Active Learning for Online Bayesian Matrix Factorization

Jorge Silva
Dept. of Electrical and Computer Engineering
Duke University
Durham, NC
jg.silva@duke.edu

Lawrence Carin
Dept. of Electrical and Computer Engineering
Duke University
Durham, NC
lcarin@duke.edu

ABSTRACT

The problem of large-scale online matrix completion is addressed via a Bayesian approach. The proposed method learns a factor analysis (FA) model for large matrices, based on a small number of observed matrix elements, and leverages the statistical model to actively select which new matrix entries/observations would be most informative if they could be acquired, to improve the model; the model inference and active learning are performed in an online setting. In the context of online learning, a greedy, fast and provably near-optimal algorithm is employed to sequentially maximize the mutual information between past and future observations, taking advantage of submodularity properties. Additionally, a simpler procedure, which directly uses the posterior parameters learned by the Bayesian approach, is shown to achieve slightly lower estimation quality, with far less computational effort. Inference is performed using a computationally efficient online variational Bayes (VB) procedure. Competitive results are obtained in a very large collaborative filtering problem, namely the Yahoo! Music ratings dataset.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications-Data mining

Keywords

Online learning, matrix factorization

1. INTRODUCTION

There has been growing interest in the problem of matrix completion, *i.e.*, estimating unknown elements in matrices based on a subset of observed entries (with the observed entries usually assumed observed at random). Concretely, assume that $\mathbf{X} \in \mathbb{R}^{N_u \times N_v}$ is a matrix, for which we know only the values of the elements x_{ij} at locations given by the set $\Omega = \{(i,j) : x_{ij} \text{ is observed}\}$. In a collaborative filtering/recommender system setting, N_u is often the number of users, while N_v is the number of items.

The matrix $\mathbf{X}$ can be very large; also, the number of observations can be very small relative to $N_u \times N_v$, but large in absolute terms, which poses computational and statistical challenges. For example, in the Netflix challenge [2] there are $N_u = 480189$ users, $N_v = 17770$ movies and the number of observed ratings is $|\Omega| = 100480507$, which corresponds to less than 1.18 percent of $\mathbf{X}$. More recently, the even larger Yahoo! Music dataset [4] has been released. It contains $|\Omega| = 262810175$ ratings (integers between 0 and 100) pertaining to $N_u = 1000990$ users and $N_v = 624961$ items (music tracks, albums, artists and genres).

For many such real problems, the matrices of interest can be assumed to be approximately low-rank, in which case it is appropriate to consider a matrix factorization model of the form

$$\mathbf{X} \approx \mathbf{U}^T \mathbf{V}, \tag{1}$$

with $\mathbf{U} \in \mathbb{R}^{K \times N_u}$, $\mathbf{V} \in \mathbb{R}^{K \times N_v}$ and $K \ll N_u, N_v$. Recent advances in matrix completion theory [3] provide bounds on the number of observations needed for successfully recovering low-rank matrices using convex optimization (and assuming the observed entries are selected at random). However, existing convex optimization algorithms are incapable of handling the very large matrix sizes typical of real collaborative filtering problems.

Hence, the most popular approaches for large-scale problems are based on solving (1), or some variant thereof, for $\mathbf{U}, \mathbf{V}$ through minimization of the ℓ_2 norm of the residual computed for the known locations Ω (see, *e.g.*, [7]). Regularization is often employed, typically using the ℓ_2 or ℓ_∞ norm [9] of the parameters (these regularizers are imposed on the columns of $\mathbf{U}$ and $\mathbf{V}$; the ℓ_1 or ℓ_0 sparseness regularizers characteristic of convex optimization solutions [3] are not used here, because we explicitly set the rank to K, with $K \ll N_u, N_v$). Due to memory and speed considerations, the minimization is often performed through stochastic gradient descent. Stochastic gradient descent is an online learning method, in which the training examples are presented sequentially, obviating the need to store the entire dataset in memory. Typically, the order of presentation is random. Stochastic gradient descent has also been used successfully in other large-scale problems such as image inpainting [11] and topic modeling [5]. It enjoys local convergence guarantees, as shown in [17].

KDD 2012 Beijing, China

Alternative methods for solving (1) include Bayesian probabilistic matrix factorization [16], which achieves state-of-the-art performance on the Netflix dataset. However, since this is a batch method, scalability to even larger problems such as the Yahoo! Music dataset is less straightforward than with stochastic gradient descent. Additionally, [16] uses Markov chain Monte Carlo inference, for which it is harder to assess convergence.

In our work, we propose a statistical factor analysis (FA) model that extends the basic low-rank factorization shown in (1), and we develop an associated variational Bayesian (VB) [1] online inference algorithm, which incorporates the scalability of stochastic gradient descent while providing approximate posterior estimates of all model parameters. Other VB methods that have been applied to collaborative filtering, such as [10, 14], are based on batch learning. Our method is an online extension of the batch VB principal component analysis (PCA) approach proposed in [14, 15, 12].

Another contribution of our paper is that we leverage the statistical model to *actively* select the best subset of observations at each step of the online learning process, instead of assuming that the training examples follow a random order of presentation as in stochastic gradient descent. This active learning procedure is motivated by the fact that each measurement x_{ij} is often obtained sequentially and requires user interaction, which may be expensive and time-consuming. Therefore, by prioritizing the most informative users and items, we can potentially guide the data acquisition process in a more efficient way. In general, the problem of finding the m best locations of $\mathbf{X}$ to measure is NP-complete. We avoid this difficulty by developing two types of greedy algorithms: (i) a near-optimal method based on [8], relying on the submodularity properties of mutual information; (ii) a simpler algorithm based on the estimated variance of the user and item factors returned by the VB inference. A non-Bayesian active algorithm is also used for comparison.

The remainder of the paper is organized as follows. Section 2 describes our proposed factor analysis model, as well as the associated VB inference. This section also briefly discusses how to incorporate time information in the model. The active learning algorithms are presented in Section 3, which also contains an explanation of the submodularity property. Experimental results with online VB and active learning on the Yahoo! Music dataset are shown in Section 4. Concluding remarks are presented in Section 5.

2. FACTOR ANALYSIS MODEL

The matrix factorization problem expressed in (1) can be cast as factor analysis (FA) with $x_{ij} = \boldsymbol{u}_i^T \boldsymbol{v}_j + \epsilon_{ij}$, where ϵ_{ij} is a residual term, assumed small, $\boldsymbol{u}_i \in \mathbb{R}^K$ is the ith column of $\mathbf{U}$, and $\boldsymbol{v}_j \in \mathbb{R}^K$ is the jth column of $\mathbf{V}$. Previous work on collaborative filtering [7, 6] shows that it is advantageous to also incorporate user and item biases, leading to the complete FA model

$$x_{ij} = \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j + \epsilon_{ij}. \tag{2}$$

The scalars α_i and β_j allow the model to adapt to phenomena such as anomalous user behavior or different item popularities. A particular item j may be popular among all users, and therefore this item does not provide as much information about individual preferences, which are captured in the feature vectors $\boldsymbol{u}_i$; if this is the case for item j, the model should infer the associated β_j as being large. Likewise, a given individual i may rate all items as being poor/good, and this individual therefore does not provide as useful information about the features of the items $\boldsymbol{v}_j$; in this case α_i would be correspondingly small/large. In the statistical literature, these inferred biases are often called random effects.

