Regression-based Latent Factor Models

Deepak Agarwal
Yahoo! Research
Sunnyvale, CA, USA
dagarwal@yahoo-inc.com

Bee-Chung Chen
Yahoo! Research
Sunnyvale, CA, USA
beechun@yahoo-inc.com

ABSTRACT

We propose a novel latent factor model to accurately predict response for large scale dyadic data in the presence of features. Our approach is based on a model that predicts response as a *multiplicative* function of row and column latent factors that are estimated through separate regressions on known row and column features. In fact, our model provides a single unified framework to address both cold and warm start scenarios that are commonplace in practical applications like recommender systems, online advertising, web search, etc. We provide scalable and accurate model fitting methods based on Iterated Conditional Mode and Monte Carlo EM algorithms. We show our model induces a stochastic process on the dyadic space with kernel (covariance) given by a polynomial function of features. Methods that generalize our procedure to estimate factors in an online fashion for dynamic applications are also considered. Our method is illustrated on benchmark datasets and a novel content recommendation application that arises in the context of Yahoo! Front Page. We report significant improvements over several commonly used methods on all 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

Sparse, Interaction, Predictive, Latent Factor, Recommender Systems, Dyadic Data, Metadata

1. INTRODUCTION

Estimating interactions for massive but highly incomplete dyadic data is an important task in several applications such as recommender systems, web advertising, social networks, etc. Typically, the training data is collected for some fixed time period, and consists of a response variable y_{ij} available for a small fraction of *dyads* or cells (i, j) (pairs of elements from two different sets referred to as rows and columns) along with features attached to rows, columns and cells. The goal is to predict the response value for a new cell $(i, j)_{new}$ at some time in the future, row i and/or column j corresponding to $(i, j)_{new}$ may not necessarily have occurred during the training period. For instance, in a movie recommender system, rows and columns correspond to users and movies respectively and the response y_{ij} may denote the explicit rating provided by user i to movie j. Row features w_i may include age, gender, geo-location of the user; column features z_j may include genre, title words, release date of the movie. One may also have cell specific features x_{ij} like: *Is the user's favorite actor playing a lead role in the movie*? The accuracy with which one could predict the rating a user i would provide to movie j at some future time point is an important component that affects the quality of the recommender system. Another example is online advertising where rows and columns correspond to queries and ads respectively, the goal is to predict the probability of a click on an ad for a given query. The success of several other applications that include e-commerce, web search, social networks also depend on similar estimation problems. For ease of exposition, we shall refer to rows and columns as *users* and *items* respectively.

Accurate estimation of interactions for large, incomplete and dynamic dyadic data entails several challenges. Lack of data (data sparseness) is a key issue; observations are available only for a small fraction of possible dyads. In fact, the distribution of available observations is highly non-random; a small fraction of users (items) typically account for a large fraction of data while the remaining are sparsely distributed among others. Moreover, the system is not static and often prediction is required for dyads that involve new users and/or items over time (also commonly referred to as the *cold-start* problem). In several applications, both users and items are associated with a set of informative features. For example, users may supply demographic information like age, gender, occupation at the time of registration, and we can infer the geo-location of a user through his/her IP address. When items are movies, we may know their genres, release dates, cast, etc. For news items, we can extract features from the article content. One approach is to build a predictive model that is entirely based on user and item features, such a method does not suffer from the cold-start problem. However, it treats old users (items) similarly as new users (items) and does not utilize past interaction data to estimate user/item centric models; it also fails to capture correlations that are often present in *residuals* associated with the same user/item. In fact, models that ignore features and are only based on users' past interaction with items have been shown to predict well for old users and items. This is evident in the plethora of methods that have been proposed in the context of the movie recommender systems.

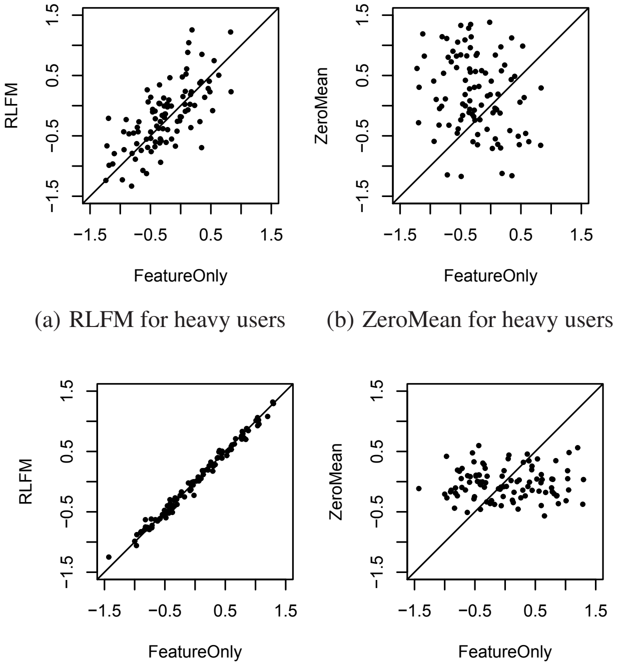
KDD'09, June 28–July 1, 2009, Paris, France.

Although effective for warm-start, they fail to address the cold-start problem. Methods that simultaneously incorporate past interaction data and features for accurate prediction and *smoothly* handle both cold-start and warm-start scenarios are attractive.

We propose a regression-based latent factor model (*RLFM*) that a) improves prediction for old user-item dyads by simultaneously incorporating features and past interactions and b) provides predictions for new dyads through features. In addition, we also provide a strategy for online estimation of latent factors. We discuss and illustrate our method on a new recommender system problem that arises in the context of Yahoo! Front Page where dyads are dynamic in nature, methods that work only for old dyads are ineffective [2] here. We also illustrate our method on movie recommender application and show that simultaneously incorporating features and past interactions through our *RLFM* significantly improves prediction accuracy for both old and new users. Since the Netflix data does not provide user features, we illustrate our methods on Movie Lens and Each Movie datasets.

The key idea of *RLFM* is to associate latent factors (or profiles) u_i and v_j to user i and item j where u_i and v_j are r-dimensional latent factors. We shall refer to r as the number of latent dimensions. Interaction is captured through a multiplicative function $u_i'v_j$. This is similar in spirit to SVD for complete matrices; the incompleteness, imbalance and noise in data makes it a significantly harder problem. In particular, it is important to *regularize* the latent factors to avoid over-fitting and achieve the best bias-variance trade-off. Several authors [24, 8, 1, 22] have recently studied the problem, especially in the context of the Netflix competition where it is referred to as *matrix factorization*. The regularization is mostly based on a zero-mean Gaussian prior on the factors, we refer to this method as *ZeroMean*. Our method also assumes a Gaussian prior but replaces the zero mean with a feature-based regression; thus, it simultaneously regularizes both user and item factors through known features. We show that such a prior poses challenging issues for scalable computation and has other theoretical implications. In particular, it addresses both cold and warm-start problems seamlessly through a single model. It also induces marginal correlations among response values that share a common user or item. Correlation provides extra information for response at a dyad (i, j) and improves predictive performance. Thus, if two movies are correlated, knowing the rating of user i on the first one in the past helps us accurately predict his/her rating on the other one. In fact, we show that our model is a stochastic process (similar to a Gaussian process) on the dyadic space with non-zero covariance only among responses with either a common user or item. While Gaussian processes are known to provide accurate predictive models in several contexts [21], the main computational bottleneck is the cubic computation associated with a dense correlation matrix. In our case, we exploit the special structure of the covariance process and reduce the computational complexity to "essentially" linear in the number of data points, users and items.

