Large-scale Collaborative Prediction Using a Nonparametric Random Effects Model

Kai Lu  
*NEC Labs*

John Lafferty  
*Carnegie Mellon University*

Shenghuo Zhu  
*NEC Labs*

Yihong Gong  
*NEC Labs*

Follow this and additional works at: [http://repository.cmu.edu/compsci](http://repository.cmu.edu/compsci)
Large-scale Collaborative Prediction
Using a Nonparametric Random Effects Model

Kai Yu
John Lafferty
Shenghuo Zhu
Yihong Gong
† NEC Laboratories America, Cupertino, CA 95014 USA
‡ Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract
A nonparametric model is introduced that allows multiple related regression tasks to take inputs from a common data space. Traditional transfer learning models can be inappropriate if the dependence among the outputs cannot be fully resolved by known input-specific and task-specific predictors. The proposed model treats such output responses as conditionally independent, given known predictors and appropriate unobserved random effects. The model is nonparametric in the sense that the dimensionality of random effects is not specified a priori but is instead determined from data. An approach to estimating the model is presented uses an EM algorithm that is efficient on a very large scale collaborative prediction problem. The obtained prediction accuracy is competitive with state-of-the-art results.

1. Introduction
In transfer learning and multi-task learning, the observed responses of interest can be seen as relational measurements under a pair of heterogeneous conditions. If there are $M$ tasks and $N$ shared observations, the standard regression model forms $M$ separate estimates

$$Y_{ij} = \mu + m_i(x_i, z_j) + \epsilon_{ij},$$

where now $m_{ij} = m(x_i, z_j)$ is a regression function based on task-specific predictors $x_i \in X$. In order to directly model the dependency between tasks, a multi-task Gaussian process approach (Yu et al., 2007; Bonilla et al., 2008) may assume

$$m \sim \text{GP}(0, \Omega \otimes \Sigma)$$

where $\Sigma(z_j, z_j'; \theta_\Sigma) \succ 0$ is a covariance function among inputs and $\Omega(x_i, x_i'; \theta_\Omega) \succ 0$ a covariance function among tasks, which means $\text{Cov}(m_{ij}, m_{i'j'}) = \Omega_{ii'} \Sigma_{jj'}$. The parameters $\theta_\Sigma$ and $\theta_\Omega$ can be estimated, for example, by maximizing the marginalized likelihood.

The model is nonparametric in the sense that $m$ lives in an infinite-dimensional function space; it can be more appropriate for capturing complex dependencies than a fixed-dimension model. But this flexibility comes at a computational price. If $Y \in \mathbb{R}^{M \times N}$ is the random response matrix, including those elements $Y_{ij}$ that are observed and those that need to be predicted, the computational complexity of inference is $O(M^2N^3)$. It is thus infeasible to do inference if $M$ and (or) $N$ are large. For example, in our experiments on the Netflix dataset, $M = 480, 189$ and $N = 17, 770$.

Just as significantly, from a modeling perspective this model may not be suitable in the situation where the responses $Y_{ij}$ cannot be fully explained by the predictors $x_i$ and $z_j$ alone. In such a case the conditional independence assumption

$$p(Y | m, x, z) = \prod_{i,j} p(Y_{ij} | m_i, x_i, z_j)$$

implied by Eq. (1) is invalid. Such is often the case for relational or networked observations. For instance,
in the application of predicting users’ unobserved ratings on movies, the observed ratings themselves are often more informative than the users’ demographic attributes and the movies’ genre attributes.

Motivated by these considerations, in this paper we propose a transfer learning approach that satisfies three desiderata. The model (1) is nonparametric with sufficient flexibility, (2) is appropriate for dependent responses that are not solely conditioned on the given predictors, and (3) supports efficient learning on very large scale data.

2. A Random Effects Model

In statistics, a random effects model is a kind of hierarchical linear model that allows dependent observations occurring in repeated or multilevel structures. The scenario of transfer learning matches such a form; in particular, the population is grouped together according to tasks, and each group repeatedly generates random observations.

