Active Learning and Search on Low-Rank Matrices

Dougal J. Sutherland
dsutherl@cs.cmu.edu

Barnabás Póczos
bapoczos@cs.cmu.edu

Jeff Schneider
schneide@cs.cmu.edu

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

ABSTRACT

Collaborative prediction is a powerful technique, useful in domains from recommender systems to guiding the scientific discovery process. Low-rank matrix factorization is one of the most powerful tools for collaborative prediction. This work presents a general approach for *active* collaborative prediction with the Probabilistic Matrix Factorization model. Using variational approximations or Markov chain Monte Carlo sampling to estimate the posterior distribution over models, we can choose query points to maximize our understanding of the model, to best predict unknown elements of the data matrix, or to find as many "positive" data points as possible. We evaluate our methods on simulated data, and also show their applicability to movie ratings prediction and the discovery of drug-target interactions.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Collaborative filtering; active learning; active search; cold-start; matrix factorization; recommender systems; drug discovery

1. INTRODUCTION

Collaborative prediction and collaborative filtering have in recent years been a topic of great interest, largely because they form the core component of many corporations' systems that recommend products or other items to their users. One of the most popular techniques for collaborative filtering is matrix factorization: since it is assumed that only a few factors affect a user's opinion of a movie, the matrix of users' ratings for items should be low-rank (or have a low trace norm, or any of other similar conditions). We can then perform a factorization similar to that of singular value decomposition to reconstruct the full matrix from the relatively few elements we know [20].

The same general approach, however, is applicable to a wide variety of problems, including tasks in computer vision [4, 32], network latency prediction [21], predicting the outcomes of sporting events [1], and many others. It can be applied in any situation where we expect "users" to behave similarly on "items", whether the "users" are professional basketball teams' offenses and the "items" are their defenses (c.f. [1]), or the "users" are drugs and the "items" are biological targets. Traditional collaborative filtering includes no side information about the content items, but there are various methods for adding this information [1, 7, 30, 32].

Most research in this area has focused on how well users' ratings may be predicted given a fixed training set. That was the only criterion considered for the well-known Netflix Prize,[1] for example. In many areas of machine learning, however, the problem of active learning is also important: how well an algorithm can select points to add to the training set that will lead to the best final result. We can ask the same question of matrix factorization methods as well [22]: if we do not know all the elements of a matrix, but we are allowed to query the labels of certain points in the matrix, then which points should we choose to gain the best understanding of the full matrix? To approach this task, we must define a *selection criterion* in addition to the *learning model* that will attempt to bring the learner to the greatest understanding as quickly as possible.

One practical situation in which this is particularly important is the "new user" (or "cold start") problem for recommender systems: such systems must quickly learn a rough sense of a new user's preferences based on little available information before users abandon the system. This cold start problem has seen a fair amount of research, but is far from the only collaborative filtering application which benefits from active learning. In the product recommendation domain, it is also often the case that companies add items to their system in fairly large "batches", at which point few or no user recommendations will be available. The problem of learning the attributes of new products is different than the task of recommending items to a new user, both because the products come in batches and because a product will not become frustrated if it is not immediately recommended to a variety of viewers.

In another application area entirely, pharmaceutical companies and researchers wish to discover which of various

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'13, August 11–14, 2013, Chicago, Illinois, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2174-7/13/08 ...$15.00.

[1]`http://www.netflixprize.com/`

candidate drugs will interact with many different biological targets. Since drugs' behavior typically has similarities to that of other drugs, and targets are acted on in similar fashion to other targets, collaborative filtering (perhaps with additional side information based on biologically relevant features) is likely to perform fairly well at predicting drug interactions. Determining whether an interaction occurs, however, is an expensive procedure that requires performing experiments in the lab; since it is impossible to test all possible actions, the researcher must choose a subset to examine. The active learning paradigm described here can assist in choosing the subset to examine [17].

In these and many other applications, accurate predictions are not the goal of the system, but rather simply a means to an end. In recommender systems, we ultimately want to suggest items that a user will *like*, not just build an accurate model of their preferences. In the drug-target scenario, we care more about finding new drugs that interact with a certain target, or finding the targets a drug affects, than we do about listing all the targets with which a given drug does not interact. Here our ultimate goal is not to actively predict all the unknown elements of the data matrix, but instead to find the *largest unknown elements* in the matrix. This problem adds a layer of the exploration-exploitation trade-off not present in the active learning for prediction error task. It can, however, also be effectively approached through the same framework; we simply need to define different selection criteria.

The main components of this work are:

- Criteria for active learning and active search on low-rank matrices, addressing several possible goals (Section 3) with different selection criteria (Section 5).
- A variational extension of the PMF model allowing for active learning (Section 4.1).
- An MCMC scheme for PMF, following [23], as another method for providing the information necessary for active learning (Section 4.2).
- Empirical evaluation of these methods on synthetic, movie rating, and drug discovery tasks (Section 6).

2. RELATED WORK

There has been a significant amount of prior research on various methods for low-rank matrix factorization. These methods play an important role in numerous machine learning and statistical tools, including principal component analysis, factor analysis, independent component analysis, dictionary learning, and collaborative filtering, just to name a few. One of the most influential recent models, which we will employ in this paper, is the Probabilistic Matrix Factorization (PMF) method [24], as well as its Markov chain Monte Carlo (MCMC) extension (Bayesian PMF, or BPMF) [23]. PMF, which will be reviewed in Section 4, is a generative model for matrices assumed to be of a certain rank.

Earlier work by [27] yielded the Maximum Margin Matrix Factorization (MMMF) model, which frames the matrix factorization problem as a semidefinite program based on the margin of predictions, and can be viewed as a generalization of support vector machines (SVMs). MMMF minimizes the trace norm of the factorization, which is a convex surrogate for the rank. Although the standard model predicts binary class labels, it can be modified for ordinal labels.

Active learning for recommender systems and collaborative filtering in general has also received a fair amount of attention. Rubens et al. [22] provide an overview of how general-purpose active learning techniques may be applied to recommendation systems. Much of the published research on this topic has focused on the Aspect Model [9], which assigns latent "aspect" variables to users and items. In this model, Yu et al. [31] select query points by considering the expected reduction in entropy in the model distribution. Boutilier et al. [2] instead seek out the item which will bring the greatest change in value to the highest ratings. Jin and Si [11] note that estimation based on the belief about a given rating under the currently most likely model can be misleading when that point estimate of the model is not very good, and give a full Bayesian treatment, which uses a posterior distribution on model states for inference. Karimi et al. [12], by contrast, give a much faster selection criterion based on considering which points will update the current user's parameters, under certain assumptions about the new user case for recommender systems.

