Explicit or Implicit Feedback? Engagement or Satisfaction?

A Field Experiment on Machine-Learning-Based Recommender Systems

Qian Zhao, F. Maxwell Harper, Gediminas Adomavicius, Joseph A. Konstan
University of Minnesota
Minneapolis, Minnesota, United States
{zhaox331,max,gedas,konstan}@umn.edu

ABSTRACT

Recommender systems algorithms are generally evaluated primarily on machine learning criteria such as recommendation accuracy or top-n precision. In this work, we evaluate six recommendation algorithms from a user-centric perspective, collecting both objective user activity data and subjective user perceptions. In a field experiment involving 1508 users who participated for at least a month, we compare six algorithms built using machine learning techniques, ranging from supervised matrix factorization, contextual bandit learning to Q learning. We found that the objective design in machine-learning-based recommender systems significantly affects user experience. Specifically, a recommender optimizing for implicit action prediction error engages users more than optimizing for explicit rating prediction error when modeled with the classical matrix factorization algorithms, which empirically explains the historical transition of recommender system research from modeling explicit feedback data to implicit feedback data. However, the action-based recommender is not as precise as the rating-based recommender in that it increases not only positive engagement but also negative engagement, e.g., negative action rate and user browsing effort which are negatively correlated with user satisfaction. We show that blending both explicit and implicit feedback from users through an online learning algorithm can gain the benefits of engagement and mitigate one of the possible costs (i.e., the increased browsing effort).

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

recommender systems, user-centric evaluation, user experiment, machine learning, contextual bandit, Q learning

ACM Reference Format:
Qian Zhao, F. Maxwell Harper, Gediminas Adomavicius, Joseph A. Konstan. 2018. Explicit or Implicit Feedback? Engagement or Satisfaction?: A Field Experiment on Machine-Learning-Based Recommender Systems. In *Proceedings of SAC 2018: Symposium on Applied Computing (SAC 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3167132.3167275

SAC 2018, April 9–13, 2018, Pau, France

ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/10.1145/3167132.3167275

1 INTRODUCTION

Machine-learning-based recommender systems are driven by user feedback data, e.g., explicit feedback data of ratings [4] and implicit feedback data of actions [11]. Typically, supervised learning models to predict rating values or action probabilities are trained in these systems based on the theory of empirical risk minimization [29], i.e., optimizing to reduce the prediction errors in historical training data with regularization to maintain generalizability.

In the history of recommender system research, there was a transition of trend from using explicit feedback data to implicit feedback data. The earlier pursuit of the Netflix Prize [4] was a (explicit-feedback) rating prediction problem. It greatly drove the progress of recommender system research. Amatriain and Basilico [3] in Netflix Blog later pushed back on researcher's focus on rating prediction, arguing that "accurate prediction of a movie's rating is just one of the many components of an effective recommendation system" and "using predicted ratings on their own as a ranking function can exclude items that the member would want to watch even though they may not rate them highly". They turned to a broad set of techniques to model the various types of user action data in the system to recommend items that the member is most likely to play and enjoy.

Until today, the difference between recommender systems built on explicit vs. implicit feedback data is not addressed in the research literature. This type of research inquiry is hard because offline metrics can not address it while live experiments require access to platforms that have real active users. With access to a movie information site that has thousands of real active users every month, we set out to compare the two types of recommenders in a live experiment based on the classical matrix factorization algorithms [16]. Being aware that there are many different algorithms proposed for modeling both explicit and implicit feedback data [11, 16, 21, 22], we chose the classical ones for better controlling the differences of algorithms while focusing on studying the differences that are inherent in the two types of feedback data. This comparison is possible also because the majority of the activities that users perform on the site is rating movies, which gives us access to not only implicit user actions but also abundant explicit ratings.

User-centric research in recommender systems [15] tends to focus on a broad set of factors and metrics that contribute to the success of recommender systems with the theoretic goal of supporting user decision making processes in front of a large amount of choices. Particularly, accuracy, diversity, novelty, serendipity, popularity, freshness and recency etc. [9, 15, 20, 33, 34] have all been studied in prior literature. Within this framework, accuracy of predicting ratings or actions only reflects part of the many factors

that are important. Previous research has shown that diversification, blending in popularity etc. on top of predictions help improve user engagement and experience [9, 34]. However, how exactly to blend multiple factors in to produce a final set of recommendations is tricky because there is no single ground-truth objective to target (e.g., the success of recommender systems is too abstract) to guide the blending process.

Adomavicius et al. [1] define the recommendation problem as a multi-criteria decision making problem (MCDM) and argue that the suitability of a recommendation for a particular user may depend on more than one utility-related aspect that the user takes into consideration when making the choice. Correspondingly in machine-learning-based recommender systems, it is typically modeled as a multi-objective optimization problem to take into account the multiple criteria [24]. However, the question that whether this approach can always help achieve an optimal solution across all criteria or whether it compromises some criteria while improving others is not yet fully understood. Online tuning in live systems is necessary to find the best combination weights but it usually takes long cycles to tune and hence is very expensive to follow.

In this research, we tried two new approaches in a live experiment to combine multiple factors in a principled way (also using machine learning techniques) inspired by the social theory of technology acceptance [30] and the reinforcement learning theory on decision making under uncertainty [13]. The first approach is to target user return (i.e., technology acceptance) as the objective to combine multiple factors. The second approach is to target online (on-policy in decision-making terms, see the related work section) user interaction following an online learning (specifically contextual bandit learning) algorithm.

The above motivation leads us to the following two research questions.