A natural extension of the previous model is to assume the $Y_{ij}$ are conditionally independent, given $m_{ij} \equiv m(x_i, z_j)$ and additional additive random effects $f_{ij}$. The model becomes

$$Y_{ij} = \mu + m_{ij} + f_{ij} + \epsilon_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$$

where the distribution of $m$ and $f$ determines the dependence among the variables $Y_{ij}$. One may wish to choose a form of $f$ that is appropriate to explain the data dependence, for example, introducing latent variables $u_i$ and $v_j$ such that $f_{ij}$ has a parametric form $f(u_i, v_j; \theta_f)$, as suggested in (Hoff, 2005).

A nonparametric random effects model may assume $f \sim \mathcal{GP}(0, \Delta \otimes \Gamma)$, where the covariances $\Delta$ and $\Gamma$ are parameters to be determined from data. While this model is natural and flexible, it is subject to the large computational limitations discussed above. We address these limitations by imposing additional structure and developing an efficient estimation procedure.

2.1. Our Model

In the following we introduce a refinement of the above model that has specific structure, but exhibits some attractive conceptual and computational properties. We again take a hierarchical linear model:

$$Y_{ij} = \mu + m_{ij} + f_{ij},$$

but now assume that

$$m \sim \mathcal{GP}(0, \Omega_0 \otimes \Sigma),$$

$$f_i \overset{\text{ind}}{\sim} \mathcal{GP}(0, \tau \Sigma), \quad i = 1, \ldots, M.$$
process,
\[
\Omega \sim \text{IWP}(\kappa, \Omega_0 + \tau \delta),
\]
\[
m \sim \text{GP}(0, \Omega \otimes \Sigma_0),
\]
\[
f_j \overset{\text{iid}}{\sim} \text{GP}(0, \Lambda\Omega), \quad j = 1, \ldots, N,
\]
where similarly \(f_j\) denotes the function values of the random effects on case \(j\), across tasks. It turns out that the two models are equivalent — in both cases, if we compute the marginal distribution of \(Y\), namely

1. \[f(Y|m, \tau, \Sigma)p(m|\Sigma, \Omega_0)p(\Sigma|\Sigma_0, \lambda, \kappa) \, dm \, d\Sigma;\]
2. \[f(Y|m, \lambda, \Omega)p(m|\Omega, \Omega_0)p(\Omega|\Omega_0, \tau, \kappa) \, dm \, d\Omega;\]

we obtain exactly the same closed-form distribution
\[Y \sim \text{MTP}(\kappa, 0, (\Omega_0 + \tau \delta), (\Sigma_0 + \lambda \delta)),\]
where MTP defines a matrix-variate Student-t distribution. That is, any subset of matrix values \(Y \in \mathbb{R}^{M \times N}\) follows a matrix-valued Student-t distribution (Gupta & Naga, 1999). The proof of the equivalence is sketched in the Appendix.

2.3. Discussion

The model is nonparametric at two levels. First, the covariance functions \(\Sigma\) and \(\Omega\) are nonparametric, because they both live in infinite dimensional function spaces. Moreover, they do not have explicit parametric forms. Second, the dimensionality of any finite \(Y\) is not specified a priori, but instead increases with the observation size.

The equivalence between the two generative processes reveals an appealing symmetry in the model. One can either view the model as adapting the free-form covariance function \(\Sigma\) to the data, or equivalently, adapting the covariance \(\Omega\) to the data. Therefore the model is somewhat similar to those that deal with both \(\Sigma\) and \(\Omega\) simultaneously, such as the multi-task Gaussian process discussed in Sec. 1. However, as we shall see in Sec. 3, the model potentially allows us to avoid the prohibitive computational cost \(O(M^3N^3)\), without sacrificing significant flexibility.

The model can easily incorporate group-specific effects, as do most classical random-effects models. For example, one can extend the feature representation by adding a constant term \(\sqrt{c}\), such that \(\Omega_0(x_i, x_i') = \langle \phi(x_i), \phi(x_i') \rangle + c\). Then \(m\) can be parameterized as
\[m_{ij} = \langle \phi(x_i), \beta_j \rangle + \gamma_j\]
where \(\gamma_j\) is the \(j\)-th element of \(\gamma \sim \text{GP}(0, c\Sigma)\), i.e., the column-specific random effects. Similarly, with \(\Sigma_0(z_j, z_j') = \langle \varphi(z_j), \varphi(z_j') \rangle + b\), one can deal with the row-specific random effects as well.

