Kernelized Bayesian Matrix Factorization

Mehmet Gönen and Samuel Kaski, *Senior Member, IEEE*

Abstract—We extend kernelized matrix factorization with a full-Bayesian treatment and with an ability to work with multiple side information sources expressed as different kernels. Kernels have been introduced to integrate side information about the rows and columns, which is necessary for making out-of-matrix predictions. We discuss specifically binary output matrices but extensions to real-valued matrices are straightforward. We extend the state of the art in two key aspects: (i) A full-conjugate probabilistic formulation of the kernelized matrix factorization enables an efficient variational approximation, whereas full-Bayesian treatments are not computationally feasible in the earlier approaches. (ii) Multiple side information sources are included, treated as different kernels in multiple kernel learning which additionally reveals which side sources are informative. We then show that the framework can also be used for supervised and semi-supervised multilabel classification and multi-output regression, by considering samples and outputs as the domains where matrix factorization operates. Our method outperforms alternatives in predicting drug-protein interactions on two data sets. On multilabel classification, our algorithm obtains the lowest Hamming losses on 10 out of 14 data sets compared to five state-of-the-art multilabel classification algorithms. We finally show that the proposed approach outperforms alternatives in multi-output regression experiments on a yeast cell cycle data set.

Index Terms—Automatic relevance determination, biological interaction networks, large margin learning, matrix factorization, multilabel classification, multiple kernel learning, multiple output regression, variational approximation

1 Introduction

Matrix factorization algorithms are very popular matrix completion methods [1], successfully used in many applications including recommender systems and image inpainting. The main idea behind these methods is to factorize a partially observed data matrix by finding a low-dimensional latent representation for both its rows and columns. The prediction for a missing entry can be calculated as the inner product between the latent representations of the corresponding row and column. Salakhutdinov and Mnih [2], [3] give a probabilistic formulation for matrix factorization and its fully Bayesian extension. However, these approaches are still incomplete in two major aspects: (i) It is not possible to integrate side information (e.g., social network or user profiles for recommender systems) into the model. (ii) It is not possible to make predictions for completely empty columns or rows (i.e., out-of-matrix prediction).

Algorithms for integrating side information into matrix factorization have been proposed earlier in the recommender systems literature. Ma et al. [4] propose a probabilistic matrix factorization method that uses a social network and the rating matrix together to find better latent components. Agarwal and Chen [5] formulate the latent components as the outputs of regression models on row and column features (i.e., side information sources). Shan and Banerjee [6] integrate side information into a probabilistic matrix factorization model using topic models to generate latent components of the rated items (e.g., movies). Agarwal and Chen [7] use a similar strategy to generate latent components of both users and items using topic models. Wang and Blei [8] also combine matrix factorization and topic models for scientific article recommendation using textual content of articles as side information. Yoo and Choi [9] factorize the rating and side information matrices jointly with a Bayesian formulation by sharing a factor matrix between these two. Park et al. [10] formulate a hierarchical Bayesian model for matrix factorization and use side information in prior distributions of latent components.

Miller et al. [11] formulate a nonparametric Bayesian method for modeling relational data such as social networks, which uses inferred latent features and input features to perform link prediction. Menon and Elkan [12] propose to solve this link prediction problem in graphs using a matrix factorization approach that can make use of explicit node or edge features.

All these algorithms are based on explicit feature representations; some are specific to count (e.g., text) data, and all model linear dependencies. Zhang et al. [13] generalize their earlier solutions to the nonlinear domain using, for example, decision trees instead of linear regression. However, their solution still requires explicit feature representations for rows and columns. We use kernels to be able to include nonlinear dependencies and to go beyond feature representations to structured objects such as sequences, trees, and graphs, considered especially in bioinformatics applications.

As in other branches of machine learning, kernel methods are also investigated for relational learning, factor analysis models, and matrix factorization. Chu et al. [14] give a nonparametric Bayesian method to integrate the relational information and explicit features of entities being modeled

• M. Gönen is with Sage Bionetworks, Seatle, WA 98109. E-mail: mehmet.gonen@sagebase.org.
• S. Kaski is with the Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University, 00076 Espoo, Finland and the Department of Computer Science, University of Helsinki, 00014 Helsinki, Finland. E-mail: samuel.kaski@aalto.fi.

Manuscript received 6 May 2013; revised 28 Jan. 2014; accepted 16 Mar. 2014. Date of publication 20 Mar. 2014; date of current version 10 Sept. 2014. Recommended for acceptance by A.J. Storkey.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2014.2313125

using Gaussian processes. Luttinen and Ilin [15] present a Bayesian factor analysis model, which formulates both the loading and factor matrices using Gaussian process priors.

Schmidt and Laurberg [16] extend nonnegative matrix factorization using Gaussian process priors on latent components. The priors enabled integrating side information about the objects into the model. Schmidt [17] further generalizes the idea to factorization of functions for nonlinear regression. Lawrence and Urtasun [18] formulate a nonlinear matrix factorization method by generating latent components via Gaussian processes but the results do not improve when side information is integrated. Adams et al. [19] couple multiple matrix factorization models with Gaussian process priors by replacing latent components with functions that operate on explicit feature representation of the objects. Recently, Zhou et al. [20] propose a kernelized probabilistic matrix factorization method using Gaussian process priors with covariance matrices on side information. However, with the modeling assumptions, only a *maximum a posteriori* (MAP) estimate for the latent components is computationally feasible, and even then the used gradient descent approach can be very slow. Furthermore, the method uses only a single kernel for each domain and needs test instances while training to be able to calculate their latent components (i.e., *transductive learning*).

All these kernel-based algorithms are able to model nonlinear dependencies with the help of kernel functions. However, they still have two limitations: (i) inability of performing out-of-matrix predictions in an inductive learning setting and (ii) inability of combining multiple side information sources about the objects in a principled way. We use a parametric model to eliminate these two restrictions.

In this paper, we introduce a kernelized Bayesian matrix factorization method and, in discussing its details, focus on the *bipartite graph inference* [21] scenario. The method can be applied to other types of matrices besides the bipartite graphs with slight modifications. The goal in the graph inference scenario is to model interaction networks between two domains (e.g., biological networks between drugs and proteins) and to estimate unknown interactions between objects from these two domains. The standard pairwise kernel approaches for this problem are based on a kernel matrix over object pairs in the training set and are computationally expensive [22]. There are also kernel-based (non-Bayesian) dimensionality reduction algorithms that map objects from both domains into a common subspace and perform prediction there [21], [23], [24].

In biological interaction networks, being composed of structured objects such as drugs and proteins, there exist several feature representations or similarity measures for the objects [25]. Instead of using a single specific kernel, we can combine multiple kernel functions in a principled way to obtain a better similarity measure, which is known as *multiple kernel learning* [26].

Our two main contributions are: (i) We formulate a novel fully conjugate probabilistic model that allows us to develop an efficient variational approximation scheme, the first fully Bayesian treatment which is still significantly faster than the earlier method for computing MAP point estimates [20]. (ii) The proposed method is able to integrate multiple side information sources by coupling matrix factorization with multiple kernel learning. We show the effectiveness on one toy data set and two drug-protein interaction data sets. We then show how the method can be used to solve supervised and semi-supervised multilabel classification problems and report classification results on 14 benchmark data sets. We also perform successful multiple output regression experiments on a yeast cell cycle data set.

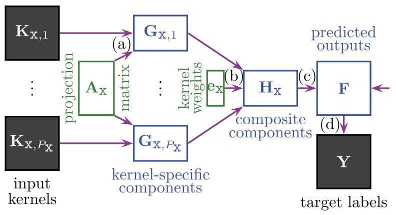
$\mathbf{K}_{\mathrm{x},1}$
$\mathbf{K}_{\mathrm{x},P_{\mathrm{x}}}$
input kernels
projection
$\mathbf{A}_{\mathrm{x}}$
matrix
(a)
$\mathbf{G}_{\mathrm{x},1}$
$\mathbf{G}_{\mathrm{x},P_{\mathrm{x}}}$
kernel-specific components
kernel weights
$\mathbf{e}_{\mathrm{x}}$
(b)
$\mathbf{H}_{\mathrm{x}}$
composite components
(c)
predicted outputs
$\mathbf{F}$
(d)
$\mathbf{Y}$
target labels

Fig. 1. Flowchart of kernelized matrix factorization with twin multiple kernel learning (the twin domain $\mathcal{Z}$ omitted for clarity).

This paper extends our preliminary conference paper [27] to (i) a detailed discussion of the method, and explicates the hyper-parameter learning, large margin learning connection, and initialization strategy. We additionally extend the method (ii) to perform Bayesian model selection using *automatic relevance determination* (ARD) and (iii) to handle partially observed outputs. (iv) Finally, the comprehensive battery of experiments includes automatic model selection, semi-supervised classification on multilabel data sets, and multiple output regression on yeast cell cycle data set.

2 Preliminaries and Notation

The objects come from two domains $\mathcal{X}$ and $\mathcal{Z}$. We are given two samples of independent and identically distributed training instances from each, denoted by $\mathbf{X} = \{\boldsymbol{x}_i \in \mathcal{X}\}_{i=1}^{N_{\mathrm{x}}}$ and $\mathbf{Z} = \{z_j \in \mathcal{Z}\}_{j=1}^{N_{\mathrm{z}}}$. For calculating similarities, we have multiple kernel functions for each domain, namely, $\{k_{\mathrm{x},m} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}\}_{m=1}^{P_{\mathrm{x}}}$ and $\{k_{\mathrm{z},n} : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}\}_{n=1}^{P_{\mathrm{z}}}$. If the side information comes in the form of features instead of similarities, the set of kernels will correspond to different notions of similarity on the same feature representation or may be using information coming from multiple feature representations (i.e., "views").

