MODELING ITEM–ITEM SIMILARITIES FOR PERSONALIZED RECOMMENDATIONS ON YAHOO! FRONT PAGE

BY DEEPAK AGARWAL, LIANG ZHANG AND RAHUL MAZUMDER

Yahoo! Labs, Yahoo! Labs and Stanford University

We consider the problem of algorithmically recommending items to users on a Yahoo! front page module. Our approach is based on a novel multilevel hierarchical model that we refer to as a User Profile Model with Graphical Lasso (UPG). The UPG provides a personalized recommendation to users by simultaneously incorporating both user covariates and historical user interactions with items in a model based way. In fact, we build a per-item regression model based on a rich set of user covariates and estimate individual user affinity to items by introducing a latent random vector for each user. The vector random effects are assumed to be drawn from a prior with a precision matrix that measures residual partial associations among items. To ensure better estimates of a precision matrix in high-dimensions, the matrix elements are constrained through a Lasso penalty. Our model is fitted through a penalized-quasi likelihood procedure coupled with a scalable EM algorithm. We employ several computational strategies like multi-threading, conjugate gradients and heavily exploit problem structure to scale our computations in the E-step. For the M-step we take recourse to a scalable variant of the Graphical Lasso algorithm for covariance selection.

Through extensive experiments on a new data set obtained from Yahoo! front page and a benchmark data set from a movie recommender application, we show that our UPG model significantly improves performance compared to several state-of-the-art methods in the literature, especially those based on a bilinear random effects model (BIRE). In particular, we show that the gains of UPG are significant compared to BIRE when the number of users is large and the number of items to select from is small. For large item sets and relatively small user sets the results of UPG and BIRE are comparable. The UPG leads to faster model building and produces outputs which are interpretable.

1. Introduction. Selecting items for display on a web page to engage users is a fundamental problem in content recommendation [Agarwal et al. (2009)]. Item selection is made to maximize some utility of interest to the publisher. For instance, a news site may display articles to maximize the total number of clicks over a long time horizon. For each display, feedback obtained from user-item interaction is used to improve item selection for subsequent visits. At an abstract level, the problem of recommending items on some module of a web page can be described as follows:

- A user visits a web page. Typically, covariates like demographic information, geographic location, browse behavior and feedback from previous user visits are available for users.
- A serving scheme selects item(s) to display on a small number of slots in the module. The number of available slots are generally smaller than the number of items to choose from. Typically, item(s) selection is based on scores computed through a statistical model.
- The user interacts with items displayed on the module and provides feedback (e.g., click or no-click).
- Based on feedback, parameter estimates of statistical models are updated. The latency of update (e.g., 5 minutes, 30 minutes, 1 day) depends on the statistical model, the delay in receiving feedback from a user visit and the engineering infrastructure available.
- The process of serving items is repeated for every user visit. On portals like Yahoo!, there are hundreds of millions of daily visits.

1.1. *Background and literature.* The item recommendation problem described above is closely related to a rich literature on recommender systems and collaborative filtering [Adomavicius and Tuzhilin (2005)], a proper survey of which is beyond the scope of this paper. We describe some popular approaches that are closely related to methods proposed in this paper.

Recommender systems are algorithms that model user-item interactions to provide personalized item recommendations that will suit the user's taste. Broadly speaking, two types of methods are used in such systems—*content based* and *collaborative filtering*.[1] Content based approaches model interactions through user and item covariates. Collaborative filtering (CF), on the other hand, refers to a set of techniques that model user-item interactions based on user's past response alone, no covariates are used. Modern day recommender systems on the web tend to use a hybrid approach that combines content based and collaborative filtering.

A popular class of methods in CF are based on item–item and/or user-user similarities [Sarwar et al. (2001); Wang, de Vries and Reinders (2006)]. These are nearest-neighbor methods where the response for a user-item pair is predicted based on a local neighborhood average. In general, neighborhoods are based on similarities between items/users that are estimated through correlation measures like Pearson, cosine similarity and others. A better approach to estimate similarities has also been recently proposed in Koren (2010).

Nearest neighbor methods have been used extensively in large-scale commercial systems [Linden, Smith and York (2003); Nag (2008)]. However, item–item similarities are measured in terms of *marginal* correlations and do not adjust for

---

[1] The term "collaborative filtering" was coined by developers of the first recommender system, Tapestry [Goldberg et al. (1992)].

the effect of other items. It is not trivial to incorporate both covariates and past responses in a principled way. Also, the algorithms do not have a probabilistic interpretation, which makes it difficult to get estimates of uncertainty. We address these issues in this paper by working in a model based framework. A crucial aspect of our approach is in explicitly incorporating item–item interactions after adjusting for covariates. In fact, we model partial associations among items; it provides more flexibility compared to the classical item–item similarity approach that only exploits marginal associations. This leads to significant improvement in performance as illustrated in Section 6.

Research in CF received a boost after Netflix ran a challenge on a movie recommendation problem. The task was to use 100M ratings provided by half a million users on roughly 18K movies to minimize out-of-sample RMSE on a test set [Bell, Koren and Volinsky (2007a)]. The publicly available data set released by Netflix does not contain any user or item covariates, hence, prediction using CF is a natural approach. Several methods were tried; the winning entry was an ensemble of about 800 models. Significant improvements in accuracy were attributable to a few methods. A new class of methods that were based on SVD style matrix factorization provided excellent performance and were significantly better than classical neighborhood based approaches in CF. These are bilinear random effects models that capture user-item interactions through a multiplicative random-effects model. [See Bell, Koren and Volinsky (2007b); Bennett and Lanning (2007); Salakhutdinov and Mnih (2008a, 2008b) for more details.] Recently, these bilinear random effects models were generalized to simultaneously account for both covariates and past ratings [Agarwal and Chen (2009)]. We shall refer to this class of models as *BIRE* (bilinear-random effects model) in the rest of the paper. Methods proposed in this paper are compared to *BIRE* in the experimental section (Section 6) along with a theoretical analysis of how the approach proposed in this paper is related to *BIRE* (Section 5). Through empirical analysis, we find that our approach has significantly better predictive accuracy than *BIRE* when the number of users is large and item set to recommend from is small; for a large sized item set and a relatively small user set the performance is comparable to *BIRE*. Indeed, for a large item set we find the predictions from both *BIRE* and our model to be similar.

In terms of deployed large scale recommender systems, there is published work describing some aspects of the Amazon system based on item–item similarity [Linden, Smith and York (2003)]. In Das et al. (2007), techniques that power recommendations on Google News are described. They are primarily based on item–item similarity and Probabilistic Latent Semantic Indexing (PLSI). We compare both these methods with ours in Section 6. A large body of work in computational advertising [Broder (2008)] that recommends ads to users is also an example of recommender problems. Most existing papers in this area focus on estimating click-rates on ads by users in a given context. Early work focused mostly on covariate based regression [Chakrabarti, Agarwal and Josifovski (2008); Richardson,

Dominowska and Ragno (2007)]. Recently, Agarwal et al. (2010) describe an approach that combines covariate based regression with publisher-ad interaction through a multilevel hierarchical model. Item–item similarities in this model are estimated by exploiting a known hierarchical clustering on the item space that is obtained from domain knowledge. No such knowledge is available in our scenario, hence, the methods described in that paper do not apply. A new and emerging scientific discipline called content optimization [Agarwal et al. (2009)] that aims at recommending appropriate content for a user visit to a web page is another example of a popular recommender problem. In fact, the motivating application on Yahoo! front page we describe in this paper is an instance of content optimization.

We also note that recommender systems in general are complex and involve simultaneous optimization of several aspects, some of these are not necessarily statistical. For instance, in constructing an item pool to recommend from, human editors on a web portal like Yahoo! may discard a Lady Gaga story if it is not compatible with the Yahoo! brand, even if it is likely to click well. In computational advertising on search engines, ads that are not topically relevant to a query are removed from the item set to begin with. Nevertheless, statistical models that estimate the propensity to respond positively when an item is displayed to a user in a given context are integral to the success of most modern day recommender systems. Data obtained from such systems consist of many categorical variables like user, item, URL, IP address, search query and many more. It is typical for such categorical variables to have a large number of levels; new levels appear routinely over time and the distribution of data is heavy-tailed (a few levels are popular and a large number have small sample size). Furthermore, the modeling involves estimating interactions among several such categorical attributes; data sparseness due to high dimensionaity and imbalance in sample size is a major issue when fitting such statistical models. The modeling approach described in this paper provides a possible solution.

1.2. *Motivating application.* Our motivating application requires recommending items on a Yahoo! front page (http://www.yahoo.com) module. Figure 1 shows the location of our front page module that we shall refer to as the Personal Assistant (PA) module. The items to recommend on the PA module could consist of web apps, RSS feeds and even websites. Some examples of the PA items include Gmail,[2] Facebook,[3] Yahoo! Travel,[4] Yahoo! Games,[5] CNN[6] and so on. For instance, the Gmail web app enables login to Gmail from the Yahoo! front page. The PA module is composed of two regions—the upper part of the module (region 1)

---

[2] http://mail.google.com/.

[3] http://www.facebook.com/.

[4] http://travel.yahoo.com/.

[5] http://games.yahoo.com/.

[6] http://www.cnn.com/.

FIG. 1. *The Personal Assistant* (*PA*) *Module on Yahoo! front page. Region* 1 *displays items selected through editorial oversight and user add and remove*, *Region* 2 *has a couple of slots where items recommended through statistical methods are placed.*

consists of items that have been added by the user and the lower part (region 2) consists of items that are recommended. There are four ways a user can interact with PA items—hover, click, add and remove. "Hover" only works in "quickview" mode; Figure 2 shows an interaction with a PA item when it is hovered upon. For some PA items, a click redirects to the corresponding website. If a user likes some recommended item displayed in region 2, it can be added to region 1 by clicking on the "Add" button. A user can also remove items from region 1. To simplify the problem, we treat both "hover" and "click" as positive feedbacks of similar strength in our models (henceforth, both are referred to as "click").

Our main focus is to recommend items for slots in region 2 to maximize the overall click-rate on the PA module. At the time of writing this paper, items in region 1 were preselected through editorial oversight. A user may, however, decide to remove some of these and add new ones. Out of five slots in region 2, three

FIG. 2. *In the "quickview" mode, the pop-up window when a user hovered on one PA item "Shine"* (*http://shine.yahoo.com/*).

showed editorially selected items while the other two (the second and the third positions) display items recommended through statistical methods. Although we anticipate items being algorithmically recommended on the entire PA module in the future, for the sake of illustration we only focus on recommending items for the second and the third positions in region 2.

1.3. *Statistical challenges.* Successful deployment of large scale recommender systems like PA involves several statistical challenges. Providing personalized recommendations is important since item affinity is often user specific. However, the frequency distribution of user visits to Yahoo! front page is skewed; a small fraction of users are frequent visitors, while the remaining are *tourists* with infrequent visits. Hence, the sample size available to estimate item affinity per user is small for a large number of users. Informative covariates are often available for users based on demographic information (through registration data), geo-location

(based on IP-address) and inferred browse behavior (based on historical user activities throughout the Yahoo! network). For users with small sample sizes, estimating item affinity through covariates is an attractive strategy. The statistical challenge is to build a model that provides user-specific item affinity for heavy users but falls back on covariate based estimates in small sample size scenarios. We propose a novel multilevel hierarchical random effects model to perform such estimation.

Multilevel hierarchical random effects models are well studied in the statistics literature [see Gelman and Hill (2007) for a review]. Simple versions provide an attractive framework to perform small sample size corrections when the number of replications have large variation across groups. However, in our scenario the number of random effects far exceeds the number of data points. We have a vector of user random effects that represents latent user affinity to the entire item set, but a typical user interacts with a small number of items. This gives rise to a large missing data problem—the full latent affinity vector for each user needs to be estimated by using partial user response data. Model fitting in such scenarios presents additional challenges that we address in this paper. In particular, we constrain the random effects through a prior that models latent item dependencies using a well studied notion in the recommender systems literature—item–item similarities. But unlike existing methods in recommender problems that estimate similarities through marginal correlations, we incorporate such item–item similarities in a model based way through partial correlations.

