Methods for large scale SVD with missing values*

Miklós Kurucz András A. Benczúr Károly Csalogány
Data Mining and Web search Research Group, Informatics Laboratory
Computer and Automation Research Institute of the Hungarian Academy of Sciences
{realace, benczur, cskaresz}@ilab.sztaki.hu

ABSTRACT

We compare recommenders based solely on low rank approximations of the rating matrix. The key difficulty lies in the sparseness of the known ratings within the matrix that cause expactation maximization algorithms converge very slow. Among the prior publicly known attempts for this problem a gradient boosting approach proved most successful in spite of the fact that the resulting vectors are non-orthogonal and prone to numeric errors. We systematically explore expectation maximization methods based both on the Lanczos algorithm and power iteration; novel in this paper is the efficient handling of the dense estimate matrix used as input to a next iteration. We also compare sequence transformation methods to speed up convergence.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences; G.1.3 [**Mathematics of Computing**]: Numerical Analysis—*Numerical Linear Algebra*

General Terms

recommender systems, singular value decomposition

Keywords

dimensionality reduction, missing data

1. INTRODUCTION

Recommender systems predict the preference of a user on a given item based on known ratings. In order to evaluate methods, in October 2006 Netflix provided movie ratings from anonymous customers on nearly 18 thousand movie titles [3].

*This work was supported by the Mobile Innovation Center, Hungary, a Yahoo Faculty Research Grant and by grant *ASTOR* NKFP 2/004/05

KDDCup.07, August 12, 2007 , San Jose, California , USA. 2007

In this paper we concentrate on recommenders based solely on low rank approximation and compare various implementations and parameter settings. The low rank approximation of the rating matrix as a recommendation is probably first described in [5, 24, 17, 27] and many others near year 2000.

The key difficulty in computing the low rank approximation lies in the abundance of missing values in the rating matrix: the Netflix matrix for example consists in 99% of missing values. While several authors describe expectation maximization based SVD algorithms dating back to the seventies [13] and [7, 28, 30] describes the method for a recommender application, we aware of no systematic studies on large scale problems. In particular all these results consider small, few thousands by few thousands submatrices of the EachMovie or Jester databases, of several orders of magnitude smaller than handled by our algorithms.

A successful approach to a low rank recommender is described by Simon Funk in [12] is based on an approach reminiscent of gradient boosting [11]. The algorithm opens a number of theoretic questions including its relation to published results that solve SVD with missing values as well as the effect of the parameters on convergence speed and overfitting. One of the main intents of this paper to understand the relation of his method to existing missing value SVD approaches.

Our main contributions are:

- The implementation of SVD based recommenders for large scale problems with specific attention to the scalability issues of handling full matrix imputation values. Note that previous results except for [12] handle data of several orders of magnitude smaller than ours.
- The comparison of various methods in terms of recommendation accuracy and convergence rate, with emphasis on the explanation of parameters that speed up convergence.

1.1 Our results and organization

The rest of the paper is organized as follows. In the rest of the Introduction we describe related approaches, the experimental setup, and the SVD algorithm implementation used. In Section 2 we measure the effect of filling missing values by zeroes, by averages and finally be the output of an item-item similarity based recommender, a challenging task since the full recommendation matrix has several billion entries.

After imputation by external values we turn to expectation maximization approaches that compute a low rank approximation in one iteration and impute the outcome for

a next iteration. In Sections 3 and 4 we use the Lanczos algorithm and power iteration, respectively. In both cases we resolve the implementational challenge of handling the full matrix arising by the previous iteration. We observe slow convergence of the methods; we evaluate methods to speed up by combining partial results.

Finally in Section 5 we describe a least squares based expectation maximization approach to directly optimize a low rank solution for small error. In this algorithm we optimize each user vector separately, thus enabling a user-by-user adaptive control on the number of dimensions used. Our key observation is that for a user with r ratings, roughly $r/25$ dimensions should be used.

In all cases we investigate the effect of the dimensionality of the low rank approximation; we observe best performance at a few dimensions and overtraining as the number of dimensions approaches 100 with good performance on the training but deterioration on the test (probe) set. All methods are compared in Section 6.

