# Beyond Collaborative Filtering: The List Recommendation Problem

Oren Sar Shalom*
Bar Ilan University and
IBM Research

Noam Koenigstein
Microsoft

Ulrich Paquet
Microsoft

Hastagiri P. Vanchinathan*
ETH Zürich

## ABSTRACT

Most Collaborative Filtering (CF) algorithms are optimized using a dataset of isolated user-item tuples. However, in commercial applications recommended items are usually served as an ordered list of several items and not as isolated items. In this setting, inter-item interactions have an effect on the list's Click-Through Rate (CTR) that is unaccounted for using traditional CF approaches. Most CF approaches also ignore additional important factors like click propensity variation, item fatigue, etc. In this work, we introduce the *list recommendation* problem. We present useful insights gleaned from user behavior and consumption patterns from a large scale real world recommender system. We then propose a novel two-layered framework that builds upon existing CF algorithms to optimize a list's click probability. Our approach accounts for inter-item interactions as well as additional information such as item fatigue, trendiness patterns, contextual information etc. Finally, we evaluate our approach using a novel adaptation of Inverse Propensity Scoring (IPS) which facilitates off-policy estimation of our method's CTR and showcases its effectiveness in real-world settings.

## Categories and Subject Descriptors

H.5 [**Information systems**]: Information retrieval—*retrieval models and ranking, retrieval tasks and goals*

## Keywords

Collaborative Filtering, Click prediction

## 1. INTRODUCTION AND MOTIVATION

The Netflix Prize competition [1] was a seminal event in recommender systems research. The original set-up of the competition set the stage for much following research. Over the years, additional directions have been explored such as implicit feedback [9], the cold-start problem [21], rank optimization [19, 23, 30] etc. Yet, the original setting of optimizing for isolated user-item tuples still dominates contemporary recommender systems research. In this work we try to break away from this paradigm and consider a structured approach to investigate and model the more complex dynamics of the *list recommendation* problem.

The list recommendation problem can be defined as follows: Given a specific user $u$ at time $t$ our goal is to produce an *ordered* personalized list of $K$ items $l_{ut} = \{i_1, i_2, \ldots, i_K\}$ that maximizes the probability that $u$ will click on an item from $l_{ut}$. Note that we assume that the same set of recommended items may yield different click probabilities when ordered differently, hence this formulation is different than that of the set recommendation problem (unordered lists). It is also different from learning to rank approaches for collaborative filtering that train on pairwise relations but ignore deeper interactions between items.

In Section 2, we review in detail the main prior work related to the list recommendation problem. The approach presented in this paper has several key advantage points over prior work: First, as we show in Section 4, the expected CTR of a list $l_{ut}$ depends on many complex factors beyond the user-item ratings such as diversity, item fatigue, popularity trends, contextual factors, the list layout on the screen, and many others. Prior approaches have considered some of these factors individually, but in this work we consider all of them jointly accounting for possible interactions. Second, many previous studies focused on the diversity factor assuming that a trade-off exists between accuracy and diversity. In contrast, we propose a structured approach in which diversity is optimized alongside the overall accuracy. Finally, many ranking approaches for collaborative filtering attempt to rank the entire catalog of items whereas in our formulation we optimize only for a list of $K$ items presented to the user. The number of items ($K$) is typically small and therefore we can consider the specific position of each item in the list. We do not assume that the list order implies an order on the quality or fitness of the items. Namely, we do not assume that the first item is better than the second item which in turn better than the third item etc. Instead, we learn the optimal ordered combination of items based on the individual characteristics of each item, its position in the list and interactions with other items. For example, in Section 4 we demonstrate a surprising pattern for the interaction be-

*This work was done while at Microsoft R&D Center in Israel.


*WWW 2016,* April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883057.

tween item-ratings (accuracy) and diversity that varies with the item's position in a list.

In this paper we treat the classical collaborative filtering problem of predicting ratings for user-item tuples as a "solved" problem and look beyond it at the list recommendation problem. We assume we already have a collaborative filtering algorithm implementation (in our case the Xbox recommender [11, 17]) that produces accurate predictions to user-item ratings. We utilize this existing algorithm and treat its predictions as features for a "second layer" of learning aimed at optimizing the list of items to be presented to the user. Hence, our set-up is a two-layered learning approach in which the collaborative filtering algorithm is trained in the first layer to serve as features (together with additional information) for the second layer. This second layer can potentially use a very different dataset from the first layer. For example, the Xbox recommender system uses a dataset of purchase signals (implicit binary ratings) to learn a matrix factorization model as our first layer [17]. Then, the second layer solves a classification problem based on a dataset of list impressions in which the binary label indicates whether a user clicked on an item from a list.

Our ultimate goal is to optimize the lists' Click-Through Rate (CTR). Therefore, one may question the need for different datasets in the first and second learning layers. Based on our experience, collaborative filtering algorithms such as matrix factorization are very effective in learning interests or "taste", yet they are less useful in optimizing for CTR. This two-layered set-up effectively extracts the relevant information from the first dataset in order to improve predictions in the second one. Furthermore, many commercial companies already possess the first collaborative filtering layer and employ different heuristics to generate recommendation lists. We chose a two-layered approach for practical reasons as it builds upon those existing systems already in production while proposing a structured approach to compose the final recommendation lists.

The contributions of this paper are threefold: (1) We introduce the *list recommendation* problem and investigate several factors that affect lists' CTR using insights from the Xbox recommender system. (2) We propose a simple yet effective two-layered approach that builds upon existing collaborative filtering solutions (first layer) in order to find the optimal list to each user (second layer). As explained in Section 2, our approach is novel and differs from previous research by considering the entire recommendation list as unit. (3) In Section 6.2, we propose a novel modification of Inverse Propensity Scoring (IPS) [25] for off-policy evaluation. Using this evaluation technique, we empirically illustrate the benefits of our approach and estimate a CTR lift of up to 3x in Xbox's main dash and up to 2.7x in Xbox's recommendations dash.

## 2. BACKGROUND AND RELATED WORK

**Collaborative Filtering** (CF) techniques have been used widely in recommender systems. One of the key developments in CF research are **Matrix Factorization** (MF) methods [12] which proved their usefulness in competitions such as the Netflix Prize [1], KDD Cup'11 [6] and others. Xbox recommendations also relies upon a MF algorithm [11, 17] which is used in this paper for the "first layer" of our method.

