuracy of SVD predictions by the involvement of additional object factors which make it possible to better model the implicit preferences of an individual user. For each object $i \in I$, another vector of latent factors $\mathbf{y}_i \in \mathbb{R}^f$ is specified. The sets $I_u = \{i \in I | r_{ui} \in K\}$ are also defined. They contain an enumeration of objects for each user u to which he has given ratings. The prediction is defined as:

$$\hat{r}_{ui} = \left(\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} \mathbf{y}_j\right) \mathbf{q}_i^T + \mu_K + b_u + b_i. \quad (5)$$

When training the SVD++ model, a search for the $\mathbf{p}_u$, $\mathbf{q}_i$, $\mathbf{y}_i$, b_u, b_i values is performed. The regularized squared error is calculated as:

$$\sum_{r_{ui} \in K} \left[\begin{array}{l} \left(r_{ui} - \left(\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} \mathbf{y}_j\right) \mathbf{q}_i^T - \mu_K - b_u - b_i\right)^2 + \\ +\lambda\left(b_i^2 + b_u^2 + \|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + \|\mathbf{y}_i\|^2\right) \end{array} \right], \quad (6)$$

and is minimized by SGD or ALS similar to the SVD (4) error.

Further modifications of the SVD family algorithms are constructed in a similar way. For example, the authors of [14] modify the error minimization process (6) including gradient perturbations to reduce the likelihood of disclosing the personal data of users. The study [15] uses additional vectors of latent factors of users in conjunction with a matrix of their mutual influence on each other. In the previously considered study [9], the improved SVD algorithm also assumes calculation of ratings and errors using similar formulas. The results presented in the mentioned studies of models based on matrix factorization [5, 7–9, 13–15] demonstrate the following features:

– the error in predicting the ratings of the trained model strongly depends on the hyperparameters f and λ used (as well as, possibly, others: learning rate constant, individual constants of regularization of different parameters, etc.);

– hyperparameters f and λ are weakly dependent on each other;

– the computational cost of training the model depends on the $|U|$, $|I|$ and f values and the dependence on f is no less than linear [16].

4. 2. Optimizing the hyperparameters

To construct a model that predicts ratings with the smallest error, it becomes necessary to optimize the hyperparameters. Formally, the problem of optimizing the hyperparameters looks like this. Let the model have N hyperparameters. Let us denote the domain of definition of the n-th hyperparameter θ_n as $\overline{\theta_n}$, and the entire space of hyperparameter configurations as $\Theta = \overline{\theta_1} \times \ldots \times \overline{\theta_N}$. In the simplest case, the dataset K is divided into training and validation parts K_{train} and K_{valid}, respectively. The model is trained with a set of hyperparameters $\theta \in \Theta$ on the K_{train} dataset. The value of the loss function shown by such a model on the K_{valid} dataset is denoted as $L(\theta)$. The challenge consists in finding the optimal set of hyperparameters θ_{best}:

$$\theta_{best} = \arg\min_{\theta \in \Theta} E\left[L(\theta)\right]. \quad (7)$$

The present-day solution to the problem is to use Bayesian optimization. It includes two main components: a surrogate model for the loss function L and a point extraction function. The surrogate model based on experimental observations $(\theta_t, L(\theta_t))$ constructs a probabilistic approximation of the objective function. With its help, predictions of the mathematical expectation $\mu(\theta) = E[L(\theta)]$ and the variance $\sigma^2(\theta) = \sigma^2[L(\theta)]$ of the loss function values become available at any point of the search space. The first few points for testing are selected at random. Points at each of the following iterations are selected using the point extraction function the extrema of which correspond to the sets of hyperparameters the most attractive for testing.

In the presented study, UCB (Upper Confidence Bound) is chosen as the point extraction function:

$$\mathrm{UCB}(\theta) = \mu(\theta) + \kappa\sigma^2(\theta), \quad (8)$$

where κ is a constant used for regulation of priority between research and operation.

The Gaussian process is often used as a surrogate model for the sought function. Also, examples of effective use of random forests in this capacity are known as well as TPE (Tree-Structured Parzen Estimator). Their main advantage over Gaussian processes is the smaller algorithmic complexity of calculation. However, for the hyperparameter optimization of the latent-factor model considered in this article, the complexity of prediction calculation is insignificant: the number of points is small (several tens) since the size of the hyperparameter configuration space is also small. The use of the Gaussian process makes it possible to fairly accurately simulate the loss function taking into account a small number of known values. Analysis of practical implementations of the Bayesian optimization shows that such a choice of a surrogate model is becoming a fairly frequent occurrence, and often, the only available solution.