Defining the independent priors

$$p(\alpha_i) = \mathcal{N}(0, \tau_\alpha^2) \tag{3}$$

$$p(\beta_j) = \mathcal{N}(0, \tau_\beta^2) \tag{4}$$

$$p(\mathbf{U}) = \prod_{k=1}^{K} \prod_{i=1}^{N_U} \mathcal{N}(u_{ki}; 0, 1) \tag{5}$$

$$p(\mathbf{V}) = \prod_{k=1}^{K} \prod_{j=1}^{N_V} \mathcal{N}(v_{kj}; 0, \rho_k^2) \tag{6}$$

$$p(\rho_k^2) = \text{InvGamma}(\kappa_1, \kappa_2) \tag{7}$$

$$p(\epsilon_{ij}) = \mathcal{N}(0, \sigma_\epsilon^2) \tag{8}$$

with hyperparameters $\sigma_\epsilon^2, \tau_\alpha^2, \tau_\beta^2, \kappa_1, \kappa_2$, and the likelihood

$$p(x_{ij}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}) = \mathcal{N}(x_{ij}; \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j, \sigma_\epsilon^2) \tag{9}$$

completes the model specification. In the above expressions, InvGamma denotes the inverse gamma distribution, and the notation u_{ki}, v_{kj} refers to the (k,i)th and (k,j)th elements of $\mathbf{U}$ and $\mathbf{V}$, respectively. For simplicity, we set the value of σ_ϵ, although it would also be possible to define a corresponding inverse gamma prior. Note that we choose a unit-variance prior for $\boldsymbol{u}_i$; otherwise, the product of the variances of $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ would not be identifiable.

Our goal is to estimate posteriors for the model parameters $\boldsymbol{\lambda} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}\}$ given the observed data $\mathbf{D} = \{x_{ij} : (i,j) \in \Omega\}$ at locations Ω.

2.1 Variational Bayes inference

The posterior distribution of the model parameters satisfies

$$\begin{aligned} p(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}|\mathbf{D}) \propto & p(\mathbf{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}) \\ & p(\boldsymbol{\alpha}) p(\boldsymbol{\beta}) p(\mathbf{U}) p(\mathbf{V}) p(\boldsymbol{\rho}), \end{aligned} \tag{10}$$

while the negative logarithm of the posterior is, up to constant terms, given by

$$\sum_{ij} \frac{1}{2\sigma_x^2}(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)^2 + \sum_{k=1}^{K} \frac{1}{2}(\|\boldsymbol{u}_k\|_2^2 + \frac{1}{\rho_k^2}\|\boldsymbol{v}_k\|_2^2), \tag{11}$$

where $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ are the kth rows of $\mathbf{U}$ and $\mathbf{V}$, respectively. Hence, the statistical model is closely related to optimization-based approaches with ℓ_2 regularization on the model components [7]. An important advantage of the statistical model is that it does not require cross-validation for regularization terms (Lagrange multipliers in optimization-based methods). Moreover, it yields an approximate posterior on all model parameters, instead of maximum *a posteriori* (MAP) point estimates.

As the true posterior $p(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}|\mathbf{D})$ is not tractable, we employ a variational approximation [1]. In general, assume that $\boldsymbol{\lambda}$ denotes parameters to be inferred, and $\mathbf{D}$ represents observed data. The likelihood of the data is $p(\mathbf{D}|\boldsymbol{\lambda})$ and the prior on the parameters is $p(\boldsymbol{\lambda})$. It follows that the posterior on the model parameters is

$$p(\boldsymbol{\lambda}|\mathbf{D}) = \frac{p(\mathbf{D}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})}{\int d\boldsymbol{\lambda} p(\mathbf{D}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})} \tag{12}$$

Let $q(\boldsymbol{\lambda}; \boldsymbol{\theta})$ be a parametric distribution with parameters $\boldsymbol{\theta}$, and consider the variational expression

$$\begin{aligned} F(\boldsymbol{\theta}) &= \int d\boldsymbol{\lambda} q(\boldsymbol{\lambda}; \boldsymbol{\theta}) \log \frac{q(\boldsymbol{\lambda}; \boldsymbol{\theta})}{p(\mathbf{D}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})} \\ &= D_{\mathrm{KL}}[q(\boldsymbol{\lambda}; \boldsymbol{\theta}) \| p(\boldsymbol{\lambda}|\mathbf{D})] - \log p(\mathbf{D}). \end{aligned} \tag{13}$$

Minimization of $F(\boldsymbol{\theta})$ in (13) with respect to $\boldsymbol{\theta}$ corresponds to minimization of the Kullback-Leibler divergence between $q(\boldsymbol{\lambda}; \boldsymbol{\theta})$ and the true posterior $p(\boldsymbol{\lambda}|\mathbf{D})$.

For the parametric distribution, we use a fully factorized form $q(\boldsymbol{\lambda}; \boldsymbol{\theta}) = \prod_l q(\lambda_l; \theta_l)$. If $q(\lambda_i; \theta_i)$ and the prior for λ_i are selected in the conjugate-exponential family, then we obtain closed-form expressions for the θ_i, based on gradient descent of $F(\theta_i)$ [1]. This procedure is guaranteed to converge to a local optimum, and it extends MAP estimation of $\boldsymbol{\theta}$, yielding an approximate posterior.

The term *variational* comes from calculus of variations; we choose a class $\mathcal{Q}$ such that $q(\boldsymbol{\lambda}; \boldsymbol{\theta}) \in \mathcal{Q}$ and then minimize a functional over $\mathcal{Q}$. As discussed above, minimization of the functional $F(\boldsymbol{\theta})$ is equivalent to minimization of the Kullback-Leibler (KL) divergence between q and the true $p(\lambda|\mathbf{D})$. We choose $\mathcal{Q}$ to be fully factorized distributions; for the problem at hand, we write

$$\begin{aligned} q(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}; \boldsymbol{\theta}) = &\prod_i q(\alpha_i) \prod_j q(\beta_j) \prod_{k,i} q(u_{ki}) \prod_{k,j} q(v_{kj}) \\ &\prod_k q(\rho_k), \end{aligned} \tag{14}$$

where

$$q(u_{ik}) = \mathcal{N}(\mu_{u_{ki}}, \sigma^2_{u_{ki}}) \tag{15}$$

$$q(v_{kj}) = \mathcal{N}(\mu_{v_{kj}}, \sigma^2_{v_{kj}}) \tag{16}$$

$$q(\alpha_i) = \mathcal{N}(\mu_{\alpha_i}, \sigma^2_{\alpha_i}) \tag{17}$$

$$q(\beta_j) = \mathcal{N}(\mu_{\beta_j}, \sigma^2_{\beta_j}) \tag{18}$$

$$q(\rho_k^2) = \mathrm{InvGamma}(\xi_{1k}, \xi_{2k}). \tag{19}$$

The posterior parameters are $\boldsymbol{\theta} = \{\{\mu_{\alpha_i}\}, \{\mu_{\beta_j}\}, \{\sigma_{\alpha_i}\}, \{\sigma_{\beta_j}\}, \{\rho_k^2\}, \{\xi_{1k}\}, \{\xi_{2k}\}, \{\mu_{u_{ki}}\}, \{\mu_{v_{kj}}\}, \{\sigma^2_{u_{ki}}\}, \{\sigma^2_{v_{kj}}\}\}$.