We also discuss how to perform fast online updates of model parameters. Online parameter update makes the model adaptive and provides better recommendations in practice [Agarwal, Chen and Elango (2010)]. Also, a website like Yahoo! front page receives hundreds of millions of visits on a daily basis, hence, scalability of the model fitting procedure is an important consideration. We scale our computations by taking recourse to the penalized quasi-likelihood procedure (PQL) for model fitting [Breslow and Clayton (1993)]. Our prior involves estimating a high-dimensional covariance matrix; we discuss methods to perform such estimation in a scalable way for large problems with thousands of items.

1.4. *Overview of our proposed modeling approach.* The item pool to recommend from in PA is small (approximately 50), and it is also hard to obtain informative covariates for the items themselves. Hence, we build a per-item regression model (*IReg*) to estimate the odds of a click on an item for a given user visit. Thus, if $\mathbf{x}_u^{(t)}$ denotes the covariate vector for user u at time t, the log-odds $\theta_{uj}^{(t)}$ of a click when item j is displayed to user u at time t is modeled as $\theta_{uj}^{(t)} = \mathbf{x}_u^{(t)'} \boldsymbol{\beta}_j$. The coefficient vector $\boldsymbol{\beta}_j$ for each item j is estimated through a logistic regression with a ridge penalty on the coefficients.

Although the per-item user covariate logistic regression model IReg is satisfactory for users with a moderate to small number of visits, it may not be the best for frequent visitors where it is attractive to have a model that exploits response

from previous user visits. Thus, it is desirable to have a model that smoothly transitions from IReg to a per user model depending on the sample size. We accomplish this by augmenting our regression model with user-specific random effects. In other words, we assume $\theta_{uj}^{(t)} = \mathbf{x}_u^{(t)'} \boldsymbol{\beta}_j + \phi_{uj}^{(t)}$, where user u has a random vector $\boldsymbol{\phi}_u^{(t)} = (\phi_{u1}^{(t)}, \ldots, \phi_{uJ}^{(t)})$ (J is the number of items). The estimated log-odds is now based on both regression and user-specific random-effects. As in all random-effects models, one has to perform smoothing by imposing an appropriate prior. We assume $\boldsymbol{\phi}_u^{(t)}$ i.i.d. $\sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$, where the prior precision matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is estimated from the data. The prior precision matrix measures partial associations between item pairs after adjusting for the covariate effects. This ensures we are more likely to recommend items that are positively correlated to items that the user liked in the past. For instance, if users who like PEOPLE.com[7] also like EW.com,[8] a new user who visits PEOPLE.com will be recommended EW.com.

Most of the problems that we are interested in are under-determined—number of observations being relatively small compared to the complexity of our covariance model. This naturally calls for a regularization, for good predictive accuracy. Though there are several possibilities for the regularization of the (inverse) item covariance, for the context of this paper we resort to a structure encoding conditional independencies among items. We achieve this via a sparse inverse covariance regularization for the item–item covariance matrix [Banerjee, El Ghaoui and d'Aspremont (2008); Friedman, Hastie and Tibshirani (2008)]—popularly known as the Graphical lasso. This ensures an interpretable *sparse* graph encoding of the partial correlations, and leads to favorable computational gains (as opposed to a dense inverse covariance) and also favors predictive performances. For the inverse covariance regularization, we used our C++ implementation of a primal block coordinate method applied to the ℓ_1 penalized (negative) log-likelihood. Our algorithm [Mazumder, Agarwal and Zhang (2011)] builds on Friedman, Hastie and Tibshirani (2008) but has some important differences, and scales better (for our experiments). The main idea of the algorithm is outlined in Section 4.1.2, but we will not elaborate on it since it is beyond the scope of this paper.

The rest of the paper is organized as follows. We describe our data (with exploratory data analysis) in Section 2. Modeling details are provided in Section 3. This is followed by model fitting details in Section 4. In Section 5 we discuss the connection between the widely-used bilinear random effects model (*BIRE*) and our proposed model. Section 6 describes results of models fitted to Yahoo! front page data and benchmark MovieLens data. We end with a discussion in Section 7.

2. The PA module data. Yahoo! front page (www.yahoo.com) is one of the most visited content pages on the web and receives hundreds of millions of user

visits on a daily basis. For a significant fraction of such visits, users interact with some items on the PA module. We measure user interaction with items through activities in a "user session." A user session is a collection of visits by the user to the front page where the inter-arrival time is less than 30 minutes [Cooley et al. (1999)]. In each such user session, if a recommended item is clicked, we interpret the response to be positive (i.e., labeled as 1). In the case of no click during the session, the response is negative (i.e., labeled as 0).

The illustrative front page data set used in this paper contains around 5M binary observations generated by about 140K Yahoo! users who interact with the PA module at least once over some period spanning July to August 2009. A small random sample of sessions for users who did not interact with the PA module at all during that time period was also added. Although not using every user who visits the front page may introduce bias in our parameter estimates, the alternative approach of including all negatives introduced significant noise and led to poor results. In fact, many users visiting the front page do not interact with the PA module at all, hence, negatives in our data set lack perfect interpretation and are noisy. Such preprocessing of negatives to reduce noise is common in recommender problem studies reported in the literature. For instance, two widely used benchmark data sets, Netflix [Bennett and Lanning (2007)] and MovieLens [available at www.grouplens.org], only include users with more than 20 ratings in the training set. In most web application studies reported in the literature, it is routine to subsample negatives. We use the PA data described above to fit our model and call it the *training* data. To test the accuracy of our models through out-of-sample predictions, we created another *test* data set that contains observations from subsequent visits during some time period in August 2009. To avoid bias in testing our methods, the observations in the test set are obtained through a randomized serving scheme—a small fraction of randomly selected visits are served with items that are randomly selected from the available item set. Our randomized test set contains approximately 528K visits. Of these, about 300K visits were by users seen in the training set, the remaining are by users who either did not visit during the training period or were not included in the training set.

We have a total of 51 items in our data set, and on average each user viewed around 16 items during the training period. The sample size and the click through rate (CTR) of each item for the data sets used in this paper are shown in Figures 3 and 4, respectively. For our data, the CTR of an item is defined as the fraction of clicks per user session. Clearly, CTR is in the range [0, 1]. As evident from Figure 4, there is heterogeneity in the CTRs of items; some items like Personals,[9]

---

[9]http://personals.yahoo.com/.

Sample Size of Each Item
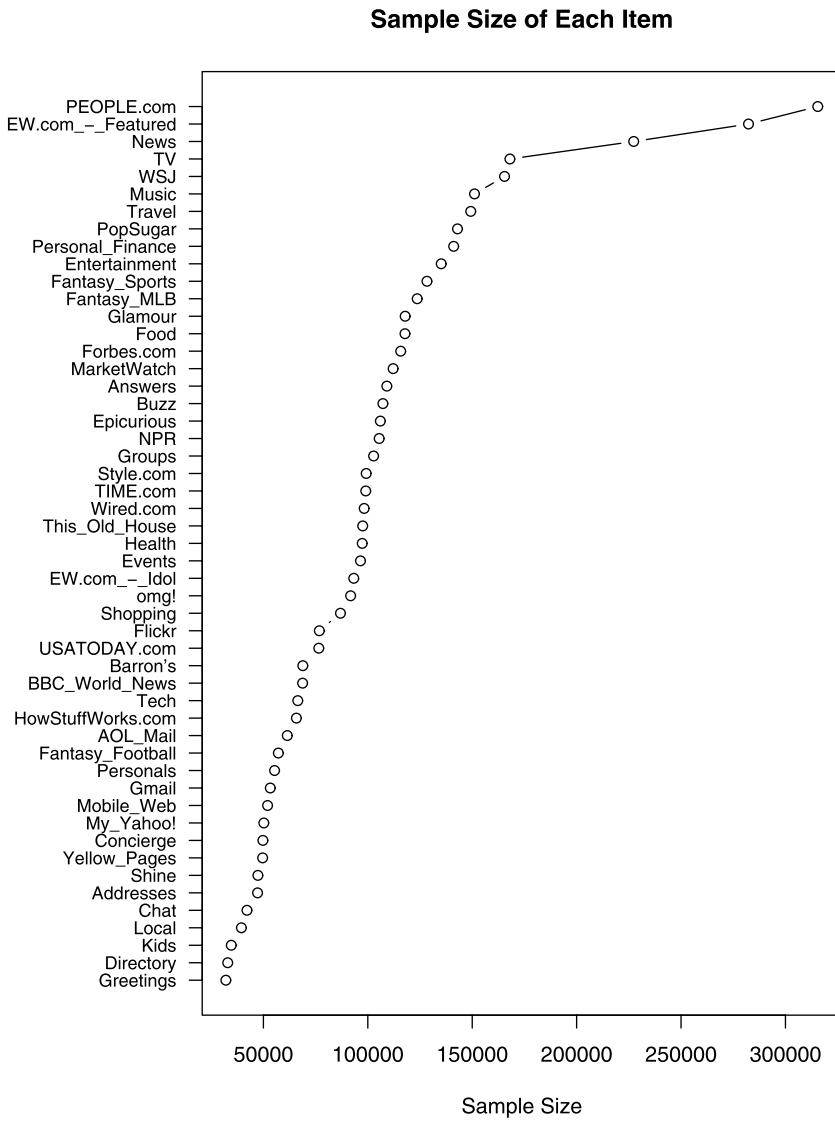
FIG. 3.   *The overall sample size of each item in the training data.*

Gmail and Music[10] have relatively high CTRs, while others like Mobile Web,[11] Addresses[12] and Local[13] have low CTRs.

---

[10]http://music.yahoo.com/.

[11]http://mobile.yahoo.com/.

[12]http://address.yahoo.com/.

[13]http://local.yahoo.com/.

CTR of Each Item
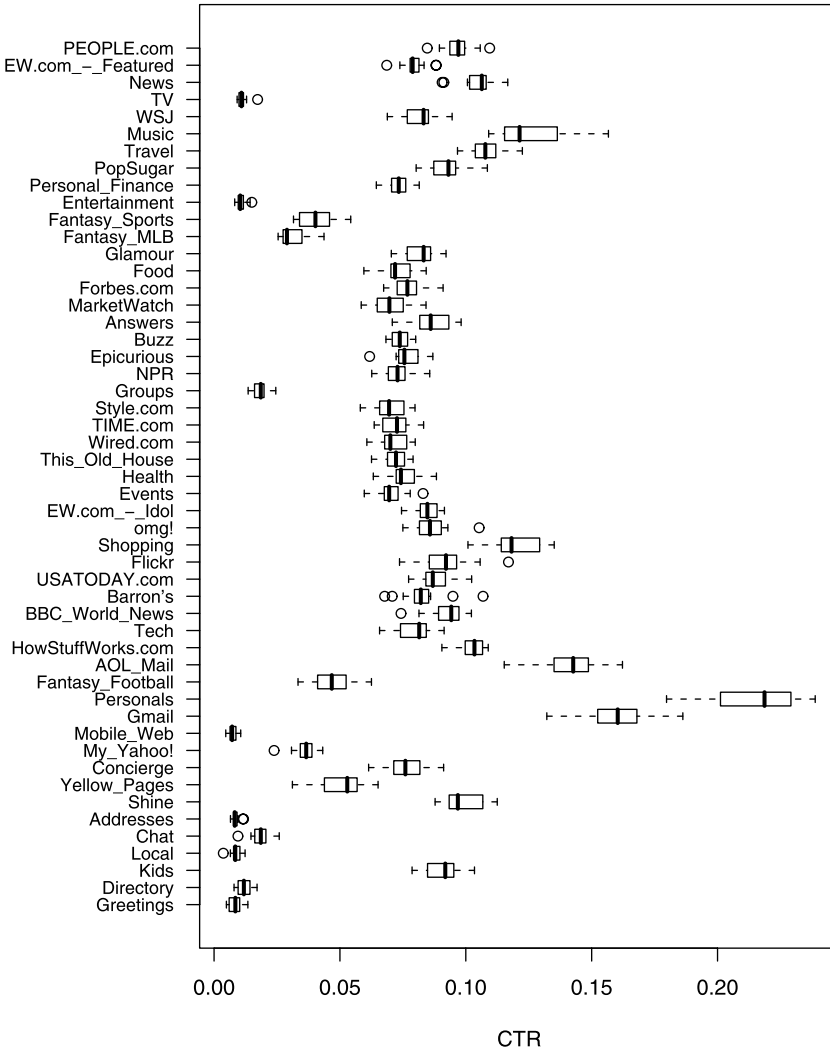
FIG. 4. *The boxplot of the click through rate* (*CTR*) *per day for each item.*

A little more than half of the users in our data set were registered when they visited Yahoo!; when these users visit the front page after "logging in," we have access to their demographic information like age, gender, occupation and so on. It is also possible to obtain a user's approximate geographic location from the IP address. When a user is not logged in, we do not have access to their demographic information. However, user activities in both logged-in and logged-out states are tracked via the browser cookie throughout the Yahoo! network; this helps us create

a browse signature for each cookie based on activity in a set of categories like Sports, Autos, Finance and so on [Chen, Pavlov and Canny (2009)]. The signature score in a given category is based on user's visits to different Yahoo! websites, what ads they clicked, what ads they viewed, search queries issued by them and other activities. Thus, for each user (logged-in or logged-out), we have a few hundred covariates describing their browse behavior on the Yahoo! network. Each covariate is a binary indicator that is turned on if a user is inferred to have activity in the corresponding category. For instance, if a user frequently visits Yahoo! Music, the corresponding browse signature covariate for Yahoo! Music will be 1; else it is 0.