The Gaussian process is determined by the a priori functions of calculation of the mathematical expectation $\mu_{prior}(x)$ and the covariance function $k(x,x')$. If $\mu_{prior}(x) = 0$, then the flow of the stochastic process is completely determined by the covariance function. The Matern 5/2 function is one of

the most commonly applied covariance functions for the Gaussian processes used in the Bayesian optimization:

$$k(x,x') = \alpha^2\left(1+\frac{\sqrt{5}d}{l}+\frac{5d^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}d}{l}\right), \quad (9)$$

where d is the Euclidean distance between x and x', α is the amplitude, l is the scale.

By definition of the Gaussian process, observations $x_t=L(\theta_t)$ are considered to be subject to a multivariate normal distribution. Therefore, the conditional probability of obtaining new $L^*=L(\theta^*)$ values at not yet visited points $\theta^* \in \Theta^*$ with already obtained $L_{obs}=\{L(\theta_1, ...)\}$ at points $\Theta_{obs}=\{\theta_1, ...\}$ is also normally distributed:

$$\Pr\left(L^* \mid L_{obs}\right) \sim \mathcal{N}\left(\mu\left(\theta^*\right),\sigma^2\left(\theta^*\right)\right), \quad (10)$$

$$\mu\left(\theta^*\right) = \mathbf{K}\left[\theta^*,\Theta_{obs}\right]\mathbf{K}\left[\Theta_{obs},\Theta_{obs}\right]^+ L_{obs}, \quad (11)$$

$$\sigma^2\left(\theta^*\right) = \mathbf{K}\left[\theta^*,\theta^*\right] - \\ -\mathbf{K}\left[\theta^*,\Theta_{obs}\right]\mathbf{K}\left[\Theta_{obs},\Theta_{obs}\right]^+ \mathbf{K}\left[\Theta_{obs},\theta^*\right], \quad (12)$$

where $\mathbf{K}[X,X']$ are covariance matrices in which each element with index (i,j) is equal to $k(x_i,x'_j)$; $\mathbf{K}^+$ is a pseudoinverse matrix for $\mathbf{K}$. Formulas (11), (12) are applied to the calculation of the mathematical expectation and variance of the loss function at any not yet investigated point θ^* of the space of configuration of Θ hyperparameters. When performing the Bayesian optimization, formulas (11), (12) are used by the point extraction function.

4. 3. The procedure used for conducting the experiments

The purpose of the experiments consisted in evaluating the effectiveness of the developed method of optimizing the hyperparameters of recommender models based on matrix factorization.

The Movielens datasets are widely used when conducting experiments for testing the recommender systems [6]. Each set consists of users, movies and ratings (ranging from 1 to 5). The problem consisted in predicting the ratings that users will give to movies that have not yet been watched. Table 1 shows the characteristics of the datasets used.

Table 1

Datasets used in the study

Name	Number of users	Number of objects	Number of ratings
Movielens 100k	943	1,682	100,000
Movielens 1M	6,040	3,706	1,000,209
Movielens 10M	69,878	10,677	10,000,054

The experimental procedure included the following stages. Each dataset was randomly divided into training and validation set in a ratio of 80 % to 20 %, respectively. SVD and SVD++ models were trained in the training part. The training rate was fixed at 0.05, the number of learning epochs was 20. Fixed regularization constant $\lambda_b=0.02$ was used for parameters b_i and b_u ((4), (6)). Values of hyperparameters f and λ were in the process of optimization. The hyperparameter f was discretely determined in the interval from 1 to 100, the hyperparameter λ was continuously determined in the interval from 0.01 to 0.1.

RMSE was used as an error metric:

$$L(\theta) = \sqrt{\frac{\sum_{r_{ui}\in K_{valid}}\left(r_{ui}-\hat{r}_{ui}\right)^2}{|K_{valid}|}}, \quad (13)$$

where $\hat{r}_{ui}$ is the prediction made by the model trained with a θ hyperparameter set on the K_{train} training set.

Classical Bayesian optimization was used as a reference optimization method. It is performed with UCB (8) as a point extraction function ($\kappa=2.576$) and the Gaussian process as a surrogate model (with Matern 5/2 core (9), $\alpha=1$, $l=1$). The effectiveness of the reference and developed methods should be compared in different experiments.

The number of Bayesian optimization iterations was limited by 10. The number of preselected random points was 1. All hyperparameters were normalized in the interval (0;1]. Since the optimization process is sufficiently dependent on random factors (initial sampling of points and sampling during the calculation of the surrogate model values), the experiment with the same parameters should be performed several times. The number of repetitions for Movielens 1M and 10M was 10. For Movielens 100k, it was increased to 100 because of its small size.

In the course of each experiment, values of the model error were measured relative to the current iterations and the time elapsed since the optimization start.

The experiments were performed on an Intel Core i5-4690 processor using 16 GB of DDR3 RAM at 1600 MHz.

