Generalizing Matrix Factorization Through Flexible Regression Priors

Liang Zhang Deepak Agarwal Bee-Chung Chen
Yahoo! Labs
4401 Great America Pkwy, Santa Clara, CA
{liangzha,dagarwal,beechun}@yahoo-inc.com

ABSTRACT

Predicting user "ratings" on items is a crucial task in recommender systems. Matrix factorization methods that computes a low-rank approximation of the incomplete user-item rating matrix provide state-of-the-art performance, especially for users and items with several past ratings (warm starts). However, it is a challenge to generalize such methods to users and items with few or no past ratings (cold starts). Prior work [4][32] generalized matrix factorization to include both user and item features for performing better regularization of factors as well as provide a model for smooth transition from cold starts to warm starts. However, the features were incorporated via linear regression on factor estimates. In this paper, we generalize this process to allow for arbitrary regression models like decision trees, boosting, LASSO, etc. The key advantage of our approach is the ease of computing — any new regression procedure can be incorporated by "plugging" in a standard regression routine into a few intermediate steps of our model fitting procedure. With this flexibility, one can leverage a large body of work on regression modeling, variable selection, and model interpretation. We demonstrate the usefulness of this generalization using the MovieLens and Yahoo! Buzz datasets.

Categories and Subject Descriptors

H.1.1 [**Information Systems**]: Models and Principles;
G.3 [**Mathematics of Computing**]: Probability and Statistics

General Terms

Algorithms, Theory, Experimentation

Keywords

Latent Factor, Recommender Systems, Matrix Factorization, Tree Models, Regression Priors

1. INTRODUCTION

Accurately predicting unknown user *ratings* on items is important for success in recommender applications like movie recommendation [1], content optimization [6], and computational advertising [13]. Ratings can be explicit like movie ratings or implicit like clicks. Training data typically consists of ratings y_{ij} given by user i to item j. In most recommender problems, ratings are available for only a small fraction of all possible user-item pairs; the goal is to accurately predict out-of-sample ratings. Usually, a fraction of predictions may involve a user-item pair where at least one entity is new in the training data, a scenario referred to as cold-start. Examples include recommendation of ephemeral items like news stories, tweets and updates, and scenarios where new users routinely use the system. We shall refer to a user-item pair where both entities are observed in the training data as warm-start.

Recent advances illustrate that methods based on low-rank matrix factorization provide state-of-the-art methods for out-of-sample predictions in warm-start scenarios [8, 1, 34, 27, 26]. In this approach, rating y_{ij} is approximated or predicted by $\boldsymbol{u}_i'\boldsymbol{v}_j$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are usually called the factor vectors of user i and item j, respectively. In the training phrase, $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are learned from past rating data. To avoid overfitting, it is common to penalize large $||\boldsymbol{u}_i||^2$ and $||\boldsymbol{v}_j||^2$. In other word, this approach "shrinks" $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ toward zero. Although matrix factorization provides excellent predictive performance in warm-start scenarios, it fails in cold-start scenarios — note that $\boldsymbol{u}_i = 0$ (or some default value) if user i is not in the training data.

To provide good predictive performance for both cold-start and warm-start, several recent papers have generalized matrix factorization to incorporate features [4, 32]. The key idea is that, instead of shrinking $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ toward zero, shrink them toward weighted sums of user i's features and item j's features, respectively, where the weights of such linear models are learned from data. We shall refer to this model as Regression-based Latent Factor Model (RLFM). Other related works are discussed in Section 4.

Our **contributions** are as follows — We propose a novel modeling framework called *Generalized Matrix Factorization* (GMF) to seamlessly handle both cold-start and warm-start scenarios. GMF generalizes RLFM by allowing latent factors $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ to "shrink" toward values predicted by any feature-based regression for Gaussian response. From a theoretical perspective, it provides a flexible class of models to capture user-item interactions in large scale recommender problems. In fact, it provides an attractive strategy to create non-linear functions of arbitrary complexity through a set of Gaussian regressions with univarate response. GMF provides a powerful framework that can leverage a large body of work on regression modeling like decision trees [11], random forests [12], gradient boosted trees [17], LASSO [33], and so on, to capture complex user-item interactions. While such generalization also presents formidable computational challenges, we show that our model fitting algorithm based on a Monte Carlo EM is efficient

RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.

and provides an attractive modularity that allows easy integration of any off-the-shelf regression procedure in the M-step. Other than generality, the ability to use any regression procedure automatically allows us to use modern day regression tools like LASSO and gradient boosted trees, where both variable selection and predictive accuracy are achieved simultaneously. We illustrate our method on MovieLens dataset and Yahoo! Buzz datasets. Our experiments suggests that significant improvement in accuracy is possible through more flexible non-linear regression functions.

2. MODEL

Our GMF model extends matrix factorization models for recommendation and multi-task learning problems by simultaneously leveraging both features and ratings. The main idea is that instead of "shrinking" the factors toward zero (when the data is sparse), one can predict those factors using a regression model that generalizes individual factors through available features.

Notation: We denote by (i, j) the pair corresponding to user i and item j in the recommendation setting, or case i and task j in the multi-task learning setting. Let y_{ij} denote the response associated with the $(i, j)^{\text{th}}$ pair. For ease of exposition, we will describe our GMF model for the recommendation problem but we note that modeling details for multi-task learning (where each case usually participates in multiple learning tasks) follows from a similar derivation. In the recommendation setting, y_{ij} denotes the "rating" of user i for item j. The GMF model is applicable to any response that can be expressed as a generalized linear model conditional on known mean for the response variable [23]. Since we always use i to denote a user and j to denote an item, by slight abuse of notations we let $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and $\boldsymbol{x}_{ij}$ denote feature vectors for user i, item j and pair (i, j), respectively. For instance, $\boldsymbol{x}_i$ may include user demographics, geo-location, browse behavior, etc. The vector $\boldsymbol{x}_j$ denotes the set of item features and may include content category, tokens extracted from the title, body when items have textual description, etc. Finally, $\boldsymbol{x}_{ij}$ is a vector of features that are not entirely attributable to either user or item; examples include position on a page where an item is displayed, time-of-day when an item is displayed, etc.

Observation model: Without loss of generality, we describe our GMF model in a Bayesian framework. Our goal is to model the *unobserved true score* s_{ij} that user i would give item j based on the observed rating y_{ij} (which may be noisy) and features. In particular, for numeric ratings, it is common to assume:

$$y_{ij} \sim N(s_{ij}, \sigma^2).$$

For binary ratings (e.g., click or no-click), it is common to assume:

$$y_{ij} \sim \text{Bernoulli}(p_{ij}) \text{ and let } s_{ij} = \log \tfrac{p_{ij}}{1-p_{ij}}.$$

For counts, it is common to assume:

$$y_{ij} \sim \text{Poisson}(n_{ij}\, p_{ij}) \text{ and let } s_{ij} = \log p_{ij},$$

where n_{ij} is a normalization constant associated with y_{ij}.