Our *RLFM* method works by *anchoring* user/item profiles around a global feature-based one whereby user/item-specific profiles are then constructed by estimating deviations from the global ones in a smooth fashion; the amount of deviation depends on sample size and correlations among observations. In particular, users/items with sparse data are deemed more unreliable to deviate and "shrunk" aggresively to the global one. Thus, users/items start out with profiles based on their known features that gets refined smoothly with the availability of more data. This ability to move from *coarse* to *fine* resolutions in a *smooth* fashion after accounting for differing sample sizes, correlation and heterogeneity are key aspects that provides for an accurate, scalable and general predictive method through *RLFM* for large scale dyadic data. To provide more intuition, Figure 1 shows the relationship among user latent factors estimated separately for a sample of heavy and light users on the Yahoo! Front Page using *RLFM* , *ZeroMean* and *FeatureOnly* . Here, *ZeroMean* is a model that uses latent factors regularized through zero mean priors(matrix factorization), *FeatureOnly* is based on features alone while *RLFM* uses latent factors regularized through regression based priors. For light users, *ZeroMean* "shrinks" the profiles close to zero and leads to biased estimates. *RLFM* on the other hand recognizes the data sparseness and falls back on the *FeatureOnly* profiles. For heavy users, *ZeroMean* is aggressive and tends to overfit the data. *RLFM* on the other hand anchors itself around *FeatureOnly* profiles and deviates from it in a smooth way that leads to better regularization.



(a) RLFM for heavy users (b) ZeroMean for heavy users

(c) RLFM for light users (d) ZeroMean for light users

Figure 1: Comparing latent factors estimated through *RLFM* , *ZeroMean* and *FeatureOnly* . Each point (x, y) in a plot reports the estimated values, x and y, of the first latent factor of a user using the two methods indicated in the plot. For light users, *ZeroMean* collapse to zero, *RLFM* on the other hand collpases to *FeatureOnly* .

Our **contributions** are as follows. We propose a novel class of latent factor models called *RLFM* that incorporates both known user/item features and past interaction data into a single model for predicting dyadic response in a generalized linear model (GLM) framework. Our method provides a single unified framework to handle both cold and warm start scenarios in a smooth way. We prove theoretical equivalence to a stochastic process model with polynomial kernel on a dyadic space. We provide a scalable, accurate and robust model fitting procedure through a Monte Carlo EM algorithm with complexity linear in number of observed dyads, users and items. We also provide a procedure to generalize our method to dynamic settings through an online updating procedure for the factors. Our methods are illustrated on benchmark datasets and on a new recommender system application that arises in the context of displaying the best stories to users visiting the Yahoo!

Front Page. Our method significantly outperforms several existing and commonly used methods on all datasets we considered.

2. MODELING DETAILS

Our regression-based latent factor model (*RLFM*) and the model fitting problems are mathematically defined in Table 1. *RLFM* is a two-stage hierarchcial latent factor model, fitting algorithms find the maximum likelihood estimate of the model parameters and posterior distribution of factors for a given training dataset. In this section, we describe the model and state some of its properties. In section 3, we describe scalable model-fitting algorithms implemented through Monte Carlo EM (MCEM) and Iterated Conditional Mode (ICM) algorithms. We begin with notations and preliminaries followed by a detailed description of *RLFM* for Gaussian response. For binary response, we fit a logistic regression through a variational approximation as discussed in Section 3.2. In Section 4, we describe a method to smooth latent factors over time.

2.1 Preliminaries

For ease of exposition, we describe our model by considering the problem of predicting user ratings on items. Let index i run through users, and index j run through items. Dyad (i, j) has an associated rating (rating given by user i to item j). We note that in another application, i may represent a web page, j may represent an ad, and dyadic response may represent the click-through rate. For the sake of discussion, we shall refer to rows, columns and dyads are as *users*, *items* and *ratings* respectively. Let M be the number of users, N be the number of items and K be the number of ratings. The notations $\{o_i\}$, $\{o_j\}$ and $\{o_{ij}\}$ denote the set of o's over all users, items and ratings respectively, where o is a placeholder. The notation $[a|b]$ denotes the conditional probability distribution of random variable a given b, and $[b]$ denotes the marginal distribution of b. Let $N(\mu, \sigma^2)$ and *MVN*(μ, Σ) denote the single-variate and multi-variate normal distributions with mean μ (vector for multi-variate) and variance σ^2 and Σ respectively. Next, we introduce notations for observed and latent data.

Observed Data: We observe ratings (the response), user features, item features and features specific to ratings. Let $w_i^{p\times 1}$ and $z_j^{q\times 1}$ denote feature vectors for user i and item j; y_{ij} and $x_{ij}^{s\times 1}$ denote the response and feature vector associated with dyad (i, j). When convenient, we use $X_{ij} = (x_{ij}, w_i, z_j)$ to denote known feature values associated with dyad (i, j). Our method learns the relationship between y_{ij} and X_{ij} from observed data to predict unobserved y_{ij}^{new} for a dyad $(i, j)_{new}$ with features X_{ij}^{new}.

Latent Data: To model $[\{y_{ij}\}|\{X_{ij}\}]$, we augment observed user and item features with *unobserved* latent factors attached to each user and item. Our approach connects the response and observed features indirectly through the factors that are predicted at a finer user/item resolution through a feature-based regression model. The estimated factors are in turn used to predict the response (see graphical model in Figure 2 and equations in Table 1 for more details). Specifically, we attach a latent "profile" $(\alpha_i, u_i^{r\times 1})$ to user i and a latent "profile" $(\beta_j, v_j^{r\times 1})$ to item j. Now, we have $(r + 1)(M + N)$ latent factors that need to be estimated from the data. When convenient, we use $\delta_{ij} = (\alpha_i, u_i, \beta_j, v_j)$ to denote all latent factors associated with dyad (i, j). Drawing analogy to state-space models, latent factors can also be interpreted as unknown *states* and our model specifies how states and observations $(\{y_{ij}\})$ are generated. Note that we use latent factor, profile and state interchangeably in this paper.

$$\begin{array}{lll} \textbf{First stage:} & y_{ij} \sim N(m_{ij}, \sigma^2), \text{or} \quad \text{(Gaussian)} & \\ & y_{ij} \sim \text{Bernoulli}(m_{ij}) \quad \text{(Logistic)} & \\ & l(m_{ij}) = x'_{ij}\boldsymbol{b} + \alpha_i + \beta_j + u'_i v_j & (1) \\ \textbf{Second stage:} & \alpha_i = g'_0 w_i + \epsilon_i^{\alpha}, \quad \epsilon_i^{\alpha} \sim N(0, a_\alpha) & (2) \\ & \beta_j = d'_0 z_j + \epsilon_j^{\beta}, \quad \epsilon_j^{\beta} \sim N(0, a_\beta) & \\ & u_i = G w_i + \epsilon_i^{u}, \quad \epsilon_i^{u} \sim \mathit{MVN}(\mathbf{0}, A_u) & \\ & v_j = D z_j + \epsilon_j^{v}, \quad \epsilon_j^{v} \sim \mathit{MVN}(\mathbf{0}, A_v) & \end{array}$$

Notations:

- y_{ij} is the observed rating given by user i to item j.
- $l(m_{ij}) = m_{ij}$ for Gaussian; $l(m_{ij}) = \log \frac{m_{ij}}{1-m_{ij}}$ for Logistic.
- $x_{ij}^{s\times 1}$, $w_i^{p\times 1}$ and $z_j^{q\times 1}$ are the dyadic, user and item feature vectors for user i and item j, respectively.
- $\alpha_i^{1\times 1}$, $\beta_j^{1\times 1}$, $u_i^{r\times 1}$ and $v_j^{r\times 1}$ are latent factors. We call r the number of latent dimensions.
- $\boldsymbol{b}^{s\times 1}$, $g_0^{p\times 1}$ and $d_0^{q\times 1}$ are regression coefficient vectors. $G^{r\times p}$ and $D^{r\times q}$ are regression coefficient matrices.
- $a_\alpha^{1\times 1}$ and $a_\beta^{1\times 1}$ are unknown variances. $A_u^{r\times r}$ and $A_v^{r\times r}$ are unknown variance-covariance matrices.