Figures 5 and 6 show the degree distribution of items and users in the training data. The distribution of the number of items clicked by users has a peak at around 8, but the distribution is heavy tailed; a small fraction of items are clicked by a large fraction of users. The degree distribution clearly reveals the imbalance in our data; modeling such data is challenging.

3. Detailed description of our models. We begin this section by describing the per-item logistic regression model IReg introduced earlier. This is followed by our per-user model that assumes click rates on different items for a given user are dependent. This dependence is modeled by associating a random vector per user and assuming the user random effects are drawn from a multivariate normal prior with an unknown precision matrix. We impose sparsity on the precision matrix through a Lasso penalty on the elements of the matrix. We shall call this the "User Profile model with Graphical lasso" (*UPG*).

3.1. *Per-item regression model*: *IReg*. Whenever applicable, we drop the time suffix from our notation. For a user u interacting with an item j, we denote by y_{uj} the binary response (click/no-click) in a session. Then, $y_{uj}|p_{uj} \sim \text{Bernoulli}(p_{uj})$ and p_{uj} is modeled through a logistic regression with log-odds denoted by θ_{uj}:

$$(1) \qquad p_{uj} = \frac{1}{1 + \exp(-\theta_{uj})} \qquad \text{where } \theta_{uj} = \mathbf{x}'_u \boldsymbol{\beta}_j$$

with $\mathbf{x}_u$ denoting the known per-user covariate vector which includes age, gender and the user browse behavior information, and $\boldsymbol{\beta}_j$ is the corresponding unknown item-specific regression coefficient vector associated with item j.

The user gender covariates include three categories: missing, male and female. Similarly, age is binned into 11 categories: missing, 0–12, 13–17, 18–20, 21–24, 25–29, 30–34, 35–44, 45–54, 55–64, and older than 64. We also use 112 binary covariates describing user's browse behavior. Finally, since we consider recommending items on two slots in region 2, an extra categorical covariate was initially added to adjust for the slot effect. In general, such adjustment for presentation bias is important since everything else being equal, slots with less exposure tend to have lower click-rates. We estimated the global slot effect by looking at data

Fraction of Users Who Viewed Each Item

FIG. 5. *Degree distribution for the items in the training set.*

obtained from a randomized serving scheme. In our data set both slots provide similar exposure.

Since some items have a smaller number of observations in our data set, the maximum likelihood estimates of $\boldsymbol{\beta}_j$ may not be stable for all j. Hence, we fit our per-item regression models through a logistic ridge regression fitted using the library LIBLINEAR [Fan et al. (2008)] that uses the trust region Newton method [Lin, Weng and Keerthi (2008)]. Specifically, for each application j, the (regual-

Distribution of Number of Items Viewed Per User
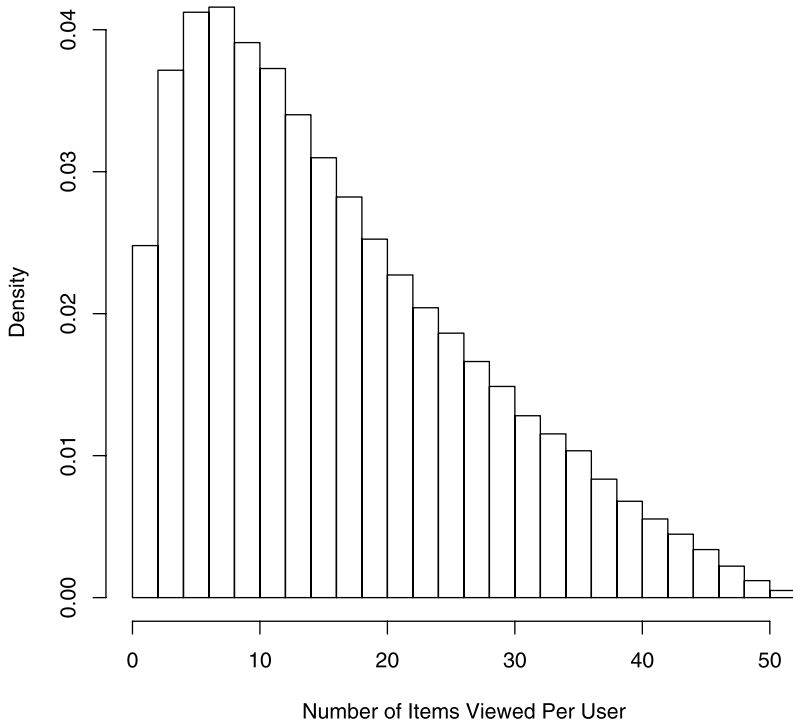
FIG. 6. *Degree distribution for the users in the training set.*

rized) coefficients are obtained by minimizing

$$(2) \qquad \frac{1}{2}\boldsymbol{\beta}'_j\boldsymbol{\beta}_j + C \sum_{u,j}(y_{uj}\log(p_{uj}) + (1 - y_{uj})\log(1 - p_{uj})),$$

where p_{uj} is given by equation 1. The tuning parameter C determines the amount of shrinkage for the regression coefficients toward zero; we select it by cross-validation.

3.2. *User profile model with gaphical lasso*: *UPG*.   The *IReg* model based on user covariates alone fails to capture variability in item interactions per-user, especially for heavy users with a large number of previous visits. For instance, there may be large variation in how users in a particular age group interact with a Facebook item on PA. For heavy users in the data, not accounting for such variation could lead to an under-specified model. To ameliorate this, we capture residual user-item interactions by augmenting the per-item regression model with additional user-specific random effects in the log-odds θ_{uj}. More specifically, we in-

troduce random effects ϕ_{uj}:

$$(3) \qquad \theta_{uj} = \mathbf{x}'_u \boldsymbol{\beta}_j + \phi_{uj}.$$

For user u, we denote the vector $\{\phi_{u1}, \ldots, \phi_{uJ}\}$ as $\boldsymbol{\phi}_u$, where J is the total number of items. This J dimensional random vector captures user u's residual latent interaction with all J items in the item set. Obviously, this is an over-parameterized model, hence, the random effects are constrained through a prior distribution. We assume the following multivariate normal prior distribution,

$$(4) \qquad \boldsymbol{\phi}_u \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\Sigma}$ is an unknown $J \times J$ covariance matrix and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix.

In the training phase (model fitting) we obtain estimates $\hat{\boldsymbol{\beta}}_j$ for each item j and the user preference vector $\hat{\boldsymbol{\phi}}_u$ for each user u. In the test period, if a user u has historical observations in training data so that $\hat{\boldsymbol{\phi}}_u$ is nonzero, then

$$(5) \qquad \hat{\theta}_{uj} = \mathbf{x}'_u \hat{\boldsymbol{\beta}}_j + \hat{\phi}_{uj}.$$

For a new user with no observations in the training period, we fall back to the *IReg* model and

$$(6) \qquad \hat{\theta}_{uj} = \mathbf{x}'_u \hat{\boldsymbol{\beta}}_j.$$

Note that for $\hat{\boldsymbol{\phi}}_u$ to be nonzero, it is enough to have partial response information on a subset of J items for user u. The random effects corresponding to the missing response items for user u are estimated by combining the likelihood of observed user response with the global prior on user random vector. The global prior that is completely specified by the precision matrix is estimated by pooling data across all users.

Although the $J \times J$ precision matrix $\boldsymbol{\Omega}$ provides an estimate of pairwise similarities between items after adjusting for the effects of user covariates and other items, the total number of parameters to estimate for J items is large ($O(J^2)$). Such estimation may get difficult for large J. It is thus desirable to impose further regularization on $\boldsymbol{\Omega}$, to avoid overfitting and improve the predictive performance. High-dimensional (inverse) covariance estimation is a challenging problem; see Banerjee, El Ghaoui and d'Aspremont (2008), Dempster (1972), Hastie, Tibshirani and Friedman (2009), Lauritzen (1996) and references therein. The form of regularization depends upon the nature of the problem, dimension of the parameter-space and computational considerations among others. A diagonal covariance, for example, is practically unrealistic since items are far from marginally independent. Regularization schemes resulting in dense and possibly unstructured precision graphs lack interpretability and will lead to increased computational burden. We propose using a regularization scheme that encourages sparsity in the precision matrix, popularly referred to as the sparse-inverse covariance selection or Graph-

ical Lasso [Banerjee, El Ghaoui and d'Aspremont (2008); Friedman, Hastie and Tibshirani (2008)].

Introducing sparsity into the precision matrix is a well-studied problem, especially in the context of graphical models with Gaussian data [Lauritzen (1996)]. In fact, for Gaussian models $\mathbf{\Omega}_{r,s} = 0$ implies ϕ_{ur} and ϕ_{us} are conditionally independent given the rest of the coordinates. Thus, sparsity in the precision matrix learns the structure of the graphical model. Due to the hierarchical model structure proposed, the estimated covariance/precision matrices have to be positive definite, rendering multiple regression or pseudo-likelihood based approaches like Besag (1975), Meinshausen and Bühlmann (2006) unsuitable for our task. The Graphical Lasso (Glasso) method estimates the precision matrix by minimizing the following regularized negative log-likelihood criterion:

$$(7) \qquad\qquad -\log\det\mathbf{\Omega} + \operatorname{tr}(\mathbf{S}\mathbf{\Omega}) + \rho\|\mathbf{\Omega}\|_1 \qquad \text{with } \mathbf{\Omega} \succ 0.$$

Here, the quantity $\|\mathbf{\Omega}\|_1$ denotes the sum of absolute values of the matrix $\mathbf{\Omega}$. The parameter ρ controls the amount of L_1 regularization and the sparsity induced on the estimated precision matrix. The optimization problem in equation (7) above is convex [Boyd and Vandenberghe (2004)]. In our model fitting procedure, the sample covariance matrix $\mathbf{S}$ is obtained in the E-step by taking expectation of the log-prior with respect to the posterior distribution of ϕ_u's assuming the hyper-parameter $\mathbf{\Omega}$ is fixed at the latest estimate in the EM procedure (see Section 4 for complete details).

To reiterate, the *UPG* model offers the following advantages over *IReg*:

- *UPG* accounts for residual variation in user preference for items after adjusting for covariates through *IReg*. Since user covariates include only coarse behavioral attributes inferred based on user activity across the Yahoo! network, it may not completely reflect user preferences for PA items.
- *UPG* exploits item–item similarity to infer user preference on items that he/she may have not been exposed to before, for instance, if users who click on item X also tend to click on Y in the historic data; a click by a new user on X would imply a click on Y with a high probability.

