Multi-output Gaussian Processes for Uncertainty-aware Recommender Systems

Yinchong Yang[1]   **Florian Buettner**[*2,3,4,5]

[1, 2]Siemens AG
[1, 2]Contributed equally to this work
[3]German Cancer Consortium
[4]German Cancer Research Center
[5]Frankfurt University

Abstract

Recommender systems are often designed based on a collaborative filtering approach, where user preferences are predicted by modelling interactions between users and items. Many common approaches to solve the collaborative filtering task are based on learning representations of users and items, including simple matrix factorization, Gaussian process latent variable models, and neural-network based embeddings. While matrix factorization approaches fail to model nonlinear relations, neural networks can potentially capture such complex relations with unprecedented predictive power and are highly scalable. However, neither of them is able to model predictive uncertainties. In contrast, Gaussian Process based models can generate a predictive distribution, but cannot scale to large amounts of data. In this manuscript, we propose a novel approach combining the representation learning paradigm of collaborative filtering with multi-output Gaussian processes in a joint framework to generate uncertainty-aware recommendations. We introduce an efficient strategy for model training and inference, resulting in a model that scales to very large and sparse datasets and achieves competitive performance in terms of classical metrics quantifying the reconstruction error. In addition to accurately predicting user preferences, our model also provides meaningful uncertainty estimates about that prediction.

1 INTRODUCTION

Collaborative filtering (CF) provides a powerful solution to recommender systems. Recommending a new item to a user is based on the assumption that users demonstrating similar rating or purchasing patterns are interested in similar items. A database describing such user-item interactions often takes the form of a matrix, where each entry describes the interaction between one user and one item. The overall rating or purchasing pattern of a user can therefore be described by the corresponding row in such a matrix. However, since there are typically large numbers of users and items in the database, and each user is usually only interested in a small subset of items, this user-item matrix is often large and sparse. It is therefore inefficient to define the similarity between users in the high dimensional feature space defined by all items. Instead, it is more advantageous to derive abstract feature vectors that represent users and items, which inspired a large variety of low-rank matrix decomposition models such as non-negative matrix decomposition [Zhang et al., 2006], biased matrix decomposition [Koren et al., 2009] and non-parametric decomposition [Yu et al., 2009]. These methods aim at learning low dimensional representations for all users and items, allowing for the prediction of the unobserved interaction between a new pair of user and item.

Most of these state-of-the-art representation learning methods in form of matrix decomposition, however, focus on point estimates and do not quantify any predictive uncertainty. This limits the applicability of these methods in safety-critical situations. For instance, the recommendation of a treatment in the context of clinical decision support needs to be interpreted with much more caution than the recommendation of movies, and transparent uncertainty estimates are crucial to assess the quality of such recommendation. Similarly, when recommender systems are used to detect novel drug-target interactions to facilitate new drug discoveries, it is crucial for such systems to be uncertainty aware, since large costs are associated with studies required for target validation. Other applications could arise in the context of multi-armed bandits for exploration and exploitation during recommendation. Furthermore, missing ratings in collaborative filtering tasks are usually not completely

---

[*] Work done for Siemens AG

random [Schnabel et al., 2016, Marlin et al., 2012], and good performance in terms of RMSE and MAE does not always promise good recommendations[Cremonesi et al., 2010]. In this context, the predictive uncertainty serves as a practical way to make this bias in the training data transparent in a principled manner. The model can communicate the uncertainty of a prediction to the user, such that the user knows which individual predictions are likely to translate into good recommendations.

Gaussian processes (GP) are a well known class of models that generates prediction uncertainty alongside point estimates via its kernel function, which measures the similarity between data samples. While Gaussian processes are primarily used to solve a wide range of supervised modelling tasks, they have also been applied to perform probabilistic matrix decomposition with a focus on dimensionality reduction, via the Gaussian Process Latent Variable Model (GPLVM). Here, a Gaussian process is used as a prior distribution for a function that maps a low-dimensional latent space to the high-dimensional data matrix. The individual components of this mapping function are modelled as independent draws from a Gaussian process, assuming all dimensions of the data matrix are independent. This independence assumption is problematic in the context of collaborative filtering, where modelling the interdependency of both users and items has proven beneficial [Dong et al., 2014, Nickel et al., 2011]. While GPLVM has been used as matrix factorization method for collaborative filtering in the past [Lawrence and Urtasun, 2009], as for other matrix decomposition approaches, practical applicability can be limited also for computational reasons. Being designed primarily as a dimensionality reduction method for data with few missing values, it is not readily amenable to the triple-based learning paradigm of collaborative filtering. That is, one trains a learning model with data samples in form of triples of indices and the corresponding target values, e.g. $(i, j, y_{i,j})$. The learning model will have to query the corresponding rows i and j from the respective factor matrices and attempt to regress them to the target entry $y_{i,j}$. This paradigm facilitates efficient modeling of large and highly sparse datasets such as large user-item matrices or even tensor representation of knowledge graphs [Nickel et al., 2015]. However, the GPLVM-based recommender systems typically require storing the dense data matrix. More specifically, in the triple-based learning paradigm we only iterate through all observed, i.e., non-zero scalar entries in a sparse matrix.

In this work, we propose a novel approach to combine multi-output Gaussian processes with representation learning for collaborative filtering via matrix decomposition. We learn a latent representation vector for each entity type (i.e. users and items) jointly with a multi-output GP that predicts not only a point estimate of the user-item interaction, but also its predictive variance. We thereby explicitly model not only dependencies between users, but also between items. We

motivate this design by connections to coregionalization with separable kernels in multi-output Gaussian processes as well as to the GPLVM. We exploit variational sparse approximations [Titsias, 2009, Hensman et al., 2015] and combine them with a sparse matrix representation so that our approach scales to datasets consisting of large numbers of users and items via minibatch learning. Furthermore, we propose a new approach for evaluating and visualizing the quality of predictive variance of such uncertainty-aware models by stratifying predictions by their associated uncertainty.

