Matrix Factorization with Time Dynamics

Umang Patel
Department of Computer Science
Columbia University, New York, NY
ujp2001@columbia.edu

Ilambharathi Kanniah
Department of Computer Science
Columbia University, New York, NY
ik2342@columbia.edu

Abstract

Recommendation problem is a well looked into topic whose applications include movies recommendation (NetFlix Challenge), products/services recommendation. We propose a model with extension suited for use in movie recommendations (collaborative filtering type) which includes the time dynamics of user state into consideration. The model is an extension to probabilistic matrix factorization with the estimation of user-state done with Kalman filter and smoother (extended to Ensemble Kalman filter and smoother) which are part of an expectation-maximization algorithm. The above model was run on a generated dataset and the extensions of Ensemble Kalman Filter and Smoother and the Initial clustering of users. All these methods performed good compared to the baseline.

1 Introduction

The movie recommendation problem is a heavily looked into problem. The problem being given a sparse matrix of movie ratings by users on movies, new recommendations based on user/item similarity needs to be provided to the user requesting for it. The Netflix Prize Challenge had many enthusiastic teams to come up with many innovative algorithms to solve this problem. The most famous one begin Collaborative Filtering via Matrix Factorization where the very sparse matrix is factored into user-factor matrix and item-factor matrix with K latent factors (the K decides the amount of total information to be captured in these two matrices). Approaches like SVD, Probabilistic Matrix Factorization (PMF)[10], Bayesian Probabilistic Matrix Factorization (B-PMF)[9] etc were introduced to solve the matrix factorization problem efficiently.
Few of the known MF-based algorithms to look at time factor are timeSVD, but the timeSVD approach is limited to temporal dynamics (where the user-factor can deviate from a central point of time linearly) rather than a global one. Another looks at the user-space variation in the target recommendation space but do not have a strong literature over the same [6]. There is a HMM based model for movie recommendation also, but doesn't include the process and measurement noises[8].
In this project, we went on to explore the effects of user-time dynamics on the movie recommendations globally. We know that the state of the user-factor (i.e the taste/preference of the user) changes with time. Hence, assuming that the user-factor is a constant across time is a great simplifying assumption done by many algorithms[12][11]. The vanilla MF-based collaborative filtering algorithms do not take into account the change of states through time.The Kalman filter [3][13] and smoother are efficient algorithms to estimate the state of a dynamic system given noisy measurements.
In this project, we propose extensions to a space-state model approach where the probabilistic MF was extended to include the evolution of the state-space across time and the states are estimated using a Kalman Filter and RTS smoother. The extension includes using *Ensemble Kalman Filter (EnKF) and Ensemble Kalman Smoother (EnKS)*[7] instead of the basic Kalman Filter and RTS Smoother. We also did an *initial clustering of users* into user-clusters then estimate the user and item factor matrices independently so that the effects of varying users across user-clusters are reduced and would give *improved performance*. We compared the extensions with the minimization

of the objective function (using CVX).

2 Problem Formulation

N - Number of Users, M - Number of Movies
$O \in \mathbb{R}^{NxM}$ (Sparse Matrix of Entries)
Matrix Factorization (MF) - $U \in \mathbb{R}^{NxK}$ and $V \in \mathbb{R}^{MxK}$ where there are K latent factors.
Then, the preference (ratings) matrix,

$$O = UV^T$$

Given K,

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

for i=1 to N and j=1 to M

One class of algorithms called as collaborative filtering algorithms are used extensively to do recommendation (say in Netflix or similar) and the algorithms involve use of expensive computations like SVD (Singular Value-Decomposition) which do not take into consideration the time factor evolution. Now, the problem formulation including the time component would be,

N - Number of Users, M - Number of Movies, T - Number of Time Intervals
$O \in \mathbb{R}^{TxNxM}$ (Sparse Matrix of Entries)
Matrix Factorization (MF) - $U \in \mathbb{R}^{TxNxK}$ and $V \in \mathbb{R}^{TxMxK}$ where there are K latent factors.

$$\min_{U,V,T} \sum_{(t,i,j)\in O} (o_{tij} - u_{ti} v_{tj}^T)^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2$$

for i=1 to N and j=1 to M and t=1 to T
This modified problem formulation includes for the user and item factor to evolve across time (even though the user and/or item factor may remain constant over some period of time)

3 Probabilistic Matrix Factorixation

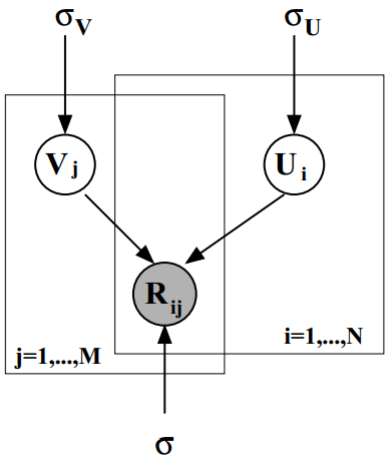


Figure 1: Probabilistic Matrix Factorization

The Probabilistic Matrix Factorization is a good approach to Matrix Factorization from a Bayesian viewpoint. This model is fit to data by finding a MAP estimate of the model parameters (this can scale very well on large datasets too). Also, unless the regularization parameters λ_1, λ_2 are tuned carefully, the results won't be good. This model alleviates that problem by using a Bayesian approach of MAP estimation.