5. The results obtained in the development of a method for optimizing hyperparameters of models based on matrix factorization

5. 1. Limitations of Bayesian optimization and requirements to the developed method

The value of the regularization constant λ (as well as the learning rate, etc.) has a rather weak dependence on the f values used. The neighborhood of the sought λ value in the vicinity of the optimal value of f can most likely be determined by testing the model with fewer factors. If this number of factors can be significantly less than the optimal one, then the time of model training can be greatly reduced. In the classical application of Bayesian optimization, no prior information about the shape of the predicted loss function is used. All data available before the optimization start are determined by testing the model at several randomly selected points of the search space. Consequently, when applying the classical method to simultaneous search for optimal values of the f and λ hyperparameters of the matrix factorization algorithm, the traversal order may turn out to be ineffective. As a result, the total search time becomes much longer than the minimum required. This is the limitation of Bayesian optimization which was proposed to be overcome in the current study.

The main requirements for the developed solution are as follows. Since the information used is a priori information, the priority of points in the area of smaller f must exist from the first iteration of the search (immediately following the test of random points). Since the sought set of hyperparameters still remains in the region of large f, the last iterations of the search should be performed without artificially prioritizing any values.

Obviously, the simplest solution that allows the use of a priori information in Bayesian optimization is a preliminary

study of different λ values with a fixed small f. However, this solution has significant drawbacks. The course of the Bayesian optimization process strongly depends on the result of testing at the initial, randomly selected points. In the standard version of the method, there is a possibility that such points will immediately fall into the vicinity of the global minimum of the loss function, however, when the values are fixed manually, it is lost. In addition, fixing the f value does not allow obtaining additional information about the shape of the function L with respect to the number of factors. After several initial iterations, only one value of L along the f axis will be available which degrades the accuracy of the constructed probabilistic model. Therefore, the method being developed does not have to do a strict fixation of f.

Moreover, the previously mentioned studies [11, 12] devoted to the solution of the related problem of the variable cost of model testing reveal an additional problem. If the initial, randomly selected points correspond to too large values of the loss function at a small number of factors, the optimization process is likely to be performed in a more resource-intensive region. At the same time, the study with lower f may remain effective in some cases. If the initial random points fall into the target region of the search space, it may be advantageous to prioritize empirical information and not do research with lower f. Therefore, the developed method should take into account the fact that the benefit from the study of the initial random points is variable.

5. 2. Modification of the Gaussian process core

To solve the listed problems and eliminate the described limitations, the following was proposed. At initial iterations of the optimization algorithm, it is possible to slightly increase the variance $\sigma^2(\theta^*)$ in the not studied region of the search space corresponding to a smaller number of factors. Using this modification allows one to achieve the following effects:

– the above problem is solved: all hyperparameters, except for f, are initially investigated at lower f which makes it possible to determine the neighborhood of their desired values in less time;

– during the first iterations, information about the form of the loss function L relative to the axis of the number of factors is also extracted;

– during subsequent iterations, the probabilistic prediction of L is not distorted in any way;

– if randomly selected starting points immediately fall into the target area of the search space, the optimization algorithm may not investigate ineffective values in a less resource-intensive area since the mathematical expectation L does not change;

– at the same time, when randomly selected initial points correspond to too large values of the loss function, there is a possibility that the study will be continued in a less resource-intensive area since the $\sigma^2(\theta^*)$ is increased;

– variance of already visited values remains zero, and variance in their vicinity changes in proportion to the original which excludes the possibility of an unnecessarily thorough study of such neighborhoods.

To implement the described idea, it is enough to change the covariance function used as a core. The matrix $\mathbf{K}[\theta,\theta^*]$ is used only when calculating the variance of values (12) but not when calculating their mathematical expectation (11). In the simplest case, such a matrix measures 1×1 and contains covariance of the unvisited point with itself (i.e., the variance). If the variance is calculated simultaneously for a large number of unvisited points, a diagonal matrix is used. Thus, if the modified covariance function $k_{mod}(\theta,\theta')$ for the points $\theta\in\Theta^*\wedge\theta=\theta'$ will return values increased in the area of smaller f, the goal will be achieved. It is also necessary to take into account that $k_{mod}(\theta,\theta')$ must be non-negatively definite:

$$\forall\alpha\in\mathbb{R}^m:\sum_{i,j=1}^{m}\alpha_i\alpha_j k\left(\theta_i,\theta_j\right)\geq 0. \tag{14}$$