The (i, j)th entry of the target label matrix $\mathbf{Y} \in \{-1, +1\}^{N_{\mathrm{x}} \times N_{\mathrm{z}}}$ is

$$y_j^i = \begin{cases} +1, & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{z}_j \text{ have an interaction}, \\ -1, & \text{otherwise}. \end{cases}$$

The superscript indexes the rows and the subscript the columns. The prediction task is to estimate unknown interactions for out-of-matrix objects, which is also known as *cold start* prediction in recommender systems.

Fig. 1 illustrates the method we propose; it is composed of four main parts: (a) kernel-based nonlinear dimensionality reduction, (b) multiple kernel learning, (c) matrix factorization, and (d) binary classification. The first two kernel-based parts are applied to each domain separately, and they are completely symmetric, hence we call them *twins*. One of the twins (i.e., the one that operates on domain $\mathcal{Z}$) is omitted for clarity. In this section, we briefly explain each part and introduce the notation. In the following sections, we formulate a fully conjugate probabilistic model and derive a variational approximation for inference.

Kernel-based nonlinear dimensionality reduction. In this part, we perform feature extraction using the input kernel matrices $\{\mathbf{K}_{\text{x},m} \in \mathbb{R}^{N_\text{x} \times N_\text{x}}\}_{m=1}^{P_\text{x}}$ and the common projection matrix $\mathbf{A}_\text{x} \in \mathbb{R}^{N_\text{x} \times R}$ where R is the resulting subspace dimensionality. We obtain the kernel-specific components $\{\mathbf{G}_{\text{x},m} = \mathbf{A}_\text{x}^\top \mathbf{K}_{\text{x},m}\}_{m=1}^{P_\text{x}}$ after the projection. The main idea is very similar to *kernel principal component analysis* or *kernel Fisher discriminant analysis*, where the columns of the projection matrix can be solved with eigendecompositions [28]. However, that solution strategy is not possible for the more complex model formulated here.

Having a shared projection matrix across the kernels has two main implications: (i) The number of model parameters is much lower than if we had a separate projection matrix for each kernel, leading to more regularization. (ii) We can combine the kernels with multiple kernel learning as explained in the following.

Multiple kernel learning. This part is responsible for combining the kernel-specific (i.e., view-specific) components linearly to obtain the composite components $\mathbf{H}_\text{x} = \sum_{m=1}^{P_\text{x}} e_{\text{x},m} \mathbf{G}_{\text{x},m}$ where the kernel weights can take arbitrary values $\boldsymbol{e}_\text{x} \in \mathbb{R}^{P_\text{x}}$. The multiple kernel learning property of our formulation can easily be seen using the following equivalence:

$$\sum_{m=1}^{P_\text{x}} e_{\text{x},m} \underbrace{(\mathbf{A}_\text{x}^\top \mathbf{K}_{\text{x},m})}_{\mathbf{G}_{\text{x},m}} = \mathbf{A}_\text{x}^\top \underbrace{\left(\sum_{m=1}^{P_\text{x}} e_{\text{x},m} \mathbf{K}_{\text{x},m} \right)}_{\text{combined kernel}},$$

where we need to have a shared projection matrix to obtain a valid linear combination of the kernels.

If we have a single kernel function for a specific domain, we can safely ignore the composite components and the kernel weights, and use the single available kernel-specific components to represent the objects in that domain [29]. The details of our method with a single kernel function for each domain are explained in Section 5.

Matrix factorization. In this part, we propose to use the low-dimensional representations of objects in the unified subspace, namely, $\mathbf{H}_\text{x}$ and $\mathbf{H}_\text{z}$, to calculate the latent output matrix $\mathbf{F} = \mathbf{H}_\text{x}^\top \mathbf{H}_\text{z}$. This corresponds to factorizing the latent outputs into two low-rank matrices.

Binary classification. This part just assigns a class label to each object pair $(\boldsymbol{x}_i, \boldsymbol{z}_j)$ by looking at the sign of the latent output f_j^i in the matrix factorization part. The proposed method can also be extended to handle other types of outputs (e.g., real-valued outputs used in recommender systems) by removing the binary classification part and directly generating the target outputs in the matrix factorization part. This corresponds to removing the latent output matrix $\mathbf{F}$ and generating target label matrix $\mathbf{Y}$ directly from the composite components $\mathbf{H}_\text{x}$ and $\mathbf{H}_\text{z}$. The details of our method for real-valued outputs are also given in Section 5.

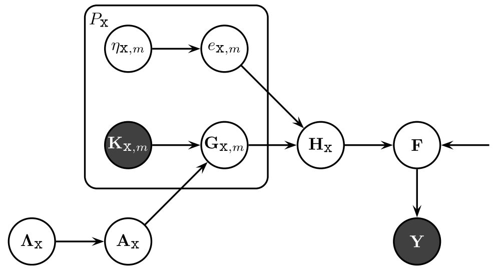

Fig. 2. Graphical model of kernelized Bayesian matrix factorization with twin multiple kernel learning.

3 Kernelized Bayesian Matrix Factorization with Twin Multiple Kernel Learning

For the method described in the previous section, we formulate a probabilistic model, called *kernelized Bayesian matrix factorization with twin multiple kernel learning* (KBMF2MKL), which has two key properties that enable us to perform efficient inference: (i) The kernel-specific and composite components are modeled explicitly by introducing them as latent variables. (ii) Kernel weights are assumed to be normally distributed without enforcing any constraints (e.g., non-negativity) on them. The reasons for introducing these two properties become clear when we explain the inference.

Fig. 2 gives the graphical model of KBMF2MKL with latent variables and their corresponding priors. There are some additions to the notation described earlier: The $N_\text{x} \times R$ matrix of priors for the entries of the projection matrix $\mathbf{A}_\text{x}$ is denoted by Λ_x. The $P_\text{x} \times 1$ vector of priors for the kernel weights $\boldsymbol{e}_\text{x}$ is denoted by $\boldsymbol{\eta}_\text{x}$. The standard deviations for the kernel-specific and composite components are σ_g and σ_h, respectively; these hyper-parameters are not shown for clarity.

The distributional assumptions of the dimensionality reduction part are

$$\begin{aligned} \lambda_{\text{x},s}^i &\sim \mathcal{G}(\lambda_{\text{x},s}^i; \alpha_\lambda, \beta_\lambda) && \forall (i,s) \\ a_{\text{x},s}^i | \lambda_{\text{x},s}^i &\sim \mathcal{N}\big(a_{\text{x},s}^i; 0, (\lambda_{\text{x},s}^i)^{-1}\big) && \forall (i,s) \\ g_{\text{x},m,i}^s | \boldsymbol{a}_{\text{x},s}, \boldsymbol{k}_{\text{x},m,i} &\sim \mathcal{N}\big(g_{\text{x},m,i}^s; \boldsymbol{a}_{\text{x},s}^\top \boldsymbol{k}_{\text{x},m,i}, \sigma_g^2\big) && \forall (m,s,i), \end{aligned}$$

where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \Sigma)$ is the normal distribution with mean vector $\boldsymbol{\mu}$ (here scalar) and covariance matrix Σ (here scalar), and $\mathcal{G}(\cdot; \alpha, \beta)$ denotes the gamma distribution with shape parameter α and scale parameter β. The multiple kernel learning part has the following distributional assumptions:

$$\begin{aligned} \eta_{\text{x},m} &\sim \mathcal{G}(\eta_{\text{x},m}; \alpha_\eta, \beta_\eta) && \forall m \\ e_{\text{x},m} | \eta_{\text{x},m} &\sim \mathcal{N}\big(e_{\text{x},m}; 0, \eta_{\text{x},m}^{-1}\big) && \forall m \\ h_{\text{x},i}^s | \{e_{\text{x},m}, g_{\text{x},m,i}^s\}_{m=1}^{P_\text{x}} &\sim \mathcal{N}\left(h_{\text{x},i}^s; \sum_{m=1}^{P_\text{x}} e_{\text{x},m} g_{\text{x},m,i}^s, \sigma_h^2\right) && \forall (s,i), \end{aligned}$$

where kernel-level sparsity can be tuned by changing the hyper-parameters $(\alpha_\eta, \beta_\eta)$. Setting the gamma priors to induce sparsity, e.g., $(\alpha_\eta, \beta_\eta) = (0.001, 1000)$, produces results analogous to using the ℓ_1-norm on the kernel weights, whereas using uninformative priors, e.g., $(\alpha_\eta, \beta_\eta) = (1, 1)$, resembles using the ℓ_2-norm. The matrix factorization part calculates the latent outputs using the inner products between the low-dimensional representations of the object pairs:

$$f_j^i | \boldsymbol{h}_{\mathrm{x},i}, \boldsymbol{h}_{\mathrm{z},j} \sim \mathcal{N}\big(f_j^i; \boldsymbol{h}_{\mathrm{x},i}^\top \boldsymbol{h}_{\mathrm{z},j}, 1\big) \;\; \forall (i,j),$$

where the latent outputs are introduced to speed up the inference procedures [30]. Finally, a binary classification is produced:

$$y_j^i | f_j^i \sim \delta\big(f_j^i y_j^i > \nu\big) \;\; \forall (i,j),$$

where the margin parameter ν is introduced to remove ambiguity in the scaling and to place a low-density region between the two classes, similarly to the margin idea in SVMs, which is generally used for semi-supervised learning [31]. Here, $\delta(\cdot)$ is the Kronecker delta function that returns 1 if its argument is true and 0 otherwise.