The list recommendation problem is related to the top-$K$ [3] optimization problem. However, most work has been concerned with ranking techniques borrowed from search problems, while work focused specifically on recommending lists has been sparse. In the context of recommender systems, **learning to rank** approaches have been applied mostly as extensions to MF models [19, 23, 24]. These approaches generally modify the loss function of the MF model to fit a ranking task. However, they maintain the "traditional" paradigm in which isolated user-item tuples are ranked one against the other rather than considering the entire recommendation list as a target for optimization. Hence they neglect many relevant factors such as inter-item interactions or contextual information. Furthermore, in recommendation scenarios it is possible that the best item should not be placed in the first position but rather in a different location depending on the specific layout. For example, in horizontal recommendation lists (e.g., Netflix) it is likely that the best items should be placed in the middle of the list rather than at the edges. In this paper we learn to optimize an ordered combination of $K$ items without ranking items one against the other.

This work is also related to different previous studies on **click prediction**. Most studies were focused on either sponsored search results or advertisement [2, 8, 15, 22]. At the heart of this paper is in fact a click prediction model designed specifically for CF recommender systems such as the Xbox recommender system.

Some prior work on **list recommendation** incorporate fatigue, temporal features and diversity independently [13, 18, 31, 35]. However, these critical building blocks of recommendation list optimization show complex interaction amongst themselves and therefore should be modeled together. For example, in the case of list diversity we provide a real-world account from Xbox showing that the benefits of diversity strongly depend on the user's preferences (Section 4.1).

A plethora of studies discusses what is known in literature as the *accuracy vs. diversity trade-off* [29, 34]. For example, Jambor and Wang [10] stated that "accuracy should not be the only concern" in a recommender system and applied constrained optimization to promote long tail items. Zhang et al., [33] performed a user study showing that introducing novelty and serendipity into music recommendations while limiting the impact on accuracy can increase user satisfaction. Rodriguez et al., [20] used A/B testing to show that a multi-objective optimization approach for LinkedIn's *TalentMatch* system can increase reply rate by 42%. In contrast to these studies and others, this paper does not address the problem in terms of balancing a multi-objective trade-off. Instead, we present a structured approach to learn the right amount of diversity from data in a manner that effectively improves our ultimate utility which is the system's CTR.

Finally, recommender systems can be viewed as online learning tasks with partial feedback in which **multi-arm bandit** algorithms can be applied. Previous papers focused mainly on the domains of online advertising and recommending news articles [14, 26, 27] where it is critical to balance between exploration and exploitation. Top-$K$ recommendations via contextual bandits were studied in [28] and shown to be superior to learning-to-rank approaches. However, in this work we present a principled approach that utilizes traditional collaborative filtering algorithm and are not concerned with the explore-exploit dilemma.

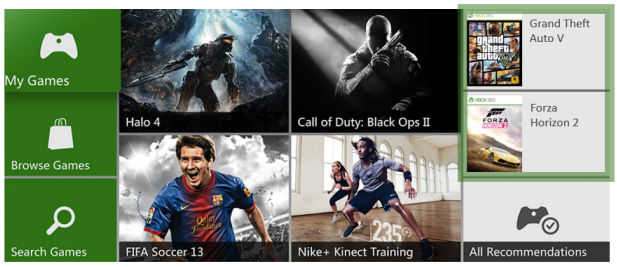



**Figure 1: Xbox 360 Main Dash - a vertical list of 2 items is highlighted on the right (in green).**

## 3. DATASET

As discussed in Section 1, our two-layered approach uses different datasets for each learning layer. The first layer is based on a one-class collaborative filtering model which is trained using purchase data from tens of millions of Xbox users. The model and dataset for the first layer are described in [17]. The predictions of the first layer are used as features for the second learning layer which is geared towards learning recommendation lists and trained using clicks collected from Xbox.

Figure 1 depicts the Xbox 360 main dash[1]. A list of two personalized items are presented on the upper right hand side. We call this list the *main dash* list. Below the main dash list, the user can find the *all recommendations* button which takes the user to the *recommendations dash*. In the recommendations dash, the user is presented with a horizontal list of 20 items. Both lists are refreshed before each impression by heuristically sampling a new diverse set of items using the predicted item scores from the first layer. In this paper we propose to replace these heuristics using a second layer based on supervised learning.

Our datasets are composed of impressions and clicks from the main dash and recommendations dash collected during one month in Q4, 2014. The total number of data points collected in this period is much larger than the datasets used for this paper. We removed users with more than 100 impressions and further "diluted" both datasets by randomly sampling users. The main dash dataset is composed of a sample of $5,339,456$ impressions served to $1,406,470$ users. We reserved the last (chronologically later) 10% impressions for an evaluation test-set. Similarly, for the recommendations dash dataset we used a sample of $6,319,843$ impressions served to $1,423,640$ users and reserved the last 10% for evaluation. Finally, we created binary supervised datasets by associating a positive label to impressions which resulted in a click (on any item) and a negative label with impressions that did not yield a click.

The characteristics of these two datasets are very different: The main dash list is a vertical list of just two items. It is an example of a *passive* scenario in which the user hasn't proactively requested recommendations. Instead, she is presented with the list while she is most likely focused on a different area of the screen. In such passive scenarios, CTR is typically very low and therefore we created a balanced dataset by uniformly sampling the negative impressions for each user. In contrast, the recommendations dash dataset is an example of an *active* scenario in which the user has actively requested recommendations. This yields much higher CTR levels, hence no dataset balancing was required. This dataset also differs in its presentation (horizontal instead of vertical) and list length (20 items as opposed to 3). In Section 7 we notice the influence of these differences on the algorithm's results.

[1]The dashboard's appearance changes with different update versions. This image captures its appearance at the time of data collection.

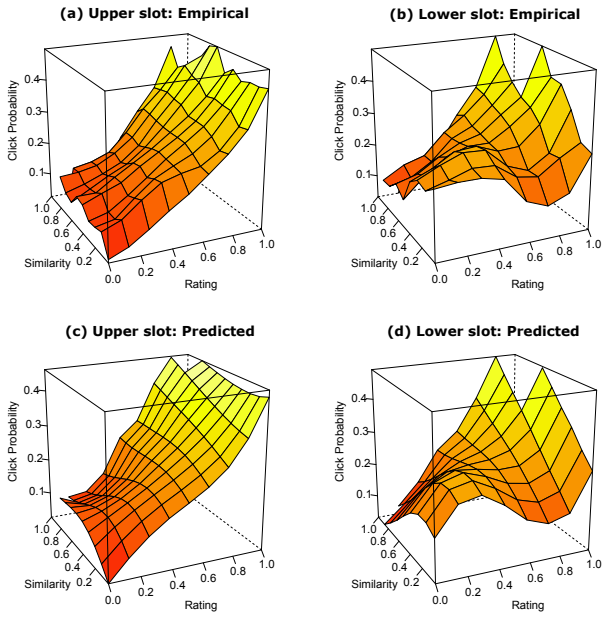



**Figure 2: Empirical and estimated click probability for items in the upper and lower slots on Xbox's main dash as a function of the predicted rating and similarity to the other item in the list.**

## 4. INSIGHTS FROM THE XBOX RECOMMENDER SYSTEM

In this section we present some insights from the Xbox recommender system that illustrate the possible shortcomings of contemporary collaborative filtering techniques.

