Fast Maximum Margin Matrix Factorization for Collaborative Prediction

Jason D. M. Rennie JRENNIE@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA

Nathan Srebro NATI@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, ON, CANADA

Abstract

Maximum Margin Matrix Factorization (MMMF) was recently suggested (Srebro et al., 2005) as a convex, infinite dimensional alternative to low-rank approximations and standard factor models. MMMF can be formulated as a semi-definite programming (SDP) and learned using standard SDP solvers. However, current SDP solvers can only handle MMMF problems on matrices of dimensionality up to a few hundred. Here, we investigate a direct gradient-based optimization method for MMMF and demonstrate it on large collaborative prediction problems. We compare against results obtained by Marlin (2004) and find that MMMF substantially outperforms all nine methods he tested.

1. Introduction

"Collaborative prediction" refers to the task of predicting preferences of users based on their preferences so far, and how they relate to the preferences of other users. For example, in a collaborative prediction movie recommendation system, the inputs to the system are user ratings on movies the users have already seen. Prediction of user preferences on movies they have not yet seen are then based on patterns in the partially observed rating matrix. The setting can be formalized as a matrix completion problem—completing entries in a partially observed data matrix Y. This approach contrasts with a more traditional feature-based approach where predictions are made based on features of the movies (e.g. genre, year, actors, external reviews) and the users (e.g. age, gender, explicitly specified preferences). Users "collaborate" by sharing their ratings instead of relying on external information.

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005.

A common approach to collaborative prediction is to fit a factor model to the data, and use it in order to make further predictions (Azar et al., 2001; Billsus & Pazzani, 1998; Hofmann, 2004; Marlin & Zemel, 2004; Canny, 2004). The premise behind a low-dimensional factor model is that there is only a small number of *factors* influencing the preferences, and that a user's preference vector is determined by how each factor applies to that user. In a linear factor model, each factor is a preference vector, and a user's preferences correspond to a linear combination of these factor vectors, with user-specific coefficients. Thus, for n users and d items, the preferences according to a k-factor model are given by the product of an $n \times k$ *coefficient matrix* U (each row representing the extent to which each factor is used) and a $k \times d$ *factor matrix* V' whose rows are the factors. The preference matrices which admit such a factorization are matrices of rank at most k. Thus, training such a linear factor model amounts to approximating the observed preferences Y with a low-rank matrix X.

The low-rank matrix X that minimizes the sum-squared distance to a *fully observed* target matrix Y is given by the leading singular components of Y and can be efficiently found. However, in a collaborative prediction setting, only some of the entries of Y are observed, and the low-rank matrix X minimizing the sum-squared distance to the *observed* entries can no longer be computed in terms of a singular value decomposition. In fact, the problem of finding a low-rank approximation to a partially observed matrix is a difficult non-convex problem with many local minima, for which only local search heuristics are known (Srebro & Jaakkola, 2003).

Furthermore, especially when predicting discrete values such as ratings, loss functions other then sum-squared loss are often more appropriate: loss corresponding to a specific probabilistic model (as in pLSA (Hofmann, 2004) and Exponential-PCA (Collins et al., 2002)) or loss functions such as hinge loss. Finding a low-rank matrix X minimizing loss functions other then squared-error is a non-convex optimization problem with multiple local minima, even when the the target matrix Y is fully observed[1].

Low-rank approximations constrain the dimensionality of the factorization $X = UV'$, i.e. the number of allowed factors. Other constraints, such as sparsity and non-negativity (Lee & Seung, 1999), have also been suggested for better capturing the structure in Y, and also lead to non-convex optimization problems.

Recently, Srebro et al. (2005) suggested a formulation termed "Maximum Margin Matrix Factorization" (MMMF), constraining the *norms* of U and V instead of their dimensionality. Viewed as a factor model, this corresponds to constraining the overall "strength" of the factors, rather than their number. That is, a potentially infinite number of factors is allowed, but only a few of them are allowed to be very important. For example, when modeling movie ratings, there might be a very strong factor corresponding to the amount of violence in the movie, slightly weaker factors corresponding to its comic and dramatic value, and additional factors of decaying importance corresponding to more subtle features such as the magnificence of the scenery and appeal of the musical score.