- *RQ1: What are the differences between recommender systems based on explicit vs. implicit user feedback data modeled with the classical matrix factorization algorithms?*
- *RQ2: Do multi-factor-blending recommendation algorithms lead to improved or changed user experience and if so, how?*

In the rest of the paper, we report on a live experiment involving more than 1.5K real users of a movie information site using six different recommenders for at least one month, measuring a broad set of user-centric metrics including objective user activities and subjective user perceptions. We found that

- a recommender based on a matrix factorization model minimizing (implicit) action prediction error engages users more (in terms of page views and interactions with the recommendations) than a matrix factorization model minimizing (explicit) rating prediction error, which empirically explains the transition from modeling explicit feedback data to implicit feedback data in recommender system research.
- the increased positive engagement is also associated with a significant amount of increase in user negative engagement (e.g., low ratings, clicking "not interested", browsing effort), likely because implicit feedback is noisier than explicit feedback about user preferences.
- blending both explicit and implicit feedback from users by targeting online (on-policy) user interaction through a contextual bandit learning algorithm can help gain the benefits of engagement and mitigate the possible cost, although it does not further significantly drive engagement.
- with our current design, targeting user return as the objective does not significantly affect user engagement (e.g., the actual future user return and churning risk) and shows a trend of hurting perception metrics compared with the baseline.

In the following sections, we first introduce the necessary background on user-centric research and machine learning techniques in recommender systems. The used techniques of this work span from classical matrix factorization (together with stochastic gradient descent) to contextual bandit learning (particularly the LinearUCB algorithm) and the Q-learning algorithm. Then we detail the method of this work and elaborate the online field experiment design. We show the results next and discuss the findings along with potentially promising future work. Lastly, we summarize this work's conclusion and contribution.

2 BACKGROUND OF USER-CENTRIC RESEARCH

User-centric research in recommender systems has been increasingly important. As pointed out by McNee et al. [18], recommendation accuracy on its own often is not a sufficient indicator of recommendation quality, and further work by Konstan et al. [15] elaborates the evolution of recommender system research from being concentrated purely on algorithms to research focused on the rich set of questions around the user experience with the recommender.

Several frameworks have been proposed and widely used by researchers to evaluate and understand user experience in recommender systems. For example, Knijnenburg et al. [14] proposed a comprehensive framework taking into account both objective system measurements and subjective user perceptions to explain user experience. McNee et al. [19] proposed an analytical process model called Human Recommender Interaction that acts as a bridge between user information seeking tasks and recommendation algorithms to help with the design and structure of recommender systems. Pu et al. [20] proposed a user centric evaluation framework by employing state-of-the-art survey designs structured and derived based on theories of human behavioral intention and reasoned action. Particularly relevant to our research, the theory of UT-AUT developed by Venkatesh et al. [30] postulates important social user factors that cause people to develop behavioral intention towards technologies and actual behavior of accepting or abandoning of the technologies. We are interested in studying whether this theory can be combined with machine learning, especially with reinforcement learning techniques, to optimize for user acceptance of recommender systems at scale. Work from Xiao et al. [32] is a direct application and further development of this theory in the domain of e-commerce recommender agents, e.g., highlighting the importance of trust, perceived ease of use, and perceived usefulness in determining the user intention of future use of the recommender agents.

3 BACKGROUND OF MACHINE LEARNING

To support explaining the recommenders we built in the experiment, this section sets up notations and gives a formal background on machine learning techniques used in this work. Note that this section is not meant to give an overview on how machine learning can be applied in recommender systems (See [2, 23] for a better review). Instead, it serves the purpose of motivating our research and explaining the perspective of approximating recommendation as a statistical learning problem.

Denote u as the representation of a user (e.g., basic profile, history interactions with the system) and c as the current context (e.g., time, location etc.) of a user entering a recommender system requesting item recommendations. Define $s = (a; c)$, i.e., s describes the the environment or state of both the user and the system. Denote a to be the action or decision that a recommender system needs to make. In the most simple case, a might be an item to recommend or a set of recommendations to present. More broadly however, it might incorporate the decision of how to present.

3.1 Empirical Risk Minimization

In the theory of supervised-learning-based recommender systems, it assumes that there is a y that represents u's preference under context c on a and it follows an unknown conditional distribution $P(y|s; a)$ (i.e., we focus on determinant statistical models here instead of generative models; see [5] for the difference). If we can reliably estimate this distribution for all possible s and a (e.g., sufficient observations are made) and fix our decision policy of making recommendations to always pick a with the largest $E(y)$, then the recommendation problem becomes the following stochastic optimization problem, where L an objective function measuring the loss or error of estimating y with a model (or function) f and $E_P(x)$ denotes the expectation of x with respect to the distribution P (also called the Statistical Decision Theory for supervised learning [7]).

$$f^* = argmin_f E_P(L(f, y)) \tag{1}$$

In reality, since we do not know the true P (assuming there exists such a P), the theory further assumes that the observed user feedback data are I.I.D samples of P. Since parametric models are popularly used as f in recommender systems, e.g., the widely used and studied matrix factorization [16], we denote f with $f(W, x)$ where $x = (s; a)$ and W are all model parameters without loss of generality. Therefore, the problem in Equation 1 is further simplified to the following Empirical Risk Minimization problem [29] (Equation 2), where N is the number of observations (x, y pairs) from user feedback data. $g(w)$ is a regularization term used to penalize large W, e.g., in terms of L_1 or L_2 norms and is the key of W^* having theoretical guarantees when generalizing to the unknown P [29]. λ is a scalar parameter controlling the strength of the penalty.

$$W^* = argmin_w \sum_{i=1}^{N} L(f(W, x_i), y_i) + \lambda g(W) \tag{2}$$

3.2 Matrix Factorization

Matrix factorization is a type of low-dimensional embedding model where $x = (u; c; a)$ are represented by low-dimensional dense vectors. The dimension noted as d here is a hyper-parameter. FunkSVD [8, 16] is a basic version of the family of matrix factorization models used in this work. See SVDFeature [6] and libFM [21] for generalized versions. Specifically following Equation 2, we have

$$f(W, x_i) = f((b, U, V), (u_i; a_i)) = b_0 + b_{u_i} + b_{a_i} + U_{u_i}^T \cdot V_{a_i} \tag{3}$$

Among the model parameters $W = (b, U, V)$, b is a bias vector and U and V are factor matrices. Note that we use u and a as the index into b, U and V. b_0 is reserved for the global bias (similar to the intercept in linear regression models).

