Collaborative Kalman Filtering for Dynamic Matrix Factorization

John Z. Sun, Dhruv Parthasarathy, and Kush R. Varshney, *Member, IEEE*

***Abstract*—We propose a new algorithm for estimation, prediction, and recommendation named the collaborative Kalman filter. Suited for use in collaborative filtering settings encountered in recommendation systems with significant temporal dynamics in user preferences, the approach extends probabilistic matrix factorization in time through a state-space model. This leads to an estimation procedure with parallel Kalman filters and smoothers coupled through item factors. Learning of global parameters uses the expectation-maximization algorithm. The method is compared to existing techniques and performs favorably on both generated data and real-world movie recommendation data.**

***Index Terms*—Collaborative filtering, expectation-maximization, Kalman filtering, learning, recommendation systems.**

I. Introduction

RECOMMENDATION systems that provide personalized suggestions are transforming or have transformed industries ranging from media and entertainment, to commerce, to healthcare, to education. Businesses often wish to use transactional or ratings data to recommend products and services to individual customers that they are likely to appreciate, need, or purchase. In both the business-to-business and business-to-consumer paradigms, such recommendations allow companies to create tailored, personalized, and desirable experiences for their customers.

Findings from a recent survey indicate that [1], "At least 80 percent of [chief marketing officers] rely on traditional sources of information, such as market research and competitive benchmarking, to make strategic decisions. But these sources only show customers in aggregate, offering little insight into what individual customers need or desire." Recommendation systems that provide individual-level customer insights are thus increasingly important components of commerce in this age of big data. An early adopter of recommendation systems has been the media and entertainment industry.

Manuscript received April 11, 2013; revised March 12, 2014; accepted May 08, 2014. Date of publication May 23, 2014; date of current version June 24, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Raviv Raich.

J. Z. Sun and D. Parthasarathy were with the Department of Electrical Engineering and Computer Science and the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: johnsun@alum.mit.edu; dhruvp@alum.mit.edu).

K. R. Varshney is with the Mathematical Sciences and Analytics Department, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: krvarshn@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2014.2326618

Nick Barton, a vice president of sales and marketing for InterContinental Hotel Group says [1], "We have to get scientific about the customer experience." The science and technology for recommendation that has been noted in the recent literature to be accurate and robust in many applications is collaborative filtering using matrix factorization (MF) techniques [2]. For instance, many entries in the Netflix prize competition, including the winning submission by BellKor's Pragmatic Chaos, relied heavily on MF to create predictions for movie ratings [3]. MF, the decomposition of a matrix into a product of two simpler matrices, has a long and storied history in statistics, signal processing, and machine learning for high-dimensional data analysis [4].

The commercial world is not static, but is full of dynamics in customer preferences, product and service offerings, and so on. User tastes and needs evolve over time both exogenously and due to interactions with the provider. In the common application domains, customer preferences often follow predictable trajectories over time. Customers may be interested in basic products at first and then higher-end products later, or products for toddlers first and for adolescents later; a customer may like particular films for only short time periods and not like them before or after. Additionally, we can distinguish recommendation for discovery and recommendation for consumption; new items are recommended in the former whereas the same item may be repeatedly recommended in the latter.

A recognized limitation of plain MF-based collaborative filtering methodologies is that they do not account for changes over time and are therefore inherently restricted. Despite their limitations, MF without any dynamic modeling and MF enhanced with fairly limited dynamic modeling have been widely and successfully used. This fact begs the question why. Is it that in real-world settings, preferences do not evolve much or only evolve in very simple ways? Or is it that a more sophisticated and expressive dynamic model can take performance to an even higher level beyond what is currently achieved? Towards this end, we propose a new algorithm, the collaborative Kalman filter (CKF), that employs such an expressive temporal model: a state space model to track user preferences over time [5], [6].

Our new contribution builds on known theory as follows. The MF approach to collaborative filtering usually includes Frobenius-norm regularization [3], which is supported by a linear-Gaussian probabilistic model known as probabilistic matrix factorization (PMF) [7]. Due to its linear-Gaussian nature, PMF lends itself to incorporating temporal trajectories through the state space representation of linear dynamical systems [8] and algorithms for estimation based on the Kalman filter [9], [10]. We propose a general recommendation model of this form and

1053-587X © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

develop an expectation-maximization (EM) algorithm to learn the model parameters from data. The Kalman filter and Rauch-Tung-Striebel (RTS) smoother [11] appear in the expectation step of the EM.

The remainder of the paper is organized as follows. In Section II, we introduce our notation and review existing techniques in matrix factorization. In Section III, we develop a dynamical system model for users, items, and ratings. In Section IV, we describe an EM algorithm to learn the model parameters from sparse data. Empirical results and comparisons to baseline methods for generated and real-world datasets are provided in Section V and Section VI, respectively. Finally, we conclude in Section VII.

II. Preliminaries

In this section, we introduce the MF approach for studying recommendation problems and introduce prior work that incorporate temporal dynamics into preference estimation.

A. Problem Formulation

In the recommendation problem, users provide explicit (e.g. ratings) or implicit (e.g. usage) information about their preferences for known items for the purpose of being introduced to new items. One simple model that has been successful in practice assumes that user preference can be captured by continuous-valued weightings on a small set of K latent factors. Each user i's weighting, called a user factor, is denoted by a row vector $u_i \in \mathbb{R}^K$. Similarly, each item j is assigned a row vector $v_j \in \mathbb{R}^K$ representing its characteristics on the same latent space. The explicit rating of item j by user i is then described by the inner product $o_{ij} = \langle u_i, v_j \rangle = u_i v_j^T$ [3].

In MF, a further assumption that users employ the same set of latent factors allows for estimates of individual preferences from population data. In this setting we collect user factors into a matrix $U \in \mathbb{R}^{N\times K}$ and item factors into a matrix $V \in \mathbb{R}^{M\times K}$. Meanwhile, ratings from N users about M items can be represented by a preference matrix $O \in \mathbb{R}^{N\times M}$. For most practical situations, only a small fraction of the entries of O are observed and may be corrupted by noise, quantization and different interpretations of the scale of preferences. By estimating the user and item factors, which is modeled to have much lower dimension, the remaining rating entries are predicted through the relationship $O = UV^T$.

Under MF, latent factors are learned from past responses of users rather than formulated from known attributes. Factors are not necessarily easy to interpret and change dramatically depending on the choice of K. The value of K is an engineering decision, balancing the tradeoff of forming a rich model to capture user behavior and being simple enough to prevent overfitting.

B. Prior Work

One popular technique to estimate the unobserved entries of O in the MF framework is by minimizing U and V in the following program:

$$\min_{U,V} \sum_{i,j\in\mathcal{O}} \left(o_{ij} - u_i v_j^T\right)^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2, \qquad (1)$$

where $\mathcal{O}$ is the set of observed ratings and (λ_1, λ_2) are regularization parameters. The SVD algorithm solves this program using stochastic gradient descent while correcting for user and item biases, and has been experimentally shown to have excellent root mean squared error (RMSE) performance [12].

The regularization in the above program was motivated by assigning Gaussian priors to the factor matrices U and V respectively [7]. Coined PMF, this Bayesian formulation means (1) is justified as producing the *maximum a posteriori* (MAP) estimate for this prior. In this case, the regularization parameters λ_1 and λ_2 are effectively signal-to-noise ratios (SNR). Since O is not a linear function of latent factors, the MAP estimate does not in general produce the best RMSE performance, which is the measure often desired in recommendation systems [13]. However, although the MAP estimate does not necessarily minimize the RMSE, it does tend to yield very good RMSE performance [14], and wisdom gained from the Netflix Challenge and experimental validation from [7] show that the MAP estimate provides very competitive RMSE performance compared to other approximation methods.

