Continuous Hyperparameter Optimization for Large-scale Recommender Systems

Simon Chan, Philip Treleaven, Licia Capra
Department of Computer Science
University College London
London, United Kingdom
{*m.chan, p.treleaven, l.capra*}*@cs.ucl.ac.uk*

***Abstract*—While the prediction accuracy of a large-scale recommender system can generally be improved by learning from more and more training data over time, it is unclear how well a fixed predictive model can handle the changing business dynamics in a real-world scenario. The adjustment of a predictive model is controlled by the hyperparameter settings of a selected algorithm. Although the problem of hyperparameter optimization has been studied for decades in various disciplines, the adaptiveness of the initially selected model is not as well understood. This paper presents an approach to continuously re-select hyperparameter settings of the algorithm in a large-scale retail recommender system. In particular, an automatic hyperparameter optimization technique is applied on collaborative filtering algorithms in order to improve prediction accuracy. Experiments have been conducted on a large-scale real retail dataset to challenge traditional assumption that a one-off initial hyperparameter optimization is sufficient. The proposed approach has been compared with a baseline approach and a widely used approach with two scalable collaborative filtering algorithms. The evaluations of our experiments are based on a 2-year real purchase transaction dataset of a large retail chain business, both its online e-commerce site and its offline retail stores. It is demonstrated that continuous hyperparameter optimization can effectively improve the prediction accuracy of a recommender system. This paper presents a new direction in improving the prediction performance of a large-scale recommender system.**

***Keywords*-Hyperparameter Optimization; Model Selection; Collaborative Filtering;**

I. Introduction

Large-scale recommender systems are increasingly important for retail businesses. Retailers provide personalized product recommendations to consumers through various mediums, for instance, web advertisement, email newsletters and in-store discount vouchers. Recommender systems attempt to accurately predict consumers' preferences, or their behaviors, so that they can recommend products according to the best expected scenarios. Collaborative Filtering (CF) is one of the most popular techniques to build recommender systems, as it relies on previous consumer behavioral data only. For instance, previous data of purchase transactions or website browsing history of consumers is sufficient for CF algorithms to predict consumers' future preferences. Consumer profiles, such as their age, gender and other attributes, are not required. Product profiles are not required either. Commercial companies such as Amazon [1] and YouTube [2] have successfully applied CF to build recommender systems for production use. The two main classes of algorithms under CF are neighborhood models [3] and latent factor models [4]. Recently, latent factor models have gained popularity in both the research community and in the industry due to their superior accuracy and scalability. The discussion and our proposed approach in this paper are therefore mainly based on latent factor models. The same approach, however, can be applied to neighborhood models directly.

Large retail businesses collect huge amounts of data every day. Scalable algorithms that can process data and construct predictive models using multiple machines in parallel are used in large-scale recommender systems. In this paper, we focus on two scalable latent factor models, namely Alternating Least Squares with Weighted Regularization (ALS-WR) [5] and Stochastic Gradient Descent (SGD) [6], that construct consumer and product feature matrices in Singular Value Decomposition (SVD) form. Both algorithms can be run on distributed Hadoop system [7]. Like almost all other algorithms, these algorithms require the setting of one or more hyperparameters, such as regularization and learning rate, to build the latent factor models. These hyperparameters affect the prediction accuracy of the resulting models directly. Hyperparameter optimization problems and techniques have been studied in several disciplines such as Machine Learning and Statistics. Grid search and random search [8] are two popular methods for global optimization. All these studies, however, take for granted that hyperparameter optimization (or setting) is a one-off initial process that needs to be done only once. No study, to the best of our knowledge, has been conducted on the effect of re-selecting new hyperparameters for learning algorithms after new data has been collected over time.

Retail business environment changes rapidly. Product prices, for instance, may be changing frequently and new products appear regularly. Simply re-training the same predictive model with the addition of new data may not be sufficient to accommodate these changes. Our hypothesis is that a recommender system model may render less accuracy, or even become invalid, as the business dynamic changes over time. We propose a continuous hyperparameter optimization approach, in which hyperparameters are re-

evaluated, and are re-selected if needed, before the model is re-trained with new data. The prediction accuracy of this approach is compared to traditional non-continuous approaches empirically based on a large and real retail datasets that contain both online e-commerce transactions and offline in-store transactions. Experiments show that the proposed approach outperforms a baseline approach and a state-of-the-art approach significantly in our retail scenarios.

The rest of the paper is organized as follows. We introduce the definitions of preference prediction and hyperparameter optimization in Section 2. We present and analyze the proposed approach in Section 3. We provide experimental evaluations in Section 4. We review the related work in Section 5 and conclude in Section 6.