1.2 Data set, evaluation and experimental setup

Netflix provided over 100 million ratings from n over 480 thousand randomly-chosen, anonymous customers on m nearly 18 thousand movie titles [3]. The company withheld certain portion of the ratings as a competition qualifying set that we will not use in this report. Netflix also identified a *probe* subset of the complete training set; we refer the remaining known ratings as the *training* data.

We use the *root mean squared error*

$$\text{RMSE}^2 = \sum_{ij \in R} (w_{ij} - \hat{w}_{ij})^2$$

as the single evaluation measure, where w_{ij} is the actual rating, an integer in the range 1–5, given by user i to movie j, and $\hat{w}_{ij}$ is the prediction given by the recommender system. We present RMSE values for the train and the separated test set but not the qualifying set.

The experiments were carried out on a cluster of 64-bit 3GHz P-D processors with 4GB RAM each and a multiprocessor 1.8GHz Opteron system with 20GB RAM.

1.3 Related work

Recommenders based on the rank k approximation of the rating matrix based on the first k singular vectors are probably first described in [5, 24, 17, 27] and many others near year 2000.

The Singular Value Decomposition (SVD) of a rank ρ matrix W is given by $W = U^T \Sigma V$ with U an $m \times \rho$, Σ a $\rho \times \rho$ and V an $n \times \rho$ matrix such that U and V are orthogonal. By the Eckart-Young theorem [16] the best rank-k approximation of W with respect to the Frobenius norm is

$$||W - U_k^T \Sigma_k V_k||_F^2 = \sum_{ij} (w_{ij} - \sum_k \sigma_k u_{ki} v_{kj})^2. \quad (1)$$

where U_k is an $m \times k$ and V_k is an $n \times k$ matrix containing the first k columns of U and V and the diagonal Σ_k containing first k entries of Σ.

The RMSE differs from the above equation only in that summation is over known ratings

$$\text{RMSE}^2 = \sum_{ij \in R} \text{err}_{ij}^2 \text{ where } \text{err}_{ij} = w_{ij} - \sum_k \sigma_k u_{ki} v_{kj} \quad (2)$$

where R denotes either the training or the test set. To simplify notation we extend err_{ij} with value 0 for $ij \notin R$.

As already emphasized in one of the early works [27], the crux in using SVD for recommenders lies in handling missing values in the rating matrix W. Goldberg et al. [15] for example require a gauge set where all ratings are known, an assumption clearly infeasible on the Netflix data scale. Azar et al. [2] prove asymptotic results on replacing missing values by zeroes and scaling known ratings inversely proportional to the probability of being observed.

The *expectation maximization* (EM) algorithm proceeds as follows. Given the output U_k, Σ_k and V_k matrices of sizes $m \times k$, $k \times k$ and $n \times k$, respectively, produced by SVD in the maximization step, the expectation step produces a matrix with entries

$$\begin{aligned} \hat{w}_{ij} &= \begin{cases} w_{ij} & \text{if } ij \in R \\ [U_k \Sigma_k V_k]_{ij} & \text{otherwise} \end{cases} \\ &= \text{err}_{ij} + [U_k \Sigma_k V_k]_{ij} \end{aligned} \quad (3)$$

where the last equality follows by the definition of err as in (2). The algorithm alternates between SVD computation (maximization) and the expectation equation (3) until convergence. It is easy to see that U, Σ and V minimizing (2) is a fixed point of this iteration; up to our best knowledge, this is the only theoretical result known about the convergence properties of the above EM missing value SVD algorithm. While in our implementation k is typically fixed as input parameter, variants of this algorithm may increase or even decrease k as the iterations proceed.

This algorithm is perhaps first used for recommenders by Canny [7] and then several others [28, 30]. Canny [7] concentrates on privacy issues; he reports experiments on much smaller scale such as a subset of the EachMovie data. Srebro and Jaakkola [28] compare methods that fill missing entries by zeroes, also by scaling known entries as in [2], using a gauge set as in [15] as well as a variant of the EM procedure. They also give a number of hints related to the convergence of the EM method. First of all they observe the algorithm may reach local optimum; it is unclear whether this may happen in the missing value case as well. They also show that different rank solutions are non-orthogonal; for this end they propose starting out with a large rank approximation and gradually reduce the rank in the EM iterations.