The SVD algorithm assumes that both user and item factors are constant in time. However, it is common for customer tastes to evolve, oftentimes cohesively as a population. This is exploited in the timeSVD algorithm, which allows user preferences to evolve linearly over time [15]. In this case, user factors are given by

$$u_i(t) \approx \bar{u}_i + \alpha_i \mathrm{dev}_i(t), \qquad (2)$$

where $\mathrm{dev}_i(t)$ is some deviation function from a central point $\bar{u}_i$ and α_i is the weightings on the deviation on each factor.

There are other works that investigate temporal dynamics in recommendation systems. The probabilistic tensor factorization approach extends the probabilistic MF formulation of [7], but requires the time factors to lie on the same latent factor space as users and items [16]. The state evolution of the spatiotemporal Kalman filter is limited and the approach encounters convergence issues [17]. The approach known as target tracking in recommendation space has no element of collaboration and requires prior knowledge of the 'recommendation space' [18]. The hidden Markov model for collaborative filtering captures the time dynamics of a known attribute among users rather than learned factors [19]. The temporal formulation of [20], which is nearest neighbor-based rather than MF-based, is known to have scaling difficulties. The dynamic nonlinear matrix factorization approach of [21], which was published after the submission of our initial work [5], is along much the same lines as the CKF, but uses a Gaussian process dynamical model instead of the linear state space model.

III. State Space Model

Given the success of MAP estimation in linear-Gaussian PMF models and our interest in capturing time dynamics, we propose a linear-Gaussian dynamical state space model of MF whose MAP estimates can be obtained using Kalman filtering. We assume that user factors $\mathbf{u}_i(t)$ are functions of time and hence states in the state space model, with bold font indicating the vector being random. In our proposed model, we have coupled dynamical systems, and to adhere to typical Kalman

filter notation, we use $\mathbf{x}_{i,t} = \mathbf{u}_i(t)$ to denote the state of user i at time t.

For each user, the initial state $\mathbf{x}_{i,0}$ is distributed according to $\mathcal{N}(\mu_i, \Sigma_i)$, the multivariate Gaussian distribution with mean vector μ_i and covariance matrix Σ_i. The user-factor evolution is linear according to the generally non-stationary transition process $A_{i,t}$ and contains transition process noise $\mathbf{w}_{i,t} \sim \mathcal{N}(0, Q_{i,t})$ to capture variability of individuals. Taken together, the state evolution is described by the set of equations:

$$\mathbf{x}_{i,t} = A_{i,t}\mathbf{x}_{i,t-1} + \mathbf{w}_{i,t} \quad i = 1, \ldots, N. \tag{3}$$

We assume that item factors evolve very slowly and can be considered constant over the time frame that preferences are collected. Also, due to the sparsity of user preference observations, a particular user-item pair at a given time t may not be known. Thus, we incorporate the item factors through a non-stationary linear measurement process $H_{i,t}$ which is composed of subsets of rows of the item factor matrix V based on item preferences observed at time t by user i. Note that all $H_{i,t}$ are subsets of the same fixed V and are coupled in this way. We also include measurement noise $\mathbf{z}_{i,t} \sim \mathcal{N}(0, R_{i,t})$ in the model. The overall observation model is:

$$y_{i,t} = H_{i,t}\mathbf{x}_{i,t} + \mathbf{z}_{i,t} \quad i = 1, \ldots, N. \tag{4}$$

The product $H_{i,t}\mathbf{x}_{i,t}$ in (4) parallels the $\langle u_i, v_j \rangle$ product in Section II-A. Again adhering to Kalman filter notation, we use $y_{i,t}$ to denote the observations, corresponding to the observed entries of O, now a tensor in $\mathbb{R}^{M \times N \times T}$.

The state space model can be generalized in many different ways that may be relevant to recommendation systems, including non-Gaussian priors, nonlinear process transformation and measurement models, and continuous-time dynamics. We focus on the linear-Gaussian assumption and defer discussion on extensions to Section VII.

IV. Collaborative Kalman Filtering

Although both SVD and timeSVD have been shown to be successful in practice, they are limited in accounting for general temporal changes in user preferences. To combat this problem, we introduce the collaborative Kalman filter to better exploit temporal dynamics in recommendation systems. The key innovation of CKF is allowing user factors to evolve through the linear state space model introduced above. Again by assuming Gaussian priors on the user and item factors, the MAP estimate can be computed optimally via a Kalman filter. Although the Kalman filter requires the knowledge of model parameters which may not be known *a priori*, the EM algorithm is used to learn these parameters efficiently.

The CKF algorithm involves learning the parameters V, $A_{i,t}$, $Q_{i,t}$, $R_{i,t}$, μ_i and Σ_i, and estimation of $\mathbf{x}_{i,t}$. In this architecture, N Kalman smoothers, one for each user, are computed in parallel utilizing the same item factor matrix V in the E-step of the EM algorithm, which for the case of Gaussian priors is the same as performing Kalman filtering. Then, we refine the model parameter estimates in the M-step, and repeat. In summary, the EM algorithm alternates between the expectation step in which the expectation of the likelihood of the observed data is evaluated for fixed parameters, and the maximization step in which the expected likelihood is maximized with respect to the parameters. Below, we explain both steps.

A. E-Step

In order to infer user factors in the expectation step, we utilize the noncausal Kalman filter, which provides the MAP estimate assuming the item factors and model parameters are known. For user $i \in \{1, \ldots N\}$, we define the state estimate and the state covariance as

$$\hat{\mathbf{x}}_{i,t|s} = \mathbf{E}[\mathbf{x}_{i,t} \mid y_{i,1}, \ldots, y_{i,s}] \tag{5}$$

$$P_{i,t|s} = \mathrm{Var}(\mathbf{x}_{i,t} \mid y_{i,1}, \ldots, y_{i,s}). \tag{6}$$

The noncausal Kalman filter, also known as the RTS smoother, is a forward-backward algorithm that forms an estimate using all observations. To begin, we run N causal Kalman filters for $t = \{1, \ldots T\}$:

$$\hat{\mathbf{x}}_{i,t+1|t} = A_{i,t+1}\hat{\mathbf{x}}_{i,t|t}; \tag{7}$$

$$P_{i,t+1|t} = A_{i,t+1}P_{i,t|t}A_{i,t+1}^T + Q_{i,t+1}; \tag{8}$$

$$\hat{\mathbf{x}}_{i,t|t} = \hat{\mathbf{x}}_{i,t|t-1} + K_{i,t}(y_{i,t} - H_{i,t}\hat{\mathbf{x}}_{i,t|t-1}); \tag{9}$$

$$P_{i,t|t} = P_{i,t|t-1} - K_{i,t}H_{i,t}P_{i,t|t-1}. \tag{10}$$

Then the smoothing steps are performed:

$$\hat{\mathbf{x}}_{i,t|T} = \hat{\mathbf{x}}_{i,t|t} + J_{i,t}(\hat{\mathbf{x}}_{i,t+1|T} - \hat{\mathbf{x}}_{i,t+1|t}); \tag{11}$$

$$P_{i,t|T} = P_{i,t|t} + J_{i,t}(P_{i,t+1|T} - P_{i,t+1|t})J_{i,t}^T; \tag{12}$$

$$P_{i,T,T-1|T} = (I - K_{i,T}H_{i,T})A_{i,T}P_{i,T-1|T-1}; \tag{13}$$

$$P_{i,t,t-1|T} = P_{i,t|t}J_{i,t-1}^T + J_{i,t}(P_{i,t+1,t|T} - A_{i,t+1}P_{i,t|t})J_{i,t-1}^T, \tag{14}$$

where

$$K_{i,t} = P_{i,t|t-1}H_{i,t}^T\left(H_{i,t}P_{i,t|t-1}H_{i,t}^T + R_{i,t}\right)^{-1}; \tag{15}$$

$$J_{i,t} = P_{i,t|t}A_{i,t+1}^T P_{i,t+1|t}^{-1}. \tag{16}$$

The estimates $\hat{\mathbf{x}}_{i,t|T}$ can then be combined to form an estimate of the user factor tensor $\widehat{U}$.