3. Large-scale Inference and Learning

Given the model and some observed elements, denoted by \(Y_O\), we can make predictions by computing the conditional expectation on unseen elements. To this end, we need to compute the posterior distribution \(p(Y|Y_O, \theta)\), where \(\theta\) includes \(\Sigma_0, \Omega_0, \lambda, \tau\) and \(\kappa\).

It is highly challenging to apply nonparametric models to large-scale problems. We will present several algorithms, each with an analysis of its computational complexity in the context of the Netflix collaborative prediction problem, which contains \(M = 480,189\) users and more than 100 millions ratings on \(N = 17,770\) movies. Our estimates of computing time are based on a 2.66GHz Intel PC with 3.0GB memory.

3.1. Modeling Large-scale Data

Assuming \(M \gg N\), e.g., the Netflix case, it is computationally more convenient to work with the row-wise model. Since the computation is always on a finite matrix \(Y \in \mathbb{R}^{M \times N}\), including those elements \(Y_{ij}\) that are observed and those that need to be predicted, and since the model is consistent under marginalization, in the following we state the model on finite matrices
\[
\Sigma \sim \text{IW}(\kappa, \Sigma_0 + \lambda I_N),
\]
\[
m \sim N(0, \Omega_0 \otimes \Sigma),
\]
\[
Y_i \sim N(m_i, \tau \Sigma), \quad i = 1, \ldots, M
\]
where IW defines a finite inverse-Wishart distribution, \(\Sigma\) and \(\Sigma_0\) are both \(N \times N\) covariance matrices, \(\Omega_0\) is \(M \times M\), \(m_i\) is the \(i\)-th row of \(m\), and \(Y_i\) is the \(i\)-th row of \(Y\).

To further facilitate a large-scale implementation, we assume \(\Omega_0\) has finite dimensions. i.e., \(\Omega_0(x, x') = \phi(x)^\top \phi(x')\), with \(\phi : \mathcal{X} \rightarrow \mathbb{R}^p\). For large-scale data, it is reasonable to assume \(M \gg N \gg p\). The finite-dimension assumption is mainly due to computational concerns for large-scale problems, though the model itself does not require this. However, the sampled latent covariance functions \(\Sigma\) and \(\Omega\) are always infinite-dimensional, and so is the function \(Y\). Therefore the overall model is still nonparametric.

Without loss of generality, we let \(\Omega_0(x, x') = (p x_i, p x_i')\). In this case, the random effects \(m\) follow
\[
m_i = \beta^\top x_i, \quad \beta_k \overset{\text{iid}}{\sim} N(0, \Sigma), \quad k = 1, \ldots, p
\]
where \(\beta\) is a \(p \times N\) random matrix, and \(\beta_k\) is its \(k\)-th row. Then the model becomes
\[
\Sigma \sim \text{IW}(\kappa, \Sigma_0 + \lambda I_N),
\]
\[
\beta \sim N(0, I_p \otimes \Sigma),
\]
\[
Y_i \sim N(\beta^\top x_i, \tau \Sigma), \quad i = 1, \ldots, M
\]
3.2. Approximate Inference – Gibbs Sampling

Based on the joint distribution \( p(Y, \beta, \Sigma | \theta) \) described by the model, the prediction can be approximated by a Markov chain Monte Carlo method

\[
\int p(Y | Y_{O}, \theta) Y dY \approx \frac{1}{T} \sum_{t=1}^{T} \delta(Y; Y^{(t)})
\]

where the samples \( Y^{(t)} \), together with \( \beta^{(t)}, \Sigma^{(t)} \) are i.i.d. drawn from \( p(Y, \beta, \Sigma | Y_{O}, \theta) \). For Gibbs sampling, whose details are omitted here due to space limitation, the computational cost of one Gibbs iteration is roughly

\[
\mathcal{O} \left( \sum_{i=1}^{M} N_{i}^{3} + 2 \left( \sum_{i=1}^{M} N_{i}^{2} \right) + 2M N^{2} + N^{3} \right),
\]

where we count mainly multiplications, and ignore the impact of factor \( p \), since \( M \gg N \gg p \). Compared with \( \mathcal{O}(M^3 N^3) \) for the direct Gaussian process, this is a dramatically reduced complexity. However, since \( M \) is large, the cost \( \mathcal{O}(2MN^2) \) is still prohibitive—each Gibbs iteration will take hundreds of hours on the Netflix data.

