Alleviating the Cold-Start Problem in Recommender Systems Using Error-Based Learning

1st Ricardo Laanen
Erasmus University Rotterdam
P.O. Box 1738
NL-3000 DR, Rotterdam, the Netherlands
ricardo.laanen@outlook.com

2nd Flavius Frasincar
Erasmus University Rotterdam
P.O. Box 1738
NL-3000 DR, Rotterdam, the Netherlands
frasincar@ese.eur.nl

***Abstract*—Information overload is increasingly common due to the overwhelming number of online products and services offered online. Web shops use recommender systems to provide personalized suggestions, which are effective for users with historical data but face challenges with new users due to the cold-start problem. It is useful to recommend items that offer a small but representative insight into new users' preferences, so that future recommendations can be improved. This research tests two error-based active learning strategies, which are especially useful because they are expected to both produce informative training points as well as reduce the predictive error of the model: the Y-change method and a modified version of the CV-based method called the error-change method. These methods rank items for predicting new user preferences, with proposed conservative, moderate, and risky versions. Results are compared with a random selection method and the PopGini method, which was the top-performing strategy in a research with a similar set-up. The moderate version of the error-change method significantly outperforms the other considered methods, while the Y-change method still outperforms PopGini.**

***Index Terms*—cold-start problem, recommender systems, error-based learning, active learning**

I. Introduction

Over the last decades, the explosive growth of the Web and the increase in available information have transformed our way of life. Most industries have moved their business online, giving us access to a global marketplace and a vast reservoir of knowledge with just a few clicks. This potential for connectivity and discovery has introduced various side effects for Web users.

The sheer volume and diversity of available information and products can overwhelm users [1]. This problem, known as information overload, makes it difficult to process, prioritize, or understand the vast amount of information at their disposal, limiting their ability to navigate resources and make decisions.

Recommender systems suggest recommendations based on users' personal information [2]. They are crucial in filtering abundant online information and have become essential tools in e-commerce (e.g., Web shops), social media (e.g., Instagram), and entertainment (e.g., Netflix). By using known user preferences, historical data, and advanced machine learning techniques, these systems guide users toward content that aligns with their interests and needs.

The basic principle of recommender systems is that dependencies between user-centric and item-centric data can be learned from an evaluation matrix [3]. During the advent of large-scale e-commerce, information overload was addressed using adaptable systems [4]. Adaptable systems allow users to modify them, while adaptive systems can autonomously adjust themselves.

A key issue with recommender systems is the cold-start problem, which occurs when a new user or item is introduced. The lack of historical interaction data means that the user or item profile does not provide enough information to the system, leading to poor performance [5]. Consequently, the system may provide inaccurate or inadequate recommendations, as traditional methods rely on historical data to make suggestions.

The cold-start problem leads to various inefficiencies. New users may disengage due to initial inaccuracies [6], and face an overwhelming number of options that could lead them to abandon the site entirely. Likewise, new items struggle to gain visibility, leading to under-utilization of potentially superior recommendations. Addressing the cold-start problem in recommender systems is crucial for improving user satisfaction and recommendation accuracy.

This paper aims to address the cold-start problem for new users in recommender systems by proposing new strategies for recommending items without historical data. Latent factor models based on matrix factorization have greatly improved accuracy and flexibility [7]. This technique describes both items and users using factors inferred from past interactions. When a user's factors align with an item's factors, that item is likely a good recommendation [7]. Besides improving accuracy, matrix factorization is advantageous because it does not rely solely on explicit feedback. While explicit data require users to actively provide their preferences, implicit data reflect user preferences through their behavior [7].

Active learning strategies can help mitigate the cold-start problem by guiding the model's sampling process based on historical data [8]. In [9] uncertainty-reduction, ensemble-based, and error-reduction strategies are proposed.

Uncertainty-reduction methods aim to select training points that reduce uncertainty in some aspects such as values [9]. However, these methods may lead to the system becoming

confident about incorrect user preferences, reducing accuracy.

Ensemble-based methods combine multiple models, which can be beneficial as different models might fit different users better [9]. The downside is that such models are computationally expensive and harder to interpret.

Error-based active learning focuses on reducing predictive error by selecting training points based on their correlation with error reduction [9]. This approach not only yields useful training points but also lowers the model's overall predictive error.

