Collaborative Filtering via Ensembles of Matrix Factorizations

Mingrui Wu
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38, 72076 Tübingen, Germany
mingrui.wu@tuebingen.mpg.de

ABSTRACT

We present a *Matrix Factorization* (MF) based approach for the Netflix Prize competition. Currently MF based algorithms are popular and have proved successful for collaborative filtering tasks. For the Netflix Prize competition, we adopt three different types of MF algorithms: regularized MF, maximum margin MF and non-negative MF. Furthermore, for each MF algorithm, instead of selecting the optimal parameters, we combine the results obtained with several parameters. With this method, we achieve a performance that is more than 6% better than the Netflix's own system.

Categories and Subject Descriptors

I.2.6 [**Machine Learning**]: Engineering applications

General Terms

Experimentation, Algorithms

Keywords

Netflix Prize, Collaborative Filtering, Matrix Factorization

1. INTRODUCTION

Collaborative Filtering (CF) aims at predicting users' interests in some given items based on their preferences so far, and the rating information of many other users.

CF can be regarded as a matrix completion task: given a matrix $\mathbf{Y} = [y_{ij}] \in \mathbb{R}^{m \times n}$, whose rows represent users, columns represent items, and non-zero elements represent known ratings, the goal is to predict the ratings for any given user-item pairs. In the matrix $\mathbf{Y}$, each element y_{ij} $(1 \leq i \leq m, 1 \leq j \leq n)$ belongs to $\{0,\ 1,\ \ldots, r\}$, where r is a given integer indicating the total level of ratings. Here $y_{ij} = 0$ means that no rating is provided by user i for item j, while $y_{ij} = s$, $1 \leq s \leq r$, is the rating given by user i for item j, with larger values corresponding to more preferences.

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

CF techniques can be applied to recommendation systems, which suggest items of interests to users based on their preferences. To build such a system, some information of the users and items can be helpful, such as ages, genders, nationalities of users and some characteristics of items such as prices. However, it is usually time consuming or even difficult to get such information. For example, when a user buys a book from a online book store, he may not be willing to reveal his personal information. Therefore CF approaches, predicting user preference only based on their ratings, can be a practical and useful solution. The fundamental assumption is that if two users rate many items similarly, they share similar tastes and hence will probably rate other items similarly.

Many approaches have been proposed for CF problems, such as memory based methods [2, 15] and model based algorithms [10, 13]. Memory based methods for CF predict new ratings by weighted averages of ratings of similar users or items, while model based algorithms use the rating data to learn a model of users and items for prediction.

Some CF algorithms only generate discrete rating values in $\{1,\ \ldots, r\}$, while others output real valued ratings. For performance evaluation, the *Mean Absolute Error* (MAE) and the *Rooted Mean Squared Error* (RMSE) are two widely used scores [9], which are defined as the following:

$$\mathrm{MAE}(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{|\mathcal{T}|} \sum_{ij \in \mathcal{T}} |\hat{y}_{ij} - y_{ij}| \quad (1)$$

$$\mathrm{RMSE}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{ij \in \mathcal{T}} (\hat{y}_{ij} - y_{ij})^2} \quad (2)$$

where $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ are the computed rating matrix and the true rating matrix respectively, while $\mathcal{T}$ is the set of user-item pairs for which we want to predict the ratings.

The choice between the above two criteria depends on the particular application. Currently in practice, MAE is common for CF algorithms with discrete ratings, while RMSE is popular for CF approaches that generate real valued outputs.

The Netflix Prize competition is an important event related to CF technologies. It is started and supported by Netflix, a company providing online movie rental services. Netflix has developed its own recommendation system called Cinematch. In October 2006, this company released a large movie rating dataset containing about 100 million ratings from over 480 thousand randomly selected customers on nearly 18 thousand movie items. The goal of the competition is to develop systems that can beat the accuracy of

Cinematch by a certain amount. In Netflix Prize competition, RMSE is adopted for performance evaluation and the algorithms in the competition are allowed to output real valued ratings. More details of this competition can be seen in [1].

