Scalable Bayesian Non-linear Matrix Completion

Xiangju Qin*, **Paul Blomstedt** and **Samuel Kaski**
Department of Computer Science, Helsinki Institute for Information Technology HIIT,
Aalto University, 00076 Espoo, Finland
xiangju.qin@helsinki.fi, paul.blomstedt@aalto.fi, samuel.kaski@aalto.fi

Abstract

Matrix completion aims to predict missing elements in a partially observed data matrix which in typical applications, such as collaborative filtering, is large and extremely sparsely observed. A standard solution is matrix factorization, which predicts unobserved entries as linear combinations of latent variables. We generalize to non-linear combinations in massive-scale matrices. Bayesian approaches have been proven beneficial in linear matrix completion, but not applied in the more general non-linear case, due to limited scalability. We introduce a Bayesian non-linear matrix completion algorithm, which is based on a recent Bayesian formulation of Gaussian process latent variable models. To solve the challenges regarding scalability and computation, we propose a data-parallel distributed computational approach with a restricted communication scheme. We evaluate our method on challenging out-of-matrix prediction tasks using both simulated and real-world data.

1 Introduction

In matrix completion—one of the most widely used approaches for collaborative filtering—the objective is to predict missing elements of a partially observed data matrix. Such problems are often characterized by large and extremely sparsely observed data sets. The classic linear solution to the problem is to find a factorization of the data matrix $\mathbf{Y} \in \mathbb{R}^{N \times D}$ as a product of latent variables $\mathbf{X} \in \mathbb{R}^{N \times K}$ and weights $\mathbf{W} \in \mathbb{R}^{D \times K}$ ($K \ll N, D$), from which elements of $\mathbf{Y}$ can be predicted as $\mathbf{Y} \approx \mathbf{X}\mathbf{W}^T$. Probabilistic matrix factorization (PMF) [Salakhutdinov and Mnih, 2008b], formulates the problem as a probabilistic model, regularized by placing priors on $\mathbf{X}$ and $\mathbf{W}$, and finds the solution as a maximum a posteriori (MAP) estimate of these matrices. Fully Bayesian matrix factorization [Salakhutdinov and Mnih, 2008a] expands this model by further placing priors on model hyperparameters, and marginalizing these along with $\mathbf{X}$ and $\mathbf{W}$. Bayesian matrix factorization brings the advantages of automatic complexity control and better robustness to overfitting. Moreover, the solution comes with an uncertainty estimate, which is useful when the completed matrix is used for decision making. For instance, sparsely observed drug-target interaction matrices are used for deciding which interactions to measure next.

Lawrence and Urtasun [2009] generalized PMF using a Gaussian process latent variable model (GP-LVM) formulation, where the relationship between $\mathbf{X}$ and $\mathbf{Y}$ is given by $\mathbf{Y} \approx \mathbf{f}(\mathbf{X})$, with a GP-prior placed over $\mathbf{f}$. The $\mathbf{X}$ is optimized to find its MAP solution. Note that this formulation also subsumes the linear model as a special case. Subsequently, a variational inference framework for fully Bayesian GP-LVM has been developed [Damianou *et al.*, 2016; Titsias and Lawrence, 2010], building on sparse GP approximations [Quiñonero Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006]. It parametrizes the covariance matrix implied by the GP kernel using a set of $M \ll N$ auxiliary inducing variables. While Bayesian GP-LVM has been successfully used for dimensionality reduction and extracting latent representations, less consideration has been given to its applicability in matrix completion tasks with extremely sparse data. Computationally, this is a much more demanding problem, because the variational updates have to be performed separately for each dimension of the data matrix, instead of being performed as a single operation.

Existing approaches for scaling up Bayesian GP-LVM make use of the insight that, conditional on the inducing variables, the data points can be decoupled for parallel computations. In this line of work, Gal et al. [2014] introduced a distributed version of Bayesian GP-LVM. Dai et al. [2014] proposed a similar framework, additionally using GPU acceleration to speed up local computations. Neither of the works demonstrated learning of latent variable models beyond moderately-sized data, nor have they been implemented for sparse matrices, which is needed for the problems considered in this paper. More importantly, current distributed solutions require the worker nodes to communicate with the master node in every iteration, which leads to an accumulating communication overhead as the number of worker units increased with the size of the problem. Vander Aa et al. [2016] reported such a phenomenon for their distributed MPI implementation of Bayesian linear matrix factorization. Finally,

*Contact Author. Currently at Institute for Molecular Medicine Finland (FIMM), University of Helsinki.

our experience indicates that existing distributed implementations may suffer from high memory consumption.