Factorization: We model the score s_{ij} as

$$s_{ij} = f(x_{ij}) + \alpha_i + \beta_j + \boldsymbol{u}_i'\boldsymbol{v}_j,$$

where $f(x_{ij})$ is a regression function based on feature vector x_{ij}; α_i and β_j are the user and item bias terms; $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are r-dimensional *user* and *item* factors respectively; and the inner product $\boldsymbol{u}_i'\boldsymbol{v}_j$ captures the interaction. We provide more discussions about regression functions later. We call r the factor dimensionality, which is pre-specified. One can interpret user factors $\boldsymbol{u}_i = (u_{i1}, ..., u_{ir})$ as user i's affinity to r different latent "topics", and item factors $\boldsymbol{v}_j = (v_{j1}, ..., v_{jr})$ as item j's affinity to those "topics", although the latent "topics" are not readily interpretable.

In addition to the parameters in f, the number of additional parameters that needs to be estimated from the data equals $(r + 1)(M + N)$, where M is the number of users and N is the number of items. In practice, $(r + 1)(M + N)$ may even be larger than the number of observed ratings; this can lead to severe *overfitting*. It is a common practice to shrink the factors toward zero, i.e., to put zero-mean Gaussian priors on the latent factors. However, shrinking toward zero is restrictive and may not generalize well, especially in cold-start scenarios.

Flexible priors: The key idea of this paper is based on the observation that instead of shrinking the factors toward zero, one can add more flexibity by shrinking factors to different means based on feature information. However, both the regressions and factor estimation have to be done simultaneously; this presents formidable computational challenges that is addressed in Section 3. Specifically, we put the following priors on α_i, β_j, $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$.

$$\alpha_i \sim N(g(x_i), \sigma_\alpha^2), \quad \boldsymbol{u}_i \sim N(G(x_i), \sigma_u^2 I),$$
$$\beta_j \sim N(h(x_j), \sigma_\beta^2), \quad \boldsymbol{v}_j \sim N(H(x_j), \sigma_v^2 I),$$

where g and h are any choices of regression functions that return scalars, and G and H are regression functions that return r-dimensional vectors.

Regression functions: We note that setting f, g, h, G, H to linear regression functions have been studied in [4, 32]. The main contribution of this paper is generalizing this approach to any choice of regression functions so that the large body of work on nonlinear regression models can be easily leveraged to provide better accuracy. For example, one can use a decision tree for f, a nearest-neighbor model for g, a random forest for h, a sparse LASSO regression model for G, and a gradient-boosted tree ensemble for H. Although the extension from linear models to nonlinear models is conceptually simple, the added flexibility can significantly improve accuracy.

We note that G and H are functions that return vectors. One can either use multivariate regression models to predict all the values in a vector jointly, or use regular univariate regression models to predict each value in the vector separately. The former may leverage the correlation between the components in the vector and potentially provide better accuracy, while the latter enjoys simplicity. Specifically, for the latter, we define $G(x_i) = (G_1(x_i), ..., G_r(x_i))$ and each $G_k(x_i)$ is an independent univariate regression function.

Remarks: To get intuition into the class of predictor functions induced by GMF, we look at the marginal distribution of y_{ij} for Gaussian response. It is easy to show that $E(y_{ij}|Data, \Theta) =$

$$f(\boldsymbol{x}_{ij}) + g(\boldsymbol{x}_i) + h(\boldsymbol{x}_j) + \textstyle\sum_{k=1}^{r} G_k(\boldsymbol{x}_i) H_k(\boldsymbol{x}_j)$$

Thus, with state-of-the-art regression functions, the function class induced to predict ratings is rich; e.g., if each G and H are decision trees, the predictive function is now a cross-product of trees.

Several modern day regression procedures like LASSO, Random forests, Gradient boosted decision trees are successful in large scale prediction tasks in the presence of several noisy predictors due to an inbuilt variable selection capability. Using such models in GMF automatically provide a mechanism to perform variable selection for both users and items. Also, using interpretable models like decision trees allows us to provide an interpretation of user and item factors and could be useful in some recommendation scenarios.

3. FITTING PROCEDURE

In this section, we provide a detailed description of our model fitting procedure that is based on a Monte Carlo Expectation Maximization (MCEM) algorithm. For ease of exposition, we first focus on the Gaussian model and discuss model fitting for other members in GLM family (e.g. Logistic, Poisson) in Section 3.3.

3.1 Optimization Problem

Let $\mathbf{\Theta} = (f, g, h, G, H, \sigma_\alpha^2, \sigma_u^2, \sigma_\beta^2, \sigma_v^2)$ be the set of prior parameters (hyper-parameters). Let $\mathbf{\Delta} = \{\alpha_i, \beta_j, \boldsymbol{u}_i, \boldsymbol{v}_j\}_{\forall i,j}$ be the set of factors (random-effects). Let $\boldsymbol{y}$ denote the set of observed ratings. The *complete data log-likelihood* is given by

$$\begin{aligned}
\log L(\mathbf{\Theta}; \mathbf{\Delta}, \boldsymbol{y}) &= \log \Pr[\boldsymbol{y}, \mathbf{\Delta} | \mathbf{\Theta}] = \text{ constant} \\
&- \tfrac{1}{2} \textstyle\sum_{ij} \left(\frac{1}{\sigma^2} (y_{ij} - f(x_{ij}) - \alpha_i - \beta_j - \boldsymbol{u}_i' \boldsymbol{v}_j)^2 + \log \sigma^2 \right) \\
&- \tfrac{1}{2\sigma_\alpha^2} \textstyle\sum_i (\alpha_i - g(x_i))^2 - \frac{M}{2} \log \sigma_\alpha^2 \\
&- \tfrac{1}{2\sigma_\beta^2} \textstyle\sum_j (\beta_j - h(x_j))^2 - \frac{N}{2} \log \sigma_\beta^2 \\
&- \tfrac{1}{2\sigma_u^2} \textstyle\sum_i ||\boldsymbol{u}_i - G(x_i)||^2 - \frac{Mr}{2} \log \sigma_u^2 \\
&- \tfrac{1}{2\sigma_v^2} \textstyle\sum_j \|\boldsymbol{v}_j - H(x_j)\|^2 - \frac{Nr}{2} \log \sigma_v^2,
\end{aligned}$$

where the second line specifies the prediction error in terms of the sum of squared differences, and the following four lines specify L_2 regularization shrinking the corresponding factors toward values predicted by regression functions. Note that the $\log \sigma_\star^2$ terms result from the assumption of normal distributions, and σ^2/σ_α^2, σ^2/σ_β^2, σ^2/σ_u^2 and σ^2/σ_v^2 can be interpreted as the strength of the corresponding regularization terms with higher values indicating more regularization.