B. M-Step

The E-step of the EM algorithm requires knowledge of model parameters such as mean and covariance of the initial states, the transition process matrices, the process noise covariances, the measurement process matrices, and the measurement noise covariances. The M-step progressively refines the estimates for these parameters by iteratively improving the log-likelihood given the observations. In learning the measurement process matrices, we also get an estimate for the item factor matrix V, which is the other ingredient in the MF problem.

Learning the large number of parameters is difficult in practice from such limited observations, but simplifications to process models yield tractable closed-form solutions. These simplifications are that $A_{i,t}$ is fixed for all users and over time, Σ_i is $\sigma_U^2 I$, $Q_{i,t}$ is $\sigma_Q^2 I$, and $R_{i,t}$ is $\sigma_R^2 I$. We will discuss the merits of such assumptions in Section VI and just present the M-step equations of the simplified model here:

$$\hat{\sigma}_U^2 = \frac{1}{NK}\sum_{i=1}^{N} \mathrm{tr}\left(P_{i,0|T} + \hat{\mathbf{x}}_{i,0|T}\hat{\mathbf{x}}_{i,0|T}^T\right); \tag{17}$$

$$\hat{\sigma}_Q^2 = \frac{1}{NKT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left\{\left(P_{i,t|T}+\hat{\mathbf{x}}_{i,t|T}\,\hat{\mathbf{x}}_{i,t|T}^T\right) -2\left(P_{i,t,t-1|T}+\hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right)A^T +A\left(P_{i,t-1|T}+\hat{\mathbf{x}}_{i,t-1|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right)A^T\right\}; \quad (18)$$

$$\hat{\sigma}_R^2 = \frac{1}{|\mathcal{O}|}\left[\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(y_{i,t}y_{i,t}^T\right) -2\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(y_{i,t}\hat{\mathbf{x}}_{i,t|T}^T H_{i,t}^T\right) +\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(H_{i,t}\left(P_{i,t|T}+\hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t|T}^T\right)H_{i,t}^T\right)\right]; \quad (19)$$

$$\widehat{A} = A_2A_1^{-1} \quad \text{where}$$
$$A_1 = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(P_{i,t-1|T}+\hat{\mathbf{x}}_{i,t-1|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right),$$
$$A_2 = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(P_{i,t,t-1|T}+\hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right); \quad (20)$$

$$\widehat{V}_j = V_{2,j}V_1(j)^{-1} \quad j=1,\ldots,M \quad \text{where}$$
$$V_1(j) = \sum_{i=1}^{N}\sum_{t=1}^{T}1_{o_{ijt}}\left(P_{i,t|T}+\hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t|T}^T\right),$$
$$V_2 = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\mathrm{fill}(y_{i,t})\hat{\mathbf{x}}_{i,t|T}^T\right). \quad (21)$$

Remembering that each $y_{i,t}$ is a subvector of $\mathbb{R}^M$ corresponding to items observed at time t, the fill operator expands its argument back to $\mathbb{R}^M$, with the observations in their appropriate positions and zeros elsewhere. We denote the jth rows of V and V_2 as V_j and $V_{2,j}$ respectively, and $1_{o_{ijt}}$ as the indicator function that a rating is observed for user i and item j at time t.

Derivations for these parameters are given in Appendix B.

V. Understanding CKF Behavior

To demonstrate the effectiveness of Kalman learning compared to existing methods, we first present results tested on generated data that follow a state space model; we will present real-world results later in Section VI. We compare the CKF to SVD and timeSVD; these are known to be fast and effective algorithms for recommendation systems, particularly on the Netflix dataset, and often serve as baselines for comparison in the literature. As SVD includes no temporal formulation, we pool together measurements from all times into one matrix.

A. Experimental Setup

There are two main reasons to consider generated data. First, a goal of the work is to understand how algorithms perform on preferences that evolved following a state space model. It is not clear that common datasets used in the recommendation systems literature match this model, and results would be too data-specific and not illuminating to the goal at hand. Second, a generated dataset gives insight on how the algorithms discussed perform in different parameter regimes, which is impossible in collected data.

We generate the item factor matrix V iid $\mathcal{N}(0,\sigma_V^2)$ and the initial user factor matrix $U(0)$ iid $\mathcal{N}(0,\sigma_U^2)$. Under the assumption that user factors do not change much with time, the stationary transition process matrix A is the weighted sum of the identity matrix and a random matrix, normalized so that the expected power of the state $\mathbf{x}_{i,t}$ is constant in time. We note that A can be more general with similar results, but the normalization is important so that preference observations do not change scales over time. Finally, iid noise is added to both the transition and measurement processes as described in (3) and (4). The observation triplets (i,j,t) are uniformly drawn iid from all possibilities from the preference tensor.

B. Results

We present performance results for a particular choice of parameters in Fig. 1, expressed in RMSE. Space limitations prevent us from giving results for other parameter choices, but they are similar when the SNR is reasonable. For arbitrary initial guesses of the parameters, we find learning of variances and process matrices to converge and stabilize after about 10–20 EM iterations. As a result, state tracking is reliable and approaches the lower bound specified by the Kalman smoother output when the parameters, including the item factor matrix V, are known *a priori*. The estimate for the entire preference tensor O also performs well, meaning that CKF is a valid approach for recommendation systems with data following a state space model.

In contrast, current algorithms such as SVD and timeSVD perform poorly on this dataset because they cannot handle general dynamics in user factors. Thus, the algorithm becomes confused and the estimates for the factor matrices tend to be close to zero, which is the best estimate when no data is observed. As shown in Fig. 2, the true trajectory of users may be that of an arc in factor space with additive perturbations. While CKF is able to track this evolution using smoothed and stable estimates, both SVD and timeSVD fail to capture this motion and hence have poor RMSE. SVD does not have temporal considerations and would give a stationary dot in the factor space. Meanwhile, timeSVD can only account for drift, meaning it can move in a linear fashion from a central point. In fact, this constraint leads to worse RMSE for most parameter choices than SVD because timeSVD overfits an incorrect model.

VI. CKF in Practice

In Section IV, we introduced the CKF algorithm and discussed simplifying assumptions that made the analysis tractable. In Section V, we then compared CKF to existing results on generated datasets to demonstrate the gains of the new algorithm. However, it is unclear whether these assumptions are reasonable or are too naïve to allow for effective prediction in practice. Here, we discuss why these assumptions are valid on datasets that are interesting for collaborative filtering and provide a case example to understand how the CKF algorithm tracks temporal changes. We also mention some implementation details associated with CKF such as runtime and robustness.

To validate CKF, we consider the Netflix dataset, which is commonly used to compare MF algorithms. The dataset contains approximately 100 million ratings by about 500,000 users