The above plate diagram explains the model in whole. The ratings R_{ij} is the sparse observation matrix, which has parents from both the User-factors (U_i) and Item/Movie factors (V_j). The state factors (both user and item factors) have priors on them.

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} [\mathcal{N}(R_{i,j}|U_i^T V_j, \sigma^2)]^{I_{i,j}}$$

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} [\mathcal{N}(U_i|0, \sigma_U^2 I)]$$

$$p(V|\sigma_V^2) = \prod_{j=1}^{M} [\mathcal{N}(V_j|0, \sigma_V^2 I)]$$

The above equations can be iteratively solved by using a EM like algorithm where the $p(R|U, V, \sigma^2)$ is expected and $p(U|\sigma_U^2), p(V|\sigma_V^2)$ are maximized (the user factors are sampled from a Gaussian distribution with zero mean and variance σ_U^2 and the same goes with the item factors too) . We will in the next section as to how this model is further extended.

4 Extension to KF in EM - Factorization via Learning

Now, in this section we see how the Probabilistic Matrix Factorization can be extended to also consider the user-state time dynamics. Let us assume that the user-state evolution is a linear transformation process (A) . So, if we know the underlying linear transformation, we could directly estimate the user-state by using the observed values (Kalman Filter and Smoother), but we do not know the underlying transformation matrix, process noise matrix and measurement noise matrix. Instead, we expect the user-state estimation and maximize the underlying model parameters through an Expectation-Maximization Algorithm. We will also see that the following algorithm is efficient in practice since the E-Step can be run in parallel for each user using the same global values maximized in the M-Step.

Expectation - Maximization Algorithm
Input : Sparse Observation Matrix Y E-Step

- Initialize User States from $\mathcal{N}(0, \hat{\sigma}_U^2)$
- Process Noise from $\mathcal{N}(0, \hat{\sigma}_Q^2)$ and Measurement Noise from $\mathcal{N}(0, \hat{\sigma}_R^2)$
- Forward Pass - Kalman Filter, Backward Pass - RTS (Rauch-Tung-Striebel) Smoother (since all observations w.r.t users and time are available) using the sparse Observation Matrix Y.

M-Step

- Maximize Log Likelihood to get maximized estimates for $\hat{\sigma}_U^2$ (Variance of Initial States)
- $\hat{\sigma}_Q^2$ (Variance for Process Noise Model), $\hat{\sigma}_R^2$ (Variance for Measurement Noise Model)
- $\hat{A}$ - Process Transition Matrix
- $\hat{V}$ - Observation Matrix (Movie Factor Matrix).

Output : The estimated User-States $\hat{U}$ and estimated complete Observation Matrix $\hat{Y}$

Algorithm: Expectation Maximization Algorithm with E-Step having Kalman Filter and Smoother