2 RELATED WORK

Matrix decomposition methods applied to collaborative filtering and recommender systems attempt to learn explanatory latent features for users and items. Non negative matrix factorization [Luo et al., 2014, Zhang et al., 2006] restrict the factor matrices to be non-negative, in order to enable better interpretability of the representations. Such methods are trained to minimize the reconstruction error in terms of the entire matrix, and are therefore no longer efficient or reliable in case of high sparsity in the matrix. In contrast, biased matrix factorization [Koren et al., 2009, Gomez-Uribe and Hunt, 2015] has proven to be very efficient in decomposing sparse matrices by training only based on observed entries. However, these methods are prone to overfitting and require careful tuning and validation [Nickel et al., 2015].

The Gaussian Process Latent Variable Model (GPLVM) [Lawrence, 2004] applies GP regression in an unsupervised setting. Given a matrix $\boldsymbol{Y}$, the GPLVM learns a latent representation vector $\boldsymbol{a}_i$ for each row $\boldsymbol{y}_i$. A GP regression model is then used to map the unobserved latent variables $\boldsymbol{a}_i$ to the multiple output dimensions $\boldsymbol{y}_{\cdot,j}$. Since the latent variables are of lower dimensionality than the outputs, GPLVM can be seen as a dimension reduction approach. While GPLVM models are typically applied to dimensionality reduction tasks, [Lawrence and Urtasun, 2009] apply GPLVM to large-scale matrices to perform item recommendation by collaborative filtering. While the GPLVM is usually viewed as a matrix factorization approach and has also an interpretation as probabilistic PCA, it can also be interpreted as a multi-output GP regression model where the vectors of different outputs are drawn independently from the same Gaussian process prior.

Multi-output GP regression models can be used to predict vector-valued functions in a supervised manner by constructing a covariance matrix that describes the dependencies between all the input and output variables. Most implementations of multi-output GPs are formulated around the framework of the Linear Model of Coregionalization (LMC) [Alvarez et al., 2012], in the context of supervised learning. Within this framework, dependencies between individual regressors are modeled with separable kernels (or the sum

thereof), which can be written as the product of two kernels. The first kernel measures similarity of samples in the input space, while the second kernel captures the similarity between each pair of output dimensions.

[Dai et al., 2017] extend multi-output GPs by representing the output dimensions with unobserved latent variables. Thereby they avoid problems with overfitting caused by having to estimate all parameters of the full covariance matrix in standard LMC, allow for the prediction of new outputs at test time and facilitate a lower computational complexity. They further make sure their model scales to large datasets by utilizing the sparse Gaussian process approximation [Titsias, 2009, Titsias and Lawrence, 2010] (SVGP). In the SVGP framework, the model is augmented by an auxiliary variable and the covariance matrix is computed on a set of inducing inputs that represent the whole dataset. The kernel distance between two input samples can then be factorized into their respective kernel distances to inducing points, thus avoiding the need to compute a full covariance matrix. For a systematic overview and comparison of sparse approximations, we refer to [Quiñonero-Candela and Rasmussen, 2005] and [Bui et al., 2017].

3 PRELIMINARY: MATRIX FACTORIZATION FOR COLLABORATIVE FILTERING

In collaborative filtering, the dataset often takes the form of a user-item matrix $\boldsymbol{Y} \in \mathbb{R}^{I \times J}$, assuming I users and J items. Each entry $y_{i,j}$ describes the interaction between user i and item j, for example in form of a rating or a purchase. Such a matrix is typically large and sparse, first due to the large number of users and items, and second due to the fact that each user is usually interested in a very small subset of items only. In order to derive new recommendations for a user, one has to predict their potential interest in an item with which no interaction has taken place yet, i.e. where there is no entry in the user-item matrix. A typical solution to this collaborative filtering task is via a matrix decomposition approach.

During training, one fits a decomposition model that can recover all entries in the user-item matrix to a certain extent. As a simple illustration, the user-item matrix can be modeled as $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{B}^T$, $\boldsymbol{A} \in \mathcal{A} = \mathbb{R}^{I \times r}, \boldsymbol{B} \in \mathcal{B} = \mathbb{R}^{J \times r}$, or equivalently, $y_{i,j} = \boldsymbol{a}_i^T \boldsymbol{b}_j \ \forall (i,j) \in [1, I] \times [1, J]$. At inference time, any entry value that is not observed in the training data, can be predicted by reading the corresponding location on the reconstructed user-item matrix: $\widehat{\boldsymbol{Y}} = \boldsymbol{A}\boldsymbol{B}^T$. The generalization power of the decomposition model lies in the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, which can be interpreted as latent representations of the users and items, respectively. In order to achieve better modeling performance, it is possible to apply more complex functions than the

dot product to join the latent representations. Writing the model definition as $y_{i,j} = g(\boldsymbol{a}_i, \boldsymbol{b}_j) \ \forall i, j \in [1, I] \times [1, J]$, the function g can be realized in various ways. For instance, SVD defines $g(\boldsymbol{a}_i, \boldsymbol{b}_j) = (\boldsymbol{\lambda} \circ \boldsymbol{a}_i)^T \boldsymbol{b}_j$, where $\boldsymbol{\lambda}$ consists of the singular values and $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ are restricted to be orthogonal. In case of biased matrix decomposition [Koren, 2009], we have $g(\boldsymbol{a}_i, \boldsymbol{b}_j) = \boldsymbol{a}_i^T \boldsymbol{b}_j + \alpha_i + \beta_j$, where α_i and β_j are user- and item-specific biases parameters, respectively. Multi-way neural networks [Dong et al., 2014], if applied to the matrix case, take the form of $g(\boldsymbol{a}_i, \boldsymbol{b}_j) = \boldsymbol{u}^T \sigma(\boldsymbol{w}^T [\boldsymbol{a}_i, \boldsymbol{b}_j] + w_0) + u_0$. Here, the latent representations are concatenated and fed into a multi-layer perceptron.