Model fitting problem: Let $\Theta = (\boldsymbol{b}, g_0, d_0, G, D, a_\alpha, a_\beta, A_u, A_v)$, $X_{ij} = (x_{ij}, w_i, z_j)$ and $\delta_{ij} = (\alpha_i, \beta_j, u_i, v_j)$. Given $\{y_{ij}\}$ and $\{X_{ij}\}$, our goal is to find the maximum likelihood estimate $\hat{\Theta}$, i.e., the estimate that maximizes $\Pr[\{y_{ij}\} \mid \{X_{ij}\}, \Theta]$, and also estimate the posterior distribution of the latent factors $[\{\delta_{ij}\} \mid \{y_{ij}\}, \{X_{ij}\}, \hat{\Theta}]$ (used for prediction).

Prediction: Let $\{y_{ij}^{old}\}$ be the observations in the training data. Given features X_{ij}^{new}, we predict $\psi(y_{ij}^{new})$ as $E[\psi(y_{ij}^{new}) \mid \{y_{ij}^{old}\}, X_{ij}^{new}, \hat{\Theta}]$. For Gaussian, $\psi(y) = y$, for Logistic $\psi(y) = \Pr(y = 1)$.

Table 1: Regression-based Latent Factor Model

Hierarchical Modeling Approach: We now describe our two-stage hierarchical modeling approach. The first stage of our model specifies $[\{y_{ij}\}|\{X_{ij}\}, \{\delta_{ij}\}, \Theta_1]$, i.e., a model to generate the response when both observed features and latent factors are known. The second-stage specifies a generative model for the latent profiles, i.e., $[\{\delta_{ij}\}|\{X_{ij}\}, \Theta_2]$. Here, $\Theta = (\Theta_1, \Theta_2)$ are fixed parameters that are required to specify the appropriate distributions and are estimated from data in practice. In particular, the parameter Θ_2 are called *hyperparameters*. The two-stages specify the joint distribution of $[\{y_{ij}\}, \{\delta_{ij}\} \mid \{X_{ij}\}, \Theta]$, which is equal to

$$[\{y_{ij}\} \mid \{X_{ij}\}, \{\delta_{ij}\}, \Theta_1] \cdot [\{\delta_{ij}\} \mid \{X_{ij}\}, \Theta_2] \qquad (1)$$

The joint distribution in Equation 1 is obtained by *stitching* together distributions in the two stages. Marginalizing (integrating out) over $\{\delta_{ij}\}$ provides the desired distribution $[\{y_{ij}\}|\{X_{ij}\}, \Theta]$ for prediction. We note that marginalization involves computing a high dimensional integral that is generally not available in closed form. This is the main computational challenge that arises when working with such hierarchical models.

2.2 First Stage Model

The fist stage model specifies how observations $\{y_{ij}\}$ are generated given the latent factors. We assume *conditional independence* of response which is mathematically expressed as

$$[\{y_{ij}\}|\{X_{ij}\}, \{\delta_{ij}\}, \Theta_1] = \prod_{ij} [y_{ij}|X_{ij}, \delta_{ij}, \Theta_1].$$

This intuitively means when latent user/item profiles are known without error, the response arising through user-item interaction for dyads are independent of each other. In general, latent profiles are estimated with error and averaging over the uncertainty induce *de-*

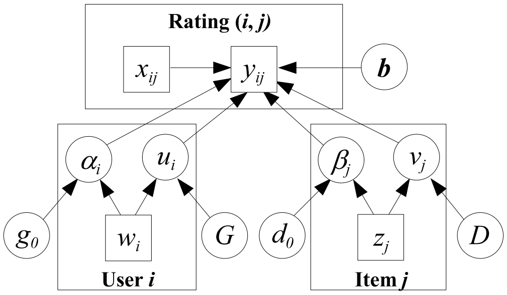


Figure 2: Graphical representation of *RLFM* . Variance components (σ^2, a_α, a_β, A_u, A_v) are omitted for succinctness.

pendencies in dyadic response as we shall see later. The first stage model is also referred to as *observation* equation.

We specify $[y_{ij}|X_{ij}, \delta_{ij}, \Theta_1]$ in a *generalized linear model* (GLM) framework , linear, logistic and Poisson regression are special cases (see [18, 4] for more details). Letting $m_{ij} = E[y_{ij}|\delta_{ij}, x_{ij}]$, we assume $l(m_{ij})$ is given by Equation (1) in Table 1, where l is some transformation of the mean whose range is the real line. In this paper, the illustrative datasets are analyzed using the linear and logistic regression models in the first stage; we provide computation details for both but note that extension to other members of the GLM family is standard. For ease of exposition, we develop the theory for Gaussian response (linear regression) where several formulae are available in closed form.

2.3 Second Stage Model

Our second stage model specifies $[\{\delta_{ij}\}|\{X_{ij}\}, \Theta_2]$ which we shall refer to as the *state* equation. For many applications, $K << MN$; in fact, the imbalance in our data is such that $K \propto (M+N)$. Moreover, there is heterogeneity in the number of observations associated with users/items – a small fraction of users/items account for a large fraction. The total number of parameters in the first stage observation model is $s+M+N+r(M+N) = s+(r+1)(M+N)$. Clearly, the number of parameters is large and maximum likelihood estimates (MLE) of model parameters overfit even for reasonably small values of r. Hence, it is imperative to put constraints on the latent factors through the state equation to avoid over-fitting. In fact, the state equation determines the quality and predictive performance of the model to a large extent. In this paper, we propose a novel state equation that assumes latent factors $\{\delta_{ij}\}$ are functions of observed features $\{X_{ij}\}$. Specifically, we assume conditional independence among latent factors

$$\begin{aligned}&[\{\alpha_i\}, \{u_i\}, \{\beta_j\}, \{v_j\} \mid \{w_i\}, \{z_j\}, \Theta_2] = \qquad (2)\\ &\prod_{i=1}^{M}[\alpha_i \mid w_i, g_0, a_\alpha] \prod_{i=1}^{M}[u_i \mid w_i, G, A_u]\\ &\prod_{j=1}^{N}[\beta_j \mid z_j, d_0, a_\beta] \prod_{j=1}^{N}[v_j \mid z_j, D, A_v]\end{aligned}$$

where $\Theta_2 = (g_0, a_\alpha, G, A_u, d_0, a_\beta, D, A_v)$. The component latent factor distributions are given by Equation (2) in Table 1. We note that it is also possible to use non-linear functions $G(w)$, $D(z)$, $g_0(w)$, $d_0(z)$, we assumed linearity to ensure scalability. To complete model specification, we note that it is possible to further regularize regression coefficients by using any off-the-shelf procedure like ridge regression, LASSO. We use a t-prior on the coefficients recently introduced by [13] which performs automatic variable selection by setting the coefficients of non-informative features to be close to zero.