Mathematically, constraining the norms of U and V corresponds to constraining the *trace-norm* (sum of singular values) of X. Interestingly, this is a convex constraint, and so finding a matrix X with a low-norm factorization minimizing any convex loss versus a partially (or fully) observed target matrix Y, is a convex optimization problem. This contrasts sharply with rank-constraints, which are not convex constraints, yielding non-convex optimization problems as described above. In fact, the trace-norm (sum of singular values) has also been suggested as a convex surrogate for the rank (number of non-zero singular values) in control applications (Fazel et al., 2001).

Fazel et al. (2001) show how a trace-norm constraint can be written in terms of a linear and semi-definite constraints. By using this form, Srebro et al. (2005) formulate MMMF as semi-definite programming (SDP) and employ standard SDP solvers to find maximum margin matrix factorizations. However, such generic solvers are only able to handle problems with no more than a few tens of thousands of constraints, corresponding to about ten thousand observations (observed user-item pairs), i.e. about a hundred users and a hundred items. This is far from the size of typical collaborative prediction problems, with thousands of users and items, yielding millions of observations.

[1]The problem is non-convex even when minimizing the sum-squared error, but for the special case of minimizing the sum-squared error versus a fully observed target matrix, all local minima are global (Srebro & Jaakkola, 2003)

In this paper, we investigate methods for seeking a MMMF by directly optimizing the factorization $X = UV'$. That is, we perform gradient-based local search on the matrices U and V. Using such methods, we are able to find maximum margin matrix factorizations for a realistically sized collaborative prediction data set, and demonstrate the competitiveness of MMMF versus other collaborative prediction methods.

In Section 2 we review the formulation of Maximum Margin Matrix Factorization suggested by Srebro et al. (2005). In Section 3 we describe the optimization methods we deploy, and in Section 4 we report our experiments using these methods.

2. Maximum Margin Matrix Factorization

Before presenting Maximum Margin Matrix Factorizations, we begin by revisiting low-rank collaborative prediction. We then present the MMMF formulation for binary and ordinal rating observations.

2.1. Factor Models as Feature Learning

Consider fitting an $n \times d$ target matrix Y with a rank-k matrix $X = UV'$, where $U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{d\times k}$. If one of the matrices, say U, is fixed, and only the other matrix V needs to be learned, then fitting each column of the target matrix Y is a separate linear prediction problem. Each row of U functions as a "feature vector;" each row of V is a linear predictor, predicting the entries in the corresponding column of Y based on the "features" in U.

In collaborative prediction, both U and V are unknown and need to be estimated. This can be thought of as learning feature vectors (rows in U) for each of the rows of Y, enabling good linear prediction across all of the prediction problems (columns of Y) concurrently, each with a different linear predictor (columns of V'). The features are learned without any external information or constraints which is impossible for a single prediction task (we would use the labels as features). The underlying assumption that enables us to do this in a collaborative prediction situation is that the prediction tasks (columns of Y) are *related*, in that the same features can be used for all of them, though possibly in different ways.

Consider adding to the loss a penalty term which is the sum of squares of entries in U and V, i.e. $\|U\|^2_{\text{Fro}} + \|V\|^2_{\text{Fro}}$ ($\|\cdot\|_{\text{Fro}}$ denotes the Frobenius norm). Each "conditional" problem (fitting U given V and vice versa) again decomposes into a collection of standard, this time regularized, linear prediction problems. With an appropriate loss function, or constraints on the observed entries, these correspond to large-margin linear discrimination problems. For example, if we learn a binary observation matrix by minimizing a hinge loss (roughly, the distance from the classification margin) plus such a regularization term, each conditional problem decomposes into a collection of support vector machines (SVMs). As in SVMs, constraining U and V to be low-dimensional is no longer necessary, as generalization performance is guaranteed by the constraints on the norms (Srebro & Schraibman, 2005).

