Collaborative Prediction Using Ensembles of Maximum Margin Matrix Factorizations

Dennis DeCoste DECOSTED@YAHOO-INC.COM
Yahoo! Research, 3333 Empire Avenue, Burbank, CA 91504 USA

Abstract

Fast gradient-based methods for Maximum Margin Matrix Factorization (MMMF) were recently shown to have great promise (Rennie & Srebro, 2005), including significantly outperforming the previous state-of-the-art methods on some standard collaborative prediction benchmarks (including MovieLens). In this paper, we investigate ways to further improve the performance of MMMF, by casting it within an ensemble approach. We explore and evaluate a variety of alternative ways to define such ensembles. We show that our resulting ensembles can perform significantly better than a single MMMF model, along multiple evaluation metrics. In fact, we find that ensembles of partially trained MMMF models can sometimes even give better predictions in total training time comparable to a single MMMF model.

1. Introduction

A growing number of interesting problems can often be productively viewed largely as "collaborative prediction" tasks. They are fundamentally matrix completion tasks: given a matrix, whose rows represent *users*, columns represent *items*, and non-zero elements represent known ratings, be able to predict the rating for any new given user,item pair. Examples include classic recommendation tasks (e.g. for books, movies, music, etc), as well as entirely new web-based applications and needs which are emerging, such as predicting preferred news articles, search results, and web sites. Due in part to an explosion in online applications and communities, rating data of various sorts (both explicit, e.g. "5 stars", and implicit, e.g. repeated visits or clicks) is growing rapidly. Thus, although content-based methods (such as movie recommendations based on user's actor and genre preferences) will undoubtedly continue to be useful (especially in "cold start" cases for new users with few ratings), there are increasing opportunities (and needs) for suitable machine learning methods to enable the growing masses of rating data to largely speak for themselves.

One key technical challenge is that the matrix is typically huge and sparse – e.g. thousands to millions of rows and columns, with only 1% or fewer of the elements being non-zero (i.e. known). Another challenge is dealing with the increased noise (or even sabotage) one would expect in the rating set, as the user population becomes more diverse. In this paper we propose an approach which shows promise to help address both challenges.

A linear factor model is a simple, natural, and well-studied (e.g. (Billus & Pazzani, 1998), (Hofmann, 2004),(Marlin & Zemel, 2004)) approach to collaborative prediction: fit the n-by-d ratings matrix by a low rank decomposition UV', where U is n-by-k and V is d-by-k and $k \ll \min(n, d)$. The assumption is that there is some small number of factors that capture each user's preferences and each item's nature, and the learned U and V respectively project users and items to that lower dimensional space. By using all the data to discover a small set of underlying factors, the factor model reflects the collaborative effect across users. Unfortunately, most approaches (including those based on spectral methods) do not fully exploit the spareness of the ratings and/or assume squared loss (which is not best suited for ordinal ratings).

Recently, (Rennie & Srebro, 2005) proposed a very promising new approach (MMMF) and reported it to be both effective (e.g. significantly improving on previous best performance on large benchmarks such as MovieLens) and efficient (using gradient-based optimization that often scales roughly linearly with the number of ratings). Despite good initial results,

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

MMMF has potential problems, including local minima and impact of noise (including outlier users).

In this paper we explore the use of ensemble methods (including bagging) with MMMF to help overcome these problems. We find that our improvements sometimes help – for example, enabling MMMF to increase its predictive accuracy on the MovieLens data by additional significant (and record level) amounts and sometimes simply achieving similar accuracies with mild reductions in variance.

2. Review of MMMF

This section briefly summarizes Maximum Margin Matrix Factorization for collaborative prediction on ratings. (Rennie & Srebro, 2005) gives more motivations.

We are given a sparse n-by-d matrix Y of ordinal ratings, where $Y_{ij} \in \{0, 1, 2, \ldots, R\}$. $Y_{ij} = 0$ indicates that no rating is provided by user i for item j.

