PCA-Rec: An Optimal Item Recommendation System

Ted An and **Utsav Bhat**
Computer Science 222
Harvard University, Cambridge, MA
{ tedan, ubhat} @fas.harvard.edu

Abstract

The rise of businesses that rely on online retail websites has resulted in a significant demand for accurate item recommender systems. A common approach to this problem is to feed the sequential item purchase history of users into a Markov chain model. Leading work in the field has shown that models that assign each user an individual transition matrix produce good results. Many such models use some form of matrix factorization to both reduce the number of parameters in order to compensate for the sparsity of the purchase history data. We adopt this model and propose a novel method of reducing the number of parameters ”optimally” by using the statistical technique of principle component analysis. (PCA) Our empirical results demonstrate that our model outperforms unpersonalized forms of recommendation like most popular. Our final mode combines the sequential and personalized aspects of the data. We show that the resulting performance outstrips the performance of our model without the sequential aspect of the data or without the personalized aspect the data. We also note that our model is theoretically applicable to datasets beyond the movie recommendations dataset our empirical results are based on.

Key Terms

Recommendation Systems, Markov Chains, Principle Component Analysis, Eigenvalues, Eigenvectors

1 Introduction

Item recommendation systems are a crucial part of many businesses that rely on online retail websites. They have been demonstrated to increase sales for firms and presumably make it more convenient for consumers to identify the prices and availability of items that they wish to purchase at some point in the future. The general goal of a recommender system is to correctly identify which items a particular consumer is likely to want to purchase.

For many of these online retailers, it is a reasonable assumption that purchase data is logged by user and timestamp, thereby endowing the purchase data with a sequential aspect. Given this sequential aspect, it becomes reasonable to imagine that there might be some underlying graph between items that users traverse when making a purchase. This conceptual model is captured by the formulation of a Markov chain transition matrix over the network of items.

A key distinction is between a recommender system that utilizes sequential effects versus a recommender system that works off general data. For example, personalized recommender systems with some look-back (Markov chain length) might be thrown off in their predictions by temporary discrepencies in user behavior. For example, a user might make a purchase of baby items as a gift for a couple’s wedding shower. This purchase might heavily influence the user’s predictions until new, more characteristic purchases are made. While a general recommender system, which intuitively predicts user ”taste”, is more resilient discrepencies in user behavior, it might also be unable to predict certain sequential tendencies users might have. For example, a user might watch the movie ’The Illusionist’ immediately after watching ’The Prestige.’

Another key distinction is between recommender systems that are unpersonalized versus recommender systems that are personalized. An example of an unpersonalized recommender system would be to simply display for each user the items that are overall most popular, i.e. the items that are purchased the most frequently. These methods generally are more efficient in terms of space and easier to implement. Personalized models tend to raise the question of how to compensate for the sparsity of the data for users who have not purchased a large number of items.

Our paper will present a model that uses a Markov chain per individual user to make predictions. This takes the form of $|U|$ $|I| \times |I|$ transition matrices where $|I|$ is the number of items and $|U|$ is the number of users in our system. This ”cube” of parameters is called the transition tensor. While tensor decompositions have been proposed before on personal Markov Chains, our primary contribution is the use of 2-dimensional Principal Component Analysis to identify the similarity structure between users and items, and between sequences of items, in some optimal sense. We can find the principal components of a data matrix indicating which users have purchased which items, and then transform the data matrix into the space of principal components. We set the values corresponding to the lower principal components equal to zero, and then perform the inverse transformation, back into the original space.

Our empirical results demonstrate that our model produces good results on the movie ratings dataset in comparison to simpler methods. We also include a short discussion on possible generalizations of our model to different item purchase history datasets.

2 Related Work

The idea of using personalized Markov chains to model the purchasing behavior of users is not new. [2, 4] suggests a tensor-factorization based approach towards these kinds of problems whereas [3] takes a more Bayesian approach. [1] extends and combines both of these approaches, filling in the tensor-factorization model with a Bayesian learning algorithm from [2].

Our paper borrows several elements from [1] but we factorize our tensors using principle component analysis, a technique that we believe to give us the optimal method for parameter estimation. We present empirical testing of our algorithm on a movie ratings dataset and give some discussion on the natural extension of this algorithm to other other problems in the field of recommender systems.