### 4.1 Inter-item Similarity Interactions

Recommending a list of size $K$ is typically very different from ranking top-$K$ items. One key difference is the fact that items' relevance is not independent and diversity/similarity plays a significant role in determining the list's click probability. In contrast to many previous works that considers an "accuracy vs. diversity trade-off", we use data from the Xbox recommender system to show that the actual relationship between accuracy and diversity is, in fact, more complex.

Figure 2 depicts the click probability items in Xbox's main dash as a function of both their rating and the Jaccard similarity to the other item presented in the same recommendation list. Sub-figures (a) and (b) depict the empirical click probability surfaces for the items in the upper and lower slots respectively. Sub-figures (c) and (d) depict the predicted probability surfaces for the upper and lower slots respectively using a simple logistic regression model with a

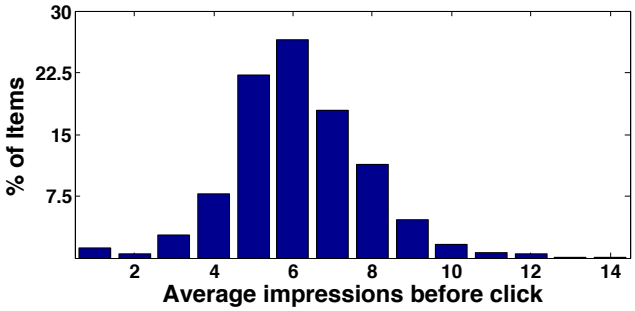



**Figure 3: A histogram of the number of impressions before a click event.**

third degree polynomial kernel. For the upper slot we see a clear positive correlation between the predicted rating coming from the MF model and the click probability, however there seems to be very little relevance to the similarity with second item below it. This is not the case with the lower item where a more complex pattern is revealed: For high predicted ratings, it is best to *not* diversify and recommend "more of the same", but if the predicted rating is low it is better to "hedge bets" by diversifying the list. We explain the difference between the upper and lower items by considering the specific layout of the main dash recommendation list (Figure 1). The typical user probably notices the upper item before the lower one. Hence, the upper item's click probability is independent of the lower item but the lower item's click probability is highly dependent on what is shown above it.

The Xbox recommender is based on a Bayesian model that models the posterior probability of a user to purchase an item [17]. These probabilities capture both preferences and uncertainty. From Figure 2 we learn that if the system has more certainty in user's preferences it is better to present a list with several similar items even if it means lower diversity. However, when the system's estimation of the user preference has less certainty, more clicks may be obtained by diversifying the items on the list. Based on these insights, the need to diversify varies and depends on both the predicted rating, the list's layout and each item's position.

## 4.2 Item Fatigue

Item fatigue occurs when a user is recommended the same item multiple times. After several impressions of the same item to the same user, the click probability decreases. In Xbox, we refresh the recommendation list prior to each visit, however some items are repeated multiple times. In the main dash there exists a noticeable effect of item fatigue on the overall CTR. In Figure 3, we demonstrate this effect by investigating the variation in the number of impressions before a click event is observed. For each click event, we counted the number of times the user was presented with that same item prior to clicking on it. We averaged these across all users and items and present the histogram of the number of impressions preceding the first click. The histogram exposes an interesting pattern: The item-fatigue is not always inversely correlated with the click probability. The histogram suggests that there is a threshold number of impressions which is required to maximize the click probability on an item. This threshold varies for different recommender systems and may even change across users and items. This insight indicates the need for a principled approach to account for item fatigue when optimizing for list recommendation.

| TimeSlot | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| % of Users | 26.98 | 25.61 | 23.62 | 19.23 | 18.42 | 12.51 |

**Table 1: Percentage of users with significantly higher CTR (compared to average CTR across whole day ) in various time slots. Note that a subset of users could feature under multiple time slots where others might not feature in any timeslot.**

## 4.3 CTR Variations by Time of Day

Click probabilities vary throughout the day and peak at different times for different users. Existing work in handling temporal dynamics for collaborative filtering either study long term temporal effects or use heuristics to optimize for session diversity [5, 12, 32]. In Table 1, we study the time-of-day click patterns of Xbox users. We split days into six time slots of four hours and consider the CTR per each time-slot. We count the number of users in each time-slot whose CTR for that time-slot is at least 30% higher than average. Note that for some users, there may be more than one time-slot in which their CTR is significantly higher than their average CTR. On the other hand, there could be another subset of users for whom there is no significant increase in CTR in any time-slot. Table 1 shows that many users have one or more preferred time-slots in which their consumption of recommendations are considerably higher than in other time-slots. This pattern can and should be exploited by recommender systems to optimize the timing and repetition of specific items.

# 5. METHODOLOGY

The insights described in Section 4 motivate a secondary supervised learning layer aimed specifically at optimizing CTR for recommendation lists. The datasets described in Section 3 form a classification problem consisting of list impressions and their consequent "click" or "no-click" labels. The second layer utilizes predictions from a first collaborative filtering layer together with many other potentially informative features in order to learn the click propensity of a list. Formally, we associate each impression in our dataset with a context feature vector denoted by $\mathbf{x}_{ul}^t$, where $u$, $l$ and $t$ represent the impression's user, list and time respectively. We denote by $y_{ul}^t$ the binary label, where $y_{ul}^t = 1$ means that user $u$ was presented with $l$ at time $t$ and clicked on at least one item from $l$, and $y_{ul}^t = -1$ means otherwise. Given a dataset $D$ of feature-label tuples $(\mathbf{x}_{ul}^t, y_{ul}^t) \in D$ we learn a predictor $h(\mathbf{x}_{ul}^t) = \hat{y}_{ul}^t$ that generalizes user-list temporal predictions denoted by $\hat{y}_{ul}^t \in [-1, 1]$. In Section 5.1 we explain this supervised model and in Section 5.2 we discuss how our contextual features $\mathbf{x}_{ul}^t$ are constructed. Finally, in Section 5.3 we describe an effective list recommendation policy, denoted by $\Phi$, which utilizes the $\hat{y}_{ul}^t$ predictions to construct and recommend lists to users.