MMMF approximates ratings matrix Y with a linear factor model $X = UV'$, by learning a n-by-k matrix U and a d-by-k matrix V, for some user-supplied low rank k (typically much less than $\min(n, d)$; e.g. 100).

A prediction for any $X_{ij} \in \Re$ is mapped into $\hat{Y}_{ij} \in \{1, 2, \ldots, R\}$ by comparing the predicted X_{ij} value (given by $X = UV'$) against the $R-1$ discretization thresholds θ_{ir} which are also learned for each user i.

MMMF minimizes the objective with loss function h:

$$J(U,V,\theta) = \frac{\lambda}{2}(||U||_F^2 + ||V||_F^2) + \sum_{r=1}^{R-1} \sum_{ij \in \mathcal{S}} h(T_{ij}^r[\theta_{ir} - U_i V_j'])$$

where [1]

$$S \equiv \{\, ij \mid Y_{ij} > 0 \,\},$$

$$T_{ij}^r = \begin{cases} +1 & \text{if } r \geq Y_{ij} \\ -1 & \text{if } r < Y_{ij}. \end{cases}$$

T_{ij}^r essentially imposes extra penalty for each threshold that is violated by the predictions of the current UV'. This makes $J(U, V, \theta)$ upper bound the Mean Absolute Error, a popular score for rating prediction:

$$\text{MAE}(\hat{Y}, Y) = \frac{1}{|\mathcal{S}|} \sum_{ij \in \mathcal{S}} |\hat{Y}_{ij} - Y_{ij}|.$$

Hinge loss is used (much like in support vector machines) to robustly (i.e. L_1) penalize each threshold violation without any penalty for being strongly on the correct side. However, a smoothed hinge loss is used to make $J(U, V, \theta)$ differentiable and optimization easier:

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

with derivative:

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

resulting in gradients:

$$\frac{\delta J}{\delta \theta_{ir}} = \sum_{j|ij \in \mathcal{S}} T_{ij}^r \; h'(T_{ij}^r[\theta_{ir} - U_i V_j'])$$

$$\frac{\delta J}{\delta U_{ia}} = \lambda U_{ia} - \sum_{r=1}^{R-1} \sum_{j|ij \in \mathcal{S}} T_{ij}^r \; h'(T_{ij}^r[\theta_{ir} - U_i V_j'])\; V_{ja}$$

$$\frac{\delta J}{\delta V_{ja}} = \lambda V_{ja} - \sum_{r=1}^{R-1} \sum_{i|ij \in \mathcal{S}} T_{ij}^r \; h'(T_{ij}^r[\theta_{ir} - U_i V_j'])\; U_{ia}$$

Unlike other factor model approaches (such as non-negative matrix factorization (Lee & Seung, 1999)), MMMF uses regularization on the norms (instead of enforcing a specific low rank per se) of U and V, through the use of parameter λ. Selecting a relatively small $k \ll \min(n, d)$) is still desirable in practice, however, simply to avoid excessive costs in computing gradients at each iteration.

(Rennie & Srebro, 2005) argues that conjugate gradient (CG) optimization of $J(U, V, \theta)$ is often much faster and more suitable for large data sets than the earlier semi-definite programming (SDP) formulation proposed in (Srebro et al., 2005). The earlier SDP formulation has the advantage of being a convex optimization, and so always converges to its global minima. However, empirically, gradient-based MMMF often converges to good minima, as Section 4 reconfirms.

In this paper, we use "MMMF" to refer to the fast gradient-based approach of (Rennie & Srebro, 2005).

2.1. A Note on Computational Complexity

(Rennie & Srebro, 2005) report that "a single optimization run for MovieLens took about 5 hours" (on a 3GHz Pentium 4); for 100 CG iterations (Rennie, 2006), each typically involving 2-3 gradient updates during line search. Such MovieLens runs had

[1] $||Z||_F^2$ is the Frobenius 2-norm: $\sum_{ij} Z_{ij}^2$.