The EM algorithm for solving SVD with missing values dates back to the seventies; [14] gives a more recent description. In early results, the generic idea of filling missing values by expectation maximization to our knowledge appears first in [19] and is perhaps best described by [9] with the explicit mention of factor analysis as an application but apparently no references between another line of work [25, 13]. To our knowledge, the first paper that presents the missing value problem is [25]; [13] generalizes the missing value problem to a weighted regression and solves it by EM.

More recently several authors reinvented EM for SVD. In [6] the idea of representing the missing data imputation matrices by their known SVD U and V appears that is key in our sparse implementation. Zhang et al. [30] give an approximate SVD algorithm with theoretical analysis, however tests are only shown on small scale data.

Theoretical works on SVD based recommenders exist [10] but we are aware of none that address the missing value problem. In particular we aware of no results on the conver-

Algorithm	$k = 10$	$k = 15$	$k = 20$	$k = 25$	$k = 30$	$k = 50$
Lanczos	1:58				7:40	
Power	1:57				5:50	
Adaptive	33.7	33	31	29	28	25

Table 1: Running times in the form of minutes or hours:minutes for a single iteration over the Netflix Prize matrix with n over 480 thousand, m nearly 18 thousand and over 100 million non-zeroes. For Lanczos and Power the top column k gives dimensionality while for Adaptive the dimensionality is $1 + r/k$ for a user with r ratings, i.e. here unlike for the other two algorithms the running time decreases with k. For Lanczos we use 40 iterations altogether and for Power we use 100 for a single dimension, hence here we get linear dependency on k for Power.

gence except for the negative experimental findings of Srebro and Jaakkola [28].

Dimensionality reduction is investigated for gene expression data as well. Several authors [20, 29] compare imputation methods including nearest neighbor as well as the EM approach with controversial findings for accuracy but a definite identification of the very slow convergence for EM.

1.4 SVD implementation

In our implementation we used the Lanczos code of `svdpack` [4] and compared it with a power iteration developed from scratch. Lanczos appears precise and efficient; in contrast power iteration used by several results [21, 8] is slightly faster but much less accurate; computing more than two dimensions is numerically very unstable due to the orthogonal projection step. Running times are shown in Table 1.

We implemented a key modification over `svdpack` that enables missing data imputation as well as very large input handling. After removing the obsolete condition on the maximum size of an input matrix, we abstracted data access within the implementation to computing the product of a vector with either the input matrix or its transpose.

While implementation issues of SVD computation are beyond the scope of the paper, we compare the performance of the Lanczos and block Lanczos code of `svdpack` [4] and our implementation of a power iteration algorithm. Hagen et al. [18] suggest fast Lanczos-type methods as robust basis for computing heuristic ratio cuts; others [21, 8] use power iteration.

We also measure the number of dimensions of the approximation. Typically in SVD use the dimensionality is restricted by efficiency considerations and for example for spectral clustering [1, 23] suggest more eigenvalues produce better quality cuts. However we observe that as the number of dimensions increase beyond roughly 10, we overtrain and prediction quality deteriorates; for this reason we also test an algorithm that adaptively selects more dimensions for users with more ratings in Section 5.

2. MISSING DATA IMPUTATION FROM EXTERNAL RESULTS

In the simplest approach we use external sources of data to fill missing ratings and optimize for error in Frobenius norm as in equation (1) in the hope that external data fit well and optimization for Frobenius yields good approximation for

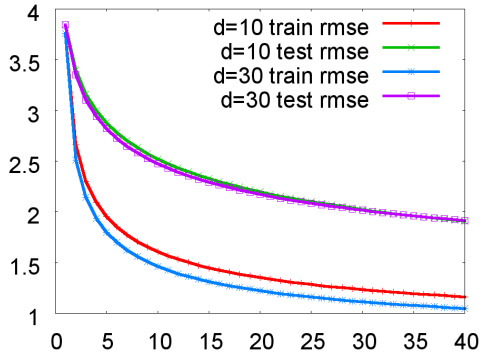

Figure 1: The rmse as the function of the iterations for the simple Lanczos EM algorithm, the missing values are filled with zeros in the initial matrix.

the RMSE equation (2) as well. First, as expected, we show filling missing data with zeroes as suggested for example by [2] badly fail over the rare Netflix data by providing recommendations near 0 due to the abundance of zeroes in the matrix after imputation. We improve performance first by using averages, then by the outcome of a more sophisticated recommender based on item-item similarities. Surprisingly user averages perform better than the output of the recommender in this case.