4 Efficient Inference Using Variational Approximation

Exact inference for the model is intractable and of the two readily available alternatives, Gibbs sampling and variational approximation, we choose the latter for computational efficiency. Variational methods optimize a lower bound on the marginal likelihood, which involves a factorized approximation of the posterior, to find the joint parameter distribution [32].

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda, \sigma_g, \sigma_h, \nu\}$, all prior variables by $\Xi = \{\boldsymbol{\eta}_\mathrm{x}, \boldsymbol{\eta}_\mathrm{z}, \Lambda_\mathrm{x}, \Lambda_\mathrm{z}\}$, and the remaining random variables by $\Theta = \{\mathbf{A}_\mathrm{x}, \mathbf{A}_\mathrm{z}, \boldsymbol{e}_\mathrm{x}, \boldsymbol{e}_\mathrm{z}, \mathbf{F}, \{\mathbf{G}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\mathbf{G}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}, \mathbf{H}_\mathrm{x}, \mathbf{H}_\mathrm{z}\}$. We omit the dependence on $\boldsymbol{\zeta}$ for clarity. We factorize the variational approximation as

$$\begin{aligned} p(\Theta, \Xi | \{\mathbf{K}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\mathbf{K}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}, \mathbf{Y}) &\approx q(\Theta, \Xi) \\ &= q(\Lambda_\mathrm{x}) q(\mathbf{A}_\mathrm{x}) q(\{\mathbf{G}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}) q(\boldsymbol{\eta}_\mathrm{x}) q(\boldsymbol{e}_\mathrm{x}) q(\mathbf{H}_\mathrm{x}) \\ &\quad q(\Lambda_\mathrm{z}) q(\mathbf{A}_\mathrm{z}) q(\{\mathbf{G}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}) q(\boldsymbol{\eta}_\mathrm{z}) q(\boldsymbol{e}_\mathrm{z}) q(\mathbf{H}_\mathrm{z}) q(\mathbf{F}) \end{aligned}$$

and define each factor according to its full conditional:

$$\begin{aligned} q(\Lambda_\mathrm{x}) &= \prod_{i=1}^{N_\mathrm{x}} \prod_{s=1}^{R} \mathcal{G}\big(\lambda_{\mathrm{x},s}^i; \alpha(\lambda_{\mathrm{x},s}^i), \beta(\lambda_{\mathrm{x},s}^i)\big) \\ q(\mathbf{A}_\mathrm{x}) &= \prod_{s=1}^{R} \mathcal{N}(\boldsymbol{a}_{\mathrm{x},s}; \mu(\boldsymbol{a}_{\mathrm{x},s}), \Sigma(\boldsymbol{a}_{\mathrm{x},s})) \\ q(\mathbf{G}_{\mathrm{x},m}) &= \prod_{i=1}^{N_\mathrm{x}} \mathcal{N}(\boldsymbol{g}_{\mathrm{x},m,i}; \mu(\boldsymbol{g}_{\mathrm{x},m,i}), \Sigma(\boldsymbol{g}_{\mathrm{x},m,i})) \;\; \forall m \\ q(\boldsymbol{e}_\mathrm{x}) &= \mathcal{N}(\boldsymbol{e}_\mathrm{x}; \mu(\boldsymbol{e}_\mathrm{x}), \Sigma(\boldsymbol{e}_\mathrm{x})) \\ q(\boldsymbol{\eta}_\mathrm{x}) &= \prod_{m=1}^{P_\mathrm{x}} \mathcal{G}(\eta_{\mathrm{x},m}; \alpha(\eta_{\mathrm{x},m}), \beta(\eta_{\mathrm{x},m})) \\ q(\mathbf{H}_\mathrm{x}) &= \prod_{i=1}^{N_\mathrm{x}} \mathcal{N}(\boldsymbol{h}_{\mathrm{x},i}; \mu(\boldsymbol{h}_{\mathrm{x},i}), \Sigma(\boldsymbol{h}_{\mathrm{x},i})) \\ q(\mathbf{F}) &= \prod_{i=1}^{N_\mathrm{x}} \prod_{j=1}^{N_\mathrm{z}} \mathcal{TN}\big(f_j^i; \mu(f_j^i), \Sigma(f_j^i), \rho(f_j^i)\big), \end{aligned}$$

where $\alpha(\cdot)$, $\beta(\cdot)$, $\mu(\cdot)$, and $\Sigma(\cdot)$ denote the shape parameter, scale parameter, mean vector, and covariance matrix, respectively. Here, $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \Sigma, \rho(\cdot))$ denotes the truncated normal distribution with mean vector $\boldsymbol{\mu}$, covariance matrix Σ, and truncation rule $\rho(\cdot)$ such that $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \Sigma, \rho(\cdot)) \propto \mathcal{N}(\cdot; \boldsymbol{\mu}, \Sigma)$ if $\rho(\cdot)$ is true and $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \Sigma, \rho(\cdot)) = 0$ otherwise.

We can bound the marginal likelihood using Jensen's inequality:

$$\begin{aligned} \log p\big(\mathbf{Y} | \{\mathbf{K}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\mathbf{K}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}\big) \\ \geq \mathrm{E}_{q(\Theta,\Xi)}\big[\log p\big(\mathbf{Y}, \Theta, \Xi | \{\mathbf{K}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\mathbf{K}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}\big)\big] \\ - \mathrm{E}_{q(\Theta,\Xi)}[\log q(\Theta, \Xi)] \end{aligned} \tag{1}$$

and optimize this bound by maximizing with respect to each factor separately until convergence. The approximate posterior distribution of a specific factor $\boldsymbol{\tau}$ can be found as

$$q(\boldsymbol{\tau}) \propto \exp\big(\mathrm{E}_{q(\{\Theta,\Xi\} \setminus \boldsymbol{\tau})}\big[\log p\big(\mathbf{Y}, \Theta, \Xi | \{\mathbf{K}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\mathbf{K}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}\big)\big]\big).$$

For our model, thanks to the conjugacy, the resulting approximate posterior distribution of each factor follows the same distribution as the corresponding factor.

The approximate posterior distribution parameters of the dimensionality reduction part can be found as

$$\begin{aligned} \alpha(\lambda_{\mathrm{x},s}^i) &= \alpha_\lambda + 1/2 \\ \beta(\lambda_{\mathrm{x},s}^i) &= \Big(1/\beta_\lambda + \widetilde{(a_{\mathrm{x},s}^i)^2}/2\Big)^{-1} \\ \Sigma(\boldsymbol{a}_{\mathrm{x},s}) &= \Bigg(\mathrm{diag}\big(\widetilde{\boldsymbol{\lambda}_\mathrm{x}^s}\big) + \sum_{m=1}^{P_\mathrm{x}} \sigma_g^{-2} \mathbf{K}_{\mathrm{x},m} \mathbf{K}_{\mathrm{x},m}^\top\Bigg)^{-1} \\ \mu(\boldsymbol{a}_{\mathrm{x},s}) &= \Sigma(\boldsymbol{a}_{\mathrm{x},s}) \sum_{m=1}^{P_\mathrm{x}} \sigma_g^{-2} \mathbf{K}_{\mathrm{x},m} \big(\widetilde{\boldsymbol{g}_{\mathrm{x},m}^s}\big)^\top, \end{aligned} \tag{2}$$

where the tilde notation denotes the posterior expectations as usual, i.e., $\widetilde{f(\boldsymbol{\tau})} = \mathrm{E}_{q(\boldsymbol{\tau})}[f(\boldsymbol{\tau})]$.

The kernel-specific components have the following approximate posterior distribution parameters:

$$\begin{aligned} \Sigma(\boldsymbol{g}_{\mathrm{x},m,i}) &= \Big(\sigma_g^{-2}\mathbf{I} + \sigma_h^{-2} \widetilde{e_{\mathrm{x},m}^2} \mathbf{I}\Big)^{-1} \\ \mu(\boldsymbol{g}_{\mathrm{x},m,i}) &= \Sigma(\boldsymbol{g}_{\mathrm{x},m,i}) \Big(\sigma_g^{-2} \widetilde{\mathbf{A}_\mathrm{x}^\top} \boldsymbol{k}_{\mathrm{x},m,i} + \sigma_h^{-2} \widetilde{e_{\mathrm{x},m}} \widetilde{\boldsymbol{h}_{\mathrm{x},i}} \\ &\quad - \sum_{o \neq m} \sigma_h^{-2} \widetilde{e_{\mathrm{x},m} e_{\mathrm{x},o}} \widetilde{\boldsymbol{g}_{\mathrm{x},o,i}}\Big), \end{aligned} \tag{3}$$

where the mean and covariance parameters are affected by the kernel weights, the composite components, and other kernel-specific components in addition to the projection matrix and the corresponding kernel matrix.