$n = 5,000$, $d = 3,952$, $k = 100$, $R = 5$, and $|\mathcal{S}| \approx 1,000,000$. With $k \gg R$, the computational complexity of each update is dominated by $|\mathcal{S}|$ dot products each of size k (to compute each $U_i V_j'$). For MovieLens, this requires about 100 million multiply-and-accumulate (MAC) operations. This cost can be shared across J, $\frac{\delta J}{\delta U}$, $\frac{\delta J}{\delta V}$, and $\frac{\delta J}{\delta \theta}$. Updating all components of $\frac{\delta J}{\delta U}$, and similarly for $\frac{\delta J}{\delta V}$, requires additional $k\ |\mathcal{S}|$ MACs. Altogether, each MovieLens update requires about 300 million MACs. On a 3 GHz Pentium 4, with roughly 3 clocks required per MAC, ideal performance would be roughly 0.3 seconds per update and 1 second per CG iteration. Our current implementation achieves 10 seconds per CG iteration – within a factor of 10 from ideal, most likely due to our not having yet fully optimized cache memory reuse nor maximally pipelined multiple operations. In contrast, the earlier (Rennie & Srebro, 2005) reported times amount to closer to a slower 180 seconds per CG iteration.

In our experiments, our MMMF implementation is 15 to 20 times faster overall than reported in (Rennie & Srebro, 2005) – e.g. reducing identical trainings for MovieLens from the previously reported 5 hours down to just 15 minutes. Thus, we find MMMF is even more practical for large data sets than originally reported.

3. MMMF Ensembles

As reported in (Rennie & Srebro, 2005) (and summarized in Tables 1 and 3 later), the prediction performance of MMMF on large benchmark rating prediction tasks are already impressive, beating the previous best reported in (Marlin, 2004) by significant amounts.

However, in extensive experiments on a variety of benchmark and commercial data, we noticed several problems with MMMF. For one, there were clear cases of local minima. Also, we suspected that, much like other related and more well-studied maximum margin approaches (such as support vector machines), MMMF might be somewhat susceptible to outliers (e.g. abnormal or even malicious raters) and other noise.

For classification tasks, ensemble methods, including bagging, are standard and often effective for such problems. [2] It is natural to ask whether ensembles and bagging could similarly help MMMF as well. However, unlike classification, the outputs of rating prediction are ordinal values. Due to ordinal ordering, plurality voting seems potentially suboptimal.

Therefore, towards better performance for this special case, we considered several alternatives:

1. *plurality voting* – pick the most common rating across the ensemble, breaking ties randomly.
2. *voting by averaging* – average the predicted discrete ratings and round to nearest discrete rating.
3. *voting by confidence* – weight votes according to how close the pre-discretized predictions were to MMMF's learned thresholds (θ).

Voting by confidence is intuitively appealing because often MMMF predictions are nearly equi-distance between two thresholds, in which case voting entirely for one rating or the other seems suboptimal, particularly given the ordinal nature of ratings. Ideally, it might be best to try to learn calibrations of MMMF outputs into probabilities, perhaps analogous to calibrating SVM outputs in probabilities as in (Platt, 1999). However, in lieu of such interesting future work, we tried something simpler which might serve as a useful baseline. For each prediction, a sum total of 1.0 of confidence is distributed between two ratings: the one predicted and the one who's threshold is closer to the raw predicted (i.e. X_{ij}) value. These confidence values are then used to weight the votes for those two ratings by that ensemble member. We put 1.0 confidence on a prediction which is halfway between its two thresholds and let it drop linearly to 0.5 (for each side of the threshold) if the value reaches either threshold. For the 2 extreme ratings, we assume their raw prediction ranges are same as their 1 neighbor's.

We tried two ways to generate ensemble members:

1. *multiple random weight seeds* – each weight seed being a random initial set of U, V, θ values.
2. *baggings* – randomly sample *users* with replacement (sample size same as original training set).