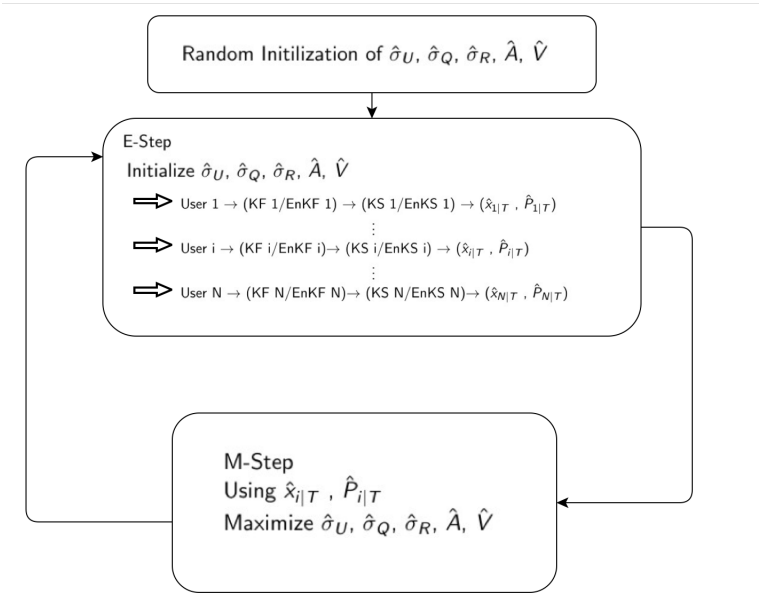


Figure 2: Process Diagram involving KF/enKF and KS/enKS in EM

4.1 Notation

$\hat{x}_{i,t+1|t}$ (Prior)- User State Estimate of User i at time t+1 given all observations till time t (inclusive). $\hat{x}_{i,t|t}$ (Posterior) - User state Estimate of User i at time t given all observations till time t (inclusive). $\hat{P}_{i,t+1|t}$ (Prior)- Covariance Estimate of User i at time t+1 given all observations till time t (inclusive). $\hat{P}_{i,t|t}$ (Posterior) - Covariance Estimate of User i at time t given all observations till time t (inclusive).

$H_{i,t}$ is the subset of rows from V for which the observations $y_{i,t}$ is available or observed (therefore, $H_{i,t}$ is much smaller compared to V)

4.2 Extension to KF in EM - E Step

The whole optimization process was split into the EM algorithm. In the E-Step, we have the Kalman Filter equations and the Kalman Smoothing Equations. We model this problem, with the assumption that the underlying user factor evolve linearly with time through a linear transformation. Here, $x_{i,t|t}$ represents the user-factor of user i at time t. The next user-factor in time $x_{i,t+1|t}$ is determined by applying the linear transformation $A_{i,t+1}$ on the current user-factor, i.e. $x_{i,t|t}$. The covariances $P_{i,t+1|t}$ are also suitably updated. The Kalman Gain ($K_{i,t}$) is calculated and used to update the posterior $x_{i,t|t}$ given the prior $x_{i,t|t-1}$ and Kalman Gain, $K_{i,t}$. The posterior covariance is also updated similarly. Here, the important point to notice is that the $H_{i,t}$ is a subset of V where the subset is taken only for the entries in $y_{i,t}$ which is a sparse vector (since it is part of the sparse matrix). The user-factor is then transformed linearly through the process transformation $H_{i,t} * x_{i,t}$ which follows similarly to $V^T * U$. The next set of equations are the Kalman Smoothing equations which are in (5). Also, the calculation of the covariance lags are made after the Kalman Smoothing step.

E-Step Forward Pass - Kalman Filter

$$\begin{aligned}
\hat{x}_{i,t+1|t} &= A_{i,t+1}\hat{x}_{i,t|t} \\
P_{i,t+1|t} &= A_{i,t+1}P_{i,t|t}A_{i,t+1}^T + Q_{i,t} \\
K_{i,t} &= P_{i,t|t-1}H_{i,t}(H_{i,t}P_{i,t|t-1}H_{i,t}^T + R_{i,t})^{-1} \\
\hat{x}_{i,t|t} &= \hat{x}_{i,t,|t-1} + K_{i,t}(y_{i,t} - H_{i,t}\hat{x}_{i,t|t-1}) \\
P_{i,t|t} &= P_{i,t|t-1} - K_{i,t}H_{i,t}P_{i,t|t-1}
\end{aligned} \tag{1}$$

Backward Pass - RTS Smoothing

$$\begin{aligned} J_{i,t} &= P_{i,t|t} A_{i,t+1}^T P_{i,t|t-1}^{-1} \\ \hat{x}_{i,t|T} &= \hat{x}_{i,t|t} + J_{i,t}(\hat{x}_{i,t+1|T} - \hat{x}_{i,t+1|t}) \\ P_{i,t|T} &= P_{i,t|t} + J_{i,t}(P_{i,t+1|T} - P_{i,t+1|t}) J_{i,t}^T \\ P_{i,T,T-1|T} &= (I - K_{i,t} H_{i,t}) A_{i,t} P_{i,T-1|T-1} \\ P_{i,t,t-1|T} &= P_{i,t|t} J_{i,t-1}^T + J_{i,t}(P_{i,t+1,t|T} - A_{i,t+1} P_{i,t|t}) J_{i,t-1}^T \end{aligned} \tag{2}$$

M-Step - The derivations of the M-Step are mentioned in the Appendix Section. After the M-step where the parameters maximizing the log-likelihood has been derived, we obtain estimates for A,V,$\hat{\sigma_U^2}$, $\hat{\sigma}_Q^2$,$\hat{\sigma}_R^2$. The estimates are obtained from the smoothed $x_{i,\hat{t|T}}$, $P_{i,\hat{t|T}}$ and P lag.