The approximate posterior distribution parameters of the multiple kernel learning part can be found as

$$\begin{aligned} \alpha(\eta_{\mathrm{x},m}) &= \alpha_\eta + 1/2 \\ \beta(\eta_{\mathrm{x},m}) &= \Big(1/\beta_\eta + \widetilde{e_{\mathrm{x},m}^2}/2\Big)^{-1} \\ \Sigma(\boldsymbol{e}_\mathrm{x}) &= \Bigg(\mathrm{diag}(\widetilde{\boldsymbol{\eta}_\mathrm{x}}) + \Big[\sigma_h^{-2} \, \mathrm{tr}\Big(\widetilde{\mathbf{G}_{\mathrm{x},m}^\top \mathbf{G}_{\mathrm{x},o}}\Big)\Big]_{m=1,o=1}^{P_\mathrm{x},P_\mathrm{x}}\Bigg)^{-1} \\ \mu(\boldsymbol{e}_\mathrm{x}) &= \Sigma(\boldsymbol{e}_\mathrm{x}) \Big[\sigma_h^{-2} \, \mathrm{tr}\Big(\widetilde{\mathbf{G}_{\mathrm{x},m}^\top} \widetilde{\mathbf{H}_\mathrm{x}}\Big)\Big]_{m=1}^{P_\mathrm{x}}, \end{aligned} \tag{4}$$

where the mean and covariance parameters of the kernel weights are calculated using the kernel-specific and composite components.

The composite components have the following approximate posterior distribution parameters:

$$\Sigma(\boldsymbol{h}_{\text{x},i}) = \left(\sigma_h^{-2}\mathbf{I} + \widetilde{\mathbf{H}_{\text{z}}\mathbf{H}_{\text{z}}^\top}\right)^{-1}$$
$$\mu(\boldsymbol{h}_{\text{x},i}) = \Sigma(\boldsymbol{h}_{\text{x},i})\left(\sum_{m=1}^{P_{\text{x}}} \sigma_h^{-2}\widetilde{e_{\text{x},m}}\widetilde{\boldsymbol{g}_{\text{x},m,i}} + \widetilde{\mathbf{H}_{\text{z}}}(\widetilde{\boldsymbol{f}^i})^\top\right), \quad (5)$$

where it can be seen that the inference transfers information between the two domains. Note that the composite components of each domain are the only random variables that have an effect on the other domain, i.e., only the $\mathbf{H}_{\text{z}}$ variables of domain $\mathcal{Z}$ are used when updating the random variables of the domain $\mathcal{X}$.

The approximate posterior distribution parameters of the latent outputs are:

$$\Sigma(f_j^i) = 1$$
$$\mu(f_j^i) = \widetilde{\boldsymbol{h}_{\text{x},i}^\top}\widetilde{\boldsymbol{h}_{\text{z},j}}$$
$$\rho(f_j^i) \triangleq f_j^i y_j^i > \nu,$$

where we need to find the posterior expectation of $\mathbf{F}$ to update the approximate posterior distributions of the composite components. Fortunately, the truncated normal distribution has a closed-form formula for its expectation.

Modeling choices. Note that using the kernel-specific and composite components as latent variables in our probabilistic model introduces extra conditional independencies between the random variables and enables us to derive efficient update rules for the approximate posterior distributions. The other key property of our model is the assumption of normality of the kernel weights, which allows us to obtain a fully conjugate probabilistic model [33]. In earlier Bayesian multiple kernel learning algorithms, the combined kernel has usually been defined as a convex sum of the input kernels, by assuming a Dirichlet distribution on the kernel weights [34], [35]. As a consequence of the nonconjugacy between Dirichlet and normal distributions, they need to use a sampling strategy (e.g., importance sampling) to update the kernel weights when deriving variational approximations.

Hyper-parameter learning. When formulating our method, we assume that the kernel functions and their hyper-parameters are fixed prior to inference. When we also want to tune the hyper-parameters (e.g., kernel width for the Gaussian kernel), we have two viable options in addition to the standard cross-validation strategy: (i) We can tune the hyper-parameters using a type-II maximum likelihood [36] or Markov chain Monte Carlo [37], [38] approach, which is still feasible even though the conjugacy would be lost. (ii) We can provide multiple copies of the same kernel with different hyper-parameter values and let the algorithm pick suitable ones with the help of multiple kernel learning part.

Large margin learning. Modeling the approximate posterior distribution of the latent output matrix $\mathbf{F}$ with the truncated normal distribution enables us to integrate the large margin property into our probabilistic model. Fig. 3 shows the effect of the truncated normal distribution for two

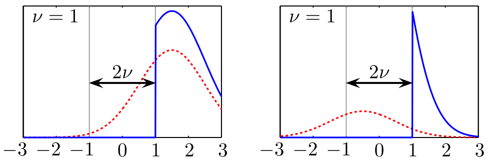

Fig. 3. Effect of the truncated normal distribution on target output values.

positively labeled object pairs. Their latent output values are assumed to have mean values +1.5 and −0.5 as can be seen from the dashed lines. Without truncation, they would be allowed to take negative values or to be within the margin, whereas after truncation, they are guaranteed to take positive values and to be outside the margin, as can be seen from the solid lines.

Initialization. Due to the large margin property introduced to the classification part, it is better to start the algorithm with parameters that satisfy the margin constraints on $\mathbf{F}$. We initialize the mean parameter to

$$\mu(f_j^i) = y_j^i\left(\left|u_j^i\right| + \nu\right),$$

where $|\cdot|$ denotes the absolute value and u_j^i is a standardized normal random variable. The mean parameters of all other normal random variables are initialized using the standardized normal distribution. The covariance parameters of all normal random variables are initialized to the identity matrix.

Convergence. The inference mechanism sequentially updates the approximate posterior distributions of the latent variables and the corresponding priors until convergence, which can be checked by monitoring the lower bound in (1). The first term of the lower bound corresponds to the sum of exponential-form expectations of the distributions in the joint likelihood. The second term is the sum of negative entropies of the approximate posteriors in the ensemble. The only nonstandard distribution in these terms is the truncated normal distribution used for the latent outputs, and the truncated normal distribution has a closed-form formula also for its entropy.

Computational complexity. The most time-consuming operations of the update equations are covariance calculations because they need matrix inversions. The time complexity of the covariance updates for the projection matrices in (2) is $\mathcal{O}(R\max(N_{\text{x}}^3, N_{\text{z}}^3))$, and we can cache $\sum_{m=1}^{P_{\text{x}}}\mathbf{K}_{\text{x},m}\mathbf{K}_{\text{x},m}^\top$ and $\sum_{n=1}^{P_{\text{z}}}\mathbf{K}_{\text{z},n}\mathbf{K}_{\text{z},n}^\top$ before starting the inference to reduce the computational cost. The covariance updates for the kernel-specific components in (3) require inverting diagonal matrices. The time complexity of the covariance updates for the kernel weights and the composite components in (4) and (5) is $\mathcal{O}(\max(P_{\text{x}}^3, P_{\text{z}}^3))$. The other calculations in these updates can be done efficiently using matrix-matrix and matrix-vector multiplications. Finding the posterior expectations of the latent outputs only requires evaluating the standardized normal cumulative distribution function and the standardized normal probability density. In summary, the total time complexity of each iteration in our variational approximation scheme is $\mathcal{O}(R\max(N_{\text{x}}^3, N_{\text{z}}^3) + \max(P_{\text{x}}^3, P_{\text{z}}^3))$, which makes the algorithm more efficient than standard pairwise kernel approaches [22] that require calculating a kernel matrix over pairs and training a kernel-based classifier using this kernel, resulting in $\mathcal{O}(N_{\text{x}}^3 N_{\text{z}}^3)$ complexity.

Prediction. Given an unseen test pair $(\boldsymbol{x}_\star, z_*)$, we need to predict the corresponding score $f_*^\star$ or target label $y_*^\star$. We first replace the posterior distributions of $\mathbf{A}_\mathrm{x}$, $\mathbf{A}_\mathrm{z}$, $\boldsymbol{e}_\mathrm{x}$, and $\boldsymbol{e}_\mathrm{z}$ with their approximate posterior distributions $q(\mathbf{A}_\mathrm{x})$, $q(\mathbf{A}_\mathrm{z})$, $q(\boldsymbol{e}_\mathrm{x})$, and $q(\boldsymbol{e}_\mathrm{z})$. Using the approximate distributions, we obtain the predictive distributions of the kernel-specific and composite components. The predictive distribution of the target label can finally be formulated as

$$\begin{aligned}&p\big(y_*^\star = +1 | \{\boldsymbol{k}_{\mathrm{x},m,\star}, \mathbf{K}_{\mathrm{x},m}\}_{m=1}^{P_\mathrm{x}}, \{\boldsymbol{k}_{\mathrm{z},n,*}, \mathbf{K}_{\mathrm{z},n}\}_{n=1}^{P_\mathrm{z}}, \mathbf{Y}\big)\\ &\quad = \big(\mathcal{Z}_*^\star\big)^{-1} \Phi\left(\frac{\mu(f_*^\star) - \nu}{\Sigma(f_*^\star)}\right)\end{aligned} \tag{6}$$

where $\mathcal{Z}_*^\star$ is the normalization coefficient calculated for the test pair and $\Phi(\cdot)$ is the standardized normal cumulative distribution function.