For GP-regression models, with $\mathbf{X}$ observed, Deisenroth and Ng [2015] proposed a framework with particularly good scaling properties and efficient use of memory. This framework utilizes a product-of-GP-experts (PoE) formulation [Cao and Fleet, 2014; Ng and Deisenroth, 2014; Tresp, 2000], which makes predictions using a product of independent local models, each operating on a subset of the data. These types of approximations are amenable to embarrassingly parallel computations, and can therefore be scaled up to arbitrarily large data sets, at least in principle. However, a direct application of PoE for nonlinear matrix completion may not produce satisfactory predictions for two reasons. First, since the target matrix is very sparsely observed, each local model has very limited information to learn an informative model without sharing information. Second, while local models could be combined into larger models to improve predictive performance, this is hindered by the general non-uniqueness of the latent variables in latent variable models.

In this work, we propose a distributed computational strategy which is almost as communication-efficient as embarrassingly parallel computation, but enables local models to share information and avoids the problem of non-uniqueness in aggregating local models. In a nutshell, one data subset is processed first, and the rest of the embarrassingly parallel computations are conditioned on the result. A similar idea was recently presented by [Qin *et al.*, 2019] for Bayesian linear matrix completion. The remainder of the paper proceeds as follows: In Section 2 we first provide a brief review of GP-LVMs. Then, in Section 3, we present our framework for scalable Bayesian non-linear matrix completion. An empirical evaluation of the method, using simulations and a benchmark dataset, is given in Section 4. The paper ends with conclusions in Section 5.

2 Gaussian Process Latent Variable Models

A Gaussian process latent variable model (GP-LVM) [Lawrence, 2005] can be constructed from a non-linear multi-output regression model,

$$p(\mathbf{Y}|\mathbf{F},\mathbf{X},\sigma^2) = \prod_{d=1}^{D} p(\mathbf{y}_{:,d}|\mathbf{f}_{:,d},\mathbf{X},\sigma^2) = \prod_{d=1}^{D}\prod_{n=1}^{N} \mathcal{N}\left(y_{n,d}|f_d(\mathbf{x}_{n,:}),\sigma^2\right),$$

by placing a GP prior over the unknown functions $f_1 \ldots, f_d$. Integrating over the space of functions with respect to a zero-mean GP then yields the likelihood as

$$p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta}) = \prod_{d=1}^{D} \int \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{f}_{:,d},\sigma^2\mathbf{I})\mathcal{N}(\mathbf{f}_{:,d}|\mathbf{0},\mathbf{K})\mathrm{d}\mathbf{f}_{:,d} = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{0},\mathbf{K}+\sigma^2\mathbf{I}), \quad (1)$$

where $\mathbf{K}$ is an $N \times N$ kernel matrix defined by a GP covariance function $k(\mathbf{x}_{s,:},\mathbf{x}_{t,:})$. We use $\boldsymbol{\theta}$ to collectively denote all parameters, including the noise variance σ^2 and the parameters of the covariance function.

When values are missing, as is the case in matrix completion, each factor of the likelihood (1) will only account for observed elements, thus we have

$$p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{\mathbf{n}_d,d}|\mathbf{0},\mathbf{K}+\sigma^2\mathbf{I}), \quad (2)$$

where $\mathbf{n}_d$ denotes the set of indices of observed elements in column d. Furthermore, the symmetric matrices $\mathbf{K}$ and $\mathbf{I}$ will only include rows and columns corresponding to the indices $\mathbf{n}_d$.

2.1 Bayesian GP-LVM

For Bayesian GP-LVM, we complement the likelihood (2) by placing a prior on $\mathbf{X}$. A standard choice is to set

$$p(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_{n,:}|\mathbf{0},\mathbf{I}). \quad (3)$$

The marginal likelihood of Bayesian GP-LVM is obtained by integrating the model with respect to $p(\mathbf{X})$:

$$p(\mathbf{Y}|\boldsymbol{\theta}) = \int p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})p(\mathbf{X})\mathrm{d}\mathbf{X}. \quad (4)$$

While this operation is intractable in general, Titsias and Lawrence [2010] introduced a variational framework, which leads to a tractable lower bound,

$$\begin{aligned} F(q) &= \int q(\mathbf{X}) \log \frac{p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})p(\mathbf{X})}{q(\mathbf{X})}\mathrm{d}\mathbf{X} \quad (5)\\ &= \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})\mathrm{d}\mathbf{X} - \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})}\mathrm{d}\mathbf{X} \\ &= \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})\mathrm{d}\mathbf{X} - \mathrm{KL}\left(q(\mathbf{X})||p(\mathbf{X})\right), \end{aligned}$$

on the log of the marginal likelihood (4). For a detailed treatment of the framework, see [Damianou *et al.*, 2016]. As a by-product of optimizing the lower bound (5), we get a variational approximation $q(\mathbf{X})$ to the posterior $p(\mathbf{X}|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})$, for which we assume a factorized Gaussian form

$$q(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_{n,:}|\boldsymbol{\mu}_n, S_n). \quad (6)$$

In our current work, we use this approximation to share information between parallel computations (see Section 3.1).