Defining the cost function $C_{KL} \triangleq F(\boldsymbol{\theta})$, we build on [14], where the gradients w.r.t. the components of $\boldsymbol{\theta}$ are derived. We depart from [14] in the following way: the dataset $\mathbf{D}$ is partitioned into smaller subsets $\mathcal{S}^{(t)}$, called *mini-batches*, which change for each iteration t. The standard stochastic gradient approach [9] is to draw each mini-batch $\mathcal{S}^{(t)}$ uniformly at random from the (fixed) training set $\mathbf{D}$. Then, in the sums over i and j below, only the indices in the current $\mathcal{S}^{(t)}$ are considered. All mini-batches have the same size m, which is chosen to yield the best trade-off between computational speed and robustness to local minima.

Example components of the gradient so computed are

$$\begin{aligned} &\frac{\partial C_{KL}}{\partial \mu_{u_{ki}}} = \mu_{u_{ki}} + \\ &\sum_{j:(i,j)\in\mathcal{S}} \frac{-(x_{ij} - \alpha_i - \beta_j - \sum_{m=1}^K \mu_{u_{mi}}\mu_{v_{mj}})\mu_{v_{kj}} + \mu_{u_{ki}}\sigma^2_{v_{kj}}}{\sigma^2_\epsilon} \end{aligned} \tag{20}$$

$$\begin{aligned} &\frac{\partial C_{KL}}{\partial \mu_{v_{kj}}} = \frac{\mu_{v_{kj}}}{\rho_k^2} + \\ &\sum_{i:(i,j)\in\mathcal{S}} \frac{-(x_{ij} - \alpha_i - \beta_j - \sum_{m=1}^K \mu_{u_{mi}}\mu_{v_{mj}})\mu_{u_{ki}} + \mu_{v_{kj}}\sigma^2_{u_{ki}}}{\sigma^2_\epsilon} \end{aligned} \tag{21}$$

$$\frac{\partial C_{KL}}{\partial \mu_{\alpha_i}} = \frac{\mu_{\alpha_i}}{\tau_\alpha^2} + \sum_{j:(i,j)\in\mathcal{S}} \frac{-(x_{ij} - \alpha_i - \beta_j - \sum_{m=1}^K \mu_{mi}\mu_{mj})}{\sigma^2_\epsilon} \tag{22}$$

$$\frac{\partial C_{KL}}{\partial \mu_{\beta_j}} = \frac{\mu_{\beta_j}}{\tau_\beta^2} + \sum_{i:(i,j)\in\mathcal{S}} \frac{-(x_{ij} - \alpha_i - \beta_j - \sum_{m=1}^K \mu_{mi}\mu_{mj})}{\sigma^2_\epsilon}. \tag{23}$$

Once we have the gradients, it is then possible to sequentially update $\mu_{u_{ki}}, \mu_{v_{kj}}, \mu_{\alpha_i}$ and μ_{β_j} according to

$$\mu^{(t)}_{u_{ki}} \leftarrow \mu^{(t-1)}_{u_{ki}} + \eta^{(t)} \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{\partial C_{KL}}{\partial \mu^{(t)}_{u_{ki}}} \tag{24}$$

$$\mu^{(t)}_{v_{kj}} \leftarrow \mu^{(t-1)}_{v_{kj}} + \eta^{(t)} \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{\partial C_{KL}}{\partial \mu^{(t)}_{v_{kj}}} \tag{25}$$

$$\mu^{(t)}_{\alpha_i} \leftarrow \mu^{(t-1)}_{\alpha_i} + \eta^{(t)} \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{\partial C_{KL}}{\partial \mu^{(t)}_{\alpha_i}} \tag{26}$$

$$\mu^{(t)}_{\beta_j} \leftarrow \mu^{(t-1)}_{\beta_j} + \eta^{(t)} \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{\partial C_{KL}}{\partial \mu^{(t)}_{\beta_j}} \tag{27}$$

where $\eta^{(t)}$ is the learning step for the gradient descent procedure. The learning step follows the schedule $\eta^{(t)} = \tau\eta^{(t-1)}$, where $\tau \in (0.5, 1]$ controls the rate of decay of the contribution of old mini-batches.

The variances can be updated according to

$$\sigma^2_{u_{ki}} \leftarrow (1-\eta^{(t)})\sigma^2_{u_{ki}} + \eta^{(t)}\left(1 + \sum_{j:(i,j)\in\mathcal{S}^{(t)}} \frac{\mu^2_{v_{kj}} + \sigma^2_{v_{kj}}}{\sigma^2_\epsilon}\right)^{-1} \quad (28)$$

$$\sigma^2_{v_{kj}} \leftarrow (1-\eta^{(t)})\sigma^2_{v_{kj}} + \eta^{(t)}\left(\frac{1}{\rho^2_k} + \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{\mu^2_{u_{ki}} + \sigma^2_{u_{ki}}}{\sigma^2_\epsilon}\right)^{-1} \quad (29)$$

$$\sigma^2_{\alpha_i} \leftarrow (1-\eta^{(t)})\sigma^2_{\alpha_i} + \eta^{(t)}\left(\frac{1}{\tau_\alpha} + \sum_{j:(i,j)\in\mathcal{S}^{(t)}} \frac{1}{\sigma^2_\epsilon}\right)^{-1} \quad (30)$$

$$\sigma^2_{\beta_j} \leftarrow (1-\eta^{(t)})\sigma^2_{\beta_j} + \eta^{(t)}\left(\frac{1}{\tau_\beta} + \sum_{i:(i,j)\in\mathcal{S}^{(t)}} \frac{1}{\sigma^2_\epsilon}\right)^{-1}. \quad (31)$$

Finally, the parameters of $q(\rho^2_k)$ obey

$$\xi_{1k} \leftarrow (1-\eta^{(t)})\xi_{1k} + \eta^{(t)}(\kappa_1 + \sum_{j:(i,j)\in\mathcal{S}^{(t)}} \frac{K}{2}) \quad (32)$$

$$\xi_{2k} \leftarrow (1-\eta^{(t)})\xi_{2k} + \eta^{(t)}(\kappa_2 + \frac{1}{2}\sum_{j:(i,j)\in\mathcal{S}^{(t)}} \mu^2_{v_{kj}}). \quad (33)$$

2.2 Using time information

Each rating in the Yahoo! Music dataset includes a time stamp. This is the case both for the training set and for each (i,j) pair in the test set, which means that it is possible to improve the predictions by incorporating temporal information into the factor model. The following approach is based on [6], who obtained one of the best single-model results on the Yahoo! dataset. The FA model in (2) is changed to

$$x_{ij} = \alpha_i + \beta_j + (\boldsymbol{u}_i + \boldsymbol{u}_{it_m} + \boldsymbol{u}_{it_h} + \boldsymbol{u}_{it_d})^T(\boldsymbol{v}_j + \boldsymbol{v}_{jt_h} + \boldsymbol{u}_{it_d}), \quad (34)$$

where t_m, t_h and t_d are indices over the total number of minutes, hours and days in the dataset, respectively. The large number of user/time factors (particularly $\boldsymbol{u}_{it_m}$) vectors poses computational challenges; however, each user only has ratings for very few minutes on average, which makes the storage requirements feasible.