2.2. Low-Norm Factorizations

Matrices with a factorization $X = UV'$, where U and V have low Frobenius norm (recall that the dimensionality of U and V is no longer bounded!), can be characterized in several equivalent ways:

Lemma 1. *For any matrix X the following are all equal:*

1. $\min_{\substack{U,V\\X=UV'}} \|U\|_{Fro}\|V\|_{Fro}$
2. $\min_{\substack{U,V\\X=UV'}} \frac{1}{2}(\|U\|^2_{Fro} + \|V\|^2_{Fro})$
3. *The sum of the singular values of X, i.e.* $\operatorname{tr}\Lambda$ *where $X = U\Lambda V'$ is the singular value decomposition of X.*

Definition 1. *The trace norm $\|X\|_\Sigma$ of a matrix is given by the three quantities in Lemma 1.*

The trace norm is also known as the *nuclear norm* and the *Ky-Fan n-norm.*

It is straight-forward to verify that the trace-norm is a convex function: For a convex combination $X = \alpha X_1 + (1-\alpha)X_2$ consider the factorizations $X_1 = U_1V_1'$ and $X_2 = U_2V_2'$ s.t. $\|X_1\|_\Sigma = \frac{1}{2}(\|U_1\|^2_{\text{Fro}} + \|V_1\|^2_{\text{Fro}})$ and respectively for X_2. We can now consider a factorization $X = UV'$ where U, V are the block matrices $U = \left[\sqrt{\alpha}U_1, \sqrt{1-\alpha}U_2\right]$ and $V = \left[\sqrt{\alpha}V_1, \sqrt{1-\alpha}V_2\right]$, yielding:

$$\begin{aligned}\|X\|_\Sigma &\le \frac{1}{2}(\|U\|^2_{\text{Fro}} + \|V\|^2_{\text{Fro}})\\ &= \alpha\frac{1}{2}(\|U_1\|^2_{\text{Fro}} + \|V_1\|^2_{\text{Fro}}) + (1-\alpha)\frac{1}{2}(\|U_2\|^2_{\text{Fro}} + \|V_2\|^2_{\text{Fro}})\\ &= \alpha\|X_1\|_\Sigma + (1-\alpha)\|X_2\|_\Sigma \qquad (1)\end{aligned}$$

We can conclude that minimizing the trace-norm, combined with any convex loss (e.g. sum-squared error, log-likelihood for a binomial model, logistic loss) or constraint, is a convex optimization problem. Here, we focus specifically on hinge-loss (as in SVMs) and a generalization of the hinge-loss appropriate for discrete ordinal ratings, as in movie rating data sets (e.g. 1–5 "stars").

2.3. Formulation

First consider binary labels $Y \in \{\pm 1\}^{n\times m}$ and *hard-margin matrix factorization*, where we seek a minimum trace norm matrix X that matches the observed labels with a margin of one: $Y_{ij}X_{ij} \ge 1$ for all $ij \in S$ (S is the set of observed index pairs). By introducing slack variables $\xi_{ij} \ge 0$, we can relax this hard constraint, requiring $Y_{ij}X_{ij} \ge 1 - \xi_{ij}$, and minimizing a trade-off between the trace-norm and the slack. Minimizing the slack variables is equivalent to minimizing the hinge-loss $h(z) = (1-z)_+ = \max(0, 1-z)$, and we can write the optimization problem as:

$$\text{minimize } \|X\|_\Sigma + C\sum_{ij\in S} h(Y_{ij}X_{ij}), \qquad (2)$$

where C is a trade-off constant.

As in maximum-margin linear discrimination, there is an inverse dependence between the norm and the margin. Fixing the margin and minimizing the trace norm (as in the above formulation) is equivalent to fixing the trace norm and maximizing the margin. As in large-margin discrimination with certain infinite dimensional (e.g. radial) kernels, the data is always separable with sufficiently high trace norm (a trace norm of $\sqrt{n|S|}$ is sufficient to attain a margin of one).