2.2 Extension to Multi-view Settings

Manifold relevance determination (MRD) [Damianou *et al.*, 2012] extends GP-LVM to a multi-view setting by reformulating the likelihood (1) as $\prod_{v\in\mathcal{V}} p(\mathbf{Y}^v|\mathbf{X},\boldsymbol{\theta}^v)$, where the elements of $\mathcal{V}$ index the *data views*, i.e. matrices conditionally independent given a single latent matrix $\mathbf{X}$. In matrix completion problems, one of the views is typically the target in which prediction is carried out, while the other views constitute side-data. When predicting values in completely unobserved (or new) rows or columns in the target, predictions are effectively done 'outside' of the observed matrix. This can be done with the help of observed data in the side-views, and is referred to as *out-of-matrix* prediction.

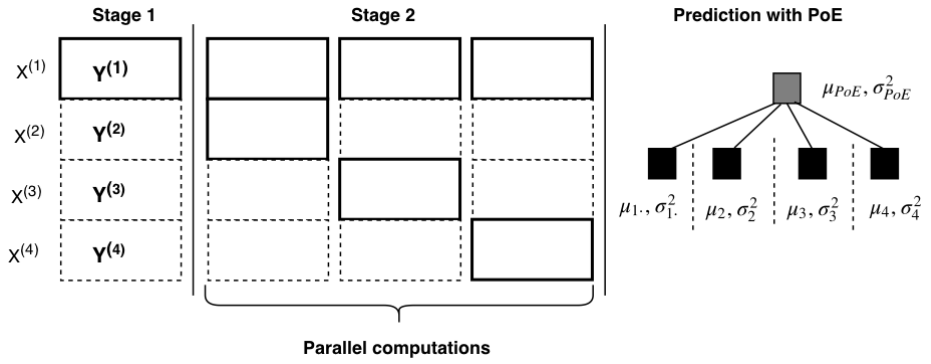

Figure 1: Overview of scalable Bayesian non-linear matrix completion. In the learning phase (left panel), a large matrix is partitioned into 4x1 subsets, which are processed in two stages: in the initial stage only one subset is processed (box with solid borders), after which the remaining subsets are processed in parallel, each coupled with the first subset using incremental learning. The prediction phase (right panel) uses a product of experts, aggregating the means μ_i and variances σ_i^2 of local experts into a joint Gaussian prediction with μ_{PoE} and σ^2_{PoE}.

3 Scalable Matrix Completion Using Bayesian GP-LVM

This section presents a computational strategy, which enables Bayesian GP-LVM to be scaled up for large-scale matrix completion problems using a product of experts (PoE). In brief, we first partition the observation matrix $\mathbf{Y}$ into I disjoint subsets $\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)} \ldots, \mathbf{Y}^{(I)}$ for parallel processing. To avoid problems resulting from the unidentifiability of latent variable models, we couple the subsets as follows: in an initial stage only one subset is processed; in the second stage, the remaining subsets are processed in parallel using incremental learning (Section 3.1). For each subset, we use an implementation of the variational inference framework for Bayesian GP-LVM [Titsias and Lawrence, 2010]. Finally, with a set of independently learned Bayesian GP-LVMs, we use PoE to predict unobserved matrix elements (Section 3.2).

The proposed strategy is summarized in Figure 1. In Section 3.3, we present a further variant of the method, which uses intermediate aggregation of the submodels, prior to PoE, to improve predictive performance. The scalability of our method is briefly discussed in Section 3.4.

3.1 Coupling Parallel Inferences Using Incremental Learning

To couple the parallel inferences over subsets in the second stage of our method, we use a probabilistic variant of the incremental learning algorithm introduced by [Yao *et al.*, 2011] for online learning of (non-Bayesian) non-linear latent variable models. Let $\mathbf{Y}^{(1)}$ be the submatrix processed in the initial stage. Furthermore, denote by $\mathbf{Y}^{(i)}_{\text{aug}} = [\mathbf{Y}^{(1)}, \mathbf{Y}^{(i)}] \in \mathbb{R}^{(N_1+N_i)\times D}$, $i = 2, \ldots, I$, the combined submatrix obtained by augmenting $\mathbf{Y}^{(i)}$ with $\mathbf{Y}^{(1)}$. The corresponding combined latent matrix is denoted by $\mathbf{X}^{(i)}_{\text{aug}} = [\mathbf{X}^{(1)}, \mathbf{X}^{(i)}] \in \mathbb{R}^{(N_1+N_i)\times K}$.