II. Preference Prediction

In this paper, we narrowly define the objective of a recommender system is to suggest the top N products to each customer that he or she will most likely purchase. This kind of system, broadly speaking, involves a two-step process: preference prediction and ranking. During the prediction step, the system predicts a preference score on each available product for a targeted consumer. Higher score means that the consumer likes the product more and therefore has a higher chance to make a purchase transaction. Then, during the ranking step, the system ranks all available products for this consumer according to the predicted score. The top-ranked N products will be returned as recommendation results. Here are the formal definitions:

Definition 1 (Prediction): In our retail scenario, a prediction of consumer preference can be regarded as an estimation of the probability a consumer will purchase a product, i.e. $p(purchase = 1|u, i)$ where u is a targeted consumer and i is a targeted product. Preference prediction of all customers on all products can be represented by a $M \times N$ matrix $\boldsymbol{r}$ where M is the total number of consumers, N is the total number of products and $r_{mn} = p(purchase = 1|m, n)$. There are two main techniques of Collaborative Filtering for prediction: the neighborhood approach and latent factor models approach. Latent Factor Models is a popular discipline for Collaborative Filtering that transforms both products and consumers to the same latent factor space.

Definition 2 (Hyperparameter Optimization): The term hyperparameter is used to distinguish it from regular parameters of a predictive model. Regular parameters characterize the model and they are estimated by model training using existing data. Hyperparameters, on the other hand, control how these regular parameters are estimated. Therefore, hyperparameter optimization is also called model selection. Like most machine learning algorithms, most collaborative filtering algorithms come with some tunable hyperparameters such as learning rate and regularization. Selecting appropriate settings for these hyperparameters is crucial for the accuracy of prediction. Hyperparameter optimization, or model selection, can be defined formally as an optimization problem itself, which can be written as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, C(\boldsymbol{p}_v, \boldsymbol{q}_v, \boldsymbol{r}_v, M(\boldsymbol{p}_{tr}, \boldsymbol{q}_{tr}, \boldsymbol{r}_{tr}, \theta))$$

and the performance of a set of hyperparameter settings can be evaluated by a cost function:

$$C(\boldsymbol{p}_{test}, \boldsymbol{q}_{test}, \boldsymbol{r}_{test}, M(\boldsymbol{p}_{tr+v}, \boldsymbol{q}_{tr+v}, \boldsymbol{r}_{tr+v}, \theta^*))$$

where C is the cost function, M is the modeling function, θ is a hyperparameter set, $\boldsymbol{p}_{tr}$ and $\boldsymbol{q}_{tr}$ are the consumer and product feature vectors of the training dataset, $\boldsymbol{p}_v$ and $\boldsymbol{q}_v$ are the consumer and product feature vectors of the validation dataset, $\boldsymbol{p}_{test}$ and $\boldsymbol{q}_{test}$ are the consumer and product feature vectors of the test dataset and $\boldsymbol{r}_{tr}$, $\boldsymbol{r}_v$ and $\boldsymbol{r}_{test}$ are the vectors of preference scores of training, validation and test datasets respectively.

III. Proposed Continuous Modeling

In this paper, we study the effect of conducting automatic hyperparameter optimization continuously on recommender systems in a retail chain business. Our hypothesis is that a continuous hyperparameter optimization approach can improve prediction accuracy of a recommender system in a retail environment. We propose a novel approach to search for better hyperparameters for a targeted algorithm before each iteration of model re-training with the addition of new data. Algorithm 1 is an overview of the process. Major components of the proposed approach is described as follows.

Data: D (dataset of month 1-24), A (algorithm)
Result: RMSE and MAP@k on each month of 13-24
$trainD$ = D[month 1-12];
for $m \in \{13..24\}$ **do**
  θ^* = select hyperparameters of A based on $trainD$;
  P = model of A trained by $trainD$ and θ^*;
  evaluate RMSE of P on $D[m]$;
  evaluate MAP@k of P on $D[m]$;
  $trainD = trainD + D[m]$;
end

Algorithm 1: Continuous Hyperparameter Optimization

A. Collaborative Filtering

Retail businesses, especially those operate as a chain, collect huge amount of data every day. A parallel distributed collaborative filtering algorithm is necessary to handle the data in a scalable way. In our experiments, two popular distributed collaborative filtering algorithms, SGD and ALS-WR, are implemented. We also apply a technique to handle data of implicit consumer feedback since consumers do not express their preferences explicitly, such as rating products, in many retail scenarios.

1) Stochastic Gradient Descent: SGD initializes feature vectors that represent the profiles of consumers and products with random values. It then computes the gradient of the cost function and updates the values with steps in the direction of the gradient based on training data [9], [10]. This paper follows the SGD implementation of [11], namely Bias SGD, which is a slightly improved version of pure SGD as it computes a bias for each consumer and product. The prediction accuracy is improved at the cost of extra computation. The tuning of three hyperparameters, which are listed on Table I, is needed in order to apply a SGD algorithm to construct a predictive model.

Parameter Name	Description
λ	regularization to prevent overfitting
$lrate$	learning rate
$decay$	learning rate decay

Table I: Hyperparameters of Bias SGD Algorithm

2) Alternating Least Squares with Weighted Regularization: ALS-WR also initializes feature vectors that represent the profiles of consumers and products with random values. It then updates the feature vectors by applying a least squares solution to solve a quadratic cost function [5]. Although the computational cost of a least squares is more expensive than the one of SGD, usually fewer ALS iterations are required to obtain similar prediction accuracy as SGD. Also, the distributed implementation of ALS is more efficient than the one of SGD. Furthermore, ALS requires less hyperparameter settings. The only hyperparameter required by ALS-WR is shown in Table II.