5 Model Variants

In this section, we provide details about three variants of our method. The variants (i) learn the subspace dimensionality using ARD, (ii) use a single kernel function on each domain instead of the multiple kernels, and (iii) predict real-valued outputs instead of the binary outputs.

Learning subspace dimensionality using automatic relevance determination. We optimize the dimensionality of the latent components using a Bayesian model selection procedure. In detail, the same set of precision priors is shared by the projection matrices, and the dimensionality is determined using ARD [39]. A similar strategy has previously been used to perform model selection in nonnegative matrix factorization models [40].

For the ARD, the precision matrices Λ_x and Λ_z are replaced with a common precision vector $\boldsymbol{\lambda}$. The distributional assumptions of the modified model are

$$\begin{aligned}\lambda_s &\sim \mathcal{G}(\lambda_s; \alpha_\lambda, \beta_\lambda) && \forall s\\ a_{\mathrm{x},s}^i | \lambda_s &\sim \mathcal{N}\big(a_{\mathrm{x},s}^i; 0, \lambda_s^{-1}\big) && \forall (i,s)\\ a_{\mathrm{z},s}^j | \lambda_s &\sim \mathcal{N}\big(a_{\mathrm{z},s}^j; 0, \lambda_s^{-1}\big) && \forall (j,s),\end{aligned}$$

where we can set the gamma priors accordingly, e.g., $(\alpha_\lambda, \beta_\lambda) = (0.001, 1000)$, to eliminate unnecessary columns of projection matrices, leading to automatic model selection.

We can use the inference algorithm given in Section 4 with slight modifications. First, we need to replace $q(\Lambda_\mathrm{x})$ and $q(\Lambda_\mathrm{z})$ with $q(\boldsymbol{\lambda})$. Then, the parameter update equations of the approximate posterior distributions become

$$\begin{aligned}\alpha(\lambda_s) &= \alpha_\lambda + N_\mathrm{x}/2 + N_\mathrm{z}/2\\ \beta(\lambda_s) &= \Big(1/\beta_\lambda + \widetilde{\boldsymbol{a}_{\mathrm{x},s}^\top \boldsymbol{a}_{\mathrm{x},s}}/2 + \widetilde{\boldsymbol{a}_{\mathrm{z},s}^\top \boldsymbol{a}_{\mathrm{z},s}}/2\Big)^{-1}\\ \Sigma(\boldsymbol{a}_{\mathrm{x},s}) &= \left(\widetilde{\lambda_s}\mathbf{I} + \sum_{m=1}^{P_\mathrm{x}} \sigma_g^{-2} \mathbf{K}_{\mathrm{x},m}\mathbf{K}_{\mathrm{x},m}^\top\right)^{-1}\\ \mu(\boldsymbol{a}_{\mathrm{x},s}) &= \Sigma(\boldsymbol{a}_{\mathrm{x},s}) \sum_{m=1}^{P_\mathrm{x}} \sigma_g^{-2}\mathbf{K}_{\mathrm{x},m}\big(\widetilde{\boldsymbol{g}_{\mathrm{x},m}^s}\big)^\top\\ \Sigma(\boldsymbol{a}_{\mathrm{z},s}) &= \left(\widetilde{\lambda_s}\mathbf{I} + \sum_{n=1}^{P_\mathrm{z}} \sigma_g^{-2} \mathbf{K}_{\mathrm{z},n}\mathbf{K}_{\mathrm{z},n}^\top\right)^{-1}\\ \mu(\boldsymbol{a}_{\mathrm{z},s}) &= \Sigma(\boldsymbol{a}_{\mathrm{z},s}) \sum_{n=1}^{P_\mathrm{z}} \sigma_g^{-2}\mathbf{K}_{\mathrm{z},n}\big(\widetilde{\boldsymbol{g}_{\mathrm{z},n}^s}\big)^\top.\end{aligned}$$

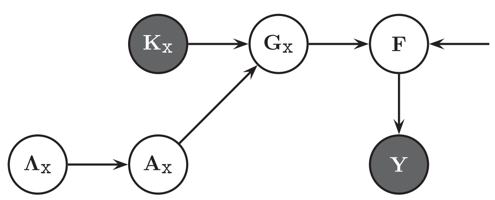

Fig. 4. Graphical model of kernelized Bayesian matrix factorization with twin kernels.

Kernelized Bayesian matrix factorization with twin kernels. We formulate a simplified probabilistic model, called *kernelized Bayesian matrix factorization with twin kernels* (KBMF2K), for the case with a single kernel function for each domain. Fig. 4 shows the graphical model of KBMF2K with latent variables and their corresponding priors.

The distributional assumptions of the simplified model are

$$\begin{aligned}\lambda_{\mathrm{x},s}^i &\sim \mathcal{G}\big(\lambda_{\mathrm{x},s}^i; \alpha_\lambda, \beta_\lambda\big) && \forall (i,s)\\ a_{\mathrm{x},s}^i | \lambda_{\mathrm{x},s}^i &\sim \mathcal{N}\big(a_{\mathrm{x},s}^i; 0, \big(\lambda_{\mathrm{x},s}^i\big)^{-1}\big) && \forall (i,s)\\ g_{\mathrm{x},i}^s | \boldsymbol{a}_{\mathrm{x},s}, \boldsymbol{k}_{\mathrm{x},i} &\sim \mathcal{N}\big(g_{\mathrm{x},i}^s; \boldsymbol{a}_{\mathrm{x},s}^\top \boldsymbol{k}_{\mathrm{x},i}, \sigma_g^2\big) && \forall (s,i)\\ f_j^i | \boldsymbol{g}_{\mathrm{x},i}, \boldsymbol{g}_{\mathrm{z},j} &\sim \mathcal{N}\big(f_j^i; \boldsymbol{g}_{\mathrm{x},i}^\top \boldsymbol{g}_{\mathrm{z},j}, 1\big) && \forall (i,j)\\ y_j^i | f_j^i &\sim \delta\big(y_j^i; f_j^i y_j^i > \nu\big) && \forall (i,j).\end{aligned}$$

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_\lambda, \beta_\lambda, \sigma_g, \nu\}$, all prior variables by $\Xi = \{\Lambda_\mathrm{x}, \Lambda_\mathrm{z}\}$, and the remaining random variables by $\Theta = \{\mathbf{A}_\mathrm{x}, \mathbf{A}_\mathrm{z}, \mathbf{F}, \mathbf{G}_\mathrm{x}, \mathbf{G}_\mathrm{z}\}$. We again omit the dependence on $\boldsymbol{\zeta}$ for clarity. We can write the factorized variational approximation as

$$\begin{aligned}p(\Theta, \Xi | \mathbf{K}_\mathrm{x}, \mathbf{K}_\mathrm{z}, \mathbf{Y}) &\approx q(\Theta, \Xi)\\ &= q(\Lambda_\mathrm{x})q(\mathbf{A}_\mathrm{x})q(\mathbf{G}_\mathrm{x})q(\Lambda_\mathrm{z})q(\mathbf{A}_\mathrm{z})q(\mathbf{G}_\mathrm{z})q(\mathbf{F})\end{aligned}$$

and define each factor in the ensemble just like its full conditional:

$$\begin{aligned}q(\Lambda_\mathrm{x}) &= \prod_{i=1}^{N_\mathrm{x}} \prod_{s=1}^{R} \mathcal{G}\big(\lambda_{\mathrm{x},s}^i; \alpha\big(\lambda_{\mathrm{x},s}^i\big), \beta\big(\lambda_{\mathrm{x},s}^i\big)\big)\\ q(\mathbf{A}_\mathrm{x}) &= \prod_{s=1}^{R} \mathcal{N}(\boldsymbol{a}_{\mathrm{x},s}; \mu(\boldsymbol{a}_{\mathrm{x},s}), \Sigma(\boldsymbol{a}_{\mathrm{x},s}))\\ q(\mathbf{G}_\mathrm{x}) &= \prod_{m=1}^{P_\mathrm{x}} \prod_{i=1}^{N_\mathrm{x}} \mathcal{N}(\boldsymbol{g}_{\mathrm{x},i}; \mu(\boldsymbol{g}_{\mathrm{x},i}), \Sigma(\boldsymbol{g}_{\mathrm{x},i}))\\ q(\mathbf{F}) &= \prod_{i=1}^{N_\mathrm{x}} \prod_{j=1}^{N_\mathrm{z}} \mathcal{TN}\big(f_j^i; \mu\big(f_j^i\big), \Sigma\big(f_j^i\big), \rho\big(f_j^i\big)\big).\end{aligned}$$

The approximate posterior distribution parameters of the ensemble can be found as

$$\begin{aligned}\alpha(\lambda_{\mathrm{x},s}^i) &= \alpha_\lambda + 1/2\\ \beta(\lambda_{\mathrm{x},s}^i) &= \big(1/\beta_\lambda + \widetilde{(a_{\mathrm{x},s}^i)^2}/2\big)^{-1}\\ \Sigma(\boldsymbol{a}_{\mathrm{x},s}) &= \Big(\mathrm{diag}(\widetilde{\boldsymbol{\lambda}_\mathrm{x}^s}) + \sigma_g^{-2}\mathbf{K}_{\mathrm{x},m}\mathbf{K}_{\mathrm{x},m}^\top\Big)^{-1}\end{aligned}$$