Imputation by zeroes and averages are fairly straightforward given control over data access within the SVD algorithm as described in Section 1.4. It is however a challenging question for a full recommendation matrix that we describe in Section 2.1.

We show RMSE values for imputation with zeroes and averages as the first iterations in Fig. 3. We observe very poor performance; in particular by filling with zeroes we are so far off from optimum that even a large number of EM iterations remain insufficient to converge.

2.1 Output of an item-item similarity based recommender

We implemented the adjusted cosine similarity [26] for an item-item similarity based recommender that recommends an unrated movie j to a given user i by the weighted average of the nearest N movies to i rated by the user. Here N is a parameter; roughly speaking, this approach increases the fraction of known values by a factor of N.

The Lanczos implementation of svdpack [4] accesses the matrix by in one step computing a product of a vector with either the matrix or its transpose. The SVD implementation may hence access the nearest neighbor lookup table whenever a matrix multiplication is needed. The implementation requires space to store the rating matrix and the nearest neighbor index. In the running time however the matrix multiplication time becomes dependent on the size of the full matrix mn instead of the much smaller number of known ratings. While this implementation is memory efficient, it is so slow that we had to give up tests in this direction.

We may however give an efficient item-item similarity based imputation by slightly regressing the item-item similarity based output towards the user average $\hat{u}_i$, as follows. We form the submatrix S of the item-item similarities where for efficiency considerations we only keep the top 100 largest

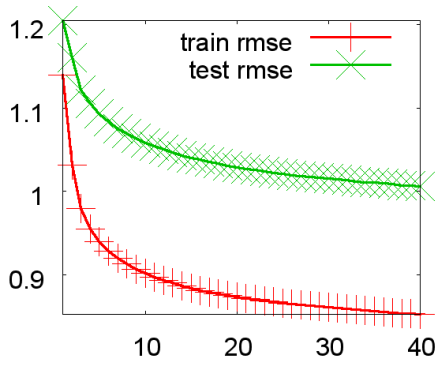

Figure 2: The rmse as the function of the number of iterations for the simple Lanczos EM algorithm with the first iteration imputed with the output of the simplified item-item similarity based recommender.

entries in each row. We even discard those of the values below 0.5. When predicting a rating for user i and movie j, we then compute the sum of $w_{ij'} - \hat{u}_i$ weighted by the similarity of j and j' for all j' where both the similarity and the rating $w_{ij'}$ are known. Next in order to give a prediction we have to add the normalized value to $\hat{u}_i$. In order to be efficiently computable, we simply normalize by 100, even though typically there are less than 100 j' terms in the sum and their similarity values may be as low as 0.5.

The algorithm proceeds as follows. We let H be an $n \times m$ matrix where each row contains the user averages $\hat{u}_i$ as identical values and

$$F_{ij} = \begin{cases} (w_{ij} - \hat{u}_i)/100 & \text{if } (ij) \in R \\ 0 & \text{otherwise.} \end{cases}$$

The product of vector x with the imputed matrix W' can be efficiently computed as

$$x \cdot W' = x \cdot H + x \cdot F \cdot S + x \cdot E$$

where

$$E_{ij} = \begin{cases} W' - H - F \cdot S & \text{if } (ij) \in R \\ 0 & \text{otherwise} \end{cases}$$

removes the effect of the similarity based recommendation where the actual rating is known. In Fig. 2 we see the RMSE for a 10-dimensional SVD started by these values.

3. SPARSE LANCZOS IMPLEMENTATION WITHIN AN EM FRAMEWORK

When using the Lanczos algorithm after an expectation step, we face the same difficulty of imputing a full matrix as in Section 2.1. We provide a similar solution below, with careful analysis of the number of operations used. Note that unlike in the previous section, we need a large number of iterations until convergence hence not just the space but also the speed of handling the dense input is crucial.

