Large-scale Collaborative Prediction Using a Nonparametric Random Effects Model

Kai Yu † KYU@SV.NEC-LABS.COM
John Lafferty ‡ LAFFERTY@CS.CMU.EDU
Shenghuo Zhu † ZSH@SV.NEC-LABS.COM
Yihong Gong † YGONG@SV.NEC-LABS.COM
† 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

$$\begin{aligned} Y_{ij} &= \mu + m_i(\mathbf{z}_j) + \epsilon_{ij},\ j = 1, \ldots, N \\ \epsilon_{ij} &\overset{\text{iid}}{\sim} \text{N}(0, \sigma^2) \end{aligned}$$

where μ is a bias parameter and $\mathbf{z}_j \in \mathcal{Z}$ is the vector of input predictors for observation j. A more flexible approach is to model all of the data jointly, as

$$\begin{aligned} Y_{ij} &= \mu + m(\mathbf{x}_i, \mathbf{z}_j) + \epsilon_{ij}, \\ \epsilon_{ij} &\overset{\text{iid}}{\sim} \text{N}(0, \sigma^2) \end{aligned} \tag{1}$$

where now $m_{ij} = m(\mathbf{x}_i, \mathbf{z}_j)$ is a regression function based on *task-specific predictors* $\mathbf{x}_i \in \mathcal{X}$. 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 $\Sigma(\mathbf{z}_j, \mathbf{z}_{j'}; \theta_\Sigma) \succ 0$ is a covariance function among inputs and $\Omega(\mathbf{x}_i, \mathbf{x}_{i'}; \theta_\Omega) \succ 0$ a covariance function among tasks, which means $\text{Cov}(m_{ij}, m_{i'j'}) = \Omega_{ii'}\Sigma_{jj'}$. The parameters θ_Σ and θ_Ω 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 $\mathbf{Y} \in \mathbb{R}^{M \times N}$ 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 $O(M^3N^3)$. 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 $\mathbf{x}_i$ and $\mathbf{z}_j$ alone. In such a case the conditional independence assumption

$$\text{p}(\mathbf{Y} \,|\, m, \mathbf{x}, \mathbf{z}) = \prod_{i,j} \text{p}(Y_{ij} | m, \mathbf{x}_i, \mathbf{z}_j)$$

implied by Eq. (1) is invalid. Such is often the case for relational or networked observations. For instance,

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

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 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 $m_{ij} \equiv m(\mathbf{x}_i, \mathbf{z}_j)$ and additional additive random effects f_{ij}. The model becomes

$$Y_{ij} = \mu + m_{ij} + f_{ij} + \epsilon_{ij}, \ \epsilon_{ij} \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

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 $\mathbf{u}_i$ and $\mathbf{v}_j$ such that f_{ij} has a parametric form $f(\mathbf{u}_i, \mathbf{v}_j; \theta_f)$, as suggested in (Hoff, 2005).

A nonparametric random effects model may assume $f \sim \mathrm{GP}(0, \Delta \otimes \Upsilon)$, where the covariances Δ and Υ 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

$$\begin{aligned} m &\sim \mathrm{GP}(0, \Omega_0 \otimes \Sigma), \\ f_i &\overset{\text{iid}}{\sim} \mathrm{GP}(0, \tau\Sigma), \quad i = 1, \ldots, M. \end{aligned}$$

where $\tau > 0$, $\Omega_0(\mathbf{x}_i, \mathbf{x}_{i'})$ is a given covariance function based on task-specific predictors, and f_i denotes the function values of the random effects on task i. Following a hierarchical Bayesian framework, we let the covariance function Σ be a random variable sampled from a hyper-prior distribution

$$\Sigma \sim \mathrm{IWP}(\kappa, \Sigma_0 + \lambda\delta),$$

which defines an *inverse-Wishart process*, where $\lambda > 0$, κ is a degree-of-freedom parameter, δ is a Dirac delta kernel function, and $\Sigma_0(\mathbf{z}_j, \mathbf{z}_{j'})$ is a given covariance function based on input data. The inverse-Wishart process (IWP) is a nonparametric prior for random covariance functions, based on a nonstandard notation of the inverse-Wishart distribution (Dawid, 1981); a brief introduction is given in the Appendix.