There is less work on active learning specifically for matrix factorization. Karimi et al. [13] give an approach for the new user case which uses an exploration step, where the algorithm queries the item with the highest expected change to the user at hand's model, followed by an exploitation step, where the algorithm picks items based on the current parameters. Karimi et al. [14] give a method they describe as a step towards the "optimal" strategy based on minimizing the expected test error, but which makes several drastic approximations for the sake of speed. The same authors more recently developed a method which queries a new user with items popular among users with similar latent factors, to avoid the problem of asking about an item unknown to the user [15]. All three of these criteria are extensively tailored to the new user case and inapplicable in general matrix factorization settings.

Rish and Tesauro [21] use MMMF to carry out active learning for general matrix factorization problems. Following work by Tong and Koller [29] and others on active learning for SVMs, they choose to query the candidate point that has the smallest margin, representing the point about which the model is least certain. This criterion has the advantage of being simple to compute once the model has been learned. Their work considers only two-class problems, though it could potentially be extended to multi-class problems by choosing the point nearest to any label threshold. They also consider only active learning with the goal of minimizing reconstruction error, but a very similar algorithm applies to the case of finding positive instances.

Silva and Carin [26] approach the problem of active learning in a general matrix factorization problem with a similar learning model to PMF, but using a different variational approach than those discussed in Section 4.1. Whereas we assume a variational distribution of a Gaussian form allowing for general covariance structures, they assume a fully factored distribution with respect to each model parameter. This assumption allows for much more efficient learning procedures than discussed here, but also represents a far more stringent restriction on the model. This work as well considers only the goal of minimizing reconstruction error and is not directly applicable to finding positive values.

The general problem of active learning to find values in a class is termed *active search* by Garnett et al. [5], who develop a strategy optimal in the sense of Bayesian decision theory. This strategy, however, requires computation exponential in the number of lookahead steps, and is therefore impractical to apply with more than a very small lookahead window. Later work [6] provides a branch-and-bound approach for pruning the search tree, which is effective in their domain of searching on graphs with nearest-neighbor classifiers, but inapplicable in the matrix factorization setting.

3. PROBLEM SETTING

We suppose that our data lie in a matrix $R \in \mathbb{R}^{N \times M}$, only certain elements of which are initially known. The binary matrix I of the same shape as R represents the known points, so that I_{ij} is 1 if R_{ij} is observed and 0 otherwise. The set of (i, j) indexes where $I_{ij} = 1$ will be denoted by $\mathcal{O}$; we will use $R_\mathcal{O}$ to represent the set of R_{ij} with $(i, j) \in \mathcal{O}$. Some of the labels for elements (i, j) not in $\mathcal{O}$ may be requested; we call this pool of available labels $\mathcal{P}$. Our algorithm will proceed by building a model for R and using that model to select a query point in $\mathcal{P}$; it then receives the value for that point, updates the model to account for the new information, and evaluation repeats. We call the set of query points chosen by the algorithm over its execution $\mathcal{A}$.

The way in which we select elements depends on our aim in the active learning process. We consider four possible goals in this work:

Prediction: minimize prediction error on the unknown values of R: $\mathbb{E}\left[(R_{ij} - \hat{R}_{ij})^2 \mid (i, j) \notin O\right]$, where $\hat{R}$ refers to the model's predictions given $\mathcal{O}$.

Model: minimize uncertainty in the distribution of models that might have generated R: $\mathrm{H}[U, V \mid R_\mathcal{O}]$. Note that this goal only makes sense when the learning model is fixed; otherwise the entropy could be made zero by concentrating the distribution at any single point.

Magnitude Search: when the active search process is complete, have queried the largest-valued points possible: $\sum_{(i,j)\in\mathcal{A}} R_{ij}$.

Search: when the active search process is complete, have queried as many positive points as possible, for some class distinction of positive and negative points. The criterion is $\sum_{(i,j)\in\mathcal{A}} I(R_{ij} \in +)$.

The Prediction and Model goals are closely related, as are the Magnitude Search and Search goals. These goals cover a variety of use cases, although in some settings we might prefer others, such as the portion of the top k ratings that are positive [30], or the recall or precision of our predictions when viewed as a binary classifier.

4. LEARNING MODEL

The basic modeling framework we will adopt here is the PMF model of [24]. This matrix factorization technique assumes that $R \approx UV^T$ for some $U \in \mathbb{R}^{N\times D}$, $V \in \mathbb{R}^{M\times D}$, where the rank D is a hyperparameter of the model. In the setting of movie rating predictions, the ith row of U, denoted u_i, is the feature vector for the ith user. The jth row of V, denoted v_j, is the feature vector for the jth movie. User i's rating for movie j is then predicted as $u_i^T v_j$.

Specifically, PMF assumes i.i.d. Gaussian noise around the prediction UV^T, so that $R_{ij} = u_i^T v_j + \varepsilon_{ij}$ where $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$. It further regularizes the parameters U and V via zero-mean spherical Gaussian priors with variances σ_U^2 and σ_V^2, respectively. For constant hyperparameters σ^2, σ_U^2, and σ_V^2, the joint log-density $\ln p(U, V \mid R_\mathcal{O})$ then becomes

$$\frac{-1}{2\sigma^2}\|I\circ(R-UV^T)\|_F^2 - \frac{1}{2\sigma_U^2}\|U\|_F^2 - \frac{1}{2\sigma_V^2}\|V\|_F^2 + C, \quad (1)$$

where $\circ$ denotes the elementwise (Hadamard) product, $\|\cdot\|_F^2$ the squared Frobenius norm, and C a constant that does not depend on U or V. To obtain the MAP estimates $\widehat{U}$ and $\widehat{V}$, we maximize (1) in U and V, e.g. through gradient ascent.

It is worth noting that (1) is biconvex in U and V, and is highly multimodal: for any invertible matrix $\Lambda \in \mathbb{R}^{D\times D}$ with $\|U\Lambda\|_F = \|U\|_F$, $\|\Lambda^{-1}V\|_F = \|V\|_F$, we have $p(U, V \mid R_\mathcal{O}) = p(U\Lambda, V\Lambda^{-1} \mid R_\mathcal{O})$, since $(U\Lambda)\left(V\Lambda^{-1}\right)^T = UV^T$. (Any Λ which simply permutes the order of the latent dimensions satisfies this property.) In practice gradient descent and similar optimizations will typically choose one such local maximum and stay in its vicinity as we update the problem with new ratings. This does, however, somewhat complicate the interpretation of our Model learning goal.

Unfortunately, (1) lends itself only to MAP estimation; the full joint posterior distribution is intractable for direct inference. In order to carry out active learning, we need some more information about the posterior $p(U, V \mid R_\mathcal{O})$, in particular statistics such as its variance or its Shannon entropy. We will therefore need to add some more information about the posterior to our model.