We suspected simple multiple random seeds may suffice for some suboptimalities due to local minima, and provide a baseline to see when bagging is essential.

We only considered bagging based on dropping or repeating an entire user's rating set (minus any test holdouts) at one time. This was based on a somewhat informed assumption that the quality of raters in the real world varies wildly, with some better off being dropped or amplified. Still, it might also be useful to consider bagging on individual votes, for future work. [3]

[2] E.g. bagging can even be useful for SVMs; e.g. see (Valentini & Dietterich, 2004).

[3] Bagged MMMF always requires 2 optimizations stages (the second with fixed V), even for "Weak" tests (see Section 4.1), because the training set will not contain all test users (bagging drops an example roughly 37% of the time).

4. Experiments

We conducted extensive large-scale experiments comparing the basic MMMF method and various ensemble and voting formulations, consuming over two years of total CPU time on a 128-CPU Beowulf-class cluster.

As in (Rennie & Srebro, 2005), we focus on two large well-known benchmark data sets (MovieLens and EachMovie) and use the same train/test setups used in both (Rennie & Srebro, 2005) and (Marlin, 2004), as summarized below.

4.1. Experimental Setup

Marlin defined both "weak" and "strong" holdout tests of generalization. In "weak" testing, one rating (i.e. movie) is randomly held out from each user's rating set to form the testing set. In "strong" testing, first a subset ($\mathcal{G}$) of test users is randomly selected and completely removed from the training set. The "strong" test set consists of one rating randomly selected from each user in $\mathcal{G}$. The remaining ratings for each user in $\mathcal{G}$ can be used to tweak the model during strong testing, but only after the initial prediction model over the bulk of the users (i.e. all not in $\mathcal{G}$) is learned. This corresponds to a usage case where a model is learned for a large set of initial users and then adjusted later for some relatively small set of new users ($\mathcal{G}$). In the MMMF approach, this is done by first finding a factorization UV' without using any users in $\mathcal{G}$ and then reoptimizing with V fixed, to find a new factorization $U_{\mathcal{G}}V'$, where $U_{\mathcal{G}}$'s rows correspond to the users $\mathcal{G}$.

The MovieLens data set consists of 6,040 users (all having 20 or more ratings) and d=3,952 movies. It contains about 1 million ratings over R=5 values. Strong set $\mathcal{G}$ consists of 1,040 randomly selected users.

The EachMovie data set consists of 36,656 users (after dropping all users with fewer than 20 ratings) and d=1,648 movies. It contains about 2.6 million ratings over R=6 values. $\mathcal{G}$ consists of 6,656 random users.

Marlin defined "Normalized MAE" (NMAE) scores so that random guessing averages a NMAE score of 1. NMAE is computed by dividing the MAE score by a factor which depends on R. The factor is 1.6 for $R = 5$ (MovieLens) and 1.944 for $R = 6$ (EachMovie).

To allow direct comparisons, we used the same value $k = 100$ as in (Rennie & Srebro, 2005) for the number (k) of columns for both U and V.

We repeated the above 4 train/test setups (i.e. Weak and Strong, for MovieLens and EachMovie) three times each, as in (Marlin, 2004). Tables 1 and 3 (discussed in Section 4.3) show the mean and standard deviation of NMAE scores for each experiment, including ensembles with voting decided by plurality. [4]

4.2. Determining λ Regularizer

To find a good regularization setting (λ) for each data set, we first determined and removed all data that would end up in any of the 12 test sets in our experiments. For the remaining data of MovieLens (or EachMovies), we held out one rating per user as validation data, ran MMMF (without any ensembles) for each candidate λ value, and computed the (weak) NMAE score for each λ for MovieLens (or EachMovies). For each candidate λ value we repeated this four times. Figures 2 and 1 plot the average scores (surrounded by outer plots of mins,maxs). The candidate λ's were $10^{\frac{i}{16}}$, for $i = 4, 5, \ldots, 40$. The best average λ for each data set was then used for all experiments described below. The plots and minima were remarkably sharp. Although somewhat suboptimal (i.e. union of all tests was held out), experiment time was more managable.