The deviation of Y from the population mean μ is decomposed into two parts, random effects m and f. Let $\Omega_0(\mathbf{x}_i, \mathbf{x}_{i'}) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_{i'}) \rangle$, where ϕ is an implicit feature mapping; then m can be parameterized as

$$m_{ij} = m_j(\mathbf{x}_i) = m(\mathbf{x}_i, \beta_j) = \langle \phi(\mathbf{x}_i), \beta_j \rangle$$

where $\{\beta_j\}$ are latent Gaussian random variables whose covariance follows $\mathbb{E}(\langle \beta_j, \beta_{j'} \rangle) = \Sigma_{j,j'}$. Because of the coupling between m_i and $\mathbf{x}_i$, m_i is *nonexchangeable* given Σ and the predictors $\mathbf{x}_i$, while f_i is *exchangeable* under the same condition. The separation of these two kinds of random effects can enable computational advantages. For instance, one can let the nonexchangeable part be parametric while the exchangeable part is nonparametric; we will develop this strategy below in Sec. 3.

We note that the independent noise ϵ_{ij} is no longer separately required in this model, but can be absorbed into the random effects f. Since the covariance Σ is a random variable, it is fully capable of modeling the total variance of the random effects and independent noise. Indeed, this treatment can considerably improve the computational efficiency, as we shall see in Sec. 3.4.

Since the population mean μ can be reliably estimated by pooling all the observations together and computing their average, hereafter, for simplicity of presentation, we assume Y_{ij} to be centered and thus μ will not occur in the model.

2.2. An Equivalent Model

Note that the above *row-wise* generative model seems to be arbitrary, because there is no particular reason to prefer it over the alternative *column-wise* sampling process,

$$\begin{aligned}\Omega &\sim \mathrm{IWP}(\kappa, \Omega_0 + \tau\delta),\\ m &\sim \mathrm{GP}(0, \Omega \otimes \Sigma_0),\\ f_j &\overset{\text{iid}}{\sim} \mathrm{GP}(0, \lambda\Omega), \quad j = 1, \ldots, N,\end{aligned}$$

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. $\int \mathrm{p}(Y|m, \tau, \Sigma)\mathrm{p}(m|\Sigma, \Omega_0)\mathrm{p}(\Sigma|\Sigma_0, \lambda, \kappa)\ \mathrm{d}m\ \mathrm{d}\Sigma$;
2. $\int \mathrm{p}(Y|m, \lambda, \Omega)\mathrm{p}(m|\Omega, \Sigma_0)\mathrm{p}(\Omega|\Omega_0, \tau, \kappa)\ \mathrm{d}m\ \mathrm{d}\Omega$;

we obtain exactly the same closed-form distribution

$$Y \sim \mathrm{MTP}\big(\kappa, 0, (\Omega_0 + \tau\delta), (\Sigma_0 + \lambda\delta)\big),$$

where MTP defines a matrix-variate Student-t process. That is, any subset of matrix values $\mathbf{Y} \in \mathbb{R}^{M\times N}$ 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 Σ and Ω 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 $\mathbf{Y}$ 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 Σ to the data, or equivalently, adapting the covariance Ω to the data. Therefore the model is somewhat similar to those that deal with both Σ and Ω 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 $\mathcal{O}(M^3N^3)$, without sacrificing significant flexibility.

The model can easily incorporate group-specific effects, as do most classical random-effects models. For example, one can expand the feature representation by adding a constant term $\sqrt{c}$, such that $\Omega_0(\mathbf{x}_i, \mathbf{x}_{i'}) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_{i'})\rangle + c$. Then m can be parameterized as

$$m_{ij} = \langle\phi(\mathbf{x}_i), \beta_j\rangle + \gamma_j$$

where γ_j is the j-th element of $\gamma \sim \mathrm{GP}(0, c\Sigma)$, i.e., the column-specific random effects. Similarly, with $\Sigma_0(\mathbf{z}_j, \mathbf{z}_{j'}) = \langle\varphi(\mathbf{z}_j), \varphi(\mathbf{z}_{j'})\rangle + b$, 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 $\mathbf{Y}_O$, we can make predictions by computing the conditional expectation on unseen elements. To this end, we need to compute the posterior distribution $p(\mathbf{Y}|\mathbf{Y}_O, \theta)$, where θ includes $\Sigma_0, \Omega_0, \lambda, \tau$ and κ.