Since our goal is to generalize factorization through features, we want to marginalize the factors $\mathbf{\Delta}$ and obtain the maximum likelihood estimates of the parameters $\mathbf{\Theta}$ (which includes the regression functions and regularization weights). That is, we want to find the value of $\mathbf{\Theta}$ that maximizes the marginal likelihood

$$\Pr[\boldsymbol{y}|\mathbf{\Theta}] = \textstyle\int L(\mathbf{\Theta}; \mathbf{\Delta}, \boldsymbol{y})\, d\, \mathbf{\Delta}.$$

It is important to note that this Bayesian formulation differs from the regular optimization formulation in the following ways. First, marginalizing over the factors usually provides better generalization. Second, except for the factor dimensionality, no tuning is needed in our formulation since the regularization weights (i.e., the prior variances) are obtained automatically from our fitting procedure; in a typical optimization formulation of this problem, the regularization weights are often tuned through a separate tuning dataset. This is difficult because the optimal values of the regularization parameters are functions of the rank parameter r. Indeed, our empirical observation clearly demonstrates that larger values of r automatically induces more regularization and supports the theoretical evidence which suggests using a large rank but performing more regularization.

3.2 EM Algorithm

The EM algorithm [16] is well suited to fit factor models. The factors in this case form the missing data that are augmented to the observed data. The EM algorithm iterates between an E-step and an M-step until convergence. Let $\hat{\mathbf{\Theta}}^{(t)}$ denote the current estimated value of $\mathbf{\Theta}$ at the beginning of the tth iteration.

- **E-step:** We take expectation of complete data log likelihood with respect to the posterior of missing data $\mathbf{\Delta}$ conditional on observed data $\boldsymbol{y}$ and the current estimate of $\mathbf{\Theta}$; i.e., compute

$$q_t(\mathbf{\Theta}) = E_{\mathbf{\Delta}}[\log L(\mathbf{\Theta}; \mathbf{\Delta}, \boldsymbol{y}) \,|\, \hat{\mathbf{\Theta}}^{(t)}]$$

  as a function of $\mathbf{\Theta}$, where the expectation is taken over the posterior distribution of $(\mathbf{\Delta} \,|\, \hat{\mathbf{\Theta}}^{(t)}, \boldsymbol{y})$.

- **M-step:** We maximize the expected complete data log likelihood from the E-step to obtain updated values of $\mathbf{\Theta}$; i.e., find

$$\hat{\mathbf{\Theta}}^{(t+1)} = \arg\max_{\mathbf{\Theta}} \; q_t(\mathbf{\Theta}).$$

Note that the actual computation in the E-step is to generate sufficient statistics for computing $\arg\max_{\mathbf{\Theta}} \; q_t(\mathbf{\Theta})$, so we do not need to scan the raw data every time we need to evaluate $q_t(\mathbf{\Theta})$. At each iteration, the EM algorithm is guaranteed not to deteriorate the value of $\int L(\mathbf{\Theta}; \mathbf{\Delta}, \boldsymbol{y})\, d\, \mathbf{\Delta}$.

The major computational bottleneck is in the E-step because the posterior of factors are not available in closed form. Hence, we follow the Monte Carlo EM (MCEM) algorithm [9] by drawing samples from this posterior and approximate the expectation in E-step by taking Monte Carlo mean. Alternatively, one can apply variational approximation to derive a closed-form formula for the expectation, or apply the iterative conditional mode (ICM) algorithm in which the expectation computation is replaced by "plugging in" the mode of the conditional distributions. However, in our experience and other studies [27], sampling usually provides better performance in terms of predictive accuracy, while still being scalable. Sampling in our experience is also resistant to over-fitting even with increasing number of factors. Thus, we focus on the MCEM algorithm.

3.2.1 Monte-Carlo E-Step

Since $E_{\mathbf{\Delta}}[\log L(\mathbf{\Theta}; \mathbf{\Delta}, \boldsymbol{y}) \,|\, \hat{\mathbf{\Theta}}^{(t)}]$ is not available in closed form, we compute the Monte-Carlo expectation based on L samples generated by a Gibbs sampler [18]. The Gibbs sampler repeats the following procedure L times. In the following, we use $(\delta \,|\, \text{Rest})$, where δ can be one of α_i, β_j, $\boldsymbol{u}_i$, and $\boldsymbol{v}_j$, to denote the conditional distribution of δ given all the other parameters. Let $\mathcal{I}_j$ denote the set of users who rated item j, and $\mathcal{J}_i$ denote the set of items rated by user i.

1. For each user i, sample α_i from $(\alpha_i \,|\, \text{Rest})$, which is Gaussian, for all i.

$$\begin{aligned}
\text{Let } o_{ij} &= y_{ij} - f(x_{ij}) - \beta_j - \boldsymbol{u}_i' \boldsymbol{v}_j \\
\mathit{Var}[\alpha_i|\text{Rest}] &= (\tfrac{1}{\sigma_\alpha^2} + \textstyle\sum_{j \in \mathcal{J}_i} \frac{1}{\sigma^2})^{-1} \\
E[\alpha_i|\text{Rest}] &= \mathit{Var}[\alpha_i|\text{Rest}](\tfrac{g(x_i)}{\sigma_\alpha^2} + \textstyle\sum_{j \in \mathcal{J}_i} \frac{o_{ij}}{\sigma^2})
\end{aligned}$$

2. For each item j, sample β_j from $(\beta_j \,|\, \text{Rest})$, which is Gaussian, for all j.

$$\begin{aligned}
\text{Let } o_{ij} &= y_{ij} - f(x_{ij}) - \alpha_i - \boldsymbol{u}_i' \boldsymbol{v}_j \\
\mathit{Var}[\beta_j|\text{Rest}] &= (\tfrac{1}{\sigma_\beta^2} + \textstyle\sum_{i \in \mathcal{I}_j} \frac{1}{\sigma^2})^{-1} \\
E[\beta_j|\text{Rest}] &= \mathit{Var}[\beta_j|\text{Rest}](\tfrac{h(x_j)}{\sigma_\beta^2} + \textstyle\sum_{i \in \mathcal{I}_j} \frac{o_{ij}}{\sigma^2})
\end{aligned}$$