Depending on the domain of y, different loss functions L are suitable. For a rating prediction problem, we use a least-squares loss function as follows.

$$L(f, y) = \frac{1}{2}(f - y)^2 \tag{4}$$

For an action prediction problem, assuming $y \in \{0, 1\}$ representing whether there is a positive action or feedback from the user or not. Define the sigmoid function $\sigma(f) = 1/(1 + exp(-f))$

$$L(f, y) = y \ln \sigma(f) + (1 - y) \ln(1 - \sigma(f)) \tag{5}$$

For both types of problems, we use L_2 norm regularization. That is,

$$g(w) = \lambda_1 |b|_2^2 + \lambda_2 |U|_2^2 + \lambda_3 |v|_2^2) \tag{6}$$

The Stochastic Gradient Descent (SGD) [16] algorithm (shown in Equation 7) is widely used to solve the optimization problem in Equation 2 when f is a matrix factorization model. Define $err = y - \hat{y}$ where $\hat{y} = f$ for rating prediction and $\hat{y} = \sigma(f)$ for binary action prediction. Denote η as the learning rate. The SGD algorithm follows the following updating rules. Note that the updating rule for b is common for b_0, b_u and b_a.

$$\begin{aligned} b &\leftarrow b + \eta(err - \lambda_1 b) \\ U_u &\leftarrow U_u + \eta(err \cdot V_a - \lambda_2 U_u) \\ V_a &\leftarrow V_a + \eta(err \cdot U_u - \lambda_3 V_a) \end{aligned} \tag{7}$$

3.3 Q Learning

Different from supervised learning, when applying the theory of reinforce learning [13, 27] in recommender systems, the recommendation problem is modeled as a sequential decision-making problem. At any time step t ($t = 1, ..., T$) where T is the horizon to consider, the system is faced with the decision of taking action a_t given a state s_t, i.e., $(u_t; c_t)$. For each a_t that the system takes, it is given a reward feedback r_t which could be proportionate to the user's rating or whether the user performs positive action on the item recommendation. The goal of the system is to find a policy, which is a mapping function $\pi(s) \rightarrow a$, to maximize its accumulative reward across the horizon T, i.e., $argmax_\pi R_T$ where $R_T = \sum_{t=1}^{T} r_t$. This accumulative reward is called the value of a policy π (*Policy Value*) if it is followed across the horizon. The *Reward Maximization* problem is unbounded when T goes to infinity unless a discounting factor is applied for future rewards, e.g., defining $R_T = \sum_{t=1}^{T} \gamma^{t-1} r_t$ where the *Discounting Rate* $\gamma \in [0, 1]$.

The full reinforcement learning problem requires learning not only the reward function $r(s, a)$ but also how the environment or

state might change because of its action, i.e., an unknown transition distribution $P(s_{t+1}|s_t, a_t)$. Define $Q(s, a)$ as the accumulative reward or value of taking action a in state s and then following the best policy π' afterwards. The algorithm that iteratively estimates $Q(s, a)$ (instead of $r(s, a)$) according to the following Bellman equation is called *Q-learning*, where s' is the next possible s after taking action a in state s. Q-learning solves the reinforcement learning problem with the solution $\pi'(s, a) = argmax_a Q(s, a)$

$$Q(s, a) = r(s, a) + \gamma \sum_{s'} P(s'|s, a) argmax_{a'} Q(s', a') \tag{8}$$

In applying Q-learning in real-world problems, we usually assume a parametric form for the Q function $Q(s, a)$. E.g., in [28], an ensemble of trees are used. In this work, a simple linear function is used (see the Method section). At the beginning of running the algorithm, the parameters of Q is randomly initialized based on which the right-hand side of Equation 8 can be calculated. Then $Q(s, a)$ adapts itself (by changing parameters) to fit the right-hand side value. This process is iteratively done until $Q(s, a)$ converges.

3.4 Regret Minimization

If we make the assumption that a does not have an effect on the state s, i.e., s_t is I.I.D samples of an unknown distribution $P(s)$, the reward maximization problem can be converted to the problem of minimizing the regret of a policy compared with the best policy π' in an assumed family of policies, i.e., the following *Regret Minimization* [25] problem:

$$\pi^* = argmin_\pi \sum_{t=1}^{T} (r(s_t, a_{\pi',t}) - r(s_t, a_{\pi,t})) \tag{9}$$

Following is a list of the key theoretical differences between the Regret Minimization problem (shortened as RM) in Equation 9 and the Empirical Risk Minimization (shortened as ERM) problem in Equation 2.

- The RM problem assumes $s_t \sim_{i.i.d} P(s)$ while the ERM problem assumes $x_t \sim_{i.i.d} P(x)$ where $x = (s; a)$.
- The algorithm to solve the RM problem is through online learning. That is, learning as t goes from $t = 1$ to T (effectively reading data once) while the algorithm to solve the ERM problem is iterative typically by reading the observation data multiple times, e.g., the SGD algorithm.
- There is a difference in terms of *On-policy* vs. *Off-policy* for the algorithm of the RM problem while the algorithm of the ERM problem does not have such as difference. This distinction actually derives from the first one. When assuming $x_t \sim_{i.i.d} P(x)$, the distribution of the observations is fixed although unknown. In contrary, if only $s_t \sim_{i.i.d} P(s)$ is assumed, different policies or algorithms can observe different distributions of a_t, $r(s, a)$ (which is closer to reality) and hence learn differently. This makes the evaluation of the algorithm for the RM problem hard because the ideal evaluation (many off-policy evaluation methods have been proposed in the literature with certain limitations [12]) takes actual running of the algorithm or policy in the system, e.g., through live experiments, which is done in this research.

3.5 Contextual Bandit and LinearUCB

The Contextual Bandit [17] has the same objective as the regret minimization problem in Equation 9. In our notation, $s = (u; c)$ is considered as the context and both s and a are represented as feature vectors (contrary to the discrete value space in the Multi-Armed Bandit [17] problem), e.g., the vectors of user profile or item attributes. A well-studied algorithm for the contextual bandit problem which is also used in this research is LinearUCB [17]. LinearUCB makes further assumptions about $r(s, a)$ to simplify the problem in Equation 9. Specifically, it assumes $(r|x = (s; a)) \sim Gaussian(x^T\theta, \sigma)$, i.e., the expected reward is linearly related to the input feature vectors. For the purpose of explaining the method of this research, the LinearUCB algorithm is elaborated as follows in Algorithm 1 with the notations in this paper. See [17] for a theoretical bound of the regret of this algorithm.