2.4 Prediction Function

After fitting the model using observations $\{y_{ij}^{old}\}$ and $\{X_{ij}^{old}\}$, we obtain an estimate of the parameters $\hat{\Theta}$. Then, given the features X_{ij}^{new} for a new dyad we would ideally predict the response y_{ij}^{new} by its posterior predictive mean $E[y_{ij}^{new} \mid \{y_{ij}^{old}\}, X_{ij}^{new}, \hat{\Theta}]$ after marginalizing over $[\delta_{ij}|\{y_{ij}^{old}\}, \{X_{ij}^{old}\}, \hat{\Theta}]$, this can be computationally expensive at runtime. For the sake of efficiency, we simply use point estimates that are commonly used in practice and does not necessarily lead to loss in prediction accuracy [27]. That is, we predict y_{ij}^{new} by

$$(x_{ij}^{new})'\hat{\boldsymbol{b}} + \hat{\alpha}_i + \hat{\beta}_j + \hat{u}_i'\hat{v}_j,$$

where $\hat{\phi} = E[\phi \mid \{y_{ij}^{old}\}, \{X_{ij}^{old}\}, \hat{\Theta}]$ for $\phi = \alpha_i$, β_j, u_i and v_j. Note that these posterior means are also the output from the fitting procedure. For new users and items, the posterior means are just prior means predicted purely by features; e.g., $\hat{\alpha}_i = \hat{g}_0'w_i$, etc. For old users and items, the posterior means provide the "right" balance between features and past activity through our model.

2.5 Special Cases of Our Model

We show some popular classes of warm-start and cold-start models arise as special cases of *RLFM* , providing further insights on how it provides a unified framework to deal with both scenarios in a smooth way. In fact, assuming $g_0 = d_0 = G = D = \mathbf{0}$, one obtains the probabilistic matrix factorization model, which we call *ZeroMean* , [23, 24, 22] that has successfully modeled explicit movie ratings data. Next, we point out the relation between em RLFM and a pure feature-based model for a Gaussian response variable. A pure feature-based model is given as

$$y_{ij} = h(x_{ij}, w_i, z_j) + \epsilon_{ij} \qquad (3)$$

where h is an unknown function and the ϵ's are zero mean white noise random variables. One natural choice with categorical predictors is a linear h, i.e.,

$$h(x_{ij}, w_i, z_j) = x_{ij}'\boldsymbol{b} + g_0'w_i + d_0'z_j + w_i'B^{p\times q}z_j.$$

The unknown matrix of coefficients B capture interactions among users and items, estimating B may however be challenging for large values of p and/or q. In general, one takes recourse to some form of penalization on the entries of B (e.g. L_0, L_1 or L_2 penalties). Another attractive approach that is often used is a low rank approximation of $B = G'D$ that reduce the number of parameters from pq to $r(p + q)$. Plugging Equation (2) into Equation (1) in Table 1 for the Gaussian case, we get

$$\begin{aligned} y_{ij} &= x_{ij}'\boldsymbol{b} + g_0'w_i + d_0'z_j + w_i'G'Dz_j + \epsilon_{ij}^* \qquad (4)\\ \epsilon_{ij}^* &= \epsilon_{ij} + \epsilon_i^\alpha + \epsilon_j^\beta + (\epsilon_i^u)'\epsilon_j^v + (\epsilon_i^u)'Dz_j + (\epsilon_j^v)'Gw_i \end{aligned}$$

Hence, the pure feature based model is obtained as a special case of *RLFM* by assuming a zero variance for random effects, i.e., $\epsilon_{ij}^* = \epsilon_{ij}$ in Equation 4. We shall refer to this as *FeatureOnly* . The additional randomness in *RLFM* introduced through a two-stage process induce dependencies in response values sharing the same user or item and improves predictive performance.

2.6 Stochastic Process on Dyadic Space

Regularizing latent factors through regression has important consequences when modeling sparse dyadic data. For users/items with little data, we obtain reliable factor estimates by using the regression estimates as a *fallback*. In fact, the model provides a unified framework to deal with both cold and warm start situations; we predict factors for new users/items through a feature-based regression but converge to a user/item level profile that may deviate substantially from the global regression for *heavy hitters*. Moreover, the regression on factors indirectly induce marginal dependencies among response variables as we discuss next.

Marginal dependencies: After adjusting for the effect of features, the residuals associated with the same user/item are expected to be correlated due to unmeasured latent effects; exploiting these correlations can significantly improve performance. For instance, if two items are correlated (people who bought A also bought B), it helps predict user's response on B given his past response to A. Our *RLFM* captures item-item and user-user similarity through a quadratic form in the features and improves prediction. In particular, pairwise correlation between user (item) ratings are constant across users (items) and depend on the item (user) features. The marginal moments can be computed in closed form for the Gaussian distribution and are given below.

$$\begin{aligned} E(y_{ij}) &= x'_{ij}\boldsymbol{b} + g'_0 w_i + d'_0 z_j + w'_i G' D z_j \qquad (5) \\ Var(y_{ij}) &= \sigma^2 + a_\alpha + a_\beta + tr(A_u A_v) + \\ &\qquad z'_j D' A_u D z_j + w'_i G' A_v G w_i \\ cov(y_{ij}, y_{ij^*}) &= a_\alpha + z'_j D' A_u D z_{j^*} \\ cov(y_{ij}, y_{i^*j}) &= a_\beta + w'_i G' A_v G w_{i^*} \end{aligned}$$

The expression shows our model defines a stochastic process on the dyadic space with polynomial kernels specifying pairwise correlations between user and item ratings. In particular, from standard stochastic process theory, the predicted value at a dyad $(i,j)^{new}$ is given by

$$\begin{aligned} E(y_{ij}^{new} | \{y_{ij}^{old}\}) &= E(y_{ij}^{new}) + \qquad (6) \\ &\quad \kappa_{ij}^{new'} P\{(y_{ij}^{old} - E(y_{ij}^{old}))\} \end{aligned}$$

where P is the precision matrix of $\{y_{ij}^{old}\}$ and

$$\kappa_{ij}^{new} = Cov(y_{ij}^{new}, \{y_{ij}^{old}\}).$$

Also note that κ_{ij}^{new} is non-zero only for entries in the training data across the i^{th} row and j^{th} column. Thus, *RLFM* predicts at a dyad by adding an *adjustment* to the linear feature-based predictor; the adjustment is a weighted average of row and column residuals with weights depending on the induced correlations based on features. Hence, if item j is correlated with other items and/or user i has correlations with others, the stochastic process will exploit them to provide an additional adjustment term that will improve performance. In practice, it is computationally infeasible to work directly with the stochastic process defined through Equation 5, our model fitting strategy that works by augmenting observed data through latent factors is one way to scale computations. In fact, the stochastic process defined through Equations (5) and (6) shows the role played by regression parameters in inducing correlations among observations. It also provides the invariant parameters that matter in prediction, namely, $G^{'}D$, $A_u A_v$, $G^{'}A_v G$ and $D^{'}A_u D$. While working directly with the latent factor model (instead of the marginalized one), this invariance is manifested in the non-estimability of u_i and v_j individually; only the product $u_i^{'} v_j$ can be uniquely identifiable from the data. One can also interpret the regression coefficient matrices G and D as providing a rectangular kernel approximation for the dyadic stochastic process.