In [9] the differences between instance-based and model-based methods are also highlighted. Instance-based methods select training points based on prediction changes, while model-based methods use fluctuations in model parameters [9]. Instance-based methods do not require knowledge of the model's underlying parameters.

This paper examines whether instance-based and error-based active learning methods can surpass existing strategies, specifically using implicit user data. Conservative, moderate, and risky versions are explored and results are compared with uncertainty-reduction methods from [10], tested on the same implicit data. The Python code used in this research is freely available at https://github.com/rmlaanen/RS-error-based-learning.

II. Methodology

In this section the notation used in the remaining parts of the paper is described and the Single Value Decomposition (SVD) techniques used to construct the prediction model are shown. The last subsection presents the error-based active learning strategies that are used to rank the candidate items for the recommender system.

A. Notation

In the dataset, the number of users is represented as N. The complete set of users is given by $U = \{u_1, u_2, \ldots, u_N\}$. The number of items is represented as M. The complete set of items is given by $I = \{i_1, i_2, \ldots, i_M\}$.

An interaction of a user u with an item i is represented as a_{ui}. The full user-item interaction matrix is given by $A \in \mathbb{R}^{N \cdot M}$. As previously mentioned, the value of the interactions is binary, which means either '0' or '1'. Therefore, the domain of a_{ui} is given by $a_{ui} \in \{0, 1\}$.

A candidate item is represented as i_x. The hypothetical cold user introduced to the system is u_0. When a candidate item is evaluated as part of the Y-change and error-based methods, a training point is added to the training set with item i_x, user u_0, and rating $y \in Y$.

The function of the preferences of users (unknown to the system) is given as $f(u, i)$, and the approximated function of user preferences is given as $\hat{f}(u, i)$. The generalization error $G(\hat{f}(u, i))$ is a measure of predictive accuracy of the estimated user preferences. The goal of the error-based learning techniques is to find a candidate item i_x which minimizes the active learning criterion $\hat{G}(i_x)$. $\hat{G}(i_x)$ is a measure of predictive accuracy after a candidate item i_x rated by a hypothetical cold user, user u_0, is added to the training set.

B. Matrix Factorization

Matrix factorization maps both user-item interaction matrix $A \in \mathbb{R}^{N \cdot M}$ to a joint latent space of dimensionality n, where user-item interactions are modeled as inner products of this joint latent space [7]. The user latent factor matrix, $p_i \in \mathbb{R}^n$, can be regarded as the preference of every user towards each latent factor, while the item latent factor matrix, $q_i \in \mathbb{R}^n$, can be regarded as the resemblance of every item to each latent factor [10].

The product of the matrixes, $q_i^T p_u$, captures the overall interest of user u in the characteristics of item i [7]. The prediction of the interaction value can be estimated using:

$$\hat{a}_{ui} = q_i^T p_u \tag{1}$$

The mapping of each item and user to the latent factor vectors q_i and p_u is usually achieved through SVD [7]. However, SVD has difficulties dealing with a sparse interaction matrix [10], such as the one used in this research. One possible solution proposed in [13] is imputation, however, in [7] it is suggested this is computationally expensive and may distort the data, instead it is proposed to directly model the observed ratings only. Through a regularized model, overfitting can be alleviated [14]. The coefficients are chosen so that they minimize the regularized squared error:

$$\min_{p^*, q^*} \sum_{(u,i) \in \mathbb{R}} \left(a_{ui} - q_i^T p_u\right) + \lambda \cdot \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{2}$$

Here, $\mathbb{R}$ denotes the set of user-item interactions $a_{ui} \subseteq A$ included in the training set. The constant λ is determined using cross-validation and controls the amount of regularization. The predicted interaction value a_{ui} can not be predicted using the scalar product $q_i^T p_u$ on its own. A bias term is included in accordance with [7]:

$$b_{ui} = \mu + b_u + b_i \tag{3}$$

Here, μ denotes the overall average rating. b_u and b_i denote the observed deviations of user u and item i from the average [7]. Equation 1 must now include the bias term:

$$\hat{a}_{ui} = q_i^T p_u + \mu + b_u + b_i \tag{4}$$

The predicted interaction value consists of four components: the user-item interaction, the global average, the user bias, and the item bias [7]. Equation 2 must now be updated with the bias term as well:

$$\min_{p,q} \sum_{(u,i) \in \mathbb{R}} \left(a_{ui} - q_i^T p_u - \mu - b_u - b_i\right)^2 + \lambda \cdot \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{5}$$