4.1 Variational approximation

One method for making inferences about the joint distribution is to make a deterministic, variational approximation. In this approach, we model the joint distribution of all the elements of U and V as a distribution q from some tractable family of distributions, so that the KL divergence of our approximation $q(U, V)$ from the modeled distribution conditional on the observed elements, $p(U, V \mid R_\mathcal{O})$ in (1), is

$$\begin{aligned}
\mathrm{KL}(q\|p) &= \int q(U, V) \ln \frac{q(U, V)}{p(U, V \mid R_\mathcal{O})} \,\mathrm{d}\{U, V\} \\
&= -\mathrm{H}[q] - \mathbb{E}_q[\ln p(U, V \mid R_\mathcal{O})] \\
&= -\mathrm{H}[q] - C \qquad\qquad (2) \\
&+ \frac{1}{2\sigma_U^2}\sum_{i=1}^N\sum_{k=1}^D \mathbb{E}_q\left[U_{ik}^2\right] + \frac{1}{2\sigma_V^2}\sum_{j=1}^M\sum_{k=1}^D \mathbb{E}_q\left[V_{jk}^2\right] \\
&+ \frac{1}{2\sigma^2}\sum_{i=1}^N\sum_{j=1}^M I_{ij}\Bigg(\sum_{k=1}^D\sum_{l=1}^D \mathbb{E}_q\left[U_{ki}V_{kj}U_{li}V_{lj}\right] \\
&\qquad -2R_{ij}\sum_{k=1}^D \mathbb{E}_q\left[U_{ki}V_{kj}\right] + R_{ij}^2\Bigg)
\end{aligned}$$

where $\mathrm{H}[q] = -\int q \ln q$ denotes the Shannon entropy of density q, $\mathbb{E}_q$ stands for the expectation operator w.r.t. distribution q, and C is the constant of (1), independent of q. We then choose the "best" approximation q by minimizing (2).

One option is to select $q(U, V)$ from the family of multivariate normal distributions, with an arbitrary mean $\mu \in \mathbb{R}^{D(N+M)}$ and covariance matrix $\Sigma \in \mathbb{R}^{D(N+M)\times D(N+M)}$. We then have a closed-form expression for each of the expectations in (2), via Isserlis' Theorem [10], whose gradient is simple; the details are in the supplement.[2] This allows

[2] cs.cmu.edu/~dsutherl/active-pmf/

us to minimize (2) in μ and Σ through projected gradient descent, projecting the covariance matrix Σ to be strictly positive-definite at each step (by replacing any nonpositive eigenvalues in its spectrum with a small positive eigenvalue; this unfortunately takes time cubic in the dimension).

Σ, however, is of dimension $D^2(N+M)^2$, which can quickly become impractically large for moderately-sized matrices R. We can ease this requirement by assuming a more restrictive form on the distribution of (U, V), for instance a matrix normal distribution on the "stacked" matrix of U and V. This distribution is parameterized by a mean matrix $\mu \in \mathbb{R}^{(N+M)\times D}$, a symmetric positive-definite row covariance matrix $\Sigma \in \mathbb{R}^{(N+M)\times(N+M)}$, and a symmetric positive-definite column covariance matrix $\Omega \in \mathbb{R}^{D\times D}$. It is equivalent to a general multivariate normal distribution with covariance equal to the Kronecker product $\Omega \otimes \Sigma$. This more restrictive distribution, while still having a fairly large number of parameters, is easier to handle, and as a subset of the full multivariate normal distribution has a similar closed form for (2) and its gradient.

In some sense, these are clearly bad models for $p(U, V)$, since the q are unimodal while the p have many equivalent modes, at least $D!$, and many more local maxima. If we choose a distribution centered around one of the global modes, however, we may still get useful inferences out.

Note that previous variational approximations to PMF, such as "Parametric PMF" [25], have different goals: they use EM methods to obtain a point estimate and do not actually give more information about the posterior.

4.2 Markov chain Monte Carlo

Another approach for learning the posterior distribution of U and V is to sample them through Markov chain Monte Carlo, as in "Bayesian PMF" [23]. In this way, we know that at least asymptotically we are sampling from the full joint distribution of the original model, rather than the quite restrictive variational approximation. BPMF extends the Gaussian priors for u_i and v_j to allow any mean μ and covariance Σ, and places Gaussian-Wishart hyperpriors on μ and Σ. Specifically, this version of the model has $R_{ij} \sim \mathcal{N}(u_i^T v_j, \sigma^2)$; $u_i \sim \mathcal{N}(\mu_U, \Sigma_U)$; $v_i \sim \mathcal{N}(\mu_V, \Sigma_V)$; $\mu_U \sim \mathcal{N}(\mu_0, \frac{1}{\beta_0}\Sigma_U)$; $\mu_V \sim \mathcal{N}(\mu_0, \frac{1}{\beta_0}\Sigma_V)$; $\Sigma_U, \Sigma_V \sim \mathcal{W}^{-1}(W_0, \nu_0)$.

Salakhutdinov and Mnih [23] initialized the chain with the $\widehat{U}$, $\widehat{V}$ estimates from the MAP procedure (1) and then sampled from the posterior of U and V through Gibbs sampling, which is simple thanks to the use of conjugate priors. We choose a somewhat different sampling scheme via Hamiltonian Monte Carlo, which can exploit the gradient of the probability density to allow for much more efficient convergence to a high-dimensional target distribution [18]. This variant of MCMC simulates the motion of fictional particles with positions θ in the parameter space, potential energies defined by the log probability, momentum r. We numerically simulate their behavior according to Hamilton dynamics:

$$H(\theta, r) = -\ln p(\theta) + \tfrac{1}{2} r^T M^{-1} r + \text{const}$$
$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial r_i} \qquad \frac{dr_i}{dt} = -\frac{\partial H}{\partial \theta_i}$$

where mass matrix M (which primarily allows for rescaling of variables), and r is initially drawn from a standard normal distribution. We perform Metropolis rejection based on the total change in the Hamiltonian due to integration error.

The step size and the number of steps in the numerical integration must be tuned to the distribution at hand for good performance. The No-U-Turn Sampler (NUTS) [8], however, provides a method to choose the step size via dual averaging and the number of steps by stopping when the particle would begin to "turn around," resulting in wasted computation. Our implementation uses the Stan inference engine [28]. Especially after appropriate reparameterizations to make the geometry of the space more uniform (sampling from standardized versions of the distributions and then making appropriate transformations to obtain the true parameters), this sampler explores the parameter space more efficiently than Gibbs sampling, allowing us to obtain a good understanding of the posteriors much more quickly.

5. SELECTION CRITERIA

Once we have learned a suitable variational model or obtained samples from an MCMC procedure, we have several options for how we select points to query.

Uncertainty sampling.