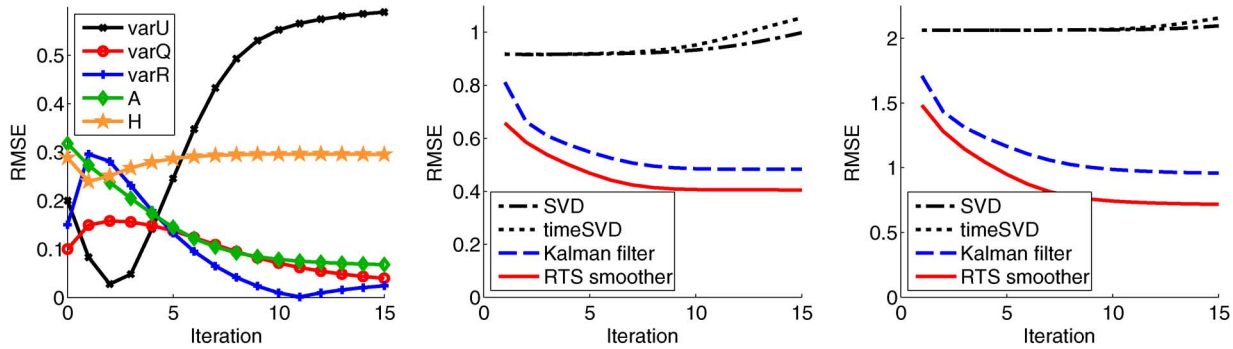


Fig. 1. For this testbench, we set model dimensions to be $(M, N, T, K) = (500, 500, 20, 5)$ and variances to be $(\sigma_U^2, \sigma_V^2, \sigma_Q^2, \sigma_R^2) = (1, 1, 0.05, 0.1)$. The item factor dynamics are controlled by A, which is a weighted average of identity and a random dense matrix. The observation ratio is 0.005, meaning only 0.5% of the entries of the preference matrix are observed. For the generated data and crude initial guesses of the parameters, as a function of iteration, we show the RMSE between the true values used in generating the data and the estimated values of (a) Kalman parameters learned via EM; (b) user factors/states; and (c) the preference matrix. We observe that EM learning is effective in estimating parameters through noisy data, and this translates to better state tracking and estimation of the preference matrix. Convergence is fast and robust to initialization of parameters.

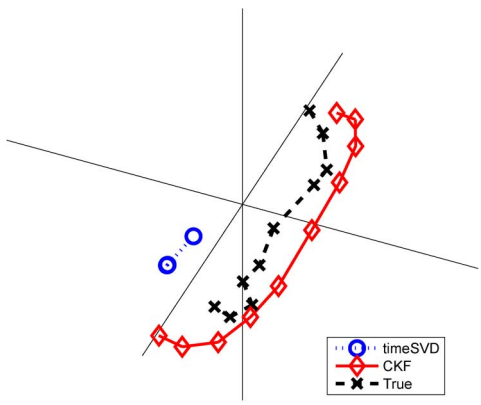


Fig. 2. State-tracking ability of CKF and time SVD in three factor dimensions. The true user factors are more accurately tracked using CKF after parameters have been learned. However, time SVD does not have flexibility to track general state evolutions and gives poor RMSE.

on 18,000 movies. Each rating is accompanied by a timestamp over a period of 84 months, ranging from 1998 to 2006. The timing information here is particularly pertinent since Netflix's interface easily allows users to indicate their preferences soon after watching the movie. This means the temporal trends contain less noise compared to datasets like MovieLens [22], where ratings are potentially collected much later than when the movie was watched.

A. Model Assumptions in CKF

There are several key assumptions that allow for efficient learning and estimation in CKF but may constrain its performance in practice. The first is that CKF is most suitable for the setting in which user tastes are approximately normally distributed over the latent factor space. In many datasets where the number of users is large, this assumption is justified. User factors are of course non-Gaussian but not severely so, as shown empirically for Netflix data in [23, Fig. 3]. There is also a Gaussian assumption on both the process and measurement noise terms, which are common in practice. The Gaussian formulation leads to a simple interpretation of the CKF solution: it is the MAP estimate conditioned on observed data and assumed population similarities.

In simplifying the learning from sparse observations, we also impose stationarity and homogeneity on the state transitions and noise variances. The stationarity simplification is not problematic if the time scales of the dataset are not long enough for dramatic shocks to influence customer behavior in unpredictable ways. It also implies that observed ratings are collected in a consistent manner over users and time. The homogeneity assumption is also important in that it says that temporal customer behavior has a universal component that is of interest to the recommendation system. This is not to say that all users have to evolve in the same way; the Kalman smoother contains a process noise component that allows for individual volatility.

In addition to efficient learning, these assumptions also provide complexity control to prevent overfitting. Moreover, they allow for better interpretation of the learned model by, e.g., business users, because a single transition matrix that highlights the main user trajectories can be more readily understood than a plethora of transition matrices. If we were to relax the homogeneity assumptions, it would be good practice to include extra regularization terms that impose similarity or smoothness between transition matrices, which then moves us towards multitask learning [24].

B. Effectiveness of Temporal Model

The central novelty of CKF and the main investigation of this paper is the temporal evolution of user factors. CKF can learn and estimate user behaviors that take the form of linear transformations of its state vectors, reminiscent of the position-tracking applications that were the original motivations of Kalman filtering. Although user factors may have more complicated trajectories over time, CKF is able to provide a robust first-order approximation.

CKF is therefore a very powerful tool for learning latent factors in datasets where user preferences markedly change over time. The Netflix dataset fits this criterion as temporal variations such as drifts and seasonal changes do occur [15]. This phenomenon is visualized in Fig. 3(a), (b), which demonstrates the variation of average ratings of action movies over time for

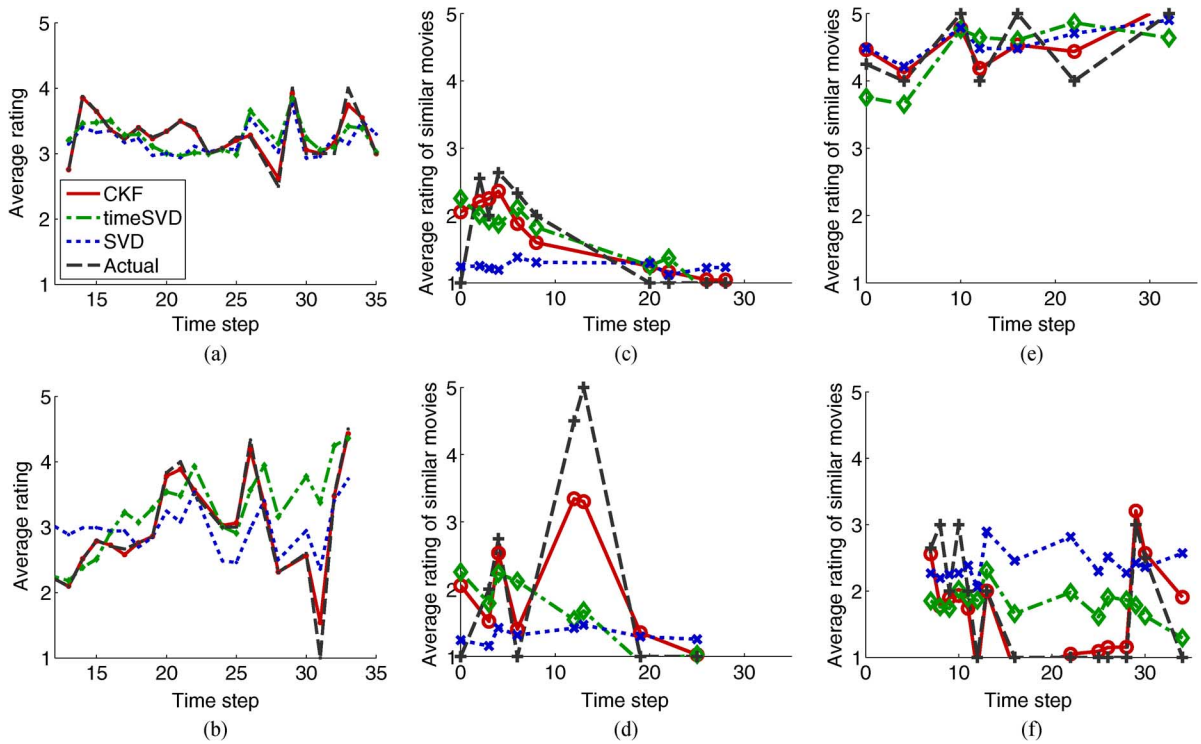


Fig. 3. (a), (b) Average observed ratings of action movies over time for two users. Predicted ratings using estimated user and item factors from the CKF, SVD and timeSVD algorithms are also plotted. The results demonstrate that user preferences have significant temporal dependence which can best be tracked using CKF. (c)–(f) Average observed ratings of a subset of action movies over time for four users. The action movies selected have similar estimated item factors using the CKF, SVD and timeSVD algorithms. We see pronounced temporal variations which is best predicted by CKF. The SVD algorithm predicts the ratings to be approximately equal because user factors do not change over time and item factors will be similar for similar movies. Meanwhile, timeSVD has very limited temporal modeling and can at most have approximately linear temporal deviations.

two users, and the ability of CKF, SVD and timeSVD to predict that behavior. We present this example to illustrate the temporal structure of user preferences and defer detailed discussion on experimental methodology to later.