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 $\mathbf{Y} \in \mathbb{R}^{M\times N}$, 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

$$\begin{aligned}\mathbf{\Sigma} &\sim \mathrm{IW}(\kappa, \mathbf{\Sigma}_0 + \lambda\mathbf{I}_N),\\ \mathbf{m} &\sim \mathrm{N}(0, \mathbf{\Omega}_0 \otimes \mathbf{\Sigma}),\\ \mathbf{Y}_i &\sim \mathrm{N}(\mathbf{m}_i, \tau\mathbf{\Sigma}), \quad i = 1, \ldots, M\end{aligned}$$

where IW defines a finite inverse-Wishart distribution, $\mathbf{\Sigma}$ and $\mathbf{\Sigma}_0$ are both $N \times N$ covariance matrices, $\mathbf{\Omega}_0$ is $M \times M$, $\mathbf{m}_i^\top$ is the i-th row of $\mathbf{m}$, and $\mathbf{Y}_i^\top$ is the i-th row of $\mathbf{Y}$.

To further facilitate a large-scale implementation, we assume Ω_0 has finite dimensions. i.e., $\Omega_0(\mathbf{x}_i, \mathbf{x}_{i'}) = \boldsymbol{\phi}(\mathbf{x}_i)^\top\boldsymbol{\phi}(\mathbf{x}_{i'})$, with $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^p$. 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 Σ and Ω are always infinite-dimensional, and so is the function Y. Therefore the overall model is still nonparametric.

Without loss of generality, we let $\Omega_0(\mathbf{x}_i, \mathbf{x}_{i'}) = \langle p^{\frac{1}{2}}\mathbf{x}_i, p^{\frac{1}{2}}\mathbf{x}_{i'}\rangle$. In this case, the random effects m follow

$$\mathbf{m}_i = \boldsymbol{\beta}^\top\mathbf{x}_i, \quad \boldsymbol{\beta}_k \overset{\text{iid}}{\sim} \mathrm{N}(0, \mathbf{\Sigma}), \quad k = 1, \ldots, p$$

where $\boldsymbol{\beta}$ is a $p \times N$ random matrix, and $\boldsymbol{\beta}_k^\top$ is its k-th row. Then the model becomes

$$\begin{aligned}\mathbf{\Sigma} &\sim \mathrm{IW}(\kappa, \mathbf{\Sigma}_0 + \lambda\mathbf{I}_N),\\ \boldsymbol{\beta} &\sim \mathrm{N}(0, \mathbf{I}_p \otimes \mathbf{\Sigma}),\\ \mathbf{Y}_i &\sim \mathrm{N}(\boldsymbol{\beta}^\top\mathbf{x}_i, \tau\mathbf{\Sigma}), \quad i = 1, \ldots, M\end{aligned}$$

3.2. Approximate Inference – Gibbs Sampling

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

$$\int \mathrm{p}(\mathbf{Y}|\mathbf{Y}_{\mathrm{O}}, \theta)\mathbf{Y}\mathrm{d}\mathbf{Y} \approx \frac{1}{\mathrm{T}} \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathbf{Y}\ \delta(\mathbf{Y}; \mathbf{Y}^{(\mathrm{t})})$$