One simple option for the PREDICTION goal is to query the element $(i, j) \in \mathcal{P}$ with the highest posterior variance: $\arg\max_{(i,j)\in\mathcal{P}} \operatorname{Var}[R_{ij} \mid R_\mathcal{O}]$. In the variational model, although the distribution of R_{ij} (the sum of products of correlated normal distributions) is not a known distribution, we can compute its mean and variance under q using the same types of identities as in (2); details are in the supplement.

In MCMC, we use the sample variance. Although it would also be possible to estimate the differential entropy of R_{ij} with one-dimensional sample entropy methods, we found in practice that the marginal posterior distributions of R_{ij} are typically close to Gaussian. Entropy methods would thus incur significant additional computational expense with little to no added information over the variance.

Prediction magnitude.

For the MAGNITUDE SEARCH task, the simplest approach is to choose the value with the largest prediction. That is, we select $\arg\max_{(i,j)\in\mathcal{P}} \mathbb{E}[R_{ij}]$, where the expectation is either under the variational distribution or approximated by the sample mean. This approach could also be used with a point estimate of U and V.

Cutoff probability.

In the SEARCH task, we instead partition the real line into "positive" and "negative" classes, which we denote as sets $+$ and $-$. For example, in the movie rating task we might choose 4 or 5 to be positive, and 1 through 3 to be negative, so that $+ = \{x \in \mathbb{R} \mid x \geq 3.5\}$ (the set of reals which round to the positive class). We would then choose $\arg\max_{(i,j)\in\mathcal{P}} P(R_{ij} \in + \mid R_\mathcal{O})$. This is the optimal no-lookahead algorithm for active search in the framework of [5]. In MCMC, we approximate this probability by the portion of samples where $R_{ij} \in +$; in the variational setting, we cannot compute this probability in closed form but instead choose to approximate it via a normal distribution with first two moments matched to those of $p(R_{ij} \mid R_\mathcal{O})$.

Lookahead methods.

We can also take a greedy lookahead approach, where we define some measure $f(q)$ of the quality of our model and choose the element (i, j) which maximizes (or minimizes) $\mathbb{E}[f(q) \mid R_{\mathcal{O}\cup(i,j)}]$. We will present only the one-step version here for simplicity; the extension to multiple steps of lookahead is straightforward, though its computational expense grows exponentially.

When the ratings in R are from a small, discrete set $\mathcal{X}$ (e.g. $\mathcal{X} = \{1, 2, 3, 4, 5\}$ in the movie ratings setting; this is true in all of the cases considered here except that of Section 6.1.1), we can compute this expectation by fitting the model for each possible value for R_{ij}, computing f for each such fit model, and taking their mean weighted by our belief about the probability of R_{ij} taking on that value: $\sum_{x\in\mathcal{X}} P(R_{ij} = x) f(R_{\mathcal{O},R_{ij}=x})$.

If the rating values are continuous or there are too many of them, we exploit the previously-mentioned fact that the marginal distributions of R_{ij} are approximately normal. We estimate $P(R_{ij} \le x)$ by a normal distribution and integrate with the trapezoid rule. We take 25 values of a evenly spaced between 0.001 and 0.999, and break up the integral over f at each ath quantile of the normal distribution.

In the variational setting, we choose to take the probability $P(R_{ij})$ by moment-matching a normal distribution to the variational approximation q's belief about R_{ij}. It would also be possible to use the MAP model, in which $R_{ij} \sim N(u_i^T v_j, \sigma^2)$, but our experiments suggested that q's belief performed slightly better.

In MCMC, with discrete output ratings, we estimate $P(R_{ij})$ by the MAP fit of a categorical distribution with a Dirichlet prior, where the prior is used to smooth out any probabilities that would otherwise be zero. (This prior could be computed based on the observed or expected distributions of the ratings for the full matrix; we simply use a flat prior where each rating obtains a pseudocount of 0.1.) For continuous outputs, we use the MLE.

Possible functions f include:

- For the Prediction task: the differential entropy of the predictions R, $\mathrm{H}[R \mid R_{\mathcal{O}}]$. In the variational setting we find an upper bound on the entropy via the determinant of its covariance matrix. This requires finding the determinant of $\mathrm{Cov}[R \mid R_{\mathcal{O}}] \in \mathbb{R}^{D(N+M)\times D(N+M)}$. To avoid constructing this large matrix, we could also use a rough standin for uncertainty of $\sum_{kl} \mathrm{Var}[R_{kl}]$; this is a flawed measure, but much easier to compute.

  In MCMC, we could employ high-dimensional nonparametric entropy estimators, but these are computationally intensive and have poor sample complexity. Instead, we assume R is matrix-normal and find the maximum likelihood estimates for the parameters $\widehat{\Sigma}, \widehat{\Omega}$ with the "flip-flop" algorithm of [3]; we then compute the MLE of the entropy based on their determinants.

- For the Model task: the differential entropy of the posterior over possible models, $\mathrm{H}[U, V \mid R_{\mathcal{O}}]$. This is simple to compute under our variational assumptions, but in MCMC has the same problems as the entropy of R. It is worth noting, however, that the matrix-normal assumption is generally much more reasonable for R than for (U, V); we therefore do not evaluate this method in the MCMC setting.

- For the Search task: The expected number of positives selected if we stop the search after one step, which is the optimal one-step lookahead algorithm for active search [5]. In this case, f for an element (i, j) is equal to the maximum probability of any queryable point *other* than (i, j) being positive, plus 1 if (i, j) was positive. For this strategy to be truly optimal, we must do complete lookahead and recurse over all $|\mathcal{X}| \cdot |\mathcal{P}|!$ branching possibilities, which is clearly infeasible.

  We could use a very similar f for the Magnitude Search task, replacing probabilities with magnitudes. We do not evaluate this method in this work due to its close similarity to the Search algorithm.

In our setting, unfortunately, the sheer number of queryable points makes lookahead methods extremely expensive. On a reasonably large problem, computing even one-step lookahead for each queryable point is infeasible. Practical implementations therefore need to subsample the queryable points, perhaps evaluating only points that an easier-to-compute heuristic finds most promising. Due to this expense, we evaluate the lookahead methods only on small synthetic datasets in this work.

6. NUMERICAL EXPERIMENTS

We will now present empirical evaluations of our active learning approaches on synthetic datasets, movie ratings, and drug discovery. We compare to the minimum-margin selection approach of [21] when possible; we do not compare to the work of [26] because there is no publicly available implementation. Lookahead criteria are evaluated only for the synthetic matrices of Section 6.1, due to their computational expense. The code and data used in these experiments are available from the supplement website.

For the regularization parameters of the variational approach, we used $\sigma^2 = 1$ and $\sigma_U^2 = \sigma_V^2 = 10^{-2}$. We found that when the number of observed elements is small, choosing parameters to maximize the likelihood (1) as suggested by [24] resulted in values that were far too small, even after adding a fairly strong log-normal hyperprior.