The objective of incremental learning is to learn the joint latent matrix $\mathbf{X}^{(i)}_{\text{aug}}$ without extensive relearning of $\mathbf{X}^{(1)}$, while still allowing it to be updated. When learning $\mathbf{X}^{(i)}_{\text{aug}}$, Yao et al. [2011] added a regularizer to the log-likelihood to prevent $\mathbf{X}^{(1)}$ from deviating too much from its initial estimate, and to speed up learning. The original incremental learning algorithm used the Frobenius norm $\|\mathbf{X}^{(1)} - \hat{\mathbf{X}}^{(1)}\|^2_F$ to regularize the updated estimate of $\mathbf{X}^{(1)}$. In our current work, we use the KL-divergence $\mathrm{KL}\left(q(\mathbf{X}^{(1)})||\hat{q}(\mathbf{X}^{(1)})\right)$ to ensure that the updated variational posterior approximation $q(\mathbf{X}^{(1)})$ remains close to the initial approximation $\hat{q}(\mathbf{X}^{(1)})$. For $\mathbf{X}^{(i)}$, we use the default prior given in Equation (3). Thus, the variational inference for the incremental learning of Bayesian GP-LVM follows the procedure introduced in Section 2.1, with the KL terms in the lower bound of Eq. (5) taking the following form:

$$\mathrm{KL}\left(q(\mathbf{X}^{(i)}_{\text{aug}})||p(\mathbf{X}^{(i)}_{\text{aug}})\right) = \mathrm{KL}\left(q(\mathbf{X}^{(1)})||\hat{q}(\mathbf{X}^{(1)})\right) + \mathrm{KL}\left(q(\mathbf{X}^{(i)})||p(\mathbf{X}^{(i)})\right).$$

For the augmented subsets, the inducing points and kernel hyperparameters are initialized to values obtained in the initial stage. For initialization of latent variables, we use posterior means for $\mathbf{X}^{(1)}$ and nearest neighbors for $\mathbf{X}^{(i)}$ [Yao *et al.*, 2011].

3.2 Prediction with Product of Experts

Product of experts (PoE) prediction for Gaussian process regression [Deisenroth and Ng, 2015] uses the simple idea of combining predictions made by independent GP-models (i.e. 'experts') as a product:

$$p(y_*|\mathbf{x}_*, \mathbf{X}) = \prod_{i=1}^{I} p_i\left(y_*|\mathbf{x}_*, \mathbf{X}^{(i)}\right), \tag{7}$$

where $\mathbf{x}_*$ is a given test input and y_* is the corresponding output value to be predicted. Under this model, a prediction is proportional to a Gaussian with parameters

$$\mu_*^{\text{poe}} = (\sigma_*^{\text{poe}})^2 \sum_{i=1}^{I} \sigma_i^{-2}(\mathbf{x}_*)\mu_i(\mathbf{x}_*),$$

$$(\sigma_*^{\text{poe}})^{-2} = \sum_{i=1}^{I} \sigma_i^{-2}(\mathbf{x}_*).$$

With latent variable models, the essential difference to the above is that the inputs are unobserved and must therefore be inferred. In matrix completion, we wish to predict the missing part of a partially observed test point $\mathbf{y}_* = (\mathbf{y}_*^O, \mathbf{y}_*^U) \in \mathbb{R}^D$, where $\mathbf{y}_*^O$ are the observed elements (or side views) and $\mathbf{y}_*^U$ are the missing values (or the unobserved target view) to be reconstructed. The prediction task can be finished in two steps. First, we infer the latent input $\mathbf{x}_*$ of the test point, which involves maximizing the variational lower bound on the marginal likelihood

$$p(\mathbf{y}_*^O, \mathbf{Y}) = \int p(\mathbf{y}_*^O, \mathbf{Y}|\mathbf{X}, \mathbf{x}_*, \boldsymbol{\theta}) p(\mathbf{X}, \mathbf{x}_*) \mathrm{d}\mathbf{X} \mathrm{d}\mathbf{x}_*$$

to obtain the approximate posterior $q(\mathbf{X}, \mathbf{x}_*) = q(\mathbf{X})q(\mathbf{x}_*)$. The lower bound has the same form as the learning objective function in Equation (5), but for its maximization, the variational distribution $q(\mathbf{X})$ over latent variables for training data and parameters $\boldsymbol{\theta}$ remains fixed during test time. After obtaining $q(\mathbf{x}_*)$, making predictions for $\mathbf{y}_*^U$ is approached as GP prediction with uncertain inputs [Damianou *et al.*, 2016; Girard *et al.*, 2003].