Further evidence of the time-dependent nature of user preference is demonstrated in Fig. 3(c)–(f). Here, we considered a subset of action movies that have very similar estimated item factor over all three recommendation algorithms and compared the predicted ratings of specific users to the actual ratings observed from the Netflix dataset. Again, we see large variability in the real data, and find that CKF does a good job of accounting for it. Meanwhile, SVD's estimates of user factors do not change over time and its rating estimates are approximately flat; timeSVD can only distinguish drift and its estimates are about linear.

From this case study, we are able to motivate the need to adopt more refined temporal models to better understand and estimate user preferences. The linear-Gaussian state space model of temporal dynamics that is the foundation of the CKF approach tracks real-world user preferences closely without overfitting, suggesting it to be a preferred temporal model for MF-based recommendation.

C. Implementation

Here, we highlight some implementation details and design choices of the CKF algorithm. These considerations were important in analyzing the Netflix dataset and apply more generally.

1) Choice of Model Parameters: Like in most MF algorithms, one degree of freedom in CKF is K, the number of latent factors. A larger K allows for a richer model and potentially better prediction performance, but will increase the runtime of the algorithm and pose the danger of overfitting the observed data. In our analysis of Netflix, we found 5 factors balanced the performance--runtime tradeoff well.

In addition, we must provide initial estimates of the transition matrix and moments of system variables in the state space model. We found that the EM algorithm is very robust to this choice as convergence did not vary greatly for different starting conditions. By scanning a large region of the parameter space, we found the RMSE was within 10% of any set point.

We do not consider regularized user, movie, and time bias terms. Instead, such idiosyncratic behavior is captured naturally as process and measurement noise in the CKF model.

2) Time Quantization: CKF assumes that data is collected in discrete time steps, which is not the case in practice. However, it is plausible that user factors tend to follow smooth trajectories within short time windows and can be closely approximated with piecewise constant estimates, which is equivalent to clustering rating times into distinct buckets. For Netflix, we quantized by month over a three-year window (2003–2006) which accounts for most of the observed data, resulting in $T = 36$ different time steps. This time quantization was effective in capturing short term variations in user preferences while allowing for tractable runtimes.

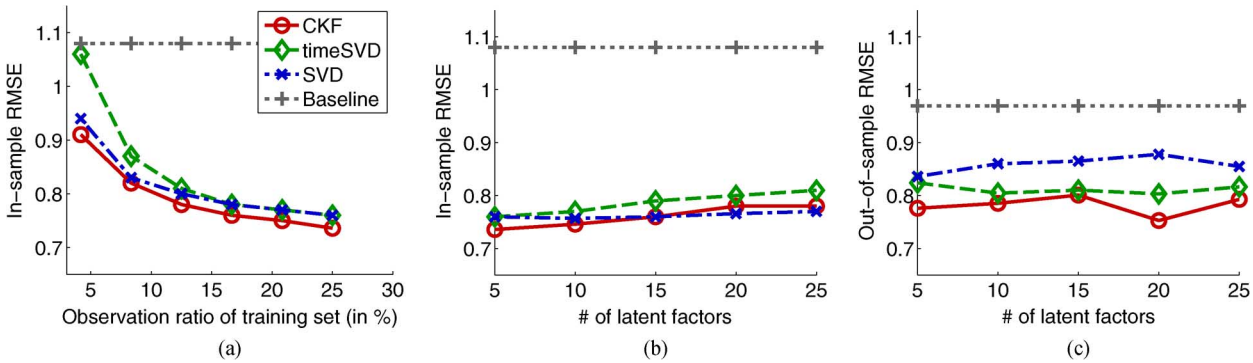


Fig. 4. Performance results for CKF, timeSVD and SVD for the action movies subset of the Netflix dataset. In-sample RMSE is given as a function of (a) observation ratio, and (b) the number of latent factors K. Out-of-sample (prediction) RMSE is given as a function of the number of latent factors K in (c).

3) Runtime Performance: One shortcoming of CKF is that it can be computationally expensive relative to SVD and timeSVD. Although the Kalman filters employed are efficient and further simplifications can dramatically decrease runtime, there are still several matrix inversions that are bottlenecks. In our experimentation, we found that the runtime of SVD and timeSVD are 40% and 50% respectively of the runtime of CKF. We find the time gap increases about linearly with K. However, increasing the number of time steps greatly increases the runtime as it requires additional EM iterations to converge.

Due to such runtime considerations, CKF is well suited for moderate-sized datasets. This is because CKF, in general, takes more time for each iteration than SVD and timeSVD. Clearly, for extremely large datasets, CKF may prove to be an intractable solution due to these reasons.

D. Data and Materials

To facilitate comparisons between the three algorithms discussed, we use a subset of the popular Netflix dataset. This subset comprises the intersection of action movies that have at least 10,000 ratings and users who have rated at least 300 movies. The resulting size of the dataset is 560 users, 959 movies, and 162,114 observations. The fraction of the observed to total ratings, called the observation ratio, is then 30% of the static observation matrix O or 0.84% of the observation tensor $O(t)$, which allows ratings to change at each of the 36 time steps.

We considered this subset of the Netflix dataset for a few specific reasons. First, CKF is more effective when users share common temporal trajectories; using a restricted set of users and movies allow for better analysis of these temporal trends. Second, the smaller subset reduces the computational runtime, which is an important consideration in CKF, especially as the number of time steps is large. Last, this dataset has the appropriate observation ratio which allows CKF to accurately predict the state space parameters. We will address some of the robustness to these choices below.

In our simulations, we first considered $K = 5$, but also tested the change in performance and runtime when K is different. As mentioned previously, we binned time into months over the entire span of the Netflix dataset, yielding 36 time steps. We seed the algorithm with initial estimates $A_{i,t} = I_{K\times K}$, $\sigma_U^2 = 1$, $\sigma_Q^2 = 1$, and $\sigma_R^2 = 1$. A wide range of seeds yielded similar RMSE performance. We tested the effectiveness of the CKF using cross validation, with the size of the validation subset being 1/6 of the total data.

For comparison, we also ran SVD and timeSVD on the datasets. We experimentally found the optimal seeding parameters of these algorithms through multiple simulations. As a result, we set λ_1, the regularization term for the user vectors, λ_2, the regularization term for the item vectors, and Γ, the learning rate, to all be 0.01 for both SVD and timeSVD. We accounted for bias by shifting ratings by a constant offset corresponding to the average movie ratings over all users and movies. We did not account for additional regularized biasing for individual users and items in order to create a direct comparison to CKF, which does not use these biases.

We ran CKF, SVD, and timeSVD for 20 iterations each to obtain the final rating predictions, which was sufficient for all three algorithms to converge.

E. Prediction Performance

In our testing, we find CKF has better RMSE performance than SVD and timeSVD on the action movie subset of the Netflix dataset. Initially, we considered $K = 5$ and an observation ratio of 25% for training (5% for test). We then decreased the observation ratio of the training set and increased the size of K to demonstrate the robustness of CKF's superior performance. RMSE comparisons for both scenarios are presented in Fig. 4(a), (b). We see that all algorithms performed better than the baseline corresponding to the RMSE incurred by imposing the average rating over all users. In general, CKF and timeSVD performed better than SVD by taking advantage of temporal deviations. This advantage is less pronounced when the observation ratio is low because there are just too few observations to learn temporal patterns well. Moreover, we found that a low number of latent factors were sufficient to yield good performance in CKF. In fact, the RMSE increased with larger K due to overfitting. Coupled with the analysis presented in Fig. 3, we see CKF forms better estimates of the temporal variations of user factors, which translates to improved tracking of ratings on the training set and better estimation on the testing set.