$$\hat{\sigma}_U^2 = \frac{1}{NK} \sum_{i=1}^{N} tr(P_{i,0|T} + \hat{x}_{i,0|T} \hat{x}_{i,0|T}^T)$$

$$\hat{\sigma}_Q^2 = \frac{1}{NKT} \sum_{i=1}^{N} \sum_{t=1}^{T} tr[(P_{i,t|T} + \hat{x}_{i,t|T} \hat{x}_{i,t|T}^T) - 2(P_{i,t,t-1|T} + \hat{x}_{i,t|T} \hat{x}_{i,t-1|T}^T) A^T + A(P_{i,t-1|T} + \hat{x}_{i,t-1|T} \hat{x}_{i,t-1|T}^T) A^T]$$

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

$$\hat{A} = A_2 A_1^{-1} A_1 = \sum_{i=1}^{N} \sum_{t=1}^{T} (P_{i,t,t-1|T} + \hat{x}_{i,t-1|T} \hat{x}_{i,t-1|T}^T), \quad A_2 = \sum_{i=1}^{N} \sum_{t=1}^{T} (P_{i,t-1|T} + \hat{x}_{i,t|T} \hat{x}_{i,t-1|T}^T)$$

$$\hat{V_j} = V_{2,j} V_1(j)^{-1} \qquad \text{j=1,...,M where}$$

$$V_1(j) = \sum_{i=1}^{N} \sum_{t=1}^{T} 1_{O_{ijt}} (P_{i,t|T} + \hat{x}_{i,t|T} \hat{x}_{i,t|T}^T) \quad , V_2 = \sum_{i=1}^{N} \sum_{t=1}^{T} (fill(y_{i,t}) \hat{x}_{i,t|T}^T)$$

4.3 Simplifying Assumptions

Movie Factor do not change much with time and hence estimated to be $\hat{V}$. All the $A_{i,t}$ are same (i.e. $\hat{A}$). All the process noise variances ($Q_{i,t}$) and measurement noise variances ($R_{i,t}$) are same and equal to $\hat{\sigma}_Q^2$ and $\hat{\sigma}_R^2$ respectively. All the initial user states are initialized from a Gaussian with zero-mean and variance $\hat{\sigma}_U^2$.

These important assumptions make the derivations and the implementation of the Kalman Filter in EM efficient and effective. Since the matrix factorization is from a very sparse matrix, the simplifying assumptions of movie factor not evolving with time, the process and measurement noises are stationary and also from a Gaussian model when in reality the noises needn't be stationary and generally don't follow a Gaussian model. But, in practice, as shown in the numerical results, the model performs well compared to the baseline approach.

We observe that the use of Kalman Filter and Kalman Smoother in E-Step has an good advantage of parallelization. All the "N" Kalman Filters and Kalman Smoothers can be run in parallel since they share the same set of global variables which do not change with respect to time. This alleviates the problem of having many users and the cumulative complexity of time since now, each one of them can be run parallely and independently.

4.4 Extension to KF in EM - E Step - after applying Assumptions

We rewrite the Kalman Filter and Smoother Equations after the assumptions are made to simplify the model.
Forward Pass - Kalman Filter

$$\begin{aligned}
\hat{x}_{i,t+1|t} &= A\hat{x}_{i,t|t} \\
P_{i,t+1|t} &= AP_{i,t|t}A^T + Q \\
K_{i,t} &= P_{i,t|t-1}H(HP_{i,t|t-1}H^T + R)^{-1} \\
\hat{x}_{i,t|t} &= \hat{x}_{i,t,|t-1} + K_{i,t}(y_{i,t} - H\hat{x}_{i,t|t-1}) \\
P_{i,t|t} &= P_{i,t|t-1} - K_{i,t}HP_{i,t|t-1}
\end{aligned} \tag{3}$$

Backward Pass - RTS Smoothing

$$\begin{aligned}
J_{i,t} &= P_{i,t|t}A^T P_{i,t|t-1}^{-1} \\
\hat{x}_{i,t|T} &= \hat{x}_{i,t|t} + J_{i,t}(\hat{x}_{i,t+1|T} - \hat{x}_{i,t+1|t}) \\
P_{i,t|T} &= P_{i,t|t} + J_{i,t}(P_{i,t+1|T} - P_{i,t+1|t})J_{i,t}^T \\
P_{i,T,T-1|T} &= (I - K_{i,t}H)AP_{i,T-1|T-1} \\
P_{i,t,t-1|T} &= P_{i,t|t}J_{i,t-1}^T + J_{i,t}(P_{i,t+1,t|T} - AP_{i,t|t})J_{i,t-1}^T
\end{aligned} \tag{4}$$