3.3. Approximate Inference – EM

We first introduce some necessary notation. Let \( J_i \subset \{1, \ldots, N\} \) be the index set of the \( N_i \) observed elements in the row \( Y_{i} \), and \( \Sigma_{[i,j]} \in \mathbb{R}^{N \times N_i} \) be the matrix obtained by keeping the columns of \( \Sigma \) indexed by \( J_i \). Then let \( M_{[j,3]} \) be obtained from \( \Sigma_{[i,j]} \) by further keeping only the rows indexed by \( J_i \). We can similarly define \( \Sigma_{[i,j]} \). Alteratively, the conditional expectation can be approximated by

\[
\int p(Y | Y_{O}, \beta, \Sigma) p(\beta, \Sigma | Y_{O}, \theta) Y d\beta d\Sigma dY \approx \int p(Y | Y_{O}, \tilde{\beta}, \tilde{\Sigma}) Y dY
\]

where \( \tilde{\beta} \) and \( \tilde{\Sigma} \) are the maximum a posteriori estimates

\[
\tilde{\beta}, \tilde{\Sigma} = \arg \max_{\beta, \Sigma} p(\beta, \Sigma | Y_{O}, \theta).
\]

When the number of observations is sufficiently large that the posterior distribution is concentrated around its mode, the above approximation can be tight. The expectation-maximization (EM) algorithm is a popular approach to finding the mode, alternating the following two steps:

- E-step: compute the posterior distribution of \( Y \) given the current \( \beta, \Sigma \)

\[
Q(Y) = \prod_{i=1}^{M} p(Y_i | Y_{O_i}, \beta, \Sigma) = \prod_{i=1}^{M} N(Y_i | v_i, C_i),
\]

where the sufficient statistics are

\[
\begin{align*}
\mathbf{m}_i &= \beta^T x_i, \\
v_i &= \mathbf{m}_i + \Sigma_{[i,j]}^{-1} (Y_{i,j} - \mathbf{m}_{i,j})^T, \\
C_i &= \tau \Sigma - \tau \Sigma_{[i,j]}^{-1} \Sigma_{[i,j]}^{-1} \Sigma_{[j,:]}.
\end{align*}
\]

- M-step: optimize \( \beta \) and \( \Sigma \)

\[
\tilde{\beta}, \tilde{\Sigma} = \arg \min_{\beta, \Sigma} \left\{ \mathbb{E}_{Q(Y)} [ - \log p(Y, \beta, \Sigma | \theta) ] \right\}
\]

and then let \( \beta \leftarrow \tilde{\beta}, \Sigma \leftarrow \tilde{\Sigma} \).

It turns out that \( \tilde{\beta} \) can be nicely decoupled from \( \tilde{\Sigma} \), and has a closed form:

\[
\tilde{\beta} = (\mathbf{x}^T x + \tau I_p)^{-1} \mathbf{x}^T \mathbf{v}.
\]

Plugging \( \tilde{\beta} \) into the optimization cost, we have

\[
\tilde{\Sigma} = \frac{\tau^{-1} [C - \mathbf{v}^T \mathbf{x}(\mathbf{x}^T x + \tau I_p)^{-1} \mathbf{x}^T \mathbf{v}] + \Sigma_0 + \lambda I_N}{M + 2N + p + \kappa}
\]

where \( C = \sum_{i=1}^{M} \mathbf{C}_i + \mathbf{v}_i \mathbf{v}_i \). Further detail on the M-step can be found in the Appendix.