In our distributed setting, the experts in PoE correspond to submodels learned from the augmented training subsets formed in the incremental learning phase. To correct for the initial subset $\mathbf{Y}^{(1)}$ being used in $I-1$ training sets, we formulate a corrected PoE as follows:

$$p(\mathbf{y}_*|\mathbf{Y}, \boldsymbol{\theta}) = p_1(\mathbf{y}_*|\mathbf{Y}^{(1)}, \boldsymbol{\theta}^{(1)}) \times \prod_{i=2}^{I} \left[p_i(\mathbf{y}_*|\mathbf{Y}_{\text{aug}}^{(i)}, \boldsymbol{\theta}^{(i)}) p_1(\mathbf{y}_*|\mathbf{Y}^{(1)}, \boldsymbol{\theta}^{(1)})^{-1} \right],$$

Finally, denoting the means and variances of the local predictive distributions as $\hat{\mu}_i$ and $\hat{\sigma}_i^2$, respectively, we compute the aggregated statistics of the corrected PoE predictions as:

$$\mu_*^{\text{cpoe}} = (\sigma_*^{\text{cpoe}})^2 \left[\hat{\sigma}_1^{-2} \hat{\mu}_1 + \sum_{i=2}^{I} \left(\hat{\sigma}_i^{-2} \hat{\mu}_i - \hat{\sigma}_1^{-2} \hat{\mu}_1 \right) \right],$$

$$(\sigma_*^{\text{cpoe}})^{-2} = \hat{\sigma}_1^{-2} + \sum_{i=2}^{I} \left[\hat{\sigma}_i^{-2} - \hat{\sigma}_1^{-2} \right].$$

In their distributed GP framework, Deisenroth and Ng [2015] used a re-weighted variant of PoE, which they coined the robust Bayesian committee machine (rBCM). Although rBCM has been shown to outperform the basic PoE for GP-regression, in our current setup, we have not observed any advantage of it over PoE. We have therefore formulated our framework using standard PoE but note that the extension to rBCM is straightforward.

3.3 Improved Solution with Intermediate Aggregation

PoE aggregates predictions from local submodels learned on data subsets, effectively using a block-diagonal approximation of the full-data covariance matrix. With larger submodels, PoE provides a closer approximation to the full covariance matrix, which can be expected to result in better predictive accuracy. Here we introduce an *intermediate aggregation* strategy, by which submodels are aggregated for improved predictive performance, while the initial training of submodels is still done on smaller subsets with lower computational cost. While latent variable models are in general non-identifiable, making a direct aggregation of local models difficult to carry out in a meaningful way, the incremental learning introduced in Section 3.1, encourages identifiability among local models, alleviating the problem.

The aggregation of submodels involves (i) stacking together local variational distributions, which are assumed to be independent across subsets, (ii) concatenating the corresponding data subsets, and finally (iii) aggregating the hyperparameters of the models. The model parameters can be approximated using suitable statistics (e.g. mode, median or mean) of the distributions. In our implementation, we use the mode to approximate the kernel and Gaussian noise variance parameters, and use averaging to estimate inducing variables.

Since the first subset $\mathbf{Y}^{(1)}$ is used multiple times through incremental learning, the corresponding variational distribution $q(\mathbf{X}^{(1)})$ is obtained through the following aggregation:

$$q\left(\mathbf{X}^{(1)}\right) = \hat{q}_1\left(\mathbf{X}^{(1)}\right) \prod_{i=2}^{I} \left[\hat{q}_i\left(\mathbf{X}^{(1)}\right) \hat{q}_1\left(\mathbf{X}^{(1)}\right)^{-1} \right],$$

$$= \prod_{n=1}^{N_1} \mathcal{N}\left(\mathbf{x}_{n,:}|\hat{\mu}_n^*, \hat{S}_n^*\right),$$

where

$$\left[\hat{S}_n^*\right]^{-1} = \left[\hat{S}_n^{(1)}\right]^{-1} + \sum_{i=2}^{I} \left(\left[\hat{S}_n^{(i)}\right]^{-1} - \left[\hat{S}_n^{(1)}\right]^{-1} \right),$$

$$\hat{\mu}_n^* = \hat{S}_n^* \Bigg[[\hat{S}_n^{(1)}]^{-1} \hat{\mu}_n^{(1)} + \sum_{i=2}^{I} \left(\left[\hat{S}_n^{(i)}\right]^{-1} \hat{\mu}_n^{(i)} - \left[\hat{S}_n^{(1)}\right]^{-1} \hat{\mu}_n^{(1)} \right) \Bigg].$$

Above, each of the variational distributions $\hat{q}_i(\mathbf{X}^{(1)})$ is Gaussian, of the form given by Equation (6).

Note that after intermediate aggregation, each training subset is used only once to make predictions, and we may use the ordinary PoE formulation in Equation (7) for prediction.

3.4 Computational Cost

Our method aims to leverage the scaling properties of sparse GP for training and those of PoE for prediction. Thus, for data partitioned into subsets of size N_i, $i = 1, \ldots, I$, and assuming that a sufficient number of parallel workers is available, the time complexity for training is $\mathcal{O}\left(\max_i(N_i M^2) \cdot D\right)$, where $M < N_i$ is the number of inducing points and D reflects the fact that variational updates have to be performed separately for each dimension of sparsely observed data. For prediction, the cost is $\mathcal{O}\left(\max_i(N_i^2)\right)$. For incremental learning and intermediate aggregation, N_i refers to the size of the concatenation of multiple subsets. By intermediate aggregation of submodels, we are able to trade off prediction cost against accuracy.