The GPLVM [Lawrence, 2004, 2007] solves the matrix decomposition task by fitting J independent GP regression models on unobserved latent variables $\boldsymbol{x}_i$ with $p(\boldsymbol{y}_{\cdot,j}) = \mathcal{N}(\boldsymbol{y}_{\cdot,j} \mid 0, \boldsymbol{K}^j + \sigma^2 \boldsymbol{I})$, where $\boldsymbol{K}^j \in \mathbb{R}^{I \times I}$ captures the covariances between each pair of latent representation $(\boldsymbol{a}_i, \boldsymbol{a}_{i'})$ defined by the covariance function $k(\boldsymbol{a}_i, \boldsymbol{a}_{i'})$. In other words, the GP-LVM can be interpreted as a multiple-output GP regression model where only the output data are given and the unobserved inputs are being optimized.

It is worth noting that the training target in case of standard GPLVM implementations is $\boldsymbol{y}_{\cdot,j}$. If the target matrix, and hence each column $\boldsymbol{y}_{\cdot,j}$, is sparse, such dense representations are computationally inefficient. In case of (biased) matrix decomposition and multi-way neural networks, however, one could formulate a training sample in form of a triple $(i, j, y_{i,j})$ and only iterate through the observed non-zero target values.

4 MULTI-OUTPUT GAUSSIAN PROCESS MODEL FOR COLLABORATIVE FILTERING

4.1 NOTATION

We denote a GP regression on target $\boldsymbol{y}$ as

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I}),$$

where σ is the hyper parameter that defines the random noise level. The vector $\boldsymbol{f}$ denotes the vector consisting of values produced by the regression model from all input data $\boldsymbol{x}_i$, which follows another multi-variate Gaussian distribution:

$$\boldsymbol{f} = (f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n))^T,$$
$$p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{0}, \boldsymbol{K}),$$

where we denote the covariance matrix using $\boldsymbol{K}$.

4.2 MODEL DEFINITION

The GPLVM treats all output dimensions in the matrix as independent, which is an assumption that may not always

hold, in particular if the matrix describes user-item interactions. For instance, the items in which a user takes interest could very well be correlated or similar [Nickel et al., 2015, Agrawal et al., 1994]. To capture such dependencies between output dimensions, we propose a new coregionalization kernel to perform multi-output Gaussian Process regression for unsupervised matrix decomposition in the spirit of the GPLVM.

Our proposed separable kernel can be written as the product of two individual kernels, where the first kernel measures similarity of samples in the input space and the second kernel captures the similarity between each pair of output dimensions. More formally, this kernel takes the form

$$(\boldsymbol{K})_{j,j'} = k^A(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) k^W(j, j')$$

or, equivalently,

$$\boldsymbol{K} = k^A(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) \boldsymbol{W},$$

where k^A and k^W are scalar kernels on $\mathcal{A} \times \mathcal{A}$ and $\{1, \ldots, J\} \times \{1, \ldots, J\}$ respectively and $\boldsymbol{W}_{i,j} = k^W(i, j)$ is a symmetric and positive semi-definite matrix, which models the dependency between each pair of outputs. While coregionalization is usually performed in the context of supervised regression, in our application the inputs $\boldsymbol{a}_i$ are unobserved and, as for the GPLVM model, need to be optimized. $\boldsymbol{W}$ being the identity matrix implies independence between outputs and the model falls back to the standard GPLVM. There is a variety of approaches for choosing $\boldsymbol{W}$, ranging from the design of a symmetric and positive semi-definite matrix based on suitable regularizers to the choice of covariance functions for the different output components [Alvarez et al., 2012].

Here, we choose to replace the coregionalization matrix $\boldsymbol{W}$ by a kernel on latent representations of items in vector space $\mathcal{B}$, such that the covariance matrix can be written as a Kronecker product of the covariance matrix of the latent variables representing the items $\boldsymbol{K}^{\mathcal{B}}$ and the covariance matrix of the latent variables representing the users $\boldsymbol{K}^{\mathcal{A}}$. We thereby combine representation learning with coregionalization in multi-output GPs for collaborative filtering via unsupervised matrix factorization.

Note that we have briefly introduced Latent Variable Multiple Output Gaussian Processes (LVMOGP) [Dai et al., 2017] as a related method in Sec. 2. In the context of supervised learning, LVMOGP augments multi-output GPs by latent variables, capturing latent information on the outputs. While the authors also replace $\boldsymbol{W}$ by a kernel on latent representations, in their supervised scenario, inputs are observed, whereas in our model both kernels act on latent variables. In Tab. 1 we compare the most closely related methods to ours, including multi-output GP regression, GPLVM and LVMOGP, in three aspects of learning task, output and input treatment, respectively.

Taken together, the kernel of our proposed model can be written as

$$\boldsymbol{K}^{coreg} = \boldsymbol{K}^{\mathcal{A}} \otimes \boldsymbol{K}^{\mathcal{B}}$$
$$\text{with} \begin{cases} \boldsymbol{K}^{\mathcal{A}}_{i,i'} = k^{\mathcal{A}}(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) = k_{\boldsymbol{A}}(i, i'), \\ \boldsymbol{K}^{\mathcal{B}}_{j,j'} = k^{\mathcal{B}}(\boldsymbol{b}_j, \boldsymbol{b}_{j'}) = k_{\boldsymbol{B}}(j, j'). \end{cases}$$

For the sake of symmetry, we choose the same covariance function for $k^{\mathcal{A}}$ and $k^{\mathcal{B}}$. As for matrix $\boldsymbol{K}^{\mathcal{A}}$, each element in $\boldsymbol{K}^{\mathcal{B}}$ measures the similarity between a pair of output dimensions (or items) (j, j'). As the rightmost term of the equations above implies, we could treat all latent variables in matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as hyper parameters in a specific kernel [Snelson and Ghahramani, 2006] that takes as input the indices. From a functional perspective, our proposed kernel

$$k^{coreg}((i, j), (i', j')) = k_{\boldsymbol{A}}(i, i') \cdot k_{\boldsymbol{B}}(j, j'),$$

measures in fact the similarity between user-item pairs (i, j) and (i', j'). The measurement is carried out by evaluating the two kernel functions at (i, i') and (j, j'), respectively, and calculating the product of both kernel values. The training samples can be thus formulated using only the indices and corresponding entries in the matrix: $(i, j, y_{i,j}) \ \forall i, j \in [1, I] \times [1, J]$. This data format is especially advantageous in case of large number of items and high sparsity in matrix $\boldsymbol{Y}$ since it does not require storing the data in dense matrix format.