Movie recommender systems have also been studied extensively, mainly as a result of the Netflix Prize challenge.[1] The best algorithms proposed for that challenge are of a significantly different nature than the item-recommendation algorithms we propose here, for example, the amalgamated results algorithms as in [5, 6]. We would like to stress the fact that while our data consists of movie ratings, our proposed recommender system was designed with general item-recommendation scenarios in mind and does not take into account certain nuances specific to the movie-recommendations problem. We provide a more formal discussion of this later in the paper in section 8 of this paper.

3 Formalization of the Sequential Item Recommendation Problem

Our recommender system will provide for each user a ranking over all of the possible items, where the item with the highest rank is the item that the model declares that the user is mostly likely to purchase next. It is important to note that our results and conclusions are derived only from the sequential aspect of the data. We do not use anything related to the identity of the actual items or users. Furthermore, while the timestamps are available to us we use them only to obtain the relative sequence of ratings for each user.

As mentioned in the introduction, we denote the set of users by $U = \{u_1, u_2, \ldots, u_{|U||}\}$. For each user, we have some sequence $(u_1, u_2, \ldots, u_{t_u})$ where u_j represents the jth item purchased by the user u, and each user purchases t_u items. Each item belongs to the set of items, given by $I = \{i_1, i_2, \ldots, i_{|I|}\}$.

We extend this framework to our movie recommendations dataset with the following: Let M be the set of movies $\{m_1, \ldots, m_{|M|}\}$. For each element $m_i \in M$ we add to I the pairs (m, r) for $r \in \{1, 2, 3, 4, 5\}$, representing ratings from one stars to five stars respectively. In this manner we obtain an item set I containing every (movie, rating) pair possible. Note that $|I| = 5|M|$.

Given formally, for each user we are given the first k movie movie ratings $(i_1, i_2, \ldots, i_{t_u - k})$. We want to produce a personalized ranking over items that assigns high rank to the $k < k$ remaining positive movie ratings for each user, where we declare a positive movie rating to be a rating of 3, 4 or 5.

4 A Personalized Markov Chain Approach

We assume that for each user, there is some underlying "true" markov chain that represents the user's purchase tendencies. This captures both the sequential and the user-dependent aspects of the model: for example, a user who tends to watch one type of movie is hypothesized to have a reasonably high probability of moving to that type of movie no matter what movie he just saw. Our goal is to estimate this transition matrix from the observed data.

4.1 One Approach: Maximum Likelihood

The simplest approach would be to use the maximum likelihood estimator of this transition cube, and for each user, set $P(u_{t+1} = j | u_t = i) \equiv P_{ij} = \frac{\{t | u_t = i, u_{t+1} = j\}}{\{t | u_t = i\}}$. In other words, the observed fraction of the time that the user has made this particular transition. In this case, this approach will not work. First of all, there is not nearly enough data to estimate $|I|^2$ parameters independently for each user, since users are not viewing millions of movies each. Second, this completely wastes the dependence structure between users, which is our most powerful tool in item recommendation.

4.2 Another Approach: Tucker Decomposition With SBPR

As in [1], we can perform a Tucker decomposition the transition cube to reduce the number of free parameters from $O(|I|^2|U|)$ to $O(k_u(|U| + |I|) + k_i|I|)$ parameters, where k_u, k_i are parameters which we can vary, and which are typically very small < 100. Let A be our transition cube. Then, roughly speaking, in order to approximate A, this decomposition is a generalization of the method of performing singular value decomposition to find a low-rank approximation to a matrix. The precise mathematical details can be found in [1].

Now, we make the simplifying assumption that we only care about the relative probabilities of different item purchases in the next time step, and so we will let $\hat{A}$ be the matrix of "ranking scores," where for a given user and last item, the items sorted in descending order by score gives the ranking of our estimated probabilities of moving to that item in the markov chain.

Again as in [1], we can let the (u, l, i) element of $\hat{A}$,

$$\hat{a}_{u,l,i} = < v_u^{U,I}, v_i^{I,U} > + < v_i^{I,L}, v_l^{L,I} >$$

where $v^{U,I}, v^{I,U}, v^{I,L}$, and $v^{L,I}$ are feature matrices of parameters which we have estimated from the data. The model independently computes the similarity of the user and the next

[1]http://www.netflixprize.com/.

item, and the last item with the next item, and combines them independently by taking a standard dot product of the feature matrices.