4 Experiments

In this section, we evaluate the predictive performance of the proposed method for out-of-matrix prediction problems on

simulated and real-world chemogenomic data, and compare it with two alternative approaches: (i) the embarrassingly parallel or subset of data (SoD) approach, which has been widely studied to scale up Gaussian Process regression models, and (ii) Macau, Bayesian multi-relational factorization with side information [Simm *et al.*, 2015], supporting out-of-matrix prediction. Macau is implemented with highly optimized C libraries, and is available for experiments on large-scale data. The comparison with the SoD approach shows the advantage of our method in sharing information among submodels, while the comparison with Macau shows the benefit of using Bayesian non-linear matrix factorization. We emphasize, however, that the choice and effectiveness of a model always depends on the problem at hand.

Simulated data. We generated synthetic data using non-linear signals corrupted with Gaussian noise, using matern data generator available in the GPy[1] software. The data has three views $\mathbf{Y} = \{\mathbf{Y}^v : v = 1, 2, 3\}$, the dimension of the views is as follows: N = 25,000, D^1 = 150, D^2 = 100, D^3 = 150. As the task is to perform out-of-matrix prediction, we randomly selected 20% of the rows as a test set, using the remaining rows as the training set. In addition, 80% of the data in the first view were masked as missing, to simulate the sparsely observed target data in a real-world application. The other two views were regarded as side information and were fully observed. Unlike Bayesian GP-LVM, Macau cannot handle missing values in the side data.

Real-world data. We performed the experiments on ExCAPE-DB data [Sun *et al.*, 2017], which is an aggregation of public compound-target bioactivity data and describes interactions between drugs and targets using the pIC50[2] measure. It has 969,725 compounds and 806 targets with 76,384,863 observed bioactivities. The dataset has 469 chem2vec features as side information which are generated from ECFP fingerprint features for the compounds using word2vec software. We used 3-fold cross validation to split the training and test set, where about 30% of the rows or compounds were chosen as test set in each fold.

4.1 Experimental Setup

The experimental setting for MRD models is: number of inducing points 100, optimization through scaled conjugate gradients (SCG) with 500 iterations. For the SoD approach, the latent variables were initialized with PPCA method. We ran Macau with Gibbs sampling for 1200 iterations, discarded the first 800 samples as burn-in and saved every second of the remaining samples yielding in total 200 posterior samples. We set the dimension of latent variables K=10 for ExCAPE-DB data, K=5 for simulated data for all methods.

For the proposed and SoD methods, we partitioned the simulated data into 10x1 subsets and ExCAPE-DB data into 400x1 subsets. Other partitions are also possible; we have chosen the size of subsets such that Bayesian inference could be performed for the subsets in reasonable time on a single CPU. Notice that the views with missing values are generally sparsely observed in many real-world applications, which makes it challenging to learn informative models for such data. Following Qin et al. [2019], we reordered the rows and columns of training data in descending order according to the proportion of observations in them. This makes the first subset the most densely observed block, thus making the resulting submodel informative and facilitating the parallel inferences in the following stages.

We evaluated performance by predictive accuracy and the quality of prediction for downstream ranking tasks. Root mean squared error (RMSE) is a common performance measure for matrix completion. In real-world applications, such as item recommendation or drug discovery, we are more interested in the performance of the ranking task, for instance how many of the recommended items the user actually clicks or buys, how many drugs recommended by models actually have the desired effect for the disease. For this purpose, we regard matrix completion as a classification task (of whether a prediction is relevant or not at a given threshold), use F1- and AUC-ROC score as performance metrics for ExCAPE-DB. Furthermore, following Qin et al. [2019], we use the wall-clock time[3] to measure the speed-up achieved by parallelization. For our method, the reported wall-clock time is calculated by summing the maximum wall-clock times of submodels for each inference stage plus the wall-clock time of making prediction.

For compound activity prediction tasks, we use a pIC50 cutoff (a.k.a. affinity level) at 5 and 6, corresponding to concentrations of 10μM and 1μM, respectively. The test set was further filtered by only keeping targets having at least 100 compounds, at least 25 active compounds, and 25 inactive compounds, to ensure a minimum number of positive and negative data points.

Macau[4] was run on compute nodes with 20 CPUs; all the other methods were run on a single CPU. Our implementation is based on the GPy[1] package.

4.2 Results

The results for simulated and ExCAPE-DB data are given in Table 1 and 2, respectively. In Table 1, column 'Full posterior' refers to the performance of MRD learned from the full data; column 'Intermediate aggregation' refers to our method which works by first aggregating multiple submodels into a model with reasonable size (as long as the compute node can still accommodate the model to make predictions) and then perform predictions by aggregating predictions from multiple experts with PoE.