3. For each user i, sample $\boldsymbol{u}_i$ from $(\boldsymbol{u}_i \,|\, \text{Rest})$, which is Gaussian, for all i.

$$\begin{aligned}
\text{Let } o_{ij} &= y_{ij} - f(x_{ij}) - \alpha_i - \beta_j \\
\mathit{Var}[\boldsymbol{u}_i|\text{Rest}] &= (\tfrac{1}{\sigma_u^2} I + \textstyle\sum_{j \in \mathcal{J}_i} \frac{\boldsymbol{v}_j \boldsymbol{v}_j'}{\sigma^2})^{-1} \\
E[\boldsymbol{u}_i|\text{Rest}] &= \mathit{Var}[\boldsymbol{u}_i|\text{Rest}](\tfrac{1}{\sigma_u^2} G(x_i) + \textstyle\sum_{j \in \mathcal{J}_i} \frac{o_{ij} \boldsymbol{v}_j}{\sigma^2})
\end{aligned}$$

4. For each item j, sample $\boldsymbol{v}_j$ from $(\boldsymbol{v}_j \,|\, \text{Rest})$, which is Gaussian, for all j.

$$\begin{aligned}
\text{Let } o_{ij} &= y_{ij} - f(x_{ij}) - \alpha_i - \beta_j \\
\mathit{Var}[\boldsymbol{v}_j|\text{Rest}] &= (\tfrac{1}{\sigma_v^2} I + \textstyle\sum_{i \in \mathcal{I}_j} \frac{\boldsymbol{u}_i \boldsymbol{u}_i'}{\sigma^2})^{-1} \\
E[\boldsymbol{v}_j|\text{Rest}] &= \mathit{Var}[\boldsymbol{v}_j|\text{Rest}](\tfrac{1}{\sigma_v^2} H(x_j) + \textstyle\sum_{i \in \mathcal{I}_j} \frac{o_{ij} \boldsymbol{u}_i}{\sigma^2})
\end{aligned}$$

Let $\tilde{E}[\cdot]$ and $\tilde{Var}[\cdot]$ denote the Monte Carlo mean and variance computed using the L Gibbs samples. The output of the E-step consists of $\tilde{E}[\alpha_i]$, $\tilde{E}[\beta_j]$, $\tilde{E}[\boldsymbol{u}_i]$, $\tilde{E}[\boldsymbol{v}_j]$, for all i and j, $\tilde{E}[\boldsymbol{u}_i'\boldsymbol{v}_j]$, for all observed (i,j) pairs, $\sum_{ij}\tilde{Var}[s_{ij}]$, $\sum_{ik}\tilde{Var}[u_{ik}]$, where u_{ik} is the kth element in $\boldsymbol{u}_i$, and $\sum_{jk}\tilde{Var}[v_{jk}]$, where v_{jk} is the kth element in $\boldsymbol{v}_j$. These are the sufficient statistics that will be used in the M-step of our fitting procedure.

3.2.2 M-Step

In the M-step, we find the parameter setting $\boldsymbol{\Theta}$ that maximizes the expectation computed in the E-step

$$q_t(\boldsymbol{\Theta}) = E_{\boldsymbol{\Delta}}[\log L(\boldsymbol{\Theta}; \boldsymbol{\Delta}, \boldsymbol{y}) \mid \hat{\boldsymbol{\Theta}}^{(t)}].$$

It can be easily seen that (f, σ^2), (g, σ_α^2), (h, σ_β^2), (G, σ_u^2), and (H, σ_v^2) can be optimized by separate regressions.

Regression for (f, σ^2): Here, we want to minimize

$$\tfrac{1}{\sigma^2}\sum_{ij}\tilde{E}[(y_{ij} - f(x_{ij}) - \alpha_i - \beta_j - \boldsymbol{u}_i'\boldsymbol{v}_j)^2] + D\log(\sigma^2),$$

where D is the number of observed ratings. It is easy to see that the optimal solution to f can be found by solving a regression problem that uses x_{ij} as features to predict $(y_{ij} - \tilde{E}[\alpha_i] - \tilde{E}[\beta_j] - \tilde{E}[\boldsymbol{u}_i'\boldsymbol{v}_j])$ as the target. Let RSS denote the residual sum of squares from this regression. Then, the optimal σ^2 is $(\sum_{ij}\tilde{Var}[s_{ij}] + \text{RSS})/D$, where RSS is the residual sum of squares of the regression. Note that f can be any regression model.

Regression for (g, σ_α^2): Similar to the previous case, the optimal g can be found by solving a regression problem that uses x_i as features to predict $\tilde{E}[\alpha_i]$, and the optimal σ_α^2 is $(\sum_i \tilde{Var}[\alpha_i] + \text{RSS})/M$, where M is the number of users.

Regression for (h, σ_β^2): The optimal h can be found by solving a regression problem using x_j as features to predict $\tilde{E}[\beta_j]$, and the optimal $\sigma_\beta^2 = (\sum_j \tilde{Var}[\beta_j] + \text{RSS})/N$, where N is the number of items.

Regression for (G, σ_u^2): For multivariate regression models, we find G by solving a regression problem using x_i as features to predict multivariate response $\tilde{E}[\boldsymbol{u}_i]$. For univariate regression models, we consider $G(x_i) = (G_1(x_i), ..., G_r(x_i))$, where each $G_k(x_i)$ returns a scalar. In this case, for each k, we find G_k by solving a regression problem that uses x_i as features to predict $\tilde{E}[u_{ik}]$. Let RSS denote the total residual sum of squares. Then, $\sigma_u^2 = (\sum_{ik}\tilde{Var}[u_{ik}] + \text{RSS})/(rM)$.

Regression for (H, σ_v^2): For multivariate regression models, we find H by solving a regression problem using x_j as features to predict multivariate response $\tilde{E}[\boldsymbol{v}_j]$. For univariate regression models, we consider $H(x_j) = (H_1(x_j), ..., H_r(x_j))$, where each $H_k(x_j)$ returns a scalar. In this case, for each k, we find H_k by solving a regression problem that uses x_j as features to predict $\tilde{E}[v_{jk}]$. Let RSS denote the total residual sum of squares. Then, $\sigma_v^2 = (\sum_{jk}\tilde{Var}[v_{jk}] + \text{RSS})/(rN)$.