Additionally, we consider the out-of-sample prediction or forecasting problem where we use the first 34 time bins as the training set and use the final two months as the testing set. We predict the ratings for all users and movies using the factors

learned for each of the three algorithms. The out-of-sample RMSE values that result are shown in Fig. 4(c). In the prediction case, we see an even greater advantage derived from the state space model dynamics than in the in-sample estimation case. The ordering of performance from worst to best matches the temporal expressiveness in the models, with CKF having the least error across all values of K.

Analysis of the state space parameters demonstrate that there are meaningful state transitions between time steps which yields the improved prediction analysis. These trajectories do not following the drift motion predicted by timeSVD.

There are some limitations of the analysis presented. Because we are only considering a subset of the Netflix dataset, it is difficult to generalize the gain of CKF over existing algorithms. We expect the performance gap to decrease as global temporal trends become less impactful for prediction. We could personalize the state transition matrices in this setting, but the estimation becomes poor unless the percentage of observed ratings is large. Moreover, we did not implement many of the biasing features of SVD and timeSVD, which will improve RMSE but will create an unbalanced comparison. Future work on CKF includes integrating regularized user and item biases, as well as idiosyncratic deviations in time.

VII. Conclusion

Recommendation systems and algorithms for business and commerce have dual objectives of providing excellent prediction accuracy and positive user experience to enable long-term revenue achievement from customers [25]. By taking temporal dynamics into account, we can contribute to both of these objectives by tracking and intelligently forecasting user preference trajectories. However, sophisticated, principled temporal models are required to fit real-world transactional or ratings data.

In this paper, we have proposed an extension to Gaussian PMF to take into account trajectories of user behavior. This has been done using a dynamical state space model from which predictions are made using the Kalman filter. We have derived an expectation-maximization algorithm to learn the parameters of the model from previously collected observations. We have validated the proposed CKF and shown its advantages over SVD and timeSVD on generated data.

Moreover, we have compared the applicability of CKF to learn and react to changing user tastes in the Netflix dataset. We find that CKF can better forecast temporal trends and that this yields improved prediction performance on user ratings than baseline methods. User factors are not necessary static nor are they restricted to evolve in a drift; accounting for more realistic temporal changes can lead to improvement in performance.

In contrast to heuristic and limited prior methods that incorporate time dynamics in recommendation, the approach proposed in this paper is a principled formulation that can take advantage of decades of developments in tracking and algorithms for estimation. To break away from linearity assumptions, the extended or unscented Kalman filter can be used. Particle filtering can be used for non-Gaussian distributions, analogous to sampling-based inference in Bayesian PMF [23].

There are several directions of future work in improving CKF. First, approximations to the Kalman filtering steps may lead to faster computation of user factor estimates and improve the runtime of the algorithm. Second, it is possible to utilize the existing Kalman filtering literature to address the cold start problem, where new users or items are introduced to the recommendation system [26]. Third, the assumptions that users should be homogeneous enough to share common temporal trajectories suggest that CKF can be effectively combined with nearest-neighbor recommendation models or multi-task learning to yield more effective predictions, beyond just looking at single-genre data. Finally, since there have been advances in the state of the art in non-dynamic matrix factorization, e.g. [27], [28], future research should combine these advances with the state space dynamics for even more powerful modeling.

In future work, we can also consider applications of matrix factorization besides recommendation systems. Matrix factorization is used in e.g., image impainting, blind source separation, and financial modeling. With the dynamical matrix factorization proposed herein, we could focus on impainting motion pictures to alleviate scratches on films, we could separate audio sources in dynamic environments such as those needed for hearing aids, and we could track evolving financial factor models [29].

Appendix A

Useful Facts From Matrix Theory

We present some useful facts for the derivations in Appendix B [30]:

Fact 1: For $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$,

$$\mathrm{E}\left[(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right] = \mathrm{tr}\left(\Sigma^{-1}\mathrm{E}\left[(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T\right]\right).$$

Fact 2:

$$\frac{d\log|X|}{dX} = \left(X^T\right)^{-1} = \left(X^{-1}\right)^T.$$

Fact 3:

$$\frac{d\mathrm{tr}(X^{-1}A)}{dX} = -\left(X^{-1}AX^{-1}\right)^T.$$

Fact 4: For square matrices A and B,

$$\frac{d\mathrm{tr}(AX^T)}{dX} = A.$$

Fact 5: For square matrices A and B,

$$\frac{d\mathrm{tr}(AXBX^TC)}{dX} = A^TC^TXB^T + CAXB.$$

Appendix B

Determining EM parameters

We now derive the EM-parameter equations given in (17)–(21). In the maximization step of the EM algorithm,

we solve for parameters that maximize the expected joint likelihood:

$$\hat{\theta}^{(n+1)} = \arg\max_{\theta} \mathrm{E}\left[p_{\mathbf{x},\mathbf{y}}(\mathbf{x}, y; \theta) \mid \mathbf{y} = y;\ \hat{\theta}^{(n)}\right], \quad (22)$$

where $\hat{\theta}^{(n)}$ is the guess of the true parameter set on the nth iteration. It is common to consider the log-likelihood to change the products in the joint likelihood to summations; the maximizing parameters are the same for either optimization. The below derivations reference proofs in [10, Chap. 13] and [31].

A. Simplification of Log-Likelihood

For CKF, the log-likelihood simplifies to

$$\begin{aligned}\log L &= \log p(x, y; \theta) \\ &= \underbrace{\sum_{i=1}^{N} \log p(x_{i,0})}_{L_1} + \underbrace{\sum_{i=1}^{N}\sum_{t=1}^{T} \log p(x_{i,t}|x_{i,t-1})}_{L_2} \\ &\quad + \underbrace{\sum_{i=1}^{N}\sum_{t=1}^{T} \log p(y_{i,t}|x_{i,t})}_{L_3}, \end{aligned} \quad (23)$$

with

$$\begin{aligned} p(x_{i,0}) &\sim \mathcal{N}(x_{i,0};\ \mu_i, \Sigma_i), \\ p(x_{i,t}|x_{i,t-1}) &\sim \mathcal{N}(x_{i,t};\ A_{i,t}x_{i,t-1}, Q_i), \\ p(y_{i,t}|x_{i,t}) &\sim \mathcal{N}(y_{i,t};\ H_{i,t}x_{i,t}, R_i). \end{aligned}$$

Using Fact 1 from Appendix A, the first term L_1 becomes

$$\begin{aligned}\mathrm{E}[L_1] = &-\frac{NK}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{N}\log|\Sigma_i| \\ &-\frac{1}{2}\sum_{i=1}^{N}\mathrm{tr}\left(\Sigma_i^{-1}\mathrm{E}\left[(\mathbf{x}_{i,0}-\mu_i)(\mathbf{x}_{i,0}-\mu_i)^T\right]\right). \end{aligned} \quad (24)$$

We then use the identity

$$\mathbf{x}_{i,0} - \mu_i = (\mathbf{x}_{i,0} - \hat{\mathbf{x}}_{i,0|T}) + (\hat{\mathbf{x}}_{i,0|T} - \mu_i)$$

and note that estimation error and innovation of a Kalman filter are uncorrelated to rewrite the expectation of L_1 to be

$$\begin{aligned}\mathrm{E}[L_1] = &\ c_1 - \frac{1}{2}\sum_{i=1}^{N}\log|\Sigma_i| \\ &-\frac{1}{2}\sum_{i=1}^{N}\mathrm{tr}\left(\Sigma_i^{-1}\left(P_{i,0|T} + (\hat{\mathbf{x}}_{i,0|T}-\mu_i)\cdot(\hat{\mathbf{x}}_{i,0|T}-\mu_i)^T\right)\right). \end{aligned} \quad (25)$$