The overall cost of a single EM iteration is

\[
\mathcal{O} \left( \sum_{i=1}^{M} N_{i}^{3} + \left( \sum_{i=1}^{M} N_{i}^{2} \right) N + \left( M + \sum_{i=1}^{M} N_{i} \right) N^{2} \right).
\]

The computational cost seems to be even higher than that of Gibbs sampling, since \( N^2 \) is multiplied by the large factor \( (M + \sum_{i=1}^{M} N_{i}) \), and \( N \) is a large multiplicative factor of \( \sum_{i=1}^{M} N_{i}^{2} \). We estimate that, in a single EM step, excluding the term \( \mathcal{O}(\sum_{i=1}^{M} N_{i}^{3}) \), the computation would take several thousands of hours for the Netflix problem.

3.4. Efficient EM Implementation

In order to introduce a faster algorithm, we now fix some notation. Let \( U_i \in \mathbb{R}^{N \times N_i} \) be a column selection operator, whose \( (s, t) \) element is one, if the index \( s \) equals to the \( t \)-th element of \( J_i \), otherwise zero, so that \( \Sigma_{[i,j]} = \Sigma U_i \) and \( \Sigma_{[i,j]} = U_i^T \Sigma U_i \). Furthermore, \( U_i \Sigma_{[j,j]} U_i^T \) restores a \( N \times N \) sparse matrix where the elements of \( \Sigma_{[j,j]} \) is placed back to their original positions in \( \Sigma \).
The major cost is from the E-step that computes the sufficient statistics \( \nu_i \) and \( C_i \) for every \( i \). Our key observation is that, the M-step only needs \( C = \sum_{i=1}^{M} C_i + v^T v \) and \( v^T x \) from the previous E-step. To obtain them, in fact it is unnecessary to compute \( v_i \) and \( C_i \) at all. For example, by using \( \Sigma_{[i:.J_i]} = \Sigma U_i \), we have

\[
\sum_{i=1}^{M} C_i = \sum_{i=1}^{M} \left( \tau \Sigma - \tau \Sigma_{[i:.J_i]} \Sigma_{[J_i:.]}^{-1} \Sigma_{[i:.]} \right) = \sum_{i=1}^{M} \left( \tau \Sigma - \tau \Sigma U_i \Sigma_{[J_i:.]}^{-1} U_i^T \Sigma \right) = \tau M \Sigma - \tau \Sigma \left( \sum_{i=1}^{M} U_i \Sigma_{[J_i:.]}^{-1} U_i^T \right) \Sigma
\]

The statistics required by the M-step are computed as \( E \)-step as the following

\[
v^T v = \beta^T x^T x \beta + \Sigma \left( \sum_{i=1}^{M} U_i a_i a_i^T U_i^T \right) \Sigma + \Sigma \left( \sum_{i=1}^{M} U_i a_i x_i^T \right) \Sigma \]

where \( a_i = \Sigma_{[J_i:.]}^{-1} (Y - m)_{[i:.]} \).

Before running the EM iterations, we pre-compute \( x^T x \) and \( (x^T x + \tau I_p)^{-1} \), since they are both repeatedly used in the algorithm. Then we implement the E-step as the following

**Reset**

\[ A_1 \in \mathbb{R}^{N \times N}, A_2 \in \mathbb{R}^{N \times p}; \]

**For** \( i = 1, \ldots, M; \)

\[ P = \Sigma_{[J_i:.]}^{-1}; \]

\[ a = P (Y_{[i:.J_i]} - x_i^T \beta_{[J_i:.]} )^T; \]

\[ A_1 \leftarrow A_1 + U_i (a a^T - \tau P) U_i^T; \]

\[ A_2 \leftarrow A_2 + U_i a x_i^T; \]

**End.**

The statistics required by the M-step are computed as

\[
C = \tau M \Sigma + \beta^T x^T x \beta + \Sigma A_1 \Sigma + \Sigma A_2 \beta + \beta^T A_2 \Sigma \]

\[
v^T x = \beta^T x^T x + \Sigma A_2 \]

With this new implementation, the major cost of a single EM step is

\[
O \left( \sum_{i=1}^{M} N_i^3 + 2 \sum_{i=1}^{M} N_i^2 + 2 N^3 \right).
\]

This is a dramatic efficiency improvement; for each EM iteration on Netflix, the original thousands of hours are now reduced to several hours. The remaining major cost \( O(\sum_{i=1}^{M} N_i) \) comes from the matrix inversion \( \Sigma_{[J_i:.]}^{-1} \). Occasionally a row may have an unusually large number of observations, so we truncate \( N_i \) by randomly selecting 2000 observations if \( N_i > 2000 \). In the end, a single EM step takes about 5 hours, and the whole optimization typically converges in about 30 steps.