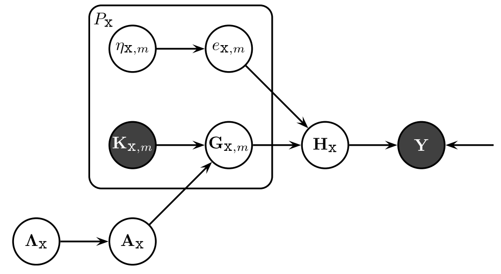

Fig. 5. Graphical model of kernelized Bayesian matrix factorization with twin multiple kernel learning for real-valued outputs.

$$\begin{aligned}
\mu(\boldsymbol{a}_{\text{x},s}) &= \Sigma(\boldsymbol{a}_{\text{x},s})\sigma_g^{-2}\mathbf{K}_{\text{x}}\widetilde{(\boldsymbol{g}_{\text{x}}^s)^\top} \\
\Sigma(\boldsymbol{g}_{\text{x},i}) &= \left(\sigma_g^{-2}\mathbf{I} + \widetilde{\mathbf{G}_{\text{z}}\mathbf{G}_{\text{z}}^\top}\right)^{-1} \\
\mu(\boldsymbol{g}_{\text{x},i}) &= \Sigma(\boldsymbol{g}_{\text{x},i})\left(\sigma_g^{-2}\widetilde{\mathbf{A}_{\text{x}}^\top}\boldsymbol{k}_{\text{x},i} + \widetilde{\mathbf{G}_{\text{z}}}\widetilde{(\boldsymbol{f}^i)^\top}\right) \\
\Sigma(f_j^i) &= 1 \\
\mu(f_j^i) &= \widetilde{\boldsymbol{g}_{\text{x},i}^\top}\widetilde{\boldsymbol{g}_{\text{z},j}} \\
\rho(f_j^i) &\triangleq f_j^i y_j^i > \nu.
\end{aligned}$$

Kernelized Bayesian matrix factorization with twin multiple kernel learning for real-valued outputs. We modify our proposed model for binary-valued outputs to also handle real-valued outputs. Fig. 5 illustrates the graphical model of the modified KBMF2MKL with latent variables and their corresponding priors.

The distributional assumptions of the modified KBMF2MKL model are

$$\begin{aligned}
\lambda_{\text{x},s}^i &\sim \mathcal{G}\left(\lambda_{\text{x},s}^i; \alpha_\lambda, \beta_\lambda\right) && \forall(i,s) \\
a_{\text{x},s}^i | \lambda_{\text{x},s}^i &\sim \mathcal{N}\left(a_{\text{x},s}^i; 0, \left(\lambda_{\text{x},s}^i\right)^{-1}\right) && \forall(i,s) \\
g_{\text{x},m,i}^s | \boldsymbol{a}_{\text{x},s}, \boldsymbol{k}_{\text{x},m,i} &\sim \mathcal{N}\left(g_{\text{x},m,i}^s; \boldsymbol{a}_{\text{x},s}^\top\boldsymbol{k}_{\text{x},m,i}, \sigma_g^2\right) && \forall(m,s,i) \\
\eta_{\text{x},m} &\sim \mathcal{G}(\eta_{\text{x},m}; \alpha_\eta, \beta_\eta) && \forall m \\
e_{\text{x},m} | \eta_{\text{x},m} &\sim \mathcal{N}\left(e_{\text{x},m}; 0, \eta_{\text{x},m}^{-1}\right) && \forall m \\
h_{\text{x},i}^s | \{e_{\text{x},m}, g_{\text{x},m,i}^s\}_{m=1}^{P_{\text{x}}} &\sim \mathcal{N}\left(h_{\text{x},i}^s; \sum_{m=1}^{P_{\text{x}}} e_{\text{x},m} g_{\text{x},m,i}^s, \sigma_h^2\right) && \forall(s,i) \\
y_j^i | \boldsymbol{h}_{\text{x},i}, \boldsymbol{h}_{\text{z},j} &\sim \mathcal{N}\left(y_j^i; \boldsymbol{h}_{\text{x},i}^\top\boldsymbol{h}_{\text{z},j}, \sigma_y^2\right) && \forall(i,j),
\end{aligned}$$

where σ_y denotes the noise level used for the target outputs.

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda, \sigma_g, \sigma_h, \sigma_y\}$, all prior variables by $\Xi = \{\boldsymbol{\eta}_{\text{x}}, \boldsymbol{\eta}_{\text{z}}, \Lambda_{\text{x}}, \Lambda_{\text{z}}\}$, and the remaining random variables by $\Theta = \{\mathbf{A}_{\text{x}}, \mathbf{A}_{\text{z}}, \boldsymbol{e}_{\text{x}}, \boldsymbol{e}_{\text{z}}, \{\mathbf{G}_{\text{x},m}\}_{m=1}^{P_{\text{x}}}, \{\mathbf{G}_{\text{z},n}\}_{n=1}^{P_{\text{z}}}, \mathbf{H}_{\text{x}}, \mathbf{H}_{\text{z}}\}$. We again omit the dependence on $\boldsymbol{\zeta}$ for clarity. We can write the factorized variational approximation as

$$\begin{aligned}
p(\Theta, \Xi | \{\mathbf{K}_{\text{x},m}\}_{m=1}^{P_{\text{x}}}, \{\mathbf{K}_{\text{z},n}\}_{n=1}^{P_{\text{z}}}, \mathbf{Y}) &\approx q(\Theta, \Xi) \\
&= q(\Lambda_{\text{x}})q(\mathbf{A}_{\text{x}})q(\{\mathbf{G}_{\text{x},m}\}_{m=1}^{P_{\text{x}}})q(\boldsymbol{\eta}_{\text{x}})q(\boldsymbol{e}_{\text{x}})q(\mathbf{H}_{\text{x}}) \\
&\quad q(\Lambda_{\text{z}})q(\mathbf{A}_{\text{z}})q(\{\mathbf{G}_{\text{z},n}\}_{n=1}^{P_{\text{z}}})q(\boldsymbol{\eta}_{\text{z}})q(\boldsymbol{e}_{\text{z}})q(\mathbf{H}_{\text{z}})
\end{aligned}$$

and define each factor in the ensemble just like its full conditional:

$$\begin{aligned}
q(\Lambda_{\text{x}}) &= \prod_{i=1}^{N_{\text{x}}}\prod_{s=1}^{R}\mathcal{G}\left(\lambda_{\text{x},s}^i; \alpha\left(\lambda_{\text{x},s}^i\right), \beta\left(\lambda_{\text{x},s}^i\right)\right) \\
q(\mathbf{A}_{\text{x}}) &= \prod_{s=1}^{R}\mathcal{N}(\boldsymbol{a}_{\text{x},s}; \mu(\boldsymbol{a}_{\text{x},s}), \Sigma(\boldsymbol{a}_{\text{x},s})) \\
q(\mathbf{G}_{\text{x},m}) &= \prod_{i=1}^{N_{\text{x}}}\mathcal{N}(\boldsymbol{g}_{\text{x},m,i}; \mu(\boldsymbol{g}_{\text{x},m,i}), \Sigma(\boldsymbol{g}_{\text{x},m,i})) \ \ \forall m \\
q(\boldsymbol{e}_{\text{x}}) &= \mathcal{N}(\boldsymbol{e}_{\text{x}}; \mu(\boldsymbol{e}_{\text{x}}), \Sigma(\boldsymbol{e}_{\text{x}})) \\
q(\boldsymbol{\eta}_{\text{x}}) &= \prod_{m=1}^{P_{\text{x}}}\mathcal{G}(\eta_{\text{x},m}; \alpha(\eta_{\text{x},m}), \beta(\eta_{\text{x},m})) \\
q(\mathbf{H}_{\text{x}}) &= \prod_{i=1}^{N_{\text{x}}}\mathcal{N}(\boldsymbol{h}_{\text{x},i}; \mu(\boldsymbol{h}_{\text{x},i}), \Sigma(\boldsymbol{h}_{\text{x},i})).
\end{aligned}$$

The approximate posterior distribution parameters of the ensemble can be found as