Figure 1 shows a depiction of these feature matrices. Given a user u and a most recent purchase item l, we can compute the ranking score of an item i by taking the dot product of the uth row of the first matrix above with the ith row of the second matrix on the left, and then adding to that the dot product of the ith row of the first matrix on the right with the lth row of the second matrix on the right. The number of columns of these matrices is a parameter we can tweak to get better results.

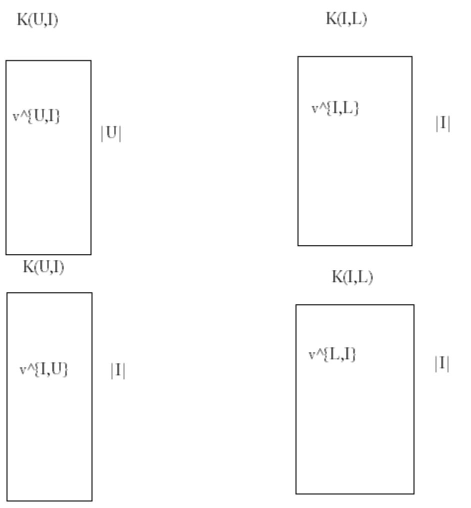

Figure 1: Depiction of the parameters of the model

Now, estimation of these parameters is conducted through the SBPR algorithm covered in [1; 3]. The algorithm models the parameter space as a neural network with certain weight decay parameters, and performs stochastic gradient descent by bootstrapping the parameter space. Essentially, starting with the vectorization of parameters $\theta \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, we randomly choose items one at a time and adjust the corresponding parameters in the direction of the gradient. An additional penalty for large weights is enforced in order to ensure the algorithm converges to a reasonable solution.

5 A PCA-Based Method to Estimate the Feature Matrices

We introduce a new approach to estimate the parameters of these feature matrices based on principal component analysis. We notice that the parameters are just some small number of linear combinations of the users, and the items, where the combinations should grouped in such a way that similar users and similar items should have similar contributions to each of the k columns of the matrix. The neural network learning algorithm SBPR imposes almost no restrictions on the parameter space, and is thus highly dependent on the parameters of the neural network model; it can very easily be underfit or overfit. It turns out there is a precise answer to the question of what the "best" k linear combinations of users and items are, in one sense. We propose Principal Component Analysis (PCA) to impose additional restrictions on the parameter space to achieve a theoretically optimal solution.

5.1 Principal Component Analysis

Formally, suppose we have a data matrix $X_{n\times p}$ which has mean vector $\bar{x}_{1\times p} = \mathbf{0}$. Then, we define the *first principal component* $g = z_1$ to be the vector that maximizes $\|Xg\|$ subject to $\|g\| = 1$. If we have already defined $n-1$ principal components, we define the *nth principal component* $g = z_n$ to be the vector that maximizes $\|Xg\|$ subject to $\|g\| = 1$, and g is orthogonal to $z_1, \ldots, z_{n-1}$.

Intuitively, z_1 is the linear combination of the columns of X which maximizes the sum of squares of each row of the data matrix, or in other words, the expected variance with each row treated as a distinct sample.

Indeed, the computation of the principal components is very simple. Taking the set of eigenvectors of $X'X$ sorted in decreasing order by the magnitude of their corresponding eigenvalues gives the n principal components in order.

Given a non-centered data matrix X, we can center each of the columns to fit within this framework. Further, we can perform the same analysis with X': this first principal component for this matrix will give the linear combination of the rows of X with maximal expected variance, this time with the columns being treated as distinct samples.

Why is this desirable? Applying PCA to X gives us a 1-dimensional summary for each of the n data points which separates them out as much as possible. For example, if our data matrix is a standardized indicator matrix of user-item purchases, then the first principal component of the users will give us the most information possible to distinguish the items.

5.2 Applying PCA to the feature matrices

Returning to our problem of estimating our feature matrices, we begin with the case of $v^{U,I}$ and $v^{I,U}$, the feature matrices corresponding to user-item similarity. We let our data matrix $X_{|I|\times|U|}$ be the $0-1$ matrix corresponding to user-item purchases or views. In our model, we subtract the column means of this matrix, and compute all the principal components of the users. We do the same with X, and store all the principal components of the items. Note that this eigenvalue and eigenvector computation is made very efficient by the Lanczos algorithm, which is designed for large, sparse matrices.