We have participated in this event. The name of our team is HAT, which means "Have a Try". In this paper, we present our approach for the Netflix Prize competition. The remaining of the paper is organized as follows: In section 2 we describe three *Matrix Factorization* (MF) techniques that we have adopted for this competition: *Regularized Matrix Factorization* (RMF), *Maximum Margin Matrix Factorization* (MMMF) and *Non-negative Matrix Factorization* (NMF). In section 3, we present our ensemble approach that combines the results of the above mentioned three MF approaches. Experimental results are provided in section 4. And we conclude the paper in the last section.

2. CF VIA MATRIX FACTORIZATION

Low dimensional linear factor model [10, 16] is one of the most popular model based CF approaches. It assumes that only a small number of factors can influence the preferences, and that a user's preference on an item is determined by how each factor applies to the user and the item. This can be formulated as a MF problem. Namely, in a k-factor model, given the rating matrix $\mathbf{Y} \in \mathbb{R}^{m\times n}$, we want to find two matrices $\mathbf{U} \in \mathbb{R}^{m\times k}$ and $\mathbf{V} \in \mathbb{R}^{k\times n}$ such that

$$\mathbf{Y} \approx \mathbf{UV} \tag{3}$$

where k is a parameter.

The MF based CF approach basically fits a linear model to the rating matrix $\mathbf{Y}$. In this approach, user i ($1 \le i \le m$) is represented by $\mathbf{u}_i \in \mathbb{R}^k$, the i-th column of matrix $\mathbf{U}^\top$, while item j ($1 \le j \le n$) is modeled by $\mathbf{v}_j \in \mathbb{R}^k$, the j-th column of matrix $\mathbf{V}$. For example, in a movie rating system, each element of $\mathbf{v}_j$ can be a feature of movie j, such as whether it is an action movie, or whether the musics in the movie are pleasant, etc. And the corresponding elements in $\mathbf{u}_i$ indicate whether user i likes these features in a movie. Thus the final rating score to movie j given by user i is the linear combination of these factors, namely,

$$y_{ij} = \sum_{l=1}^{k} u_{il}v_{jl} = \mathbf{u}_i^\top \mathbf{v}_j$$

Up to now, We have focused on the MF based approaches for the Netflix Prize competition. In particular, we have tried three MF methods in our approach, which are described in the following three subsections respectively.

2.1 RMF

To find matrices $\mathbf{U}$ and $\mathbf{V}$ in (3), we can solve the following optimization problem:

$$\min_{\mathbf{U}\in\mathbb{R}^{m\times k},\mathbf{V}\in\mathbb{R}^{n\times k}} \frac{\lambda}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \sum_{ij\in\mathcal{S}} (y_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 \tag{4}$$

where $\lambda > 0$ is a regularization parameter, and

$$\mathbf{S} = \{ij \mid y_{ij} > 0\} \tag{5}$$

while $\|\cdot\|_F$ is the Frobenius 2-norm. Namely, $\|\mathbf{U}\|_F^2 = \sum_{pq} u_{pq}^2$.

Using RMF for CF is to fit a linear factor model (3) to the observed rating matrix $\mathbf{Y}$, such that for the known rating values y_{ij} ($1 \le i \le m, 1 \le j \le n$), the corresponding elements in $\mathbf{UV}$, i.e. $\mathbf{u}_i^\top \mathbf{v}_j$, are close to y_{ij}. The regularization term $\frac{\lambda}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2)$ is to restrict the domains of $\mathbf{U}$ and $\mathbf{V}$ in order to prevent over fitting, so that the resulting model has a good generalization performance. Namely, for any unknown rating y_{st} ($1 \le s \le m, 1 \le t \le n$), $\mathbf{u}_s^\top \mathbf{v}_t$ is a good estimations of y_{st}.