We finally write our model as: $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}^{coreg} + \sigma^2 \boldsymbol{I}) \in \mathbb{R}^{I \cdot J}$, $\boldsymbol{K}^{coreg} \in \mathbb{R}^{(I \cdot J) \times (I \cdot J)}$. The vector $\boldsymbol{f}$ has length $I \cdot J$, indicating that it consists of the outcome of all possible combinations of I users and J items.

4.3 MODEL FITTING

The major challenge in fitting such a model lies in the size of the covariance matrix $\boldsymbol{K}^{coreg} \in \mathbb{R}^{(I \cdot J) \times (I \cdot J)}$, since there are typically large numbers of users and items. Computing the inverse of the covariance matrix, which is required for computing the log marginal likelihood, has a complexity of $\mathcal{O}(n^3)$ where n is the number of training samples, in our case $n = I \cdot J$. We take sparse GP approaches to address this challenge. A sparse GP model introduces $m \ll n$ inducing points, denoted as $\boldsymbol{Z}_m$ that represent the entire dataset. These inducing points are optimized as additional parameters in a GP model. We apply SVGP [Titsias, 2009] and augment the model with inducing variables $\boldsymbol{u}$ with a Gaussian prior $p(\boldsymbol{u})$, that contain the values of function f at inducing points $\boldsymbol{Z}_m$. With this augmentation, a variational lower bound of the marginal log likelihood can be derived

	Multi-output GPR	GPLVM	LVMOGP	Ours
Unsupervised task	×	✓	×	✓
Dependency between outputs	✓	×	✓	✓
Coregionalization by kernel	×	×	✓	✓
Unobserved input	×	✓	×	✓

Table 1: A comparison of our methods with mostly related approaches. Coregionalization by kernel refers to learning latent vectors that represent output dimensions. In contrast, standard coregionalization handles dependency between outputs using a full covariance matrix whose only constraint is being symmetric and positive semi-definite.

[Hensman et al., 2015]:

$$
\begin{aligned}
\log p(\boldsymbol{y}) \geq & \log \mathcal{N}(\boldsymbol{y}|\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}) \\
& - \frac{1}{2\sigma^2} tr(\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{\Sigma}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}) \quad (1) \\
& - \frac{1}{2\sigma^2} tr(\boldsymbol{K}_{nn} - \boldsymbol{Q}_{nn}) - KL(q(\boldsymbol{u})||p(\boldsymbol{u}))
\end{aligned}
$$

with $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ being the variational posterior distribution on the inducing variables $\boldsymbol{u}$, and $\boldsymbol{Q}_{nn} = \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}$. $\boldsymbol{K}_{mm}$ is the covariance function evaluated between all the inducing points and $\boldsymbol{K}_{nm}$ is the covariance function between all inducing points and training points. This formulation enjoys the advantage of being able to factorize across data, i.e., to be updated using single samples or mini batches, and can thus scale to large training data.

The essential motivation of applying inducing points is to avoid calculating and inverting the full covariance matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$. Instead, one only calculates the kernel between a training sample and all inducing points as in $\boldsymbol{K}_{nn}$, and the kernel between two inducing points as in $\boldsymbol{K}_{mm}$, which is defined as follows in our specific case of coregionalization:

$$
\begin{aligned}
\boldsymbol{K}_{mm} = & K(\boldsymbol{Z}^A, \boldsymbol{Z}^A) \otimes K(\boldsymbol{Z}^B, \boldsymbol{Z}^B) \\
& \in \mathbb{R}^{(m_A m_B) \times (m_A m_B)}.
\end{aligned}
$$

SVGP therefore reduces the computational complexity from $\mathcal{O}((I \cdot J)^3)$ to $\mathcal{O}((I \cdot J) \cdot (m_A \cdot m_B)^2)$. Our triple-based formulation means that the set of training samples consists only of observed triples; by making use of the SVGP approximation, we only ever need to compute the kernel between such a training sample and all inducing points or between pairs of inducing points. This in turn means that for matrices with sparsity S (fraction of *observed* user-item interactions), the computational complexity reduces to $\mathcal{O}(S \cdot (I \cdot J) \cdot (m_A \cdot m_B)^2)$. In practice, we can further reduce this computational complexity if we choose the same number of inducing points m in $\mathcal{A}$ and $\mathcal{B}$: this allows us to learn coupled pairs of inducing points in both spaces (see next subsection for implementation details). By tying the parameters of both sets of latent representations, we can then reduce the effective size of $\boldsymbol{K}_{mm}$ to $m \times m$ and further reduce computational complexity to $\mathcal{O}(S \cdot (I \cdot J) \cdot m^2)$.

4.4 COUPLING INDUCING POINTS

The adjacency matrix characterizing interactions between users and items in collaborative filtering tasks is typically highly sparse. An efficient representation of such data is via a triple store where an observed training sample is represented as $(i, j, y_{i,j}) \ \forall (i,j) \in [1, I] \times [1, J]$ with users i, items j and corresponding entries in the user-item matrix $y_{i,j}$. In this data formulation for sparse data, input i and output j of one training sample are inherently coupled. We would like for this coupling to be also reflected in the sparse approximation and in particular the choice of inducing points, in order to fully take advantage of the benefits of this data representation in terms of computational complexity.