Let Γ_1 be the matrix of principal components for the users, and let Γ_2 be the matrix of principal components for the items (i.e. every column is a principal component vector). We compute $Z_{|I|\times|U|}$, the "score" matrix, whose i, j element stores the ranking score of the ith item principal component with the jth user principal component. This score of a principal component is the sum of the individual scores of each user-item pair, weighted by the product of their coefficients in the principal components. More precisely, we define

$$Z(i,j) = \sum_{s=1}^{|I|}\sum_{r=1}^{|U|} X(s,r)\cdot\Gamma_2(s,i)\cdot\Gamma_1(r,j)$$

where we have treated these arrays as though they are 1-indexed. This is our representation of X in the principal component space. Now, we only want to use k_u dimensions of the users and k_i for the items, so we set the last $|I| - k_i$ columns and the last $|U| - k_u$ rows equal to zero, and perform the inverse transformation to get back to item-user space.

We do this by computing

$$\hat{X}(i,j) = \sum_{s=1}^{k_u} \sum_{t=1}^{k_t} Z(t,s) \cdot \Gamma_2(i,t) \cdot \Gamma_1(j,s)$$

For the item-item dependence, let the i, j element of a new data matrix be the number of times across all users that a user bought i then j immediately after. Then, standardize the rows so that they sum to 1, and call this X. We can de-mean the columns of X and apply PCA to this matrix, yielding the principal components of the last items, and then we can de-mean the columns of X', whose principal components represent the next items. We can proceed in the same fashion as above: transform into the space of principal components, discard the last $n - k$ scores, and then transform back into the original space to get a matrix of scores.

Now, we have left out one issue: the data matrices we started out with for the user-item interactions and for the item-item interactions were not on the same scale. All of the entries of this second matrix are less than 1, and if many users tend to buy the same sorts of items, some of these entries may be very small. PCA is sensitive to scale change, so letting X_1 be the user-item data matrix and letting X_2 be the item-item data matrix, we adjust for this by computing

$$w = \frac{\sigma_{vec(X_1)}}{\sigma_{vec(X_2)}}$$

where $\sigma_{vec(X_i)}$ is the standard deviation of the vectorization of X_i. Intuitively, since PCA maximizes the sums of squares, or expected variance, we compute our final ranking score to be

$$\hat{a}_{u,i,l} = \hat{X}_1(i,u) + w^2 \hat{X}_2(l,i)$$

6 Example of Application of PCA

We give an example of the application of PCA to a small data matrix. Suppose the purchase data we have observed is represented by the following matrix X. The columns correspond to users where the users correspond to the columns, and the purchased items correspond to the rows. An entry in the matrix is filled 1 iff the particular user has bought the corresponding particular item.

$$X = \begin{pmatrix} & & \text{Users} & \\ \text{Items} & x & y & z \\ a & 1 & 0 & 1 \\ b & 1 & 1 & 1 \\ c & 1 & 0 & 0 \\ d & 0 & 1 & 0 \end{pmatrix}$$

Now we apply PCA to the standardized versions of both X and X to find the principal components for the users and items respectively. We find that the principal component matrix of the users is:

$$\Gamma_1 = \begin{pmatrix} z_1 & z_2 & z_3 \\ -0.6426 & 0 & 0.7662 \\ 0.5418 & 0.7071 & 0.4544 \\ -0.5418 & 0.7071 & -0.4544 \end{pmatrix}$$

And the principal component matrix of the items is:

$$\Gamma_2 = \begin{pmatrix} z_1 & z_2 & z_3 & z_4 \\ -0.6280 & 0.3251 & -0.7052 & -0.0520 \\ -0.0000 & 0.0000 & 0.0736 & -0.9973 \\ -0.4597 & -0.8881 & -0.0000 & 0.0000 \\ 0.6280 & -0.3251 & -0.7052 & -0.0520 \end{pmatrix}$$

In more familiar terms, Γ_1 is the matrix of eigenvectors of $X'X$, sorted in descending order by the magnitude of the corresponding eigenvalues. Similarly, Γ_2 is the matrix of eigenvectors for XX', also sorted in descending order by the magnitude of the corresponding eigenvalues. For a k-dimensional analysis, we delete everything but the first k columns of these matrices.