3.3 Remarks

Number of Gibbs samples: Replacing the precise E-step with a Monte Carlo average no longer guarantees an increase in the marginal likelihood at each step. If the Monte Carlo sampling error associated with $\hat{\boldsymbol{\Theta}}^{(t)}$ (an estimate of $\hat{\boldsymbol{\Theta}}_\infty^{(t)}$ that is obtained from an exact E-step, e.g., by using infinite number of samples) is large relative to $\|\hat{\boldsymbol{\Theta}}^{(t-1)} - \hat{\boldsymbol{\Theta}}_\infty^{(t)}\|$, then the Monte Carlo E-step is wasteful since it is swamped by the Monte Carlo error. There are no rigorous solutions to this problem in the literature (especially when samples are dependent) except for some practical guidelines [9]. For instance, it is better to use fewer Monte Carlo simulations during early iterations. We performed extensive experiments with various schemes and found 20 EM iterations with 100 samples (drawn after 10 burn-in samples) at each iteration performed adequately in our experiments. In fact, the performance was not too sensitive to the choice of number of samples, even small number of samples like 50 did not hurt performance by much. We also note that Gibbs sampler was chosen due to its simplicity; better sampling methods to make the sampler *mix* faster is an open research problem in the context of bilinear factor models.

Scalability: Fixing the factor dimensionality, the number of EM iterations and the number of Gibbs samples per iteration, the MCEM algorithm is essentially *linear* in the number of observations. In our experience, we observe that the MCEM algorithm converges quickly after a fairly small number of EM iterations (usually around 10). We also note that the algorithm is highly parallelizable. In the E-step, when drawing the ℓth Gibbs sample, the factors for each user can be drawn independently of other users. Thus, this sampling step can be done in parallel. This is the same for items. The M-step requires solving several regression problems. Any scalable software package can be used here.

Fitting non-Gaussian responses: When y_{ij} can not be fitted by a Gaussian model (e.g. binary responses or counts), we follow [20] and [30] to construct a pseudo Gaussian response variable. Let $E[y_{ij}] = \mu_{ij}$, and define the link function $l(\mu_{ij}) = s_{ij}$. For Logistic regression, $l(\mu_{ij}) = \log\frac{\mu_{ij}}{1-\mu_{ij}}$. For Poisson regression, $l(\mu_{ij}) = \log\frac{\mu_{ij}}{n_{ij}}$. In the E-step at the t-th iteration, denote the current estimate of s_{ij} and μ_{ij} as $\hat{s}_{ij}^{(t)}$ and $\hat{\mu}_{ij}^{(t)}$ respectively. The pseudo response variable

$$z_{ij} = \hat{s}_{ij}^{(t)} + (y_{ij} - \hat{\mu}_{ij}^{(t)})(l'(\mu_{ij})|_{\hat{\mu}_{ij}^{(t)}}), \tag{1}$$

and approximately,

$$z_{ij} \sim N(s_{ij}, Var[y_{ij}|\hat{s}_{ij}^{(t)}](l'(\mu_{ij})|_{\hat{\mu}_{ij}^{(t)}})^2). \tag{2}$$

Hence the parameters in $\boldsymbol{\Delta}$ can directly be sampled following the MCEM procedure. For logistic regression, this can also be done through a variational approximation that involves a weighted Gaussian regression after each EM iteration (see [4] for details).

Closer look at the shrinkage estimator for factors: In the following, we take a closer look at how GMF is estimating the factors as a linear combination of regression function and collaborative filtering. Without loss of generality, let us consider the factor estimate $\boldsymbol{u}_i$. For simplicity of exposition, consider the Gaussian case and assume $r = 1$; also let o_{ij} to be the rating of user i on item j after adjusting for the features $\boldsymbol{x}_{ij}$ and user and item bias. Then, letting $\lambda = \frac{\sigma^2}{\sigma_u^2}$, we see

$$E[u_i|\{\boldsymbol{v}, Data, \Theta\}] = \tfrac{\lambda}{\lambda + \sum_{j\in\mathcal{J}_i} v_j^2}G(\boldsymbol{x}_i) + \tfrac{\sum_{j\in\mathcal{J}_i} v_j o_{ij}}{\lambda + \sum_{j\in\mathcal{J}_i} v_j^2},$$

which for fixed $\boldsymbol{v}$ is clearly a linear combination of regression G and ratings o_{ij} of user i. Note that the weights attached to different components in the linear combination depend on both the global shrinkage parameter λ and the item factors that are rated by the user. We also note that the contribution from regression is negligible when $\sum_{j\in\mathcal{J}_i} v_j^2$ is significantly larger than λ, in this scenario the user factor estimate is obtained by performing a linear regression on items rated by the users with the item factors as covariates. This clearly shows that when a user rates a large number of items for which we have enough data to obtain reliable item factor es-

timates, rating information takes over and regression is no longer important.

Assuming the hyper-parameters are known, remarkably the marginal expectation of u_i conditional on data is still a linear combination of regression and ratings by user i with the weights being given by $E(\frac{\lambda}{\lambda+\sum_{j\in\mathcal{J}_i} v_j^2})$ and $E(\frac{v_j}{\lambda+\sum_{j\in\mathcal{J}_i} v_j^2})$, $j \in \mathcal{J}_i$, where the expectations are w.r.t. the marginal posterior of item factors v_j's. This provides insights on how GMF model is estimating the factors by achieving a compromise between regression and ratings. Interestingly, we note that although the shrinkage estimator is a linear combination of ratings and regression, the weights are highly non-linear functions and depend on both global shrinkage parameters along with local rating information.

4. RELATED WORK

Our work extends the rich literature on factorization models that provide the state-of-the-art performance for recommender problems. A theoretical perspective of this problem was first provided in [31] who formulated it as a maximum margin matrix factorization problem that minimizes a user-specified convex loss function for the observed entries subject to constraints on the nuclear norm of the underlying complete matrix. Although the optimization problem was convex, it required an expensive semi-definite program and was not scalable to large scale problems. To achieve scalability, [26] replaced the trace-norm convex objective by an upper bound that directly optimized the user and item factors. Although non-convex, they showed such an objective could be optimized in a highly scalable way and achieved good local optima through gradient methods. Since then, several authors have successfully used this strategy to achieve good predictive performance in collaborative filtering applications [8, 1]. In [27, 28], the above non-convex objective was formulated in a probabilistic framework using a hierarchical random-effects model where the user and item factors were multivariate random-effects (factor vectors) that were regularized through zero-mean multivariate Gaussian priors. The model in [22] further generalizes the objective by assuming user/item factors are obtained from a reproducing kernel Hilbert space; as usual the computations are carried out using the "kernel" trick.