In MCMC sampling, we used the same hyperparameters as in [23]: $\sigma = \frac{1}{2}$, $\mu_0 = 0$, $\beta_0 = 2$, $W_0 = I$, and $\nu_0 = D$. All of the experiments presented here used a warmup of 100 samples to for NUTS adaptation and to approach a local maximum, followed by inference based on 200 samples. When computing lookaheads, we use a warmup of 50 followed by 100 samples.

We typically learn on a centered version of the data matrix, so that we can assume that U and V have mean 0. This helps make the priors more sensible and allows for easy initialization.

In both approaches, we initialized the parameters at random elements near the origin (for means) or the identity matrix (for covariances). After learning the label of a query point, we initialized optimization or sampling for the next step at the parameters obtained by the previous one. (In MCMC, we initialized at the sample from the previous step with the highest probability density.)

For the comparisons to MMMF, we used code from Nathan Srebro[3] which employs an SDP solver; we used regularization parameter $C = 1$ throughout. For higher-dimensional problems the SDP solver became quite slow; the direct formulation of [19] would probably be preferable.

[3] `ttic.uchicago.edu/~nati/mmmf/`

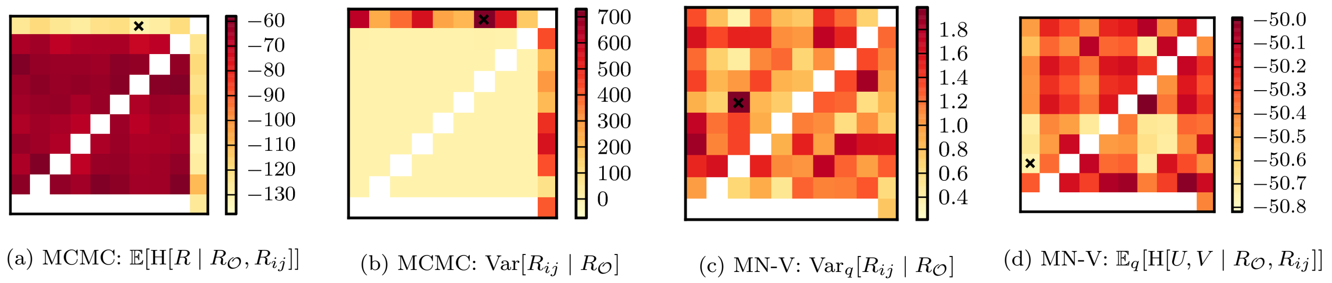


Figure 1: Selection criteria evaluated on a 10×10, rank 1 matrix. The x marks the point chosen by the criterion; white squares are known by the learner. (a) and (b) employ the MCMC framework, (c) and (d) the matrix-normal variational framework.

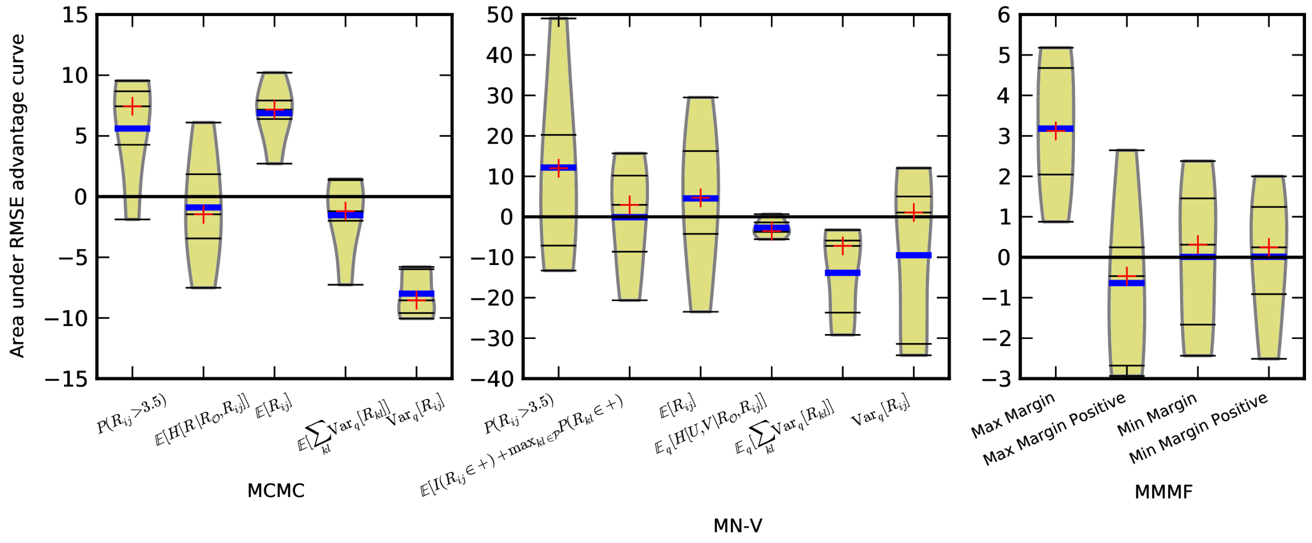


Figure 2: PREDICTION results from five runs of the 10×10 rank 4 synthetic experiment. The beanplots show areas under the curve of RMSEs for a given method minus the RMSE of random selection on the same data. Negative values mean that the method outperformed random selection.

6.1 Synthetic data

We will first evaluate our methods on small synthetic problems to see some characteristics of their performance.

6.1.1 Continuous with known utility

To motivate our selection criteria, we first consider a simple problem where the value of selecting points is known. Let us reconstruct a rank 1 matrix $R \in \mathbb{R}^{10 \times 10}$, where the off-diagonal elements and all but one element of the bottom row have already been observed (the white squares in Figure 1). The bottom-left 9×9 square is thus constrained perfectly, as the diagonal establishes the factor by which the bottom row must be multiplied. The rightmost column and top row are unknown, but learning any entry there will give us enough information to know the full matrix. Thus, picking an element in the bottom-left 9×9 square provides no information, while picking any element in the rightmost column or top row allows us to know the entire matrix perfectly.

Figure 1 shows our evaluation criteria on one such matrix, generated by sampling 10×1 U and V from a normal distribution with mean 10 and standard deviation 2.[4] Colors represent the value of the criterion at hand; the square with the black x is the best choice according to that criterion. We can see that the MCMC lookahead method of Figure 1a performs quite well, with a clear separation between the good and the bad choices. The method based on sample variance (Figure 1b) also performs well: all the bad choices are evaluated as bad, though the margin between good and bad points is much narrower. The variational criteria, by contrast, both seem essentially random; each one picks a useless point, indicating that the approximation is ineffective here.