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

$$\mathcal{O}\left(\sum_{i=1}^{M} N_i^3 + 2\left(\sum_{i=1}^{M} N_i^2\right) + 2MN^2 + N^3\right), \qquad (2)$$

where we count mainly multiplications, and ignore the impact of factor p, since $M \gg N \gg p$. Compared with $\mathcal{O}(M^3N^3)$ 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 $\mathbf{Y}_i$, and $\boldsymbol{\Sigma}_{[:,J_i]} \in \mathbb{R}^{N \times N_i}$ be the matrix obtained by keeping the columns of $\boldsymbol{\Sigma}$ indexed by J_i. Then let $\boldsymbol{\Sigma}_{[J_i,J_i]} \in \mathbb{R}^{N_i \times N_i}$ be obtained from $\boldsymbol{\Sigma}_{[:,J_i]}$ by further keeping only the rows indexed by J_i. We can similarly define $\boldsymbol{\Sigma}_{[J_i,:]}$, $\mathbf{Y}_{[i,J_i]}$ and $\mathbf{m}_{[i,J_i]}$.

Alteratively, the conditional expectation can be approximated by

$$\begin{aligned}&\int \mathrm{p}(\mathbf{Y}|\mathbf{Y}_{\mathrm{O}}, \theta)\mathbf{Y}\mathrm{d}\mathbf{Y} \\ &= \int\int\int \mathrm{p}(\mathbf{Y}|\mathbf{Y}_{\mathrm{O}}, \boldsymbol{\beta}, \boldsymbol{\Sigma})\mathrm{p}(\boldsymbol{\beta}, \boldsymbol{\Sigma}|\mathbf{Y}_{\mathrm{O}}, \theta)\mathbf{Y}\ \mathrm{d}\boldsymbol{\beta}\mathrm{d}\boldsymbol{\Sigma}\mathrm{d}\mathbf{Y} \\ &\approx \int \mathrm{p}(\mathbf{Y}|\mathbf{Y}_{\mathrm{O}}, \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\Sigma}})\mathbf{Y}\ \mathrm{d}\mathbf{Y}\end{aligned}$$

where $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\Sigma}}$ are the maximum *a posteriori* estimates

$$\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\Sigma}} = \arg\max_{\boldsymbol{\beta},\boldsymbol{\Sigma}} \mathrm{p}(\boldsymbol{\beta}, \boldsymbol{\Sigma}|\mathbf{Y}_{\mathrm{O}}, \theta).$$

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 $\mathbf{Y}$ given the current $\boldsymbol{\beta}, \boldsymbol{\Sigma}$

$$Q(\mathbf{Y}) = \prod_{i=1}^{M} \mathrm{p}(\mathbf{Y}_i|\mathbf{Y}_{O_i}, \boldsymbol{\beta}, \boldsymbol{\Sigma}) = \prod_{i=1}^{M} \mathrm{N}(\mathbf{Y}_i|\boldsymbol{\upsilon}_i, \mathbf{C}_i),$$

  where the sufficient statistics are

$$\begin{aligned}\mathbf{m}_i &= \boldsymbol{\beta}^\top \mathbf{x}_i, \\ \boldsymbol{\upsilon}_i &= \mathbf{m}_i + \boldsymbol{\Sigma}_{[:,J_i]}\boldsymbol{\Sigma}_{[J_i,J_i]}^{-1}(\mathbf{Y}_{[i,J_i]} - \mathbf{m}_{[i,J_i]})^\top, \\ \mathbf{C}_i &= \tau\boldsymbol{\Sigma} - \tau\boldsymbol{\Sigma}_{[:,J_i]}\boldsymbol{\Sigma}_{[J_i,J_i]}^{-1}\boldsymbol{\Sigma}_{[J_i,:]}.\end{aligned}$$

- M-step: optimize $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$

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

  and then let $\boldsymbol{\beta} \leftarrow \widehat{\boldsymbol{\beta}}, \boldsymbol{\Sigma} \leftarrow \widehat{\boldsymbol{\Sigma}}$.