In [10] the inclusion of a second regularization parameter for the bias b_{ui} is proposed, as this makes the model more

flexible. Thus, the regularized squared error is minimized as follows:

$$\min_{p,q} \sum_{(u,i)\in\mathbb{R}} \left(a_{ui} - q_i^T p_u - \mu - b_u - b_i\right)^2 + \lambda_1 \cdot \left(b_u^2 + b_i^2\right) + \lambda_2 \cdot \left(\|q_i\|^2 + \|p_u\|^2\right) \quad (6)$$

In this equation, λ_1 controls the extent of regularization on the bias term as proposed by [10], and λ_2 controls the extent of regularization on the latent factors as proposed in [7]. The predictions need to be converted into a binary variable, for which the following logistical function can be used [10]:

$$\hat{a}_{ui}^* = \left[\frac{1}{1 + exp(-\hat{a}_{ui})}\right] \quad (7)$$

Here, $\hat{a}_{ui}^*$ represents the predicted binary value of the user-item interaction, as the hard brackets round the enclosed function to the nearest integer (either 0 or 1).

Equation 6 is solved by numerical methods. In [7] two methods are suggested: Stochastic Gradient Descent (SGD) and Alternating Least Squares (ALS). SGD is used in this paper instead of ALS, since the amount of computations per iteration is lower for SGD [10]. For each training point, the model computes $\hat{a}_{ui}$ and the associated prediction error with the following equation [7]:

$$e_{ui}^{pred} = a_{ui} - \hat{a}_{ui} \quad (8)$$

Consequently, b_u, b_i, p_u and q_i are updated using the learning parameters γ_1 and γ_2 [10]:

$$b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \gamma_1 \cdot b_u) \quad (9)$$

$$b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \gamma_1 \cdot b_i) \quad (10)$$

$$p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \gamma_2 \cdot p_u) \quad (11)$$

$$q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot p_u - \gamma_2 \cdot q_i) \quad (12)$$

C. Employed Strategies

For a cold user, the user vector is empty, as a cold user has not interacted with any items, preventing personalized recommendations [10]. To provide such recommendations, the preferences of user u_0 towards several items need to be determined, requiring the true values of $\{a_{u1}, a_{u2}, ..., a_{uj}\}$, where j is finite and $j \ll M$ [10].

The goal is to establish u_0's preferences for these j items to achieve the best personalized recommendations [10]. Three variants of two error-based active learning strategies are employed to rank items, with the highest-ranking items presented to cold users. These strategies are non-personalized, as they do not use user-specific information and require each user to rate the same set of items [8]. Results are compared against a random strategy to assess whether the rankings are superior to random selection and against the PopGini strategy, identified by [10] as the top-performing active learning method.

a) Random Strategy: The first strategy is the random strategy, which selects items randomly to create the item ranking. It is anticipated to be outperformed by all other active learning methods [10] and serves as a baseline to evaluate improvements from other strategies.

b) Y-Change Strategy: In [11] the output estimate change (Y-change) model is proposed, under the assumption that changes in output estimates generally lead to increased accuracy. An item that causes significant changes in output estimates is deemed useful [9]. To apply this strategy, the user-item interaction matrix A, is divided into a training set T with 70% of interactions and a test set A^{Test} with the remaining 30%. Cold user interactions are retained in the matrix to avoid computational expense and to ensure sufficient data are available. The impact on item ranking is expected to be minimal. The Y-change generalization error of a candidate item, $\hat{G}_{\Delta Y}(i_x)$, measures the change in output estimates of A^{Test} when the candidate item i_x is added to the training set T. The change in output estimates is calculated using the following equation:

$$\hat{G}_{\Delta Y}(i_x) = - \sum_{y\in Y,(u,i)\in A^{Test}} \mathcal{L}(\hat{f}_T(u,i), \hat{f}_{T\cup(i_x,u_0,y)}(u,i)) \quad (13)$$