3. MODEL FITTING

In this section, we provide details of our scalable model fitting algorithm for the Gaussian model. We begin with a brief description of existing methods to fit *ZeroMean* followed by a detailed description of the Monte Carlo EM (MCEM) algorithm to fit *RLFM* ; we also briefly discuss an alternative method, the Iterated Conditional Mode (ICM) algorithm. We then describe a variational approximation for the Logistic model.

Prior fitting methods: Several methods to fit *ZeroMean* have been reported in the literature, these include alternate least squares [8], Iterated Conditional Mode through gradient ascent [24] and Gibbs Sampling [23]. The difficulty in fitting *ZeroMean* arises from the fact that the posterior distribution of latent factors $\{u_i\}$ and $\{v_j\}$ is multi-modal; predictive accuracy depends on how rapidly an algorithm converges to a reasonable mode in the posterior space. The additional complexity of estimating regression parameters in *RLFM* makes model fitting more challenging.

Our model fitting approach: We adopt an EM approach [11] and consider several variations. In particular, we experimented with three methods, alternate least squares (ALS), ICM and MCEM (based on Gibbs sampling), on several datasets (real and simulated) and found MCEM to be the best, both in terms of predictive accuracy and resistance to overfitting with increasing number of latent dimensions r. The latter is an important property since it is computationally expensive to determine the appropriate value of r in practice. We found that ICM tends to overfit when r is large, and ALS performs poorly compared to MCEM and ICM.

3.1 Algorithms for the Gaussian Model

We first describe the MCEM algorithm, then the ICM algorithm and a brief comparison between the two. Our goal is to find Θ that maximizes the marginal log-likelihood of observed response $[\{y_{ij}\} \mid \{X_{ij}\}, \Theta]$; marginalization is performed with respect to latent variables $\{\delta_{ij}\} = (\{\alpha_i\}, \{\beta_j\}, \{u_i\}, \{v_j\})$. This marginal distribution is hard to be maximized directly since the hyperparameters appear in the inverse and determinant of a large $K \times K$ covariance matrix. Thus, we work with the *complete* data likelihood of $[\{y_{ij}\}, \{\delta_{ij}\} \mid \{X_{ij}\}, \Theta]$.

EM Algorithm: Starting from some initial estimate Θ_{init}, the EM algorithm maximizes the marginal log-likelihood by iterating through expectation (E) and maximization (M) steps until the solution converges. Each sweep of the E and M steps is guaranteed not to reduce the marginal log-likelihood. Let Θ_{curr} be the current estimate of Θ in the iterative process. In the E-step, we compute the expected log-likelihood of complete data with respect to the conditional distribution of $[\{\delta_{ij}\}|\{y_{ij}\}, \Theta_{curr}]$. In the M-step, we maximize the expected log-likelihood (from E-step) with respect to Θ and obtain a new estimate. To ensure convergence, it is sufficient to use any *hill climbing* method in the M-step that provides a new improved value of Θ; such variants are called generalized EM (GEM) [19].

Monte Carlo E-step: The E-step in our model cannot be computed in closed form due to the multiplicative terms $u'_i v_j$ in the first-stage model, this makes $[\{\delta_{ij}\}|\{y_{ij}\}, \{X_{ij}\}, \Theta_{curr}]$ a non-standard distribution. However, it is easy to draw Monte Carlo samples through Gibbs sampling from $[\{\delta_{ij}\}|\{y_{ij}\}, \{X_{ij}\}, \Theta_{curr}]$ since the full-conditionals of

$$[\alpha_i|Rest], [\beta_j|Rest], [u_i|Rest], [v_j|Rest]$$

are all Gaussian with means ane variances given in Table 2, where $Rest$ denotes other variables. In fact, user (item) popularity α_i (β_j) parameters are all conditionally independent, and the same holds for user (item) factors u_i's (v_j's). This provides a scalable implementation of each Gibbs iteration through parallelization across users followed by parallelization across items in drawing samples from conditional Gaussian distribution. Hence, we replace the E-step by a Monte Carlo E-step, providing an MCEM algorithm ([9] and references therein). The details of the Gibbs sampling are given in Algorithm 1. For ease of exposition, we use ϕ_k to denote any one of α_i, β_j, u_i and v_j, where k represents either user i or item j. We note that 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 Θ_{new} (an estimate of Θ^*_{new} that is obtained from true E-step) is large relative to $||\Theta_{curr} - \Theta^*_{new}||$, Monte Carlo E-step is wasteful since it is swamped by the Monte Carlo error. There are no rigorous solution 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 30 EM iterations with 5 samples for the first 5 iterations, 20 for next 5 and 100 each for last 20 provides good performance in our applications.

Algorithm 1 Monte-Carlo EM (MCEM) Algorithm

```
Input: $\{y_{ij}\}$, $\{X_{ij}\}$ and initial estimates of $\{\delta_{ij}\}$ and $\Theta$.
Output: $\hat{\Theta}$ that maximizes $\Pr[\{y_{ij}\} \mid \{X_{ij}\}, \Theta]$.
 1: for $t = 1, ..., T$ do
 2:   E-step: Draw $L$ samples of $\{\delta_{ij}\}|\{y_{ij}\}, \Theta_{curr}$ using Gibbs sampling
 3:   for $\ell = 1, ..., L$ do
 4:     for each $\phi_k \in \{\alpha_i\} \cup \{\beta_j\} \cup \{u_i\} \cup \{v_j\}$ do
 5:       Update $\phi_k$ drawn from $N(E[\phi_k|\text{Rest}], \mathit{Var}[\phi_k|\text{Rest}])$
 6:     end for
 7:   end for
 8:   Let $\phi^*_k$ and $\mathit{Var}^*(\phi_k)$ denote the empirical mean and variance of $\phi_k$ computed from $L$ Gibbs samples.
 9:   M-step: Find $\Theta$ that maximizes $E^*\log[\{y_{ij}\}\,\{\delta_{ij}\}|\,\{X_{ij}\}, \Theta]$
10:   for each of the five regression problems (see footnotes) do
11:     Let $(\phi_k, \pi, f_k, a)$ denote the regression problem, where $\pi$ and $a$ are elements in $\Theta$.
12:     Update $\pi$ by regressing response $\phi^*_k$ using feature $f_k$. (We use the bayesglm package in R.)
13:     Let $RSS$ and $n$ be the residual sum of squares and the number of observations in the above regression.
14:     Update $a$ by $(RSS + \sum_k \mathit{Var}^*(\phi_k))/n$.
15:   end for
16: end for
```

Note: We use a quadruple (ϕ_k, π, f_k, a) to denote a regression problem, which represents (response, weight vector or matrix, feature vector and variance). The M-step considers the following five regression problems:

$(y_{ij} - \alpha_i - \beta_j - u_i'v_j,\ \boldsymbol{b},\ x_{ij},\ \sigma^2_{ij})$
$(\alpha_i, g_0, w_i, a_\alpha)$, $(\beta_j, d_0, z_j, a_\beta)$,
(u_i, G', w_i, a_u), (v_j, D', z_j, a_v),

For simplicity, we only consider $A_u^{r\times r} = a_u^{1\times 1}I$ and $A_v^{r\times r} = a_v^{1\times 1}I$ in this algorithm description. When ϕ_k represents u_i or v_j, $\mathit{Var}^*(\phi_k)$ is a vector of variances (the covariances are ignored), and $\sum_k \mathit{Var}^*(\phi_k)$ on line 14 also sums over all the elements in each vector.