Let $L(\mathbf{U}, \mathbf{V})$ denote the objective function in (4). To minimize it, we can use gradient decent algorithm. To this end, we need to compute the gradients:

$$\frac{\partial L}{\partial u_{il}} = \lambda u_{il} - 2 \sum_{j|ij\in\mathcal{S}} (y_{ij} - \mathbf{u}_i^\top \mathbf{v}_j) v_{jl} \tag{6}$$

$$\frac{\partial L}{\partial v_{jl}} = \lambda v_{jl} - 2 \sum_{i|ij\in\mathcal{S}} (y_{ij} - \mathbf{u}_i^\top \mathbf{v}_j) u_{il} \tag{7}$$

Thus to solve problem (4), we can randomly initialize $\mathbf{U}$ and $\mathbf{V}$, and update these two matrices iteratively. Let $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$ denote $\mathbf{U}$ and $\mathbf{V}$ in the t-iteration, then

$$u_{il}^{(t+1)} = u_{il}^{(t)} - \tau \frac{\partial L}{\partial u_{il}^{(t)}} \tag{8}$$

$$v_{jl}^{(t+1)} = v_{jl}^{(t)} - \tau \frac{\partial L}{\partial v_{jl}^{(t)}} \tag{9}$$

Another possible way for minimizing (4) is to use an EM-like algorithm, where we update $\mathbf{U}$ and $\mathbf{V}$ alternatively. Namely, we first fix $\mathbf{V}$ and compute the optimal $\mathbf{U}$, then fix $\mathbf{U}$ and compute the optimal $\mathbf{V}$. This procedure is repeated until convergence. In this case, computing the optimal $\mathbf{U}$ (or $\mathbf{V}$) just requires solving an unconstrained quadratic optimization problem since $\mathbf{V}$ (or $\mathbf{U}$) is fixed, which can be done easily.

2.2 MMMF

MMMF is proposed for CF in [14, 4]. The output of MMMF is a discrete matrix whose elements only take values from $\{1, \ldots, r\}$. This is different from the MF methods described in the last and next subsections.

In MMMF, in addition to the factor matrices $\mathbf{U}$ and $\mathbf{V}$, $r-1$ thresholds θ_{ia} ($1 \le a \le r-1$) are also learned for each user i. For user i and item j, the value of $\mathbf{u}_i^\top \mathbf{v}_j$ is compared against $r-1$ thresholds θ_{ia}, $1 \le a \le r-1$, to generate discrete rating values in $\{1, \ldots, r\}$.

In MMMF, we need to minimize the following objective function with respect to $\mathbf{U}$, $\mathbf{V}$ and $\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is the matrix of all thresholds θ_{ia}, $1 \le i \le m$, $1 \le a \le r-1$

$$J(\mathbf{U}, \mathbf{V}, \boldsymbol{\theta}) = \frac{\lambda}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \sum_{a=1}^{r-1} \sum_{ij\in\mathcal{S}} h(T_{ij}^a[\theta_{ia} - \mathbf{u}_i^\top \mathbf{v}_j]) \tag{10}$$

where $\mathcal{S}$ has the same meaning as in (5) and

$$T_{ij}^a = \begin{cases} +1 & \text{if } a \ge y_{ij} \\ -1 & \text{if } a < y_{ij} \end{cases} \tag{11}$$

In (10), $h(\cdot)$ is a smoothed hinge loss function:

$$h(z) = \begin{cases} \frac{1}{2} - z & \text{if } z < 0 \\ 0 & \text{if z > 1} \\ \frac{1}{2}(1-z)^2 & \text{otherwise} \end{cases} \tag{12}$$

From the cost function (10), we can see that in contrast to RMF, MMMF does not require $\mathbf{u}_i^\top \mathbf{v}_j$ to be close to y_{ij}. Instead, as in the maximum margin approach used in the

Support Vector Machine (SVM) [17], it just expects that compared with θ_{ia}, $\mathbf{u}_i^\top \mathbf{v}_j$ should be as small as possible for $a \geq y_{ij}$, and as large as possible for $a < y_{ij}$, without caring too much about the specific value of $\mathbf{u}_i^\top \mathbf{v}_j$.

Figure 2.2 shows the hinge loss function that is adopted in SVM and the smoothed hinge loss function (12). Both of them can be used for function $h(\cdot)$, and they penalize each threshold violation without any penalty for being strongly on the correct side. However, as suggested in [14, 4], with the smoothed hinge loss, we can use gradient based algorithms to minimize the objective function, which is faster and easier than optimizing the non-smoothed one.