Ratings The data sets we more frequently encounter in collaborative prediction problem are of ordinal ratings $Y_{ij} \in \{1, 2, \ldots, R\}$. To relate the real-valued X_{ij} to the discrete Y_{ij} we use $R-1$ thresholds $\theta_1, \ldots, \theta_{R-1}$. In a hard-margin setting, we would require

$$\theta_{Y_{ij}-1} + 1 \le X_{ij} \le \theta_{Y_{ij}} - 1$$

where for simplicity of notation $\theta_0 = -\infty$ and $\theta_R = \infty$. When adding slack, we not only penalize the violation of the two immediate constraints $\theta_{Y_{ij}-1} + 1 \le X_{ij}$ and $\le X_{ij} \le \theta_{Y_{ij}} - 1$, but also the violation of all other implied threshold constraint $X_{ij} \ge \theta_r + 1$ for $r < Y_{ij}$ and $X_{ij} \le \theta_r - 1$ for $r \ge Y_{ij}$. Doing so emphasizes the cost of crossing multiple rating-boundaries and yields a loss function which upper bounds the mean-absolute-error (MAE—the difference, in levels, between the predicted level and the true level). The resulting optimization problem is:

$$\begin{aligned}&\text{minimize } \|X\|_\Sigma + C\sum_{ij\in S}\left(\sum_{r=1}^{Y_{ij}-1} h(X_{ij} - \theta_r) + \sum_{r=Y_{ij}}^{R-1} h(\theta_r - X_{ij})\right)\\ \equiv\ &\text{minimize } \|X\|_\Sigma + C\sum_{ij\in S}\sum_{r=1}^{R-1} h(T^r_{ij}(\theta_r - X_{ij})) \qquad (3)\end{aligned}$$

where $T^r_{ij} = \begin{cases} +1 & \text{for } r \ge Y_{ij}\\ -1 & \text{for } r < Y_{ij}\end{cases}$.

The thresholds θ_r can be learned from the data. Furthermore, a different set of thresholds can be learned for each user, allowing users to "use ratings differently" and alleviates the need to normalize the data. The problem can then be written as:

$$\text{minimize } \|X\|_\Sigma + C \sum_{ij \in S} \sum_{r=1}^{R-1} h(T_{ij}^r(\theta_{ir} - X_{ij})) \quad (4)$$

where the variables optimized over are the matrix X and the thresholds θ. In other work, we find that such a formulation is highly effective for rating prediction (Rennie & Srebro, 2005).

Although the problem was formulated here as a single optimization problem with a combined objective, $\|X\|_\Sigma + C \cdot \textit{error}$, it should really be viewed as a dual-objective problem of balancing between low trace-norm and low error. Considering the entire set of attainable $(\|X\|_\Sigma, \text{error})$ pairs, the true object of interest is the exterior "front" of this set, i.e. the set of matrices X for which it is not possible to reduce one of the two objectives without increasing the other. This "front" can be found by varying the value of C from zero (hard-margin) to infinity (no norm regularization).

All optimization problems discussed in this section can be written as semi-definite programs (Srebro et al., 2005).

3. Optimization Methods

We describe here a local search heursitic for the problem (4). Instead of searching over X, we search over pairs of matrices (U, V), as well as sets of thresholds θ, and attempt to minimize the objective:

$$J(U, V, \theta) \doteq \frac{1}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2) + C \sum_{r=1}^{R-1} \sum_{ij \in S} h\Big(T_{ij}^r(\theta_{ir} - U_i V_j')\Big). \quad (5)$$