## 5.1 Gradient Boosted Trees

Our secondary layer utilizes Gradient Boosted Trees (GBT) [7] to solve the list recommendation problem. GBT is an en-

semble method [4] which constructs a classifier

$$\hat{y}_{ul}^t = h_M(\mathbf{x}_{ul}^t) = \sum_{m=1}^{M} g_m(\mathbf{x}_{ul}^t) \quad (1)$$

by sequentially adding $M$ "weak" base estimators denoted by $g_m$. Formally, at stage $1 \leq m \leq M$ the full estimator is $h_{m-1} = \sum_{i=1}^{m-1} g_i$ and a new base estimator $g_m$ is added so that the new full estimator becomes $h_m = h_{m-1} + g_m$. The base estimators are chosen to fit the pseudo-residual

$$\sum_{(\mathbf{x}_{ul}^t, y_{ul}^t) \in D} \frac{\partial L(y_{ul}^t, h_{m-1}(\mathbf{x}_{ul}^t))}{\partial h_{m-1}(\mathbf{x}_{ul}^t)} ,$$

where $L$ is the binomial deviance loss function $L(y, f(\mathbf{x})) = \log(1 + \exp(-2yf(\mathbf{x})))$ as in [7]. Each base estimator is a tree that partitions the input space into $J$ disjoint regions. The number of base estimators $M$ and the number of leaves $J$ were determined by means of cross validation and set to $M = 98$ and $J = 16$.

In Section 4 we showed the existence of complex feature interactions as well as non-linearities with respect to the click probability. For the task of list recommendation, we choose the GBT algorithm because it can easily detect and exploit these patterns without the need to carefully craft a kernel. In Section 7, we illustrate this advantage by comparison of GBT to linear regression and Support Vector Machine (SVM) classifiers.

## 5.2 Feature Extraction

Recall that given a specific user $u$ at time $t$ our goal is to produce an *ordered* personalized list of $K$ items $l_{ut} = \{i_1, i_2, \ldots, i_K\}$. Our second layer learns the click propensity of a list $l_{ut}$ using a contextual vector $\mathbf{x}_{ul}^t$. Here we consider $\mathbf{x}_{ul}^t$ and the features used to construct it.

### *"First Layer" or Collaborative Filtering.*

Foremost of our features are the predicted ratings of the list's items as generated by our first layer (the collaborative filtering layer). We denote by $r_{uj}$ the predicted rating for a specific user $u$ on the item at position $j$. The first $K$ features in $\mathbf{x}_{ul}^t$ are the ratings to each item. Namely, $\mathbf{x}_{ul}^t = (r_{u1}, r_{u2}, \ldots, r_{uK}, \ldots)$. Note that in contrast to most ranking algorithms this formulation doesn't assume a descending order on the items and allows the model to learn any possible order. For example, for horizontal recommendation lists we may learn that the best items should be placed at middle of the list rather than at the beginning.

In this work the $r_{uj}$ values are the probabilities for the user to purchase the item given by a probabilistic matrix factorization model as explained [17]. Nevertheless, the approach in this paper is not limited to any specific type of CF model and can even benefit from integrating several different CF models. With these features the contextual vector $\mathbf{x}_{ul}^t$ grows linearly with $K$. Generally, the number of items that can be simultaneously presented to a user is limited and the list's length $K$ is not too long. In our case $K = 2$ for the *main dash* list and $K = 20$ for the *recommendations dash* list.

### *Similarity/ Diversity Features.*

As shown in Section 4.1, CTR depends on interaction effects between predicted item rating, item similarities and list position. We can learn these complex interactions by encoding item-to-item similarity features in $\mathbf{x}_{ul}^t$. In this paper we used the Jaccard similarity between item $i$ to item $j$ given by $sim_{i,j} = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$, where $U_i$ and $U_j$ are the sets of users that purchased items $i$ and $j$ respectively. The total number of item-to-item similarities grows quadratically in $K$, but using feature selection techniques we found that for our purposes, similarities between adjacent items in the list are sufficient. Hence, the number of features stays linear in $K$ and the next $K-1$ features in $\mathbf{x}_{ul}^t$ are: $\mathbf{x}_{ul}^t = (r_{u1}, \ldots, r_{uK}, sim_{1,2}, sim_{2,3}, \ldots, sim_{K-1,K}, \ldots)$.

### *Item Fatigue.*

The relationship between CTR and item fatigue is discussed in Section 4.2. In $\mathbf{x}_{ul}^t$ we employ two types of fatigue features: For a user $u$ and item $i$ at time $t$ we (a) count the number of times that $u$ has been exposed to $i$ in the previous week, denoted by $c_{ui}^t$, and (b) measure the time duration (in minutes) since $i$'s last impression to $u$, denoted by $m_{ui}^t$. The fatigue features for each item in $l$ are included in $\mathbf{x}_{ul}^t$ according to the item's position in a similar fashion to the previous features.

### *Trendiness and Temporal Features.*

Typically, there is an increased public interest in recently released items. This phenomenon can be attributed to external advertisement campaigns and word-of-mouth "viral" effects. We therefore denote by $d_i$ the number of days since item $i$'s release date and include these features in $\mathbf{x}_{ul}^t$ for each item in the list according to its position. Based on our experience, we also know that user behavior tend to change between weekdays and weekends [2]. Hence, we included additional $is\_weekend^t$ binary features that indicates whether time $t$ is a weekend or not. Finally, as discussed in Section 4.3, many users have a preferred time of day in which they are more likely to consume a recommendation. By dividing the day into six slots, each representing a period of four hours, we find users "favorite" time-slots as discussed in Section 4.3 and include a binary $is\_best\_timeslot_u^t$ feature for the user $u$ at time $t$.

### *User Features.*

We included features that aggregate a user's historical behavior by looking at the items the user clicked on in the past year and extracting user-specific features. With slight overloading of notation we denote by $d_u$ the average number of days since an item's release date and the date in which user $u$ clicked on it. Additionally, for each item $i$ in the list we denote by $pd_{iu}$ the price difference between item $i$ and the average price of items that $u$ has purchased in the past. We include $pd_{iu}$ for each item and $d_u$ in $\mathbf{x}_{ul}^t$.