Since in a Lanczos iteration we require the product of vector $\mathbf{x} = (x_1, x_2, \ldots)$ with $\hat{w}$ (or similarly with its transpose), by the EM algorithm equation (3) we compute

$$\text{err} \cdot \mathbf{x} + U_k \Sigma_k V_k \cdot \mathbf{x}.$$

The space required by this algorithm is equal to $O(kn+km)$ for multiplying $\mathbf{x}$ with the imputed low rank approximation term by term from right to left, in addition to the number of non-missing values in the rating matrix. Hence the additional work due to imputation is negligible in the Lanczos computation.

3.1 Speeding up convergence

In order to speed up convergence we apply the generic method of finding an optimal linear combination of the values $\hat{w} = \sum_k \sigma_k u_{ki} v_{kj}$ in the current and $w^{(t-1)}$ in the previous iterations: we minimize the quadratic expression of λ in the RMSE equation (2):

$$\sum_{ij \in R} (w_{ij} - \lambda \hat{w}_{ij} - (1-\lambda) w_{ij}^{(t-1)})^2.$$

We then let $w^{(t)} = \lambda \hat{w} - (1-\lambda) w^{(t-1)}$ for the λ value at the minimum.

In the above naive form however matrix values in the next iteration will arise as a linear combination of a rank k matrix and the previous $w^{(t-1)}$, which in turn depends on $w^{(t-2)}$ and inductively on all previous partial results. Since it is infeasible to store either full matrices or all partial results, we have to relax the above algorithm. We give two versions next that obey the scalability requirements.

Our two implementations of linearly combining current and previous results use the last two low rank approximations $U_k^{(t)}$, $\Sigma_k^{(t)}$, $V_k^{(t)}$ and $U_k^{(t-1)}$, $\Sigma_k^{(t-1)}$, $V_k^{(t-1)}$. In the first algorithm we combine as

$$\lambda U_k^{(t)} \Sigma_k^{(t)} V_k^{(t)} + (1-\lambda) U_k^{(t-1)} \Sigma_k^{(t-1)} V_k^{(t-1)},$$

i.e. using only the low rank matrix instead of the combined iteration $t-1$ approximation. The minimum of the quadratic expression in λ is attained, with the notation of $A = U_k^{(t)} \Sigma_k^{(t)} V_k^{(t)}$ and $A' = U_k^{(t-1)} \Sigma_k^{(t-1)} V_k^{(t-1)}$, at

$$\lambda = -\frac{\sum_{ij \in R} (A_{ij} - A'_{ij})(A'_{ij} - w_{ij})}{\sum_{ij \in R} (A_{ij} - A'_{ij})^2}.$$

In the second variant we combine the low rank decomposition elements: from $U_k^{(t)}$, $V_k^{(t)}$ and the previous result we get $\hat{U}_k^{(t)}$ and $\hat{V}_k^{(t)}$. We ignore $\Sigma_k^{(t-1)}$ based on the observation that the Σ_k converges fast. With the simplified notation U_i and U'_i for the i-th row of $U_k^{(t)}$ and $U_k^{(t-1)}$, respectively and the same notation for the columns of the V, we minimize

$$\sum_{ij \in R} \left((\lambda (U_i - U'_i) + U'_i) \Sigma (\lambda' (V_i - V'_i) + V'_i) - w_{ij} \right)^2.$$

Here for each λ' there is a corresponding optimum λ' given by

$$X_{ij} = \lambda' (U_i - U'_i)(V_j - V'_j) + (U_i - U'_i) V'_j,$$

$$\lambda = -\frac{\sum_{ij \in R} \lambda' U_i \Sigma (V_j - V'_j) + U'_i V'_j - w_{ij}) X_{ij}}{\sum_{ij \in R} X_{ij}^2};$$

we select the best by substituting a low number of probe λ values.