For any U, V we have $\|UV\|_\Sigma \le \frac{1}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2)$ and so $J(U, V, \theta)$ upper bounds the minimization objective of (4), where $X = UV'$. Furthermore, for any X, and in particular the X minimizing (4), some factorization $X = UV'$ achieves $\|X\|_\Sigma = \frac{1}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2)$. The minimization problem (4) is therefore equivalent to:

$$\text{minimize } J(U, V, \theta). \quad (6)$$

The advantage of considering (6) instead of (4) is that $\|X\|_\Sigma$ is a complicated non-differentiable function for which it is not easy to find the subdifrential. Finding good descent directions for (4) is not easy. On the other hand, the objective $J(U, V, \theta)$ is fairly simple. Ignoring for the moment the non-differentiability of $h(z) = (1 - z)_+$ at one, the gradient of $J(U, V, \theta)$ is easy to compute. The partial derivative with respect to each element of U is:

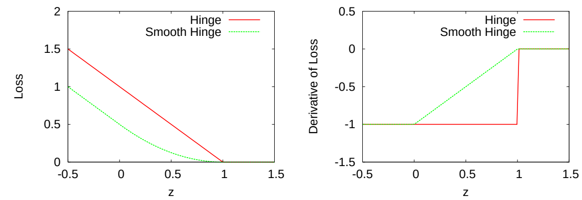

Figure 1. Shown are the loss function values (left) and gradients (right) for the Hinge and Smooth Hinge. Note that the gradients are identical outside the region $z \in (0, 1)$.

$$\frac{\partial J}{\partial U_{ia}} = U_{ia} - C \sum_{r=1}^{R-1} \sum_{j|ij \in S} T_{ij}(k) h'\Big(T_{ij}^r(\theta_{ir} - U_i V_j')\Big) V_{ja} \quad (7)$$

The partial derivative with respect to V_{ja} is analogous. The partial derivative with respect to θ_{ik} is

$$\frac{\partial J}{\partial \theta_{ir}} = C \sum_{j|ij \in S} T_{ij}^r h'\Big(T_{ij}^r(\theta_{ir} - U_i V_j')\Big). \quad (8)$$

With the gradient in-hand, we can turn to gradient descent methods for localy optimizing $J(U, V, \theta)$. The disadvantage of considering (6) instead of (4) is that although the minimization objective in (4) *is* a convex function of X, θ, the objective $J(U, V, \theta)$ *is not* a convex function of U, V. This is potentially bothersome, and might inhibit convergence to the global minimum.

3.1. Smooth Hinge

In the previous discussion, we ignored the non-differentiability of the Hinge loss function $h(z)$ at $z = 1$. In order to give us a smooth optimization surface, we use an alternative to the Hinge loss, which we refer to as the *Smooth Hinge*. Figure 1 shows the Hinge and Smooth Hinge loss functions. The Smooth Hinge shares many properties with the Hinge, but is much easier to optimize directly via gradient descent methods. Like the Hinge, the Smooth Hinge is not sensitive to outliers, and does not continuously "reward" the model for increasing the output value for an example. This contrasts with other smooth loss functions, such as the truncated quadratic (which is sensitive to outliers) and the Logistic (which "rewards" large output values). We use the Smooth Hinge and the corresponding objective for our experiments in Section 4.

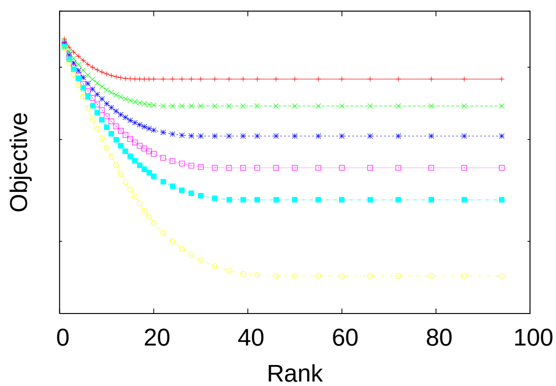

Figure 2. Objective value after learning U and V for various regularization values on a 100x100 subset of the MovieLens data set. The "rank" axis indicates the number of columns we used for U and V (the value of k). Each line corresponds to a different regularization constant (C). Each point corresponds to separate, randomly initialized optimization.