It was proposed to modify the covariance function as follows:

$$k_{\text{mod}}\left(\theta,\theta'\right)=\begin{cases}k\left(\theta,\theta'\right)\omega\left(\theta\right) & \text{for} \quad \theta\in\Theta^*\wedge\theta=\theta',\\ k\left(\theta,\theta'\right) & \text{otherwise}\end{cases} \tag{15}$$

$$\omega\left(\theta\right)=\frac{1}{1+\exp\left(\theta_f\right)}\cdot\frac{\max\left(c_{decay}-n_{iter},0\right)}{c_{decay}}\cdot c_{scale}+1, \tag{16}$$

where $\mathbf{k(\theta,\theta')}$ is any valid covariance function; n_{iter} is the number of the current iteration (starting from zero); $c_{decay}>0$ is the constant defining the number of iterations during which the modification is active; $c_{scale}>0$ is scale; θ_f is the value of hyperparameter f normalized from 0 to 1. The specified function $\omega(\theta)$ is a mirrored sigmoid. It is scaled in height, decays linearly with increasing the iteration number, and is offset by 1 along the vertical axis (so as not to distort the covariance values for large f). If necessary, additional parameters can be added that horizontally stretch or compress the sigmoid or shift its center. Since $k(\theta,\theta')$ is nonnegatively definite, then $\omega(\theta)>0$ for any θ, and $k_{mod}(\theta,\theta')$ changes only at $\theta=\theta'$, the condition (14) is obviously satisfied.

Fig. 1 shows an example of predictions of the Gaussian process with a modified Matern 5/2 covariance function, $c_{decay}=3$, $c_{scale}=1e\text{-}3$. The filled area reflects the variance, the curved line is the mathematical expectation. Rendering was done for four n_{iter} values. With $n_{iter}=3$, the graph corresponds to a Gaussian process with an unmodified covariance function.

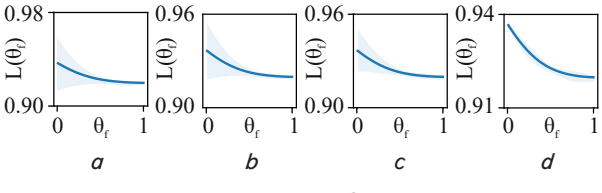

Fig. 1. Values predicted by the Gaussian process with a modified core for different n_{iter}: a – $n_{iter}=0$; b – $n_{iter}=1$; c – $n_{iter}=2$; d – $n_{iter}=3$

The graphs in Fig. 1 demonstrate that the unmodified process gives priority to exploring the right side of the space while the variance predicts a possible minimum on the left side in the version with modification with n_{iter} values equal to 0 and 1. In this case, priority may be given to it (depending on the used point extraction function). For other random initial points, the shape of the mathematical expectation curve may be different resulting in that the left-hand side may not be studied at all or studied more strongly.

5. 3. Experimental tests

To compare the reference optimization method with the developed one, it is sufficient to use different cores of the

Gaussian process. Three different covariance functions (c.f.) were used in experiments to this end:

– c.f. Matern 5/2 (9), $\alpha=1$, $l=1$;

– modified (15), (16) c.f. Matern 5/2, $\alpha=1$, $l=1$, c_{scale}=1e-3, $c_{decay}=3$;

– modified (15), (16) c.f. Matern 5/2, $\alpha=1$, $l=1$, c_{scale}=1e-3, $c_{decay}=4$.

The source code for the experiments is publicly available on Github [17]. To enable a large number of experiments, fast implementations of SVD and SVD++ based on [18] were used. Implementation of [19] was used for the Bayesian optimization.

Table 2 shows the results of estimating the time spent on the implementation of the entire optimization process. Table 3 shows the RMSE values of the models after optimization. Values in both tables are presented with a 95 % confidence interval. «Original c.f.» means covariance function Matern 5/2, «Modified c.f.» means modified functions with different values of the c_{decay} parameter.

For clarity, Fig. 2 shows visualizations of the error reduction processes (in time and iterations) in some interesting cases.

Without modification, the average time to run the optimization was approximately 13–20 seconds for the Movielens 100k dataset, 105–115 seconds for the Movielens 1M. The interval is much higher on the Movielens 10M set: 963.03 seconds for SVD and 1634.60 seconds for SVD++. On the same dataset, the total execution time of all iterations was consistently lower when the modification was applied. For the SVD model, the reduction was about 5 % (or 45 seconds), and for the SVD++ model, the reduction was about 16 % (or 263 seconds). There was also a decrease in the Movielens 1M set but it was unstable and comparatively lower (from 0.8 % to 5 % in different experiments). For Movielens 100k, it was practically not observed, in some cases the total time turned out to be even higher (within one second).