[4]With small matrix sizes, the biconvexity of (2) often causes problems in gradient descent when the factors are zero mean. If all the known elements of a row and column are close to zero, there will be an asymptotic non-global maximum with some of the factors' signs flipped. This does not occur when the factor means are far from zero. On the other hand, the MCMC algorithm does poorly if started too far away from a local mean. In practical situations, normalizing the ratings to be zero mean is sufficient, but that makes this matrix no longer rank 1; we instead initialize the sampling at the MAP estimate from PMF and set $\mu_0 = 10$.

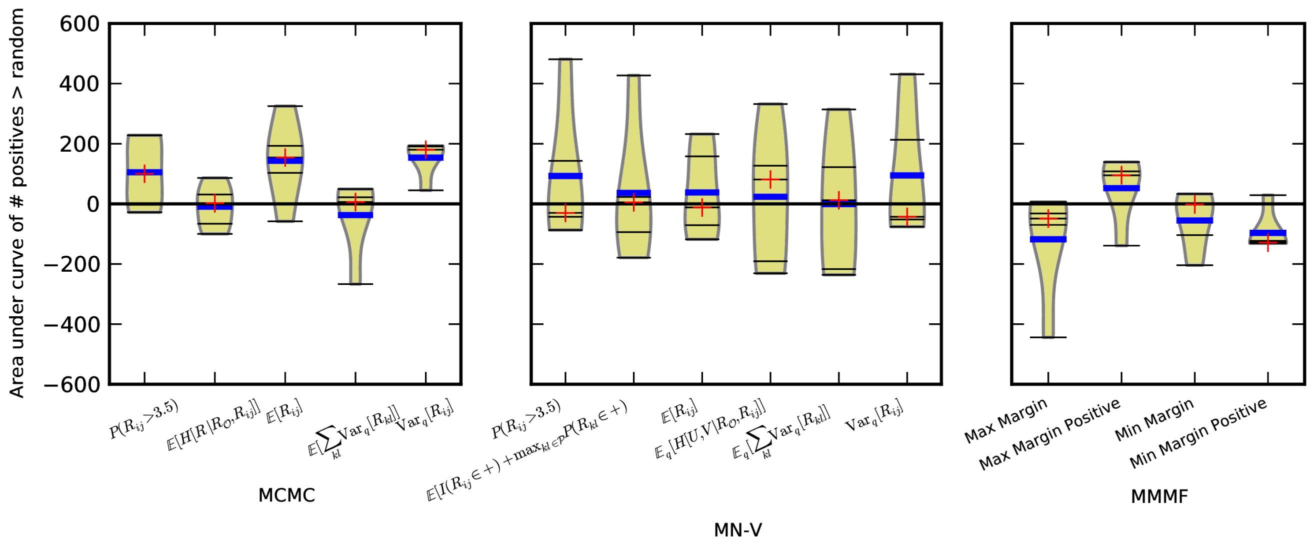


Figure 3: Search results from five runs of the 10×10 rank 4 synthetic experiment. The beanplots show areas under the curve of the number of positives selected along the active learning curve for a given method, minus the number of positives selected by random selection at the same point. Positive values indicate that the method outperformed random selection.

6.1.2 *Integer-valued*

We now turn to a slightly more realistic example: 10×10 matrices with integer values in the range 1 to 5, approximately of rank 4.[5] Figure 2 shows the mean advantage (in terms of RMSE) of each method compared to random selection over the course of the full evaluation. That is, we draw the curve where the horizontal axis is the number of points queried and the vertical axis the RMSE for selection with the given criterion and random selection; Figure 2 then shows the difference between the area under each curve.

MCMC methods and variational approaches that do lookahead based on the MAP belief about rating distributions seem to all do somewhat better than random. In MCMC methods, criteria related to the Search goals tend to hurt, while uncertainty sampling clearly helps and the lookahead methods help somewhat. Variational methods fare more or less similarly, though with a wider spread. In this case, MMMF active learning does not appear useful, though it is worth noting that even random MMMF outperforms the best of the PMF-based methods here. Figure 3 shows the same analysis for the Search criterion, treating 4 or 5 as positive and 1 through 3 as negative; here we see that the MCMC Search methods seem to help, the variational methods may as well but less consistently, and the MMMF max margin positive method helps only a little.

6.2 Movielens

The Movielens-100k dataset consists of 100,000 ratings of 1682 movies by 943 users of `movielens.org`. Ratings range from 1 (worst) to 5 (best). We ran on a subset consisting of the 50% of users with the most ratings and enough of their most-rated movies to cover 70% of their ratings, which resulted in a set of 472 users and 413 movies. There are 58,271 ratings in this subset, so that just under 30% of the matrix is known, as opposed to the full dataset where only 6% of the ratings are known.

In our experiments, we started from a "near-scratch" learning state where 5% of the ratings are known. The subset of known entries is chosen randomly in such a way that at least one entry is known in each row and each column. We selected a test set of another 5% of the known ratings uniformly from the unknown ratings, and then ran our MCMC learning algorithms for 200 steps, allowing the model to update its parameters and choose any element not in either the known or test sets at each step. 200 steps is insufficient to see any improvement in the RMSE on this larger model, but Figure 4 shows the number of positives selected as the algorithm proceeds. (Error bars are not shown for clarity of presentation, but each individual run looked similar.)

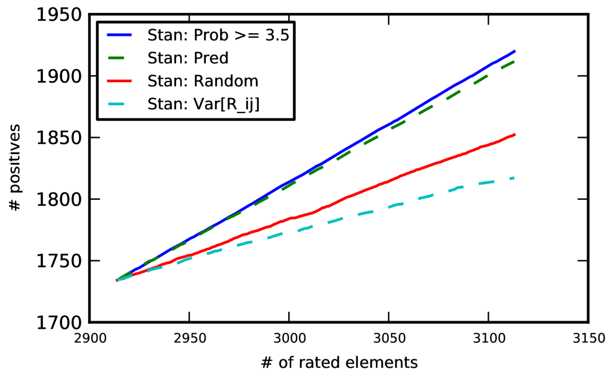
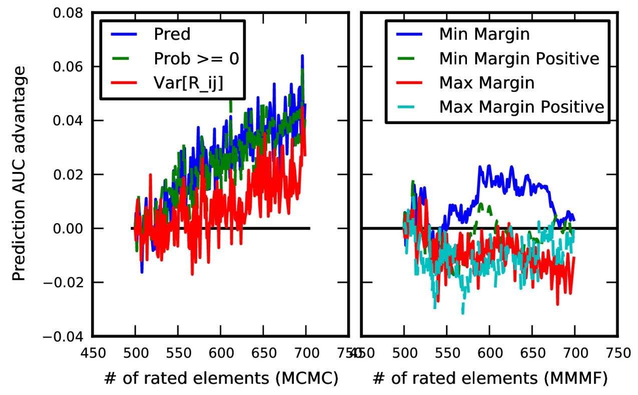
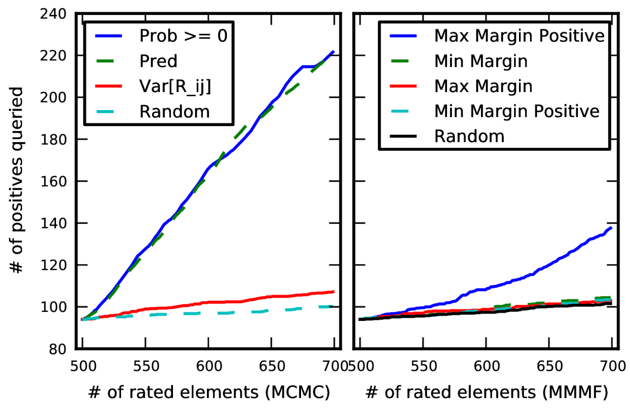