Here, $\hat{G}_{\Delta Y}(i_x)$ represents the generalization error computed using the Y-change method for the candidate item i_x. To determine which candidate item results in the greatest change in output estimates, all possible ratings $y \in Y$ (in this paper, 0 or 1) are considered [9]. The generalization error is calculated using a loss function $\mathcal{L}$. $\hat{f}_T(u,i)$ denotes the estimated rating for an item i by user u using the learned function from the current training set T. $\hat{f}_{T\cup(i_x,u_0,y)}(u,i)$ denotes the estimated rating for item i by user u using the learned function from the training set $T \cup (i_x, u_0, y)$, where a training point with the candidate item i_x, hypothetical cold user ID u_0, and hypothetical rating y has been added to T. This training point represents a hypothetical rating by a cold user for the candidate item. The new estimates are compared against those from the original training set T, addressing the question: "How much would our predictions change if a cold user u_0 rated candidate item i_x with rating y?" Equation 13 can be rewritten as follows:

$$\hat{G}_{\Delta Y}(i_x) = - \sum_{y\in Y,(u,i)\in A^{Test}} (\hat{f}_T(u,i) - \hat{f}_{T\cup(i_x,u_0,y)}(u,i))^2 \quad (14)$$

In this equation, the loss function has been replaced by the squared error function, which is the squared difference between the previous estimates learned from $\hat{f}_T$ and the new estimates learned from $\hat{f}_{T\cup(i_x,u_0,y)}$.

Here, $\hat{G}_{\Delta Y}(i_x)$ calculates the generalization error using all possible ratings, and adds these together. However, the three proposed variants of the Y-change method each require a modification to the equation to arrive at the final respective generalization errors. The conservative variant $\hat{G}_{\Delta Y-C}(i_x)$

uses the ratings which minimize the change in output estimates. The moderate variant $\hat{G}_{\Delta Y-M}(i_x)$ takes the average of all ratings, while the risky variant $\hat{G}_{\Delta Y-R}(i_x)$ uses the ratings which maximize the changes in output estimates:

$$\hat{G}_{\Delta Y-C}(i_x) = -\min_{y \in Y} \sum_{(u,i) \in A^{Test}} (\hat{f}_T(u,i) - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{15}$$

$$\hat{G}_{\Delta Y-M}(i_x) = -\frac{1}{|Y|} \sum_{y \in Y,(u,i) \in A^{Test}} (\hat{f}_T(u,i) - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{16}$$

$$\hat{G}_{\Delta Y-R}(i_x) = -\max_{y \in Y} \sum_{(u,i) \in A^{Test}} (\hat{f}_T(u,i) - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{17}$$

c) Error-Change Strategy: The second strategy is the error-change method, based on the CV-based method introduced by [12], which is a strategy based on predictive accuracy. The candidate item is selected based on how well it could allow for approximating ratings which are in the test set [9]. A new training point with item i_x, user u_0, and rating y is added to the training set T, after which an approximation $\hat{f}_{T \cup (i_x,u_0,y)}(u,i))$ of the user preferences is obtained, which accuracy is evaluated using the test set A^{Test} [9]. The candidate item's usefulness is measured using the following equation:

$$\hat{G}_{\Delta E}(i_x) = \sum_{y \in Y,(u,i) \in A^{Test}} \mathcal{L}(a_{ui}, \hat{f}_{T \cup (i_x,u_0,y)}(u,i)) \tag{18}$$

Here, $\hat{G}_{\Delta E}(i_x)$ is the generalization error determined using the error-change method for the candidate item i_x. The generalization error is calculated using a loss function $\mathcal{L}$. a_{ui} are the known ratings. $\hat{f}_{T \cup (i_x,u_0,y)}(u,i)$ are the estimated ratings for an item when the candidate training point with item i_x, user u_0, and the hypothetical rating y has been added to the training set T. Equation 18 can be rewritten as follows:

$$\hat{G}_{\Delta E}(i_x) = \sum_{y \in Y,(u,i) \in A^{Test}} (a_{ui} - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{19}$$

In this equation, the loss function has been replaced by the squared error function, which is the squared difference between the actual ratings a_{ui} and the new estimates learned by $\hat{f}_{T \cup (i_x,u_0,y)}(u,i)$.