The derivative of the function $h(\cdot)$ is calculated as

$$h'(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z > 1 \\ z - 1 & \text{otherwise} \end{cases} \tag{13}$$

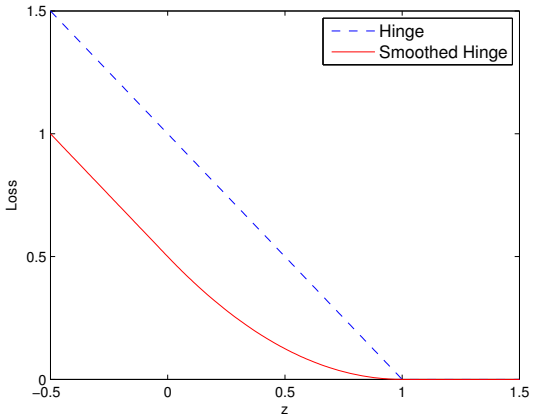

Figure 1: Hinge loss and smoothed hinge loss

And the gradients of the variables to be optimized can be calculated as:

$$\frac{\partial J}{\partial \theta_{ia}} = \sum_{j|ij\in\mathcal{S}} T_{ij}^a h'(T_{ij}^r[\theta_{ia} - \mathbf{u}_i^\top \mathbf{v}_j])$$

$$\frac{\partial J}{\partial u_{il}} = \lambda u_{il} - \sum_{a=1}^{r-1} \sum_{j|ij\in\mathcal{S}} T_{ij}^a h'(T_{ij}^a[\theta_{ia} - \mathbf{u}_i^\top \mathbf{v}_j]) v_{jl}$$

$$\frac{\partial J}{\partial v_{jl}} = \lambda v_{jl} - \sum_{a=1}^{r-1} \sum_{i|ij\in\mathcal{S}} T_{ij}^a h'(T_{ij}^a[\theta_{ia} - \mathbf{u}_i^\top \mathbf{v}_j]) u_{il}$$

In [14, 4], the conjugate gradient algorithm is adopted for MMMF, which can generally lead to better results than the simple gradient descent method. However, the gradient descent is much easier to implement than the conjugate gradient, so similarly as in subsection 2.1, here we also use the gradient descent approach for simplicity.

2.3 NMF

In subsections 2.1 and 2.2, we did not put any constraints on the elements of factor matrices $\mathbf{U}$ and $\mathbf{V}$. So these two matrices can contain both positive and negative elements.

In this subsection, we restrict the elements of $\mathbf{U}$ and $\mathbf{V}$ to be non-negative. This is reasonable from the factor analysis point of view. For example, suppose that in a movie rating system, u_{i1} represents whether user i likes action movies, a large u_{i1} means user i likes action movies very much, while a small u_{i1} means the opposite. Correspondingly, for vector $\mathbf{v}_j$, suppose that the larger the value of v_{j1}, the more likely that movie j is an action movie. Then it can be seen that $u_{i1} \times v_{j1}$ contributes to the rating of user i for movie j with respect to the factor of "action movie". Now let us consider an example, where $u_{i1} = 10$ and $v_{j1} = -10$. This results in a strong negative score -100. This means user i does not like this movie at all with respect to the "action movie" factor. However, in this case we can not really make such a strong conclusion, because user i may still like movie j although it is not an action movie. In contrast, if the elements of $\mathbf{U}$ and $\mathbf{V}$ are non-negative, then the above example becomes $u_{i1} = 10$ and $v_{j1} = 0$, leading to $u_{i1} \times v_{j1} = 0$, indicating that we can not say anything about the interest of user i in movie j only based on the "action movie" factor.

Inspired by the above, we have tried NMF algorithms in the Netflix Prize competition. NMF techniques have been proposed for learning part-based object representations [11, 6], document clustering [19], face recognition [18], and also for CF [20].