None of the matrix factorization methods described above addressed the problem of incorporating user/item features in the model. A desirable approach would be to have a model that provides predictive accuracy as matrix factorization for warm-start scenarios but fallbacks on a feature-based regression model in cold-start scenarios. In [2, 3], expected ratings of user i and item j was modeled as kronecker product of kernels defined on the user and item space. To ensure the prediction uses both the collaborative filtering and feature information, the authors suggested to work with user/item kernels that are convex combination of a kernel based on only features and a dirac kernel that is turned "on" only when there is a match between users/items. However, the weights in this convex combinations are global and not user/item specific; this is in sharp contrast to our generalized factorization model which provides user/item factor estimates that are non-linear functions of both features and ratings with the weights being user/item specific. The ease of adding any regression model to the priors of the factors allows us to work with flexible kernels based on features, and the shrinkage estimators provided by our hierarchical model achieve the right balance between collaborative filtering and feature information in estimating factors. Moreover, by leveraging recent research in regression-based methods that perform both variable selection and estimation in applications with large number of predictors we are able to perform feature selection with relative ease. For instance, Section 5 illustrates automatic feature selection in our framework through LASSO and various other methods based on decision trees.

The hierarchical model presented in this paper is an extension of recent work [4, 32] that assumes linear functions of features for estimating the priors of factors. In [5], the authors also explored a non-linear approach that assumes item factors are discrete and obtained through an LDA prior. However, this approach is not general and only works well for data where features have a bag-of-words representation.

Our approach of course is related to the large body of literature on non-parametric function estimation for supervised learning problems [21]. In fact, we use such function estimation techniques as an input to perform the overall prediction task that combines both information from features and collaborative filtering. We also note that the number of functions that we estimate equals $2(r+1)$ (one for each user factors, one for the user bias, and same for items).

Finally, the central idea of this paper to incorporate both warm-start and cold-start scenarios simultaneously in collaborative filtering is a well studied problem with a rich literature. Several methods that combine content and collaborative filtering have been studied. For instance, [7] presents a recommender system that computes user similarities based on content-based profiles. In [15], collaborative filtering and content-based filtering are combined linearly with weights adjusted based on absolute errors of the models. In [24], content based models are used to fill up the rating matrix followed by recommendation based on similarity (memory) based methods [10]. [19] and [25] used *filterbots* to improve cold-start recommendations. A filterbot is an automated agent that rates items algorithmically; these are treated as additional users in a collaborative filtering system. [29] extends the aspect model to combine the item content with user ratings under a single probabilistic framework. However, recent work have clearly illustrated that matrix factorization where factors are regularized through features are superior than the classical methods of dealing with warm-start and cold-start scenarios simultaneously.

5. EXPERIMENTS

We show the effectiveness of the generalized matrix factorization (GMF) model using simulation and two real datasets, the MovieLens dataset (available at www.grouplens.org) and the Yahoo! Buzz dataset (introduced in [5]). The main advantage of GMF is the ability of plugging in arbitrary regression models to predict the prior means of the factors (i.e., $f(x_{ij})$, $g(x_i)$, $h(x_j)$, $G(x_i)$ and $H(x_j)$). In this empirical study, we compare the predictive performance of GMF using six different classes of regression models.

RLFM is the model proposed in [4], which is a special case of GMF using linear regression models. **GMF-LASSO** is the GMF model based on LASSO [33], linear regression models with L_1 regularization. **GMF-RP** is the GMF model using recursive partitioning regression trees [11] as the regression models. A single regression tree is built for each factor by recursively selecting partitions that maximize the decrease of lack-of-fit, and pruning the tree using cross-validation at the end. **GMF-RF** is the GMF model using random forests [12] as the regression models. **GMF-GB** is the GMF model using gradient boosting machines [17] as the regression models. **GMF-BART** is the GMF model using Bayesian additive regression trees [14] as the regression models. BART is a Bayesian approach of building the "sum-of-tree" model by treating each tree component as random and imposing regularization priors on the trees to keep the individual tree effects small. [14] show empirically BART can outperform other competitive tree models such

as random forests and gradient boosting. All the above regression models fitting are implemented via the statistical software **R** where automatic parameter tuning using cross-validation on training data are built in (e.g. the regularization parameter for the lasso penalty).

Through extensive experiments, we show that GMF models using nonlinear regression models provide statistically significant accuracy improvement over RLFM, which is the state-of-the-art method, on all the datasets. These results confirm the practical importance of having flexible regression priors in GMF, although the extension from RLFM to GMF is conceptually simple. We note that, in this empirical study, we select six off-the-shelf regression models based on software package availability; by simply plugging in more advanced regression models, further accuracy improvement may be achieved. In fact, because GMF allows easy plug-and-play of arbitrary regression models, any advances in regression modeling can be easily leveraged to improve factorization-based recommender systems. In addition to accuracy improvement, we also study the feature selection capability of GMF using LASSO and tree models. The empirical results are encouraging.

5.1 Simulation Results

We first compare GMF models using different classes of regression functions on 10 simulated datasets, and show that when there are nonlinearity or noisy features in data, GMF models using nonlinear models or models that perform feature selection significantly outperform RLFM (i.e., the GMF model using linear models). For each of the 10 datasets, we simulated 100,000 Gaussian responses as the ratings for the training set and 100,000 for the test set. Each training set has 1,000 users and 1,000 items, and each test set contains all of the old users and items plus 500 new users and 500 new items. Each user or item has 200 features drawn from $N(0,1)$, where only 10 of them are used in the regression functions; others are just used to create noise in data, in order to test variable selection capability. We set the number of factors to be 10. To generate nonlinearity in the feature-based prior means of the factors, we followed [17] to construct 20 random Gaussian function for each factor. For instance, α_i is simulated from $N(\hat{\alpha}_i, 1)$ where $\hat{\alpha}_i$ is obtained from a random function $F_\alpha(\boldsymbol{x}_i)$ of the user features $\boldsymbol{x}_i$:

$$F_\alpha(\boldsymbol{x}_i) = \sum_{\ell=1}^{20} w_\ell \, b_\ell(A_\ell \boldsymbol{x}_i), \tag{3}$$

where $w_\ell \sim N(0,1)$ is a random weight drawn once per dataset; A_ℓ is a diagonal 0/1 matrix that randomly select n_ℓ features from the pre-specified 10 useful features (note that n_ℓ is drawn from the exponential distribution with mean 3.5); and $b_\ell(\cdot)$ is a random Gaussian function

$$b_\ell(\boldsymbol{z}) = exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu}_\ell)'\boldsymbol{V}_\ell(\boldsymbol{z}-\boldsymbol{\mu}_\ell)\right), \tag{4}$$

where $\boldsymbol{\mu}_\ell$ is drawn from $N(0, \boldsymbol{I})$ and $\boldsymbol{V}_\ell$ is a random positive definite matrix with eigenvalues drawn from *Uniform*$(0.01, 4)$.