It turns out that $\widehat{\boldsymbol{\beta}}$ can be nicely decoupled from $\widehat{\boldsymbol{\Sigma}}$, and has a closed form:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{x}^\top\mathbf{x} + \tau\mathbf{I}_p)^{-1}\mathbf{x}^\top\boldsymbol{\upsilon}.$$

Plugging $\widehat{\boldsymbol{\beta}}$ into the optimization cost, we have

$$\widehat{\boldsymbol{\Sigma}} = \frac{\tau^{-1}[\mathbf{C} - \boldsymbol{\upsilon}^\top\mathbf{x}(\mathbf{x}^\top\mathbf{x} + \tau\mathbf{I}_p)^{-1}\mathbf{x}^\top\boldsymbol{\upsilon}] + \boldsymbol{\Sigma}_0 + \lambda\mathbf{I}_N}{M + 2N + p + \kappa}$$

where $\mathbf{C} = \sum_{i=1}^{M}(\mathbf{C}_i + \boldsymbol{\upsilon}_i\boldsymbol{\upsilon}_i)$. 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 $\mathbf{U}_i \in \mathbb{R}^{N \times N_i}$ 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 $\boldsymbol{\Sigma}_{[:,J_i]} = \boldsymbol{\Sigma}\mathbf{U}_i$ and $\boldsymbol{\Sigma}_{[J_i,J_i]} = \mathbf{U}_i^\top\boldsymbol{\Sigma}\mathbf{U}_i$. Furthermore, $\mathbf{U}_i\boldsymbol{\Sigma}_{[J_i,J_i]}\mathbf{U}_i^\top$ restores a $N \times N$ sparse matrix where the elements of $\boldsymbol{\Sigma}_{[J_i,J_i]}$ is placed back to their original positions in $\boldsymbol{\Sigma}$.

The major cost is from the E-step that computes the sufficient statistics $\boldsymbol{v}_i$ and $\mathbf{C}_i$ for every i. Our key observation is that, the M-step only needs $\mathbf{C} = \sum_{i=1}^{M} \mathbf{C}_i + \boldsymbol{v}^\top \boldsymbol{v}$ and $\boldsymbol{v}^\top \mathbf{x}$ from the previous E-step. To obtain them, in fact it is unnecessary to compute $\boldsymbol{v}_i$ and $\mathbf{C}_i$ at all. For example, by using $\boldsymbol{\Sigma}_{[:,J_i]} = \boldsymbol{\Sigma}\mathbf{U}_i$, we have

$$\begin{aligned}
\sum_{i=1}^{M} \mathbf{C}_i &= \sum_{i=1}^{M} \left(\tau\boldsymbol{\Sigma} - \tau\boldsymbol{\Sigma}_{[:,J_i]}\boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}\boldsymbol{\Sigma}_{[J_i,:]}\right) \\
&= \sum_{i=1}^{M} \left(\tau\boldsymbol{\Sigma} - \tau\boldsymbol{\Sigma}\mathbf{U}_i\boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}\mathbf{U}_i^\top\boldsymbol{\Sigma}\right) \\
&= \tau M\boldsymbol{\Sigma} - \tau\boldsymbol{\Sigma}\left(\sum_{i=1}^{M} \mathbf{U}_i\boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}\mathbf{U}_i^\top\right)\boldsymbol{\Sigma}
\end{aligned}$$

Since multiplication with $\mathbf{U}_i$ is done by memory access only, above reduces $\mathcal{O}(\sum_{i=1}^{M} N_i^3 + NN_i^2 + N^2N_i)$ to $\mathcal{O}(\sum_{i=1}^{M} N_i^3 + 2N^3)$. In a similar way we have

$$\begin{aligned}
\boldsymbol{v}^\top\boldsymbol{v} &= \boldsymbol{\beta}^\top\mathbf{x}^\top\mathbf{x}\boldsymbol{\beta} + \boldsymbol{\Sigma}\left(\sum_{i=1}^{M}\mathbf{U}_i\mathbf{a}_i\mathbf{a}_i^\top\mathbf{U}_i^\top\right)\boldsymbol{\Sigma} + \\
&\quad \boldsymbol{\Sigma}\left(\sum_{i=1}^{M}\mathbf{U}_i\mathbf{a}_i\mathbf{x}_i^\top\right)\boldsymbol{\beta} + \boldsymbol{\beta}^\top\left(\sum_{i=1}^{M}\mathbf{U}_i\mathbf{a}_i\mathbf{x}_i^\top\right)^\top\boldsymbol{\Sigma} \\
\boldsymbol{v}^\top\mathbf{x} &= \boldsymbol{\beta}^\top\mathbf{x}^\top\mathbf{x} + \boldsymbol{\Sigma}\left(\sum_{i=1}^{M}\mathbf{U}_i\mathbf{a}_i\mathbf{x}_i^\top\right)
\end{aligned}$$

where $\mathbf{a}_i = \boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}(\mathbf{Y} - \mathbf{m})_{[i,J_i]}$.