Here, $\hat{G}_{\Delta E}(i_x)$ calculates the error-change generalization error using every potential rating. However, similar to the Y-change method, there are three proposed variants of the error-change method. The conservative variant $\hat{G}_{\Delta E-C}(i_x)$ uses the ratings which maximize the errors. The moderate variant $\hat{G}_{\Delta Y-M}(i_x)$ takes the average error of all ratings, while the risky variant $\hat{G}_{\Delta E-R}(i_x)$ uses the ratings which minimize the errors of the estimates. This is reflected in the following equations:

$$\hat{G}_{\Delta E-C}(i_x) = \max_{y \in Y} \sum_{(u,i) \in A^{Test}} (a_{ui} - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{20}$$

$$\hat{G}_{\Delta E-M}(i_x) = \frac{1}{|Y|} \sum_{y \in Y,(u,i) \in A^{Test}} (a_{ui} - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{21}$$

$$\hat{G}_{\Delta E-R}(i_x) = \min_{y \in Y} \sum_{(u,i) \in A^{Test}} (a_{ui} - \hat{f}_{T \cup (i_x,u_0,y)}(u,i))^2 \tag{22}$$

To perform item ranking using the error-change method, a similar set-up is used as for the Y-change method. For every possible combination of candidate item $i_x \in I_u$ and rating $y \in Y$, a single interaction is added to the train set and the ratings of the user-item interactions in the test set are predicted using the updated train set. Last, these are compared against the real interactions. This will answer the question: "What is the change in predictive error if a cold user evaluated candidate item i_x with rating y?" for every candidate item and possible rating.

d) PopGini Strategy: The third strategy is the PopGini strategy, which is a combination of the popularity strategy and the Gini strategy [10].

The popularity strategy ranks the most popular items, which are those items that have been most interacted with in the dataset, as the highest [10]. The Gini strategy uses the Gini impurity measure to determine the rank of items [10]. The Gini impurity measure for a candidate item i_x is given by the following formula:

$$Gini(i_x) = 1 - \sum_{y \in Y} (p(y|i_x))^2 \tag{23}$$

Here, $p(y|i_x)$ is the relative frequency of positive evaluations (where $y = 1$) and negative evaluations (where $y = 0$).

The Gini impurity measure interprets the ratio of positive and negative interactions, where an item with a high Gini impurity measure would be useful for dividing a set of (cold) users into groups that will either like or dislike a certain item [10]. The PopGini strategy is a combination of the popularity and Gini impurity measures. The PopGini score of a candidate item i_x is computed as follows:

$$PopGini(i_x) = \omega_p \times \log_{10}(A_{ui_x}) + \omega_g \times (1 - \sum_{y \in 0,1} (p(y|i_x))^2) \tag{24}$$

Here, ω_p and ω_g are weights for the popularity and Gini scores, respectively, which can be adjusted. To further control for outliers, the logarithm with base 10 is taken of the interaction frequency, given as A_{ui_x}, or the total number of interactions of all users with item i_x.

III. Evaluation

To evaluate the proposed strategies, this research uses a dataset from De Bijenkorf, a Dutch luxury department store with a Web shop that handles over 100,000 visitors daily and €250,000 in turnover from over 200,000 items [10]. The dataset comprises 2,563,878 user-product interactions, collected from July 14th, 2015 to July 13th, 2016. Interactions are binary: a value of '1' indicates a purchase or a higher number of purchases than returns, while '0' indicates an equal number of purchases and returns. A value of '1' signifies a positive rating, and '0' a negative rating. Data is implicit as users do not provide explicit ratings. To manage computational demands, 100,000 interactions were randomly sampled from the full dataset, resulting in 73,446 unique users and 55,529 items.

A. Set-Up

The strategies in this research are evaluated using the Root Mean Square Error (RMSE). The RMSE metric is chosen because it is widely used in the evaluation of models in the field of recommender systems and will allow for easy interpretation and comparison. The RMSE is calculated using the following formula:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in T}(a_{ui} - \hat{a}^*_{ui})^2}{|I^{test}|}} \tag{25}$$

Here, a_{ui} represents the actual rating of item i by user u, while $\hat{a}^*_{ui}$ denotes the predicted rating of item i by user u. The term $|I^{test}|$ denotes the total number of items in the test set. A lower RMSE indicates better model performance, as it reflects more accurate predictions.

The evaluation follows a similar experimental setup to the study by [10]. 25% of users were randomly selected to act as cold users, a choice informed by previous studies [10]. Cold users were shown 10, 25, 50, or 100 items to assess the strategies across different numbers of items.