$$\begin{aligned}
\alpha(\lambda_{\text{x},s}^i) &= \alpha_\lambda + 1/2 \\
\beta(\lambda_{\text{x},s}^i) &= \left(1/\beta_\lambda + \widetilde{(a_{\text{x},s}^i)^2}/2\right)^{-1} \\
\Sigma(\boldsymbol{a}_{\text{x},s}) &= \left(\operatorname{diag}(\widetilde{\boldsymbol{\lambda}_{\text{x}}^s}) + \sum_{m=1}^{P_{\text{x}}}\sigma_g^{-2}\mathbf{K}_{\text{x},m}\mathbf{K}_{\text{x},m}^\top\right)^{-1} \\
\mu(\boldsymbol{a}_{\text{x},s}) &= \Sigma(\boldsymbol{a}_{\text{x},s})\sum_{m=1}^{P_{\text{x}}}\sigma_g^{-2}\mathbf{K}_{\text{x},m}\widetilde{(\boldsymbol{g}_{\text{x},m}^s)^\top} \\
\Sigma(\boldsymbol{g}_{\text{x},m,i}) &= \left(\sigma_g^{-2}\mathbf{I} + \sigma_h^{-2}\widetilde{e_{\text{x},m}^2}\mathbf{I}\right)^{-1} \\
\mu(\boldsymbol{g}_{\text{x},m,i}) &= \Sigma(\boldsymbol{g}_{\text{x},m,i})\Bigg(\sigma_g^{-2}\widetilde{\mathbf{A}_{\text{x}}^\top}\boldsymbol{k}_{\text{x},m,i} + \sigma_h^{-2}\widetilde{e_{\text{x},m}}\widetilde{\boldsymbol{h}_{\text{x},i}} \\
&\qquad - \sum_{o\neq m}\sigma_h^{-2}\widetilde{e_{\text{x},m}e_{\text{x},o}}\widetilde{\boldsymbol{g}_{\text{x},o,i}}\Bigg) \\
\alpha(\eta_{\text{x},m}) &= \alpha_\eta + 1/2 \\
\beta(\eta_{\text{x},m}) &= \left(1/\beta_\eta + \widetilde{e_{\text{x},m}^2}/2\right)^{-1} \\
\Sigma(\boldsymbol{e}_{\text{x}}) &= \left(\operatorname{diag}(\widetilde{\boldsymbol{\eta}_{\text{x}}}) + \left[\sigma_h^{-2}\operatorname{tr}\left(\widetilde{\mathbf{G}_{\text{x},m}^\top\mathbf{G}_{\text{x},o}}\right)\right]_{m=1,o=1}^{P_{\text{x}},P_{\text{x}}}\right)^{-1} \\
\mu(\boldsymbol{e}_{\text{x}}) &= \Sigma(\boldsymbol{e}_{\text{x}})\left[\sigma_h^{-2}\operatorname{tr}\left(\widetilde{\mathbf{G}_{\text{x},m}^\top}\widetilde{\mathbf{H}_{\text{x}}}\right)\right]_{m=1}^{P_{\text{x}}} \\
\Sigma(\boldsymbol{h}_{\text{x},i}) &= \left(\sigma_h^{-2}\mathbf{I} + \sigma_y^{-2}\widetilde{\mathbf{H}_{\text{z}}\mathbf{H}_{\text{z}}^\top}\right)^{-1} \\
\mu(\boldsymbol{h}_{\text{x},i}) &= \Sigma(\boldsymbol{h}_{\text{x},i})\left(\sum_{m=1}^{P_{\text{x}}}\sigma_h^{-2}\widetilde{e_{\text{x},m}}\widetilde{\boldsymbol{g}_{\text{x},m,i}} + \sigma_y^{-2}\widetilde{\mathbf{H}_{\text{z}}}(\boldsymbol{y}^i)^\top\right).
\end{aligned}$$

6 Application Scenarios

We motivate our algorithm with the bipartite graph inference scenario, but the method extends easily to other applications. We explicate three scenarios: (i) multilabel classification, (ii) multiple output regression, and (iii) learning with an incomplete label matrix.

Multilabel classification. In multilabel classification, each sample $\boldsymbol{x}_i$ is associated with a set of class labels $\boldsymbol{y}_i \in \{\pm 1\}^L$, where L is the number of labels, instead of just a single one. This setup can be cast into our formulation as follows: Samples and labels are assumed to come from domains $\mathcal{X}$ and $\mathcal{Z}$, respectively. The class membership matrix that contains the class labels of all training samples corresponds to the target label matrix $\mathbf{Y}$ to be factorized in our model. After learning the model parameters, predictions for an unseen test point $x_\star$ can be done as out-of-matrix predictions for a complete row in the target label matrix $y_\star$ using (6). Note that our model needs only kernel values between training samples for inference (i.e., inductive learning not transductive), and we can use the model parameters to make predictions for test samples using kernel values between training and test samples.

Our method allows us to integrate side information about samples and labels in the form of kernel matrices. For example, we can exploit the correlation between labels by integrating a kernel calculated over them into the model.

Multiple output regression. In multiple output regression, similarly to multilabel classification, each sample is associated with a set of real-valued target outputs. This setup has also been studied under the names *multivariate regression* and *multitask learning*. Multiple output regression can be cast into our framework using the formulation given in Section 5.

Learning with an incomplete label matrix. Up to this point, we assume that the target label matrix $\mathbf{Y}$ is fully observed. When there are missing entries in $\mathbf{Y}$, we have to modify the update equations for the composite components because they are the only random variables within the Markov blanket of $\mathbf{Y}$. Let $\mathcal{I}$ be the index set that contains the indices of observed entries in $\mathbf{Y}$. The approximate posterior distribution parameters of the composite components can be found as

$$\Sigma(\boldsymbol{h}_{\mathrm{x},i}) = \left(\sigma_h^{-2}\mathbf{I} + \sum_{(i,j)\in\mathcal{I}} \widetilde{\boldsymbol{h}_{\mathrm{z},j}\boldsymbol{h}_{\mathrm{z},j}^\top}\right)^{-1}$$

$$\mu(\boldsymbol{h}_{\mathrm{x},i}) = \Sigma(\boldsymbol{h}_{\mathrm{x},i})\left(\sum_{m=1}^{P_\mathrm{x}} \sigma_h^{-2}\widetilde{\boldsymbol{e}_{\mathrm{x},m}}\widetilde{\boldsymbol{g}_{\mathrm{x},m,i}} + \sum_{(i,j)\in\mathcal{I}} \widetilde{\boldsymbol{h}_{\mathrm{z},j}}\widetilde{f_j^i}\right),$$

where the approximate posterior distribution of each object now has a different covariance matrix parameter, leading to increased computational complexity. The same extension can also be applied to single kernel and regression scenarios, as explained in Section 5.

7 Experiments

We first run our method on a toy data set to illustrate its kernel learning capability. We then test its performance in a biomedical application with experiments on two drug-protein interaction data sets. One of them is a standard data set with a single view for each domain and the other one is a larger multiview data set. We also perform supervised and semi-supervised classification experiments on 14 benchmark multilabel data sets in order to show the suitability of our matrix factorization framework with side information in multilabel classification scenario. We finally perform multiple output regression experiments on a yeast cell cycle data set. Our Matlab and R implementations are available at http://research.ics.aalto.fi/mi/software/kbmf/.

7.1 Toy Data Set

We create a toy data set consisting of samples from two domains and real-valued outputs for object pairs. The data generation process is:

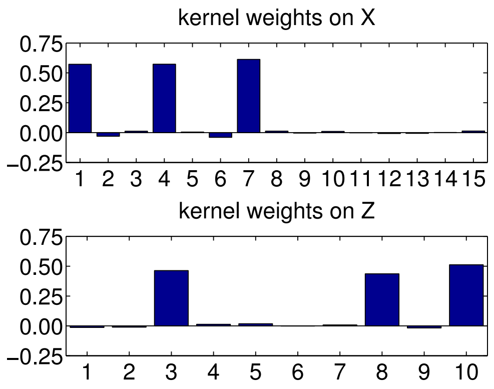

Fig. 6. Posterior means of the kernel weights found by our method on the toy data set.

$$\begin{aligned} x_i^m &\sim \mathcal{N}(x_i^m; 0, 1) && \forall(m,i) \\ z_j^n &\sim \mathcal{N}(z_j^n; 0, 1) && \forall(n,j) \\ y_j^i|\boldsymbol{x}_i, \boldsymbol{z}_j &\sim \mathcal{N}(y_j^i; x_i^1 z_j^3 + x_i^4 z_j^8 + x_i^7 z_j^{10}, 1) && \forall(i,j), \end{aligned}$$

where $(N_\mathrm{x}, N_\mathrm{z}) = (40, 60)$, the samples from $\mathcal{X}$ and $\mathcal{Z}$ are generated from 15- and 10-dimensional isotropic normal distributions with unit variance (i.e., $m \in \{1, \ldots, 15\}$ and $n \in \{1, \ldots, 10\}$), respectively, and the target outputs are generated using only three features from each domain. Note that this data set has not been generated from our probabilistic model.

In order to have multiple kernel functions for each domain, we calculate a separate linear kernel for each feature of the data points, i.e., $(P_\mathrm{x}, P_\mathrm{z}) = (15, 10)$. We then learn our model, intended to work as a predictive model that identifies the relevant features for the prediction task and has a good generalization performance. We use uninformative priors for the projection matrices and the kernel weights by setting $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$. The standard deviations are set to $(\sigma_g, \sigma_h, \sigma_y) = (0.1, 0.1, 1)$. The subspace dimensionality is arbitrarily set to five (i.e., $R = 5$).

Fig. 6 shows the found posterior means of the kernel weights. We see that our method correctly identifies the relevant features for each domain (i.e., the first, fourth, and seventh features for $\mathcal{X}$ and the third, eighth, and 10th features for $\mathcal{Z}$). The *root mean square error* between the target and predicted outputs is 0.99 in accordance with the level of noise added.

7.2 Drug-Protein Interaction Data Sets

As the first case study, we analyze a drug-protein interaction network of humans, involving enzymes in particular. This drug-protein interaction network contains 445 drugs, 664 proteins, and 2,926 experimentally validated interactions between them. The data set consists of the chemical similarity matrix between drugs, the genomic similarity matrix between proteins, and the target matrix of known interactions provided by [23]. The similarity matrices are used as kernel matrices. In the target interaction matrix, existing interactions are denoted by $+1$, and missing interactions are denoted by -1.