Parameter Name	Description
λ	regularization to prevent overfitting

Table II: Hyperparameters of ALS Algorithm

3) Implicit Feedback: In real world retail environment, consumer behavioral data is usually collected implicitly. Consumers are not required to express their preferences on products explicitly as it may intrude their shopping experiences. Therefore, the most common form of data being collected is non-negative feedback, such as purchase transactions and website browsing history. Explicit feedback such as rating is rarely collected. Under this situation, many collaborative filtering algorithms cannot be applied directly as we lack feedback on which products consumers dislike. Some techniques can be applied to solve this problem. For ALS-WR, consumers' purchase history is transformed into confidence for preference. The cost function takes not only consumer and product pairs that have interactions in the past into consideration, but all other unobserved consumer and product pairs as well. The details of this technique has been discussed in [12]. For SGD, we follow the technique described in [13], which is a straightforward procedure to add negative examples randomly for unobserved consumer and product pairs.

B. Update with New Data

Existing literature assumes that hyperparameter optimization for collaborative filtering algorithms needs to be done once only at the initialization stage. Under this assumption, recommender systems re-train models with the same hyperparameter settings when new data comes in. This paper challenges this status quo. Our hypothesis is that the dynamics of real world environments, especially in retail businesses, change rapidly, a model that is appropriate in the past may become less effective as time goes by. Prediction accuracy of a recommender system should be improved if the model is updated, not only re-trained, continuously. To update the model continuously, the system should be able to automatically select new hyperparameter settings for the underlying algorithm when more data is collected. Tuning these hyperparameters manually is not practical as it would require human involvement every time. Automatic hyperparameter optimization technique is discussed in the next subsection.

C. Automatic Hyperparameter Optimization

Automatic hyperparameter tuning has been studied in various disciplines, such as Machine Learning and Statistics. While some techniques are algorithm-specific such as [14] and [15], some global optimization approaches can be applied to any algorithms, including collaborative filtering algorithms. Two widely used global optimization approaches are *grid search* and *random search* [8]. Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. Random search, on the other hand, selects a value for each hyperparameter independently using a probability distribution. Both approaches evaluate the cost function based on the generated hyperparameter sets. Despite its simplicity, random search has proven to be more efficient than grid search in some cases empirically [8]. Since we mainly concern about the comparison between the traditional one-off modeling approach and our proposed continuous modeling approach in this paper, the detailed study of the efficiency of hyperparameter optimization techniques is not within the scope. It is therefore reasonable to choose either one of these techniques to perform automatic hyperparameter optimization. The random search technique is chosen due to its simplicity.

IV. Experimental Evaluations

A. Data Set

Our dataset, which is provided by a large UK retail chain business, is a 2-year anonymized product purchase records of loyalty card holders on the retailer's e-commerce site and in all physical stores of the retailer in the UK. It contains

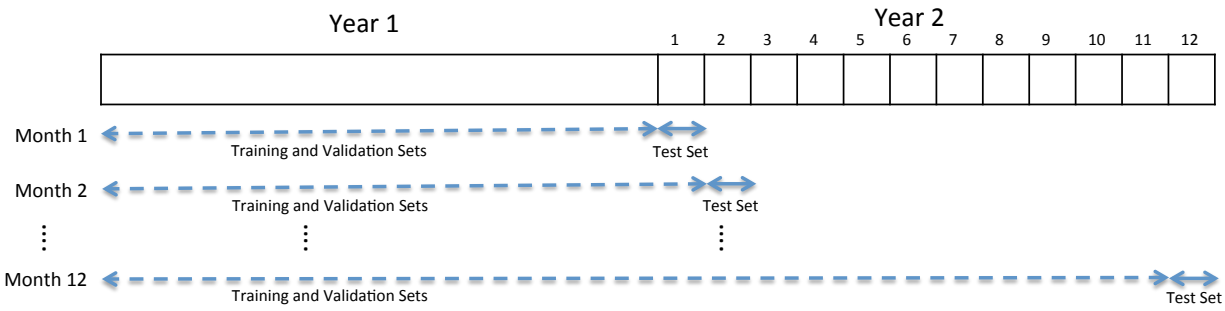


Figure 1: Training, Validation and Test Sets Split for 12 Months

complete transaction records of 10,217,972 unique loyalty card holders and 2,939 unique products under 10 selected brands. There are 21,668,137 in-store purchase transaction records and 2,583,531 online purchase transaction records. All data is collected in a real non-experimental setting.

We take all records of purchase transactions and give each of them a preference score of 1. For instance, consumer u purchases a product i at time t is represented by a comma-separated line: $u, i, t, \mathbf{1}$. The resulting dataset contains only consumer-product pairs with preference score of 1 only. This transaction dataset is further divided into two: One contains transaction records of the online e-commerce site, and the other one contains transaction records of offline retail stores.

B. Evaluation Metrics

One way to evaluate the prediction accuracy of a recommender system is to measure its prediction error of preference scores directly. Recommender systems predict consumer preferences based on predictive models that are built for predicting preference scores of unseen consumer-product pairs. A preference score ranges between 0 and 1, which can also be seen as an estimated probability of a consumer purchasing a product, i.e. $p(purchase = 1|u, i)$ where u is the targeted consumer and i is the targeted product. A higher score means that the consumer is more likely to purchase the product. The predicted scores will be evaluated against test sets that contain the actual purchase transactions i.e. consumer-product pairs with score of 1. Root Mean Squared Error (RMSE) can be used as the prediction error metric: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}$, where n is the total number of pairs in a test set, r_i is the true score which is always 1 in this case and $\hat{r}_i$ is the predicted score. The lower the RMSE score, the more accurate the predictive model is.