We see that the first principal component for the users places positive weight on user 2, and negative weights on users 1 and 3. The first principal component for the items places a positive weight on item 4, and a negative weight on items 1 and 3. Now consider what the first principal component represents, intuitively: it is the particular linear combination that maximizes the variance. Since user 2 and item 4 appear to be dissimilar to the other users and items we see that it is logical that they are given weights with different signs by PCA.

In this particular example, our matrix of scores is given by:

$$Z = \begin{pmatrix} 1.2071 & 0.3792 & -0.0572 \\ -0.3233 & 0.7326 & -0.3299 \\ -0.0952 & 0.2096 & 0.1135 \\ 0.2615 & -0.5754 & -0.3117 \end{pmatrix}$$

As we can see from our earlier discussion on the 1st users vector and the 1st items vector, it is logical that the score in the $(1,1)$ or (a,x) position is very high. A high score corresponds to the similarity between the particular linear combination of users and items; since there is a 1 in the $(4,2)$ or (d,y) entry of X, and since user y and item d appear to be fairly dissimilar to the other users and items, this observation is logical.

Now, we reduce the dimensionality of this matrix, and compute the prediction for individual user-item combinations. Setting $k_u = k_t = 2$, here is our matrix of ranking scores:

$$R = \begin{pmatrix} & & \text{Users} & \\ \text{Items} & x & y & z \\ a & 0.5546 & -0.4676 & 0.4676 \\ b & 0.0 & 0.0 & 0.0 \\ c & 0.1721 & -0.7284 & -0.4384 \\ d & -0.5546 & 0.4676 & -0.4676 \end{pmatrix}$$

Interestingly, the second row is all zeros. This is because the principal component corresponding to this row was the last one, (the 4th one) and we deleted that data by using a

2-dimensional analysis. This data was represented by the 4th principle component because every user bought this item, so the item had zero contribution to the variance, and does not help distinguish the users at all.

We see that the entries that correspond to the 1-entries in our original matrix Y generally have high values: this is reasonable because those items were actually purchased. We are more concerned about the scores that correspond to the 0-entries Y because these scores can be used to come up with a ranking for which zero-entries in our original matrix are most likely to become ones. This model predicts that the most likely new purchase is user z buying item c, which seems reasonable, since user z is similar to user a, who has bought item c.

An added note: given all of our principle components, that is, setting $k_u = 3, k_t = 4$, we recover our original data matrix X, as expected.

7 Evaluation

We demonstrate the effectiveness of our PCA algorithm on a dataset of movie ratings, and give some comparisons between our algorithm and some of the other approaches we have mentioned.

7.1 Dataset

We have a set of 100,000 movie recommendations. The set is fairly sparse, but every user has rated at least 20 movies. There are $|U| = 943$ users and $|M| = 1682$ movies. Under the model we described earlier, this gives us a set of items that has size $|I| = 5|M| = 8410$. We split the entire dataset into two pieces: for each user we remove the final 5 movie ratings. The remaining data (95285 movie recommendations) is our D_{train}, the data that we will use to train our algorithm.

The final 5 movie ratings for each user is our D_{test}, the data that we will evaluate the movie-rankings produced by our algorithm on.

7.2 Adaption to Movie Ratings

As we mentioned previously, our model is designed for item recommendations, which are inherently different from movie ratings, for which we were able to procure data. While we deemed every (movie,rating) pair an item in the training set, this may not be the best evaluation metric for the test set, since the interpretation of successfully predicting a (movie, low rating) pair is unclear. Therefore, we modify the prediction score to give just one ranking score for every item, which we compute as the sum of (item,3) + (item,4) + (item,5). Then, we throw out the items in the test set which received either a 1 or a 2 rating, and evaluate our model as a predictor of individual movies. One thing to note is that this is not the same as discarding all data for which the rating was a 1 or a 2: these observations have a an impact on the overall dependence of items. Also of note: since we have the underlying assumption that users do not forget about movies and/or items they have bought, we remove from our ranking of recommendations the items that they have already rated, that is, the elements belonging to that user in D_{train}.