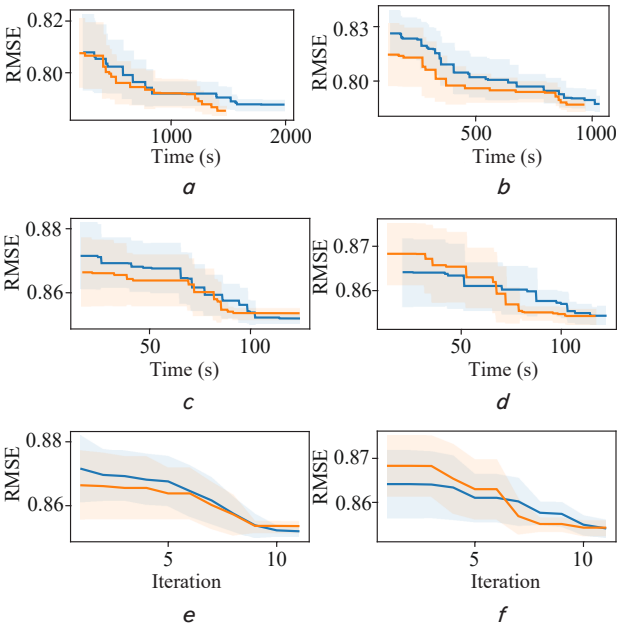

Fig. 2. Error reduction process (95 % confidence interval):
a – Movielens 10M, SVD++, $c_{decay}=3$;
b – Movielens 10M, SVD, $c_{decay}=4$;
c – Movielens 1M, SVD, $c_{decay}=3$;
d – Movielens 1M, SVD++, $c_{decay}=4$;
e – Movielens 1M, SVD, $c_{decay}=3$ (by iterations);
f – Movielens 1M, SVD++, $c_{decay}=4$ (by iterations);
original core; modified core

It should be especially noted that the RMSE values of the models measured at the end of optimization with different cores, practically coincided with each other. The observed difference in most cases was two orders of magnitude less than the direct decrease in error in the optimization time. The difference between the model errors on the Movielens 10M set was less than the others and practically nonexistent.

The effect of the c_{decay} parameter on the final execution time was ambiguous. For SVD models, it turned out to be slightly lower with $c_{decay}=4$ in all cases and for SVD++ models, it was more often with $c_{decay}=3$. However, the observed difference was negligible. The more important difference consists in a spread of the error values within the confidence interval. At $c_{decay}=3$, the absolute width of the confidence interval along the RMSE axis decreased more strongly in all cases with increasing time and at the final iteration, it was less than at $c_{decay}=4$.

Mathematical expectation and variance of the error at the first iteration of the search (for $n_{iter}=0$) are determined by randomly selected points of the hyperparameter configuration space. Based on the results of all experiments, the following data were obtained regarding the further course of the optimization process. In all cases in which optimization with the use of modification was performed with the worst initial choice, its error reduction graph intersected the similar optimization graph without modifications

Table 2

Time spent on hyperparameter optimization (seconds, 95 % confidence interval)

Model	Dataset	Original c.f.	Modified c.f., $c_{decay}=3$	Modified c.f., $c_{decay}=4$
SVD	Movielens 100k	13.03±0.16	12.72±0.10	12.58±0.15
SVD	Movielens 1M	114.06±2.92	108.20±3.46	105.36±2.68
SVD	Movielens 10M	963.03±25.94	918.44±24.04	912.82±16.95
SVD++	Movielens 100k	19.88±0.20	20.04±0.24	21.29±0.53
SVD++	Movielens 1M	115.38±4.37	114.45±3.90	113.54±4.33
SVD++	Movielens 10M	1634.60±94.93	1371.33±39.24	1378.82±44.12

Table 3

RMSE of models after optimization (95 % confidence interval)

Model	Dataset	Original c.f.	Modified c.f., $c_{decay}=3$	Modified c.f., $c_{decay}=4$
SVD	Movielens 100k	0.9459±0.0005	0.9462±0.0004	0.9462±0.0005
SVD	Movielens 1M	0.9608±0.0021	0.9589±0.0093	0.9621±0.0027
SVD	Movielens 10M	0.8455±0.0003	0.8454±0.0003	0.8455±0.0003
SVD++	Movielens 100k	0.9448±0.0008	0.9452±0.0005	0.9452±0.0006
SVD++	Movielens 1M	0.9434±0.0091	0.9462±0.0031	0.9491±0.0022
SVD++	Movielens 10M	0.8437±0.0001	0.8436±0.0001	0.8435±0.0004

in the region of iterations 6–9. If the initial choice of points approximately coincided, then the optimization graph with modification was higher than the similar one, as a rule, at first iterations but it crossed it again at subsequent iterations. If the initial choice of points for optimization with modification was better, then the error reduction was observed earlier than after the expiration of c_{decay} iterations. At the same time, the final result is no worse than when applying optimization without modifications. It is also extremely important that due to faster first iterations, error reduction is faster when using the modification than without it. For the same reason, the intersection of lines is in most cases more to the left on the time chart than on the iteration chart. If we limit the available time and compare the errors within about 70 % of the total execution time, the following can be observed. The error of the model with the hyperparameters selected at that time with optimization with modification was in all cases lower than at optimization without modification.