### *Popularity Features.*

We also consider items' aggregate characteristics averaged over the past month. We denote by $ctr_i$ and $pc_i$, item $i$'s monthly average CTR and daily purchase count, respectively, and include these in $\mathbf{x}_{ul}^t$ for each item in the list.

## 5.3 List Recommendation Policy

So far we have described a method to estimate the click propensity $\hat{y}_{ul}^t$ of a list based on its characterizing feature vector $\mathbf{x}_{ul}^t$. We will now describe a *recommendation policy* which will utilize these predictions in order to choose the actual recommendation lists to be presented to users. A de-

terministic policy might present each user $u$ with the list that maximizes $\hat{y}^t_{ul}$ in Equation (1). However, this "exploitation" strategy lacks any freshness and may result in very similar lists being repeatedly presented to the user. The fatigue features will prevent the lists from being identical, yet we wish to facilitate more control over diversity and possibly some exploration. Hence, we define a stochastic list recommendation *policy* that utilizes the $\hat{y}^t_{ul}$ predictions in order to generate better recommendation lists. We denote our list recommendation policy by $\Phi$, and $\phi^t_u(l)$ denotes the probability that user $u$ will be presented a list $l$ at time $t$ according to policy $\Phi$. We denote Xbox's current production policy by $\Pi$, and $\pi_u(l)$ denotes the probability that user $u$ will be presented a list $l$ according to $\Pi$. Note that $\Pi$'s list probabilities are not time dependent.

We begin by defining a naive policy $\Phi$ using the Boltzmann distribution over the lists as follows: When user $u$ requires a recommendation list at time $t$, our policy $\Phi$ will sample a list $l$ from the set of all possible lists $L$ according to probabilities $\phi^t_u(l)$ given by

$$\phi^t_u(l) = \frac{e^{\alpha \hat{y}^t_{ul}}}{\sum_{k \in L} e^{\alpha \hat{y}^t_{uk}}}. \quad (2)$$

The non-negative parameter $\alpha$ controls exploitation with $\alpha = 0$ for uniform sampling and $\alpha \to +\infty$ for maximal exploitation.

According to the definition of $\Phi$ in Equation 2, each possible list (i.e., each combination of items) has some probability to be presented to the user given by $\phi^t_u(l)$. Therefore, this initial formulation extrapolates predictions over many unobserved list combinations. On the other hand, the production policy $\Pi$ employs several heuristics and business rules that filter many items and considers only a subset of lists for each user. For example, by design, $\Pi$ will never recommend a list containing individual items that we predict the user will dislike. This is one example of many similar business rules heuristically employed by $\Pi$ and optimized using online experimentations. Reviewing the business rules employed by our production policy $\Pi$ is sensitive and regardless it cannot be covered in the context of this paper due to space limitations.

It is clear that the extrapolation defined in Equation 2 includes many cases of bad recommendation lists. However, our training data is based on the policy $\Pi$ and doesn't have support over the entire feature space. Therefore, this extrapolation includes many unreliable predictions that should not be considered at all. Moreover, the extrapolation in Equation 2 prohibits any reasonable estimation of $\Phi$'s performance using the off-policy evaluation techniques that we develop in Section 6.2. Therefore, we constrain $\Phi$ to use similar heuristics as $\Pi$ which will prevent it from extrapolating predictions over lists that were never observed in the past. We denote by $L_u \subset L$ the subset of lists that user $u$ was exposed to by $\Pi$, and alter our definition of $\Phi$ to consider only lists in $L_u$ as follows:

$$\phi^t_u(l) = \begin{cases} \frac{e^{\alpha \hat{y}^t_{ul}}}{\sum_{k \in L_u} e^{\alpha \hat{y}^t_{uk}}}, & \text{if } l \in L_u \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Now, policy $\Phi$ is constrained to present each user a list from the same subset of lists as $\Pi$ would have shown her. This change serves two goals: First, it prevents the system from "drifting" away and suggesting poor recommendation lists due to unreliable extrapolation. Second, the off-policy evaluation technique we employ in Section 6.2 requires that the evaluated policy will have the same probability support over lists as the production policy. The definition in Equation 3 enables us to evaluate $\Phi$ using logs from $\Pi$.

# 6. OFFLINE VS. OFF-POLICY EVALUATION

We use classical *offline* evaluation techniques in order to evaluate the ability of the GBT model to generalize the second layer learning task. Offline evaluation is important in order to quantify and optimize the learning task, however it does not provide a method to estimate the potential effect on the system's CTR. Therefore, in Section 6.2 we present a novel adaptation of Inverse Propensity Scoring (IPS) [25] in order to estimate the CTR if policy $\Phi$ was to replace the production policy $\Pi$.

## 6.1 Offline Evaluation

As explained above, we utilize predictions from our "first layer" MF model together with additional features in order to create a contextual feature vector $\mathbf{x}^t_{ul}$. Our "second layer" uses these contextual vectors $\mathbf{x}^t_{ul}$ in order to predict if a list will be clicked or not. Our offline evaluation is based on comparing the Receiver Operating Characteristic (ROC) curves of different classifiers and measuring the Area Under the Curve (AUC). Results are presented in Section 7.1.

## 6.2 Off-Policy Evaluation

In order to evaluate a recommendation policy like $\Phi$, one may implement the policy and perform a controlled experiment (A/B testing). However, implementing a new recommender system on Xbox, as on any other real-world large scale system, is expensive and time consuming. Hence, we wish to have an evaluation of $\Phi$ using a dataset $D$ generated by the current production policy or *logging policy* $\Pi$. We present here a novel adaptation of Inverse Propensity Scoring (IPS) [25] designed to estimate $\Phi$'s improvement over $\Pi$ using off-policy techniques.