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

```
Reset  $\mathbf{A}_1 \in \mathbb{R}^{N\times N}, \mathbf{A}_2 \in \mathbb{R}^{N\times p}$;
For    $i = 1,\ldots,M$;
           $\mathbf{P} = \boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}$;
           $\mathbf{a} = \mathbf{P}(\mathbf{Y}_{[i,J_i]} - \mathbf{x}_i^\top\boldsymbol{\beta}_{[:,J_i]})^\top$;
           $\mathbf{A}_1 \leftarrow \mathbf{A}_1 + \mathbf{U}_i(\mathbf{a}\mathbf{a}^\top - \tau\mathbf{P})\mathbf{U}_i^\top$;
           $\mathbf{A}_2 \leftarrow \mathbf{A}_2 + \mathbf{U}_i\mathbf{a}\mathbf{x}_i^\top$;
End.
```

The statistics required by the M-step are computed as

$$\mathbf{C} = \tau M\boldsymbol{\Sigma} + \boldsymbol{\beta}^\top\mathbf{x}^\top\mathbf{x}\boldsymbol{\beta} + \boldsymbol{\Sigma}\mathbf{A}_1\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\mathbf{A}_2\boldsymbol{\beta} + \beta^\top\mathbf{A}_2^\top\boldsymbol{\Sigma}$$

$$\boldsymbol{v}^\top\mathbf{x} = \boldsymbol{\beta}^\top\mathbf{x}^\top\mathbf{x} + \boldsymbol{\Sigma}\mathbf{A}_2$$

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

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

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 $\mathcal{O}(\sum_{i=1}^{M} N_i^3)$ comes from the matrix inversion $\boldsymbol{\Sigma}^{-1}_{[J_i,J_i]}$. 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 ϵ_{ij}. Otherwise, one has to spend $\mathcal{O}(\sum_{i=1}^{M}(2N_i^2 + 2N_i^3))$ 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 Σ shared across tasks (Lawrence & Platt, 2004; Schwaighofer et al., 2005; Yu et al., 2005), and those that additionally consider the covariance Ω between tasks (Yu et al., 2007; Bonilla et al., 2008). The methods that only use Σ 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 $\mathrm{GP}(0,\Sigma)$ and the other from $\mathrm{GP}(0,\Omega)$, 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-

Table 1. Prediction Error on EachMovie Data

Method	RMSE	Standard Error
User Mean	1.4251	0.0004
Movie Mean	1.3866	0.0004
FMMMF	1.1552	0.0008
PPCA	1.1045	0.0004
BSRM-1	1.0902	0.0003
BSRM-2	1.0852	0.0003
NREM-1	1.0816	0.0003
NREM-2	1.0758	0.0003

cipal 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:

Table 2. Computation Time on EachMovie Data

Method	Run Time (hours)
FMMMF	4.94
PPCA	1.26
BSRM-1	1.67
BSRM-2	1.70
NREM-1	0.59
NREM-2	0.59

- 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 ℓ_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 Σ_0 and Ω_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 $Y_{ij} \in$

$\{1, 2, 3, 4, 5\}$. 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 direct 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).
- PMF-VB: probabilistic matrix factorization using a variational Bayes method for inference (Lim & Teh, 2007).

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 heterogenous 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.

Table 3. Performance on Netflix Data

Method	RMSE	Run Time (hours)
Cinematch	0.9514	-
SVD	0.920	300
PMF	0.9265	-
RBM	0.9060	-
PMF-VB	0.9141	-
BPMF	0.8954	1100
BSRM-2	0.8881	350
NREM-1	0.8876	148
NREM-2	0.8853	150

References

Bell, R. M., Koren, Y., & Volinsky, C. (2007). *The BellKor solution to the Netflix prize* (Technical Report). AT&T Labs.