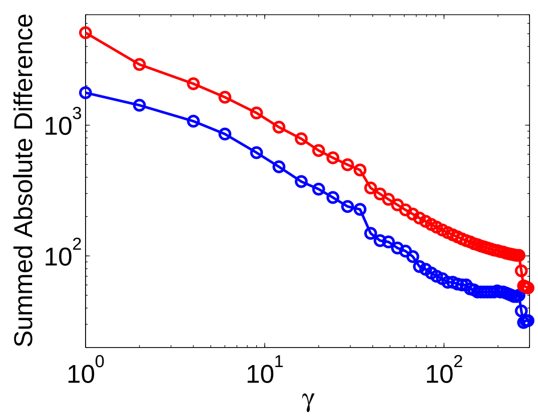

Figure 3. Shown is an example of summed absolute difference between the X (top) and Y (bottom) matrices produced by CG solution of the SGL objective and SDP solution of the Hinge objective as a function of γ. The matrix is 100x100 and there are 5 rating levels, so absolute difference for Y could be as large as 40,000.

3.2. Implementation Details

In the MMMF formulation, we use the norm of X for regularization, so the rank of the U, V decomposition of X is effectively unbounded. However, there is no need to consider rank larger than $k = \max(n, d)$. And, in practice, we find that we can get away with much smaller values of k. For our experiments in Section 4, we use a value of $k = 100$. While using a too-small value of k may lead to a sub-optimal solution, there tend to be a wide range of values of k that yield near-identical solutions. Figure 2 shows the objective value for various regularization values and rank-truncated U, V matrices on a subset of the MovieLens data set. Although X is 100x100, values of $k \in (20, 40)$ (depending on C) achieve nearly the same objective value as $k = 100$. Learning using truncated U, V is significantly faster than using $k = \max(n, d)$.

For optimization of U, V and θ, we used the Polak-Ribière variant of Conjugate Gradients (Shewchuk, 1994; Nocedal & Wright, 1999) with the consecutive gradient independence test (Nocedal & Wright, 1999) to determine when to "reset" the direction of exploration. We used the Secant line search suggested by (Shewchuk, 1994), which uses linear interpolation to find an approximate root of the directional derivative. We found PR-CG to be sufficiently fast, yielding matrix completion on a 30000x1648 EachMovie rating matrix (4% observed entries, using rank $k = 100$ U, V matrices) in about 15 hours of computation time (single 3.06Ghz Pentium 4 CPU).

3.3. Local Minima

The direct optimization problem (6) is not convex. In fact, $U = V = 0$ is a critical point that is clearly not the global optimum. We know that there are critical points; there may even be local minima. The important practical question is: how likely we are to get stuck at a local minimum with reasonable, e.g. random, initialization? We would like to compare the solution found by our local-search Conjugate Gradients (CG) algorithm against the global optimum. We do not currently have the tools to find the global optimum of the Smooth Hinge objective. But, we can solve for the global optimum of the Hinge objective (on small problems) using an SDP solver. Then, to evaluate our CG optimization method, we use an upper bound on the Hinge loss function that can be made increasingly tight as we increase a parameter. We call this upper bound the shifted generalized Logistic[2] (SGL for short):

$$h(z) = \frac{1}{\gamma} \log(1 + \exp(\gamma(1 - z))). \tag{9}$$

Note that as $\gamma \to \infty$, this function approaches $h(z) = (1 - z)_+$. In tests on the 100x100 subset of the MovieLens data set, we find that CG optimization of the SGL objective finds solutions very close to those found by the SDP solver. Figure 3 shows differences in the solution matrices of a CG optimization of the SGL objective compared to a SDP optimization of the Hinge objective. As γ increases, the X and Y matrices produced by the CG optimization grow increasingly similar to those produced by the SDP optimization. Numerical issues made it impossible for us to explore values of $\gamma > 300$, but the trend is clear—the

[2] Zhang and Oles (2001) discuss the generalized Logistic.