In standard formulations of SVGP for multi-output GPs, the covariance matrix of the inducing variables $\boldsymbol{K}_{mm}$, is computed as $\boldsymbol{K}_{mm} = \boldsymbol{K}_{mm}^A \otimes \boldsymbol{K}_{mm}^B$. Here, $\boldsymbol{K}_{mm}^A = K^A(\boldsymbol{Z}^A, \boldsymbol{Z}^A)$ and $\boldsymbol{K}_{mm}^B = K^B(\boldsymbol{Z}^B, \boldsymbol{Z}^B)$ are computed on independent sets of inducing points $\boldsymbol{Z}^A$ and $\boldsymbol{Z}^B$ with kernel functions k_A and k_B that represent input space $\mathcal{A}$ and output space $\mathcal{B}$ respectively [Dai et al., 2017]. Similarly, the cross-covariance between training samples and inducing points $\boldsymbol{K}_{nm}$ is constructed as $\boldsymbol{K}_{nm} = \boldsymbol{K}_{nm}^A \otimes \boldsymbol{K}_{nm}^B$, with $\boldsymbol{K}_{nm}^A \in \mathbb{R}^{I \times m_A}$ and $\boldsymbol{K}_{nm}^B \in \mathbb{R}^{J \times m_B}$, where I and J correspond to the number of inputs (users) and outputs (items) as before. $\boldsymbol{K}_{nm}^A$ is the cross-covariance between latent inputs $\boldsymbol{A}$ and $\boldsymbol{Z}^A$ and $\boldsymbol{K}_{nm}^B$ the cross-covariance between latent outputs $\boldsymbol{B}$ and $\boldsymbol{Z}^B$. The covariance between training samples $\boldsymbol{K}_{nn}$ is computed as $\boldsymbol{K}_{nn} = \boldsymbol{K}_{nn}^A \otimes \boldsymbol{K}_{nn}^B$, with $\boldsymbol{K}_{nn}^A$ being the covariance matrix of $\boldsymbol{A}$ constructed with k_A and $\boldsymbol{K}_{nn}^A$ being the covariance matrix of $\boldsymbol{B}$ constructed with k_B.

However, this formulation does not reflect the sparse triple-based data formulation and results in a computational complexity that is quartic in m (c.f. previous section). If we decide to choose the same number of inducing inputs for $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$, we can couple the inducing points for pairs of inputs and outputs and reformulate the construction of $\boldsymbol{K}_{nm}$, $\boldsymbol{K}_{nm}$ and $\boldsymbol{K}_{nn}$, such that it reflects the coupling between input i and output j of a training sample (i, j).

More specifically, when we compute the cross-covariance

between the q-th training sample (i, j) * and the l-th pair of inducing points in order to build $\boldsymbol{K}_{nm}$, we do so by pairing the corresponding latent representations $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ and using a product kernel to compute similarities. That is,

$$
\begin{aligned}
&K_{nm}((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{Z}^A, \boldsymbol{Z}^B))_{q,l} \\
=&k([\boldsymbol{a}_i, \boldsymbol{b}_j], [\boldsymbol{z}_l^A, \boldsymbol{z}_l^B]) \\
=&k_A(\boldsymbol{a}_i, \boldsymbol{z}_l^A) k_B(\boldsymbol{b}_j, \boldsymbol{z}_l^B)
\end{aligned} \tag{2}
$$

Using this paired formulation, it becomes immediately clear that

$$
\boldsymbol{K}_{nm} = K_{nm}((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{Z}^A, \boldsymbol{Z}^B))
$$

becomes substantially smaller, with $\boldsymbol{K}_{nm}^{\mathrm{paired}} \in \mathbb{R}^{(I \cdot J) \times m}$ compared to the standard case where the sets of inducing points are treated as independent and $\boldsymbol{K}_{nm} \in \mathbb{R}^{(I \cdot J) \times (m_A \cdot m_B)}$.

We construct the covariance matrix $\boldsymbol{K}_{mm}$ using the same pairing approach and compute the covariance between the o-th and p-th inducing points as

$$
\begin{aligned}
&K_{mm}((\boldsymbol{Z}^A, \boldsymbol{Z}^B), (\boldsymbol{Z}^A, \boldsymbol{Z}^B))_{o,p} \\
=&k([\boldsymbol{z}_o^A, \boldsymbol{z}_o^B], [\boldsymbol{z}_p^A, \boldsymbol{z}_p^B]) \\
=&k_A(\boldsymbol{z}_o^A, \boldsymbol{z}_p^A) k_B(\boldsymbol{z}_o^B, \boldsymbol{z}_p^B)
\end{aligned} \tag{3}
$$

That means, using paired inducing points, $\boldsymbol{K}_{mm}$ can be written as elementwise product between $\boldsymbol{K}_{mm}^A$ and $\boldsymbol{K}_{mm}^B$: $\boldsymbol{K}_{mm} = \boldsymbol{K}_{mm}^A \odot \boldsymbol{K}_{mm}^B$. As before, $\boldsymbol{K}_{mm}$ becomes substantially smaller since it is constructed via an elementwise product rather than via the Kronecker product.

Finally, we use the same pairing formalism when constructing $\boldsymbol{K}_{nn}$ and compute the covariance between the q-th training sample (i, j) and the l-th training sample (i', j') as

$$
\begin{aligned}
&K_{nn}((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{A}, \boldsymbol{B}))_{k,l} \\
=&k([\boldsymbol{a}_i, \boldsymbol{b}_j], [\boldsymbol{a}_{i'}, \boldsymbol{b}_{j'}]) \\
=&k_A(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) k_B(\boldsymbol{b}_j, \boldsymbol{b}_{j'})
\end{aligned}
$$

Constructing $\boldsymbol{K}_{nn}$ in this manner rather than using the Kronecker product between $\boldsymbol{K}_{nn}^A$ and $\boldsymbol{K}_{nn}^B$ is particularly suited for highly sparse data, where the number of training samples N is substantially lower than total number of possible samples $I \cdot J$. In this case the paired approach directly leads to a natural construction of a small covariance matrix of observed training samples of size $N \times N$†.