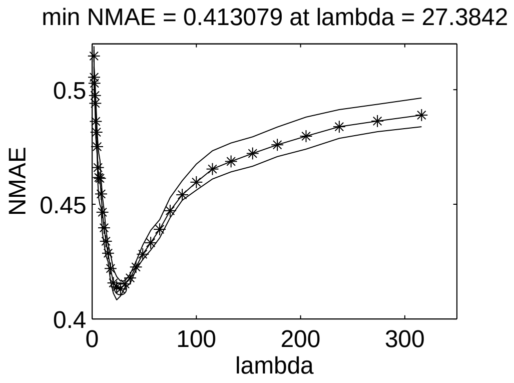

Figure 1. MovieLens: Validation scores for each λ.

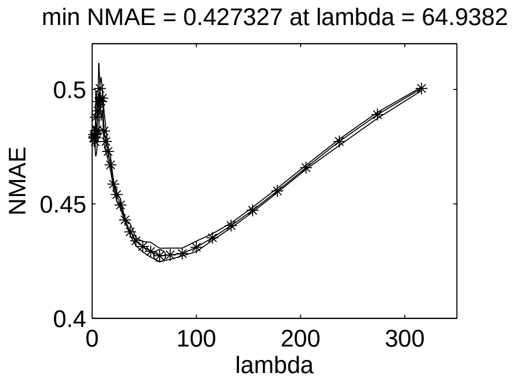

Figure 2. EachMovie: Validation scores for each λ.

[4]Unless otherwise stated, we ran MMMF for 100 conjugate gradient iterations for each case, as was done in (Rennie & Srebro, 2005) (Rennie, 2006). For Strong tests (or bagged MMMF's), we similarly ran 100 iterations of conjugate gradient on the 2nd (fixed V) optimization.

Table 1. MovieLens: vote by plurality.

Method	Weak NMAE	Strong NMAE
URP	0.4341 ± 0.0023	0.4444 ± 0.0032
Attitude	0.4320 ± 0.0055	0.4375 ± 0.0028
MMMF	0.4156 ± 0.0037	0.4203 ± 0.0138
MMMF-A1	0.4282 ± 0.0040	0.4237 ± 0.0126
MMMF-A10	0.4256 ± 0.0019	0.4208 ± 0.0092
MMMF-A100	0.4255 ± 0.0023	0.4183 ± 0.0068
MMMF-A200	0.4257 ± 0.0025	0.4180 ± 0.0068
MMMF-W10	0.4157 ± 0.0047	0.4137 ± 0.0127
MMMF-B10	0.4102 ± 0.0014	0.4109 ± 0.0106
MMMF-W100	0.4129 ± 0.0051	0.4077 ± 0.0100
MMMF-B100	**0.4046** ± 0.0006	0.4075 ± 0.0109
MMMF-W200	0.4127 ± 0.0043	0.4079 ± 0.0100
MMMF-B200	**0.4052** ± 0.0004	0.4085 ± 0.0093
MMMF20-A1	0.4245 ± 0.0066	0.4167 ± 0.0121
MMMF20-A5	0.4261 ± 0.0012	0.4178 ± 0.0109
MMMF20-W5	0.4135 ± 0.0017	0.4087 ± 0.0162
MMMF20-B5	0.4105 ± 0.0020	0.4159 ± 0.0089

Table 3. EachMovie: vote by plurality.