We repeat the analysis for L_2 using the identity

$$\begin{aligned}\mathbf{x}_{i,t} - A_{i,t}\mathbf{x}_{i,t-1} &= \mathbf{x}_{i,t} - A_{i,t}\mathbf{x}_{i,t-1} + \hat{\mathbf{x}}_{i,t|T} - \hat{\mathbf{x}}_{i,t|T} \\ &\quad + A_{i,t}\hat{\mathbf{x}}_{i,t-1|T} - A_{i,t}\hat{\mathbf{x}}_{i,t-1|T} \\ &= \left(\hat{\mathbf{x}}_{i,t|T} - A_{i,t}\hat{\mathbf{x}}_{i,t-1|T}\right) - \left(\hat{\mathbf{x}}_{i,t|T} - \mathbf{x}_{i,t}\right) \\ &\quad + A_{i,t}\left(\hat{\mathbf{x}}_{i,t-1|T} - \mathbf{x}_{i,t-1}\right). \end{aligned}$$

We then rewrite the expectation of L_2 as

$$\begin{aligned}\mathrm{E}[L_2] = &-\frac{NTK}{2}\log(2\pi) - \frac{T}{2}\sum_{i=1}^{n}\log|Q_i| \\ &-\frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(Q_i^{-1}\mathrm{E}\left[(\mathbf{x}_{i,t}-A_{i,t})(\mathbf{x}_{i,t}-A_{i,t})^T\right]\right) \\ = &\ c_2 - \sum_{i=1}^{n}\frac{T}{2}\log|Q_i| \\ &-\frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(Q_i^{-1}\mathrm{E}\left[\left\{(\hat{\mathbf{x}}_{i,t|T} - A_{i,t}\hat{\mathbf{x}}_{i,t-1|T})\right.\right.\right. \\ &\qquad -(\hat{\mathbf{x}}_{i,t|T}-\mathbf{x}_{i,t})+A_{i,t}(\hat{\mathbf{x}}_{i,t-1|T}-\mathbf{x}_{i,t-1})\big\} \\ &\qquad \cdot\big\{(\hat{\mathbf{x}}_{i,t|T} - A_{i,t}\hat{\mathbf{x}}_{i,t-1|T})-(\hat{\mathbf{x}}_{i,t|T} - \mathbf{x}_{i,t}) \\ &\qquad \left.\left.\left. +A_{i,t}(\hat{\mathbf{x}}_{i,t-1|T} - \mathbf{x}_{i,t-1})\big\}^T\right]\right). \end{aligned} \quad (26)$$

Expanding everything and again noting that the Kalman estimation error and innovation are uncorrelated, (26) simplifies to

$$\begin{aligned}\mathrm{E}[L_2] = &\ c_2 - \frac{T}{2}\sum_{i=1}^{n}\log|Q_i| \\ &-\frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{tr}\left(Q_i^{-1}\left\{\left(P_{i,t|T} + \hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t|T}^T\right)\right.\right. \\ &\qquad -2\left(P_{i,t,t-1|T} + \hat{\mathbf{x}}_{i,t|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right)A_{i,t}^T \\ &\qquad \left.\left. +A_{i,t}\left(P_{i,t-1|T}+\hat{\mathbf{x}}_{i,t-1|T}\hat{\mathbf{x}}_{i,t-1|T}^T\right)A_{i,t}^T\right\}\right). \end{aligned} \quad (27)$$

A similar derivation is employed for L_3 utilizing

$$y_{i,t} - H_{i,t}\mathbf{x}_{i,t} = (y_{i,t} - H_{i,t}\hat{\mathbf{x}}_{i,t|T}) + H_{i,t}(\hat{\mathbf{x}}_{i,t|T} - \mathbf{x}_{i,t}). \quad (28)$$

Some care is needed in writing R_i in L_3 since $y_{i,t}$ can be of different lengths depending on the observation tensor O and hence a subset of the noise covariance matrix is needed at each time step. To resolve this, we define a fill function that expands the observation vector back to $\mathbb{R}^M$ and a diagonal binary matrix $J_{i,t} \in R^{M\times M}$ with ones in the diagonal positions where ratings are observed for user i at time t.

Currently, the formulation is extremely general and parameters may change with users and in time. We can maximize with respect to the log-likelihood but the resulting estimation would be poor and does not exploit the possible similarities between a population of users. To fully realize the benefits of CKF, we make simplifying assumptions that $\mu_i = 0$, $\Sigma_i = \Sigma$, $A_i = A$, $Q_i = Q$, and $R_i = \sigma_R^2 I_M$. We now move summations into the trace operator and the log-likelihood simplifies to

$$\mathrm{E}[L_1] = c_1 - \frac{N}{2}\log|\Sigma| - \frac{1}{2}\mathrm{tr}(\Sigma^{-1}\Gamma_1), \quad (29)$$

$$\mathrm{E}[L_2] = c_2 - \frac{NT}{2}\log|Q| - \frac{1}{2}\mathrm{tr}(Q^{-1}\Gamma_2), \quad (30)$$

$$\mathrm{E}[L_3] = c_3 - \frac{|\mathcal{O}|}{2}\log\sigma_R^2 - \frac{1}{2\sigma_R^2}\mathrm{tr}(\Gamma_3), \quad (31)$$

where

$$\Gamma_1 = \sum_{i=1}^{N} \left(P_{i,0|T} + \hat{\mathbf{x}}_{i,0|T} \hat{\mathbf{x}}_{i,0|T}^T \right), \tag{32}$$

$$\begin{aligned} \Gamma_2 = \sum_{i=1}^{N} \sum_{t=1}^{T} \Big\{ & \left(P_{i,t|T} + \hat{\mathbf{x}}_{i,t|T} \hat{\mathbf{x}}_{i,t|T}^T \right) \\ & - 2 \left(P_{i,t,t-1|T} + \hat{\mathbf{x}}_{i,t|T} \hat{\mathbf{x}}_{i,t-1|T}^T \right) A_t^T \\ & + A_t \left(P_{i,t-1|T} + \hat{\mathbf{x}}_{i,t-1|T} \hat{\mathbf{x}}_{i,t-1|T}^T \right) A_t^T \Big\}, \end{aligned} \tag{33}$$

$$\begin{aligned} \Gamma_3 = \sum_{i=1}^{N} \sum_{t=1}^{T} \Big\{ & \text{fill}(y_{i,t}) \text{fill}(y_{i,t})^T - 2 \text{fill}(y_{i,t}) \hat{\mathbf{x}}_{i,t|T}^T V^T \\ & + (J_{i,t} V) \left(P_{i,t|T} + \hat{\mathbf{x}}_{i,t|T} \hat{\mathbf{x}}_{i,t|T}^T \right) (J_{i,t} V)^T \Big\}. \end{aligned} \tag{34}$$

B. Determining $\widehat{\Sigma}$, $\widehat{Q}$ and $\widehat{R}$

To maximize with respect to Σ, we can differentiate (29), set to zero, and solve. Using Facts 2 and 3 from Appendix A,

$$\frac{\partial \mathrm{E}[L_1]}{\partial \Sigma} = -\frac{N}{2} (\Sigma^{-1})^T + \frac{1}{2} (\Sigma^{-1} \Gamma_1 \Sigma^{-1})^T, \tag{35}$$

and solving gives

$$\widehat{\Sigma} = \frac{1}{N} \Gamma_1. \tag{36}$$

If we had further assumed that $\Sigma = \sigma_U^2 I_K$, then (29) would simplify to

$$\mathrm{E}[L_1] = c_1 - \frac{NK}{2} \log \sigma_U^2 - \frac{1}{2\sigma_U^2} \mathrm{tr}(\Gamma_1),$$

and maximization yields (17).