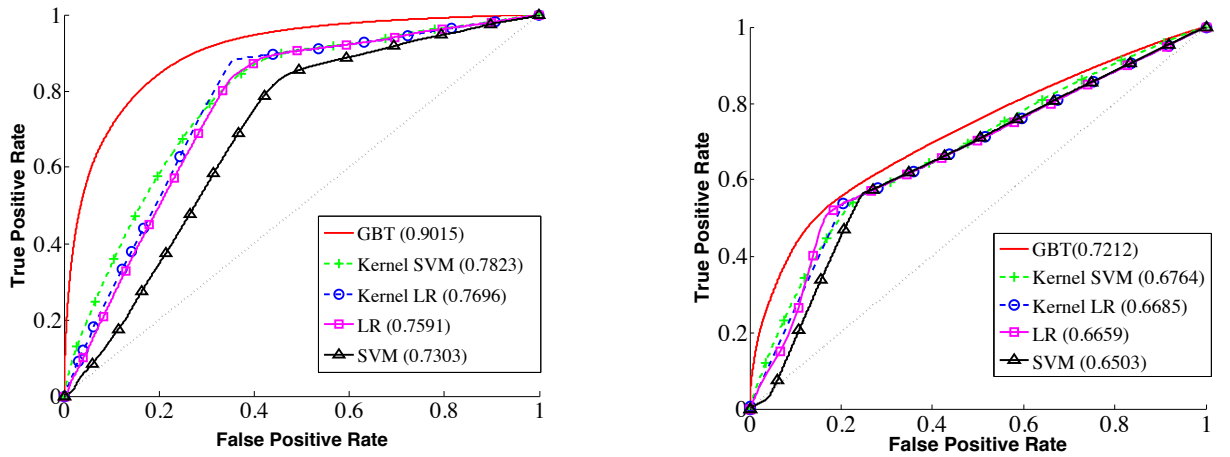
We denote by $\mathbb{E}^{ut}_{\Phi}[y^t_u]$ the expected CTR for user $u$ at time $t$ under policy $\Phi$. Note that $\Phi$ is a *stateful* policy i.e., the probabilities $\phi^t_u(l)$ depend on the user and change with time. We therefore wish to compute an expectation of $\mathbb{E}^{ut}_{\Phi}[y^t_u]$ over the time interval $[0, T]$. Let $A_u$ be a user impression event which occurs every time user $u$ appears in the system. We denote by $p(A_u|t)$ the probability of an impression $A_u$ given the time $t$. It can be shown that the expected CTR for user $u$ over $[0, T]$ is equivalent to the expected number of clicks in $[0, T]$ divided by the expected number of impressions:

$$\mathbb{E}^u_{\Phi}[y^t_u] \frac{\int_{t=0}^{T} p(A_u|t)\, \mathbb{E}^{ut}_{\Phi}[y^t_u]\, dt}{\int_{t=0}^{T} p(A_u|t)\, dt}. \quad (4)$$

Next, we estimate $\mathbb{E}^u_{\Phi}\left[y^t_u\right]$ as follows:

$$\mathbb{E}^u_{\Phi}[y^t_u] = \frac{\int p(A_u|t)\, \mathbb{E}^{ut}_{\Phi}[y^t_u]\, dt}{\int p(A_u|t)\, dt} = \frac{\int p(A_u|t)\, \mathbb{E}^{ut}_{\Pi}\left[\frac{\phi^t_u(l) y^t_{ul}}{\pi^t_u(l)}\right] dt}{\int p(A_u|t) dt}$$
$$= \mathbb{E}^u_{\Pi}\left[\frac{\phi^t_u(l) y^t_{ul}}{\pi_u(l)}\right] \approx \frac{1}{|D_u|} \sum_{y^t_{ul} \in D_u} \frac{\phi^t_u(l) y^t_{ul}}{\pi_u(l)}, \quad (5)$$

where $D_u$ are user's $u$ ratings in the dataset $D$.




(a) Main Dash (b) Recommendations Dash

**Figure 4: ROC Curves for different classification algorithms. The AUC values is noted in the legend. All AUC measures have significance levels (p-values) $\leq 0.0001$.**

The second equality in Equation 5 follows from:

$$\mathbb{E}^{ut}_{\Phi}[y^t_u] = \sum_{l \in L_u} \phi^t_u(l) y^t_{ul} = \sum_{l \in L_u} \pi_u(l) \frac{\phi^t_u(l) y^t_{ul}}{\pi_u(l)} = \mathbb{E}^{ut}_{\Pi}\left[\frac{\phi^t_u(l) y^t_{ul}}{\pi_u(l)}\right]. \quad (6)$$

Equation 6 is key for the off-policy evaluation as it allows replacing an expectation over $\Phi$ with an expectation over the logging policy $\Pi$. This equation is valid only if both policies have the same probability support over lists. By design this is indeed the case.

The third equality in Equation 5 follows from using the policy $\Pi$ in the definition given by Equation 4. It is true under the reasonable assumption that the probability for an impression event $p(A_u|t)$ is independent of the recommendation policy. Finally, the last equality in Equation 5 is simply an empirical estimate of the expectation $\mathbb{E}^u_{\Pi}\left[\frac{\phi^t_u(l) y^t_{ul}}{\pi_u(l)}\right]$.

*Relation to Inverse Propensity Scoring.*

Our estimate of $\mathbb{E}^u_{\Phi}[y^t_u]$ bears resemblance to the Inverse Propensity Scoring (IPS) of [25]. In fact, it can be considered as a specific case of IPS. However, despite reaching a very similar result, both papers employ different derivations in order to justify it. Moreover, our version differs from [25] in several other key aspects: First, in our case the logging policy $\Pi$ has sufficient support over all lists by way of design. When compared to the IPS description in [25], this means that we may drop the *max* operation in [25]'s estimator and use $\tau = 0$ (Equation 2 in [25]). Furthermore, we have familiarity of the logging policy and assume accurate estimates of its probabilities $\pi_u(l)$. Hence, the "regret" term given by Equation 5 in [25] is assumed to be zero and our estimates are guaranteed to be unbiased. In both versions the logging policy is time independent, however in [25] the evaluated policy is deterministic and in our case we evaluate a non-deterministic stateful policy. While this extension does not incur any estimation bias, we do not provide any bounds on its accuracy and convergence with respect to the dataset size. In fact, the per-user $\mathbb{E}^u_{\Phi}[y^t_u]$ estimates are potentially very noisy. However, as we show next, the final measures we report are stable and overcome this hurdle by averaging the $\mathbb{E}^u_{\Phi}[y^t_u]$ estimates of 1.4 million users.

# 7. RESULTS

We present here the offline and off-policy evaluation results of our method using the datasets from Xbox's main dash and recommendations dash.