Method	Weak NMAE	Strong NMAE
URP	0.4422 ± 0.0008	0.4457 ± 0.0008
Attitude	0.4520 ± 0.0016	0.4550 ± 0.0023
MMMF	0.4397 ± 0.0006	0.4341 ± 0.0025
MMMF-A1	0.4291 ± 0.0021	0.4301 ± 0.0030
MMMF-A10	0.4288 ± 0.0020	0.4302 ± 0.0029
MMMF-A100	0.4287 ± 0.0020	0.4300 ± 0.0031
MMMF-W10	0.4288 ± 0.0023	0.4302 ± 0.0031
MMMF-B10	0.4293 ± 0.0023	0.4303 ± 0.0029
MMMF-W100	0.4288 ± 0.0022	0.4299 ± 0.0032
MMMF-B100	0.4287 ± 0.0023	0.4301 ± 0.0035
MMMF1k-W10	0.4287 ± 0.0023	0.4303 ± 0.0031
MMMF1k-B10	0.4291 ± 0.0020	0.4305 ± 0.0031

Table 2. MovieLens: vote by averaging.

Method	Weak NMAE	Strong NMAE
MMMF	0.4156 ± 0.0037	0.4203 ± 0.0138
MMMF-W10	0.4134 ± 0.0028	0.4139 ± 0.0101
MMMF-B10	**0.4057** ± 0.0030	0.4097 ± 0.0135
MMMF-W100	0.4122 ± 0.0036	0.4087 ± 0.0103
MMMF-B100	**0.4029** ± 0.0027	0.4071 ± 0.0093
MMMF-W200	0.4117 ± 0.0025	0.4089 ± 0.0088
MMMF-B200	**0.4034** ± 0.0037	0.4081 ± 0.0093

Table 4. EachMovie: vote by averaging.

Method	Weak NMAE	Strong NMAE
MMMF	0.4397 ± 0.0006	0.4341 ± 0.0025
MMMF-W10	0.4288 ± 0.0023	0.4301 ± 0.0026
MMMF-B10	0.4292 ± 0.0024	0.4304 ± 0.0024
MMMF-W100	0.4288 ± 0.0022	0.4299 ± 0.0031
MMMF-B100	0.4287 ± 0.0021	0.4295 ± 0.0030
MMMF1k-W10	0.4288 ± 0.0023	0.4302 ± 0.0031
MMMF1k-B10	0.4293 ± 0.0021	0.4306 ± 0.0027

4.3. Discussion of Results: Plurality Voting

Tables 1 and 3 show, for MovieLens and EachMovie respectively, mean ± standard deviation of NMAE scores over 3 random trials, for both Weak and Strong tests.

The first three methods in each table recall the previous best reported results. Lines labelled `URP` and `ATTITUDE` recall the best performers in (Marlin, 2004). Line labelled `MMMF` recalls the improved results reported in (Rennie & Srebro, 2005).

Lines labelled `MMMF-Axx` indicate the average statistics over xx random weight seedings (i.e. initial values of U, V, and θ) of single MMMFs (without any voting).

Lines labelled `MMMF-Wxx` indicate the voted results for ensembles trained using xx random weight seedings of MMMF. Lines labelled `MMMF-Bxx` indicate the voted results for ensembles trained using xx random baggings of MMMF (including one random weight seed per bag). Votes over these ensembles were determined by plurality (as described in Section 3).

Lines labelled `MMMFyy-Bxx` indicate the voted results for ensembles trained using xx random baggings – as in `MMMF-Bxx`, except this time only using yy iterations (e.g. 20 or 1000), instead of the default 100. Similarly for lines labelled `MMMFyy-Axx` and `MMMFyy-Wxx`.

Result `MMMF-A1` is equivalent to a single MMMF optimization (per trial). It should be comparable to the `MMMF` result (recalled from (Rennie & Srebro, 2005)). For MovieLens, our single MMMF results are roughly comparable (within standard deviations). However, for EachMovie, `MMMF-A1` is better than `MMMF`, presumably due to variance in our respective λ selections. Therefore, in this paper we consider an ensemble approach to improvement on a single MMMF only if it beats the `MMMF-A1` result.