The user-item interactions of the remaining 75% of regular users are used for training. Additionally, interactions of cold users with items in the item set under consideration (10, 25, 50, and 100 items) are included in the training set, while interactions with items not in this set are part of the test set.

Cold users who have not interacted with at least one item in the item set are excluded from both the training and test sets. This exclusion ensures that recommendations are not made to users who have no opinions on any of the considered items, preventing non-personalized recommendations.

This approach to creating the train and test sets simulates the scenario where recommendations are made to a cold user, relying only on interactions from previous users and the cold user's feedback on the items under consideration. Note that this split is distinct from the splits used to compute item rankings for the Y-change and error-change methods.

B. Model Parameters

The recommender system model was optimized through a random grid search, considering 100 combinations of model parameters, including the number of factors, bias regularization (λ_1), and latent factor regularization (λ_2).

The number of factors varied across 10, 20, 50, 100, 200, 500, and 1000. Regularization terms ranged from 1×10^{-1} to 1×10^{-8} in decreasing orders of magnitude. Each parameter combination was evaluated using 5-fold cross-validation.

The optimal parameters were found to be 100 factors, $\lambda_1 = 1 \times 10^{-8}$ for bias regularization, and $\lambda_2 = 1 \times 10^{-5}$ for latent factor regularization. These parameters were used in the recommender system model, implemented in Python with Scikit Surprise [15]. The weights for the PopGini strategy, ω_p and ω_g, were set at 0.9 and 1.0, respectively, as per [10].

C. Results

TABLE I
RMSE FOR DIFFERENT NUMBERS OF SHOWN ITEMS PER STRATEGY

Strategy	Number of Items				Mean
	10	*25*	*50*	*100*	
Random	0.482	0.450	0.446	0.446	0.456
Conservative Y-change	0.417	0.439	0.424	0.444	0.431
Moderate Y-change	0.402	0.430	0.441	0.441	0.428
Risky Y-change	0.450	0.435	0.451	0.450	0.447
Conservative error-change	0.401	0.422	0.439	0.441	0.423
Moderate error-change	**0.352**	**0.385**	**0.401**	**0.422**	**0.390**
Risky error-change	0.426	0.445	0.453	0.430	0.438
PopGini	0.453	0.449	0.448	0.441	0.448

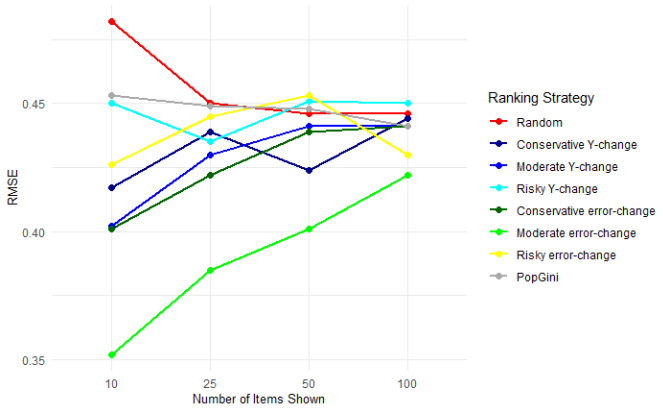

Fig. 1. Visualization of RMSE for different numbers of shown items per strategy

Table 1 shows the RMSE for each tested strategy in combination with the number of items which were shown.

The results are visualized for easy comparison in Figure 1. The RMSE decreases for most strategies as the number of items shown increases. The moderate error-change strategy (highlighted in bold) outperforms the other strategies for every number of shown items. All other error-change variants and all variants of the Y-change method yield worse results, but outperform both the PopGini and random strategy.

The moderate error-change achieves the lowest average RMSE in all scenarios, especially with fewer items, which is crucial for cold users. This suggests that items improving prediction accuracy are valuable for preference elicitation, aligning with the idea that a candidate item enhancing accuracy is useful.

The moderate variants perform best, while risky variants are less effective. A balanced approach is ideal; evaluating all possible ratings for a candidate item is more effective than using overly optimistic ratings, which often result in higher errors.

The Y-change strategy's relatively poor performance compared to error-change indicates that large prediction changes are not always accurate. The Y-change method may highlight items that cause significant but inaccurate changes, whereas the error-change method, which focuses on accuracy improvements, performs better. Despite its initial effectiveness with a small number of items, the Y-change method shows limited improvement as the number of items increases, sometimes performing worse than random selection.