Figure 4: Mean numbers of positive elements selected in five independent runs on the 58,000-rating Movielens subset, with a rank-15 model.

[5]These matrices are constructed by choosing a random matrix with values 1-5, reconstructing based on the first four singular values, and then rounding to be 1-5 valued.



(a) The mean difference between the prediction AUC of a method and the prediction AUC achieved by random selection on the data.



(b) Mean number of positives queried. Error bars are not shown for clarity, but each of the runs had similar slopes.

Figure 5: Five runs on the 94 × 425 DrugBank subset with a rank-10 model.

6.3 DrugBank

As mentioned previously, collaborative prediction algorithms are applicable to a large number of domains outside those of recommender systems. One such possibility is the task of predicting interactions between drugs and various targets for those drugs, including diseases, genes, proteins, and organisms. The DrugBank dataset [16] is a comprehensive source of this information, containing information on over 6,000 drugs and 4,000 targets. We extracted only the presence or absence of interactions into a matrix with drugs as rows and targets as columns. Only positive interactions are present in the database, consisting of about one in 2,000 possible pairs. We therefore assumed that all interactions not listed in the database truly do not occur.

We used a subset of this matrix containing 94 drugs and 425 targets, such that each drug had at least one interaction with a present target (maximum 59, median 16) and each target had interactions with multiple drugs (most had 2 or 3; some had as many as 22). This matrix contains 1,521 interactions and 38,429 non-interactions.

We chose an initial training set containing exactly one interaction for each drug, and 406 negatives selected such that each target had at least one initially known point. We chose a test set of 500 positives and 1,000 negatives uniformly from the remaining data, and as before ran the learning process for 200 steps. We used a model of rank 20 and did five independent runs (which used the same 94 × 425 data subset but different training and test sets).

Because of the binary nature of the problem and the skewed test distribution, we evaluate not on RMSE but on the area under the ROC curve of binary classifier defined by the predictions (on the test set). Figure 5a shows the mean of these AUCs over the learning process for various MCMC selection criteria and for the MMMF criterion. We can see that all three of our active learning criteria strongly help boost the ROC curve of the predictions in the MCMC setting, while the assistance due to the MMMF active learning approach of [27] is small if present at all. In this case, where positives are quite rare (around 2% of the points available to query), it seems that discovering an element is positive is likely to convey much more information than finding an element is negative, so it is unsurprising that our SEARCH-oriented heuristics outperformed uncertainty sampling in terms of performance. It is also worth noting that the baseline performance of the MCMC approach (e.g. with random selection) is substantially superior to that of MMMF.

Figure 5b shows the effectiveness of various criteria for finding positives in the data. We see that the MCMC-based criteria far outstrip the MMMF-based ones in their rate of finding positives, though the max-margin positive criterion is better than random.

7. DISCUSSION

We gave approaches for active learning and active search in the PMF framework with four goals (PREDICTION, MODEL, MAGNITUDE SEARCH, and SEARCH). We examined these criteria on synthetic examples, and then showed the effectiveness of the non-lookahead versions on two real-world datasets. On the important problem of understanding and seeking out interactions in the drug discovery process, our methods greatly outperformed the MMMF-based methods in both PREDICTION and SEARCH.

We found that variational approaches based on a matrix-normal factorization of the posterior were both computationally expensive and did not perform especially well. It seems that the MCMC approaches considered here, or the fullly-factorized variational approach of [26], are superior.

Many potential enhancements to this model are possible. Perhaps most important is a method for choosing elements to examine in looakahead criteria. It is also worth noting that our methods may be applied almost unchanged to models which incorporate side information into matrix factorization through Gaussian Process priors (e.g. [1, 7, 32]). Combining the power of collaborative filtering with that of feature-based methods might yield an effective method for guiding experimental processes such as seeking out drug-target interactions or protein-protein interactions.

8. REFERENCES

[1] R. P. Adams, G. E. Dahl, and I. Murray. Incorporating side information in probabilistic matrix factorization with Gaussian processes. 2010.

[2] C. Boutilier, R. S. Zemel, and B. Marlin. Active collaborative filtering. In *UAI*. Morgan Kaufmann Publishers Inc, 2002.

[3] P. Dutilleul. The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999.

[4] A. Eriksson and A. Van Den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm. *CVPR*, pages 771–778, 2010.

[5] R. Garnett, Y. Krishnamurthy, D. Wang, J. Schneider, and R. Mann. Bayesian Optimal Active Search on Graphs. In *Ninth Workshop on Mining and Learning with Graphs*, 2011.

[6] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. Mann. Bayesian optimal active search and surveying. In *ICML*, 2012.

[7] M. Gönen, S. A. Khan, and S. Kaski. Kernelized Bayesian matrix factorization. *arXiv.org*, stat.ML, 2012.

[8] M. D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, In press.

[9] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. *International Joint Conference on Artificial Intelligence*, 16:688–693, 1999.

[10] L. Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12:134–139, 1918.

[11] R. Jin and L. Si. A Bayesian approach toward active learning for collaborative filtering. *UAI*, pages 278–285, 2004.

[12] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Active learning for aspect model in recommender systems. *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 162–167, 2011.

[13] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Non-myopic active learning for recommender systems based on matrix factorization. *Information Reuse and Integration (IRI)*, pages 299–303, 2011.

[14] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Towards optimal active learning for matrix factorization in recommender systems. In *Tools with Artificial Intelligence (ICTAI)*, pages 1069–1076, 2011.

[15] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Exploiting the characteristics of matrix factorization for active learning in recommender systems. In *RecSys '12*, 2012.

[16] C. Knox, V. Law, T. Jewison, P. Liu, S. Ly, A. Frolkis, A. Pon, K. Banco, C. Mak, V. Neveu, Y. Djoumbou, R. Eisner, A. C. Guo, and D. S. Wishart. DrugBank 3.0: a comprehensive resource for 'omics' research on drugs. *Nucleic Acids Research*, 39(Database):D1035–D1041, 2010.