Bonilla, E. V., Chai, K. M. A., & Williams, C. K. I. (2008). Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems 20 (NIPS)*, 153–160.

Dawid, A. P. (1981). Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, *68*, 265–274.

Gupta, A. K., & Naga, D. K. (1999). *Matrix variate distributions*. Chapman & Hall/CRC.

Hoff, P. (2005). Bilinear mixed-effects models for dyadic data. *Journal of the American Statistical Assosciation*, *100*, 286–295.

Kurucz, M., Benczur, A. A., & Csalogany, K. (2007). Methods for large scale SVD with missing values. *Proceedings of KDD Cup and Workshop*.

Lawrence, N. D., & Platt, J. C. (2004). Learning to learn with the informative vector machine. *Proc. of the 21st International Conference on Machine Learning (ICML)*, 65–72.

Lim, Y. J., & Teh, Y. W. (2007). Variational Bayesian approach to movie rating prediction. *Proceedings of KDD Cup and Workshop*.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. *Proc. of the 22nd International Conference on Machine Learning (ICML)*, 713–719.

Salakhutdinov, R., & Mnih, A. (2008a). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. *Proc. of the 25th International Conference on Machine Learning (ICML)*, 880–887.

Salakhutdinov, R., & Mnih, A. (2008b). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20 (NIPS)*, 1257–1264.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proc. of the 24th International Conference on Machine Learning (ICML)*, 791–798.

Schwaighofer, A., Tresp, V., & Yu, K. (2005). Hierarchical Bayesian modelling with Gaussian processes. *Advances in Neural Information Processing Systems 17 (NIPS)*, 1209–1216.

Srebro, N., Rennie, J. D. M., & Jaakola, T. S. (2005). Maximum-margin matrix factorization. *Advances in Neural Information Processing Systems 18 (NIPS)*, 1329–1336.

Teh, Y., Seeger, M., & Jordan, M. (2005). Semiparametric latent factor models. *The 8th Conference on Artificial Intelligence and Statistics (AISTATS).*

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statisitical Scoiety, B*, 611–622.

Yu, K., Chu, W., Yu, S., Tresp, V., & Xu, Z. (2007). Stochastic relational models for discriminative link prediction. *Advances in Neural Information Processing Systems 19 (NIPS)*, 1553–1560.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. *Proc. of the 22nd International Conference on Machine Learning (ICML)*, 1012–1019.

Yu, K., Zhu, S., Lafferty, J., & Gong, Y. (2009). Fast nonparametric matrix factorization for large-scale collaborative filtering. *The 32nd SIGIR conference.*

Yu, S., Yu, K., Tresp, V., & Kriegel, H.-P. (2006). Collaborative ordinal regression. *Proc. of the 23rd International Conference on Machine Learning (ICML)*, 1089–1096.

Zhu, S., Yu, K., & Gong, Y. (2009). Stochastic relational models for large-scale dyadic data using MCMC. *Advances in Neural Information Processing Systems 21 (NIPS)*, 1993–2000.