As mentioned in Sec. 2.1, considerable computation is also saved by avoiding to separately model the noise \( \epsilon_{ij} \). Otherwise, one has to spend \( O(\sum_{i=1}^{M} (2N_i^2 + 2N_i^3)) \) operations to compute the noise statistics in a single EM step, which amounts to about 10 hours of computing time.

### 4. Related Work

There is a large body of research on multi-task learning using Gaussian processes, including those that learn the covariance \( \Sigma \) shared across tasks (Lawrence & Platt, 2004; Schwaighofer et al., 2005; Yu et al., 2005), and those that additionally consider the covariance \( \Omega \) between tasks (Yu et al., 2007; Bonilla et al., 2008). The methods that only use \( \Sigma \) have been applied to collaborative prediction (Schwaighofer et al., 2005; Yu et al., 2006). However, due to the computational cost, they were both evaluated on very small data sets, where \( M \) and \( N \) are several hundreds only.

Our work differs from the above models in two aspects: First, a new nonparametric model is proposed to explicitly combine known predictors and random effects for multi-task predictions; Second, considerations in both model designing and algorithm engineering are put together to scale the nonparametric model to very large data sets.

Another class of related work is the so-called semi-parametric models, where the observations \( Y_{ij} \) are linearly generated from a finite number of basis functions randomly sampled from a Gaussian process prior (Teh et al., 2005). Our approach is more related to a recent work (Zhu et al., 2009), where \( Y_{ij} \) is modeled by two sets of multiplicative factors, one sampled from GP(0, \( \Sigma \)) and the other from GP(0, \( \Omega \)), namely, \( Y \) becomes a low-rank random function.

Low-rank matrix factorization is perhaps the most popular and the state-of-the-art method for collaborative filtering, e.g., (Salakhutdinov & Mnih, 2008a). Our model generalizes them in the sense that it models an infinite-dimensional relational function. A simplification of our work leads to a nonparametric prin-
principal component analysis (NPCA) model (Yu et al., 2009). A non-probabilistic nonparametric method, i.e., maximum-margin matrix factorization, was introduced in (Srebro et al., 2005). Very few matrix factorization methods use known predictors. One such a work is by (Hoff, 2005) that introduces low-rank multiplicative random effects, in addition to known predictors, to model networked observations.

5. Experiments

5.1. EachMovie Data

We did experiments on the entire EachMovie data, which include 74,424 users’ 2,811,718 numeric ratings \( Y_{ij} \in \{1, 2, 3, 4, 5, 6\} \) on 1,648 movies. The data are very sparse because 97.17% of the elements are missing. We randomly selected 80% of the observed ratings of each user for training and used the user’s rest 20% ratings for testing. This random partition was repeated 10 times independently. We calculated RMSE (root mean square error) for each partition and averaged the 10 results to compute the mean and the standard error.

In our experiments, for each evaluated method, we used one training/testing partition to do model selection, including determining hyper parameters and dimensionality. The selected model was then used for other partitions. For low-rank matrix factorization methods, we chose the dimensionality from \( D = 50, 100, 200 \). We found that larger dimensionality generally gave rise to better performances, but the accuracy was tending to saturate after the dimensionality got more than 100.