7.3 Comparison Models

We compared our model with a model which ranks the most popular movies seen in the training set for every user, and we compared our full model with the reduced-versions: the Unpersonalized Markov Chain model (UMC), and the Non-sequential User Model (NUM). The UMC regards only the item-item sequential scores, and the NUM uses only the user-item ranking scores, while the full model takes their sum.

7.4 Attempted Comparison with FPMC-BPR [1]

As mentioned earlier, the authors of [1] use a neural network model with highly influential parameters, including α, the learning rate, $\boldsymbol{\lambda_\theta}$, the weight decay parameters, and σ^2, the initial starting point of the learning algorithm. Unfortunately, the authors do not include their values for these parameters, or provide any indication whether α changes over time.

In our experiments we have tried repeatedly to find the values of λ and α that give (1) convergence and (2) reasonable predictions for our dataset, but it appears that $\boldsymbol{\lambda_\theta}$ and α must be calibrated to a fine degree to attain both conditions. We hope that at some point Rendle *et al.* will take note of our queries and respond. The best result we were able to obtain was a mean of 527, with calibration parameters $\sigma^2 = 1, \lambda_\theta = .05$ for all θ, and $\alpha = .15$, as they are defined in [1]. This result took 200 million trials, and was the only instance of such a large number of trials where the parameters didnt all converge to either zero or infinity.

In terms of a more abstract, theoretical concern, note that we do not use a learning algorithm and are able to compute our results directly, and therefore have no need for re-calibration on different datasets.

7.5 Metrics of Evaluation and Results

We evaluated the performance of our model in three different ways

- Visual inspection of histograms of the predicted rank of items purchased
- Mean ranking of items purchased. This can also be thought of as $|I| -$ Area under ROC curve .
- Visual inspection of graph of x vs. fraction of the time that the item purchased was ranked at least that high

Figure 2 shows the graph of the mean ranking scores of our model and the baseline most popular with different numbers of dimensions. The blue circle at the bottom shows the best performance we were able to achieve: 16 user-item factorization dimensions, and 32 item-item factorization dimensions. For more than one dimension, the full model substantially outperforms the baseline. Interestingly, NUM seems to do worse with 32 dimensions then with just 16 dimensions of analysis. This is perhaps a result of both overfitting and not capturing enough of the inherent dependence.

Figure 3 shows histograms for 3 of our models and the baseline representing the distribution of predicted rankings of purchased items. We can see from Figure 3 that our full algorithm consistently does better than the most-popular baseline for 8, 16 and 32 dimensional analysis.

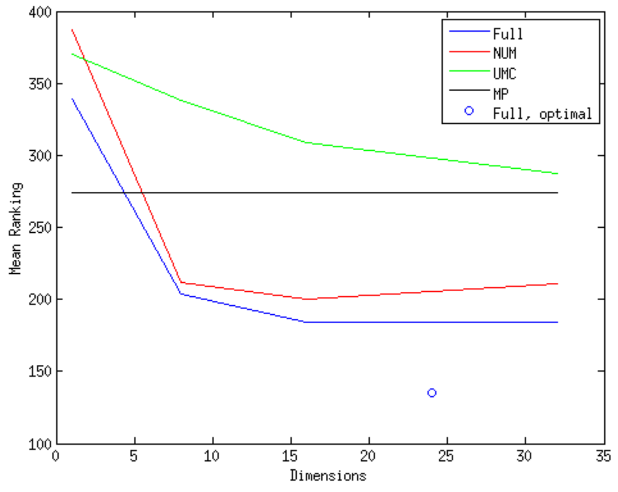

Figure 2: Mean ranking from various models

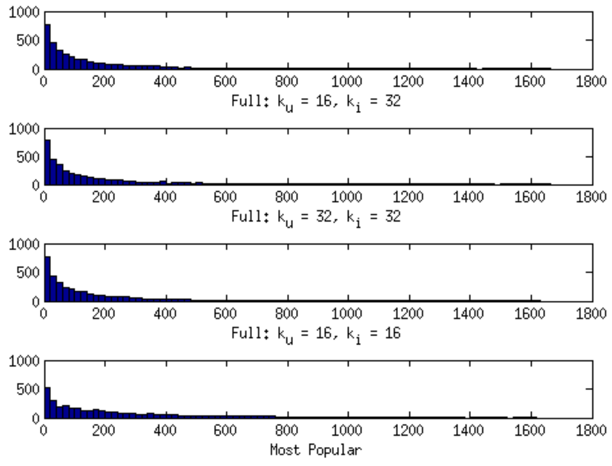

Figure 3: Histograms of ranking scores for various models

Figure 4 is a very interesting graph. Every value on the x-axis gets mapped to the number of items which were ranked at least that high. This shows that our models improvement over the baseline is especially high when we want a top k recommendation for a user.