4. POWER ITERATION WITHIN AN EM FRAMEWORK

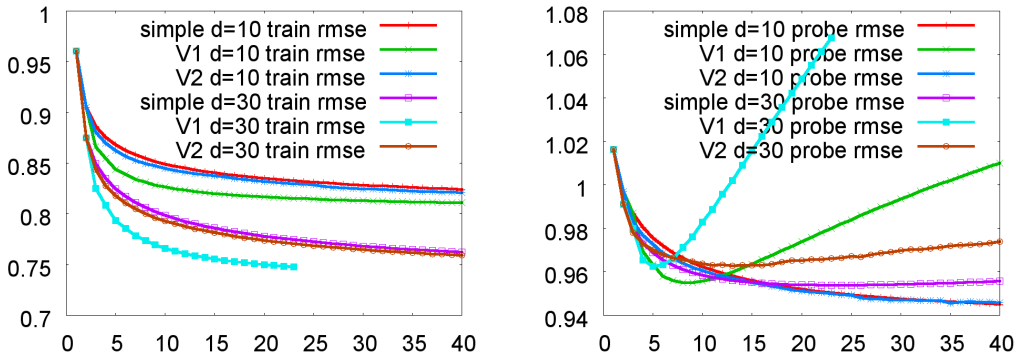

Figure 3: The rmse as the function of the iterations for the simple Lanczos EM algorithm and the two convergence boosting variants. Curves correspond to different dimensionality with V1 and V2 denoting the two convergence boosting variants.

For a full matrix W, power iteration proceeds by repeatedly letting

$$u^{(t+1)} = W^T \cdot v^{(t)}/||v^{(t)}||, \quad (4)$$
$$v^{(t+1)} = W \cdot u^{(t)}/||u^{(t)}||. \quad (5)$$

The algorithm converges to the first singular vectors also called the "hub" and "authority" vectors [22]; due to numeric errors this holds even if we start out with an initial $v^{(0)}$ orthogonal to the first vector $V_{\cdot 1}$ unless we *orthogonalize*, i.e. project each or some $v^{(t)}$ onto the hyperplane orthogonal to $V_{\cdot 1}$. By orthogonalization to the first $k-1$ singular vectors V_{k-1} however we may obtain the next $V_{\cdot k}$ by iteration (4–5).

In the presence of missing data we may use (4–5) in the expectation maximization framework by filling $ij \notin R$ by $w_{ij} = \sigma_1 u_i^{(t)} \cdot v_j^{(t)}$. First we observe that if $v^{(t)}$ is a good approximation of $V_{\cdot 1}$, then $||v^{(t+1)}|| \approx \sigma_1$, hence the iteration turns to

$$u_i^{(t+1)} = \sum_{j:ij\in R} w_{ji} v_j^{(t)}/\sigma_1 + \sum_{j:ij\notin R} {v_j^{(t)}}^2 \cdot u_i^{(t)}, \quad (6)$$
$$v^{(t+1)} = W \cdot u^{(t)}/||u^{(t)}||. \quad (7)$$

This approach is split into two different heuristic implementations in the next two subsections. The RMSE for the basic implementation (6–7) is shown in Fig. 4.

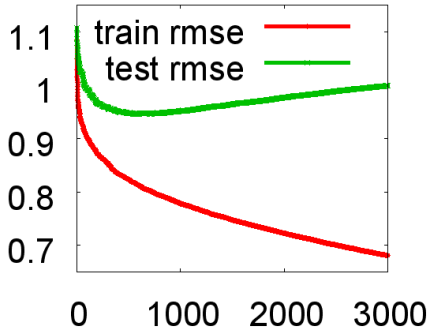

Figure 4: The rmse as the function of the number of iterations for the basic power iteration method given by equations (6–7).

4.1 Method of individual increments

We rewrite (6) as

$$u_i^{(t+1)} = \sum_{j:ij\in R} \left(\frac{w_{ji}}{\sigma_1} - v_j^{(t)} \cdot u_i^{(t)}\right) v_j^{(t)} + \sum_j {v_j^{(t)}}^2 \cdot u_i^{(t)}. \quad (8)$$

Given the assumption that the v are normalized that we may enforce in our algorithm, the second term is simply $u_i^{(t)}$. As a heuristic speedup, we split (8) into increments over u_i for individual j, replacing u_i by a new value in each step. This yields an algorithm with a cycle over

$$u_i \leftarrow u_i + (w_{ji}/\sigma_1 - v_j \cdot u_i) v_j \quad (9)$$

very closely reminiscent of Simon Funk's steps [12]

$$u_i \leftarrow (1 - \mathrm{lRate}) u_i + K(w_{ij} - \sigma_1 u_i v_j) v_j. \quad (10)$$