The priors for the new factors that depend on t_d are

$$p(\boldsymbol{u}_{it_d}) = \prod_{k=1}^{K} \mathcal{N}(u_{kit_d}; 0, 1) \quad (35)$$

$$p(\boldsymbol{v}_{jt_d}) = \prod_{k=1}^{K} \mathcal{N}(v_{kjt_d}; 0, \rho^2_k) \quad (36)$$

and similarly for those factors that depend on t_h and t_m. We apply the VB algorithm to (34) in much the same way as in the original formulation, with new variational posteriors $q(\boldsymbol{u}_{it_d})$, $q(\boldsymbol{v}_{jt_d}), \ldots$, of the same form as in Section 2.1. Writing out (34) as a sum of cross-products, we have

$$x_{ij} = \alpha_i + \beta_j + \boldsymbol{u}_i^T\boldsymbol{v}_j + \boldsymbol{u}_i^T\boldsymbol{v}_{jt_h} + \boldsymbol{u}_i^T\boldsymbol{u}_{it_d} + \boldsymbol{u}_i^T\boldsymbol{v}_{jt_m} + \ldots \quad (37)$$

which means that the gradient updates for $\alpha_i, \beta_j, \boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are changed only slightly from Section 2.1. For example, defining

$$e_{ij} = x_{ij} - \alpha_i - \beta_j - (\boldsymbol{\mu}_{\boldsymbol{u}_i} + \boldsymbol{\mu}_{\boldsymbol{u}_{it_m}} + \boldsymbol{\mu}_{\boldsymbol{u}_{it_h}} + \boldsymbol{\mu}_{\boldsymbol{u}_{it_d}})^T (\boldsymbol{\mu}_{\boldsymbol{v}_j} + \boldsymbol{\mu}_{\boldsymbol{v}_{jt_h}} + \boldsymbol{\mu}_{\boldsymbol{v}_{it_d}}),$$

we now have

$$\frac{\partial C_{KL}}{\partial \mu_{u_{ki}}} = \mu_{u_{ki}} + \sum_{j:(i,j)\in\mathcal{S}} \frac{-e_{ij}(\mu_{v_{kj}} + \mu_{v_{kjt_h}} + \mu_{v_{kjt_d}})}{\sigma^2_\epsilon} + \sum_{j:(i,j)\in\mathcal{S}} \frac{\mu_{u_{ki}}(\sigma^2_{v_{kj}} + \sigma^2_{v_{kjt_h}} + \sigma^2_{v_{kjt_d}})}{\sigma^2_\epsilon} \quad (38)$$

$$\frac{\partial C_{KL}}{\partial \mu_{v_{kj}}} = \frac{\mu_{v_{kj}}}{\rho^2_k} + \sum_{j:(i,j)\in\mathcal{S}} \frac{-e_{ij}(\mu_{u_{ki}} + \mu_{u_{kit_h}} + \mu_{u_{kit_d}} + \mu_{u_{kit_m}})}{\sigma^2_\epsilon} + \sum_{j:(i,j)\in\mathcal{S}} \frac{\mu_{v_{kj}}(\sigma^2_{v_{kj}} + \sigma^2_{u_{kit_h}} + \sigma^2_{u_{kit_d}} + \sigma^2_{u_{kit_m}})}{\sigma^2_\epsilon}. \quad (39)$$

The gradients for the new time-related factors are computed similarly to those for $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$.

3. ACTIVE LEARNING

In the following, we address the problem of finding the best mini-batch $\mathcal{S}^{(t)} = \{(i,j) : x_{ij} \text{ is observed at step } t\}$ of size m, thereby extending standard gradient descent. Instead of having a fixed training set $\mathbf{D}$ as before, the goal is now to incrementally construct a set of observed locations $\mathcal{A}$ by starting with $\mathcal{A} = \emptyset$ and sequentially appending $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{S}^{(t)}$. Define $\mathcal{V} = \{(i,j), \forall_{i,j}\}$ to be the set of all entries of $\mathbf{X}$, whether observed or not. Hence, $\mathcal{V}\backslash\mathcal{A}$ is the set of missing locations. It is known [8] that finding the best $\mathcal{S}^{(t)}$ among $\mathcal{V}\backslash\mathcal{A}$ is, in general, an NP-hard problem.

The active selection of mini-batches $\mathcal{S}^{(t)}$ may be constituted in at least two forms. In one case, all of the (sparsely sampled) data may be available *a priori*, and rather than selecting $\mathcal{S}^{(t)}$ from that data in a random manner, one may use active learning to select the most-informative samples. In a second case, we may actually seek entries (acquire the database) actively and sequentially, based on the results of active learning. This may be done by asking selected customers to rate selected items. There may be other settings that are a combination of the above two cases.

We wish to leverage the current posterior estimate of $\boldsymbol{\theta}$ for selecting the m most informative observations in $\mathcal{V}\backslash\mathcal{A}$. One possible strategy is to select the observations x_{ij} for which the posterior implies the greatest variance, and thus uncertainty. This method, while simple, does not account for correlations between similar samples, *i.e.*, the fact that many high variance users/items may be very similar to each other, and therefore redundant. Moreover, selecting high variance users/items tends to unduly emphasize outliers.

A better alternative is to select the observations that are most informative about $\mathcal{V}\backslash\mathcal{A}$. Toward this end, we build on [8], where near-optimal, polynomial-time greedy algorithms are proposed. These algorithms exploit *submodularity* properties of mutual information, under a Gaussian process formulation.

3.1 Gaussian process formulation

In general, assume that each observation $x_s \in \mathbb{R}$, where s is an index over elements of $\mathcal{V}$, has an associated covariate $\boldsymbol{r}_s \in \mathbb{R}^d$. We wish to impose that, when $\boldsymbol{r}_s$ and $\boldsymbol{r}_{s'}$ are similar, then x_s and $x_{s'}$ are correlated. If we have a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$ drawn from a Gaussian process (GP), with covariance function given by a Mercer kernel $\mathcal{K}(\cdot,\cdot)$, then $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with

$$\boldsymbol{\Sigma}_{s,s'} = \mathcal{K}(\boldsymbol{r}_s, \boldsymbol{r}_{s'}). \tag{40}$$

A frequent choice for $\mathcal{K}(\cdot,\cdot)$ is the radial basis function (RBF) kernel

$$\mathcal{K}(\boldsymbol{r}_s, \boldsymbol{r}_{s'}) = \exp(-\frac{1}{\sigma^2}\|\boldsymbol{r}_s - \boldsymbol{r}_{s'}\|^2). \tag{41}$$

A key property of GPs is that, given a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$ with associated covariates $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$, one may readily draw x_{N+1} for any new covariate $\boldsymbol{r}_{N+1}$. This property facilitates active learning in the following way. Assume that the set $\mathcal{V}$ defines a set of covariates and the associated data vector is $\boldsymbol{x} \sim \mathrm{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\mathcal{V}\mathcal{V}})$. The active learning problem now becomes one of choosing a subset $\mathcal{A} \subset \mathcal{V}$ of covariates at which to sample data.

To connect the GP formulation with our matrix factorization problem, it is appropriate to consider the user and item factors, $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, as covariates. The ratings x_{ij} are considered to be drawn from a GP where the covariance matrix is related to the $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ covariates, which are *latent*, *i.e.*, non-observed. The online VB procedure is employed to provide approximate posteriors for these latent covariates.