Algorithm 1: The LinearUCB algorithm

Data: A sequence of $s = (u; c)$ while the algorithm is running in the system. The system has M possible actions (items), each of which represented by a feature vector a. Denote $x_{t,k}$ to be $(s_t; a_k)$ for $k = 1, 2, ..., M$.
Parameters: α, β, d is the dimension of $x_{t,k}$
Initialization: $A = \beta I_d$, $h = 1_d$, i.e., equal weights for all input features.

1 **for** $t = 1, 2, ..., T$ **do**
2 $\theta_t = A^{-1}h$
3 **for** *k=1, 2, ..., K* **do**
4 $\hat{r}_{s_t,a_k} = x_{t,k}^T\theta_t + \alpha\sqrt{x_{t,k}^T A^{-1} x_{t,k}}$
5 **end**
6 Take action $a_t = argmax_k \hat{r}_{s_t,a_k}$, let $x_t = (s_t; a_t)$
7 Receive actual reward feedback r_t
8 $A \leftarrow A + x_t x_t^T$
9 $h \leftarrow h + x_t r_t$
10 **end**
Result: θ_T

Note that in real systems, a page of items will be recommended at one time t instead of a single-item action. In this work, we update Step 6 in Algorithm 1 to take K actions or items $a_1, a_2, ..., a_K$ at the same time t and we observe the user feedback for all items and then Step 8-9 are executed for each of them.

4 METHOD

We conducted an online field experiment on MovieLens [1] to answer our research questions. MovieLens is a movie information website that provides users with a database of movie information and personalized movie recommendations. It has thousands of real active users every month. The *front* page of the site has different sections showing movie cards picked according to different criteria. At the top is a special section with eight movie cards (i.e., top-K=8) for personalized movie recommendations powered by recommendation algorithms. In each section, users can click "see more" to go to another type of page which we call an *explore* page to browse more

[1] https://movielens.org

movie cards with that section criterion (k=24 movie cards per page). Users can rate a movie using a 5-star rating widget (0.5-star rating granularity) displayed along with the movie card according to their preference for the movie. They can dismiss showing a movie by clicking the "not interested" icon. They can also click a movie card to transition to another movie detail page to see details of the movie. To better understand the effects of recommendation algorithms on user engagement, we define the following two categories of user actions on recommendations.

- **Positive Actions** are defined as high ratings (rating>=4.0), clicking to see details of movies or adding movies into a wishlist.
- **Negative Actions** are defined as low ratings (rating<=3.5) or clicking the "not interested" icon.

The experiment follows a between-subjects design, i.e., a user is randomly and persistently assigned into one of the six recommenders in Table 1 (all users need to sign in to use the site features). During the experiment, when MovieLens users visit the front page, we display a prominent invitation link at the top asking users whether they would like to experience a new recommender. If they click the link, an informed consent page is displayed where we briefly introduce the purpose of the study (not the experiment details). Users can either accept or decline to participate in the experiment. If the user accepts, we randomly assign the user persistently through the experiment into one of the six recommenders. After that, this user's site browsing is powered by the assigned recommender, including the item display in the top recommendation section of the front page, the recommendation explore page (potentially with additional user specified filters, e.g., genres or release dates). Users can click a link at the top right corner of the site to opt out from the experimental recommender anytime going back to their original recommender. If a user chooses to opt out, the user cannot go back into the experiment anymore. This study was approved by the Institutional Review Board of our organization.

4.1 The Six Recommenders

Table 1 lists the six recommenders we build for the experiment. For all of the LinearUCB algorithms, we set the exploration parameter $\alpha = 0$ and prior $\beta = 1$ (since there is natural exploration because we make a page of recommendations at once, we leave the study of the exploration effect as future work). For the latter four recommenders (Bandit-* and Reinforce-State), besides predictions made by MF-Rating and MF-Action, another two factors from item attributes are introduced: *Recency* and *Popularity* as defined in the following list. In order to fairly combine these four factors at the beginning (since they are in different scales), we map them into percentiles looking at one year of historical recommendation data. By default, these four factors refer to percentiles instead of the raw values in the rest of the paper.

- *predicted rating*: the predicted rating of MF-Rating for a user ID on an item ID.
- *predicted action*: the predicted positive action probability of MF-Action for a user ID on an item ID.
- *recency*: the release year of the movie item
- *popularity*: the total number of ratings on the movie in the system

Reinforce-State recommender is inspired by the theory of UT-AUT [30] in which the ultimate goal of a technology could be optimizing for user adoption or acceptance. While whether we can directly optimize for user acceptance is an unanswered question, we set out to approach it by employing Q-learning techniques that can handle delayed reward feedback. Reward here becomes whether the user returns within a certain period of time (one week is used in this work). That is, whether a user's session with a recommender system is good or bad is determined by whether the user will come back with a new session within the coming week. The same linear function is used as in Bandit-State recommender to approximate Q function for the purpose of fair comparison across recommenders as in the following Equation 10. Note that it also models the interaction effects of a and s.

$$Q(s,a) = \sum_{i=1}^{4} a_i \cdot \theta_i^{(a)} + a_i \cdot s_i \cdot \theta_i^{(s)} \tag{10}$$

We also need to model user state and its transition in order for the Bellman Equation 8 to run (i.e., we employ a model-based Q-learning instead of model-free [13]). From the theory UT-AUT [30], important psychological metrics characterize the user state, which however is unobservable. The PO-MDP [26] model is designed for this case, but to have better experimental control, tractability and model interpretation, we use a deterministic state transition model. As shown in Table 1, we characterize the current user state s as the aggregate means of the four involved factors calculated on all historically recommended items (with size N_u) to the user u. Then the state transition is straightforward when we make a new recommendation as follows in Equation 11 (i.e., re-calibrating the means). Based on this transition model and the Bellman Equation 8, Algorithm 2 shows in details how we train the Reinforce-State model targeting the delayed reward user return. In the algorithm, Steps 8-9 are equivalent to a LinearUCB update step with exploration parameter $\alpha = 0$, i.e., using the LinearUCB algorithm as an online algorithm to estimate the linear Q function. Figure 1 shows the learning process of the Reinforce-State model. The shape of the curve shows a trend of convergence after 20 epochs. It also reflects that this recommender tends to recommend items that after being displayed users have more sessions afterwards. Note that Q-learning is an off-policy algorithm and hence can be trained on historical data although the quality of the estimated Q values depend on how well the state and action spaces are already explored in the data set.