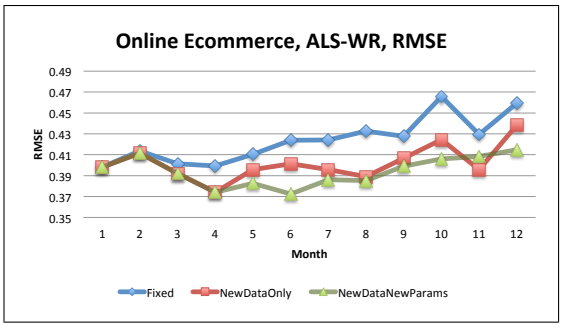
Normally, the end result of a recommender system is to return a top-K recommendation list. Another way to evaluate the prediction accuracy of a recommender system is, therefore, to treat the prediction as a ranking problem. Specifically, we want to evaluate the average precision of the predicted recommendation list for each consumer [16]. Suppose a consumer has purchased n products in the test dataset and the system can recommend up to K products to this consumer. The average precision score at K, i.e. ap@K, is: $ap@K = \sum_{k=1}^{K} P(k)/min(n, K)$ where $P(k) = 0$ if the consumer has not purchased the k-th product of the recommended list in the test dataset and $P(k) = k$ if otherwise. The mean average precision for M consumers at K, i.e. MAP@K, is the average of the average precision of each consumer, which is defined as: $MAP@K = \sum_{m=1}^{M} ap@K/M$. The higher the MAP@K score, the better the recommender system performs. We are going to evaluate our proposed continuous hyperparameter optimization approach with both RMSE and MAP@K metrics.

C. Experimental Setup

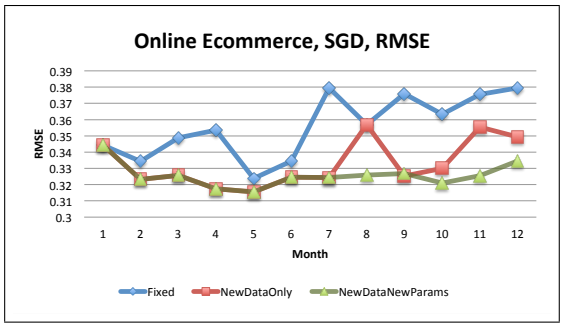
In this experiment, we evaluate the effect of continuous hyperparameter optimization under two separate environments, namely online e-commerce and offline stores. Under each environment, the prediction accuracy over time of three approaches is compared. In particular, this experiment evaluates and compares the consumer preference prediction accuracy of predictive models built, and updated, by three approaches on each month of the second year of our dataset, as illustrated in Fig. 1. The detailed procedure is described as follows.

We first evaluate the online e-commerce environment along with ALS-WR algorithm. We use $D1$ to represent all online e-commerce transaction data of the first year and $D2_m$ to represent all online e-commerce transaction data of the m month of the second year. For instance, $D2_1$ means all transaction data of the 1^{st} month of the second year. In this experiment, we define the width of the feature vectors for consumers and products as 20 and the maximum number of iterations as 30. By using random search technique, a pool of 1000 available hyperparameter sets, known as $\boldsymbol{\theta}$, is generated for the only hyperparameter of ALS-WR: the search range of λ is [0.01, 0.1]. The RMSE and MAP@20 of the following three approaches on each $D2_m$ are evaluated:

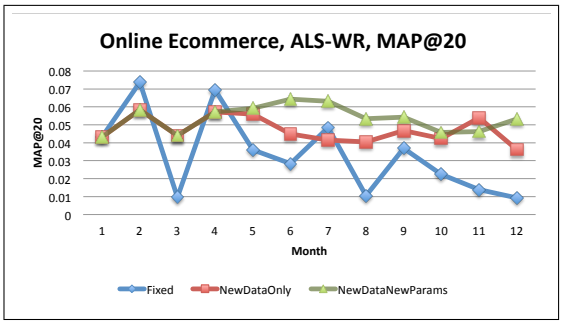
Fixed: The first approach starts by selecting one set of hyperparameters from $\boldsymbol{\theta}$ that performs the best based on the first year data. This set of hyperparameters is written as



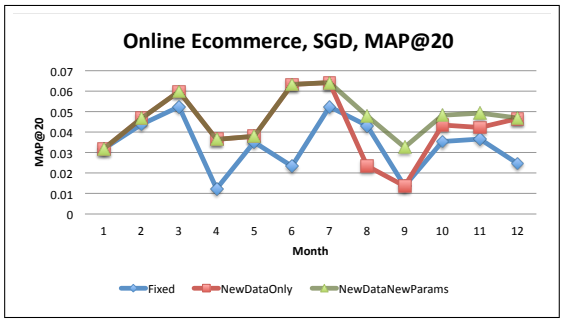
(a) RMSE on ALS-WR - 2nd year e-commerce



(b) RMSE on SGD - 2nd year e-commerce



(c) MAP@20 on ALS-WR - 2nd year e-commerce



(d) MAP@20 on SGD - 2nd year e-commerce

Figure 2: Results on Online E-commerce Dataset

$S_{\boldsymbol{\theta}}(D1)$. A predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built using $D1$ and $S_{\boldsymbol{\theta}}(D1)$. We evaluate the RMSE and MAP@20 of this model on each month of the second year, i.e. $D2_m$. This model is not updated with any new data. It serves as the baseline to compare with the other two approaches.