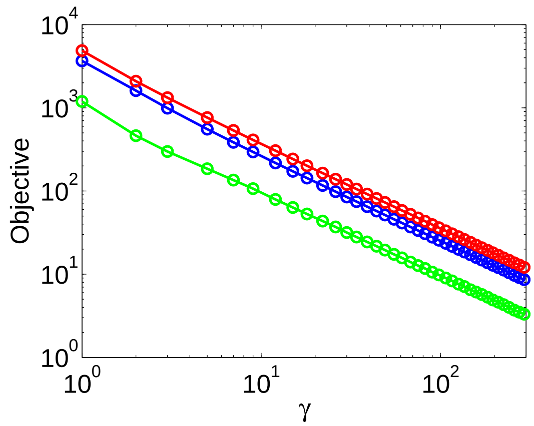

Figure 4. Shown is an example of objective values as a function of γ. We subtract the Hinge objective value for the SDP solution from all y-axis values. The top two lines show the SGL objective values for the (top) SDP, and (middle) CG solutions. The bottom line gives the Hinge objective value for the CG solution. In all three cases, there is a clear trend toward zero as $\gamma \to 0$.

difference tends to zero as $\gamma \to \infty$. Results on a variety of regularization parameters and randomly drawn training sets are similar. Figure 4 shows objective values compared to the Hinge objective of the (optimal) SDP solution. The SGL loss of the SDP solution (top line) is uniformly greater than SGL loss of the CG solution (middle line). This indicates that the CG solution is close to the global minimum of the SGL objective. Furthermore, the Hinge loss of the CG solution (bottom line) tends toward zero as $\gamma \to \infty$, indicating that, in the limit, the CG solution will achive the same global minimum that is found by the SDP solver. We also found that CG always returned the minimum Frobenius norm U, V decomposition for the X matrix. That is, given the X matrix returned by CG, no U, V decomposition would have yielded a lower objective value.

4. Experiments

Here we report on experiments conducted on the 1M Movielens and EachMovie data sets. We mimic the setup used by Marlin (2004) and compare against his results. We find that MMMF with the Smooth Hinge loss substantially outperforms all algorithms that Marlin tested.

Marlin tested two types of generalization, "weak" and "strong." We conducted test on both types. "Weak generalization" is a single stage process which involves the learner filling-in missing entries of a rating matrix. "Strong generalization" is a two-stage process where the learner trains a model on one set of users and then is asked to make predictions on a new set of users. The learner is given sample ratings on the new set of users, but may not utilize those ratings until after the initial model is constructed.

The EachMovie data set provides 2.6 million ratings for 74,424 users and 1,648 movies. There are six possible rating values, $\{1, 2, \ldots, 6\}$. As did Marlin, we discarded users with fewer than 20 ratings. This left us with 36,656 users. We randomly selected 30,000 users for the "weak generalization" set and used the remaining 6,656 users for the "strong generalization" set.

The MovieLens data set provides 1 million ratings for 6,040 users and 3,952 movies. There are five possible rating values, $\{1, 2, \ldots, 5\}$. All users had 20 or more ratings, so we utilized all users. We randomly selected 5,000 users for the "weak generalization" set and used the remaining 1,040 users for the "strong generalization" set.

As did Marlin, we repeated the selection process three times for each data set. We randomly withheld one movie for each user to construct the test set. To select the regularization parameter for MMMF, we withheld one additional movie per user to construct a validation set; we selected the regularization parameter with lowest validation error. We computed Normalized Mean Absolute Error (NMAE) as Marlin describes. The normalization constant for MovieLens (5 rating values) is 1.6; the normalization constant for EachMovie (6 rating values) is 1.944. For both data sets, we truncated the U and V matrices at rank $k = 100$. This led to (weak generalization) U and V matrices of size 30000x100 and 1648x100 for EachMove and 6040x100 and 3952x100 for MovieLens (respectively). The U and V matrix sizes influenced the computational time required for optimization. We found that optimization of a single training set and regularization parameter for EachMovie took about 15 hours on a single 3.06Ghz Pentium 4 CPU; a single optimization run for MovieLens took about 5 hours.