$$s_i'(s_i, a_i) = \frac{s_i \cdot N_u + a_i}{N_u + 1}, i = 1, ..., 4 \tag{11}$$

The models of MF-Rating, MF-Action and Reinforce-State are trained offline, while the three models of contextual bandits learn online (on-policy). During the experiment, we train the three offline models in batch every week using the past one year of historical data (around 52M movie displays with 2M positive actions and 2.1M ratings; note that the majority of user activities in the site are ratings). We see accuracy gains when using recent one year of data compared with using all historical data evaluating in a temporal way (i.e., training on historical data excluding the most recent week and testing on this most recent week of data). The accuracies on

Table 1: The six recommenders studied in this work

Recommender	Input	Output	Model	Algorithm
MF-Rating	u: user ID a: item ID	y: rating	Equation 3 for f Equation 4 for L	SGD in Equation 7
MF-Action	u: user ID a: item ID	y: positive action or not	Equation 3 for f Equation 5 for L	Same as MF-Rating
Bandit-Two	$a = (a_1; a_2)$ a_1: predicted rating a_2: predicted action	r: positive action or not on-policy	$\theta = (\theta_1; \theta_2)$ $E(r(a)) = a^T\theta$	LinearUCB in Algorithm 1
Bandit-Four	$a = (a_1; ...; a_4)$ a_1, a_2: Same as Bandit-Two a_3: recency of item ID a_4: popularity of item ID	Same as Bandit-Two	$\theta = (\theta_1; ...; \theta_4)$ $E(r(a)) = a^T\theta$	Same as Bandit-Two
Bandit-State	$x = (s; a)$ a: Same as Bandit-Four $s = (s1; ...; s4)$ $s_i = \frac{1}{N_u}\sum_{j=1}^{N_u} a_{i,j}$, for $i = 1, ..., 4$	Same as Bandit-Four	$\theta = (\theta^{(a)}; \theta^{(s)})$ $E(r(s,a)) = \sum_{i=1}^{4} a_i \cdot \theta_i^{(a)}$ $+a_i \cdot s_i \cdot \theta_i^{(s)}$	Same as Bandit-Four
Reinforce-State	Same as Bandit-State	$r_t = 1$ if the user returns in a week and t is the end of a session $r_t = 0$ otherwise	$\theta = (\theta^{(a)}; \theta^{(s)})$ $Q(s,a)$ is in Equation 10	Q-learning in Algorithm 2

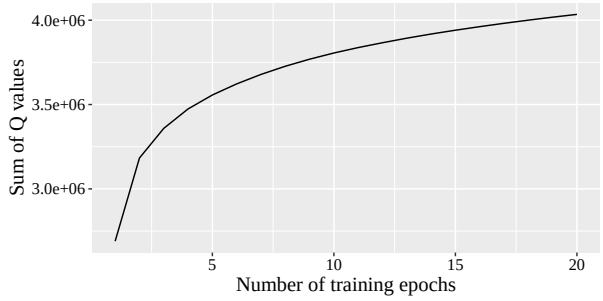

Figure 1: The Q-learning curve in training the reinforcement delayed reward model using one year of historical data. The y-axis is the sum of Q-values for all $t = 1, ..., T$ in each epoch on the x-axis.

average are 0.957 in testing RMSE (Root Mean Squared Error), 0.735 in testing MAE (Mean Average Error) for the rating prediction model, and 0.703 in testing AUC (Area Under the ROC) for the action prediction model.

When training the two models each week during the experiment, we use the recent week of data as the validation set to avoid overfitting (regularization parameters for both bias and factor terms in Equation 3 are set to 1e-5, 20 epochs at maximum, 30 latent dimensions after tuning offline). Then we further update the trained models by running the SGD steps in Equation 7 over the validation data once to make sure the models are up to date. During the week before the next batch training, the models are updated in real-time by running the same SGD steps once whenever the users rate or browse movies in the site.

4.2 Objective Measurements

We measure the following objective activity metrics. In order to have better controlled observations, we set an observation time window for one month, i.e., we only look at each user's activities for one month after the user joins the experiment. We exclude users who joined the experiment too recently to collect data across the full time window. If a user opts out from the experiment during the observation time window, we cut off the data collection after the moment that the user opts out.

- *#sessions*: the number of sessions, i.e., how many session-level visits to the site do users have. Each user has one observation for this metric.
- *#frontView*: the number of front page views. Each user has one observation for this metric.
- *#exploreView*: the number of explore page views. Each user has one observation for this metric.
- *frontPositive*: whether there is any positive action on the eight(top-K=8) recommendations in the front page. Each front page view has one observation for this metric.
- *frontNegative*: whether there is any negative action on the eight(top-K=8) recommendations in the front page. Each front page view has one observation for this metric.
- *explorePositive*: whether there is any positive action on the recommendations (top-K>=24) in the explore pages. Each explore page view has one observation for this metric.
- *exploreNegative*: whether there is any negative action on the recommendations (top-K>=24) in the explore pages. Each explore page view has one observation for this metric.

Algorithm 2: The Q-learning algorithm for Reinforce-State recommender.