The derivations for $\widehat{Q}$ and $\widehat{R}$ follow similarly and lead to (18) and (19) respectively.

C. Determining $\widehat{A}$ and $\widehat{V}$

Rewriting (30) as

$$\mathrm{E}[L_2] = c_A + \mathrm{tr}(Q^{-1} A_2 A^T) - \frac{1}{2} \mathrm{tr}(Q^{-1} A A_1 A^T), \tag{37}$$

where c_A is the collection of terms that do not depend on A, we maximize using the same procedure as for $\widehat{\Sigma}$. We utilize Facts 4 and 5 while noting that A_1, A_2 and $\widehat{Q}$ are symmetric and invertible, and the maximization yields (20).

Following a similar procedure for optimization of $\widehat{V}$, we express (31) as

$$\begin{aligned} \mathrm{E}[L_3] = c_V + \frac{1}{\sigma_R^2} \mathrm{tr}(V_2 V^T) \\ - \frac{1}{2\sigma_R^2} \mathrm{tr} \left(\sum_{i=1}^{N} \sum_{t=1}^{T} J_{i,t} V V_{i,t} V^T J_{i,t}^T \right). \end{aligned} \tag{38}$$

In this case, V cannot be expressed as a matrix product, but each row can. Noting $J_{i,t} = J_{i,t}^T$ and $J_{i,t} J_{i,t} = J_{i,t}$, the maximization over each row yields (21).

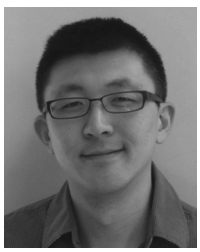
ACKNOWLEDGMENT

The authors thank K. Subbian for discussions, V. K Goyal and A. Mojsilović for support, and the anonymous reviewers for many helpful comments.

REFERENCES

[1] "From stretched to strengthened: Insights from the global chief marketing officer study," IBM Corp., Somers, NY, Oct. 2011.
[2] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," [Online]. Available: http://arxiv.org/pdf/1205.3193 May 2012
[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[4] N. Srebro, "Learning with matrix factorizations," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, USA, 2004.
[5] J. Z. Sun, K. R. Varshney, and K. Subbian, "Dynamic matrix factorization: A state space approach," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 1897–1900.
[6] J. Z. Sun, K. R. Varshney, and K. Subbian, "Dynamic matrix factorization: A state space approach," [Online]. Available: http://arxiv.org/pdf/1110.2098 Aug. 2012
[7] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2008, vol. 20, pp. 1257–1264.
[8] A. E. Bryson, Jr. and Y.-C. Ho, *Applied Optimal Control*. Waltham, MA, USA: Ginn and Company, 1969.
[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.—T. ASME*, vol. 82, no. 1, pp. 35–45, Mar. 1960.
[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[11] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA J.*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965.
[12] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Las Vegas, NV, USA, Aug. 2008, pp. 426–434.
[13] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer, 2011, pp. 257–297.
[14] Y. Bar-Shalom, "Optimal simultaneous state estimation and parameter identification in linear discrete-time systems," *IEEE Trans. Autom. Control*, vol. AC-17, no. 3, pp. 308–319, Jun. 1972.
[15] Y. Koren, "Collaborative filtering with temporal dynamics," *Commun. ACM*, vol. 53, no. 4, pp. 89–97, Apr. 2010.
[16] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *Proc. SIAM Int. Conf. Data Min.*, Columbus, OH, Apr.–May 2010, pp. 211–222.
[17] Z. Lu, D. Agarwal, and I. S. Dhillon, "A spatio-temporal approach to collaborative filtering," in *Proc. ACM Conf. Recommender Syst.*, New York, NY, USA, Oct. 2009, pp. 13–20.
[18] S. Nowakowski, C. Bernier, and A. Boyer, "Target tracking in the recommender space: Toward a new recommender system based on Kalman filtering," Nov. 2010 [Online]. Available: http://arxiv.org/pdf/1011.2304.
[19] N. Sahoo, P. V. Singh, and T. Mukhopadhyay, "A hidden Markov model for collaborative filtering," in *Proc. Winter Conf. Business Intell.*, Salt Lake City, UT, USA, Mar. 2011.
[20] N. Lathia, S. Hailes, and L. Capra, "Temporal collaborative filtering with adaptive neighbourhoods," in *Proc. ACM SIGIR Conf. Res. Dev. Inf.Retrieval*, Boston, MA, USA, Jul. 2009, pp. 796–797.
[21] R. Li, B. Li, C. Jin, X. Xue, and X. Zhu, "Tracking user-preference varying speed in collaborative filtering," in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Aug. 2011, pp. 133–138.
[22] J. A. Konstan and J. Riedl, "Recommended for you," *IEEE Spectrum*, vol. 49, no. 10, pp. 54–61, Oct. 2012.
[23] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 880–887.
[24] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Seattle, WA, USA, Aug. 2004, pp. 109–117.

[25] J. A. Konstan and J. Riedl, "Recommender systems: From algorithms to user experience," *User Model. User-Adap.*, vol. 22, no. 1–2, pp. 101–123, Apr. 2012.
[26] T. Jambor, J. Wang, and N. Lathia, "Using control theory for stable and efficient recommender systems," in *Proc. Int. World Wide Web Conf.*, Lyon, France, Apr. 2012, pp. 11–20.
[27] L. W. Mackey, A. S. Talwalkar, and M. I. Jordan, "Divide-and-conquer matrix factorization," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, Dec. 2011, vol. 24, pp. 1134–1142.
[28] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, Jun. 2013, vol. 2, pp. 82–90.
[29] A. Y. Aravkin, K. R. Varshney, and D. M. Malioutov, "Dynamic factor modeling via robust subspace tracking," presented at the Industr.-Acad. Workshop Optim. Finance Risk Manag., Toronto, ON, Canada, Sep. 2013.
[30] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Nov. 2008.
[31] T. Rosenbaum and A. Zetlin-Jones, "The Kalman filter and the EM algorithm," Dec. 2006.



John Z. Sun received the B.S. degree with honors in electrical and computer engineering (*summa cum laude*) from Cornell University, Ithaca, NY, in 2007. He received the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2009. He completed his Ph.D. degree in electrical engineering and computer science from MIT in 2013.

Dr. Sun was awarded the student best paper award at IEEE Data Compression Conference in 2011 and was the recipient of the Claude E. Shannon Research Assistantship in 2011. In 2013, he received the inaugural EECS Paul L. Penfield Student Service Award. His research interests include signal processing, information theory, and statistical inference.

Dhruv Parthasarathy received dual B.S. degrees in mathematics and electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2013. He is currently completing his M.S. degree in electrical engineering and computer science from MIT, where he works with Professor Devavrat Shah in the Laboratory of Information and Decision Systems.

Kush R. Varshney (S'00–M'10) was born in Syracuse, NY, in 1982. He received the B.S. degree (*magna cum laude*) in electrical and computer engineering with honors from Cornell University, Ithaca, NY, in 2004 and the S.M. and Ph.D. degrees, both in electrical engineering and computer science, from the Massachusetts Institute of Technology (MIT), Cambridge, in 2006 and 2010, respectively.

He is a research staff member in the Mathematical Sciences and Analytics Department at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY. While at MIT, he was a research assistant with the Stochastic Systems Group in the Laboratory for Information and Decision Systems and a National Science Foundation Graduate Research Fellow. His research interests include statistical signal processing, statistical learning, data mining, and image processing.

Dr. Varshney is a member of Eta Kappa Nu and Tau Beta Pi. He received a Best Student Paper Travel Award at the 2009 International Conference on Information Fusion, the Best Paper Award at the 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, and several IBM awards for contributions to business analytics projects. He is on the editorial board of *Digital Signal Processing* and a member of the IEEE Signal Processing Society Machine Learning for Signal Processing Technical Committee.