M-step: In the M-step, we want to find Θ that maximizes $E\log[\{y_{ij}\}\,\{\delta_{ij}\}|\,\{X_{ij}\}, \Theta]$, where the expectation is taking over $\{\delta_{ij}\}|\{y_{ij}\}, \Theta_{curr}$. The major computation in the M-step involves regressing estimated averages $\{\phi^*_k\}$ from Monte-Carlo samples in E-step against features to get updated values of Θ. Here, one can use any off-the-shelf regression technique; we used a linear regression with t-priors on the coefficient vectors that was recently proposed by [13] as mentioned in Section 2. In fact, the rationale behind adopting an EM instead of a fully Bayesian approach was to easily utilize a large number of effective methods that exist for conducting accurate and scalable linear regression. The details of M-step is given in Algorithm 1.

Scalability: The MCEM is scalable and easily parallelizable. Let T denote the number of EM iterations and L denote the average number of Gibbs samples we draw per iteration. Drawing each sample goes though all the data points 4 times (we have K observations). Thus, the total time complexity of all the E-steps is $O(KLT)$. In all of our experiments, LT in the order of a few thousands gives good result. Also note when drawing a sample for $\{\alpha_i\}$ or $\{\beta_j\}$ or $\{u_i\}$ or $\{v_j\}$, each element in a set across users (items) are conditionally independent and draws could be made independently. For instance, the popularity parameters for users given other profiles are independent and can be sampled simultaneously across users. This can speed up each Gibbs iteration by straightforward parallelization. Each M-step mainly consists of solving five standard linear regression problems. Any scalable linear regression software can be used to achieve efficiency.

ICM Algorithm: ICM replaces the Gibbs sampling in the E-step by finding $\{\delta_{ij}\}$ that maximizes $\Pr[\{\delta_{ij}\} \mid \{y_{ij}\}, \{X_{ij}\}, \Theta_{curr}]$. As noted in [24], this ignores uncertainty in the posterior of $\{\delta_{ij}\}$ and is inferior to Gibbs sampling. It can be easily seen that to maximize $Pr[\{\delta_{ij}\} \mid \{y_{ij}\}, \{X_{ij}\}, \Theta_{curr}]$ is to minimize the negative complete data log-likelihood Δ (as defined in Table 2) with respect to $\{\delta_{ij}\}$. We implemented the ICM algorithm using the conjugate gradient (CG) descent method with gradients given in Table 2. A popular alternative is the gradient descent method; we found CG to be better and more automatic since gradient descent requires careful tweaking of learning rate parameter for each new application.

Comparison between MCEM and ICM In our experiments, we found that MCEM is better than ICM. The following table shows an example for MovieLens data (with 1M total ratings) where we fit our *RLFM* using both algorithms. The first half (based on time) of data is used for training and the root mean square errors of both methods on the test data (last half) are reported in the table. MCEM performs better since the Gibbs sampler explore the multi-modal posterior more efficiently than ICM. Also note that as the latent dimension r increases, ICM starts to overfit, while MCEM is more resistant to over-fitting. Similar results were observed on other simulated datasets.

Latent dimension r	2	5	10	15
ICM	.9736	.9729	.9799	.9802
MCEM	.9728	.9722	.9714	.9715

3.2 Logistic Regression

We generalize our model to perform logistic regression through a variational approximation as discussed in [26]. The approximation involves creating a new Gaussian response variable T_{ij} with variance σ^2_{ij} after each EM iteration. This is given by

$$\begin{aligned} T_{ij} &= (2y_{ij} - 1)\eta_{ij}/tanh(\eta_{ij}/2) \\ \sigma^2_{ij} &= \eta_{ij}/tanh(\eta_{ij}/2) \end{aligned} \quad (7)$$

where η_{ij} is the log-odds at the current estimated value of latent factors $\{\delta_{ij}\}$.

4. DYNAMIC LATENT FACTOR MODEL

It is common to have new items and/or users appear over time in applications. Moreover, systems are often dynamic and require temporal adaptation of parameters. One can adapt latent factors to such changes by giving more weight to recent data through an

Negative complete data log-likelihood:

$$\begin{aligned}\Delta = -\log(\Pr[\{y_{ij}\},\{\delta_{ij}\} \mid \{X_{ij}\},\Theta]) = \text{Constant} &+ \tfrac{1}{2}\textstyle\sum_{ij}\left(\tfrac{1}{\sigma_{ij}^2}(y_{ij}-\alpha_i-\beta_j-x'_{ij}\boldsymbol{b}-u'_iv_j)^2+\log\sigma_{ij}^2\right)\\ &+\tfrac{1}{2}\textstyle\sum_i\left(\tfrac{1}{a_\alpha}(\alpha_i-g'_0w_i)^2+\log a_\alpha+(u_i-Gw_i)'A_u^{-1}(u_i-Gw_i)+\log(\det A_u)\right)\\ &+\tfrac{1}{2}\textstyle\sum_j\left(\tfrac{1}{a_\beta}(\beta_j-d'_0z_i)^2+\log a_\beta+(v_j-Dz_j)'A_v^{-1}(v_j-Dz_j)+\log(\det A_v)\right)\end{aligned}$$

Latent factors $\{\alpha_i\}$

Let $o_{ij} = y_{ij} - \beta_j - x'_{ij}\boldsymbol{b} - u'_iv_j$

$$\tfrac{\partial}{\partial\alpha_i}\Delta = (\tfrac{1}{a_\alpha} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{1}{\sigma_{ij}^2})\alpha_i - (\tfrac{g'_0w_i}{a_\alpha} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{o_{ij}}{\sigma_{ij}^2})$$

$$\mathit{Var}[\alpha_i|\text{Rest}] = (\tfrac{1}{a_\alpha} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{1}{\sigma_{ij}^2})^{-1}$$

$$E[\alpha_i|\text{Rest}] = \mathit{Var}[\alpha_i|\text{Rest}](\tfrac{g'_0w_i}{a_\alpha} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{o_{ij}}{\sigma_{ij}^2})$$

Latent factors $\{\beta_i\}$

Let $o_{ij} = y_{ij} - \alpha_i - x'_{ij}\boldsymbol{b} - u'_iv_j$

$$\tfrac{\partial}{\partial\beta_i}\Delta = (\tfrac{1}{a_\beta} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{1}{\sigma_{ij}^2})\beta_j - (\tfrac{d'_0z_j}{a_\beta} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{o_{ij}}{\sigma_{ij}^2})$$

$$\mathit{Var}[\beta_j|\text{Rest}] = (\tfrac{1}{a_\beta} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{1}{\sigma_{ij}^2})^{-1}$$

$$E[\beta_j|\text{Rest}] = \mathit{Var}[\beta_j|\text{Rest}](\tfrac{d'_0z_j}{a_\beta} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{o_{ij}}{\sigma_{ij}^2})$$