[17] R. F. Murphy. An active role for machine learning in drug development. *Nature Publishing Group*, 7(6):327–330, 2011.

[18] R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, Handbooks of Modern Statistical Methods. Chapman & Hall/CRC, 2011.

[19] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719. 2005.

[20] F. Ricci, L. Rokach, B. Shapira, and P. Kantor. *Recommender Systems Handbook*. Springer, 2011.

[21] I. Rish and G. Tesauro. Active collaborative prediction with maximum margin matrix factorization. *Inform. Theory and App. Workshop*, 2007.

[22] N. Rubens, D. Kaplan, and M. Sugiyama. Active learning in recommender systems. In P. Kantor, F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 735–767. Springer, 2011.

[23] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, pages 880–887, 2008.

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

[25] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030, 2010.

[26] J. Silva and L. Carin. Active learning for online Bayesian matrix factorization. In *KDD*, 2012.

[27] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, volume 17, pages 1329–1336, 2005.

[28] Stan Development Team. Stan: A C++ library for probability and sampling, version 1.1, 2013.

[29] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.

[30] X. Yang, H. Steck, Y. Guo, and Y. Liu. On top-k recommendation using social networks. In *RecSys '12*, 2012.

[31] K. Yu, A. Schwaighofer, and V. Tresp. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. *UAI*, pages 616–623, 2002.

[32] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SIAM Data Mining*, pages 403–414, 2012.

9. ACKNOWLEDGMENTS

This work was funded in part by the National Science Foundation under grant NSF-IIS0911032 and the Department of Energy under grant DESC0002607.

APPENDIX

We will now present the closed forms of the expectations required to compute (2) and the selection criteria. All of these expectations are the products of variables which are part of a joint multivariate normal distribution. For simplicity of presentation, we will consider this joint distribution to be over vectors X. Note also that most of these formulae assume that Σ is positive definite.

The two-term expectation for not necessarily distinct indices a, b is

$$\mathbb{E}[X_a X_b] = \mathbb{E}[X_a]\,\mathbb{E}[X_b] + \mathrm{Cov}[X_a, X_b] = \mu_a\mu_b + \Sigma_{a,b}. \quad (3)$$

We can calculate the four-term expectation of the product of distinct indices a, b, c, d by the following formula, derived from Isserlis' Theorem [10]:

$$\begin{aligned}\mathbb{E}\left[X_a X_b X_c X_d\right] &= \mu_a\mu_b\mu_c\mu_d \\ +\mu_c\mu_d\Sigma_{ab}+\mu_b\mu_d\Sigma_{ac}&+\mu_b\mu_c\Sigma_{ad}+\mu_a\mu_d\Sigma_{bc}+\mu_a\mu_c\Sigma_{bd}+\mu_a\mu_b\Sigma_{cd} \\ &+ \Sigma_{ab}\Sigma_{cd} + \Sigma_{ac}\Sigma_{bd} + \Sigma_{ad}\Sigma_{bc}. \quad (4)\end{aligned}$$

We also need to calculate $\mathbb{E}[X_a^2 X_b^2]$ in (2). This is, for $\Sigma_{aa}, \Sigma_{bb} > 0$ and positive-definite Σ:

$$\begin{aligned}\mathbb{E}[X_a^2 X_b^2] &= 4\mu_a\mu_b\Sigma_{ab} + 2\Sigma_{ab}^2 + \left(\mu_a^2 + \Sigma_{aa}\right)\left(\mu_b^2 + \Sigma_{bb}\right) \quad (5) \\ &= 4\,\mathbb{E}[X_a X_b] + 2\,\mathrm{Cov}[X_a, X_b]^2 + \mathbb{E}[X_a^2]\,\mathbb{E}[X_b^2].\end{aligned}$$

We will also need this identity:

$$\begin{aligned}\mathbb{E}[X_a^2 X_b X_c] &= (\mu_a^2 + \Sigma_{aa})(\mu_b\mu_c + \Sigma_{bc}) \\ &+ 2\mu_a\mu_c\Sigma_{ab} + 2\mu_a\mu_b\Sigma_{ac} + 2\Sigma_{ab}\Sigma_{ac}. \quad (6)\end{aligned}$$

We can use (3-5) in (2) to calculate $\mathrm{KL}(q\|p)$ for a given μ and Σ. Taking partial derivatives in order to carry out gradient descent is then straightforward; we choose to do so with respect to a triangular half of Σ, so that Σ_{ab} and Σ_{ba} are considered the same variable when taking derivatives. The gradient of $\ln(\det(\Sigma))$ in this case works out to be $\Sigma^{-1}+\Sigma^{-T}$ minus the diagonal elements of Σ^{-1}.

Many of our selection criteria require an expression for $\mathrm{Var}[R_{ij}|R_\mathcal{O}]$. We have:

$$\begin{aligned}\mathrm{Var}[R_{ij} \mid R_\mathcal{O}] &= \mathbb{E}[\mathrm{Var}[R_{ij} \mid U, V] \mid R_\mathcal{O}] + \mathrm{Var}[\mathbb{E}[R_{ij} \mid U, V] \mid R_\mathcal{O}] \\ &= \mathbb{E}[\sigma^2] + \mathrm{Var}[u_i^T v_j \mid R_\mathcal{O}] \\ \mathrm{Var}[u_i^T v_j \mid R_\mathcal{O}] &= \mathrm{Var}\left[\sum_{k=1}^{D} U_{ki}V_{kj} \mid R_\mathcal{O}\right] \\ &= \sum_{k=1}^{D}\sum_{l=1}^{D} \mathrm{Cov}[U_{ki}V_{kj}, U_{li}V_{lj} \mid R_\mathcal{O}] \\ &= \sum_{k=1}^{D}\sum_{l=1}^{D} \mathbb{E}\left[U_{ki}V_{kj}U_{li}V_{lj}|R_\mathcal{O}\right] - \mathbb{E}\left[U_{ki}V_{kj}|R_\mathcal{O}\right]\mathbb{E}\left[U_{li}V_{lj}|R_\mathcal{O}\right]. \\ & \quad (7)\end{aligned}$$

The first term of (7) will require Equations (4) to (6), depending on how many of the labels are equal. $\mathrm{Cov}[R_{ij}, R_{kl} \mid R_\mathcal{O}]$ is similar.

Since the vectorization of a matrix normal distribution is simply a subset of a full multivariate normal distribution, the expectations in the matrix normal distribution are quite similar. We simply need to replace Σ_{ab} with $\Sigma_{ij}\Omega_{kl}$, where a refers to the index (i, j) and b to the index (k, l).