Our full model subsumes both NUM and UMC, and accordingly, we see that the full model does significantly better than either NUM and UMC. This is a positive result, as it shows that both the sequential and the personalized aspects of the data we are capturing make valuable contributions to the overall quality of the prediction.

8 Conclusion

We have proposed and tested a recommender system that uses principle component analysis to reduce the number of parameters in a theoretically optimal way. We have incorporated into our model elements that have been found to be effective in other papers in the field: for example, we use personalized Markov transition matrices for each user, an feature first proposed by [1]. Our proposed method of PCA theoretically provides an optimal way of reducing the number of parameters to a manageable amount. We hope to obtain more robust results in the near future.

8.1 Further Work

Our algorithm is widely applicable, and we have demonstrated that it produces good results on even a dataset of movie ratings. We take this as a good sign, that our algorithm can definitely do better on a dataset that has more underlying structure. We fully intend to procure a dataset similar to the one used by [1], that is, some kind of drugstore-item purchasing data. Other possible candidates are book-purchase datasets or music-purchase datasets.

In terms of tractibility we believe that our algorithm should allow for analysis up to 128 dimensions. However, we have not been able to procure enough uninterrupted computational time to finish calculating the analysis with 64 and 128 dimensions. In practice, a recommender system would have to be able to handle online updates to be implemented. This model fits this description perfectly, as the Lanczos algorithm for computing eigenvalues and eigenvectors is an iterative algorithm that approaches the true solution. Adding a new element likely will not change the eigenvectors significantly, and convergence could be very rapid.

References

[1] Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th international Conference on World Wide Web* (Raleigh, North Carolina, USA, April 26 - 30, 2010). WWW '10. ACM, New York, NY, 811-820.

[2] Rendle, S., Balby Marinho, L., Nanopoulos, A., and Schmidt-Thieme, L. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Paris, France, June 28 - July 01, 2009). KDD '09. ACM, New York, NY, 727-736.

[3] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial intelligence* (Montreal, Quebec, Canada, June 18 - 21, 2009). Uncertainty in Artificial Intelligence. AUAI Press, Arlington, Virginia, 452-461.

[4] Rendle, S. and Schmidt-Thieme, L. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the Third ACM international Conference on Web Search and Data Mining*

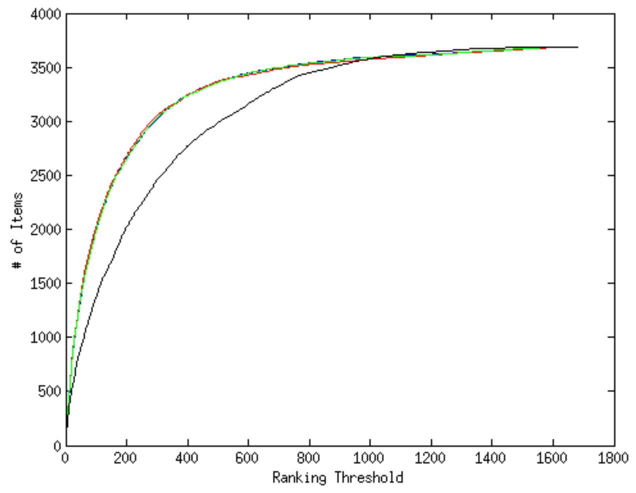

Figure 4: Threshold versus number of items ranked at least that high

(New York, New York, USA, February 04 - 06, 2010). WSDM ’10. ACM, New York, NY, 81-90.

[5] R. Bell, Y. Koren, and C. Volinsky. Chasing 1,000,000: How we won the Netflix progress prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2):4-12, December 2007.

[6] R. M. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Neflix Prize. Technical report, AT&T Labs Research, 2007. http://www.netflixprize.com/.