5 Extension-Ensemble Kalman Filter and Ensemble Smoother in EM

We make an extension to the above model where we observe that the covariances $P_{i,t+1|t}$ and $P_{i,t|t}$ needn't be explicitly stored as that would take up more memory. To avoid the same we use an ensemble of user-states for each time and each of the ensemble member is updated accordingly. This avoids the need for explicit storage of the covariances instead the ensemble members are used directly in the calculation of the required terms. Here, N_s is the ensemble size. Also, in the Ensemble Smoothing case, the complexity of operations can be reduced by introducing an intermediate variable w

Forward Pass - Ensemble Kalman Filter

$$\begin{aligned}
&\hat{x}_{i,t+1|t} = A_{i,t+1}\hat{x}_{i,t|t} + q_{i,t} \qquad\qquad q_{i,t} \sim N(0, Q_{i,t}) \\
&PH^T = \frac{1}{N_s - 1}\sum_{j=1}^{N_s}(x_{i,t+1|t}^j - \bar{x}_{i,t+1|t})(Hx_{i,t+1|t}^j - \overline{Hx_{i,t+1|t}})^T \\
&HPH^T = \frac{1}{N_s - 1}\sum_{j=1}^{N_s}(Hx_{i,t+1|t}^j - \overline{Hx_{i,t+1|t}})(Hx_{i,t+1|t}^j - \overline{Hx_{i,t+1|t}})^T \\
&K_{i,t} = P_{i,t|t-1}H_{i,t}(H_{i,t}P_{i,t|t-1}H_{i,t}^T + R_{i,t})^{-1} \\
&\hat{x}_{i,t|t} = \hat{x}_{i,t,|t-1} + K_{i,t}(y_{i,t} - H_{i,t}\hat{x}_{i,t|t-1})
\end{aligned} \tag{5}$$

Backward Pass - Ensemble RTS Smoothing

$$\begin{aligned}
P_{i,t|t-1}w &= (\hat{x}_{i,t+1|T} - \hat{x}_{i,t+1|t}) \\
\hat{x}_{i,t|T} &= \hat{x}_{i,t|t} + P_{i,t|t}Aw
\end{aligned} \tag{6}$$

$$\begin{aligned}
P_{i,t|t} &= \frac{1}{N_s}\sum_{j=1}^{N_s}(x_{i,t|t}^j - \bar{x}_{i,t|t})(x_{i,t|t}^j - \overline{x_{i,t|t}})^T \\
P_{i,t|t-1} &= \frac{1}{N_s}\sum_{j=1}^{N_s}(x_{i,t|t-1}^j - \bar{x}_{i,t|t-1})(x_{i,t|t-1}^j - \overline{x_{i,t|t-1}})^T
\end{aligned} \tag{7}$$

6 Initial Clustering of User Matrix and then running KF

Matrix Factorization with Initial User Clustering

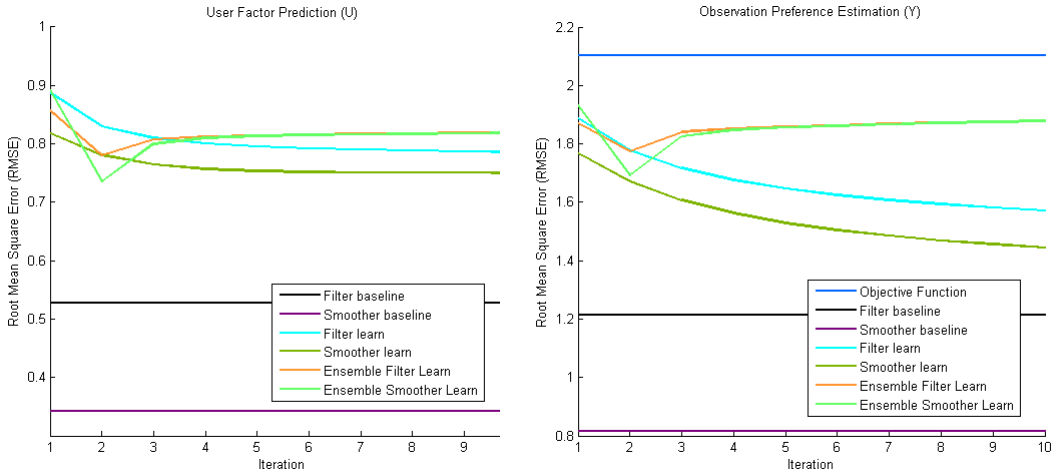
- Input : Sparse Observation matrix Y
- Form initial clusters by decomposing initial observation Ytrue with SVD. Set of Clusters , SC having NC clusters.
- For each cluster in SC set of clusters.
  - E-Step
    - Initialize User States from $\mathcal{N}(0, \hat{\sigma}_U^2)$
    - Process Noise from $\mathcal{N}(0, \hat{\sigma}_Q^2)$ and Measurement Noise from $\mathcal{N}(0, \hat{\sigma}_R^2)$
    - Forward Pass - Ensemble Kalman Filter, Backward Pass - Ensemble Kalman Smoother (since all observations w.r.t users and time are available) using the sparse observation matrix Y
  - M-Step
    - Maximize Log Likelihood to get maximized estimates for $\hat{\sigma}_U^2$ (Variance of Initial States)
    - $\hat{\sigma}_Q^2$ (Variance for Process Noise Model), $\hat{\sigma}_R^2$ (Variance for Measurement Noise Model)
    - $\hat{A}$ - Process Transition Matrix
    - $\hat{V}$ - Observation Matrix (Movie Factor Matrix).
- Combine the estimated observations for all users in the clusters and movies(items) into estimated complete $\hat{Y}$.
- Ouput the estimated complete $\hat{Y}$.