It is clear from Table 1 that the model with full posterior performs better than other methods in terms of predictive performance; our intermediate aggregation method achieves competitive results while being much faster than the full posterior. The intermediate aggregation method also performs

[1]https://sheffieldml.github.io/GPy/

[2]IC50 (units in μM) is the concentration of drug at which 50% of the target is inhibited. The lower the IC50 of the drug, the less likely the drug will be to have some off-target effect (e.g. potential toxicity) that is not desired. pIC50 $= -\log_{10}(\text{IC50})$.

[3]Wall-clock time measures the real time between the start and the end of a program. For parallel processes, we use the wall-clock time of the slowest process.

[4]We ran the Macau version available in SMURFF software: https://github.com/ExaScience/smurff.

Kernel	Macau	SoD	Proposed methods		Full posterior
			PoE	Intermediate aggregation	
RMSE: the smaller, the better.					
Linear	0.8927 ± 0.010	0.747 ± 0.034	0.685 ± 0.041	$\mathbf{0.656 \pm 0.038}$	$\mathbf{0.656 \pm 0.039}$
RBF	-	0.825 ± 0.034	0.791 ± 0.048	0.683 ± 0.038	$\mathbf{0.658 \pm 0.039}$
Matern32	-	0.824 ± 0.032	0.772 ± 0.048	0.687 ± 0.039	$\mathbf{0.656 \pm 0.038}$
Spearman correlation score: the larger, the better.					
Linear	0.6971 ± 0.044	0.713 ± 0.056	0.726 ± 0.048	$\mathbf{0.744 \pm 0.044}$	$\mathbf{0.744 \pm 0.045}$
RBF	-	0.689 ± 0.060	0.651 ± 0.080	0.721 ± 0.048	$\mathbf{0.742 \pm 0.045}$
Matern32	-	0.684 ± 0.064	0.672 ± 0.083	0.718 ± 0.047	$\mathbf{0.744 \pm 0.044}$
Wall-clock time (secs.)					
Linear	171.44	27730.748	55191.296	57055.027	249782.967
RBF	-	31371.620	53211.605	57306.449	176075.462
Matern32	-	36757.294	67197.138	71945.822	106303.016

Table 1: Comparison of performance metrics for different methods on simulated data. The results are averaged over 5 folds.

Affinity level	Model	RMSE	F1-score	AUC-ROC score	Ratio of successful queries	Wall-clock time (secs.)
5	Macau	1.108 ± 0.069	0.886 ± 0.003	0.805 ± 0.009	0.319 ± 0.024	37041.8
5	SoD	0.914 ± 0.023	0.890 ± 0.011	0.791 ± 0.003	0.363 ± 0.006	63110.16
Proposed methods:						
5	PoE	0.831 ± 0.021	0.900 ± 0.001	0.805 ± 0.002	0.309 ± 0.008	92419.06
5	Intermediate aggregation (nAggs=10)	0.743 ± 0.009	$\mathbf{0.919 \pm 0.005}$	0.811 ± 0.006	0.405 ± 0.018	93331.23
5	Intermediate aggregation (nAggs=20)	$\mathbf{0.736 \pm 0.004}$	0.914 ± 0.003	$\mathbf{0.813 \pm 0.004}$	$\mathbf{0.455 \pm 0.008}$	100492.9
6	Macau	1.123 ± 0.065	0.783 ± 0.013	0.799 ± 0.006	0.318 ± 0.003	37041.8
6	SoD	0.930 ± 0.021	0.787 ± 0.011	0.791 ± 0.005	0.381 ± 0.019	63110.16
Proposed methods:						
6	PoE	0.837 ± 0.022	0.846 ± 0.011	0.811 ± 0.003	0.285 ± 0.004	92419.06
6	Intermediate aggregation (nAggs=10)	$\mathbf{0.775 \pm 0.028}$	$\mathbf{0.851 \pm 0.015}$	$\mathbf{0.817 \pm 0.003}$	0.376 ± 0.025	93331.23
6	Intermediate aggregation (nAggs=20)	0.789 ± 0.006	0.838 ± 0.005	0.816 ± 0.004	$\mathbf{0.434 \pm 0.016}$	100492.9

Table 2: Comparison of RMSE, F1-score, AUC-ROC score and the ratio of successful queries (i.e. queries with AUC-ROC score larger than 0.7 for the targets) for out-of-matrix prediction on ExCAPE-DB by different methods. The first three metrics are calculated for only the successful queries, the ratio is defined as #successfulQueries / #validQueries, where a query target is considered to be valid if it has at least 100 observed bioactivity, at least 25 active and 25 inactive compounds. The results are averaged over 3 runs.

better than the SoD approach and the variant of our method without the intermediate aggregation step. With a linear kernel, all MRD models perform better than Macau.

For ExCAPE-DB data, our intermediate aggregation method (by aggregating 10 or 20 submodels to obtain larger models for prediction) performs much better than all the other methods in all performance metrics for different affinity levels. At both affinity levels, all versions of our proposed method perform better than the SoD method in terms of RMSE, F1-score and AUC-ROC score. Again, we observed that all MRD methods perform better than Macau in all performance metrics. In both tables, the wall-clock times of our methods are larger than that of the SoD approach. This is due to the two-stage parallel inferences of the proposed scheme.