Figure 1 shows the percentage difference in RMSE (root mean squared error; the lower, the better) relative to RLFM on the 10 simulated data for 5 different models: GMF-LASSO, GMF-RP, GMF-RF (with 100 trees), GMF-GB (with 10,000 trees and shrinkage parameter 0.001) and GMF-BART (with 100 trees). It can be seen that all GMF models have negative differences; that means they have lower RMSE than RLFM. These results are statistically significant since no boxplot overlaps zero. In particular, by putting a LASSO penalty on linear regression, the model performs feature selection and reduces RMSE by around 8%. Notice that GMF-LASSO does not model nonlinearity in data; the improvement is due to the feature selection capability provided by LASSO. The other GMF models are using non-parametric tree models and provide improvement roughly by an extra 2% because they are able to perform feature selection and nonlinearity modeling simultaneously.

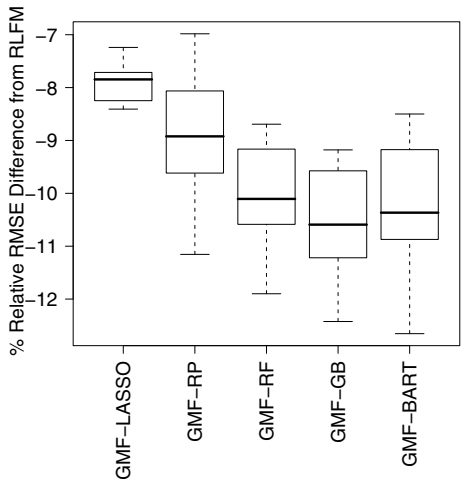

Figure 1: Boxplot of percentage RMSE difference relative to RLFM for 10 simulated datasets

Table 1: The test-set RMSE on MovieLens-1M

Model	Test RMSE	Warm-Start RMSE	Cold-Start RMSE
GMF-BART	0.9340	0.8780	0.9753
GMF-LASSO	0.9341	0.8779	0.9755
GMF-RF	0.9343	0.8777	0.9760
GMF-GB	0.9344	0.8791	0.9753
GMF-RP	0.9359	0.8784	0.9783
RLFM	0.9363	0.8814	0.9766
fLDA	0.9381	—	—
Factor-Only	0.9422	—	—
FilterBot	0.9517	—	—
MostPopular	0.9726	—	—
Feature-Only	1.0906	—	—
Constant	1.1190	—	—

5.2 MovieLens Dataset

We now show an extensive comparative study on the benchmark MovieLens-1M dataset, which consists of 1M ratings from 6,040 users to 3,706 movies. Following [4] and [5], we sort the ratings by time and put the first 75% of ratings into the training set, and set aside the rest 25% as the test set. Note that while most items in the test set are also present in the training set, 900 of 2,035 users in the test set are new users who do not have any ratings in the training period (i.e. cold start). For each user, we use age, gender, occupation and the first digit of zip code as features. For each movie, the movie genre is used as the feature.

Table 1 shows the RMSE of different models. Note that GMF-RF uses 100 trees, GMF-GB uses 10,000 trees and shrinkage parameter 0.001 and GMF-BART uses 100 trees. We run GMF methods and RLFM for 50 iterations, and choose number of factors equal to 12. For comparison purposes, we also include the RMSE numbers of a number of baseline methods reported in [4, 5]. Specifically, **fLDA** is the factorization model based on latent Dirichlet allocation [5]; **Factor-Only** is the usual matrix factorization model

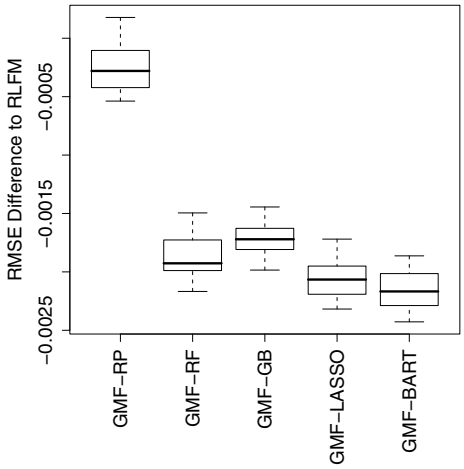

Figure 2: Boxplot of test-set RMSE differences from RLFM on 20 bootstrap samples of MovieLens-1M

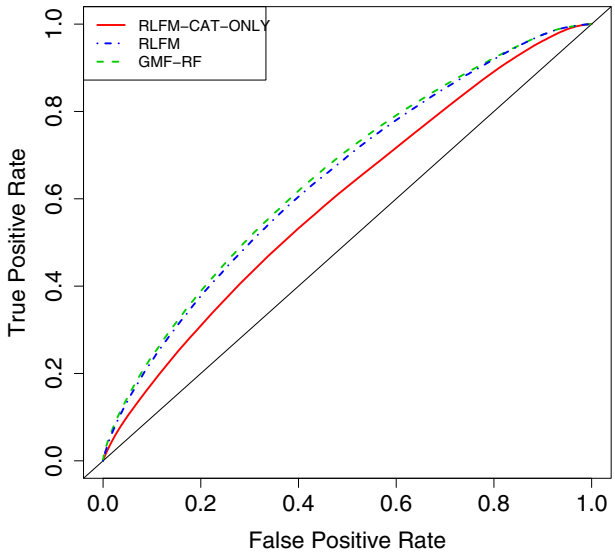

Figure 3: ROC curves for Yahoo! Buzz data.

with zero prior means on both user and movie factors; **Feature-Only** is a regression model using only user and movie features; **FilterBot** is a similarity-based collaborative filtering method [25]; **MostPopular** is a model only based on user and item bias which are regularized using features; and **Constant** is the model that predicts all ratings as the mean of the ratings in the training set. It can be clear seen that all GMF models are better than the other models.

For the matrix-factorization-based models, we also show the RMSE for the warm-start users (i.e., users who rated movies in training set) and the RMSE for the cold-start users (i.e. users who did not have any rating in the training set). It can be seen that GMF models using nonlinear models can usually improve the accuracy for both cold-start and warm-start over RLFM.

To understand statistical significance, we generate 20 bootstrap samples by sampling with replacement from the test set with the sample size equal to the number of observations in the test set, and report the RMSE differences relative to RLFM (the lower, the better) for the GMF models on the 20 bootstrap samples. The results are shown in Figure 2. It is clear that GMF-LASSO, GMF-RF, GMF-GB and GMF-BART consistently outperform RLFM. Since GMF-LASSO performs as well as the GMF tree models, variable selection plays a more important role in this data set than nonlinearity modeling. Note that GMF-RP does not perform as well as other GMF tree models because it is based on a single tree per factor dimension, where others are based on an ensemble of trees through bagging or boosting. It is well known that ensemble methods usually provide improvement in accuracy.