4. Model fitting procedure. The model fitting procedure for the *IReg* model is implemented through a trust region Newton method as described in Section 3. In this section we describe the fitting procedures for our UPG model. We fit our UPG model through a penalized quasi-likelihood (PQL) method [Breslow and Clayton (1993)]. Although other fitting methods based on MCMC and better approximation of the marginal likelihood through Gauss–Hermite quadrature [Pinheiro and Bates (2000)] are possible, we use PQL for scalability. In fact, preliminary experiments conducted using MCMC methods clearly revealed the difficulty of scaling to data sets analyzed in this paper. In particular, we tested the MCMC method in the statistical software R based on Langevin Metropolis–Hastings algorithms

[Roberts and Rosenthal (2001)] using a diagonal precision matrix as prior (random walk Metropolis proposal was slow to converge). Even for this simplified model, we need approximately 7K posterior draws of $\hat{\boldsymbol{\phi}}_u$ per user to obtain 1K posterior samples (based on MCMC diagnostics, a burn-in of 2K and thinning of 5 was adequate). Obtaining samples for all users in our training data took approximately 7 days for the PA data due to high dimensionality of random effects and a large number of users. More crucially, a sampling scheme with L_1 penalty on the elements of the precision matrix is nontrivial to construct since it is not clear what prior on the precision matrix would be equivalent to L_1 regularization. For regression problems, L_1 penalty is equivalent to a double exponential prior on the regression coefficients, but this does not hold in our case due to the additional constraint of positive definiteness. Hence, we take advantage of the Glasso mechanism for estimating $\boldsymbol{\Omega}$ and to do so, we found the PQL procedure more amenable. Thus, in this paper we only focus on PQL and leave the exploration of other fitting procedures to future work. For instance, a parametric bootstrap procedure discussed in Kuk (1995) can perhaps be modified to remove any bias incurred in estimating the precision matrix. Bootstrap is a better strategy for large scale application like ours since multiple runs can be performed in parallel.

Before describing the PQL fitting procedure, we begin with some notation. Let $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J\}$ be the set of regression coefficients, and $\boldsymbol{\Theta} = (\boldsymbol{\beta}, \boldsymbol{\Omega})$ denote the fixed effects to be estimated in the UPG model. The PQL method works as follows—at the current value $\boldsymbol{\Theta} = \boldsymbol{\Theta}_0$, we form "working residuals" Z_{uj} corresponding to the response y_{uj} (the response for user u and item j) through a Taylor series expansion. The residuals are used to obtain the posterior distribution of random-effects at $\boldsymbol{\Theta}_0$ (E-step). This is followed by an updated estimate of $\boldsymbol{\Theta}$ in the M-step. The formation of working residuals and the EM steps on the working residuals are iterated until convergence. We note that for Gaussian responses, working residuals coincide with true responses and no approximation is incurred. The complete mathematical details on the PQL procedure for fitting UPG are provided below.

4.1. *Algorithm for learning the UPG model.* If $\hat{\boldsymbol{\phi}}_{uj}$ and $\hat{\boldsymbol{\Theta}} = (\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Omega}})$ denote the current estimates of the random-effects and parameters, respectively, the working residual Z_{uj} for binary response y_{uj} is given by

$$(8) \qquad Z_{uj} = \hat{\eta}_{uj} + \frac{y_{uj} - \hat{p}_{uj}}{\hat{p}_{uj}(1 - \hat{p}_{uj})},$$

where

$$\hat{\eta}_{uj} = \mathbf{x}'_u \hat{\beta}_j + \hat{\phi}_{uj} \quad \text{and} \quad \hat{p}_{uj} = \frac{1}{1 + \exp(-\hat{\eta}_{uj})}.$$

With these we have approximately,

$$(9) \qquad Z_{uj} \sim N(\phi_{uj} + \mathbf{x}'_u \boldsymbol{\beta}_j, V_{uj}) \qquad \text{where } V_{uj} = \big(\hat{p}_{uj}(1 - \hat{p}_{uj})\big)^{-1}.$$

The updated estimates of random effects and parameters are now obtained by solving the model in equation (9) through an EM algorithm [see Breslow and Clayton (1993) for more details on PQL in general]. The EM algorithm treats the random effects as missing data [Dempster, Laird and Rubin (1977)]. Thus, the E-step involves computing the expected log-likelihood of the complete data with respect to the conditional distribution of random effects given $\hat{\boldsymbol{\Theta}}, \mathbf{Z}$.

Let $e_{uj} = Z_{uj} - \mathbf{x}'_u \hat{\boldsymbol{\beta}}_j$. Denote by n_{uj} the number of replicates where user u interacts with item j, and let N_u be the total number of users. Also, let

$$\mathbf{K}_u = \operatorname{diag}\left(\frac{n_{u1}}{V_{u1}}, \ldots, \frac{n_{uJ}}{V_{uJ}}\right), \qquad \mathbf{U}_u = \left(\sum_{r=1}^{n_{u1}} \frac{e_{u1,r}}{V_{u1}}, \ldots, \sum_{r=1}^{n_{uJ}} \frac{e_{uJ,r}}{V_{uJ}}\right),$$

where $e_{uj,r}$ represents the rth replicate for e_{uj}.

The conditional distribution of $\boldsymbol{\phi}_u$ given $\hat{\boldsymbol{\Theta}}, \mathbf{Z}$ is

$$(10) \qquad \boldsymbol{\phi}_u | \mathbf{Z}, \hat{\boldsymbol{\Theta}} \sim \operatorname{MVN}(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u),$$

where,

$$(11) \qquad \boldsymbol{\mu}_u = (\mathbf{K}_u + \hat{\boldsymbol{\Omega}})^{-1} \mathbf{U}_u, \boldsymbol{\Sigma}_u = (\mathbf{K}_u + \hat{\boldsymbol{\Omega}})^{-1},$$

where $\boldsymbol{\mu}_u = (\mathbf{K}_u + \hat{\boldsymbol{\Omega}})^{-1} \mathbf{U}_u$, and $\boldsymbol{\Sigma}_u = (\mathbf{K}_u + \hat{\boldsymbol{\Omega}})^{-1}$. Thus, $\hat{\boldsymbol{\phi}}_u = \boldsymbol{\mu}_u$ and the updated values of $\boldsymbol{\Omega}$ and $\boldsymbol{\beta}$ are obtained as follows. We use $\hat{\boldsymbol{\phi}}_u$ as offset and update the estimates of $\boldsymbol{\beta}$ through the LIBLINEAR routine. To update $\boldsymbol{\Omega}$, we make use of the following observation:

$$(12) \qquad \begin{aligned} E_{\boldsymbol{\phi}|\hat{\boldsymbol{\Omega}}, \mathbf{Z}}\left[\sum_u \log p(\boldsymbol{\phi}_u | \boldsymbol{\Omega})\right] &= -\frac{pN_u}{2} \log(2\pi) + \frac{N_u}{2} \log|\boldsymbol{\Omega}| \\ &\quad - \frac{1}{2} \sum_u \operatorname{tr}(\boldsymbol{\Omega}\boldsymbol{\Sigma}_u) + \boldsymbol{\mu}'_u \boldsymbol{\Omega}\boldsymbol{\mu}_u. \end{aligned}$$

The updated value of the precision matrix, that is, $\boldsymbol{\Omega}$, is obtained from the regularized likelihood criterion (7). In particular, for the special case with $\rho = 0$, corresponding to the unregularized maximum likelihood, the covariance estimate (assuming it exists) is given by

$$(13) \qquad \hat{\boldsymbol{\Omega}}^{-1} = \frac{\sum_u (\boldsymbol{\Sigma}_u + \boldsymbol{\mu}_u \boldsymbol{\mu}'_u)}{N_u}.$$

When $\rho \neq 0$, we treat the result of equation (13) as the sample covariance matrix $\mathbf{S}$, and use the Glasso regularization to obtain a sparse $\hat{\boldsymbol{\Omega}}$ and corresponding covariance matrix $\hat{\boldsymbol{\Sigma}}$.

4.1.1. *Large scale implementation of the E-step*. With N_u users and J items, a less sophisticated implementation of the E-step (to obtain $\mathbf{\Sigma}_u$ and $\boldsymbol{\mu}_u$ for all the users) is at least $O(N_u J^3)$, due to the expensive computation for computing $\mathbf{\Sigma}_u = (\mathbf{K}_u + \hat{\mathbf{\Omega}})^{-1}$ (11). This can make the training process prohibitively slow when J is large (e.g., a few thousands). However, we note that the matrix inversion need not be done from scratch. If $\hat{\mathbf{\Omega}}^{-1}$ is available from the previous iteration of the EM algorithm,[14] $(\mathbf{K}_u + \hat{\mathbf{\Omega}})^{-1}$ can be obtained via a low-rank update, where the low-rank is given by the number of nonzero entries of the diagonal matrix $\mathbf{K}_u$. Since in most recommender problems, a large fraction of users only interact with a small fraction of items, $\|\mathbf{K}_u\|_0$—the number of nonzero diagonal values in $\mathbf{K}_u$, is usually small. An application of the Sherman–Morrison–Woodbury formula gives

$$(14) \qquad (\hat{\mathbf{\Sigma}}^{-1} + \mathbf{K}_u)^{-1} = \hat{\mathbf{\Sigma}} - \hat{\mathbf{\Sigma}}\sqrt{\mathbf{K}_u}(\mathbf{I} + \sqrt{\mathbf{K}_u}\hat{\mathbf{\Sigma}}\sqrt{\mathbf{K}_u})^{-1}\sqrt{\mathbf{K}_u}\hat{\mathbf{\Sigma}},$$

where computing $(\mathbf{I} + \sqrt{\mathbf{K}_u}\hat{\mathbf{\Sigma}}\sqrt{\mathbf{K}_u})^{-1}$ takes only $O(\|\mathbf{K}_u\|_0^3)$. However, we show below that this can be even more efficient by exploiting the structure of the sufficient statistics.

Note that in the M-step the object of interest is actually $\sum_u \mathbf{\Sigma}_u$ instead of the individual $\mathbf{\Sigma}_u$'s. Using equation (14), we have the following simplification:

$$(15) \qquad \sum_u \mathbf{\Sigma}_u = \sum_u (\hat{\mathbf{\Sigma}}^{-1} + \mathbf{K}_u)^{-1}$$

$$= N_u \hat{\mathbf{\Sigma}} - \hat{\mathbf{\Sigma}}\left(\sum_u \sqrt{\mathbf{K}_u}(\mathbf{I} + \sqrt{\mathbf{K}_u}\hat{\mathbf{\Sigma}}\sqrt{\mathbf{K}_u})^{-1}\sqrt{\mathbf{K}_u}\right)\hat{\mathbf{\Sigma}}.$$

The computational cost in obtaining $\sum_u \mathbf{\Sigma}_u$ using (15) is now $O(\sum_u \|\mathbf{K}_u\|_0^3) + O(J^2)$. Alternatively, using decomposition (14) and then summing over all users to obtain $\sum_u \mathbf{\Sigma}_u$ has a complexity of $O(\sum_u \|\mathbf{K}_u\|_0^3) + O(N_u J^2)$, which is definitely much higher than $O(\sum_u \|\mathbf{K}_u\|_0^3) + O(J^2)$ for large values of N_u. We would like to point out that a sparse precision matrix $\hat{\mathbf{\Omega}}$ does not lead to much computational benefit in the strategies just described via (14) and (15). This is because we operate on covariance matrices which may still be dense even if the corresponding precision matrices are sparse. The sparsity of the precision matrices, however, plays a crucial role in updating the posterior mean $\boldsymbol{\mu}_u$ for each user u.

For obtaining $\boldsymbol{\mu}_u$, we need to solve the sparse symmetric linear systems $(\mathbf{K}_u + \hat{\mathbf{\Omega}})\boldsymbol{\mu}_u = \mathbf{U}_u$ for all the users. Direct factorization based methods can be quite expensive for solving these linear systems for arbitrary sparsity patterns in $\hat{\mathbf{\Omega}}$. For this purpose we employ iterative methods based on conjugate gradients

---

[14]While using the Glasso regularization in the M-step, we see that this is indeed the case, since the algorithm returns both $\hat{\mathbf{\Omega}}^{-1}$ and $\hat{\mathbf{\Omega}}$, without the cost of an explicit inversion.