Taken together, this means that the paired approach reduces the computational complexity of of our method from $\mathcal{O}(I \cdot J \cdot m_A^2 \cdot m_B^2)$ to $\mathcal{O}(S \cdot (I \cdot J) \cdot m^2)$ (with sparsity S, c.f. section 4.2).

We summarize the training algorithm of our approach as follows. Given the number of inducing pairs m and rank r as hyper parameters: We describe our algorithm as online

---

Algorithm 1: Training algorithm

input : Training N tuples $(i, j, y_{i,j})$ where $i \in [1, I], j \in [1, J]$, and $y_{i,j}$ is the (i, j)-th entry in matrix $\boldsymbol{Y}$;
output : $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Z}^A, \boldsymbol{Z}^B, \boldsymbol{\mu}, \boldsymbol{S}$ and the RBF parameters;
Initialize latent representations $\boldsymbol{A} \in \mathbb{R}^{I \times r}, \boldsymbol{B} \in \mathbb{R}^{J \times r}$, and inducing pairs $\boldsymbol{Z}^A \in \mathbb{R}^{m \times r}, \boldsymbol{Z}^B \in \mathbb{R}^{m \times r}$;
while *not converged* **do**
  for *each training epoch* **do**
    for *a training sample* $(i, j, y_{i,j})$ **do**
      Calculate the covariance matrix of inducing pairs $\boldsymbol{K}_{mm}$ using Eq. (3);
      Derive the bijective index q of (i, j);
      Calculate its covariance values $K_{nm}[q, l]$ with all inducing pairs, using Eq. (2).
      Update the ELBO in Eq. (1) w.r.t. $\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{z}_l^A, \boldsymbol{z}_l^B, \boldsymbol{\mu}, \boldsymbol{S}$ and the RBF kernel parameters.
    end
  end
end

---

training with single samples for a more concise notation as well as to emphasize the scalability of the training. In the step of updating the ELBO, one only needs to iterate once through all inducing pairs. In practice, one would often like to train on batches. The complexity of this case merely becomes the product of batch size and number of inducing pairs. One should also note that in a standard SVGP setting, one updates the kernel parameters and inducing points. In our approach, however, we update additionally the *input* to the GP which are the latent representations.

We have implemented our method using tensorflow/GPflow [De G. Matthews et al., 2017] and provide an easy-to-use python package at `https://github.com/Tuyki/mogp-decomposition` with experimental results. For all experiments, we used an RBF kernel, for both k_A and k_B. Experiments were run on hardware utilising 4 CPUs and one GeForce GTX TITAN X GPU.

4.5 EVALUATION

Due to the fact that the missing entries in CF tasks may be not be completely random, high performance in terms of MAE and RMSE might not always guarantee good recommendations [Marlin et al., 2012, Cremonesi et al., 2010]. To this end, we propose an additional metrics to evaluate uncertainty-aware CF models, including ours and GPLVM, by sorting the point estimates based on the predictive variance. Specifically, we denote the ground truth values as $(y_i)_{i=1}^n$, and the predictive distribution as $(f_i \sim \mathcal{N}(m_i, s_i^2))_{i=1}^n$. We calculate Q quantiles of all the standard deviations $(s_i)_{i=1}^n$. For each quantile $q \in \{\frac{1}{Q}, \frac{2}{Q}, \dots, 1\}$

---

*There is always a bijection / double-indexing relation between (i, j) and q.

†Note: the full covariance matrix is not needed during inference. Model training using SVGP merely requires the computation of the trace of $\boldsymbol{K}_{nn}$

where the predictive standard deviation s_i is smaller than or equal to the q-quantile, we evaluate all point estimates m_i against the corresponding ground truth values y_i. Formally let,

$$d_q = R(\{(y_i, m_i) \mid \forall s_i \leq q\text{-Quantile}\}),$$

where R could be any evaluation metric defined on a set of pairs of ground truth and point estimate. Each d_q is the evaluation of the predictive performance of the model on its top-$q \cdot n$ most confident predictions. Ideally, an uncertainty-aware prediction model is expected to have better performances on test samples about which it demonstrates higher confidence, i.e., lower predictive variance, and vice versa. If we plot all d_q values on y-axis against the corresponding q values on the x-axis, we would expect a monotonously increasing plot line. We name this evaluation approach QP-plot, where Q stands for quantile and P for performance.

5 EXPERIMENTS

We conduct experiments on three well established datasets for evaluating recommender systems: movielens-1m (ML-1M), movielens-10m (ML-10M) and Jester joke recommendations. The task is to predict the rating value of an arbitrary pair of user and item, which can be later interpreted as a score quantifying the interest of the user in the item.

The ML-1M consists of 1000209 ratings of 6040 users on 3706 items, the ML-10M consists of 10000054 ratings of 69878 users on 10677 items, and the jester dataset consists of 1728847 ratings of 50692 users on 140 items. The ML-1M has as ratings only natural numbers between 1 and 5, while ML-10M has rating in the same range but the ratings also include one value between each pair of natural numbers, such as $\{1, 1.5, 2, 2.5, ..., 5\}$. The Jester dataset contains arbitrary real numbers between -10 and 10. Our model, based on a GP regression, models all three types of ratings as real values.

We performed a hyper-parameter search especially with regard to the size of the latent vectors and number of inducing points. On a grid search of batch sizes $\{2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}\}$, a relatively large size of 2^{16} produced the best performance. We also perform a grid search for the optimal number of inducing points. Among $\{32, 64, 128, 256, 512, 1024\}$ inducing points, 128 outperforms other choices. This comparison is also illustrated as QP-plot in supplementary materials. We also find out that the model is robust with respect to the size of latent vector. For all the experiments, we use the best performing size of 8, which is a relatively small number compared to, e.g., SMA [Li et al., 2016], which requires a rank of up to 50.