Data: A sequence of $(userID_t, itemID_t, return_t)$ where $t = 1, ..., T$. $return_t$ is pre-calculated by organizing the historical data as sessions and setting $return_t = 1$ if t is the last recommendation of a session and the user has another session within the coming week. Otherwise, $return_t = 0$

Parameters: The maximum number of rounds: $epoch_{max}$; learning rate η; the number of features $d = 8$ here

Initialization: Initialize $A = \alpha I_d$, $h = \beta 1_d$; Pre-train the models of MF-Rating and MF-Action

for $epoch = 1, 2..., epoch_{max}$ **do**
  for $t = 1, 2, ..., T$ **do**
    Calculate $(s_t; a_t)$ by making predictions on $(userID_t, itemID_t)$ using MF-Rating and MF-Action models.
    Calculate s' based on Equation 11.
    $\theta = (\theta^{(s)}; \theta^{(a)}) = A^{-1}h$,
    Calculate $Q(s', a)$ for all possible a based on Equation 10 (which involves making predictions for $userID_t$ on all item IDs with MF-Rating and MF-Action models to get a)
    Calculate the right-hand side of the Bellman Equation 8. Denote the value as Q'_t.
    $A \leftarrow A + (s_t; a_t)(s_t; a_t)^T$
    $h \leftarrow h + (s_t; a_t)Q'_t$
  end
end

Result: $\theta_{epoch_{max}, T}$

- *optOut*: whether a user opts out from the assigned recommender during the time window. Each user has one observation for this metric.

4.3 Subjective Measurements

We also deploy surveys with the following list of question items asking users about several perceptional aspects of the recommenders. For some of the classical metrics, e.g., accuracy, diversity, novelty etc., we directly use the design of the prior research by Pu et al. [20]. We designed the rest of the questions for measuring the specific aspects that might be affected by our manipulation. While users are browsing recommendations in the explore page, we prompt users through a banner or pop-up; we started with the banner format and later turned to the pop-up format because the response rate of a banner is too low to collect enough survey feedback. Each user is prompted twice at maximum with the first and the second showing respectively three minutes or a week after a user joins the experiment (and if they show up browsing the explore page). We leave the survey link persistent on the page throughout the experiment so that users can give feedback anytime.

- *accuracy*: The recommendations match my tastes in movies.
- *diversity*: The recommendations have a diverse selection of movies.
- *novelty*: The recommendations help me discover interesting movies that I did not know.
- *temporary interest*: The recommendations reflect my recent interest in movies.
- *attractiveness*: I am interested in seeing or knowing more about the movies in the recommendations.
- *confusion*: I get disoriented sometimes by the change of the recommendations.
- *balance of recency*: The balance between new and old movies in the recommendations is appropriate to me.
- *balance of popularity*: The balance between popular and less popular movies in the recommendations is appropriate for me.
- *understandability*: I understand why the recommender is recommending the movies in my top-picks.
- *reactivity*: The recommendations change appropriately in reaction to what I do in MovieLens.
- *satisfaction*: Overall, I am satisfied by the recent recommendations from the recommender.

5 RESULTS

Table 2: The coefficients (and standard errors in the parentheses) of the activity metrics predicting user overall satisfaction. These estimates come from an ordinal regression (cumulative link) model treating *satisfaction* response as ordinal values. Each observation is a user who has answered the *satisfaction* question for at least once.

Activity Metric	Coefficient (vs. Satisfaction)
log(#sessions)	0.524 (0.236)
log(#frontView)	0.0946 (0.213)
log(#exploreView)	-0.446 (0.152) *
frontPositive rate	0.522 (0.368)
frontNegative rate	-0.342 (0.302)
explorePositive rate	0.336 (0.245)
exploreNegative rate	-0.052 (0.162)

The experiment was launched on Feb. 8, 2017 and the analysis was run on Aug. 27, 2017. During this period, 1508 users joined the experiment for at least one month, which gives us around 250 users for each recommender condition. They generated 12,627 sessions and 279,632 activities, including 92,289 ratings (41,651 of which >= 4.0), 34,587 clicks, and 12,919 wishlist additions. The remaining measured activities are mostly front page and explore page views. In total, we collected 3887 survey responses which gives us around 60 responses for each question on each recommender. We only use the last response of a user on a question if the user has multiple responses on the same question (we found using all survey responses of each user gives us the same results). We rely on different types of regression modeling techniques (including their p-values and effect sizes) treating MF-Rating as the baseline to draw conclusions and to avoid excessive pairwise comparisons. In addition, we performed the Benjamini-Hochberg correction procedure [10] to control the False Discovery Rate of the analysis (effectively using around p<=0.0053 as the significance level). Before doing

Table 3: The coefficients (and standard errors in the parentheses) of the perception metrics predicting user overall satisfaction. These estimates come from an ordinal regression (cumulative link) model treating *satisfaction* response as ordinal values while others as continuous values. Each observation is a user who has completed all the survey questions for at least once.

Perception	Coefficient (vs. Satisfaction)
accuracy	1.31 (0.282) *
diversity	0.595 (0.242)
novelty	-0.0294 (0.198)
temporary interest	0.847 (0.297) *
attractiveness	1.07 (0.241) *
confusion	-0.244 (0.167)
balance of recency	0.520 (0.209)
balance of popularity	-0.0165 (0.209)
understandability	0.255 (0.204)
reactivity	-0.198 (0.259)

the analysis, we confirmed that the user activities were not significantly different for users in different conditions in terms of our activity measurements during the one month before they joined the experiment. This is to make sure that the effects come from our recommender manipulation.

5.1 Measurements Interpretation

We first conduct analysis to correlate user activity metrics and perception metrics with the user's overall satisfaction to understand how different types of activities and different perception aspects of the recommender contribute to its success or failure.

Table 2 and 3 show the results of predicting user overall satisfaction with both objective activity metrics and subjective perception metrics. We can see that perceived accuracy and attractiveness (interest users to know more about the items) are the two most important factors predicting user satisfaction. Whether recommendations reflect users' temporary interest is positively predicting satisfaction but with a smaller effect size. Table 2 also shows that the amount of browsing (*#exploreView*) is negatively associated with user satisfaction. The data shows a trend that higher positive action rate corresponds to higher user satisfaction while high negative action rate corresponds to lower user satisfaction.

5.2 RQ1

What are the differences between recommender systems based on explicit vs. implicit user feedback data modeled with the classical matrix factorization algorithms?