In NMF, we need to find the two non-negative matrices $\mathbf{U}$ and $\mathbf{V}$ that satisfy (3). To this end we need to minimize some cost functions that measure the difference between $\mathbf{Y}$ and $\mathbf{UV}$. Following the approach in [12], we adopt the following measure:

$$D(A||B) = \sum_{ij} (a_{ij} \log \frac{a_{ij}}{b_{ij}} - a_{ij} + b_{ij}) \tag{14}$$

which quantifies the difference between any two equal sized non-negative matrices $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$. This measure is lower bounded by zero, and vanishes if and only if $\mathbf{A} = \mathbf{B}$. Instead of "distance", it is usually called "divergence" of $\mathbf{A}$ from $\mathbf{B}$ since it is not symmetric in $\mathbf{A}$ and $\mathbf{B}$. Furthermore, it reduces to the Kullback-Leibler divergence, or relative entropy when $\sum_{ij} a_{ij} = \sum_{ij} b_{ij} = 1$, [12].

Thus, in order to perform NMF for CF, we need to minimize $D(\mathbf{Y}||\mathbf{UV})$ with respect to $\mathbf{U}$ and $\mathbf{V}$, subject to the constraints $u_{il}, v_{jl} \geq 0$, where $1 \leq i \leq m$, $1 \leq l \leq k$, $1 \leq j \leq n$. To this end, we can apply the multiplicative update rules suggested in [12]:

$$u_{il} \leftarrow u_{il} \frac{\sum_j v_{jl} y_{ij} / (\mathbf{u}_i^\top \mathbf{v}_j)}{\sum_a v_{al}} \tag{15}$$

$$v_{jl} \leftarrow v_{jl} \frac{\sum_i u_{il} y_{ij} / (\mathbf{u}_i^\top \mathbf{v}_j)}{\sum_a u_{al}} \tag{16}$$

To conclude this section, we point out that all the above mentioned three MF problems are not convex, therefore we probably obtain a local minimum of the cost function at the end of optimization. To mitigate this problem, a popular and reasonable choice is to run the algorithm several times with different initializations, and then pick up the one with the minimal cost function value. However, as will be described in the next section, we use the ensemble of different results rather than only take a particular single one.

3. MF ENSEMBLES

In the last section, we described RMF, MMMF and NMF approaches for CF. In order to get a concrete result, we still need to answer the following questions: Which of these three MF approaches should be applied? For each of them, what is the proper value of k, i.e. the number of factors?

For RMF and MMMF, how to determine the regularization parameter λ (cf. (4) and (10))?

One of the popular answer to the above questions is to use a validation set. Namely, a part of known ratings in the matrix $\mathbf{Y}$ is replaced with 0, then different MF approaches with different parameters are applied on the remaining ratings. The results are evaluated on the removed ratings and the one corresponding to the best performance is selected.

However, in the Netflix Prize competition, we have applied an ensemble approach, which learns and retains multiple models and combines their outputs for prediction. Ensemble approaches have already been used for classification, where different classifiers are combined together to reduce the generalization error [3]. For example, *boosting* algorithm [7], an ensemble method that learns a series of "weak" classifiers, each one focusing on correcting the errors made by the previous ones, has been widely used.

In [4], ensemble methods are applied to CF problems by combining the results generated by the MMMF algorithms with different initializations, and good results are obtained. In this paper, we use a simple ensemble scheme for CF. All of the three MF approaches mentioned before are applied. For each of them, different parameters are used. And the average of the resulting multiple predictions is computed as the final prediction result.

One reason why ensemble methods can work is that they can reduce the variance of learning algorithms [8, 5]. In a learning problem, usually several different models can give similar accuracy on the training data. If we choose one of these models to compute the output, there is a risk that the chosen model will not perform well on the unseen test data. In this case, a simple vote or average of these equally-good models can reduce the risk. More detailed analysis of ensemble learning methods can be found in [5].