Accordingly, we wish to define kernels which make use of the estimated posterior distributions $q(\boldsymbol{u}_i)$ and $q(\boldsymbol{v}_i)$. A natural choice consists of the symmetrized KL divergence kernel, which for user factors $\boldsymbol{u}_i$ is defined as

$$\mathcal{K}(\boldsymbol{u}_i, \boldsymbol{u}_{i'}) = \exp\left\{D_{\mathrm{KL}}(q(\boldsymbol{u}_i)||q(\boldsymbol{u}_{i'})) + D_{\mathrm{KL}}(q(\boldsymbol{u}_{i'})||q(\boldsymbol{u}_i))\right\} \tag{42}$$

$$D_{\mathrm{KL}}(q(\boldsymbol{u}_i)||q(\boldsymbol{u}_{i'})) = \frac{1}{2}[\mathrm{tr}(\boldsymbol{\Phi}_{i'}^{-1}\boldsymbol{\Phi}_i) +$$

$$\|\boldsymbol{\mu}_{\boldsymbol{u}_i'} - \boldsymbol{\mu}_{\boldsymbol{u}_i}\|^T \boldsymbol{\Phi}_{i'}^{-1}\|\boldsymbol{\mu}_{\boldsymbol{u}_i'} - \boldsymbol{\mu}_{\boldsymbol{u}_i}\| + \log\frac{|\boldsymbol{\Phi}_i|}{|\boldsymbol{\Phi}_{i'}|} - K], \tag{43}$$

where $\boldsymbol{\Phi}_i \triangleq \mathrm{diag}(\sigma_{u_{1i}}^2, \ldots, \sigma_{u_{Ki}}^2)$. Item factors $\boldsymbol{v}_j$ are treated similarly, substituting $\boldsymbol{v}_j$ for $\boldsymbol{u}_i$ and $\boldsymbol{\Psi}_j \triangleq \mathrm{diag}(\sigma_{v_{1j}}^2, \ldots, \sigma_{v_{Kj}}^2)$ for $\boldsymbol{\Phi}_i$. Note that the matrices $\boldsymbol{\Phi}_i$ and $\boldsymbol{\Psi}_j$ are diagonal, which simplifies inversions and determinants. An attractive feature of the symmetrized KL divergence kernel is the absence of tuning parameters such as kernel bandwidths.

3.2 Submodularity and mutual information

An appropriate criterion for choosing $\mathcal{A}$ is to seek to maximize the mutual information $\mathrm{MI}(\mathcal{A}) \triangleq \mathrm{I}(\mathcal{A}; \mathcal{V}\backslash\mathcal{A}) = h(\boldsymbol{x}_\mathcal{A}) - h(\boldsymbol{x}_\mathcal{A}|\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}})$, with $h(\cdot)$ denoting the differential entropy. Given $\mathcal{A}$, it is straightforward to compute $\mathrm{MI}(\mathcal{A})$ within the GP model. However, when $|\mathcal{A}| > 1$ the computation to select the set $\mathcal{A}$ that maximizes this mutual information is NP-complete. This difficulty can be overcome by using the fact that $\mathrm{MI}(\mathcal{A})$ is *submodular*.

Definition of submodularity [13]: A set function $F : \mathcal{A} \rightarrow F(\mathcal{A})$ is submodular if, for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$, and $y \in \mathcal{V}\backslash B$, it holds that $F(\mathcal{A} \cup y) - F(\mathcal{A}) \geq F(\mathcal{B} \cup y) - F(\mathcal{B})$.

Intuitively, F is submodular if it has a "diminishing returns" property, *i.e.*, a new datum y has more impact when the set $\mathcal{A}$ is smaller. The following theorem ensures that choosing $\mathcal{A}$ to maximize $F(\mathcal{A})$ can be done in a near-optimal manner by a greedy algorithm.

Theorem [13]: Let F be a monotone submodular set function over a finite ground set $\mathcal{V}$ with $F(\emptyset) = 0$. Let $\mathcal{A}$ be the set of the first m elements chosen to maximize $F(\cdot)$ by a greedy algorithm, and let $\mathrm{OPT} = \max_{\mathcal{A}\subset\mathcal{V};|\mathcal{A}|=m} F(\mathcal{A})$. Then

$$F(\mathcal{A}) \geq (1 - \frac{1}{e})\mathrm{OPT}. \tag{44}$$

Importantly, the mutual information $\mathrm{MI}(\mathcal{A}) \triangleq \mathrm{I}(\mathcal{A}; \mathcal{V}\backslash\mathcal{A})$ is submodular [8], and $\mathrm{MI}(\emptyset)=0$. There are conditions of practical relevance for which the mutual information metric is a monotone function, namely when $|\mathcal{A}|$ is small relative to the total number of entries that may searched over, which is certainly the case for the massive matrices of interest here; see [8] for a related discussion, although that work did not consider the matrix-completion problem. Therefore, there exist polynomial-time greedy algorithms for finding $\widehat{\mathcal{A}}$ such that

$$\mathrm{MI}(\widehat{\mathcal{A}}) > (1 - 1/e) \max_{\mathcal{A}\subset\mathcal{V},|\mathcal{A}|=m} \mathrm{MI}(\mathcal{A}) - m\epsilon \tag{45}$$

for some small ϵ. The mutual information metric discourages acquisition of samples that are anticipated to be correlated to each other based upon the GP covariance $\mathcal{K}(\cdot,\cdot)$.

Specifically, Algorithm 1 in [8] finds candidate $y \in \mathcal{V}\backslash\mathcal{A}$ to maximize the increase in mutual information

$$\mathrm{MI}(\mathcal{A} \cup y) - \mathrm{MI}(\mathcal{A}) = h(\boldsymbol{x}_y|\boldsymbol{x}_\mathcal{A}) - h(\boldsymbol{x}_y|\boldsymbol{x}_{\bar{\mathcal{A}}}), \tag{46}$$

where $\bar{\mathcal{A}}$ denotes $\mathcal{V}\backslash(\mathcal{A} \cup y)$. The conditional distributions needed to compute the conditional differential entropies in (46) are readily manifested by leveraging the ability of GP to compute such densities, as discussed above. This property is a key reason for imposing the GP representation for matrix values, with associated covariates defined by the inferred feature vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$. In the GP model, the maximization is done by computing

$$\max_y \delta_y = \frac{\mathcal{K}(y,y) - \boldsymbol{\Sigma}_{y\mathcal{A}}\boldsymbol{\Sigma}_{\mathcal{A}\mathcal{A}}^{-1}\boldsymbol{\Sigma}_{\mathcal{A}y}}{\mathcal{K}(y,y) - \boldsymbol{\Sigma}_{y\bar{\mathcal{A}}}\boldsymbol{\Sigma}_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1}\boldsymbol{\Sigma}_{\bar{\mathcal{A}}y}}. \tag{47}$$

Equation (47) is a ratio of conditional covariances of the form $\sigma_{y|\mathcal{A}}^2 = \mathcal{K}(y,y) - \boldsymbol{\Sigma}_{y\mathcal{A}}\boldsymbol{\Sigma}_{\mathcal{A}\mathcal{A}}^{-1}\boldsymbol{\Sigma}_{\mathcal{A}y}$, which due to the GP properties can be computed without first measuring x at location y.

Unfortunately, the aforementioned algorithm requires $O(|\mathcal{V}\backslash\mathcal{A}|^4)$ operations, which is indeed polynomial-time but infeasible in many situations of interest. For instance, in the case of the Yahoo! dataset, we have $N_u \approx 10^6$ and $N_v \approx 6 \times 10^5$, leading to $|\mathcal{V}\backslash\mathcal{A}| \approx 6 \times 10^{11}$. This problem can be mitigated by employing Algorithm 3 in [8] which uses *local kernels*, *i.e.*, only a small neighborhood around each y is considered. More specifically, in expression (46) the approximation $h(y|\tilde{\mathcal{A}}) \approx h(y|\mathcal{A})$ is used, where $\tilde{\mathcal{A}}$ results from removing all but the k-nearest neighbors of y from $\mathcal{A}$. This approximation drastically reduces the size of all $\mathbf{\Sigma}_{(\cdot,\cdot)}$ matrices in (47), to a maximum of $k \times k$, at a small cost in the objective function [8]. The overall complexity is thus reduced to $O(|\mathcal{V}\backslash\mathcal{A}|)$, which nevertheless remains impractical for our purposes.