The first two lines of Table 4 compare MF-Rating (based on explicit feedback data) with MF-Action (based on implicit feedback data) on user objective activity measurements. Our overall measure of user satisfaction was not signficantly different between the two recommenders, but there are several differences in user activities. Specifically, MF-Action significantly increases the front page view, explore page view, frontPositive rate, frontNegative rate and exploreNegative rate. We find that in terms of the *balance of recency*, MF-Action is perceived to be significantly better than MF-Rating.

We did not find significant difference for metrics of *#sessions* (mean: 4.9 per user; similar analytical model as *#frontView*), *explorePositive* rate (mean=0.658; similar analytical model as *exploreNegative* rate) and *optOut* rate (mean=0.185; a logistic regression model) after correction. We did not find significant differences on the other subjective measurements that we deployed after correction.

5.3 RQ2

Do multi-factor-blending recommendation algorithms lead to improved or changed user experience and if so, how?

As shown in the rest of the Table 4, there is no significant difference on user satisfaction, but we see potential useful differences on user experience and activities. Bandit-Two recommender is very similar to MF-Action (sharing the benefit of increased front page view engagement, front page positive action rate, the cost of increased front page negative action rate and explore page negative action rate and improved perception of the balance of item recency) except that it does not significantly increases the amount of explore page view. From this aspect, Bandit-Two gains some potential benefit in terms of user satisfaction over MF-Action because explore page view is negatively predicting user satisfaction as shown in Table 2.

Bandit-Four recommender is almost identical to MF-Action in terms of net effects on user activities although it has very different inputs and algorithms. In other words, the introduced two additional factors (recency and popularity) compared with Bandit-Two seem to only increase the amount of explore page browsing. Bandit-State does not significantly affect the amount of front page view and explore page view, but significantly increases users' rate of actions compared with MF-Rating, including not only front page positive action rate but also front page negative and explore page negative action rate. Comparing with all previous recommenders, it seems to gain some benefits over MF-Rating, MF-Action and Bandit-Four, but has some cost compared with Bandit-Two because it loses the benefit of increased front page view engagement.

We did not find significant benefits in terms of the perceived balance of popularity and recency for Bandit-Four (which explicitly models these two factors) compared with Bandit-Two, MF-Action and MF-Rating, with our current model design. On contrary, Bandit-Two and Bandit-State gains some benefit in the perceived balance of recency compared with MF-Rating.

Reinforce-State recommender only has a significant negative effect on the front page action rate compared with MF-Rating and loses more benefits compared with Bandit-State and others. This is also consistent with our subjective measurements, where Reinforce-State has a trend of hurting many perceptional aspects including perceived accuracy, novelty and attractiveness compared with MF-Rating.

6 DISCUSSION AND FUTURE WORK

The above section empirically demonstrates the differences between recommender systems based on user explicit rating feedback and

Table 4: Means of both objective and subjective metrics for the six recommenders. Significant differences after correction are marked with *. Metrics that are not presented here do not show significant differences. The numbers in parentheses for #frontView and #exploreView are standard errors. These estimates come from two negative binomial regression models using MF-Rating as the baseline. The intervals for frontPositive rate, frontNegative rate, exploreNegative rate are 95% confidence intervals. These estimates come three mixed-effects logistic regression models using MF-Rating as the baseline (treating user ID as a random intercept effect). The means of *balance of recency* and *satisfaction* are calculated treating the survey responses as continuous values. The coefficients in the parentheses come from a ordinal regression (cumulative link) model using MF-Rating as the baseline (treating responses as ordinal values).

Recommender	#frontView	#exploreView	frontPositive rate	frontNegative rate	exploreNegative rate	balance of recency	satisfaction
MF-Rating	9.64 (0.733)	41.8 (4.47)	0.0740 (0.057, 0.0947)	0.0198 (0.0134, 0.0291)	0.175 (0.135, 0.224)	2.73 (N.A.)	2.89 (N.A.)
MF-Action	13.1 (1.00)*	72.6 (7.92)*	0.147 * (0.120, 0.180)	0.0521 * (0.0383, 0.0706)	0.330 * (0.272, 0.393)	3.20 (0.879)*	3.03 (0.253)
Bandit-Two	14.1 (1.15)*	48.4 (5.58)	0.133 * (0.105, 0.166)	0.0485 * (0.0346, 0.0674)	0.308 * (0.247, 0.377)	3.32 (0.969)*	3.00 (0.129)
Bandit-Four	13.9 (1.04)*	63.8 (6.78)*	0.125 * (0.102, 0.153)	0.0471 * (0.0350, 0.0631)	0.324 * (0.268, 0.386)	3.04 (0.501)	2.78 (-0.170)
Bandit-State	9.63 (0.741)	41.4 (4.44)	0.140 * (0.113, 0.173)	0.0605 * (0.0445, 0.0818)	0.327 * (0.264, 0.396)	3.26 (0.969)*	3.17 (0.485)
Reinforce-State	10.5 (0.828)	35.2 (3.92)	0.101 (0.0789, 0.128)	0.0405 * (0.0286, 0.0571)	0.213 (0.164, 0.271)	3.13 (0.647)	2.62 (-0.436)

implicit action feedback. Depending on the goals of the system, different recommendation algorithms might be used.

If the goal is to increase engagement, then predicting implicit action is much more effective than predicting explicit ratings. However, if the goal of the system is to improve user satisfaction, we need to be cautious not to exclusively optimize for predicting implicit action, because this type of recommender does not seem to be as precise as a rating-based recommender, as reflected by the increased negative user engagement, e.g., negative action rate and increased user browsing effort. This observation points to the future direction of exploring to learn from both positive and negative feedback with a particular focus on penalizing items with negative user feedback.

Blending the two types of recommenders by optimizing for online user interaction achieves the same level of user engagement as using implicit-action-based recommender alone, but it does not lead to more user browsing, which seems to achieve a trade-off between the goals of user engagement and satisfaction. We think this effect may generalize to other algorithms modeling explicit or implicit feedback data because it likely reflects the fundamental property of the two types of feedback signals instead of the specific algorithms (although more studies are necessary to confirm). It also suggests that if we truly want to capture user satisfaction, we might need to go beyond ratings or actions, e.g., measuring how the system assists the user's decision making tasks and optimizing for it.