For each dataset we report the performance in terms of MAE and RMSE on the test set in a 5-fold cross validation setting. Quantitative comparisons regarding the ML-1M,

	RMSE	MAE
CF-NADE [Zheng et al., 2016]	0.829	-
GPLVM [Lawrence and Urtasun, 2009]	0.880	**0.644**
Sparse FC [Muller et al., 2018]	**0.824**	-
GC [Berg et al., 2017]	0.832	-
CWOCF [Lu et al., 2013]	0.958	0.761
LLORMA [Lee et al., 2013]	0.865	-
Biased MF	0.863	0.678
SVD++	0.893	0.705
Bayesian GPLVM at $q = 100\%$	0.889	0.698
MW-GP at $q = 100\%$	0.866	0.676
MW-GP at $q = 90\%$	0.838	0.657
MW-GP at $q = 80\%$	0.821	0.643

Table 2: Performance comparison on the ML-1M dataset.

	RMSE	MAE
CF-NADE [Zheng et al., 2016]	0.771	-
GPLVM [Lawrence and Urtasun, 2009]	0.874	0.635
Sparse FC[Muller et al., 2018]	0.769	-
GC [Berg et al., 2017]	0.777	-
CWOCF [Lu et al., 2013]	0.903	0.701
LLORMA [Lee et al., 2013]	0.822	-
SMA [Li et al., 2016]	**0.768**	-
Biased MF	0.797	**0.612**
SVD++	0.826	0.636
MW-GP at $q = 100\%$	0.808	0.618
MW-GP at $q = 90\%$	0.774	0.594
MW-GP at $q = 80\%$	0.753	0.579

Table 3: Performance comparison on the ML-10M dataset.

ML-10M and Jester datasets can be found in Tab. 2, 3 and 4, respectively. In each table, We have two groups of evaluation references. First, we refer to the leading results that are reported by other works on the same dataset, but not necessarily with the same cross validation split. When applicable, we include the reported ranks of these methods. Second, we conduct new experiments with Biased MF, SVD++ and Bayesian GPLVM [Titsias and Lawrence, 2010] using exactly the same cross validation setting and the same size of latent vectors. We show the average performance of our model on a varying standard deviation quantile between 80% and 100%, and highlight (in shade) the values between which the best reported performances fall.

We also demonstrate performances of our method using the QP-plot in Fig. 1, 2 and 3. We visualize the average performance of all 5 splits with a solid line and the standard deviation across the 5 splits as error-bars. On the plots regarding the ML-1M and ML-10M datasets, we plot as dashed horizontal line the best performance reported in literature, which cut our curve between the quantiles of 80% and 90% in both of ML-1M in terms of MAE and RMSE. In the

	RMSE	MAE
[Desrosiers and Karypis, 2010]	4.480	3.541
[Goldberg et al., 2001]	-	3.740
Biased MF	5.826	4.300
SVD++	4.253	3.117
MW-GP at 100%	**4.171**	**3.092**
MW-GP at 90%	4.034	3.019
MW-GP at 80%	3.916	2.944

Table 4: Performance comparison on the Jester dataset.

case of ML-10M, the best reported RMSE lies between our quantile 90% and 100%, the best reported MAE between 90% and 100%. In other words, for those 70% to 90% of test samples for which our model is most confident, it also produces equal or better point estimates than state-of-the-art recommender systems. It is also worth noting that our model outperforms the only uncertainty-aware model, GPLVM, in all quantiles of variances, presumably due to the explicit modeling of output dependency.

Our training time on the ml-1m dataset is 13.4 min (1 CPU, 1 GPU) for all 500 epochs. For the same setting, the biased MF model requires 16.1 min and SVD++ 290.8 min (1 CPU). The uncertainty-aware Bayesian GPLVM has a substantially longer training time of more than 10 hours (16 CPUs, wall time). We perform inference for all tuples in the test set simultaneously. In case of ml-1m, for a test set of size 200K, the average inference time is 0.506 seconds. This demonstrates the computational efficiency of our solution to couple the inducing points (section 4.4). We also realize that, since the kernel evaluation is independent for different pairs of vectors, our model can profit from the parallelization power of GPUs.

6 CONCLUSION AND DISCUSSION

In this manuscript, we propose a novel uncertainty-aware approach to perform collaborative filtering in the context of recommender systems. To this end, we introduce a coregionalization kernel to perform multi-output Gaussian Process regression for unsupervised matrix decomposition in the spirit of GPLVM. In contrast to the GPLVM we do not treat output dimensions as independent, but model a coregionalization matrix with a kernel on latent variables representing the output. Our approach not only accounts for dependencies between output dimensions, but also allows our model to handle extremely sparse data in form of triple-stores in an efficient manner.