6. Discussion of the results obtained in the development of a method for optimizing hyperparameters of the models based on matrix factorization

For a better understanding of features of optimization of hyperparameters of the considered models, we can consider visualization of the observed error obtained in previous studies [8] in the space of the hyperparameter configuration. The number of factors f varied from 10 to 100 with a step of 10 and the regularization constant λ varied from 0.01 to 0.1 with a step of 0.01. Fig. 3 shows four graphs showing features that are important for explaining the results.

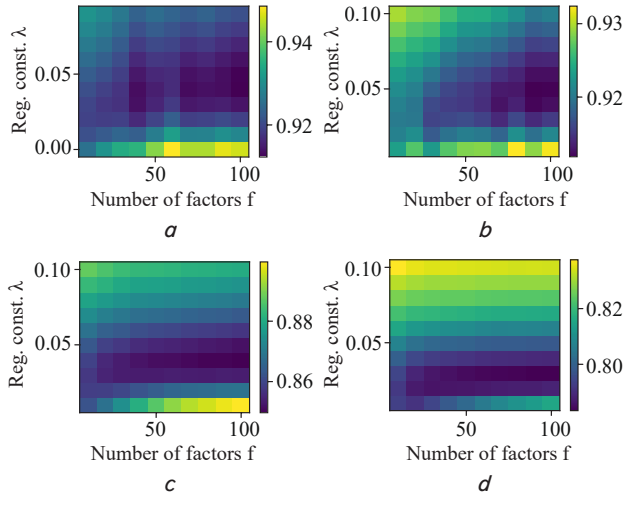

Fig. 3. Error values over the entire space of the hyperparameter configuration: *a* – Movielens 100k, SVD; *b* – Movielens 100k, SVD++; *c* – Movielens 1M, SVD++; *d* – Movielens 10M, SVD++

The dependence of optimal λ value on optimal f value is indeed rather weak. The target neighborhood λ at a smaller number of factors practically coincides with that at a target number of factors. In this case, the function of dependence of $L(\theta)$ on λ at a fixed f has only a global minimum but does not have pronounced local minima. These facts explain the increased rate of error reduction when the c_{decay} iterations are reached: the study continues with a larger f but already in the desired neighborhood λ. With an increase in the size of the dataset and complication of the model, the neighborhood of the sought λ noticeably narrows, and the difference between adjacent values increases. This could serve as an additional explanation for why the optimization time was more reduced for a more complex model on larger datasets. The results for Movielens 100k are also explained by peculiarities of the used implementation. The training time of the SVD model on this set is within a few seconds and is more influenced by disk operations and other overheads than by the computation time itself.

Sets of the search space points visited during optimization are of particular interest. The analysis showed that the sought effects from the use of the modified core were observed in all experiments. Sets of the points selected in the first four search iterations when optimizing the models on the Movielens 10M dataset are the most representative. They are shown in Fig. 4.

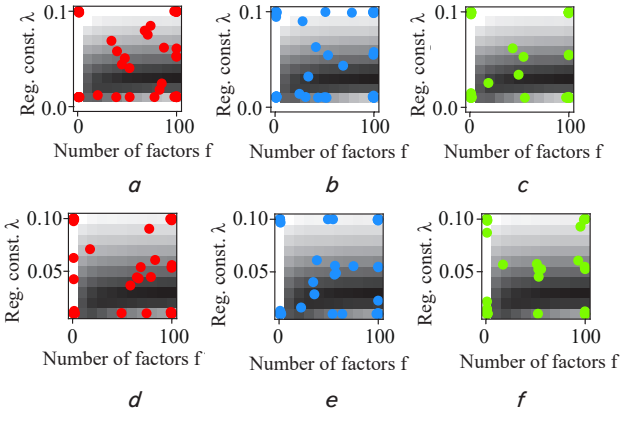

Fig. 4. Points visited in the first four iterations during optimization of the models trained on the Movielens 10M dataset: *a* – SVD, original core; *b* – SVD, modified core, c_{decay}=3; *c* – SVD, modified core, c_{decay}=4; *d* – SVD++, original core; *e* – SVD++, modified core, c_{decay}=3; *f* – SVD++, modified core, c_{decay}=4

The graphs in Fig. 4 clearly show how the selection of points is shifted to the left side of the space corresponding to smaller f when using modification. The points are distributed much more evenly without modification. Obviously, the offset is stronger at c_{decay}=4. The only exception where the bias is weak is the SVD++ graph on the Movielens 10M dataset with c_{decay}=4. This is due to the good choice of starting points. In this case, the points on the right side of the graphs in Fig. 4 are present in all visualizations. This confirms the fact that the proposed optimization process does not spend extra time on testing models with smaller f with a sufficiently good choice of random starting points. It is also worth noting that all corner points of the space were checked in all cases. This feature arises because of the use of Gaussian processes that predict increased variance at the boundaries of the area under study.