Our proposed approach circumvents these limitations by exploiting the structure of the matrix factorization problem. Instead of exploring all possible pairs (i, j), of which there are approximately $N_u N_v$, our main idea is to choose the best row and the best column *separately*. The resulting computations are now $O(N_u) + O(N_v)$, which is feasible.

The objective function for rows of $\mathbf{X}$ is, therefore,

$$\delta_i = \frac{\sigma_i^2 - \mathbf{\Sigma}_{i\mathcal{A}_u} \mathbf{\Sigma}_{\mathcal{A}_u\mathcal{A}_u}^{-1} \mathbf{\Sigma}_{\mathcal{A}_u i}}{\sigma_i^2 - \mathbf{\Sigma}_{i\bar{\mathcal{A}}_u} \mathbf{\Sigma}_{\bar{\mathcal{A}}_u\bar{\mathcal{A}}_u}^{-1} \mathbf{\Sigma}_{\bar{\mathcal{A}}_u i}}, \tag{48}$$

where $\bar{\mathcal{A}}_u$ denotes $\{1, \ldots, N_u\}\backslash(\mathcal{A}_u \cup y)$, with y the candidate ith row. An identical objective function is defined for the columns of $\mathbf{X}$, substituting j for i and v for u.

The goal is now to find optimum sets $\mathcal{S}_u = \{i_1, \ldots, i_m\}$, $\mathcal{S}_v = \{j_1, \ldots, j_m\}$ sorted by δ_i and δ_j, respectively. Then, we form $\mathcal{S}$ based on $\mathcal{S}_u$ and $\mathcal{S}_v$. There are multiple possibilities for combining $\mathcal{S}_u$ and $\mathcal{S}_v$. We choose to construct $\mathcal{S} = \{(i_1, j_1), \ldots, (i_m, j_m)\}$. We then sequentially build $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{S}$, as prescribed initially. The entire procedure is summarized in Algorithm 1, which we call VB-MI.

Algorithm 1 Active learning for matrix factorization with VB-MI

1: Initialize $q(\mathbf{U}) : \mu, \phi$, and $q(\mathbf{V}) : \nu, \psi$
2: Initialize active set $\mathcal{A} \leftarrow \emptyset$
3: **while** Stopping criterion is not met **do**
4: **for** $i \in \{1, \ldots, N_u\}\backslash\{i : i \in \mathcal{A}\}$ **do**
5: Compute δ_i based on $\mu, \phi, \mathcal{A}$ and kernel $\mathcal{K}$
6: **end for**
7: Repeat steps 4–6 for $j \in \{1, \ldots, N_v\}\backslash\{j : j \in \mathcal{A}\}$, substitute δ_j, ν, ψ
8: $\mathcal{S}_u = \{i_1, \ldots, i_m\}$ sorted in descending order of δ_i
9: $\mathcal{S}_v = \{j_1, \ldots, j_m\}$ sorted in descending order of δ_j
10: $\mathcal{S} = \{(i_1, j_1), \ldots, (i_m, j_m)\}$
11: Collect observations $x_{ij}, \forall (i, j) \in \mathcal{S}$
12: $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{S}$
13: Update μ, ϕ and ν, ψ via online VB
14: **end while**

3.3 Simplified active learning

A simpler alternative to the aforementioned submodularity approach consists of finding $\mathcal{S}_u$ and $\mathcal{S}_v$ by sorting the user and item factors in decreasing order of the posterior trace-covariance, given by $\mathrm{tr}(\mathbf{\Phi}_i)$ and $\mathrm{tr}(\mathbf{\Psi}_j)$, respectively. The mini-batch $\mathcal{S}$ is then constructed as in VB-MI and appended to $\mathcal{A}$. The resulting algorithm makes direct use of $q(\boldsymbol{u}_i)$ and $q(\boldsymbol{v}_j)$ without necessitating any GP formulation or kernel, and the only overhead over online VB consists of two sorting steps per mini-batch, which is negligible.

As discussed above, a limitation of this approach is that it does not account for correlations, and therefore many high-variance items may also have high correlation, and consequently it is redundant to sample all of them. Nevertheless, as discussed below, this has been found to work well in practice, at low computational expense, for the large-scale examples considered. This approach explicitly exploits the products (parameter variance) from the VB analysis, and it is referred to below as VB-Variance.

3.4 Non-Bayesian method

For the purpose of comparison, and also to isolate the contribution of the active learning method towards performance from that of the VB algorithm, we have implemented an active learning method capable of utilizing products from a non-Bayesian matrix analysis; in that analysis, the RMSE is directly minimized w.r.t. $\{\alpha_i, \beta_i, \boldsymbol{u}_i, \boldsymbol{v}_j\}$ by traditional gradient descent (we learn a "point" estimate for these parameters, rather than estimating a full posterior). This is distinct from the VB method, which minimizes the KL divergence between $q(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}; \boldsymbol{\theta})$ and the posterior $p(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{U}, \mathbf{V}, \boldsymbol{\rho}|\mathbf{X})$ w.r.t. the posterior parameters $\boldsymbol{\theta}$.

We follow an approach similar to [9] and estimate the parameters $\alpha_i, \beta_j, \boldsymbol{u}_i, \boldsymbol{v}_j$ by solving the optimization problem

$$\begin{aligned} \min_{\alpha_i, \beta_j, \boldsymbol{u}_i, \boldsymbol{v}_j} & \sum_{(i,j)\in\mathcal{A}} (x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)^2 \\ \text{subject to} \quad & \max\{\|\boldsymbol{u}_i\|_2^2, \|\boldsymbol{v}_j\|_2^2\} \le B, \forall_{ij} \\ & |\alpha_i|^2 \le B, \forall i, \quad |\beta_j|^2 \le B, \forall j \end{aligned} \tag{49}$$

where the constraint on the norms of the factors $\boldsymbol{u}_i$, $\boldsymbol{v}_j$ is equivalent to max-norm regularization of $\mathbf{U}^T\mathbf{V}$, controlled by the parameter B. The max-norm of a matrix $\mathbf{Z}$ is defined as $\|\mathbf{Z}\|_{\max} = \max\{|z_{ij}|\}$. The bias vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are regularized in the same way.

Problem (49) can be solved by slightly modifying the simple projected gradient method proposed in [9] to take α_i and β_j into account. Given one training example x_{ij}, the following updates are computed for $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$:

$$\boldsymbol{u}_i \leftarrow P_B[\boldsymbol{u}_i - \eta(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)\boldsymbol{v}_j] \tag{50}$$

$$\boldsymbol{v}_j \leftarrow P_B[\boldsymbol{v}_j - \eta(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)\boldsymbol{u}_i], \tag{51}$$

with $P_B(\cdot)$ a projection operator defined by

$$P_B(\boldsymbol{z}) = \begin{cases} \boldsymbol{z} & \|\boldsymbol{z}\|_2^2 \le B \\ \frac{\boldsymbol{z}}{\sqrt{B}} & \|\boldsymbol{z}\|_2^2 > B \end{cases}. \tag{52}$$

Note how the gradient term $-(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)\boldsymbol{v}_j$ is also present in the VB update equation (20), where it is weighted by the noise level $1/\sigma_\epsilon^2$ and combined with other terms that depend on the model variances. The updates for α_i and β_j are given by

$$\alpha_i \leftarrow P_B[\alpha_i + \eta(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)] \tag{53}$$

$$\beta_j \leftarrow P_B[\beta_j + \eta(x_{ij} - \alpha_i - \beta_j - \boldsymbol{u}_i^T \boldsymbol{v}_j)], \tag{54}$$

which also resemble the corresponding VB updates.