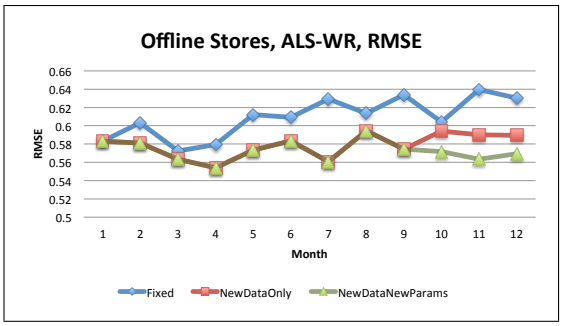
NewDataOnly: The second approach also starts by selecting the same set of hyperparameters, i.e. $S_{\boldsymbol{\theta}}(D1)$, from $\boldsymbol{\theta}$. A predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built as well. We evaluate the RMSE and MAP@20 of this model on the *first month* of the second year, i.e. $D2_1$, only. Then, we re-train the predictive model using $S_{\boldsymbol{\theta}}(D1)$ and the first year data *plus* the data of the first month of the second year, which becomes $M(D1 + D2_1, S_{\boldsymbol{\theta}}(D1))$. We evaluate the RMSE and MAP@20 of this updated model on $D2_2$. Similarly, we build $M(D1+D2_1+D2_2, S_{\boldsymbol{\theta}}(D1))$ and evaluate its RMSE and MAP@20 on $D2_3$ and so on, until we have evaluated all twelves $D2_m$. The hyperparameter set remains unchanged throughout the whole process. **NewDataNewParams:** The third approach, as outlined in Algorithm 1, also starts by selecting the same set of hyperparameters, i.e. $S_{\boldsymbol{\theta}}(D1)$, from $\boldsymbol{\theta}$. Again, a predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built. Similar to the second approach, we evaluate the RMSE and MAP@20 of this model on the *first month* of the second year, i.e. $D2_1$, only. Uniquely for this proposed approach, we do the hyperparameter optimization again at this stage. We try to find another set of hyperparameters from $\boldsymbol{\theta}$ that performs the best based on the first year data *plus* the data of the first month of the second year. This new hyperparameter set, $S_{\boldsymbol{\theta}}(D1 + D2_1)$, may be the same as $S_{\boldsymbol{\theta}}(D1)$ if it is still the best performing set. Then, we re-train the predictive model using $S_{\boldsymbol{\theta}}(D1 + D2)$ and the first year data *plus* the data of the first month of the second year, which becomes $M(D1 + D2_1, S_{\boldsymbol{\theta}}(D1 + D2_1))$. We evaluate the RMSE and MAP@20 of this updated model on $D2_2$. Similarly, we re-select the best hyperparameter set from $\boldsymbol{\theta}$ and evaluate the RMSE and MAP@20 of $M(D1 + D2_1 + D2_2, S_{\boldsymbol{\theta}}(D1 + D2_1 + D2_2))$ on $D2_3$ and so on, until we have evaluated all twelves $D2_m$.

For all these approaches, the selection of the best set of hyperparameters is done by evaluating RMSE or MAP@20, according on the final evaluation metrics, with k-fold cross-validation where $k = 10$. Also, techniques to handle implicit feedback data for ALS-WR and SGD discussed previously have been applied.

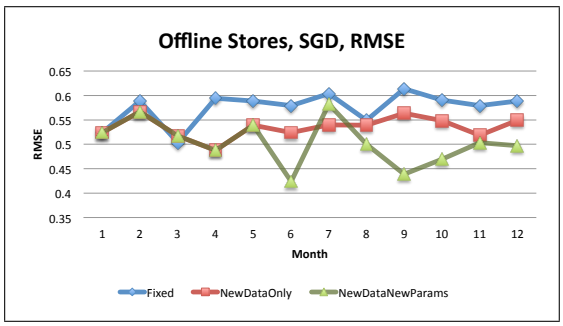
We want to observe two issues in particular:

- whether a hyperparameter set that doesn't perform the best at earlier month would become the best choice in later months
- compare the prediction accuracy of **Fixed**, **NewDataOnly** and **NewDataNewParams**

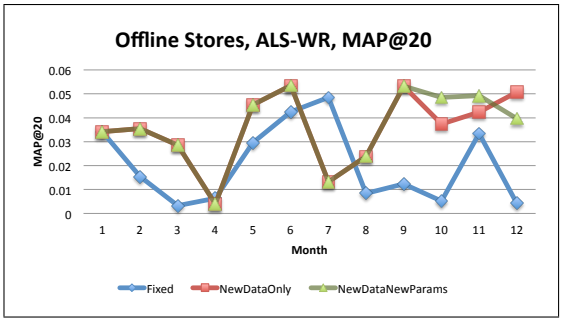
The results of the experiment are shown in Fig. 2a and 2c. This experiment is repeated with another collaborative filtering algorithm - SGD. For SGD, we define the width of the feature vectors for consumers and products as 20 and