7. Appendix

7.1. Inverse-Wishart Processes

A nonstandard definition of inverse-Wishart distribution $\mathbf{\Sigma} \sim \mathrm{IW}(\kappa, \mathbf{\Psi})$ was introduced in (Dawid, 1981), with its p.d.f. proportional to $|\mathbf{\Sigma}|^{-\frac{\kappa+2N}{2}} \mathrm{etr}(\frac{-1}{2}\mathbf{\Sigma}^{-1}\mathbf{\Psi})$. The distribution is consistent under marginalization, namely, its any principal submatrix $\mathbf{\Sigma}_{11}$ follows $\mathrm{IW}(\kappa, \mathbf{\Psi}_{11})$. Therefore, just like Gaussian processes, one can characterize a random covariance function Σ 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 $\mathbf{S} \sim \mathrm{W}(\kappa + N - 1, \mathbf{\Sigma}^{-1})$, $\mathbf{X} \sim \mathrm{N}(0, \mathbf{I}_N \otimes \mathbf{\Omega})$, if $\mathbf{Y} = \mathbf{X}^\top(\mathbf{S}^{-\frac{1}{2}})$, then $\mathbf{Y}$ follows a matrix-variate Student-t distribution $\mathrm{MT}(\kappa, 0, \mathbf{\Omega}, \mathbf{\Sigma})$. The theorem implies: If $\mathbf{S}^{-1} \sim \mathrm{IW}(\kappa, \mathbf{\Sigma})$ and $\mathbf{Y} \sim \mathrm{N}(0, \mathbf{\Omega} \otimes \mathbf{S}^{-1})$, then $\mathbf{Y} \sim \mathrm{MT}(\kappa, 0, \mathbf{\Omega}, \mathbf{\Sigma})$. This can be used to derive our result in the case where $\mathbf{Y}$ 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

$$\underbrace{\mathbb{E}_{Q(\mathbf{Y})}\left[-\log \mathrm{p}(\mathbf{Y}|\boldsymbol{\beta}, \mathbf{\Sigma})\right]}_{J_1(\boldsymbol{\beta},\mathbf{\Sigma})+\mathrm{const}_1} - \underbrace{\log \mathrm{p}(\boldsymbol{\beta}|\mathbf{\Sigma})}_{J_2(\boldsymbol{\beta},\mathbf{\Sigma})+\mathrm{const}_2} - \underbrace{\log \mathrm{p}(\mathbf{\Sigma}|\theta)}_{J_3(\mathbf{\Sigma})+\mathrm{const}_3}$$

where

$$\begin{aligned}
J_1 =& \frac{1}{2\tau}\mathrm{tr}\left\{\mathbf{\Sigma}^{-1}\left[\mathbf{C} - 2\boldsymbol{\beta}^\top\mathbf{x}^\top\boldsymbol{\upsilon} + \boldsymbol{\beta}^\top\mathbf{x}^\top\mathbf{x}\boldsymbol{\beta}\right]\right\} + \frac{M}{2}\log|\mathbf{\Sigma}| \\
J_2 =& \frac{1}{2}\mathrm{tr}\left[\mathbf{\Sigma}^{-1}(\boldsymbol{\beta}^\top\boldsymbol{\beta})\right] + \frac{p}{2}\log|\mathbf{\Sigma}| \\
J_3 =& \frac{1}{2}\mathrm{tr}\left[\mathbf{\Sigma}^{-1}(\mathbf{\Sigma}_0 + \lambda\mathbf{I}_N)\right] + \frac{\kappa+2N}{2}\log|\mathbf{\Sigma}|
\end{aligned}$$

The part containing $\boldsymbol{\beta}$ can be organized into

$$\frac{1}{2\tau}\mathrm{tr}\left\{\mathbf{\Sigma}^{-1}\left[(\boldsymbol{\beta}-\mathbf{b})^\top(\mathbf{x}^\top\mathbf{x} + \tau\mathbf{I}_p)(\boldsymbol{\beta}-\mathbf{b}) - \boldsymbol{\upsilon}^\top\mathbf{x}\mathbf{b}\right]\right\}$$

where $\mathbf{b} = (\mathbf{x}^\top\mathbf{x} + \tau\mathbf{I}_p)^{-1}\mathbf{x}^\top\boldsymbol{\upsilon}$. This clearly suggests $\widehat{\boldsymbol{\beta}} = \mathbf{b}$. Removing irrelevant constants, we have

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

where $\mathbf{\Psi} = \tau^{-1}(\mathbf{C} - \boldsymbol{\upsilon}^\top\mathbf{x}(\mathbf{x}^\top\mathbf{x} + \tau\mathbf{I}_p)^{-1}\mathbf{x}^\top\boldsymbol{\upsilon}) + \mathbf{\Sigma}_0 + \lambda\mathbf{I}_N$. Then a closed form $\widehat{\mathbf{\Sigma}}$ is obtained from $\frac{\partial J}{\partial \mathbf{\Sigma}} = 0$.