[Demmel (1997); Hestenes and Stiefel (1952)]. For a specific user, this method returns approximate solutions to the linear system at the cost of a few multiplications of the matrix $(\mathbf{K}_u + \hat{\boldsymbol{\Omega}})$ with a vector, hence the complexity being *linear* (or better) in J for sufficiently sparse $\hat{\boldsymbol{\Omega}}$. For dense $\boldsymbol{\Omega}$, however, the computational cost increases to $O(J^2)$ (see Section 6.3 for computational results). Though well-chosen pre-conditioners can decrease the number of conjugate gradient iterations, all the experimental results reported in this paper are without any pre-conditioning.

Finally, since the computation of $\sum_u \boldsymbol{\Sigma}_u$ and $\sum_u \boldsymbol{\mu}_u \boldsymbol{\mu}_u'$ can easily be parallelized across users, we have used multiple-threading (e.g., 7 threads) to further expedite the computations.

4.1.2. *Computational considerations in the M-step*: *The l_1 regularized loglikelihood*.   For the l_1 regularized log-likelihood, that is, Glasso regularization in the M-step, the input covariance matrix is $\mathbf{S} = \sum_u (\boldsymbol{\Sigma}_u + \boldsymbol{\mu}_u \boldsymbol{\mu}_u')/N_u$. The Glasso implementation [Friedman, Hastie and Tibshirani (2008)] involves computation of J Lasso regressions, and a $J \times J$ Lasso regression has a worst case complexity $O(J^3)$, hence, the computational complexity of Glasso could be as high as $O(J^4)$ in the worst case. However, significant computational advances in Lasso type computations can make this computation faster, especially for large sparsity parameter ρ. Many such computational nuances have already been incorporated into the Glasso code by Friedman, Hastie and Tibshirani (2008). In particular, each Lasso is performed through a fast coordinate descent procedure that yields computational savings through residual and active set updates. In fact, in Friedman, Hastie and Tibshirani (2008) the authors empirically demonstrate that computational complexity of Glasso is $O(J^3)$ for dense problems and much faster for sparse problems with large ρ. Our algorithm [see Mazumder, Agarwal and Zhang (2011) for details] enjoys the the major computational advantages of the Glasso algorithm of Friedman, Hastie and Tibshirani (2008). As mentioned earlier, we choose to avoid the details of our algorithm, since it is beyond the scope of this current paper. We outline some of its salient features, leaving the details to a future paper. It is essentially a primal block-coordinate method, which also requires solving for every row/column a Lasso problem. In fact, a partial optimization in the Lasso problems suffices, for convergence to hold. Our algorithm maintains *both* the precision and its inverse, that is, the covariance matrix along the course of the algorithm, and is amenable to early stopping. This is actually a crucial advantage of our method, which, as our experiments suggest, it struggles with the implementation of Friedman, Hastie and Tibshirani (2008). Our algorithm has very competitive computational complexity as the Glasso for sparse problems, as our experiments suggest. It goes without saying that the algorithm of Friedman, Hastie and Tibshirani (2008), when compared with ours, gives the same models, upon convergence— since they both solve the same convex optimization criterion. However, we chose our algorithm for our experiments since it scaled favorably for our experiments, especially for the MovieLens data with approximately 4000 items.

Based on the above discussion, we conclude that large values of ρ that induce more sparsity in $\mathbf{\Omega}$ have two computational advantages: it speeds up the conjugate gradient computations in the E-step and it also accelerates the M-step of our algorithm.

4.2. *The UPG-online model.* We have observed that updating the posterior distribution of $\boldsymbol{\phi}_u$'s in an online fashion, as new observations are obtained, leads to better predictive performances. This is because for a large fraction of users, the posterior of $\boldsymbol{\phi}_u$ is based on a small number of visits in the training period. Keeping per-user posteriors updated using all prior user visits leads to posterior estimates that provide a better model fit. We observed that it is not necessary to update the precision matrix $\mathbf{\Omega}$ too quickly if the item set does not change. This is because $\mathbf{\Omega}$ is a global parameter estimated by using large amounts of training data. We now describe how the posterior of $\boldsymbol{\phi}_u$ gets updated online with the arrival of a new observation $y_{uj,\text{new}}$ for a fixed $\mathbf{\Omega}$.

Let the current posterior of $\boldsymbol{\phi}_u \sim \text{MVN}(\boldsymbol{\mu}_u, \mathbf{\Sigma}_u)$ as given in equation (10). Assuming a new response $y_{uj,\text{new}}$ updates the counters $\mathbf{K}_u$ and $\mathbf{U}_u$ to $\mathbf{K}_{u,\text{new}}$ and $\mathbf{U}_{u,\text{new}}$, the new posterior mean $\boldsymbol{\mu}_{u,\text{new}}$ is given by solving $(\mathbf{K}_{u,\text{new}} + \hat{\mathbf{\Omega}})\boldsymbol{\mu}_{u,\text{new}} = \mathbf{U}_{u,\text{new}}$ through conjugate gradient as described in Section 4.1.1. We note that for a sparse $\mathbf{\Omega}$, this computation is fast even for a large J. Also note that posterior variance need not be updated explicitly, as it is automatically updated implicitly once we obtain an updated $\mathbf{K}_u$. This is an important implementation detail for large recommender problems; we only need to store the counters in $\mathbf{K}_u$ and $\mathbf{U}_u$ for items that have been shown to users in the past. This reduces the memory requirements per user and helps with scalability in large scale systems with hundreds of millions of users. The actual implementation that scales to large number of users that visit Yahoo! front page requires state-of-the-art databases like BigTable and Hbase that can store petabytes of data across multiple commodity servers and among other things, they can support realtime serving. While it is difficult to produce the result of such an experiment for an academic paper, such implementations are becoming more common for large web applications. They may, however, involve significant hardware and other engineering costs that are typically affordable for recommender applications with a massive number of visits.

5. Comparing *BIRE* and *UPG*. In this section we discuss the connection between *BIRE* and *UPG* since the former is considered state-of-the-art in the existing recommender literature. To simplify the discussion, we will assume that our response y_{uj} is Gaussian. We also assume our response has been adjusted to remove the effects of covariates and main effects. Hence, the *UPG* model in this case is given as

$$
(16) \qquad y_{uj}|\phi_{uj} \sim N(\phi_{uj}, \sigma^2) \qquad \text{where } \boldsymbol{\phi}_u \overset{\text{i.i.d.}}{\sim} \text{MVN}(\mathbf{0}, \mathbf{\Omega}^{-1}).
$$

Conditional on the $\boldsymbol{\phi_u}$'s, the responses are independent but marginally they are not. In fact, responses by the same user u on different items are dependent. Denoting by $\mathbf{Y}_u$ the response of user u on J items, we see

$$\mathbf{Y}_u \sim \mathrm{MVN}(\mathbf{0}, \sigma^2 \mathbf{I} + \boldsymbol{\Omega}^{-1})$$

and the L_1 regularization on the precision, as we have described before helps both in computation and predictive performance.

For the *BIRE* model, we have

(17)
$$y_{uj} | \mathbf{q}_u, \mathbf{v}_j \sim N(\mathbf{q}_u' \mathbf{v}_j, \sigma^2)$$
$$\text{where } \mathbf{q}_u \overset{\text{i.i.d.}}{\sim} \mathrm{MVN}(\mathbf{0}, I) \text{ and } \mathbf{v}_j \overset{\text{i.i.d.}}{\sim} \mathrm{MVN}(\mathbf{0}, aI).$$

Here $\mathbf{q}_u$ and $\mathbf{v}_j$ are K-dimensional user and item random effects (also called factors). Marginalizing over user factors $\mathbf{q}_u$, we see

$$\mathbf{Y}_u \sim \mathrm{MVN}(\mathbf{0}, \sigma^2 \mathbf{I} + \mathbf{V}\mathbf{V}'),$$

where $\mathbf{V}$ is a $J \times K$ matrix of item factors stacked together. Thus, the *BIRE* model estimates item–item similarities through a low-rank decomposition in contrast to the UPG model that assumes a more general structure, with the sparsity regularization controlling the degrees of freedom of the estimator.

5.1. *Computational complexity of fitting BIRE.* Several model fitting strategies have been used to fit the *BIRE* model in the literature. Of these, stochastic gradient descent (SGD) and Monte Carlo EM (MCEM) have emerged as methods of choice. We note that since the posterior is multi-modal, model fitting methods influence the local optima obtained that in turn affects prediction accuracy. In particular, MCEM seems to provide the best performance in terms of out-of-sample predictive accuracy [Agarwal and Chen (2009); Salakhutdinov and Mnih (2008a)]. The E-step of the MCEM procedure computes the expected log-posterior by drawing samples from the posterior of $\{\mathbf{q}_u, \mathbf{v}_j : u = 1, \ldots, N_u; j = 1, \ldots, J\}$, conditional on the current hyper-parameter estimate a (and σ^2 for Guassian responses). We note that conditional on $\mathbf{V}$, the $J \times K$ matrix of item factors stacked together, the $\mathbf{q}_u$'s are independent. Similarly, conditioning on $\mathbf{Q}$, the $N_u \times K$ matrix of user factors, $\mathbf{v}_j$'s are independent. This provides a simple Gibbs sampling strategy to obtain samples from the posterior. The posterior samples are used to obtain an estimate of the expected log-prior with respect to the latest posterior which is then used to obtain an updated estimate of a (and σ^2 when applicable) in the M-step. In fact, it is trivial to see that the updated estimate of a (and σ^2 when applicable) in the M-step is obtained in closed form; thus, the computational complexity of the fitting procedure is mainly due to the E-step. For Guassian responses, the condi-

tional distribution of $(\mathbf{q}_u|\mathbf{V}, a, \sigma^2)$ is Guassian with mean and variance given by

$$\text{Var}(\mathbf{q}_u|\text{Rest}) = \left(I + \sum_{j \in \mathcal{N}_u} \frac{\mathbf{v}_j \mathbf{v}_j'}{\sigma^2} \right)^{-1},$$

$$E(\mathbf{q}_u|\text{Rest}) = \text{Var}(\mathbf{q}_u|\text{Rest}) \sum_{j \in \mathcal{N}_u} \frac{y_{uj} \mathbf{v}_j}{\sigma^2},$$

where $\mathcal{N}_u$ denotes the set of items user u interacted with, and $\|K_u\|_0$ denotes the size of this set. The computational complexity of computing the outer-product in $\text{Var}(\mathbf{q}_u|\text{Rest})$ is $O(\|K_u\|_0 K^2)$ and the inversion takes $O(K^3)$. Similarly, for updating the conditional distribution of $\mathbf{v}_j$'s, the computational complexity is dominated by $O(K^3)$. Recall for UPG the computational complexity for the E-step for each user u is $O(\|K_u\|_0^3 + LJ^2)$. While $\|K_u\|_0^3$ is generally smaller than K^3 in practical applications since a large fraction of users interact with a small number of items, the additional $O(LJ^2)$ term due to the conjugate gradient step adds considerable complexity to UPG for large item sets. Hence, introducing sparsity through Glasso that reduces $O(LJ^2)$ to almost linear in J helps with speeding up computation for the E-step in UPG. However, introducing such sparsity comes at the cost of performing a Glasso in the M-step. For small J like in our PA application, UPG computations are more scalable.

6. Experiments and model comparions. In this section we provide empirical analysis of our models with comparisons to others. We report performance on two different data sets—(a) The benchmark MovieLens 1M data set (described in Section 6.1) from a movie recommender system that has been studied in the literature before, and (b) The Yahoo! PA data set described earlier. For both data sets, we compare our *UPG* models with several existing methods.

6.1. *Benchmark MovieLens* 1*M data.* We conducted experiments on a benchmark MovieLens 1M data set (available at www.grouplens.org) that consists of 1M ratings provided by 6,040 users on a set of 3,706 movies. The ratings (response) r_{uj} are on a 5-point ordinal scale and the root mean squared error (RMSE) on out-of-sample predictions has been used to evaluate different modeling methods on this data before. Since reducing RMSE is the goal, statistical models assume the response (ratings) to be Gaussian for this data. We sort the training data by time stamp associated with each record and create a 75% : 25% training-test split to evaluate performance. Note that in this experiment we do not use any user or item covariates for any of the models.