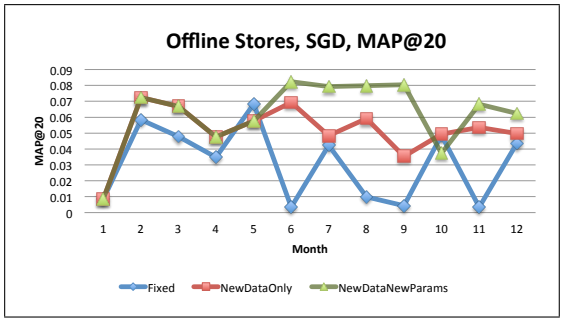
(a) RMSE on ALS-WR - 2nd year offline stores



(b) RMSE on SGD - 2nd year offline stores



(c) MAP@20 on ALS-WR - 2nd year offline stores



(d) MAP@20 on SGD - 2nd year offline stores

Figure 3: Results on Offline Stores Dataset

the maximum number of iterations as 30. By using random search technique, a pool of 1000 available hyperparameter sets is also generated for the three hyperparameters of SGD: the search range of λ is [0.001, 0.01], the range of $lrate$ is [0.001, 0.01] and the range of $decay$ is [0.1, 1]. The results are shown in Fig. 2b and 2d.

At the end, the whole experimental setup is re-run again for the purchase transaction dataset of offline stores. The results are shown in Fig. 3a, 3b, 3c and 3d.

D. Effectiveness on the Online Site

Fig. 2a shows the RMSE evaluation of the consumer preference prediction using ALS-WR for the second year of the online e-commerce dataset. The *NewDataOnly* approach clearly outperforms the *Fixed* baseline approach in every month. It means that re-training the predictive model with new data every month helps to improve preference score prediction in this case. For the proposed *NewDataNewParams* approach, new hyperparameter sets are selected at month 5 and month 10, as shown in Table. III. The predictive model is therefore re-trained based on a new λ value from month 5 to month 9 and another new value from month 10 to month 12. During the period between month 5 to month 12, inclusively, *NewDataNewParams* outperforms *NewDataOnly* in every month except month 11. Fig. 2c shows the MAP@20 evaluation of Top-20 recommendation, i.e. MAP@20, for the same ALS-WR. As shown, the curve representing the Fixed approach fluctuates in a relatively large range while the curves of the other two approaches both appear to be relatively smoother. For the Fixed approach, the results of the top 20 recommendation perform badly on month 3, 8 and 12. The *NewDataOnly* approach improves the recommendation results in these and some other months significantly. It outperforms the *Fixed* baseline approach in 8 out of the 11 months after an exactly same starting in the first month. For the *NewDataNewParams* approach, it outperforms *NewDataOnly* every month from month 5 to month 9 - the period during which the predictive model is re-trained by the first new hyperparameter set. From month 10 to 12, when another new hyperparameter set is used, *NewDataNewParams* outperforms *NewDataOnly* in 2 out of 3 months.

	Month	λ
Online	1 to 4	0.054
	5 to 9	0.038
	10 to 12	0.033
Offline	1 to 9	0.062
	10 to 12	0.048

Table III: Selected ALS-WR Hyperparameters

Fig. 2b and 2d show the RMSE and MAP@20 evaluation of prediction using SGD respectively. For the proposed *NewDataNewParams* approach, a new hyperparameter set is selected at month 8, as shown in Table. IV. Similar to the results for ALS-WR, the *NewDataOnly* approach

outperforms the *Fixed* baseline approach in most months. Unlike ALS-WR, a new hyperparameter set is selected once only for the *NewDataNewParams* at month 8. Since then, the *NewDataNewParams* approach slightly outperforms the *NewDataOnly* approach in every month except at month 9 based on RMSE evaluation. In addition, the *NewDataNewParams* approach slightly outperforms the *NewDataOnly* approach in every month based on MAP@20 evaluation. The observations are similar to those under ALS-WR.

	Month	λ	lrate	decay
Online	1 to 7	0.009	0.004	0.98
	8 to 12	0.006	0.004	0.95
Offline	1 to 5	0.006	0.005	0.91
	6 to 12	0.005	0.001	0.78

Table IV: Selected SGD Hyperparameters

In this experiment for the online e-commerce site, there are two main observations: Firstly, a hyperparameter set that is not the best choice at the beginning may become the best choice as more data is being collected over time; Secondly, conducting hyperparameter optimization continuously as more data is being collected improves prediction accuracy in general.