We use an idea similar to the convergence acceleration in Section 3: we multiply the increment in (9) by a factor δ that minimizes the RMSE of the approximation with modified u_i,

$$\delta = \sum_{j':ij'\in R} \left(w_{ij'} - \sigma v_{j'}(u_i + \delta v_j \mathrm{err}_{ij})\right)^2.$$

The minimum is easily computed as

$$\delta = \frac{\sum_{j':ij'\in R} v_{j'} \mathrm{err}_{ij'}}{\sum_{j':ij'\in R} v_{j'}^2} \sigma v_j \mathrm{err}_{ij}.$$

Unfortunately this algorithm appeared to diverge due to numeric errors for ratings with very small err_{ij}. Best results are obtained by using a median value near $\sigma/100$ very close to that suggested by [12].

4.2 Repeated hub and authority steps

If we repeat the "hub" iteration (4) more than once before an "authority" iteration (5), we observe no change in the full matrix case since the right hand side of (4) is independent of u. This is no longer the case for missing values however since imputation values depend on u. If we let

$$\Delta = \sum_{j:ij\in R} (w_{ij}/\sigma - u_i v_j) v_j \ , \quad v = \sum_{j:ij\in R} v_j^2$$

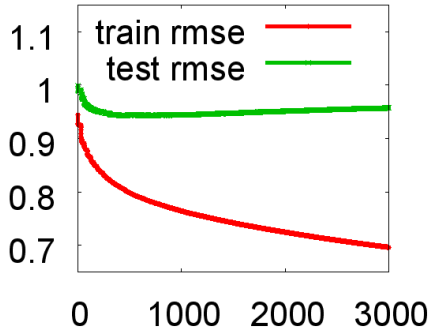

Figure 5: The rmse as the function of the number of iterations for power iteration with individual increments given by equation (9).

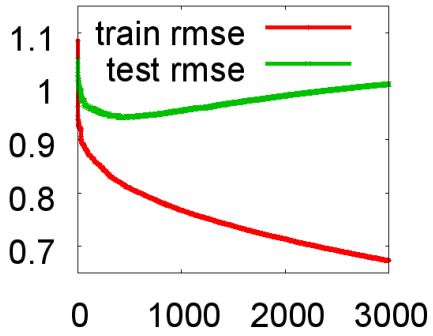

Figure 6: The rmse as the function of the number of iterations for power iteration with repeated hub and authority steps given by equation (12).

then t iterations of (6) give

$$u_i \leftarrow (1-h)^t u_i + (1-(1-v)^t)v^{-1}\Delta. \quad (11)$$

Best results are obtained in Fig. 6 if we use values $k = 5$ for the first singular vector computation and then decrease to 3, 2 and finally 1 for next dimensions. In addition we also combined this technique with individual increments as in (8):

$$u_i \leftarrow (1-v_j^2)^t u_i + (1-(1-v^2)^t)v^{-2}\Delta, \quad (12)$$

a formula again reminiscent of (10) of [12].

We may repeat the idea of the previous subsection and compute these steps individually for each i and j

5. A LEAST SQUARES APPROACH WITH ADAPTIVE DIMENSIONALITY

We give an algorithm that alternatedly computes and optimal U_k for a fixed V_k and then exchanges the role of U and V, similar to the "hub" (4) and "authority" (5) steps of the power iteration. For fixed V_k, optimizing the RMSE equation (2) can be done separately for the columns of U_k as

$$\sum_j (w_{ij\in R} - \sum_k \sigma_k u_{ki} v_{kj})^2 \quad (13)$$

as n regression problems.

The key idea in our algorithm is that once the regression is made separate for each user, we may adaptively select the right dimensionality for user i depending on the amount of ratings r given by her. Fig. 7 depicts RMSE for different values of constant K where we use the first $1+r/K$ dimensions in the above expression. We observe overfitting on the training set: the more dimensions used, the better is the RMSE; however over the probe set values of K between $25 \ldots 30$ perform the best; for a large number of iterations apparently $K = 25$ takes lead.