6.1.1. *Methods compared on MovieLens data.* We describe several collaborative filtering methods that are compared to our approach. Some of these methods provide simple baselines, others are state-of-the-art methods used in recommender problems.

Constant—We assume $r_{uj} \sim N(\mu, \sigma^2)$ and predict every rating to be a constant μ estimated as the global mean of training data.

Item–item similarity: IIS—This is a classic model used in recommender problems. In fact, according to published sources [Linden, Smith and York (2003); Sarwar et al. (2001)], this could be one of the key technologies behind Amazon's recommendation engine. For the movie recommender problem, the rating of user u on item j is predicted as

$$(18) \qquad \bar{r}_j + \frac{\sum_{k \neq j} w_{jk}(r_{uk} - \bar{r}_k)}{\sum_{k \neq j} w_{jk}},$$

where w_{jk} measures similarity between items j and k, and $\bar{r}_k$ denotes the average rating on item k. For movie ratings data, measuring w_{jk} through Pearson's correlation is popular [Breese et al. (1998)].

Most popular: MP—We include both user and item main effects into the model. The main-effects are treated as random-effects and shrinkage is done using a normal prior. In other words, we assume the following model for ratings r_{uj}:

$$r_{uj}|(\alpha_u, \beta_j, \mu, \sigma^2) \sim N(\mu + \alpha_u + \beta_j, \sigma^2),$$

where α_u and β_j are user and item random effects, respectively, and μ is the global intercept. The priors are given by

$$\alpha_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\alpha^2); \beta_j \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\beta^2).$$

Bilinear random effects: BIRE—We augment *MP* to include a multiplicative random effects term. In other words,

$$r_{uj}|(\alpha_u, \beta_j, \mu, \mathbf{q}_u, \mathbf{v}_j, \sigma^2) \sim N(\mu + \alpha_u + \beta_j + \mathbf{q}_u'\mathbf{v}_j, \sigma^2),$$

$$\alpha_u \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\alpha^2); \qquad \beta_j \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\beta^2);$$

$$\mathbf{q}_u \overset{\text{i.i.d.}}{\sim} \text{MVN}(\mathbf{0}, I); \qquad \mathbf{v}_j \overset{\text{i.i.d.}}{\sim} \text{MVN}(\mathbf{0}, aI).$$

The inner product of the K dimensional user ($\mathbf{q}_u$'s) and item ($\mathbf{v}_j$'s) random effects, respectively, captures residual interaction. The variance components for both *MP* and *BIRE* are estimated by fitting the model through an EM algorithm. For *BIRE*, we use an MCEM algorithm [Agarwal and Chen (2009); Salakhutdinov and Mnih (2008b)]. This model has been extensively studied in the literature and has been shown to provide state-of-the-art performance compared to several other methods [Koren, Bell and Volinsky (2009)].

UPG and UPG-online—We fit the Gaussian version of our *UPG* and *UPG-online* models for comparison to the methods described above. Naturally, for this case, the PQL approximation is redundant.

TABLE 1
Test-set RMSE on MovieLens Data. The number of factors for BIRE and the sparsity parameter ρ for UPG reported obtained the best performance

Method	Constant	MP	IIS	BIRE (15 factors)	UPG ($\rho = 0.002$)	UPG-online ($\rho = 0.002$)
RMSE	1.119	0.9643	0.9584	0.9435	0.9426	0.8733

6.1.2. *Discussion of results.* In Table 1 we report the RMSE of various methods on the 25% held out test set. The *Constant* model has poor performance, and adding main effects to obtain *MP* significantly improves performance. The *IIS* model is better than *MP* but significantly worse than *BIRE*. Our *UPG* model is almost equivalent to *BIRE* in performance. For both *BIRE* and *UPG*, the hyperparameters (number of factors for *BIRE*, sparsity parameter ρ for *UPG*) have an impact on the RMSE performance. Figure 7 shows the predictive accuracy for different hyper-parameter values. In practice, these parameters are estimated by cross-validation on a tuning set. Note that for *UPG* when $\rho = 0$, the RMSE is worse than the performance with $\rho = 0.002$ (corresponds to 1.8% nonzero diagonal entries for the UPG model); this shows adding Glasso for precision matrix regularization is important for large-item-set problems (we have approximately $3.7K$ items in this case). To compare *BIRE* and *UPG*, we analyzed the residuals from both these models on the test data. Interestingly, the scatter plot revealed the residuals to be strikingly similar with a Pearson's correlation of 0.97. For this data,
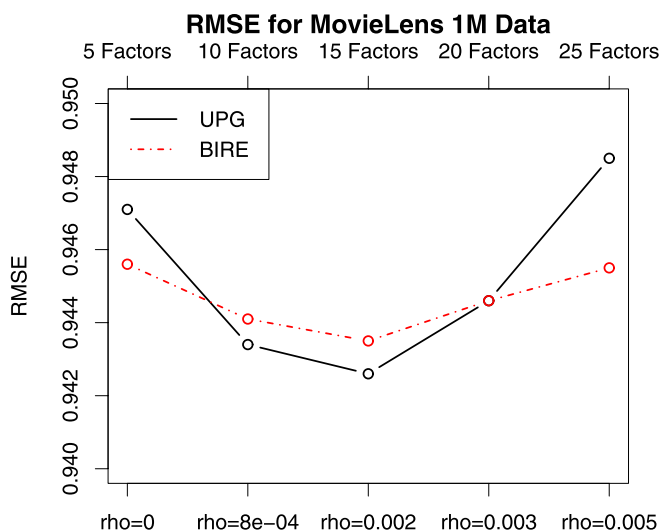

FIG. 7. *For MovieLens* 1M *data*, *the RMSE performance of UPG model with rho equal to* 0, 8e–04, 0.002, 0.003, 0.005 *compared to the BIRE model with* 5, 10, 15, 20 *and* 25 *factors*.

the more generic assumption for the precision matrix in *UPG* yields results that are similar to a low-rank approximation provided by *BIRE*.

Following a suggestion by a referee, we also compared our method with Fast Maximum Margin Matrix Factorization (*FMMMF*) [Rennie and Srebro (2005)] that predates the *BIRE* model. This approach also predicts ratings through the multiplicative random effects model as in *BIRE* but replaces the Gaussian assumption (squared error loss) on the movie ratings by a hinge loss function that incorporates the ordinal nature of the ratings. Due to nondifferentiability of the hinge loss, it is approximated with a smooth hinge loss with Gaussian priors on the user and item factors as in *BIRE*. The optimization is performed through conjugate-gradient instead of the MCEM algorithm as in *BIRE*. As suggested in that work, we compare performance in terms of mean absolute error (MAE).[15] The MAE of *FMMF* obtained after optimizing all parameters through Matlab code available publicly from the authors was 0.7997, compared to 0.7077 achieved by *UPG* ($\rho = 0.002$).

The *UPG*-online model leads to a significant improvement in accuracy since there are a large number of new users in the test set. For these users, online updates to the posterior based on their prior ratings on items help in obtaining better posterior estimates of ϕ_u's and lead to more accurate predictions of ratings. We computed the difference between mean absolute residuals obtained from *UPG*-online versus *UPG* normalized by the sample variance as done in the standard two-sample t-test. The test statistic values for users with zero observation and at least one observation in the training set were 42.03 and 11.04, respectively. Both groups had large sample sizes and the p-values from the t-test are close to zero; this is suggestive of larger improvements through *UPG*-online for users with no observations in the training set.

The practical significance of RMSE improvements on the actual movie recommender system is hard to gauge. Large scale recommender systems deployed in practice are complex and predictive models are only one aspect (albeit an important one) that determine quality. But to provide some idea, the RMSE differences of top-50 entries in the recently concluded Netflix competition were within 1% of the winning entry.

For the best ρ value which is 0.002, the *UPG* model gives a precision matrix with around 1.8% nonzero off-diagonal entries. The sparsity of the precision matrix not only improves the RMSE performance but also provides interpretable results in terms of item–item conditional similarities. We analyze the estimated partial correlations from the *UPG* model. For each pair of items i and j, we consider the partial correlation ρ_{ij} [Kendall and Stuart (1961)] between the random effects ϕ_{ui} and ϕ_{uj} defined as

$$(19) \qquad \rho_{ij} = \frac{-\Omega_{ij}}{\sqrt{\Omega_{ii}\Omega_{jj}}}.$$

---

[15]Performance of *FMMMF* was much worse than *BIRE* and *UPG* in terms of RMSE, hence not reported.

TABLE 2
Pairs of movies with top 10 *absolute values of partial correlations in the precision matrix from UPG $\rho = 0.002$*

The pair of movies	Partial correlation
The Godfather (1972) The Godfather: Part II (1974)	0.622
Grumpy Old Men (1993) Grumpier Old Men (1995)	0.474
Patriot Games (1992) Clear and Present Danger (1994)	0.448
The Wrong Trousers (1993) A Close Shave (1995)	0.443
Toy Story (1995) Toy Story 2 (1999)	0.428
Austin Powers: International Man of Mystery (1997) Austin Powers: The Spy Who Shagged Me (1999)	0.422
Star Wars: Episode IV—A New Hope (1977) Star Wars: Episode V—The Empire Strikes Back (1980)	0.417
Young Guns (1988) Young Guns II (1990)	0.395
A Hard Day's Night (1964) Help! (1965)	0.378
Lethal Weapon (1987) Lethal Weapon 2 (1989)	0.364

Intuitively, user preferences on two items i and j are associated if $|\rho_{ij}|$ is large. If $\rho_{ij} = 0$, then user random effects for items i and j are conditionally independent. For MovieLens 1M data, the top-10 movie pairs with the highest absolute values of partial correlations are shown in Table 2. Note that all pairs are sequels and have positive partial correlation values. Also, if we look for the highly related movies to a specific movie in the precision matrix, for example, Toy Story (1995), we obtain movies such as Toy Story 2 (1999), Mulan (1998), A Bug's Life (1998) and The Lion King (1994), etc., which are all cartoons.

6.2. *The Yahoo! PA data*.

6.2.1. *Methods compared.* We provide a detailed analysis of PA data with comparison to several existing methods in the recommender literature. The methods compared would differ slightly from those used for MovieLens 1M data due to the binary nature of the response. Since maximizing total clicks in a given time period is our goal, we consider a metric that provides an unbiased estimate of total clicks obtained for a set of visits. To ensure unbiasedness, we compute this

metric on a small fraction of data that is obtained by randomly selecting visits on
Yahoo! front page and serving items at random for each of them. Obtaining such
randomized data with no serving bias is unique and not typically available. We
shall prove how the metric computed on this data provides an unbiased estimate of
lift in click-rates. We begin by describing the comparative methods for this data.

Per-item regression: IReg—This is our per item logistic regression model as
described in Section 3. User affinity to items is measured only through user co-
variates, we do not consider a per-user model.

Item–Item Similarity: IIS—The similarity measure w_{jk} in equation (18) is given
by the Jaccard coefficient [Jaccard (1901)] which is the fraction of users who click
on both items j and k out of all users who click on either items j or k. The rating
r_{uj} in this case is the click-rate.

Probabilistic Latent Semantic Indexing: PLSI—PLSI was developed by Hof-
mann (1999) to model the joint distribution of user and item responses in col-
laborative filtering applications. It was recently used for a news recommendation
application by Google [Das et al. (2007)]. The main idea is to use a mixture model
where, conditional on the mixture memberships, user and items are independent.
More formally,

$$p(j|u) = \sum_{l=1}^{K} p(j|l)p(l|u),$$

where l denotes the latent membership variable and symbol p denotes appropriate
distributions. Model fitting is conducted through an EM algorithm.

BIRE, UPG, UPG-online—Other than the methods described above, we also
compare *BIRE* with *UPG* and *UPG*-online.