Algorithm: Initial User Clustering with Expectation Maximization

To find initial User similarity using SVD, then run individual KF (and KS) on each cluster and obtain user factors (U_k) and item factors (V_k) for each cluster k and finally merge the $(U_k)(V_k)^T$ for each cluster and calculated the RMSE.
RMSE came out to be 0.43. Baseline RMSE came out to be 0.409 [(N,M,T,K,E) = (100,100,20,5)]
The rationale for this approach was that the initial clustering of users into groups based on user similarity may pave a way for better matrix factorization. Now, in this model, not all users across different clusters do not share the same underlying parameters but different ones for each cluster (particularly H process transformation matrix). In this way, users who are very different from other are allowed to evolve differently in their own cluster having similar users.

7 Implementation

In this implementation of the above proposed extension using EnKF to the model, the Expectation-Maximization was implemented with Kalman Filter (or Ensemble Kalman Filter) and Kalman Smoother (or Ensemble Kalman Smoother) in the E-Step and the log likelihood maximization of the underlying simplified assumption parameters, A,V,$\hat{\sigma}_U^2$,$\hat{\sigma}_Q^2$,$\hat{\sigma}_R^2$. The dataset we used in this implementation is a generative dataset (synthetic one).
The number of latent factors (K) to use was an important choice as it reflects on the time complexity of the problem and also the amount of information to be captured in total (i.e. the more K used leads to capturing more information about the user and movie factors), we chose to have K = 5. This choice was made on the pretext that the observation matrix is very sparse (only $\tilde{2}0\%$ of the entries are filled in).
For the generative dataset (gold standard), the underlying model parameters chosen were, $A_{i,t} = I_{K,K}$,V was randomly initialized ,$\hat{\sigma}_U^2$=1,$\hat{\sigma}_Q^2$=1 ,$\hat{\sigma}_R^2$=1 as was done in the proposed model.
The initial clustering of the users were made by taking an SVD on the sparse observation matrix and applying K-Means on the User-factor Matrix to obtain the initial clustering.



(a) User Factor Prediction (b) Observation Preference Estimation

Figure 3: E-Step Parameter Predictions - (N,M,T,K,E) = (300,200,20,20,10)

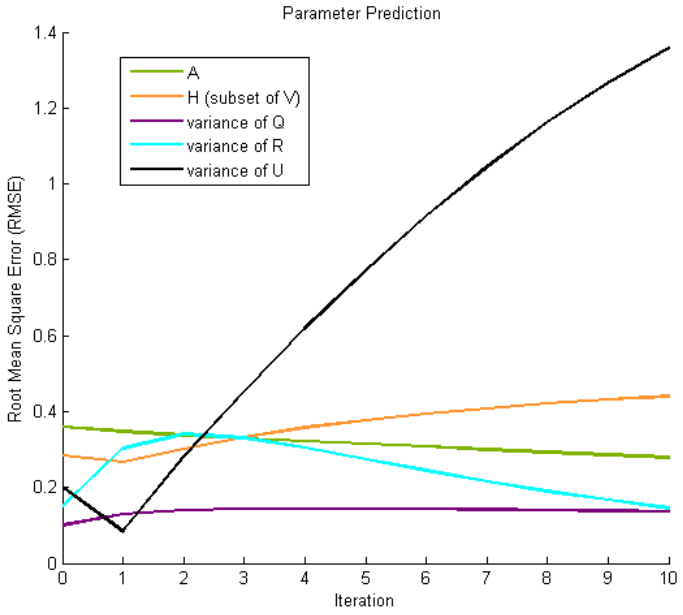
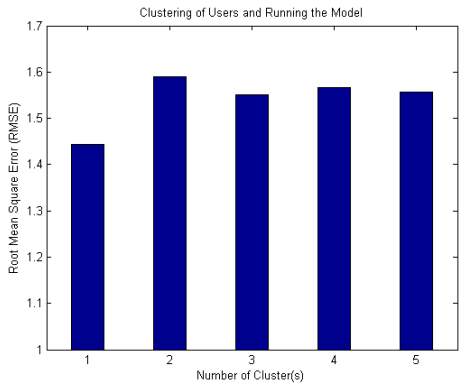