As expected, `MMMF-Axx` results are no better than any `MMMF-Wxx` or `MMMF-Bxx` results – they simply provide a reality check that improvements for ensembles are not due simply to variance in the initial weight seedings. In general, as xx increases we expect and note that `MMMF-Axx` converges to an accurate mean (and lower variance) estimate of the performance of a single MMMF.

We notice that the bagged ensembles often reduce variance more than simple reseeded ensembles; e.g. standard deviation of 0.0006 for `MMMF-B100` vs 0.0051 for `MMMF-W100` for MovieLens Weak tests. Ensembles with more bags tend to reduce variance more, as expected. The NMAE mean accuracy is sometimes also significantly better as well; e.g. 0.4046 for `MMMF-B100` vs 0.4129 for `MMMF-W100`, for MovieLens Weak tests.

Also interesting is that `MMMF20-B5` gives significantly better Weak and Strong NMAE for MovieLens than `MMMF-A1`, even though both have comparable training times (i.e. 20 iterations for 5 ensemble bags versus 100 iterations for a single MMMF). This provides some preliminary support for the suggestion in Section 3 to bag numerous approximate MMMF trainings instead of training a single MMMF more fully.

We noticed that MMMF ensembles give some significant improvements for MovieLens, but none for EachMovies. To see whether this might be due to our early stopping of conjugate gradient at 100 iterations, we also trained small MMMF ensembles on EachMovie out to 1000 iterations. However, neither `MMMF1k-W10` nor `MMMF1k-B10` improved significantly over their counterparts (`MMMF-W10` and `MMMF1k-B10`).

4.4. Results: Voting by Averaging

Tables 2 and 4 report results for voting by averaging (as described in Section 3), using the same `MMMF-*` notation as in Tables 1 and 3. Since `MMMF-Axx` results do not depend on the type of voting, we do not repeat them in these tables. However, for comparison we still first repeat the `MMMF` result recalled from (Rennie & Srebro, 2005).

Comparison of Tables 1 and 2 shows some slight advantage for voting by averaging instead of by plurality, in the case of MovieLens. In Table 2, the result of 0.4029 Weak NMAE for MovieLens for `MMMF-B100` is significantly better than the `MMMF` result of 0.4156 – *despite* our ensembles starting at a somewhat worse baseline (of 0.4282 for `MMMF-A1`), presumably because of differences in λ selection. This gives some evidence to the notion that ensembles of MMMF may provide some degree of protection from somewhat bad local minima or suboptimal λ selection. At the very least, it seems noteworthy that none of our ensemble results are significantly worse than single MMMF results. Thus, if one can afford the extra training costs, it seems worthwhile to try such ensembles, in case it helps – sometimes significantly so.

4.5. Results: Voting by Confidence

We attempted voting by confidences (as described in Section 3), but so far found it to be worse that the other voting methods we tried above. For example, for `MMMF-B10` for Weak and Strong EachMovie tests, it gave NMAE scores of 0.4453 and 0.4449 respectively. Part of the problem may well be due to our simple linear model of confidence across the thresholds; using more sophisticated future methods based on analogs to (Platt, 1999) may improve the voting by confidence approach.

In any case, during our investigation, we discovered that often neighboring thresholds (θ) for a given user were in reverse order (i.e. $\theta_{ir} > \theta_{i,r+1}$) sometimes by large amounts. The MMMF forumation does not enforce that constraint, so this should not be surprising – except how often it seems to occur in some cases. Future work is needed to understand how important this problem is, and how costly it might be to avoid during optimization.

5. Discussion

In this paper we have made several important contributions.

Our results, especially on MovieLens, provide an existence proof that ensembles of MMMF's can give significantly stronger predictions than a single MMMF (the previous state-of-the-art on the benchmark data sets that we examined). We reported cases where simple voting (over multiple trainings using different random seedings) helped significantly and others where bagged voting helped even more. In extensive experiments representing over 2 years of CPU time, we found no cases where bagged voting performed significantly worse than simple voting, nor simple voting worse than a single MMMF. Thus, our results suggest that ensembles of MMMF's offer a promising way to help overcome the potential local minima suboptimalities (an open concern raised in (Rennie & Srebro, 2005)).