Lastly, if the goal of the system is to improve users' perceived balance of recency on the recommendations, blending the two types of recommenders helps with this goal as well.

This work also provides design implications for future work on blending multiple factors (i.e., ensemble of multiple recommenders) and utilizing user return as a learning feedback signal for recommender systems. Based on our observation in this work, we think that

- a better control is necessary (e.g., by introducing a broader set of observations on the context of the user return) in order to use user return to explain whether a session of recommendations are good or bad.
- the state (transition) model and the Q function approximation model need to have enough power (e.g., linear models may not be enough as used in this work) to capture both the actual user state (transition) and the complexity of Q value (a projection of the future value of a recommendation). Recently, Wu et al. [31] proposed to use recurrent neural networks for the state transition model and demonstrated some benefits, which we think is a promising direction to try.
- the balance of recency or popularity are inherently personalized and high-order (e.g., quadratic) because different users prefer different kinds of balance [9] and hence going beyond linear models here might be necessary as well.

We studied several new perceptional metrics for recommendations different from prior literature [20]. These factors are especially prominent in a live interactive recommender system, e.g., whether the recommendations reflect users' temporary interest (which might be only valid in the field), whether users are interested to know more about the recommendations and whether users are disoriented by the change of the recommendations (changing dynamics are common in live systems, e.g., Youtube, Amazon etc.). We find interesting results that besides the perceived accuracy, whether recommendations can interest users to know more about them and reflect users' temporary preference are significant positive factors in predicting the overall user satisfaction. However, future work is necessary to study the psychological constructs of these question items and how they (together with the behavioral metrics) can be generalized to platforms that directly involve user consumption (which could be a stronger type of implicit feedback

signals compared with the user interactions with movie information used in this study and might demonstrate significant differences in these metrics).

7 CONCLUSION

In this work, we conducted a large-scale randomized, controlled between-subjects field experiment to study six recommenders built using machine learning techniques, ranging from supervised matrix factorization, contextual bandit learning to Q learning. We believe user-centric evaluation in recommender systems is important and demonstrate a possible way of evaluating complex machine-learning-based recommenders from a user-centric perspective. We found that the objective design in machine-learning-based recommender systems significantly affects user experience. Specifically, optimizing for implicit action prediction error engages users more than optimizing for explicit rating prediction error modeled with the classical matrix factorization algorithms. However, the effects are mixed in a way that not only positive engagement but also negative engagement are increased substantially, which gives us a caution from the user's perspective on targeting implicit action only because overall user satisfaction could be negatively affected. We show that blending signals in both explicit and implicit user feedback through an online interactive learning algorithm gain the benefits of engagement and mitigate one of the possible costs (i.e., the increased browsing effort). We tested the approach of utilizing user return as a delayed feedback signal on recommendation quality through Q-learning. It does not improve user experience with our current design but provides with potentially helpful implications for future work on recommender systems applying reinforcement learning.

8 ACKNOWLEDGEMENT

This work was supported by the National Science Foundation under grant IIS-1319382. The first author was also supported by the Doctoral Dissertation Fellowship, 2016-17, by the Graduate School at the University of Minnesota. We thank Liangjie Hong (Etsy Inc., previously at Yahoo Research) and Yue Shi (Facebook, previously at Yahoo Research) for their helpful discussions on reinforcement-learning-based recommender systems. We also thank all the MovieLens users who participated in our study.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2015. Multi-criteria recommender systems. In *Recommender Systems Handbook*. Springer, 847–880.
[2] Charu C Aggarwal. 2016. *Recommender systems*. Springer.
[3] Xavier Amatriain and Justin Basilico. 2012. Netflix recommendations: beyond the 5 stars (part 1). *Netflix Tech Blog* 6 (2012).
[4] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, NY, USA, 35.
[5] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.
[6] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, Dec (2012), 3619–3622.
[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.
[8] Simon Funk. 2006. Netflix update: Try this at home. (2006).
[9] F Maxwell Harper, Funing Xu, Harmanpreet Kaur, Kyle Condiff, Shuo Chang, and Loren Terveen. 2015. Putting users in control of their recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 3–10.
[10] Yosef Hochberg and Yoav Benjamini. 1990. More powerful procedures for multiple significance testing. *Statistics in medicine* 9, 7 (1990), 811–818.
[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.
[12] Nan Jiang and Lihong Li. 2015. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722* (2015).
[13] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.
[14] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* 22, 4-5 (2012), 441–504.
[15] Joseph A Konstan and John Riedl. 2012. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 101–123.
[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[17] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 661–670.
[18] Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*. ACM, 1097–1101.
[19] Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Making recommendations better: an analytic model for human-recommender interaction. In *CHI'06 extended abstracts on Human factors in computing systems*. ACM, 1103–1108.
[20] Pearl Pu, Li Chen, and Rong Hu. 2011. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 157–164.
[21] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.
[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
[23] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. 2015. *Recommender systems handbook*. Springer.
[24] Mario Rodriguez, Christian Posse, and Ethan Zhang. 2012. Multiple objective optimization in recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 11–18.
[25] Shai Shalev-Shwartz and Yoram Singer. 2007. Online learning: Theory, algorithms, and applications. (2007).
[26] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.
[27] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
[28] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. 2015. Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees.. In *IJCAI*. 1806–1812.
[29] Vladimir Vapnik. 1992. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*. 831–838.
[30] Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. 2003. User acceptance of information technology: Toward a unified view. *MIS quarterly* (2003), 425–478.
[31] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 495–503.
[32] Bo Xiao and Izak Benbasat. 2007. E-commerce product recommendation agents: use, characteristics, and impact. *MIS quarterly* 31, 1 (2007), 137–209.
[33] Qian Zhao, Gediminas Adomavicius, F Maxwell Harper, Martijn C Willemsen, and Joseph A Konstan. 2017. Toward Better Interactions in Recommender Systems: Cycling and Serpentining Approaches for Top-N Item Lists.. In *CSCW*. 1444–1453.
[34] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.