To summarise, the proposed method with an intermediate aggregation step achieves a better trade-off between predictive accuracy and computation time. The proposed method performs better than the embarrassingly parallel approaches for scalable Gaussian process models and a state-of-the-art highly optimized implementation of linear Bayesian matrix factorization with side information.

5 Conclusion

In this paper, we have introduced a scalable approach for Bayesian non-linear matrix completion. We have argued that a key factor in constructing distributed solutions for massive-scale data is to limit the communication required between computational units. To this end, we have introduced a computational scheme which leverages embarrassingly parallel techniques developed for Gaussian process regression by suitably adapting them for Bayesian Gaussian process latent variable models. The resulting framework is almost as communication-efficient as embarrassingly parallel computation, adding only one additional stage of communication, while achieving accuracy close to the non-distributed full data solution.

Acknowledgements

The authors gratefully acknowledge the computational resources provided by the Aalto Science-IT project and support by the Academy of Finland (Finnish Center for Artificial Intelligence, FCAI, and projects 319264, 292334).

References

[Cao and Fleet, 2014] Yanshuai Cao and David J. Fleet. Generalized product of experts for automatic and principled fusion of Gaussian process predictions. In *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS*, 2014.

[Dai *et al.*, 2014] Zhenwen Dai, Andreas Damianou, James Hensman, and Neil Lawrence. Gaussian process models with parallelization and GPU acceleration. *arXiv preprint arXiv:1410.4984*, 2014.

[Damianou *et al.*, 2012] Andreas Damianou, Carl Henrik Ek, Michalis K. Titsias, and Neil D. Lawrence. Manifold relevance determination. In *ICML*, 2012.

[Damianou *et al.*, 2016] Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *JMLR*, 17(42):1–62, 2016.

[Deisenroth and Ng, 2015] Marc Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *ICML*, pages 1481–1490, 2015.

[Gal *et al.*, 2014] Yarin Gal, Mark van der Wilk, and Carl Edward Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In *NIPS*, pages 3257–3265, 2014.

[Girard *et al.*, 2003] Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *NIPS*, pages 545–552, 2003.

[Lawrence and Urtasun, 2009] Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In *ICML*, pages 601–608, 2009.

[Lawrence, 2005] Neil Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *JMLR*, 6(Nov):1783–1816, 2005.

[Ng and Deisenroth, 2014] Jun W. Ng and Marc P. Deisenroth. Hierarchical mixture-of-experts model for large-scale Gaussian process regression. *arXiv preprint arXiv:1412.3078*, 2014.

[Qin *et al.*, 2019] Xiangju Qin, Paul Blomstedt, Eemeli Leppäaho, Pekka Parviainen, and Samuel Kaski. Distributed Bayesian matrix factorization with limited communication. *Machine Learning*, pages 1–26, 2019.

[Quiñonero Candela and Rasmussen, 2005] Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *JMLR*, 6:1939–1959, 2005.

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

[Salakhutdinov and Mnih, 2008b] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, 2008.

[Simm *et al.*, 2015] Jaak Simm, Adam Arany, Pooya Zakeri, Tom Haber, Jörg K. Wegner, Vladimir Chupakhin, Hugo Ceulemans, and Yves Moreau. Macau: Scalable bayesian multi-relational factorization with side information using mcmc. *arXiv preprint arXiv:1509.04610*, 2015.

[Snelson and Ghahramani, 2006] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264. MIT Press, 2006.

[Sun *et al.*, 2017] Jiangming Sun, Nina Jeliazkova, Vladimir Chupakhin, Jose-Felipe Golib-Dzib, Ola Engkvist, Lars Carlsson, Jörg Wegner, Hugo Ceulemans, Ivan Georgiev, Vedrin Jeliazkov, Nikolay Kochev, Thomas J. Ashby, and Hongming Chen. ExCAPE-DB: an integrated large scale dataset facilitating big data analysis in chemogenomics. *Journal of Cheminformatics*, 9(1):17:1–17:9, 2017.

[Titsias and Lawrence, 2010] Michalis Titsias and Neil Lawrence. Bayesian Gaussian process latent variable model. In *AISTATS*, pages 844–851, 2010.

[Tresp, 2000] Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[Vander Aa *et al.*, 2016] Tom Vander Aa, Imen Chakroun, and Tom Haber. Distributed Bayesian probabilistic matrix factorization. In *2016 IEEE International Conference on Cluster Computing*, pages 346–349, 2016.

[Yao *et al.*, 2011] Angela Yao, Juergen Gall, Luc V Gool, and Raquel Urtasun. Learning probabilistic non-linear latent variable models for tracking complex activities. In *NIPS*, pages 1359–1367, 2011.