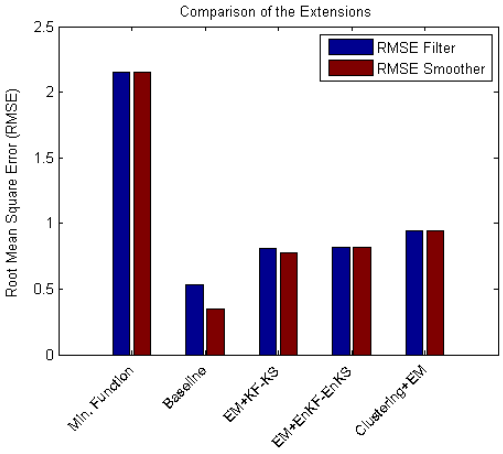
Figure 4: M-Step Parameter Predictions - (N,M,T,K,E) = (300,200,20,20,10)

Table 1: Time Taken and Final RMSE values for a given run on the dataset (N,M,T,K,E) = (300,200,20,20,10)

Method Run	**Time Taken (seconds)**	**RMSE (Filter)**	**RMSE (Smoother)**
Stochastic Gradient Descent	165.2428	2.15	2.15
KF Baseline and KF Smoother Baseline	0.285073	0.53	0.345
KF and KS	5.041342	0.81	0.773
EnKF and EnKS	46.705548	0.82	0.82
Clustering of Users (3 clusters)	246.705548	0.94	0.94



(a) Initial Clustering of Users and then running the model on each cluster(s)



(b) Comparison Graph

8 Numerical Results

Methods (and Extensions considered)

- Objective Minimization (Using solvers e.g. CVX)
- Kalman Filter and Kalman Smoother
- Ensemble Kalman Filter and Ensemble Kalman Smoother
- Initial Clustering of Users.

The code was run with the following parameters - (N,M,T,K,E) = (300,200,20,20,10). The stochastic gradient descent algorithm with tuned regularizers to solve the objective function directly yields a bad RMSE (2.15) and also takes a long time to converge (165.24 second(s)) (Though using Parallelized Stochastic gradient descent, the complexity time can be reduced [14]). We could see that the RMSE values are lower when the underlying model parameters are known. Also, the Kalman Filter does only a forward pass and hence its RMSE is greater than the RMSE of Kalman Smoother in every iteration to follow. The same result applies to Ensemble Kalman Filter and Ensemble Kalman Smoother. The time taken by Kalman Filter-Kalman Smoother combination is the least with 5.041 second(s) followed by Ensemble Kalman Filter- Ensemble Kalman Smoother combination with 46.7055 second(s). The advantage of using EnKF-EnKS would be the alleviating the need to store the covariances explicitly and hence less memory footprint. The initial clustering of users into a fixed number of clusters and the results are illustrated in the graph below. Even though, the model is run independently for each cluster(s), we see very good and comparable performance to the baseline (where the baseline is having only one cluster). The reason for a slightly higher RMSE would be the fact that now each cluster is working with much lesser data in them.
We also observe that the comparison of final RMSE values across the various methods and extensions. The method involving the minimization function directly using CVX and other solvers is not only slow but also having poor RMSE. The Baseline RMSE is having an RMSE $\tilde{0}.6$ because though we assume we know the underlying model, we apply the simplifying assumptions which increases the RMSE slightly. The EM with Ensemble Kalman Filter and Ensemble Kalman Smoother fairs really well compared to the baseline (also, it is memory saving). Using more ensemble members will finally approach the vanilla Kalman Filter and Smoother.

9 Conclusions

Recommendation systems play a very important role in the user-experience enhancement and also generate revenue on the long run (for example, by recommending the right products/services at the right time, the customer tends to continue using the recommendation service and hence revenue is generated).
In this paper, we give importance to the user time dynamics which is ignored in other models like SVD, PMF etc. The timeSVD model also includes a time-factor around a central time but ignores the evolution of user-factor across in a dynamic fashion. The above model proposed effectively captures the user-factor state movement through time and the simplifying assumptions are practically efficient and effective.
The extension of Expectation-Maximization using Kalman Filter and Smoother to Expectation-Maximization using Ensemble Kalman Filter and Ensemble Kalman Smoother was new and sounds promising. Also the extension involving the running the same model on user clusters (where the users are clustered based on initial observation matrix).