4. EXPERIMENTAL RESULTS

Experimental results are summarized in Figure 4. It presents the RMSE values of the 16 submissions on the quiz dataset provided by Netflix.[1]

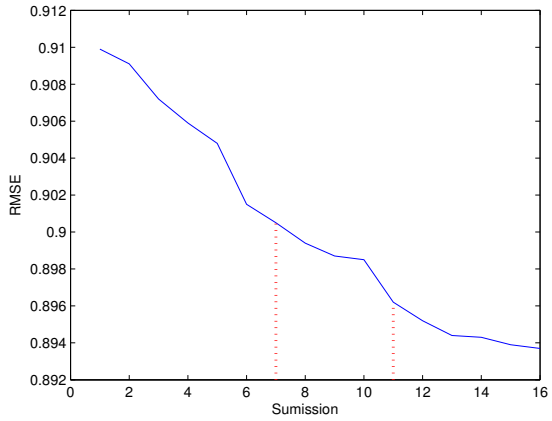

Figure 2: RMSE of our 16 submissions on the quiz dataset.

In each submission, the result is the combination of previous results and the current one, which is obtained with new algorithms or new parameters. The RMSE curve is partitioned into three segments. The first segment contains the results obtained by RMF, the second one corresponds to the results from MMMF, combined with RMF, while the last segment consists of the results obtained by NMF, combined with RMF and MMMF.

For all these submissions, the number of factors k (cf. (3)) ranges from 10 to 100, the regularization parameter λ (cf. (4) and (10)) changes from 0.01 to 1, while the learning rate τ (cf. (8)–(9)) equals 0.01 or 0.02.

Note that actually our latest submission represents a combination of more than 50 different results, because each of the 16 submissions itself is also the average of several results obtained with one particular MF algorithm using different parameters. And before we started to run a new MF algorithm, based on the Probe dataset provided by Netflix, we had observed that combining more parameters with the same MF method could hardly improve the result any more.

It can be seen from Figure 4 that by combining the results of different parameters and different MF approaches, the RMSE can be improved. And our latest submission improves the performance of Cinematch by more than 6%. This illustrates the effectiveness of the ensemble approache.

5. CONCLUSIONS

We have described a Matrix Factorization (MF) based approach that we have applied to the Netflix Prize competition. Three different MF techniques are adopted: regularized MF, maximum margin MF and non-negative MF. To improve the performance, we combine the results of these MF methods with different parameters. Experimental results are provided to validate the effectiveness of our approach.

6. ACKNOWLEDGMENTS

We thank Netflix for releasing a nice dataset and supporting the competition. We thank Bing Liu, chair of KDD cup and workshop, for his kind feedbacks.

7. REFERENCES

[1] J. Bennett and S. Lanning. The netflix prize. In *Proc. KDD Cup and Workshop*, 2007.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[4] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. 23rd International Conference on Machine Learning*, 2006.

[5] T. G. Dietterich. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks*. The MIT Press, 2002.

[6] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16*. 2004.

[7] Y. Freund and R. E. Schapire. Experiments with a new boosing algorithm. In *Proc. 13th International Conference on Machine Learning*, 1996.

[1]We have totally submitted 18 results. Two of them are not correct due to the errors in our program. Therefore we do not show the results of these two submissions.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Verlag, 2001.

[9] J. Herlocker, J. Konstan, J. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[10] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22:89–115, 2004.

[11] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negativ matrix factorization. *Nature*, 401:788–791, 1999.

[12] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 12*. 2000.

[13] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proc. 21st International Conference on Machine Learning*, 2004.

[14] J. M. M. Rennie and N. Srebro. Fast maximum margin factorization for collaborative prediction. In *Proc. 22nd International Conference on Machine Learning*, 2005.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW Conference*, 2001.

[16] N. Srebro and T. Jaakkola. Weighted low rank approaimation. In *Proc. 20th International Conference on Machine Learning*, 2003.

[17] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[18] Y. Wang and Y. Jia. Non-negative matrix factorization framework for face recognition. *Journal of Pattern Recognition and Artificial Intelligence*, 19:495–511, 2005.

[19] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. 26th International ACM SIGIR Conference on Research and Development in Information Retriveal*. 2003.

[20] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proc. 6th SIAM Conference on Data Mining*. 2006.