We used 1% of the training ratings for setting the stopping criterion: for each method, we terminated the iterations if RMSE on the holdout set is observed to increase, and then included the holdout set to run one more iteration of training. We recorded the total run time of each method. In the experiments, we implemented the following methods:

- User Mean and Movie Mean: baseline methods that predict ratings by computing the empirical mean of the same users’ observed ratings, or, other users’ ratings on the same movie.
- FMMMF: fast max-margin matrix factorization (Rennie & Srebro, 2005), a low-rank method with \( \ell_2 \) norm regularization on the factors, optimized by conjugate gradient descent.
- PPCA: probabilistic principal component analysis (Tipping & Bishop, 1999), which is a probabilistic low-rank matrix factorization method optimized by EM algorithm.
- BSRM: Bayesian stochastic relational model (Zhu et al., 2009), a semi-parametric model implemented by Gibbs sampling, where \( Y \) is finite-dimensional. BSRM-1 does not use additional user/movie attributes, while BSRM-2 does.
- NREM: Nonparametric random effects model, the work of this paper, which models \( Y \) as an infinite-dimensional random function. NREM-1 does not use additional attributes, i.e., we let \( \Sigma_0 \) and \( \Omega_0 \) be a constant \( c \), as suggested in Sec. 2.3.

For both BSRM-2 and NREM-2, we obtain the user and movie attributes from the top 20 eigenvectors of the binary matrix that indicates whether or not ratings are observed in the training set. Note that there can be different ways to obtain useful attributes, which is not the focus of this paper.

All the results are summarized in Tab. 1 and Tab. 2. Our models achieved lower prediction errors than other methods. This result is perhaps not entirely surprising, because nonparametric models are generally more flexible to explain complex data than parametric models. On the other hand, the training of our models is even faster, which is a significant result.

5.2. Netflix Problem

The data contain 100,480,507 ratings from 480,189 users on 17,770 movies. In this case \( Y_{ij} \in \)}
In addition, Netflix.com provides a set of validation data with 1,408,395 ratings. Therefore there are 98.81% of elements missing in the rating matrix. For evaluation, a set of 2,817,131 ratings are withheld. People need to submit their results in order to obtain the RMSE result from Netflix.com. We decided to directly compare our results with those reported in the literature, since they were all evaluated on the same test data.

The results are shown in Tab. 3, where Cinematch is the baseline provided by Netflix. Besides those introduced in Sec. 5.1, there are several other methods:

- SVD: a method almost the same as FMMMF, using a gradient-based method for optimization (Kurucz et al., 2007).
- RBM: Restricted Boltzmann Machine trained by contrast divergence (Salakhutdinov et al., 2007).
- PMF and BPMF: probabilistic matrix factorization (Salakhutdinov & Mnih, 2008b), and its Bayesian version (Salakhutdinov & Mnih, 2008a).

Note that sometimes the running time was not found in the papers. For BPMF, we gave a rough estimation by assuming it went through 300 iterations (each took 220 minutes as reported in the paper). Similar to the EachMovie experiments, BSRM-2 and NREM-2 used the top 40 eigenvectors of the binary indicator matrix as additional attributes. As shown in Tab. 3, if compared with those matrix factorization methods, NREM-1 used no additional attributes either, but generated more accurate predictions. Furthermore, NREMs are highly competitive if compared with the semi-parametric method BSRM. In terms of efficiency for training, our nonparametric model turns out to be even faster than those compared parametric and semi-parametric models.

We note that the top performers in the Netflix competition reported better results by combining heterogeneous models. For example, the progress award winner in 2007 combined predictions from about one hundred of different models (Bell et al., 2007). However, our focus here is not on developing ensemble methods.

6. Conclusion

In this paper a nonparametric model for multi-task learning is introduced. The contributions are twofold: First, the model provides a novel way to use random effects and known attributes to explain the complex dependence of data; Second, an algorithm is proposed to speed up the model on very large-scale problems. Our experiments demonstrate that the algorithm works very well on two challenging collaborative prediction problems. In the near future, it will be promising to perform a full Bayesian inference by a parallel Gibbs sampling method.