5.3 Yahoo! Buzz Dataset

Finally, we show the experimental results on the Yahoo! Buzz dataset [5]. We gain access to this dataset through the authors of [5]. Yahoo! Buzz (buzz.yahoo.com) is a website which pulls in stories from plenty of web publishers and recommends the "hottest" ones to users. Users can influence the ranking of articles by rating the articles; a user can cast a "buzz up" vote (or a "buzz down" vote) for an article if he/she likes (or dislikes) it. The dataset was collected to study predictability of user's preference on articles based on their historical buzz up/down voting activities. It consists of 620,883 votes on 10,468 articles from 4,026 users in 3 months; the votes in the first two months are used as training data, and the last month data is used for test. While every user has votes in both training and test period, most articles in test data are new. In the dataset, there are three rating values: 1, -1 and 0. 1 represents that the user casted a buzzed up vote for the article; -1 represents buzz down. Since the majority of the voting activities are buzz up, for each user who has N votes, a randomly sampled set of N articles that the user did not vote on are selected and their ratings are set to 0. Intuitively this assumes users only have strong preferences on a small set of the articles and they are not interested in random articles. For each user we use age and gender as features. For the articles, besides using the categories, we use 800 frequent stemmed words from their titles and descriptions by removing stop words and using the Porter stemmer.

We ran all the GMF models on the dataset. The performance of each model is evaluated based on its ROC curve. However, to make the ROC plot clear, we only report GMF-RF (with Number of factors equal to 15, 10 iterations, and 50 trees per factor dimension); the GFM models are all close to one another. For comparison purposes, we include RLFM (with 800 words as features) and RLFM-CAT-ONLY (which is RLFM with only article category features and no keywords). To plot the ROC curves, in the test data we treat both rating 0 and -1 as negative and rating 1 as positive. The ROC curves of the three models are shown in Figure 3. It can be seen that using the words as features significantly improves the prediction accuracy, and using random forest to select important keywords and model nonlinearity provides further improvement, although the improvement is not large.

As a byproduct of using random forests, we get a ranked list of features by their importance. In the following, we present a case study of using GMF-RF to select important keywords. In random forest, the importance of a feature is measured by the percentage of the prediction accuracy decrease when the values of this feature in the out-of-bag samples are randomly permuted and re-evaluated by each tree. Therefore, the worse the prediction accuracy becomes after permuting the values of a feature, the more important the feature is. Table 2 shows the top 10 stemmed words together with the importance scores (mean decrease in mean squared error) and frequencies in articles. The importance values are computed by taking

Table 2: The top important words selected by random forest with 50 trees for Yahoo! Buzz data.

Word	Frequency	Importance (% MSE Decrease)
reuter	15.44%	64.41
obama	24.75%	39.14
presid	21.73%	15.17
u.s.	17.54%	14.85
barack_obama	14.48%	13.49
barack	14.72%	12.53
israel	3.10%	11.45
isra	3.60%	9.14
flu	2.85%	7.54
said	29.51%	7.07

the mean of the importance values for random forests modeling the item bias β_j and the item factors $\boldsymbol{v}_j$.

6. CONCLUSION

In this paper, we generalized matrix factorization by incorporating feature-based regression models. The idea of using regression to regularize factors was explored in [4, 32], but only linear models were considered. We made a key observation that linear models can be substituted by arbitrary regression models. Although this idea is a simple extension to [4] conceptually, it presents formidable model fitting challenges that are handled with ease in our MCEM fitting procedure by leveraging off-the-shelf regression routines in the M-step. The generalization presented is practically important because it allows advances in regression modeling (e.g., nonlinearity modeling and feature selection) to be directly leveraged – one can simply plug and play with any regression models. In particular, we described a scalable algorithm for fitting such a model and rigorously showed that the ability of using off-the-shelf regression methods provides significant accuracy improvement over linear models on both real and simulated data. We conclude that our generalized matrix factorization framework provides a rich class of models to seamlessly handle both cold and warm start scenarios in recommender problems in a unified framework and provides excellent accuracy in practice.

7. REFERENCES

[1] KDD cup and workshop. 2007.

[2] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. Low-rank matrix factorization with attributes. *CoRR*, abs/cs/0611124, 2006.

[3] J. Abernethy, F. R. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering:operator estimation with spectral regularization. *JMLR*, 2009.

[4] D. Agarwal and B. Chen. Regression-based latent factor models. In *KDD*, 2009.

[5] D. Agarwal and B. Chen. fLDA: matrix factorization through latent Dirichlet allocation. In *WSDM*, 2010.

[6] D. Agarwal, B. Chen, P. Elango, N. Motgi, S. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS*, 2009.

[7] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Comm. of the ACM*, 1997.

[8] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, 2007.

[9] J. Booth and J. Hobert. Maximizing generalized linear mixed model likelihoods with an automated monte carlo EM algorithm. *J. R. Statist. Soc. B*, 1999.

[10] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[11] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[12] L. Breiman. Random forests. *Machine learning*, 2001.

[13] A. Z. Broder. Computational advertising and recommender systems. In *RecSys*, 2008.

[14] H. Chipman, E. George, and R. McCulloch. BART: Bayesian additive regression trees. *Annals of applied statistics*, 2010.

[15] M. Claypool and A. Gokhale, et al. Combining content-based and collaborative filters in an online newspaper. In *RecSys Workshop*, 1999.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 1977.

[17] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 2001.

[18] A. E. Gelfand. Gibbs sampling. *JASA*, 1995.

[19] N. Good and J. B. Schafer, et al. Combining collaborative filtering with personal agents for better recommendations. In *AAAI*, 1999.

[20] P. Green. Penalized likelihood for general semi-parametric regression models. *International Statistical Review/Revue Internationale de Statistique*, 1987.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2009.

[22] N. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *ICML*, 2009.

[23] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, 1989.

[24] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, 2002.

[25] S.-T. Park and D. Pennock, et al. Naïve filterbots for robust cold-start recommendations. In *KDD*, 2006.

[26] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005.

[27] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, 2008.

[28] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2008.

[29] A. I. Schein, L. K. Saul, and L. H. Ungar. A generalized linear model for principal component analysis of binary data. In *AISTATS*, 2003.

[30] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML*, 2003.

[31] N. Srebro, J. Rennie, and T. Jaakkola. Maximum margin matrix factorizations. In *NIPS*, 2005.

[32] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, 2009.

[33] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. of the Royal Stat. Soc. B*, 1996.

[34] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *SIGIR*, 2007.