Table 1 give the results of our experiments. We reproduce the results of the two algorithms that yielded lowest errors in Marlin's experiments. MMMF gives lower NMAE on both data sets and for both weak and strong generalization. The differences are substantial—in all cases, the MMMF errors are at least one standard deviation better than the best result reported by Marlin. In many cases, the MMMF result is better by a margin of multiple standard deviations.

5. Discussion

In this work, we have shown that it is possible to "scale-up" MMMF to large problems. We used gradient descent on U, V and θ to find an approximate minimum to the MMMF objective. Although $J(U, V, \theta)$ is not convex, an empirical analysis indicated that local minima are, at worst, rare. However, there is still the need to determine whether local minima exist and how likely it is that gradient descent will get stuck in such minima.

	EachMovie			MovieLens	
Algorithm	Weak NMAE	Strong NMAE	Algorithm	Weak NMAE	Strong NMAE
URP	.4422 $\pm$ .0008	.4557 $\pm$ .0008	URP	.4341 $\pm$ .0023	.4444 $\pm$ .0032
Attitude	.4520 $\pm$ .0016	.4550 $\pm$ .0023	Attitude	.4320 $\pm$ .0055	.4375 $\pm$ .0028
MMMF	**.4397** $\pm$ .0006	**.4341** $\pm$ .0025	MMMF	**.4156** $\pm$ .0037	**.4203** $\pm$ .0138

Table 1. MMMF results on (left) EachMovie and (right) MovieLens; we also reproduce Marlin's results for the two best-performing algorithms (URP and Attitude). We report average and standard deviation of Normalized Mean Absolute Error (NMAE) across the three splits of users. For MMMF, we selected the regularization parameter based on a validation set taken from the training data; Marlin's results represent the lowest NMAE across a range of regularization parameters.

Acknowledgments

Jason Rennie was supported in part by the DARPA CALO project. We thank Tommi Jaakkola for valuable comments and ideas.

References

Azar, Y., Fiat, A., Karlin, A. R., McSherry, F., & Saia, J. (2001). Spectral analysis of data. *ACM Symposium on Theory of Computing* (pp. 619–626).

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. *Proc. 15th International Conf. on Machine Learning* (pp. 46–54). Morgan Kaufmann, San Francisco, CA.

Canny, J. (2004). Gap: a factor model for discrete data. *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval* (pp. 122–129). Sheffield, United Kingdom: ACM Press.

Collins, M., Dasgupta, S., & Schapire, R. (2002). A generalization of principal component analysis to the exponential family. *Advances in Neural Information Processing Systems 14*.

Fazel, M., Hindi, H., & Boyd, S. P. (2001). A rank minimization heuristic with application to minimum order system approximation. *Proceedings American Control Conference*.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, *22*, 89–115.

Lee, D., & Seung, H. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*, 788–791.

Marlin, B. (2004). Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, Computer Science Department.

Marlin, B., & Zemel, R. S. (2004). The multiple multiplicative factor model for collaborative filtering. *Proceedings of the 21st International Conference on Machine Learning*.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Springer-Verlag.

Rennie, J. D. M., & Srebro, N. (2005). Loss functions for preference levels: Regression with discrete ordered labels. *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/~jrs/jrspapers.html.

Srebro, N., & Jaakkola, T. (2003). Weighted low rank approximation. *20th International Conference on Machine Learning*.

Srebro, N., Rennie, J. D. M., & Jaakkola, T. (2005). Maximum margin matrix factorization. *Advances In Neural Information Processing Systems 17*.

Srebro, N., & Schraibman, A. (2005). Rank, trace-norm and max-norm. *Proceedings of the 18th Annual Conference on Learning Theory*.

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*, 5–31.