To perform active learning with this model, the user and item submodular objective functions δ_i and δ_j are computed as before, with the difference that we can no longer use the KL divergence kernel. Instead, we define standard RBF kernels

$$\mathcal{K}(\boldsymbol{u}_i, \boldsymbol{u}_{i'}) = \exp(-\frac{1}{\sigma^2}\|\boldsymbol{u}_i - \boldsymbol{u}_{i'}\|_2^2) \quad (55)$$

$$\mathcal{K}(\boldsymbol{v}_j, \boldsymbol{v}_{j'}) = \exp(-\frac{1}{\sigma^2}\|\boldsymbol{v}_j - \boldsymbol{v}_{j'}\|_2^2) \quad (56)$$

where σ^2 is the bandwidth parameter. Typically, σ^2 is set via cross-validation (which we emphasize is avoided with the VB inference).

4. RESULTS

We first present results for the online VB algorithm applied to the dataset of Yahoo! Music ratings provided for the KDD Cup 2011 challenge [4]. The purpose of this analysis is to examine the performance of the online VB solution on such large-scale data. In that analysis the mini-batches in the online analysis are selected randomly, as done in most previous related research (although virtually none of that work was done with a VB analysis). We then compare results in which the mini-batches for the online method are selected randomly versus actively, via the method discussed in Section 3.

4.1 Results of online VB on Yahoo! dataset

We consider the FA model and online VB inference from Section 2.1 on the Yahoo! Music dataset. We set the number of factors to $K = 20$ and choose the learning step size to follow the rule $\eta^{(t)} = \tau\eta^{(t-1)}$, with $\eta^{(0)} = 2.5 \times 10^{-7}$ and $\tau = 0.9$. The mini-batch size was set to $m = |\mathcal{S}^{(t)}| = 100000$. Hyperparameter values were set to $\sigma_\epsilon^2 = 10^{-6}$,$\tau_\alpha^2 = \tau_\beta^2 = 1$,$\kappa_1 = \kappa_2 = 10^{-6}$, which yield standard "flat" (nearly non-informative) priors. The variational posterior parameters $\boldsymbol{\theta}$ were initialized at random, and the algorithm was executed for 40 epochs. Each epoch consists of a full pass through the training set, *i.e.*, approximately 252 million ratings. A further 6 million ratings is used for validation (these validation ratings were supplied to the users), while the test set consists of 4 million ratings whose scores were withheld at the time of the KDD Cup 2011 challenge. These scores have been released after the competition. The performance measure is the root mean square error (RMSE) over the test dataset, defined as

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbf{D}_{\text{test}}|}\sum_{x_{ij}\in\mathbf{D}_{\text{test}}}(x_{ij} - \mu_{\alpha_i} - \mu_{\beta_j} - \boldsymbol{\mu}_{\boldsymbol{u}_i}^T\boldsymbol{\mu}_{\boldsymbol{v}_j})^2}, \quad (57)$$

where the posterior means of $\boldsymbol{\theta}$ are used to provide point estimates.

We show the RMSE and computation time per epoch on both the validation and test sets for our VB method in Table 4.1. For comparison, we also show the RMSE and time achieved by the baseline model supplied by Yahoo!, which models the ratings as

$$x_{ij} \approx a_i + b_j + c, \quad (58)$$

with a_i, b_j and c learned by standard stochastic gradient descent. The performance gain over the baseline is over 10%,

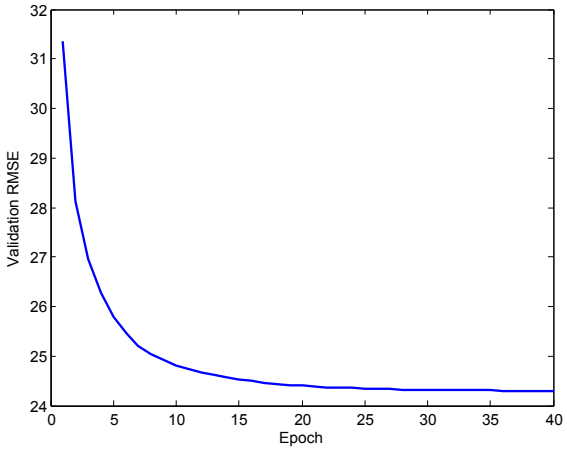

Figure 1: RMSE on the validation set for 40 epochs. Each epoch is a full pass over approximately 252M ratings. The total running time is slightly less than 10 hours.

which is considered quite significant and places the method near the top 100 in the competition. Note that the initialization is random and that there are no separate learning steps for the various parameters. Moreover, this is a single model (the best competitors employed a blend of dozens of different models) and it has relatively few parameters.

The convergence behavior is illustrated in Figure 4.1. While it takes slightly less than 10 hours to execute 40 epochs (we have used MATLAB code, non-optimized, on a 2.4 GHz CPU), it is apparent that the algorithm has essentially converged after approximately 25 epochs.

Additionally, we have implemented the time-dependent FA model described in Section 2.2, and achieved a further RMSE drop from 25.94 to 24.8. The most demanding aspect of the time-dependent model is the large number of user/minute factors $\boldsymbol{u}_{t_m}$, since $t_m \in \{0, \ldots 5726101\}$, although the fact that, on average, each user only has ratings for104 minutes in the dataset makes the storage requirements feasible. Currently, we are pursuing the implementation of active learning methods with this model (with time dependence), which will likely require extensive parallelization in order to overcome the large computational requirements.

4.2 Results with active learning

We have applied the three active learning procedures described in Section 3 (VB-MI, VB-Variance and Non-Bayesian) to the Yahoo! Music problem, and compared to online VB with random sampling. In this analysis we consider the model in (2), without time dependence. These experiments differ from using online VB as in Section 4.1. Here we synthesize all ratings, in order to be able to measure any desired x_{ij}, and not limit ourselves to the fixed training dataset $\mathbf{D}$ supplied by Yahoo!. We first use online VB to learn a factor model of the form (2) using all available data from $\mathbf{D}$, as in Section 4.1. The resulting posterior parameters $\boldsymbol{\theta}^*$ are withheld. Then, we discard the data supplied by Yahoo! and use the posterior mean components of $\boldsymbol{\theta}^*$ to synthesize

Table 1: Online VB method with random mini-batches vs. Yahoo! baseline model

Model	Validation RMSE	Test RMSE	Time/Epoch
Yahoo!	26.68	28.98	~ 0.5 min.
Online VB	24.34	25.94	~ 15 min.

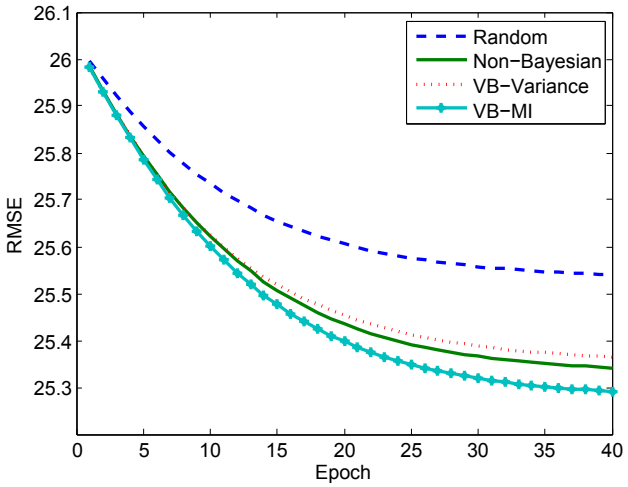

Figure 2: RMSE over 40 epochs. The VB-MI algorithm is based on mutual information with the KL divergence kernel, VB-Variance is based on sampling the users and items with highest variance and the Non-Bayesian method uses mutual information with the RBF kernel.

any observation x_{ij} on demand, by computing

$$x_{ij} = \mu^*_{\alpha_i} + \mu^*_{\beta_j} + (\mu^*_{\boldsymbol{u}_i})^T(\mu^*_{\boldsymbol{v}_j}). \tag{59}$$

We report the RMSE on a test set of size 4 million, separately drawn at random from the factor model and withheld from the algorithms.