Latent factors $\{u_i\}$

Let $o_{ij} = y_{ij} - \alpha_i - \beta_j - x'_{ij}\boldsymbol{b}$

$$\tfrac{\partial}{\partial u_i}\Delta = (A_u^{-1} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{v_jv'_j}{\sigma_{ij}^2})u_i - (A_u^{-1}Gw_i + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{o_{ij}v_j}{\sigma_{ij}^2})$$

$$\mathit{Var}[u_i|\text{Rest}] = (A_u^{-1} + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{v_jv'_j}{\sigma_{ij}^2})^{-1}$$

$$E[u_i|\text{Rest}] = \mathit{Var}[u_i|\text{Rest}](A_u^{-1}Gw_i + \textstyle\sum_{j\in\mathcal{J}_i}\tfrac{o_{ij}v_j}{\sigma_{ij}^2})$$

Latent factors $\{v_j\}$

Let $o_{ij} = y_{ij} - \alpha_i - \beta_j - x'_{ij}\boldsymbol{b}$

$$\tfrac{\partial}{\partial v_j}\Delta = (A_v^{-1} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{u_iu'_i}{\sigma_{ij}^2})v_j - (A_v^{-1}Dz_i + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{o_{ij}u_j}{\sigma_{ij}^2})$$

$$\mathit{Var}[v_j|\text{Rest}] = (A_v^{-1} + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{u_iu'_i}{\sigma_{ij}^2})^{-1}$$

$$E[v_j|\text{Rest}] = \mathit{Var}[v_j|\text{Rest}](A_v^{-1}Dz_i + \textstyle\sum_{i\in\mathcal{I}_j}\tfrac{o_{ij}u_j}{\sigma_{ij}^2})$$

Note that $\mathcal{I}_j$ denotes the set of users who rated item j and $\mathcal{J}_i$ denotes the set of items that user i rated. For the Gaussian model, $\sigma_{ij} = \sigma$. For the Logistic model, σ_{ij} is set according to Section 3.2.

Table 2: Detail formulas for model fitting

online learning process. We describe our *batched* online learning scheme below.

Assume the hyperparameters Θ have been estimated through large amount of training data; i.e., we assume Θ is fixed and only update the latent factors $\{\delta_{ij}\}$. We also assume data is updated in a batched fashion. In general, updating model parameters for each observation is expensive; batched updates every few hours (or every few observations) are more reasonable.

At the end of our training phase, we run the Gibbs sampler for large number of iterations $(200 - 500)$ and estimate the posterior mean and covariance of the latent factor each user and item. For each batch of observations received online, the current posterior becomes the prior and combined with the data likelihood provides the new updated posterior. However, computing the posterior accurately requires both mean and variance estimates; the latter requires a large number of Gibbs samples, making the method slow in online settings. We perform an approximation that only computes the posterior mode (ICM or small number of Gibbs sample) and combines with previous factors through an exponentially-weighted moving average (EWMA) to obtain the updated factors. The EWMA ensures a stable estimate of the factors over time. In fact, we provide different EWMA weights to new and old user/items; old elements evolve much slower than the new ones in our model. After careful tweaking, we found EWMA weights of .5 an .99 on new and old elements performed well in our applicatons. For movie datasets, we updated batches of size 10K, for Yahoo! Frontpage we update a batch of 1K at a time. We denote this method by *Dyn-RLFM*.

5. EXPERIMENTS

We illustrate our methods on benchmark datasets (MovieLens and EachMovie) and on a novel recommender system that arises in the context of Yahoo! Front Page. We did not consider Netflix data since it does not provide user features; hence it is not a good example to illustrate our method. In fact, improving performance on Netflix is not the focus of this paper; our goal is to present a new methodology that provides effective predictions for large scale incomplete dyadic data by smoothly merging both cold-start (through features) and warm-start (through past interactions) situations into a single unified modeling framework. For movie datasets, we use RMSE (root mean square error) as the performance metric, this is popular in the movie recommendation domain. Note that even a 0.01 improvement in RMSE is significant (the top 40 competitors of the Netflix Prize have RMSE differences within 0.015). For Yahoo! data, we use ROC curves.

Methods: We evaluate our *RLFM* by comparing it with the following methods: *ZeroMean* and *FeatureOnly* (described in Section 2.5) that are special cases of our model; *MostPopular* is a baseline method that recommends the most popular items in the training set to users in the test set; *FilterBot* [20] is a hybrid method designed to handle cold-start collaborative filtering. We used 13 bots (based on global popularity, movie genre and the popularity in each of the 11 user groups defined based on age and gender) coupled with an item-based algorithm [15]. Several other collaborative filtering algorithms (including pure item-item similarity, user-user similarity, regression-based) were also tried. Since *FilterBot* was uniformly better among these, we only report results for it.

MovieLens data: We conducted experiments on two MovieLens datasets: MovieLens-100K, which consists of $100K$ ratings with 943 users and 1,682 movies, and MovieLens-1M, which consists of $1M$ ratings with 6,040 users and 3,706 movies (although the readme file mentions 3,900 movies). User features used include age, gender, zipcode (we used the first digit only), occupation; item features include movie genre. MovieLens-100K comes with 5 pre-specified training-testing splits for 5-fold cross validation. We report the RMSEs of *RLFM* , *ZeroMean* and *FeatureOnly* on this dataset with $r = 5$. We note that for this data, there are no new users and items in the test set; the gain obtained through *RLFM* relative to *ZeroMean* is entirely due to better regularization achieved through feature-based prior (see Figure 1 for an example).

	RLFM	*ZeroMean*	*FeatureOnly*
MovieLens-100K	0.8956	0.9064	1.0968

However, testing methods based on random splits may end up using the *future* to predict the *past*. This does not correspond to the real

	MovieLens-1M			EachMovie		
Model	30%	60%	75%	30%	60%	75%
RLFM	0.9742	0.9528	0.9363	1.281	1.214	1.193
ZeroMean	0.9862	0.9614	0.9422	1.260	1.217	1.197
FeatureOnly	1.0923	1.0914	1.0906	1.277	1.272	1.266
FilterBot	0.9821	0.9648	0.9517	1.300	1.225	1.199
MostPopular	0.9831	0.9744	0.9726	1.300	1.227	1.205
Constant Model	1.118	1.123	1.119	1.306	1.302	1.298
Dyn-*RLFM*			0.9258			1.182

Table 3: Test-set RMSE on MovieLens and EachMovie

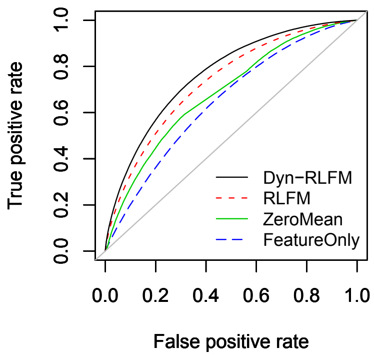


Figure 3: ROC curves of different methods on Y!FP data

world scenario where the goal is to predict ratings for dyads that occur in the future. A more realistic training/test splitting should be based on time. Surprisingly, we did not find prior work in the movie recommendation literature that performs evaluation using a strict time-based split. For MovieLens-1M, we report results on more realistic time-based splits. We set aside the last 25% of ratings as the test data and train each model on three training sets, which consist of the first 30%, 60% and 75% of the ratings, respectively. The test-set RMSEs are reported in Table 3.