6.2.2. *Metrics to evaluate performance.* We begin by defining an estima-
tor that provides an unbiased estimate of expected total clicks in T visits [see
Langford, Strehl and Wortman (2008)]. To ensure unbiasedness, we collect data
through a completely randomized recommendation scheme. For a small fraction
of randomly selected visits, we display two randomly selected items in the recom-
mended slots in region 2. Due to the randomization, the set of visits that obtained
a click is a random subsample. Confining ourselves only to the clicked subsample
on position 1, we measure the number of times the model would have selected the
clicked item. In other words, we measure the concordance between the top-ranked
items selected by our statistical method and the ones that got clicked. More specif-
ically, we measure the performance of a model M through the measure defined
as

$$(20) \qquad S(M) = J \sum_{\text{visits with click}} 1(\text{item clicked} = \text{item selected by M}).$$

We show $S(M)$ is an unbiased estimator of total clicks obtained by serving
items on position 1 through model M. Let $c_{t,M(u_t)}$ denote the binary click

random variable when item $j_t = M(u_t)$ is served on visit t to user u_t by model $M(t = 1, \ldots, T)$. Then, total expected clicks $V(M) = E(\sum_{t=1}^{T} c_{t,M(u_t)}) = \sum_{t=1}^{T} \sum_{j=1}^{J} E(c_{t,j} 1(M(u_t) = j)) = T \sum_{j=1}^{J} E(c_j 1(M(u) = j))$, assuming $(u_t, c_{t,1}, \ldots, c_{t,J})$ are i.i.d. from some distribution. Now, note that $S(M) = \sum_{t=1}^{T} c_{t,j_t} \times 1(M(u_t) = j_t)$, where j_t is the item selected by the randomized serving scheme for visit t:

$$
\begin{aligned}
E(S(M)) &= J \sum_{j=1}^{J} E\left(\sum_{t \,:\, j_t = j} c_{t,j} 1\big(M(u_t) = j\big) \right) \\
(21) \qquad\qquad &= J \frac{T}{J} \sum_{j=1}^{J} E\big(c_j 1(M(u) = j)\big) \\
&= V(M).
\end{aligned}
$$

Note that the second inequality follows because $c_{t,j}$ is independent of u_t since we are evaluating on randomized data, and since u_t's are random samples, $c_{t,j} 1(M(u_t) = j)$ has the same distribution. Also, $|t : j_t = j| = T/J$ under a randomized serving scheme. To compare the click-lift of model M_1 relative to M_2, we use $100(\frac{S(M_1)}{S(M_2)} - 1)$. We report click-lift of various models under consideration relative to the random serving scheme. Proof of unbiasedness under more general scenarios is provided in Langford, Strehl and Wortman (2008); we included it for the randomized data here for easy reference.

6.2.3. *Discussion of results on PA data.* We discuss results obtained for the click-lift measure over the random model as shown in Figure 8. The boxplot for each model represents the performance of 20 bootstraps on the test data. We note that all models achieve a significant click lift compared to the random model. The models *IIS* and *PLSI* do not incorporate user covariates and are worse than all others. As expected, *IReg* is better than *IIS* and *PLSI* but *BIRE* (15 factors) does better. However, for this data set *UPG* is significantly better than *BIRE*. Furthermore, the best *UPG* model is based on a nonzero Lasso penalty. As expected, *UPG*-online provides better results than *UPG* offline.

Collaborative filtering approaches that do not incorporate covariates (*IIS* and *PLSI* in this case) perform poorly. The per-item regression *IReg* and state-of-the-art *BIRE* perform much better, but our new model that combines covariates with item–item similarity in a model based way is significantly better and achieves almost an 80% improvement in click-lift over random.

6.2.4. *Interpretability of UPG for PA.* As in MovieLens, partial correlations for the *UPG* model are interpretable. We report item pairs with top 10 absolute partial correlations in Table 3. Interestingly, all of the partial correlations shown in the
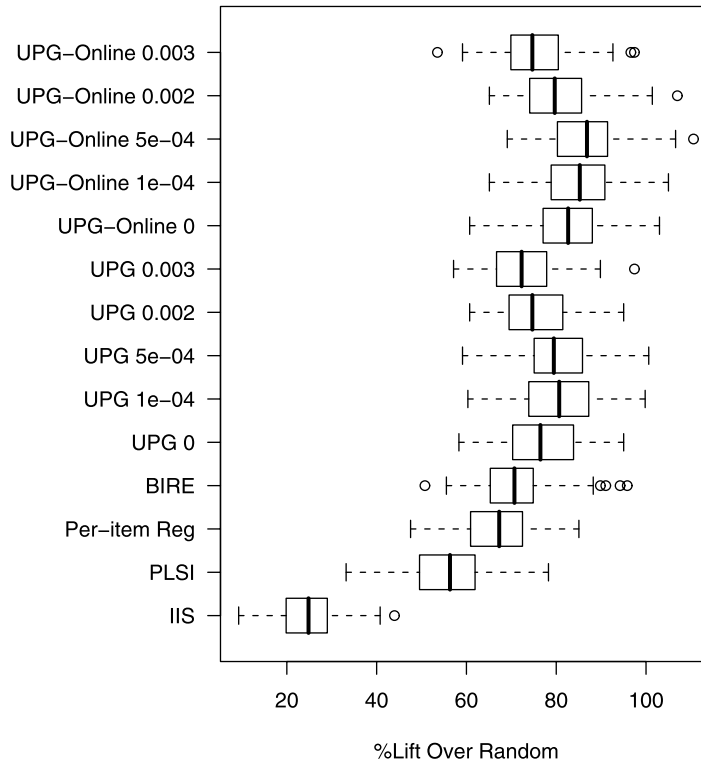
The Click–Lift Measure for PA Data

FIG. 8. *For the PA data, the click-lift measure over random model for the following models: Item–item similarity model (IIS), PLSI, Per-application logistic regression model (IReg), BIRE with 15 factors, UPG with $\rho = 0$, UPG with $\rho = $ 1e–04, UPG with $\rho = $ 5e–04, UPG with $\rho = 0.002$, UPG with $\rho = 0.003$, UPG-online with $\rho = 0$, UPG-online with $\rho = $ 1e–04, UPG-online with $\rho = $ 5e–04, UPG-online with $\rho = 0.002$, and UPG-online with $\rho = 0.003$.*

table are positive. The top two pairs are {"Fantasy Sports,"[16] "Fantasy MLB"[17]}, {"Fantasy Sports," "Fantasy Football"[18]}. Note that "Fantasy MLB" and "Fantasy Football" are two different kinds of fantasy games in "Fantasy Sports." The 3rd ranked pair {"AOL Mail,"[19] "Gmail"} are very well-known email services provided by AOL and Google, respectively. The 4th ranked pair {"PEOPLE.com," "EW.com Featured"[20]} are both news site on celebrities and entertainment.

---

[16]http://sports.yahoo.com/fantasy.

[17]http://baseball.fantasysports.yahoo.com/.

[18]http://football.fantasysports.yahoo.com/.

[19]http://webmail.aol.com.

[20]Top stories in http://www.ew.com/ew.

TABLE 3
Pairs of items with top 10 *absolute values of partial correlations in the dense precision matrix from UPG* ($\rho = 0$)

Item 1	Item 2	Partial correlation
Fantasy Sports	Fantasy MLB	0.556
Fantasy Sports	Fantasy Football	0.434
AOL Mail	Gmail	0.367
PEOPLE.com	EW.com Featured	0.265
Shopping	Personals	0.237
PEOPLE.com	PopSugar	0.224
Travel	Shopping	0.222
News	Shopping	0.208
EW.com Featured	PopSugar	0.182
News	Personals	0.181

6.3. *Timing comparison between UPG and BIRE.* For MovieLens 1M data and Yahoo! PA data, we compare the time required for training the *UPG* models and *BIRE*. Section 4.1.1 describes the computational complexity of the *UPG*.

The timing comparison between *BIRE* and *UPG* with different sparsity parameter ρ is shown in Table 4. The M-step for *UPG* includes only the timing for Glasso, the computation of the sample covariance being included in the E-step. Based on the table, we see that *UPG* takes more time than *BIRE*. Following our discussions in Section 5.1, this is because of the (major) time spent in updating $\sum_u \Sigma_u$ and μ_u, which is large due to the size of J. The timings of both E-steps and M-steps decreased with increasing sparsity level, as expected from our prior discussions. The E-step and M-step timings are not directly comparable, since the E-step uses multiple threading, where the M-step does not.

For relatively low dimensional problems (small J) but a large number of users, *UPG* could be much *faster* than *BIRE*, which is shown in the case of the Yahoo!

TABLE 4
For MovieLens 1M *data, the timing comparison* (*s*) *between BIRE and UPG* (7 *threads*) *with different sparsity parameter* (ρ) *values. BIRE uses* 15 *factors and* 100 *MCMC samples per E-step, single thread. Note that "*0*" means time is negligible because both BIRE and UPG* ($\rho = 0$) *do not use Glasso in the M-step*

	BIRE 15 factors	UPG $\rho = 0$	UPG $\rho = 8e$–04	UPG $\rho = 0.002$	UPG $\rho = 0.003$	UPG $\rho = 0.005$
E-Step	208.9 (s)	6555.0 (s)	5254.7 (s)	4049.7 (s)	3466.9 (s)	2857.0 (s)
M-Step	0	0	3502.3 (s)	2405.5 (s)	1964.3 (s)	1622.9 (s)
% Nonzero off-diagonals of Ω	NA	100%	7.3%	1.8%	0.9%	0.3%

PA data set. For 140K users and 51 items, *UPG* takes only 7 seconds per iteration using 7 threads, while *BIRE* with 15 factors and 100 samples per E-step takes 3378.1 seconds per iteration. With larger ρ and increased sparsity, the timings do improve slightly, but the improvement is not as prominent as in the MovieLens example (with larger number of items).

Therefore, for many real-life problems such as Yahoo! PA, the *UPG* models can have significant edge over *BIRE* both in terms of accuracy and computation time.

7. Discussion. Although not widely studied in the statistics literature, the problem of algorithmically recommending items to users has received a lot of attention in computer science and machine learning in the last decade. Large scale recommendation systems are mostly operated by big organizations like Amazon, Netflix, Yahoo! and Google; sharing data for academic research is difficult due to privacy concerns. A significant breakthrough was achieved when Netflix decided to run a competition and released a large amount of movie ratings data to the public in October, 2006. Since then, several methods have been published in the academic literature. Of these, the *BIRE* model described earlier has emerged to be the best, other classical methods like item–item similarity are not as accurate. Methods based on classical and successful data mining techniques like restricted Boltzmann machines (*RBM*) were also tried, but they were comparable and in some cases worse than *BIRE* [Salakhutdinov, Mnih and Hinton (2007)]. However, the errors in predictions made by *BIRE* and *RBM* were found to be uncorrelated on Netflix, hence, an ensemble approach that combined *BIRE*, *RBM* and several other methods eventually won the competition.

In this paper we proposed a new hierarchical model called *UPG* that generalizes *BIRE* by moving away from modeling item–item similarity in terms of a low rank matrix. Through extensive analysis (one benchmark data and a new data set from Yahoo! front page) we show that our approach is significantly better in one application (Yahoo! PA data) and comparable to *BIRE* on the benchmark data set. In fact, comparing residuals obtained from *UPG* and *BIRE* on the MovieLens data, we found them to be highly correlated (Pearson's correaltion coefficient was about 0.97!). Thus, even in applications where a low-rank approximation does provide a good model, the performance of our *UPG* model is comparable. In applications like PA where a low-rank approximation is not suitable, the flexibility of *UPG* leads to better accuracy. Other than accuracy, *UPG* also provides interpretable results in terms of item–item similarity. In many practical applications, item–item methods are still used since they provide interpretable results. Our *UPG* model has both features—accuracy and interpretability. We believe this makes it a promising method that could potentially lead to interesting future research in this area.