References


### Table 3. Performance on Netflix Data

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Run Time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cinematch</td>
<td>0.9514</td>
<td>-</td>
</tr>
<tr>
<td>SVD</td>
<td>0.920</td>
<td>300</td>
</tr>
<tr>
<td>PMF</td>
<td>0.9205</td>
<td>-</td>
</tr>
<tr>
<td>RBM</td>
<td>0.9060</td>
<td>-</td>
</tr>
<tr>
<td>PMF-VB</td>
<td>0.9141</td>
<td>-</td>
</tr>
<tr>
<td>BPMF</td>
<td>0.8954</td>
<td>1100</td>
</tr>
<tr>
<td>BSRM-2</td>
<td>0.8881</td>
<td>350</td>
</tr>
<tr>
<td>NREM-1</td>
<td>0.8876</td>
<td>148</td>
</tr>
<tr>
<td>NREM-2</td>
<td>0.8853</td>
<td>150</td>
</tr>
</tbody>
</table>


7. Appendix

7.1. Inverse-Wishart Processes

A nonstandard definition of inverse-Wishart distribution $\Sigma \sim \text{IW}(\kappa, \Psi)$ was introduced in (Dawid, 1981), with its p.d.f. proportional to $|\Sigma|^{-\frac{\kappa+n}{2}} \text{etr}(\frac{1}{2} \Sigma^{-1} \Psi)$. The distribution is consistent under marginalization, namely, its any principal submatrix $\Sigma_{11}$ follows $\text{IW}(\kappa, \Psi_{11})$. Therefore, just like Gaussian processes, one can characterize a random covariance function $\Sigma$ by an inverse-Wishart process.

7.2. Proof Sketch of the Model Equivalence

The proof is based on Theorem 4.2.1 in (Gupta & Naga, 1999). Using our notation, it states: Let $S \sim W(\kappa + N - 1, \Sigma^{-1})$, $X \sim N(0, I_N \otimes \Omega)$, if $Y = X^\top (S^{-\frac{1}{2}})$, then $Y$ follows a matrix-variate Student-$t$ distribution $\text{MT}(\kappa, 0, \Omega, \Sigma)$. The theorem implies: If $S^{-1} \sim \text{IW}(\kappa, \Sigma)$ and $Y \sim N(0, \Omega \otimes S^{-1})$, then $Y \sim \text{MT}(\kappa, 0, \Omega, \Sigma)$. This can be used to derive our result in the case where $Y$ is a random matrix. Since both inverse-Wishart and Gaussian are consistent under marginalization (Dawid, 1981), the result can be generalized to the case of random functions $Y$.

7.3. Derivation of the M-step

The optimization cost of the M-step can be written as

$$J_q(\beta, \Sigma) + \text{const}_1 - \log p(Y|\beta, \Sigma) - \log p(\beta|\Sigma) - \log p(\Sigma|\theta) = J_1(\beta, \Sigma) + \text{const}_2 + J_3(\Sigma) + \text{const}_3$$

where

$$J_1 = \frac{1}{2} tr \left\{ \Sigma^{-1} \left[ C - 2\beta^\top x^\top v + \beta^\top x^\top x^\beta \right] \right\} + \frac{M}{2} \log |\Sigma|$$

$$J_2 = \frac{1}{2} tr \left\{ \Sigma^{-1} (\beta^\top \beta) \right\} + \frac{p}{2} \log |\Sigma|$$

$$J_3 = \frac{1}{2} tr \left\{ \Sigma^{-1} (\Sigma_0 + \lambda I_N) \right\} + \frac{\kappa + 2N}{2} \log |\Sigma|$$

The part containing $\beta$ can be organized into

$$\frac{1}{2\tau} tr \left\{ \Sigma^{-1} \left[ (\beta - b)^\top (x^\top x + \tau I_p)(\beta - b) - v^\top x b \right] \right\}$$

where $b = (x^\top x + \tau I_p)^{-1} x^\top v$. This clearly suggests $\beta = b$. Removing irrelevant constants, we have

$$J(\beta, \Sigma) = \frac{M + 2N + p + \kappa}{2} \log |\Sigma| + \frac{1}{2} tr \left\{ \Sigma^{-1} \Psi \right\}$$

where $\Psi = \tau^{-1} (C - v^\top x(x^\top x + \tau I_p)^{-1} x^\top v) + \Sigma_0 + \lambda I_N$. Then a closed form $\hat{\Sigma}$ is obtained from $\frac{\partial J}{\partial \Sigma} = 0$. 