First, we see that the pure feature based model *FeatureOnly* has poor performance (although it is better than a constant model, hence features are weakly predictive), in fact the item popularity model is significantly better than *FeatureOnly* . The *ZeroMean* model outperforms all the existing collaborative filtering methods we experimented with. Our *RLFM* based on factors regularized through features and item popularity significantly outperforms all other static methods. A large fraction of dyads in the test set (almost 56 percent) involve new users (most of the items are old). By adaptively estimating factors for new users starting out from a feature based prior through our dynamic *RLFM* , we obtain significant boost in predictive accuracy over the static *RLFM* model.

EachMovie data: The EachMovie dataset is similar to MovieLens but is far more noisy (RMSE for constant model is close to best models) with a large fraction of users missing one of more features. It contains 2,811,983 ratings with 72,916 users and 1,628 movies. We cleaned up this dataset by only including 2,559,107 “real” ratings (those with weights equal to 1) and then linearly scaled the ratings so that the ratings are from 0 to 5. We created training/test splits in the same way as the MovieLens case. The test-set RMSEs are reported in Table 3. Results are qualitatively similar to that of Movie-lens. The *RLFM* provides the best offline trained model, dynamic version of *RLFM* significantly outperforms all other methods.

Yahoo! Front Page (Y!FP) data: Our main motivating application is the Today Module on Yahoo! Front Page[3]. This module consists of four tabs (Featured, Entertainment, Sports and Video) and recommends four stories in the Featured tab for each user visit. Our goal is to develop algorithms that maximize the number of clicks by recommending “tasteful” stories to each user. Stories in our application have short lifetimes (usually less than one day) and, for reasons of scalability, models can only be re-trained periodically in an offline fashion. Hence, when a model trained offline is deployed, almost all stories (and large fraction of users) are new. Classical collaborative filtering algorithms that assume stories have some user ratings in the training set does not apply in this scenario. Models that can use both features and users’ past ratings are attractive. We also note that we only have a few *live* items in the system at any given time, hence online updates of latent item factors is an attractive method to obtain good performance. A dataset, which we call Y!FP, was created to evaluate performance of our recommendation algorithms. This dataset consists of 1,909,525 “binary ratings” (clicks or views without any subsequent click) given by 30,635 frequent Yahoo! users (each has at least 30 ratings in 5 months) to 4,316 stories. User features include age, gender, geolocation and browsing behavior that are inferred based on a user’s network wide activity (search, ad-clicks, page views, subscriptions, etc.) on the Yahoo! portal. In fact, a user is assigned an intensity score in a few thousand content categories based on his/her activity pattern in the recent past; we reduced these to a few hundred features by conducting a principal components analysis on the training data. Item features consists of hand-labelled categories that are assigned to a story by editors.

Results on Y!FP: Every model is significantly better than a model which predicts constant score for all examples (ROC curve for this model is a straight line). We note that almost all items that occur in the test set are new in this application. Hence for *ZeroMean* , the item factors (β_j, v_j) are zero; the only contribution is from user popularity term α_i. Thus, prediction based on user’s click propensity alone performs better than a pure feature-based model. The static *RLFM* model which exploits granular user profiles along with item features significantly outperforms the *ZeroMean* indicating presence of strong user-item interactions in the data. As for other datasets, the dynamic version of *RLFM* that estimates item profiles (β_j, v_j) of new items in an online fashion is the most granular model and has the best performance.

Discussion of Experimental Results: In previous research on recommender systems, past interaction data has been found to be more predictive. We observed similar results in our experiments. Models purely based on features are inferior to those based on user-item specific statistics. That said, the features are predictive and when combined together with past interaction data through our model provides significant improvement in accuracy. Online updating of factors is also important since most applications in practice are dynamic in nature. We believe future work in this area should evaluate algorithms using a time-based split to get a realistic estimate of an algorithm’s performance. Commonly used evaluation methods that do not use time-based split at best only provide performance indicators for persistent user-item dyads.

6. RELATED WORK

Our *RLFM* is related to research in the following areas.

Latent Factor Models: Latent factor models for dyadic data have been studied extensively in the literature, a proper review of this field is not possible here. The factors could be *discrete* [16, 7, 4] or *continuous* [22, 17], our work falls in the latter category. In fact, we are related to SVD stype matrix factorization methods that have been been successfully applied to movie recommendation data (especially in the context of the Netflix competition, see [24, 23, 8, 1] for illustrations). However, none of these methods model factors as function of features to address cold-start, nor do they model temporal variation in the factors. In fact, providing a principled modeling framework through hierarchical models that adds more flexibility to the factorization methods is the main contribution of our work.

Hierarchical Random Effects Model: Our method is related to the well studied area of hierarchical random effects models [12] in statistics. Such models provide a flexible framework to model complex processes, handle data sparseness, data imbalance and incorporate excess heterogeneity. However, parameter identifiability and computational scalability present formidable challenges requiring novel research for different applications. The work that is most closely related to ours in this area are hierarchical Bayesian models that are used in marketing for studying consumer preferences for products[5]. These models assume latent user factors are modeled through regression as we do for u_i's but there are no item factors in the model; instead, the interaction is captured $u_i^{'} z_j$, where z_j is the fixed feature vector for item j. A similar model was studied recently in [28] for recommender systems; however, their method is limited and did not consider features to model u_i.

Collaborative Filtering: Models to address both cold-start and warm-start have been studied in collaborative filtering. Several hybrid methods that combine content and collaborative filtering techniques have been proposed. In [6], the authors' present the first hybrid recommender system that computes user similarities based on content-based profiles. In [10], collaborative filtering and content-based filtering are combined linearly with weights adjusted based on absolute errors of the models. [14], [20] 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. [25] extends aspect model to combine item content and user ratings under a single probabilistic framework. Again, none of these simultaneously incorporate user features, item features, diverse response types, data imbalance, excess heterogeneity into a single modeling framework as *RLFM* .

7. DISCUSSION

We showed latent factor models with regression priors provide a flexible class of models for prediction with large scale dyadic data and significantly outperform existing state-of-the-art methods. In general, pure feature based models are weakly predictive but when combined with user-item centric models, they perform significantly better. Incorporating temporal variation substantially improves performance. Simulation based fitting algorithms like MCMC explore the posterior space more efficiently and provide better performance. We are currently investigating sequential Monte Carlo techniques to perform efficient posterior computations in an online fashion. Our code will be submitted as R software package after completing our employer's code releasing process.

8. REFERENCES

[1] KDD cup and workshop. 2007.

[2] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click rates. In *WWW*, 2009.

[3] D. Agarwal and B.-C. Chen, et al. Online models for content optimization. In *NIPS*, 2008.

[4] D. Agarwal and S. Merugu. Predictive discrete latent factor models. In *KDD*, 2007.

[5] G. Allenby, P. Rossi, and R. McCulloch. Hierarchical bayes models: A practitioner's guide. *http://ssrn.com/abstract=655541*, 2005.

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

[7] A. Banerjee and I. Dhillon, et al. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *J. of Machine Learning Research*, 2007.

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

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, 1977.

[12] A. Gelman and J. Hill. *Data Analysis using Regression and Multilevel/Hierarchical Models*. Cambridge, 2006.

[13] A. Gelman and A. Jakulin, et al. A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics*, 2008.

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

[15] J. L. Herlocker and J. A. Konstan, et al. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[16] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.

[17] D. L. Lee and S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

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

[19] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, 1998.

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

[21] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

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

[23] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML'08*.

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

[25] A. I. Schein and R. Popescul, et al. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

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

[27] R. Smith. *Bayesian and Frequentist Approaches to Parametric Predictive Inference*. Oxford University, 1999.

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