We also showed that MMMF can be implemented to be much faster (at least 15-fold faster, so far) than originally reported. Combined with the previous scaling advantages that MMMF was already reported to have over the earlier semi-definite programming formulation of (Srebro et al., 2005), our work here establishes ensembles of conjugate gradient-based MMMF to be perhaps the leading contender, especially for large real data sets.

There have been some related work on combining and voting among multiple recommendation systems. However, those approaches have focused on combining diverse models, especially where some are content based and others rating/collaborative based (e.g. (Melville et al., 2002), (Good et al., 1999), and (Claypool et al., 1999)). They also did not utilize state-of-the-art large-scale methods such as MMMF, for which further improvements can be challenging to demonstrate.

To fairly put our improvements in proper perspective, it is important to note that neither minimizing simple scores such as MAE, nor performing collaborative prediction in general, is sufficient for solving end-to-end recommendation tasks in practice. For one, there is the well-known issue that typically the best items to recommend next are not simply the ones that a given user is most likely to rate highly, since the most useful recommendations typically would also factor in many others issues, such as surprise and diversity. In the end, in-field evaluations (whether active, such as surveys, or passive, such as usage/churn/engagement statistics) are essential, since what ultimately matters most is whether the users find the recommendations useful, not how well the system predicts their preferences per se.

Nevertheless, significant improvements to MAE scores still seems a worthy intermediate goal for continued research. Though simplistic, MAE scores on benchmark datasets such as MovieLens and EachMovie do facilitate direct comparison of previous methods on easy to replicate benchmark results. The MAE score seems a useful surrogate for the ultimate goal, if one assumes that accurately predicting preferences is a useful (e.g. necessary but not sufficient) intermediate step. Also, methods to improve the efficiency of acquiring new ratings data from users, such as active learning, might also benefit from even relatively modest improvements in predictive model accuracy – perhaps especially from improvements in characterizing model uncertainties.

Future work includes integrating this approach with content-based information (e.g. movie actors and genres), to see how much of the gain due to ensembles remains once other such background information is exploited.

Other interesting ensemble alternatives to bagging which might be worth exploring for MMMF and ordinal rating prediction in general include: variance optimized bagging (vogging) (Derbeko et al., 2002) and analogs to online Bayes Point Machines (Harrington et al., 2003)) (i.e. sampling without replacement).

Also, as discussed in Section 3, voting based on probability calibration methods such as (Platt, 1999) might be preferrable to our simple linear model of confidence.

Acknowledgments

Thanks to Jason Rennie for discussions and making his MMMF code publicly available to help replicate his results for comparisons.

References

Billus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. *ICML.*

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. *ACM SIGIR Workshop on Recommender Systems.*

Derbeko, P., El-Yaniv, R., & Meir, R. (2002). Variance optimized bagging.

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B. M., Herlocker, J. L., & Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. *AAAI/IAAI* (pp. 439–446).

Harrington, E., Herbrich, R., Kivinen, J., Platt, J. C., & Williamson, R. C. (2003). Online bayes point machines. *Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 241–252).

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, *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.

Marlin, B., & Zemel, R. S. (2004). The multiple multiplicative factor model for collaborative filtering. *ICML.*

Melville, P., Mooney, R., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendation. *AAAI.*

Platt, J. C. (1999). Probabilities for support vector machines. In B. S. D. S. A. Smola (Ed.), *Advances in large margin classifiers*, 61–74. MIT Press.

Rennie, J. D. M. (2006). Personal Communication.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin factorization for collaborative prediction. *ICML.*

Srebro, N., Rennie, J. D. M., & Jaakola, T. S. (2005). Maximum-margin matrix factorization. *NIPS.*

Valentini, G., & Dietterich, T. G. (2004). Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, *5*, 725–775.