Finally, we would like to stress that the aim of our method is not to improve prediction quality that can be measured by metrics such as RMSE or MAE. Instead, we are interested in exploring the possible solutions to add uncertainty awareness to collaborative filtering approaches and recommender
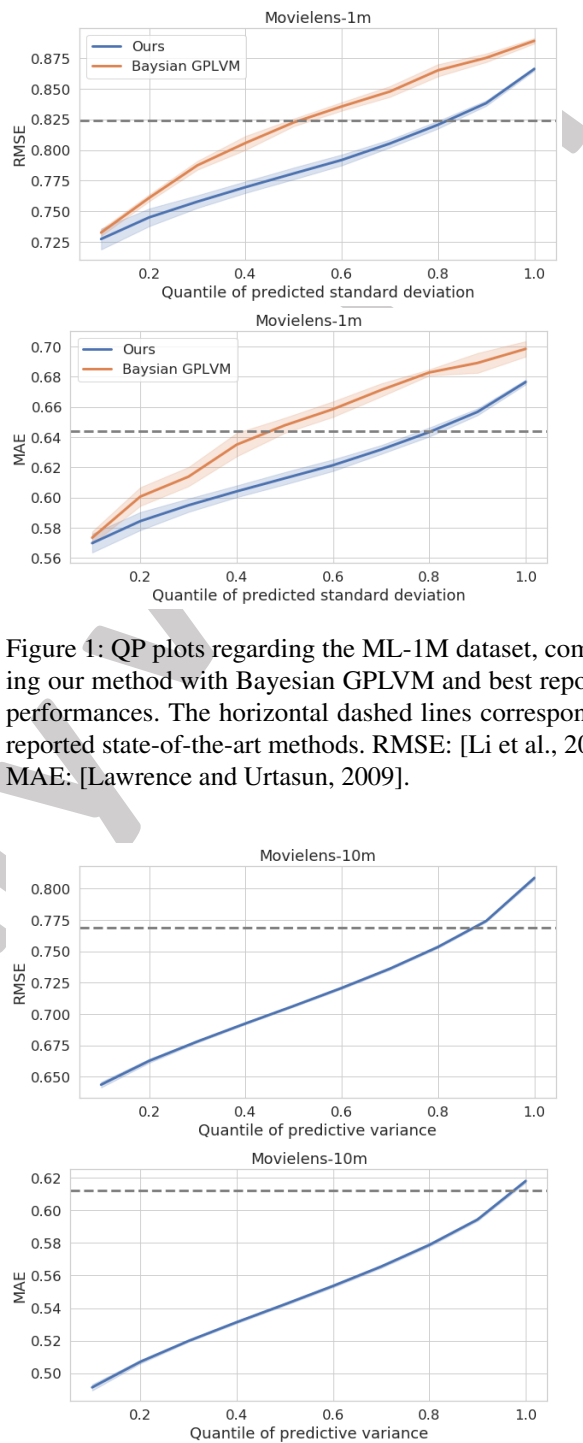

Figure 1: QP plots regarding the ML-1M dataset, comparing our method with Bayesian GPLVM and best reported performances. The horizontal dashed lines correspond to reported state-of-the-art methods. RMSE: [Li et al., 2016], MAE: [Lawrence and Urtasun, 2009].

Figure 2: QP plots regarding the ML-10M dataset, comparing our method with Bayesian GPLVM and best reported performances. The horizontal dashed lines correspond to reported state-of-the-art methods. RMSE: [Zheng et al., 2016]. In terms of MAE, the Biased MF outperforms our method only at quantile 100%. The GPLVM seems to suffer from the data volume and fails to converge in training.
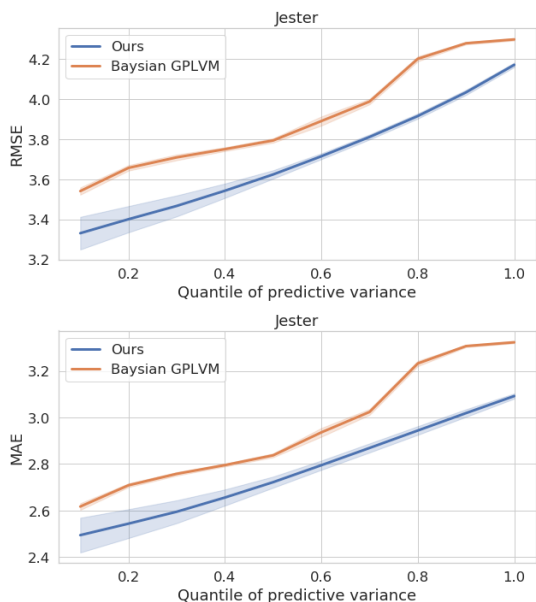
Figure 3: QP plots regarding the Jester dataset, comparing our method with Bayesian GPLVM. We do not find reported evaluations that outperform ours.

systems. More specifically, we enable these methods to express its uncertainty about every prediction. With the help of our evaluation method, the QP-plot, we can show that such uncertainty quantification are valid: for predictions that deviate from the ground truth, the corresponding predictive variances tend to be larger. With our experiments, we also show that Bayesian GPLVM fails to scale to large datasets in terms of both training and inference. Our approach in contrast merely requires training and inference time that is comparable to highly optimized linear models such as SVD++.

7 ACKNOWLEDGMENT

References

Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Rianne van den Berg, Thomas N Kipf, and Max Welling.

Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Thang D Bui, Josiah Yan, and Richard E Turner. A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *The Journal of Machine Learning Research*, 18(1):3649–3720, 2017.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

Zhenwen Dai, Mauricio Alvarez, and Neil Lawrence. Efficient modeling of latent information in supervised learning using gaussian processes. In *Advances in Neural Information Processing Systems*, pages 5131–5139, 2017.

Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Christian Desrosiers and George Karypis. A novel approach to compute similarities and its application to item recommendation. In *Pacific Rim International Conference on Artificial Intelligence*, pages 39–51. Springer, 2010.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151, 2001.

Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81(2009):1–10, 2009.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*, pages 329–336, 2004.

Neil D Lawrence. Learning for larger datasets with the gaussian process latent variable model. In *Artificial Intelligence and Statistics*, pages 243–250, 2007.

Neil D Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th annual international conference on machine learning*, pages 601–608. ACM, 2009.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In *International conference on machine learning*, pages 82–90, 2013.

Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen Chu. Low-rank matrix approximation with stability. In *International Conference on Machine Learning*, pages 295–303, 2016.

Jing Lu, Steven Hoi, and Jialei Wang. Second order online collaborative filtering. In *Asian Conference on Machine Learning*, pages 325–340, 2013.

Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10 (2):1273–1284, 2014.

Benjamin Marlin, Richard S Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. *arXiv preprint arXiv:1206.5267*, 2012.

Lorenz Muller, Julien Martel, and Giacomo Indiveri. Kernelized synaptic weight matrices. In *International Conference on Machine Learning*, pages 3651–3660, 2018.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104 (1):11–33, 2015.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6 (Dec):1939–1959, 2005.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352*, 2016.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

Michalis Titsias and Neil D Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 844–851, 2010.

Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218, 2009.

Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 549–553. SIAM, 2006.

Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477*, 2016.