10 Discussion and Future Work

The future work would include looking at the same problem from an optimization perspective [2] and using conjugate gradient solvers instead of iterative methods. Also, the use of linear model in the Kalman Filter can be replaced by a general function transformation and use extended Kalman Filter or Particle Kalman Filtering on the same. Also, avoiding the cold-start [5] problem is important (in the current model, we remove users who don't have even one rating and if there are no ratings in any given time interval, the user-factor state in that time interval is carried over to the next time interval). Also, we assume that the covariances across the states are independent which can be removed to give a dynamic system with state dependent covariances [1]. Also, the initial clustering requires the number of clusters to be formed. An extension where the number of clusters is determined automatically would be a promising approach [4] (for example, using Hierarchical Clustering)

References

[1] A. Y. Aravkin and J. V. Burke. Smoothing Dynamic Systems with State-Dependent Covariance Matrices. *ArXiv e-prints*, November 2012.

[2] AleksandrY. Aravkin, JamesV. Burke, and Gianluigi Pillonetto. Optimization viewpoint on kalman smoothing with applications to robust and sparse estimation. In Avishy Y. Carmi, Lyudmila Mihaylova, and Simon J. Godsill, editors, *Compressed Sensing & Sparse Filtering*, Signals and Communication Technology, pages 237–280. Springer Berlin Heidelberg, 2014.

[3] Tristan Fletcher. The kalman filter explained - tutorial, 2010.

[4] Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 297–304, New York, NY, USA, 2005. ACM.

[5] YangSok Kim, Alfred Krzywicki, Wayne Wobcke, Ashesh Mahidadia, Paul Compton, Xiongcai Cai, and Michael Bain. Hybrid techniques to address cold start problems for people to people recommendation in social networks. In Patricia Anthony, Mitsuru Ishizuka, and Dickson Lukose, editors, *PRICAI 2012: Trends in Artificial Intelligence*, volume 7458 of *Lecture Notes in Computer Science*, pages 206–217. Springer Berlin Heidelberg, 2012.

[6] Samuel Nowakowski, Cédric Bernier, and Anne Boyer. A new recommender system based on target tracking: a kalman filter approach. *CoRR*, abs/1012.3280, 2010.

[7] Herschel L. Mitchell P. L. Houtekamer. Ensemble kalman filtering, 2005.

[8] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. A hidden markov model for collaborative filtering. *MIS Q.*, 36(4):1329–1356, December 2012.

[9] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *In ICML 08: Proceedings of the 25th International Conference on Machine Learning*, 2008.

[10] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[11] John Z. Sun, Kush R. Varshney, and Karthik Subbian. Dynamic matrix factorization: A state space approach. *CoRR*, abs/1110.2098, 2011.

[12] Ariel Zetlin-Jones Ted Rosenbaum. The kalman filter and the em algorithm, 2006.

[13] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[14] Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *NIPS*, pages 2595–2603, 2010.

11 Appendix

In the M-Step, the log-likelihood is maximized which has the following form,

$$\begin{aligned}
&\log L = \log(p(x, y; \theta)) \\
&= \sum_{i=1}^{N} \log(p(x_{i,0})) + \sum_{i=1}^{N} \sum_{t=1}^{T} \log(p(x_{i,t}|x_{i,t-1})) + \sum_{i=1}^{N} \sum_{t=1}^{T} \log(p(y_{i,t}|x_{i,t})) \\
&\text{where} \\
&p(x_{i,0}) \sim N(x_{i,0}; \mu_i, \Sigma_i) \\
&p(x_{i,t}|x_{i,t-1}) \sim N(x_{i,t}; A_{i,t}x_{i,t-1}, Q_i) \\
&p(y_{i,t}|x_{i,t}) \sim N(y_{i,t}; H_{i,t}x_{i,t}, R_i)
\end{aligned} \tag{8}$$

Deriving the equations [11] gives the following which are all independent terms and derivative w.r.t to the parameters can be taken independently for each of the L_1, L_2 and L_3 terms.

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

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

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

where

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

$$\Gamma_2 = \sum_{i=1}^{N} \sum_{t=1}^{T} tr[(P_{i,t|T} + \hat{x}_{i,t|T}\hat{x}_{i,t|T}^T) - 2(P_{i,t,t-1|T} + \hat{x}_{i,t|T}\hat{x}_{i,t-1|T}^T)A_t^T + A_t(P_{i,t-1|T} + \hat{x}_{i,t-1|T}\hat{x}_{i,t-1|T}^T)A_t^T]$$

$$\Gamma_3 = \sum_{i=1}^{N} \sum_{t=1}^{T} [fill(y_{i,t}) fill(y_{i,t})^T - 2 fill(y_{i,t})\hat{x}_{i,t|T}^T V^T + (J_{i,t}V)(P_{i,t|T} + \hat{x}_{i,t|T}\hat{x}_{i,t|T}^T)(J_{i,t}V)^T]$$