E. Effectiveness on Offline Stores

Fig. 3a shows the RMSE evaluation of the consumer preference prediction using ALS-WR for the second year of the offline stores dataset. Same as the result for the online e-commerce dataset, the *NewDataOnly* approach clearly outperforms the *Fixed* baseline approach in every month. For the proposed *NewDataNewParams* approach, a new hyperparameter set is not selected until month 10 and it is the only time a new hyperparameter set is selected, as shown in Table. III. The predictive model is therefore re-trained based on a new λ value from month 10 to month 12. In these 3 months, *NewDataNewParams* always outperforms *NewDataOnly* . Fig. 3c shows the MAP@20 evaluation of Top-20 recommendation for ALS-WR. As shown, the curve representing the Fixed approach fluctuates in a relatively large range while the curves of the other two approaches appear to be relatively smoother. Unlike the online e-commerce environment, the *NewDataOnly* approach and the *NewDataNewParams* approach do not seem to have smoothen the curve when comparing to the *Fixed* approach under the offline stores environment. Both approaches outperform the *Fixed* approach in all months except month 4 and month 7. For the *NewDataNewParams* approach, a new hyperparameter set is selected at month 10. It outperforms *NewDataOnly* on month 10 and 12 but underperforms on month 11. The advantage of *NewDataNewParams* over *NewDataOnly* is relatively uncertain in this case.

Fig. 3b and 3d show the RMSE and MAP@20 evaluation of prediction using SGD respectively. For RMSE evaluation, the *NewDataOnly* approach outperforms the *Fixed* baseline approach in all months except month 3. A new hyperparameter set is selected once only for the *NewDataNewParams* at month 6, as shown in Table. IV, which is much earlier than the case of ALS-WR. Since then, the *NewDataNewParams* approach outperforms the *NewDataOnly* approach in every month except at month 7 based on RMSE. For MAP@20 evaluation, the *NewDataOnly* approach outperforms the *Fixed* baseline approach significantly in all months except month 5. After the predictive model is re-trained with the newly selected hyperparameter set at month 6, the *NewDataNewParams* approach outperforms the *NewDataOnly* approach significantly, except in month 10 where *NewDataNewParams* performs even worse than the baseline. In this experiment for the offline stores of the retail business, the general outcomes are similar to those of the online e-commerce site. The major difference is that a new hyperparameter set for ALS-WR is not selected until a much later stage in the offline stores environment. Because of this, the advantage of the *NewDataNewParams* approach over the *NewDataNewParams* approach is not conclusive in this single scenario. One possible reason is that the business dynamics or consumer preference does not change much in a year in the offline stores environment, therefore a new model selection is not necessary until a later time. Another possible reason, however, is that the hyperparameter optimization technique we apply is not efficient enough to find an optimal hyperparameter set that can improve the performance earlier. Identifying the real reason may become part of our future work.

V. Related Work

In this section, we review the related work, which can be categorized into two parts: collaborative filtering and hyperparameter optimization.

Collaborative Filtering. The term CF was started by the first CF recommender system - Tapestry [17]. Since CF approach requires no domain knowledge and it is generally more accurate due to the power of uncovering hidden patterns, it attracts a lot of attention from the research community [18], [19] and it is widely used in commercial systems [1]. There are two primary techniques for CF to build predictive models using previous behavioral data, they are: the neighborhood approach and latent factor models. Neighborhood approach, such as item-based algorithm [3], estimates the relationships between product or consumer pairs, which are used for making prediction. Latent factor models, such as Singular Value Decomposition (SVD), estimates feature vectors that represent product and consumer profiles, which make them directly comparable. Latent factor models tend to be more accurate than neighborhood models [11]. The two widely used scalable algorithms of latent factor models are SGD [6], [11] and ALS-WR [5], which are implemented in the experiments of this paper.

Automatic Hyperparameter Optimization. Almost all collaborative filtering algorithms come with one or more adjustable hyperparameters, such as regularization and learning rate. The setting of these values plays a key role on the generalizability and accuracy of the predictive model [20]. The problem of finding better hyperparameter values for an algorithm to improve prediction accuracy or to optimize certain goals is called hyperparameter optimization or model selection. This problem has been studied in multiple disciplines in the past, for instance, response surface methodology [21] is a popular method for tuning parameters in statistics while in Machine Learning, advanced methods to optimize hyperparameters have been proposed, such as the use of Gaussian Process [22], Random Forests [23], Tree-structured Parzen Estimator [24] and Reinforcement learning [25]. On the other hand, two simple techniques are widely used in commercial industries, they are: grid search and random search [8]. Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. Random search, on the other hand, selects a value for each hyperparameter independently using a probability distribution. Existing literature in collaborative filtering, or in recommender systems in general, handle hyperparameter settings by trial and error manually. While this approach may be acceptable if we need to conduct hyperparameter optimization only once at the initial stage, it is impractical when we want to conduct continuous hyperparameter optimization when new data comes it repeatedly. We, therefore, apply random search as a baseline hyperparameter optimization technique on collaborative filtering algorithms in this paper.

VI. Conclusion and Future Work

In this paper, we investigate whether re-selecting hyperparameters for model selection repeatedly after the introduction of new data would have any impact on the prediction accuracy of recommender systems. Previous literatures assume that the process of hyperparameter optimization, or model selection, is needed only once at the initial stage. Therefore, the widely accepted approach nowadays is to re-train the predictive model periodically using the *same* hyperparameter settings after new data is collected.