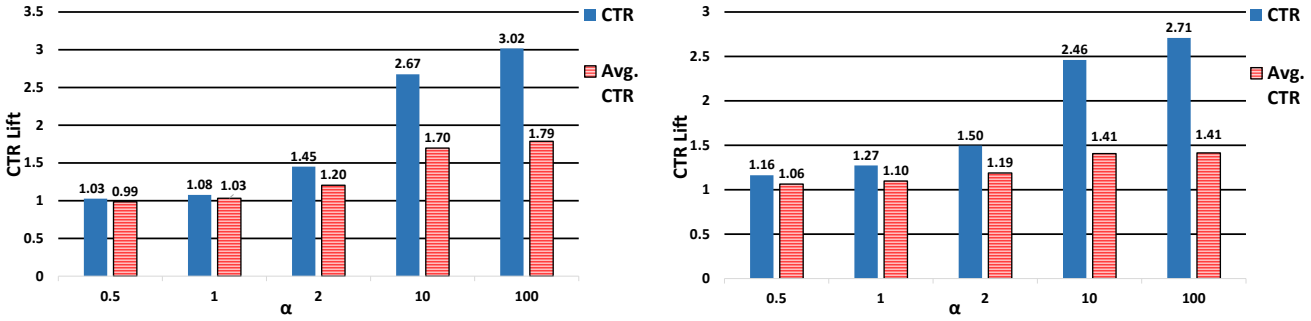
## 7.1 Offline Results

As explained in Section 5.1, we predict the click propensity $\hat{y}^t_{ul}$ from a context vector $\mathbf{x}^t_{ul}$ using a GBT classifier.We compare this classifier to baselines employing Logistic Regression (LR) and Support Vector Machines (SVM). In their basic form, these are linear models which are inferior to GBT in their ability to capture complex interactions in the input space. We therefore add an evaluation using a kernelized version of these baselines utilizing a polynomial kernel with degree $d = 4$.

Figure 4 depicts the ROC curves of the different algorithms. The corresponding AUC values are noted in the legend. In both the main dash and the recommendations dash GBT is the most appropriated for the prediction task at hand. The kernelized SVM algorithm has the best performance out of the baselines. In general, the kernelized version of each baseline always outperform the linear version. This implies the existence of non-linear complex interactions between features in the input space. The prediction task seems somewhat harder in the recommendations dash where AUC values are lower compared to the main dash. However as we show next, both the main dash and the recommendations dash can benefit dramatically by employing the method described in this paper which will yield a significant increase in CTR.

## 7.2 Off-policy Results

Our goal is to predict the CTR improvement of the Xbox system if we were to use $\Phi$ instead of $\Pi$. We estimate the overall CTR by aggregating the per-user estimates according




(a) Main Dash (b) Recommendations Dash

**Figure 5: CTR lift values for the main dash and the recommendations dash of this paper as a function of $\alpha$.**

to their relative share in the dataset as follows:

$$CTR_\Phi = \frac{1}{|D|} \sum_{u \in U} |D_u| \cdot \mathbb{E}^u_\Phi \left[ y^t_u \right], \tag{7}$$

where $U$ is the set of all users in our dataset $D$. The measure $CTR_\Phi$ is an estimate of the overall CTR using $\Phi$. Naturally, $CTR_\Phi$ emphasizes heavy users who drive more clicks than others. It is common practice in recommender systems research to define measures in which all users are weighted equally in the test-set regardless of their activity level. That was the case in many prominent competitions such as the Netflix Prize [1] and KDD Cup'11 [6]. We therefore define a secondary measure which measures the *average* CTR per-user:

$$Avg\_CTR_\Phi = \frac{1}{|U|} \sum_{u \in U} \mathbb{E}^u_\Phi \left[ y^t_u \right]. \tag{8}$$

We report results in terms of lift over our current production system: $\text{lift} = \frac{\text{Estimated CTR}}{\text{Production CTR}}$. As you may recall from Section 3, the main dash dataset is extremely unbalanced (very low CTR) and we balanced it by uniformly sampling the negative examples. In this evaluation we used the unbalanced version of the test-set in order to get a real estimate of the CTR which is also comparable to the current production CTR. We did not change our training-set and all previous results and discussions are still valid.

Figure 5 summarizes the CTR lifts on both datasets for different values of $\alpha$. As expected, the results improve with higher values of $\alpha$: At $\alpha = 0$ the lists are chosen uniformly and the lift values are lower than 1 indicating a reduction in CTR with respect to the production policy. At $\alpha = 100$ the system is at "full exploitation" mode and the best list (in terms of $\hat{y}^t_{ul}$) is almost certainly chosen. When comparing the main dash to the recommendation dash we see that the lift improvement in the main dash is much better. This is in accordance with the offline results as seen in Figure 4. Finally, we note that the overall CTR values exhibit higher improvement rates than the per-user values. Our training-set do not balance data instances per user, hence users with many ratings will have better rating predictions. This explains the higher values for the overall CTR.

### *Additional Baselines.*

Thus far we showed significant lift values for the approach presented in this paper when compared to Xbox's current recommender system. Xbox's system applies many heuristics tightly tuned to optimize its CTR and serves as the first baseline for this paper. Here, we wish to compare our approach with additional prominent approaches taken from previous studies. Due to the lack of relevant datasets, most previous works in the context of collaborative filtering try to heuristically balance between the predicted accuracy and other factors such as diversity [13, 35]. The current Xbox system also employs heuristics which are already finely tuned for CTR optimization. Therefore, it is very hard to show any significant CTR lift using other heuristic approaches. However, we found that approaches based on multi-objective optimization [10, 20] techniques can in fact show improvement on top of the current system.

The first baseline utilizes constrained multi-objective optimization. It attempts to maximize the predicted ratings subject to constraints on the relevant features. The features are the same as in the context vector $\mathbf{x}^t_{ul}$ except for the first $K$ rating features which are used in the objective of the optimization. Formally, for a specific user $u$ at time $t$, let $f^l_i$ be the $i$'th feature in $\mathbf{x}^t_{ul}$ after the first $K$ rating features. Namely, $\mathbf{x}^t_{ul} = (r_{u1}, r_{u2}, \ldots, r_{uK}, f^l_1, f^l_2, \ldots)$[2]. Our first baseline finds a list $l$ of $K$ items that optimizes the following objective:

$$\max_{l \in D_u} \sum_{k=1}^{K} r_{uk}$$
$$s.t. \quad \forall i: \quad T^{min}_i \leq f^l_i \leq T^{max}_i, \tag{9}$$