6. COMPARISON OF METHODS AND CONCLUSION

In our findings the best method is Lanczos with 10 dimensions. Unfortunately the iterations are relative costly and convergence boosting approaches tend to give minor improvements only. Power iteration based methods, though performing very similar steps as Funk's algorithm [12], tend to overfit the training set. We believe more careful tuning could improve performance. The runner up is the adaptive dimensionality least squares approach.

The most carefully tuned implementation of Funk's algorithm (10) [12] reaches an RMSE slightly below 0.92 on the probe set with 95 dimensions in over 100 iterations and $K = 0.015$, lRate $= .001$. By setting $K = 0$ performance similar to ours is reported. We believe a thorough measurement over our algorithms might find improvements, however our main goal here was to understand the behavior of the missing value problem by investigating a large number of related algorithms.

For further work we propose the implementation and comparison of fast SVD approximations and experiments with graphs of even larger scale. We also plan to mix results, a method that is known to yield significant improvement and in addition sometimes prefer weaker recommenders and thus slightly redraw the picture.

7. REFERENCES

[1] C. J. Alpert and S.-Z. Yao. Spectral partitioning: the more eigenvectors, the better. In *DAC '95: Proceedings of the 32nd ACM/IEEE conference on Design automation*, pages 195–200, New York, NY, USA, 1995. ACM Press.

[2] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 619–626, 2001.

[3] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[4] M. W. Berry. SVDPACK: A Fortran-77 software library for the sparse singular value decomposition. Technical report, University of Tennessee, Knoxville, TN, USA, 1992.

[5] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[6] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV (1)*,

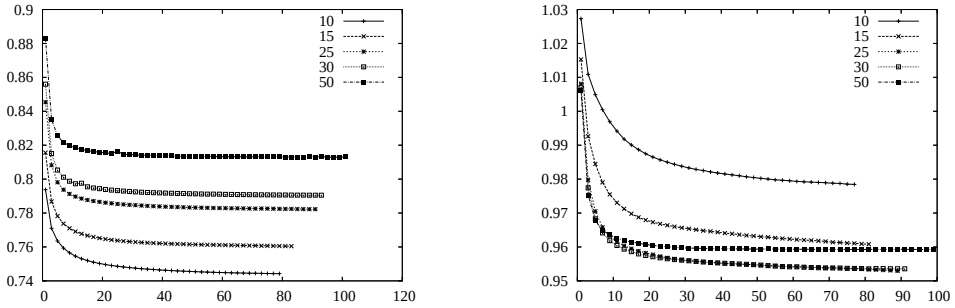

Figure 7: The rmse as the function of the number of iterations for adaptive least squares given by equation (13). Different curves correspond to values of K where we compute least squares over the first $1 + r/K$ dimensions for a user with r ratings. Left: rmse over the test set. Right: rmse over the probe set.

pages 707–720, 2002.

[7] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, USA, 2002. ACM Press.

[8] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divide-and-merge methodology for clustering. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 196–205, New York, NY, USA, 2005. ACM Press.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm.

[10] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 82–90, 2002.

[11] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[12] S. Funk. Netflix update: Try this at home. http://sifter.org/˜ simon/journal/20061211.html, 2006.

[13] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21:489–498, 1979.

[14] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

[15] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001.

[16] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1983.

[17] D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg. Jester 2.0 (poster abstract): evaluation of an new linear time collaborative filtering algorithm. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–292, New York, NY, USA, 1999. ACM Press.

[18] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[19] H. O. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14:174–194, 1958.

[20] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, O. Alter, D. Botstein, and P. Brown. Imputing missing data for gene expression arrays. Technical report, Department of Statistics, Stanford University, 2000.

[21] R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *IEEE:2000:ASF*, pages 367–377, 2000.

[22] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[23] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Comput. Vision*, 43(1):7–27, 2001.

[24] M. H. Pryor. The effects of singular value decomposition on collaborative filtering. Technical report, Dartmouth College, Hanover, NH, USA, 1998.

[25] A. Ruhe. Numerical computation of principal components when several observations are missing. Technical report, UMINF-48, Umeå, Sweden, 1974.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[28] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In T. Fawcett and N. Mishra, editors, *ICML*, pages 720–727. AAAI Press, 2003.

[29] O. G. Troyanskaya, M. Cantor, G. Sherlock, P. O.

Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[30] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 257–264, Washington, DC, USA, 2005. IEEE Computer Society.