The results of the experiments performed demonstrate that the expected effects from the use of the proposed modification were successfully achieved. Regardless of the choice of starting points, the Bayesian optimization with modification has turned out to be either more profitable to use than without it, or, in some cases, of the same efficiency. The changes were less noticeable for smaller datasets and a simpler model.

The quality of the sets of hyperparameters found in less time practically coincided with the quality of the sets obtained by the classical method.

c_{decay}=3 and c_{scale}=1e-3 can be considered optimal parameters of the function (16) (in this case, c_{scale}, obviously, changes with other rating scales). Parameters of the rest of the optimization components do not differ from those when using the original core.

The simplicity of modification implementation is its separate advantage. Since the calculation of the diagonal variance matrix at unvisited points is often taken out in a separate procedure for performance reasons, it is enough to redefine it.

The following can be highlighted as the factors limiting the applicability of the study results.

Firstly, the specific area of the most effective application of the proposed optimization method remains an open question. There are various modifications of the SVD and SVD++ algorithms. In addition, other datasets may have meta-characteristics different from Movielens: by the ratio of the number of users and objects, the sparseness of the rating matrix, variety of interaction patterns, etc. The application of the described method makes sense if the loss function of the models trained on them has a similar form in a similar hyperparameter configuration space.

As regards modifications of the SVD and SVD++ algorithms, it can be assumed that the considered optimization method will remain effective if the minimized loss function is similar to (4) and (6). Since the complexity of calculation is always growing with the number of factors and the regularization method does not change, it can be expected that the key properties of the loss function will not be changed and the initial study of the regularization constants with smaller f remains justified. It is not possible to give an answer regarding other data sets without experimental testing. However, the results presented in the current study show that it is possible to check the optimization efficiency for a large dataset by using only a part of it. It should retain all characteristic features of the initial set and the model training time on the selected part should be at least several tens (or hundreds) of seconds. In this case, when performing at least $2c_{decay}$ optimization iterations, it will be possible to assess how applicable the method is for the dataset under consideration.

Second, the study was conducted using exclusively the Matern 5/2 function as the original core, and UCB as the point extraction function. Using other covariance functions as a core is likely to lead to the same results. This assumption is substantiated by the fact that the described modification introduces the same changes in values of any valid covariance function. The use of other point extraction functions will change the contribution made by the variance of the predictions at a given point to the probability of its selection. Since the shape of the loss function does not change, the initial investigation of model errors with fewer factor numbers remains beneficial. In both cases, differences can only be in the parameters of the function (16). Nevertheless, experimental verification is necessary for a definite answer.

Thus, the potential development of the study presented in this article may be as follows. Additional experimental testing can be performed, which will make it possible to clarify limits of applicability of the described method, as well as optimal values of the parameters of function (16) in individual cases. It may also be of interest to test the proposed method for solving the optimization problem for a wider set of hyperparameters. The main obstacle to doing additional experiments is the need to perform a lot of resource-intensive computations.

In addition, the results can be improved if the modification (15) is performed by a function other than the sigmoid (16). In a more general case, the possibility of developing similar modifications for surrogate models other than Gaussian processes is of interest. In these cases, the theoretical component of the study should be significantly expanded. The difficulty lies in the fact that other models may not provide similar ways of changing the priorities of selecting points for testing.

7. Conclusions

1. An analysis of the limitations of using classical Bayesian optimization to optimize model hyperparameters based on matrix factorization was made. The main disadvantage limiting its effectiveness in solving this problem consists in the use of only a posteriori information about the form of the loss function and the cost of its testing. For matrix factorization algorithms, it is possible to speed up the search for hyperparameters without losing quality by examining values of the regularization constant with fewer numbers of factors at first iterations. Main requirements to the developed special method were identified. They concern the prioritization of a smaller number of factors only at first iterations, the lack of strict fixation of values, and also the consideration of an initial random selection of points.

2. To ensure the sought changes in the process of optimization, a modification of the Gaussian process core was proposed. It is used as a surrogate model for the loss function in Bayesian optimization. The sought modification at first iterations insignificantly increases the variance of the values predicted by the surrogate model in the region of the search space corresponding to a smaller number of factors f. It was theoretically substantiated that this behavior would lead to the desired investigation of the values of all hyperparameters, except for f, in the vicinity of smaller f. At the same time, all requirements specified for the special method were fulfilled. The proposed covariance function was mathematically described. Its correctness and applicability as the Gaussian process core have been confirmed. The changes introduced by the application of the modified core, depending on the current iteration, were graphically demonstrated.