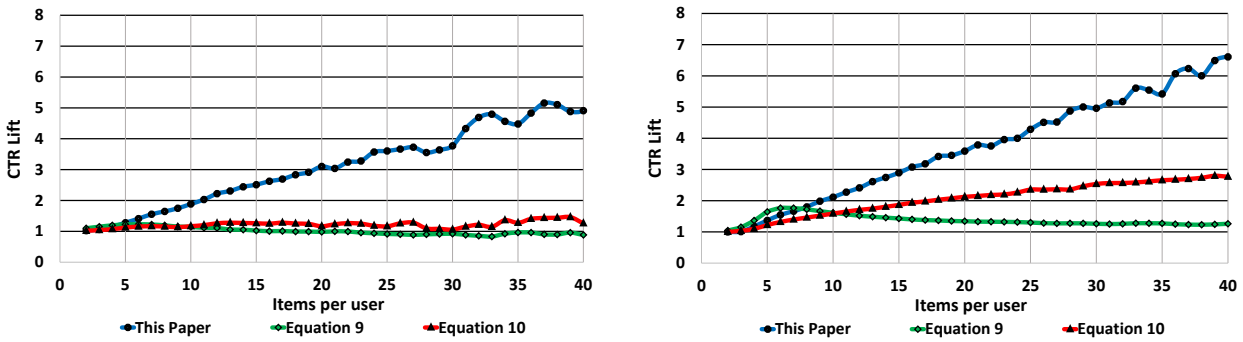
where $T^{min}_i$ and $T^{max}_i$ are minimal and maximal thresholds for the $i$'th feature. A second baseline is based on maximizing the predicted ratings as well as a weighted sum of the features as follows:

$$\max_{l \in D_u} \sum_{k=1}^{K} r_{uk} + \sum_i w_i f^l_i. \tag{10}$$

The threshold values $T^{min}_i$ and $T^{max}_i$ from Equation 9 and the weights $w_i$ from Equation 10 are parameters that were chosen to optimize the CTR using Nelder-Mead optimization [16].

Table 2 compares these baselines with the approach presented in this paper. Optimization based on Equation 10 consistently yields better results than than Equation 9. Nev-

[2]We suppress the indexes $u$ and $t$ from $f^l_i$ for brevity.




(a) Main Dash (b) Recommendations Dash

**Figure 6: A breakdown of CTR lift values for users with different number of purchased Xbox games.**

**Main Dash**

| | This Paper | Equation 9 | Equation 10 |
|---|---|---|---|
| $CTR_\Phi$ | 3.02 | 1.28 | 1.36 |
| $Avg\_CTR$ | 1.79 | 1.16 | 1.21 |

**Recommendations Dash**

| | This Paper | Equation 9 | Equation 10 |
|---|---|---|---|
| $CTR_\Phi$ | 2.71 | 1.13 | 1.19 |
| $Avg\_CTR$ | 1.41 | 1.06 | 1.08 |

**Table 2: Expected CTR and average CTR lift values using the approach proposed in this paper ($\alpha = 100$) vs. alternative approaches based on multi-objective optimization.**

ertheless, the approach presented in this paper outperforms both baselines. We observe that the main dash lift values are better than those of the recommendations dash and that the overall CTR values exhibit higher improvement rates than the per-user values. Both observations are consistent with those of Figure 5.

Finally, Figure 6 depicts a break down of the lift values for users with different number of purchased Xbox games (from the dataset used for the first layer). The approach presented in this paper outperforms the baselines in most cases with a clear trend of improvement as the number of items increase (i.e, for "heavy users"). The matrix factorization model is using purchases as its training signal hence its predicted ratings are more accurate for users with more purchases. Furthermore, "heavy users" with more games typically tend to spend more time in front of their consoles and click on more recommendations. Therefore, click events from these users are more common in our training-set for the second layer which translates to better lift values as discussed earlier.

## 8. CONCLUSIONS

This paper discusses the list recommendation problem and proposes a second learning layer on top of traditional CF approaches. The benefits of our approach are empirically illustrated using a novel modification of Inverse Propensity Scoring (IPS) . We believe these results showcase the need for recommendation list optimization and indicate that there is much room for improvement on top of current state of the art CF techniques.

## 9. REFERENCES

[1] J. Bennett and S. Lanning. The Netflix Prize. In *Proc. KDD Cup and Workshop*, 2007.

[2] Haibin Cheng and Erick Cantú-Paz. Personalized click prediction in sponsored search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 351–360, 2010.

[3] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 39–46, 2010.

[4] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, 2000.

[5] Gideon Dror, Noam Koenigstein, and Yehuda Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 2011.

[6] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The Yahoo! music dataset and KDD-Cup'11. In *Proceedings of KDDCup 2011*, 2011.

[7] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[8] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 13–20, 2010.

[9] Y. F. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*, 2008.

[10] Tamas Jambor and Jun Wang. Optimizing multiple objectives in collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 55–62, 2010.

[11] Noam Koenigstein and Ulrich Paquet. Xbox movies recommendations: Variational Bayes matrix factorization with embedded feature selection. In

*Proceedings of the 7th ACM Conference on Recommender Systems*, 2013.

[12] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, 2009.

[13] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, 2010.

[14] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.

[15] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1222–1230, 2013.

[16] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4), 1965.

[17] Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 999–1008, 2013.

[18] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[20] Mario Rodriguez, Christian Posse, and Ethan Zhang. Multiple objective optimization in recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, 2012.

[21] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 253–260, 2002.

[22] Benyah Shaparenko, Ozgur Cetin, and Rukmini Iyer. Data-driven text features for sponsored search click prediction. In *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, ADKDD '09, pages 46–54, 2009.

[23] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM Conference on Recommender Systems*, RecSys '12, pages 139–146, 2012.

[24] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 269–272, 2010.

[25] Alexander L. Strehl, John Langford, Lihong Li, and Sham Kakade. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems 23*, pages 2217–2225, 2010.

[26] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 73–80, 2014.

[27] Liang Tang, Romer Rosales, Ajit Singh, and Deepak Agarwal. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 1587–1594, 2013.

[28] Hastagiri P Vanchinathan, Isidor Nikolic, Fabio De Bona, and Andreas Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 225–232, 2014.

[29] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122. ACM, 2009.

[30] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alexander J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems 20*, pages 1593–1600, 2008.

[31] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–732, 2010.

[32] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010*, pages 211–222, 2010.

[33] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 13–22, 2012.

[34] T. Zhou, Z. Kuscsik, J.G. Liu, M. Medo, J.R. Wakeling, and Y.C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences (PNAS)*, 107(10):4511–4515, 2010.

[35] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, pages 22–32, 2005.