The objective of the PA recommender problem described in this paper is to maximize the total clicks on the module in some long time horizon. This is a bandit problem since there is positive utility associated with taking risk and showing items with low mean but high variance [Berry and Fristedt (1985)]. There exists

a rich literature on bandits that is related to this problem [Auer, Cesa-Bianchi and Fischer (2002); Lai and Robbins (1985); Sarkar (1991); Wang, Kulkarni and Poor (2005); Yang and Zhu (2002)]. However, bandit algorithms without dimension reduction may converge slowly in high-dimensional problems. Our *UPG* models provide a possible way to achieve such dimension reduction by exploiting correlations among response for different items; this reduces the amount of exploration required to perform personalization at the user level. Since we work with a well defined statistical model, it is possible to obtain both mean and uncertainty estimates. These can be coupled with bandit schemes like UCB [Auer, Cesa-Bianchi and Fischer (2002)] to obtain faster convergence to the best item per user. We can also use the mean estimates alone with bandit schemes that do not require explicit variance estimates from statistical models [Auer et al. (1995)]. In our PA application, we found simple ε-greedy to work well since the number of items in the pool was small and sample size available on Yahoo! front page is large. In other scenarios where sample size available per item is small due to large item pool and/or item churn, other bandit schemes like UCB may perform significantly better and can be easily coupled with the output of our *UPG* models. Constructing a bandit scheme that is optimal for our *UPG* model is more involved and is an example of correlated bandits. Some works on correlated bandits exist in the literature, but they do not directly apply to our *UPG* model [Kleinberg, Slivkins and Upfal (2008); Pandey, Chakrabarti and Agarwal (2007); Srinivas et al. (2009)]. We leave the investigation of optimal bandit schemes for *UPG* as future work. Similar comments apply to *BIRE* for which constructing optimal bandit schemes has not been investigated and is still an open problem.

Several other open issues remain to be addressed carefully. Although we were able to scale our method to approximately 4K items, scaling to larger item sets requires more research. As we pointed out in Sections 4.1.1 and 5.1, the main computational bottlenecks for *UPG* are the $O(J^2)$ conjugate gradient computations in the E-step and $O(J^3)$ computation for Glasso. Sparsity helps significantly but more research is required. For scaling up the E-step, more large scale distributed computation can help (we only used 7 threads in this paper). However, the Glasso algorithm needs more work with large J. For instance, in our current implementation we optimize each intermediate graphical-Lasso till moderate/high accuracy. Since partial optimization of the convex objective in the M-step is good enough in the early iterations of our EM procedure, this strategy deserves further investigation. We tried to modify the Glasso algorithm of Friedman, Hastie and Tibshirani (2008) along these lines, but we observed that the positive-definiteness of the returned precision matrix was not always preserved, thus our algorithm [Mazumder, Agarwal and Zhang (2011)] for this purpose was more favorable. Other than improving optimization, it is also worthwhile to scale the computations by using model approximations. One possibility is to cluster the item set into a smaller number of categories and capture item–item dependencies through category-category associations. Such a compression, though scalable, may lead to a poor fit and hurt

prediction accuracy. How to perform such clustering to achieve a good compromise between scalability and accuracy is an interesting question. Also, in several web recommender problems, there is significant item churn over time. We are currently investigating methods that generalize our methodology to update the similarity matrix for such applications. Overall, we believe our *UPG* model opens up new research avenues to study problems arising in the area of recommender problems.

REFERENCES

ADOMAVICIUS, G. and TUZHILIN, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.* **17** 734–749.

AGARWAL, D. and CHEN, B. (2009). Regression-based latent factor models. In *Proceedings of the* 15*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 19–28. ACM, New York.

AGARWAL, D., CHEN, B. C. and ELANGO, P. (2010). Fast online learning through offline initialization for time-sensitive recommendation. In *Knowledge Discovery and Data Mining Conference* 703–712. ACM, New York.

AGARWAL, D., CHEN, B., ELANGO, P., RAMAKRISHNAN, R., MOTGI, N., ROY, S. and ZACHARIAH, J. (2009). Online models for content optimization. *Adv. Neural Inf. Process. Syst.* **21** 17–24.

AGARWAL, D., AGRAWAL, R., KHANNA, R. and KOTA, N. (2010). Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *Proceedings of the* 16*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 213–222. ACM, New York.

AUER, P., CESA-BIANCHI, N. and FISCHER, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47** 235–256.

AUER, P., CESA-BIANCHI, N., FREUND, Y. and SHAPIRE, R. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *FOCS* 322–331. IEEE Computer Society, Washington, DC.

BANERJEE, O., EL GHAOUI, L. and D'ASPREMONT, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J. Mach. Learn. Res.* **9** 485–516. MR2417243

BELL, R., KOREN, Y. and VOLINSKY, C. (2007a). Chasing 1,000,000: How we won the Netflix progress prize. *ASA Statistical and Computing Graphics Newsletter* **18** 4–12.

BELL, R., KOREN, Y. and VOLINSKY, C. (2007b). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD* 95–104. ACM, New York.

BENNETT, J. and LANNING, S. (2007). The netflix prize. In *Proceedings of KDD Cup and Workshop* 2007, *San Jose, California* 3–6. ACM, New York.

BERRY, D. A. and FRISTEDT, B. (1985). *Bandit Problems*: *Sequential Allocation of Experiments*. Chapman & Hall, London. MR0813698

BESAG, J. (1975). Statistical analysis of non-lattice data. *The Statistician* **24** 179–195.

BOYD, S. and VANDENBERGHE, L. (2004). *Convex Optimization*. Cambridge Univ. Press, Cambridge. MR2061575

BREESE, J., HECKERMAN, D., KADIE, C. et al. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the* 14*th Conference on Uncertainty in Artificial Intelligence* 43–52. AUAI Press, Corvallis, OR.

BRESLOW, N. and CLAYTON, D. (1993). Approximate inference in generalized linear mixed models. *J. Amer. Statist*. *Assoc*. **88** 9–25.

BRODER, A. (2008). Computational advertising and recommender systems. In *Proceedings of the* 2008 *ACM Conference on Recommender Systems* 1–2. ACM, New York.

CHAKRABARTI, D., AGARWAL, D. and JOSIFOVSKI, V. (2008). Contextual advertising by combining relevance with click feedback. In *WWW'*08: *Proceedings of the* 17*th International Conference on World Wide Web* 417–426. ACM.

CHEN, Y., PAVLOV, D. and CANNY, J. (2009). Large-scale behavioral targeting. In *Proceedings of the* 15*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 209–218. ACM, New York.

COOLEY, R., MOBASHER, B., SRIVASTAVA, J. et al. (1999). Data preparation for mining world wide web browsing patterns. *Knowl*. *Inf*. *Syst*. **1** 5–32.

DAS, A., DATAR, M., GARG, A. and RAJARAM, S. (2007). Google news personalization: Scalable online collaborative filtering. In *Proceedings of the* 16*th International Conference on World Wide Web* 271–280. ACM, New York.

DEMMEL, J. W. (1997). *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA. MR1463942

DEMPSTER, A. (1972). Covariance selection. *Biometrics* **28** 157–175.

DEMPSTER, A. P., LAIRD, N. M. and RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *J. Roy. Statist*. *Soc*. *Ser. B* **39** 1–38. MR0501537

FAN, R., CHANG, K., HSIEH, C., WANG, X. and LIN, C. (2008). LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res*. **9** 1871–1874.

FRIEDMAN, J., HASTIE, T. and TIBSHIRANI, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9** 432–441.

GELMAN, A. and HILL, J. (2007). *Data Analysis Using Regression and Multilevel/hierarchical Models*. Cambridge Univ. Press, Cambridge.

GOLDBERG, D, NICHOLS, D., OKI, B. M. and TERRY, D. (1992). Using collaborative filtering to weave an information tapestry. *Commun*. *ACM* **35** 61–70.

HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2009). *The Elements of Statistical Learning*: *Data Mining*, *Inference*, *and Prediction*, 2nd ed. Springer, New York. MR2722294

HESTENES, M. R. and STIEFEL, E. (1952). Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards* **49** 409–436. MR0060307

HOFMANN, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the* 22*nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* 50–57. ACM, New York.

JACCARD, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull*. *Soc*. *Vaudoise Sci*. *Nat*. **37** 547–579.

KENDALL, M. and STUART, A. (1961). *The Advanced Theory of Statistics*, *Vol. I*, *II*, *III*. Griffin, London.

KLEINBERG, R., SLIVKINS, A. and UPFAL, E. (2008). Multi-armed bandits in metric spaces. In *STOC'*08 681–690. ACM, New York. MR2582691

KOREN, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data* (*TKDD*) **4** 1–24.

KOREN, Y., BELL, R. and VOLINSKY, C. (2009). Matrix factorization techniques for recommender systems. *Computer* **42** 30–37.

KUK, A. Y. C. (1995). Asymptotically unbiased estimation in generalized linear models with random effects. *J. Roy. Statist. Soc. Ser. B* **57** 395–407. MR1323346

LAI, T. L. and ROBBINS, H. (1985). Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.* **6** 4–22. MR0776826

LANGFORD, J., STREHL, A. and WORTMAN, J. (2008). Exploration scavenging. In *Proceedings of the* 25*th International Conference on Machine Learning, ICML* '08 528–535. ACM, New York.

LAURITZEN, S. L. (1996). *Graphical Models. Oxford Statistical Science Series* **17**. Oxford Univ. Press, New York. MR1419991

LIN, C.-J., WENG, R. C. and KEERTHI, S. S. (2008). Trust region Newton method for large-scale logistic regression. *J. Mach. Learn. Res.* **9** 627–650. MR2417250

LINDEN, G., SMITH, B. and YORK, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* **7** 76–80.

MAZUMDER, R., AGARWAL, D. and ZHANG, L. (2011). Block proximal point methods for large scale covariance selection. Technical report, Stanford Univ.

MEINSHAUSEN, N. and BÜHLMANN, P. (2006). High-dimensional graphs and variable selection with the lasso. *Ann. Statist.* **34** 1436–1462. MR2278363

NAG, B. (2008). Vibes: A platform-centric approach to building recommender systems. *IEEE Data Eng. Bulletin* **31** 23–31.

PANDEY, S., CHAKRABARTI, D. and AGARWAL, D. (2007). Multi-armed bandit problems with dependent arms. In *ICML* 721–728. ACM, New York.

PINHEIRO, J. C. and BATES, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York.

RENNIE, J. D. M. and SREBRO, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the* 22*nd International Conference on Machine Learning, ICML* '05 713–719. ACM, New York.

RICHARDSON, M., DOMINOWSKA, E. and RAGNO, R. (2007). Predicting clicks: Estimating the click-through rate for new ads. In *WWW'07: Proceedings of the* 16*th International Conference on World Wide Web* 521–530. ACM, New York.

ROBERTS, G. O. and ROSENTHAL, J. S. (2001). Optimal scaling for various Metropolis–Hastings algorithms. *Statist. Sci.* **16** 351–367. MR1888450

SALAKHUTDINOV, R., MNIH, A. and HINTON, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the* 24*th International Conference on Machine Learning, ICML* '07 791–798. ACM, New York.

SALAKHUTDINOV, R. and MNIH, A. (2008a). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the* 25*th International Conference on Machine Learning* 880–887. ACM, New York.

SALAKHUTDINOV, R. and MNIH, A. (2008b). Probabilistic matrix factorization. *Adv. Neural Inf. Process. Syst.* **20** 1257–1264.

SARKAR, J. (1991). One-armed bandit problems with covariates. *Ann. Statist.* **19** 1978–2002. MR1135160

SARWAR, B., KARYPIS, G., KONSTAN, J. and REIDL, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW* '01: *Proceedings of the* 10*th International Conference on World Wide Web* 285–295. ACM, New York.

SRINIVAS, N., KRAUSE, A., KAKADE, S. M. and SEEGER, M. (2009). Gaussian process bandits without regret: An experimental design approach. *Computing Research Repository—CoRR*. Available at arXiv:0912.3995v1.

WANG, J., DE VRIES, A. P. and REINDERS, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR* '06: *Proceedings of the* 29*th*

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval 501–508. ACM, New York.

WANG, C.-C., KULKARNI, S. R. and POOR, H. V. (2005). Bandit problems with side observations. *IEEE Trans. Automat. Control* **50** 338–355. MR2123095

YANG, Y. and ZHU, D. (2002). Randomized allocation with nonparametric estimation for a multi-armed bandit problem with covariates. *Ann. Statist.* **30** 100–121. MR1892657

D. AGARWAL
L. ZHANG
4401 GREAT AMERICA PKWY
SANTA CLARA, CALIFORNIA 95054
USA
E-MAIL: dagarwal@yahoo-inc.com
          liangzha@yahoo-inc.com

R. MAZUMDER
DEPARTMENT OF STATISTICS
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305
USA
E-MAIL: rahulm@stanford.edu