The main contribution of the paper is to present a novel study on the effect of optimizing hyperparameters continuously. Three approaches have been evaluated: Firstly, it is $Fixed$, which is a baseline approach that does not re-train the predictive model at all. Secondly, it is $NewDataOnly$, which is the existing state-of-the-art approach used by researchers and practitioners in the industry. For this approach, the predictive model is re-trained periodically with new data. However, the hyperparameter settings are fixed once they are defined at the initial stage. Thirdly, it is our proposed $NewDataNewParams$, which tries to optimize the hyperparameter settings every time before the predictive model is re-trained with new data. We have shown that the proposed continuous hyperparameter optimization approach, i.e. $NewDataNewParams$, improves prediction accuracy of a recommendation system most of the time empirically. In particular, experiments show that the improvement has been achieved in both scalable collaborative algorithms (ALS-WR and SGD) that we have implemented. Experiments also show that the proposed approach improves prediction accuracy of recommender systems for both online e-commerce site and offline retail stores of the retail chain business that we have investigated.

We conclude that, in our retail scenarios, not only does our proposed approach outperforms the baseline approach $Fixed$ significantly, it also outperforms the existing state-of-the-art approach $NewDataOnly$: For experiments on the online e-commerce dataset, $NewDataNewParams$ has an average of 15.8% and 22.6% improvement over $NewDataOnly$ in terms of MAP@20 for ALS-WR and SGD respectively, with a maximum improvement of 51.9% and 139.5% respectively on the best months. $NewDataNewParams$ also has an average of 1.9% and 2% improvement over $NewDataOnly$ in terms of RMSE for ALS-WR and SGD respectively, with a maximum improvement of 7.2% and 8.7% respectively on the best months. For experiments on the offline stores dataset, $NewDataNewParams$ has an average of 2.1% and 22.65% improvement over $NewDataOnly$ in terms of MAP@20 for ALS-WR and SGD respectively, with a maximum improvement of 30% and 126.7% respectively on the best months. $NewDataNewParams$ also has an average of 1% and 5.6% improvement over $NewDataOnly$ in terms of RMSE for ALS-WR and SGD respectively, with a maximum improvement of 4.5% and 22.1% respectively on the best months. We have also discovered that a hyperparameter set that is not selected at the initial stage may become the best choice at a later time.

This paper presents a new direction to improve the prediction performance of large-scale recommender systems in real-world retail scenarios. It is worth mentioning that a new hyperparameter set can only be selected in much later iteration in some situations in our experiments. Further work needs to be done to determine whether it is caused by the inefficiency of the selected automatic hyperparameter tuning technique, or that it is due to some limitations of the continuous hyperparameter optimization approach in certain cases. Future work should also consider the use of other automatic hyperparameter optimization techniques that are more efficient and less expensive computationally.

References

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76 – 80, jan/feb 2003.

[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The youtube video recommendation system," in *Proceedings of the fourth ACM conference on Recommender systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 293–296. [Online]. Available: http://doi.acm.org/10.1145/1864708.1864770

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01*, 2001.

[4] R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '07. New York, NY, USA: ACM, 2007, pp. 95–104. [Online]. Available: http://doi.acm.org/10.1145/1281192.1281206

[5] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proc. 4th Intíl Conf. Algorithmic Aspects in Information and Management, LNCS 5034*. Springer, 2008, pp. 337–348.

[6] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 69–77. [Online]. Available: http://doi.acm.org/10.1145/2020408.2020426

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Mar. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188395

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

[10] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, Jun. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1577069.1577091

[11] Y. Koren, "Factorization meets the neighborhood: a multi-faceted collaborative filtering model," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

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

[13] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, 2008, pp. 502–511.

[14] H. Dongyuan and C. Xiaoyun, "Tuning svm hyperparameters in the primal," in *Computational Intelligence and Natural Computing Proceedings (CINC), 2010 Second International Conference on*, vol. 1, 2010, pp. 201–204.

[15] J. Acevedo, S. Maldonado, P. Siegmann, S. Lafuente, and P. Gil, "Tuning l1-svm hyperparameters with modified radius margin bounds and simulated annealing," in *Proceedings of the 9th international work conference on Artificial neural networks*, ser. IWANN'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 284–291. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768409.1768448

[16] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 5–53, January 2004. [Online]. Available: http://doi.acm.org/10.1145/963770.963772

[17] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[18] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/421425

[19] D. Militaru and C. Zaharia, "A survey of collaborative filtering-based systems for online recommendation," in *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, ser. ICEC '10. New York, NY, USA: ACM, 2010, pp. 43–47. [Online]. Available: http://doi.acm.org/10.1145/2389376.2389383

[20] P. M. Rasmussen, L. K. Hansen, K. H. Madsen, N. W. Churchill, and S. C. Strother, "Model sparsity and brain pattern interpretation of classification models in neuroimaging," *Pattern Recognition*, vol. 45, no. 6, pp. 2085 – 2100, 2012, brain Decoding. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320311003906

[21] C. Weihs, K. Luebke, and I. Czogiel, "Response surface methodology for optimizing hyper parameters," Technical Report/Universität Dortmund, SFB 475 Komplexitätsreduktion in Multivariaten Datenstrukturen, Tech. Rep., 2006.

[22] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *arXiv preprint arXiv:1206.2944*, 2012.

[23] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[24] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms," *CoRR*, vol. abs/1208.3719, 2012.

[25] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," *Applied Optimization*, vol. 86, pp. 523–544, 2003.