IV. Conclusion

Recommender systems address information overload and personalized recommendations are now standard for online businesses. SVD algorithms with matrix factorization are crucial for predicting preferences but struggle with sparse user-item matrices and the cold-user problem.

This research evaluates two error-based active learning strategies from [9] that rank candidate items for cold users to gather preferences. The aim is to select the most informative training points to efficiently determine cold user preferences. These strategies are compared with the PopGini strategy, identified as the best in [10], and a random strategy to assess improvement over random selection. The study is novel in applying these strategies to implicit user data and introduces two new variants of the Y-change and error-change methods.

The results indicate that the moderate error-change strategy outperforms all other strategies in terms of RMSE, with the conservative error-change strategy and Y-change strategy also performing well. Overall, the error-change strategy is the best, followed by the Y-change strategy.

A. Limitations

Many other active learning strategies remain to be evaluated under similar conditions to identify the optimal approach. [9] highlights additional error-based strategies like parameter change-based, variance-based, and image restoration-based methods. Unlike the instance-based Y-change and error-change methods, these are model-based. Other potential strategies include uncertainty-reduction and ensemble-based methods [9].

Another limitation of the current research methodology is noted by [10]. In this setup, the same items are shown to all cold users, without considering individual user feedback after the items are displayed.

B. Suggestions for Future Research

The results indicate that both the error-change and Y-change strategies could surpass established methods like PopGini. Future research should explore further improving these methods.

Both Y-change and error-change strategies assume that items providing substantial information are useful. The model could be enhanced by evaluating cold users' feedback after each candidate item is shown. Ideally, the most highly ranked items should improve predictions significantly, so incorporating cold users' opinions as each item is rated might further optimize recommendations.

Combining Y-change and error-change into a single method could address their weaknesses. Y-change focuses on large prediction changes but ignores their accuracy impact, while error-change considers overall accuracy improvement without assessing how much information is gained. A hybrid method could integrate both approaches, weighting each strategy's generalization error to create a ranking that balances accuracy and informational impact.

References

[1] B. Schwartz, The Paradox of Choice: Why More Is Less, Harper Perennial, 2004.

[2] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Recommender Systems Handbook, Springer, 2011.

[3] C. C. Aggarwal, Recommender Systems: The Textbook, Springer, 2016.

[4] T. W. Malone, K. A. Grant, and F. Turbak, "The information lens: An intelligent system for information sharing in organizations," SIGCHI Bulletin, vol. 17, no. 4, 1986, pp. 1–8.

[5] N. T. Nguyen, M. Rakowski, M. Rusin, J. Sobecki, and L. C. Jain, "Hybrid filtering methods applied in Web-based movie recommendation system," in Proceedings of the 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES-2007), Springer, 2007, pp. 206–213.

[6] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, "Generalization of recommender systems: Collaborative filtering extended to groups of users and restricted to groups of items," Expert Systems With Applications, vol. 39, no. 1, 2012, pp. 172–186.

[7] Y. Koren, R. E. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," IEEE Computer, vol. 42, no. 8, 2009, pp. 30–37.

[8] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," Computer Science Review, vol. 20, no. 2, 2016, pp. 29–50.

[9] N. Rubens, D. Kaplan, and M. Sugiyama, "Active learning in recommender systems," in Recommender Systems Handbook, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2010, pp. 735–767.

[10] T. Geurts and F. Frasincar, "Addressing the cold user problem for model-based recommender systems," in Proceedings of the International Conference on Web Intelligence (WI-2017), ACM, 2017, pp. 745–752.

[11] N. Rubens, R. Tomioka, and M. Sugiyama, "Output divergence criterion for active learning in collaborative settings," IPSJ Online Transactions, vol. 2, 2009, pp. 240–249.

[12] S. A. Danziger, J. Zeng, Y. Wang, R. K. Brachmann, and R. G. Lathrop, "Choosing where to look next in a mutation sequence space: Active learning of informative p53 cancer rescue mutants," Bioinformatics, vol. 23, no. 13, 2007, pp. 104–114.

[13] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender systems: A case study," Technical report, University of Minnesota, 2000.

[14] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2007), ACM, 2007, pp. 39–42.

[15] N. Hug, "Surprise: A Python library for recommender systems," Journal of Open Source Software, vol. 5, no. 52, 2020, pp. 2174.