In all cases, the algorithms were initialized with one single mini-batch of online VB and then executed for 40 epochs. The mini-batch size is $m = 100000$. In this setting, since there no longer exists a dataset $\mathbf{D}$ over which to perform a full pass, we define an epoch as the number of steps necessary for the set $\mathcal{A}$ to have the same size as $\mathbf{D}$. Once this happens, we reset $\mathcal{A} \leftarrow \emptyset$ and begin a new epoch. The factor model parameters carry over between successive epochs.

As seen in Figure 2, all active methods outperform random sampling by a significant margin. The differences in RMSE between the three active methods are less pronounced, with the mutual-information-based VB-MI algorithm slightly outperforming the others. It is particularly notable that the simpler VB-Variance method is competitive with the submodularity based approaches, at a fraction of the computational cost. The running times, per epoch, of VB-MI, Non-Bayesian and VB-Variance are 5100 seconds, 4700 seconds and 1100 seconds, respectively. Thus, the VB-Variance method is only slightly more demanding than online VB.

Figure 3 shows histograms of the (scalar) components of the user and item factors, respectively $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$. The model assigns low variance to most components, which can be seen as an indicator of good model fit.

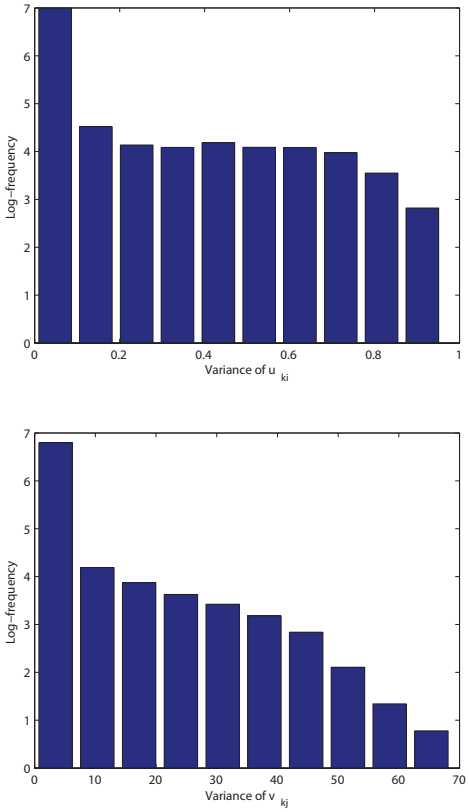

Figure 3: Histograms of the variance of u_{ki} (top) and v_{kj} (bottom), where i, j, k are indices over users, items and factors, respectively. Shown in logarithmic scale. The majority of the components have low variance.

The plot in Figure 3 depicts the trace-covariance of the item factors $\boldsymbol{v}_j$, as returned by the online VB algorithm. There exists a clear relationship with the number of item ratings. As anticipated, rarely rated items tend to have higher variance, although a few spikes in the variance for relatively highly rated items may warrant further investigation. We do not show an equivalent plot for user factors due to the fact that the relationship with the number of ratings is much less evident.

5. CONCLUSIONS

We have presented an online variational Bayesian approach to the problem of large-scale matrix completion, with applications in collaborative filtering. Unlike standard matrix factorization approaches, the developed statistical model en-

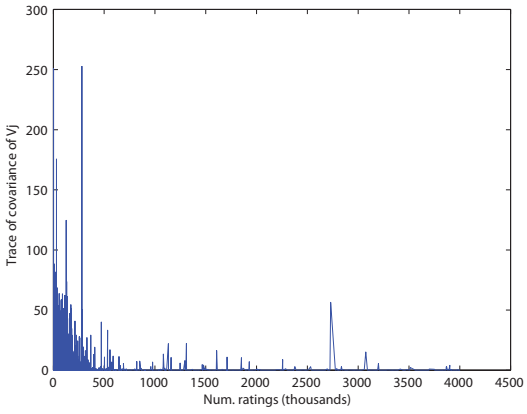

Figure 4: Trace of covariance of v_j (item factors) as a function of number of ratings. Rarely rated items tend to have higher variance.

ables sequential selection of near-optimal subsets of users and items through active learning. This can be done using efficient greedy algorithms which rely on submodularity.

We have also described an alternative method based directly on the inferred variance of the factors, which relaxes the near-optimality guarantees and requires much lower computational effort, at the cost of a small performance penalty. The aforementioned methods have been compared with random sampling and with a non-Bayesian submodularity-based algorithm.

We have demonstrated the proposed framework using the challenging Yahoo! Music dataset, with promising results. Notably, all proposed active learning algorithms outperform random sampling. Comparison with the non-Bayesian method shows comparable performance. However, the RBF kernel in non-Bayesian method requires tuning of the bandwidth parameter, while the KL divergence kernel used in the Bayesian approaches is parameter-free. Moreover, the computational overhead of the submodular methods far outweighs the difference in computations between the Bayesian and non-Bayesian algorithms. Importantly, neither the KL divergence kernel nor the fast variance-based active learning algorithm would be possible without the (approximate) posterior estimate provided by the Bayesian method.

Future research will focus on incorporating temporal information into the active learning algorithms. This should be possible via developments in parallel computation.

Acknowledgements

The research reported here was supported by ARO, NGA, ONR and DARPA (MSEE program).

6. REFERENCES

[1] H. Attias. A variational bayesian framework for graphical models. *Advances in Neural Information Processing Systems (NIPS)*, 12(1-2):209–215, 2000.

[2] R. Bell and Y. Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[3] E. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[4] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! Music Dataset and KDD-Cup'11. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD), KDD Cup Workshop*, 2011.

[5] M. Hoffman, D. Blei, and F. Bach. Online learning for latent Dirichlet allocation. *Advances in Neural Information Processing Systems (NIPS)*, 23:856–864, 2010.

[6] M. Jahrer and A. Töscher. Collaborative filtering ensemble. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD), KDD Cup Workshop*, 2011.

[7] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434, 2008.

[8] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[9] J. Lee, B. Recht, R. Salakhutdinov, N. Srebro, and J. Tropp. Practical large-scale optimization for max-norm regularization. *Advances in Neural Information Processing Systems (NIPS)*, 23:1297–1305, 2010.

[10] Y. Lim and Y. Teh. Variational bayesian approach to movie rating prediction. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD), KDD Cup Workshop*, pages 15–21, 2007.

[11] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[12] S. Nakajima and M. Sugiyama. Implicit regularization in variational Bayesian matrix factorization. In *27th International Conference on Machine Learning (ICML)*, 2010.

[13] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[14] T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for large scale problems with lots of missing values. *Machine Learning: ECML 2007*, pages 691–698, 2007.

[15] T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, 8:155–201, 2007.

[16] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

[17] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.