3. The developed method of optimizing the hyperparameters was implemented in software. Its effectiveness was tested on common datasets. The source code of the modified core and the experiments performed have been published. It has been experimentally confirmed that all the specified requirements for a special method are met. A statically significant reduction in overall optimization time of up to 16 % has been demonstrated on larger datasets. In the worst case, the timing was the same as for the original core. The expected error in rating the predictions practically coincided in the models with their hyperparameters found using the classical Bayesian optimization method and the developed method. Consequently, the quality of optimization was not reduced while reducing the time spent which confirms the achievement of the study objective. It has also been demonstrated that the use of core modification can be beneficial in a limited time environment. The results show that the larger the dataset and the more complex the model, the more effective the method is.

References

1. Ricci, F., Rokach, L., Shapira, B., Kantor, P. B. (Eds.) (2011). Recommender Systems Handbook. Springer, 842. doi: https://doi.org/10.1007/978-0-387-85820-3
2. Rendle, S., Zhang, L., Koren, Y. (2019). On the Difficulty of Evaluating Baselines: A Study on Recommender Systems. arXiv.org. Available at: https://arxiv.org/abs/1905.01395
3. Dacrema, M. F., Cremonesi, P., Jannach, D. (2019). Are we really making much progress? A worrying analysis of recent neural recommendation approaches. Proceedings of the 13th ACM Conference on Recommender Systems. doi: https://doi.org/10.1145/3298689.3347058
4. Aggarwal, C. C. (2016). Recommender Systems. Springer, 498. doi: https://doi.org/10.1007/978-3-319-29659-3
5. Fathan, G., Bharata Adji, T., Ferdiana, R. (2018). Impact of Matrix Factorization and Regularization Hyperparameter on a Recommender System for Movies. Proceeding of the Electrical Engineering Computer Science and Informatics, 5 (5), 113–116. doi: https://doi.org/10.11591/eecsi.v5i5.1685
6. Harper, F. M., Konstan, J. A. (2016). The MovieLens Datasets. ACM Transactions on Interactive Intelligent Systems, 5 (4), 1–19. doi: https://doi.org/10.1145/2827872
7. Galuzzi, B. G., Giordani, I., Candelieri, A., Perego, R., Archetti, F. (2020). Hyperparameter optimization for recommender systems through Bayesian optimization. Computational Management Science, 17 (4), 495–515. doi: https://doi.org/10.1007/s10287-020-00376-3
8. Nechaev, A. A., Meltsov, V. Yu. (2021). Investigating the hyperparameter configuration space of matrix factorization recommendation models. Nauchno-tekhnicheskiy vestnik Povolzh'ya, 5, 96–100. Available at: https://elibrary.ru/item.asp?id=46124660
9. Tran, T., Lee, K., Liao, Y., Lee, D. (2018). Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. doi: https://doi.org/10.1145/3269206.3271730
10. Feurer, M., Hutter, F. (2019). Hyperparameter Optimization. The Springer Series on Challenges in Machine Learning, 3–33. doi: https://doi.org/10.1007/978-3-030-05318-5_1
11. Snoek, J., Larochelle, H., Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. arXiv.org. Available at: https://arxiv.org/abs/1206.2944
12. McLeod, M., Osborne, M. A., Roberts, S. J. (2018). Practical Bayesian Optimization for Variable Cost Objectives. arXiv.org. Available at: https://arxiv.org/abs/1703.04335
13. Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J.-G., Wang, Y. (2019). λOpt. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. doi: https://doi.org/10.1145/3292500.3330880
14. Xian, Z., Li, Q., Li, G., Li, L. (2017). New Collaborative Filtering Algorithms Based on SVD++ and Differential Privacy. Mathematical Problems in Engineering, 2017, 1–14. doi: https://doi.org/10.1155/2017/1975719
15. Shi, W., Wang, L., Qin, J. (2020). User Embedding for Rating Prediction in SVD++-Based Collaborative Filtering. Symmetry, 12 (1), 121. doi: https://doi.org/10.3390/sym12010121
16. Cline, A. K., Dhillon, I. S. (2006). Computation of the Singular Value Decomposition. Handbook of Linear Algebra, 45-1–45-13. doi: https://doi.org/10.1201/9781420010572-45
17. Nechaev, A. (2021). Speeding up Bayesian Optimization of Matrix Factorization Recommender Models Hyperparameters. GitHub. Available at: https://github.com/dapqa/speeding-up-bo-for-cf-public
18. Zhao, E. (2018). Optimized-for-speed Eigen implementations of SVD, SVD++ and TimeSVD++ algorithms. Available at: https://github.com/dapqa/svdistic
19. Nogueira, F. (2014). Bayesian Optimization: Open source constrained global optimization tool for Python. Available at: https://github.com/fmfn/BayesianOptimization