We compare one baseline and three matrix factorization methods: (i) `Baseline` simply calculates the target output

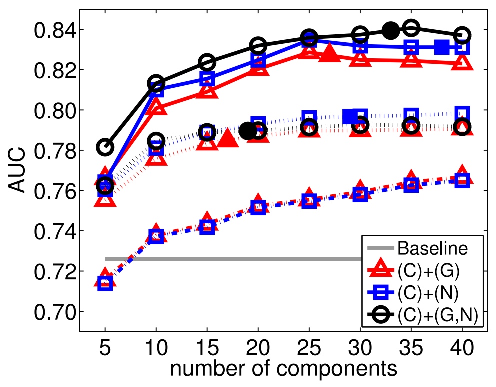

Fig. 7. Average prediction performances (area under ROC curve) on the drug-protein data set of [23]. Gray solid line: `Baseline`; other solid lines: `KBMF` with binary outputs; dotted lines: `KBMF` with real-valued outputs; dash-dotted lines: `KPMF`. Filled points show the model orders selected by our ARD variant.

averages over each column or row as the predictions, (ii) *kernelized probabilistic matrix factorization* (`KPMF`) method of [20] with real-valued outputs, (iii) our *kernelized Bayesian matrix factorization* (`KBMF`) method with real-valued outputs, and (iv) `KBMF` with binary outputs.

The experimental methodology is as follows: For `KPMF`, the standard deviation σ_y is set to one. For both `KBMF` variants, we use uninformative priors for the projection matrices and the kernel weights, i.e., $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$, and the standard deviations (σ_g, σ_h) are set to $(0.1, 0.1)$. For `KBMF` with real-valued outputs, the standard deviation σ_y is set to one. For `KBMF` with binary outputs, the margin parameter ν is arbitrarily set to one. We perform simulations with eight different numbers of components, i.e., $R \in \{5, 10, \dots, 40\}$. We run five replications of five-fold cross validation over drugs (i.e., leaving 20 percent of drugs out at each fold and making out-of-matrix prediction for them) and report the average *area under ROC curve* (AUC) over the 25 results as the performance measure. We also perform simulations using our variant with ARD to see whether it can select the model order automatically. We set $R = 40$ because our results showed that the performance stabilizes before 40 components with the original algorithms.

In the results, `C` and `G` mark the chemical similarity between drugs and the genomic similarity between proteins, respectively, whereas `N` marks the similarity between proteins calculated from the interaction network. It is defined as the ratio between (i) the number of drugs that are interacting with both proteins and (ii) the number of drugs that are interacting with at least one of the proteins, (i.e., the Jaccard index).

The results in Fig. 7 reveal that `KPMF` is above the baseline when the number of components is larger than 5, and both variants of `KBMF` for all component counts. Both variants of our new `KBMF` outperform the earlier `KPMF` for all types of inputs. The difference is not due to `KPMF` having been introduced only for real-valued outputs, as even the real-output variant of `KBMF` is better. The difference is not due to the inability of the current version of `KPMF` to handle multiple data views either, as the single-kernel `KBMF` outperforms it. Hence the differences in the performance are due to the differences in the inference: MAP point estimates versus fully Bayesian inference. The best results are obtained with the binary-output `KBMF` when using all data sources. We can also say that our variant with ARD is able to select the model orders automatically because the results with the selected model orders are very close to the best performance values obtained.

Note that when we combine the genomic and network similarities between proteins using our method, the resulting similarity measure for proteins is better than those of single-kernel scenarios, leading to better prediction performance. This shows that when we have multiple side information sources about the objects, integrating them into the matrix factorization model in a principled way improves the results.

We study an additional drug-protein interaction network of humans, containing 855 drugs, 800 proteins, and 4,659 experimentally validated interactions between them, extracted from the drugs and proteins of the data set provided by [41]. We have two views consisting of two standard 3D chemical structure descriptors for drugs, namely, 1,120-dimensional *Amanda* [42] and 76-dimensional *VolSurf* [43]. In order to calculate the similarity between drugs, we use a Gaussian kernel whose width parameter is selected as the square root of the dimensionality of the data points. In the target interaction matrix, existing interactions are denoted by $+1$, and missing interactions are denoted by -1.

We repeat the same experimental procedure as in the previous experiment with only one minor change. We perform simulations with 16 different numbers of components, i.e., $R \in \{5, 10, \dots, 80\}$, due to the larger size of the interaction network. However, this time, we set $R = 80$ for our simulations with ARD variant as the performance stabilizes before 80 components with the original algorithms.

We compare four different ways of including the drug property data. `Amanda` and `VolSurf` correspond to using a single view when calculating the kernel between drugs. `Early` corresponds to concatenating the two views, which is known as *early integration* [25], before calculating the kernel between drugs. `MKL` corresponds to calculating two different kernels between drugs and combining them with our kernel combination approach, which is known as *intermediate integration* [25].

The overall ordering of the results of the different matrix factorization methods is the same as in the previous case study (Fig. 8). The `KPMF` outperforms `Baseline` after 20 components, whereas `KBMF` is consistently and significantly better (by at least 4 percentage units) than `KPMF` for all single-kernel scenarios. We again see that our variant with ARD selects the model orders successfully. `KBMF` with five components is already better than `Baseline` for all scenarios. We do not report the results of `KBMF` with real-valued outputs in order not to clutter the figure.

For `KBMF` with binary outputs, we see that `Amanda` and `VolSurf` are significantly better than `Baseline` and obtain similar prediction performances. `Early` outperforms `Amanda` and `VolSurf` with a slight margin, whereas `MKL` obtains consistently better results than all the other scenarios after five components.

Our method can also be interpreted as a metric learning algorithm since each kernel function can be converted into a distance metric. We test this property on the task of

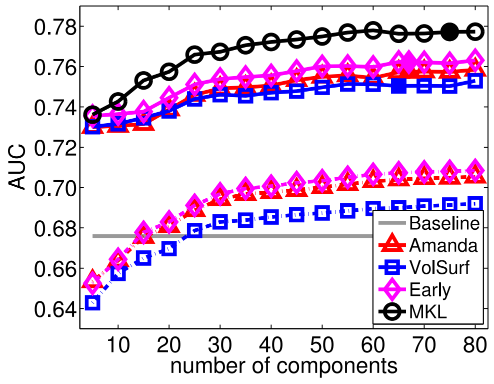

Fig. 8. Average prediction performances (area under ROC curve) on the drug-protein data set of [41]. Gray solid line: `Baseline`; solid lines: `KBMF` with binary outputs; dash-dotted lines: `KPMF`. Filled points show the model orders selected by our ARD variant.

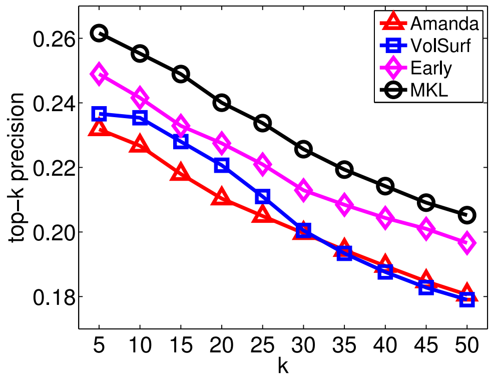

Fig. 9. Average precision (over query drugs) of retrieval as a function of number k of retrieved drugs. See the text for details.

finding or *retrieving* drugs with similar functions. The idea is that since the targets are centrally important for the action mechanisms of drugs, a metric useful for predicting targets could be useful for retrieval of drugs as well. As the ground truth for relevance we use a standard therapeutic classification of the drugs according to the organ or system on which they act and/or their chemical characteristics (not used during learning); drugs having the same class are considered relevant. Fig. 9 gives the precision at top-k retrieved drugs, when each drug in turn is used as the query and the rest of the 855 drugs are retrieved in the order of similarity according to the learned metric. `Early` is better than `Amanda` and `VolSurf`, and `MKL` is the best. This shows that our method is able to learn a kernel function between drugs that is better for retrieval than the kernels either on single or concatenated views.

7.3 Multilabel Classification

We compare our algorithm `KBMF` with five state-of-the-art multilabel classification algorithms, namely, (i) `RankSVM` [44], (ii) `ML-KNN` [45], (iii) `Tang's` [46], (iv) `RML` [47], and (v) `Zhang's` [48]. Note that [48] is also based on multiple kernel learning. We perform experiments on 14 benchmark multilabel classification data sets whose characteristics are given in Table 1. Here, $N_{\texttt{train}}$, $N_{\texttt{test}}$, D, and L denote the numbers of training instances, test instances, features, and labels, respectively. The first three data sets are obtained from http://mulan.sourceforge.net/datasets.html and the remaining 11 from http://www.kecl.ntt.co.jp/as/members/ueda/yahoo.tar.gz. We use the provided train/test splits in order to have comparable results with earlier studies.

We formulate multilabel classification as an out-of-matrix prediction problem in our matrix factorization framework as described in Section 6. The experimental methodology is as follows: The similarities between samples are measured with five different Gaussian kernels whose widths are selected as $\sqrt{D/4}$, $\sqrt{D/2}$, $\sqrt{D}$, $\sqrt{2D}$, and $\sqrt{4D}$, whereas the similarity between labels is measured with the Jaccard index over the labels of training samples. We use uninformative priors for the projection matrices and the kernel weights, i.e., $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$, and the standard deviations (σ_g, σ_h) are set to $(0.1, 0.1)$. The margin parameter ν is arbitrarily set to one. The number of components R is selected from $\{1, \ldots, \min(L, 15)\}$ according to training performance. We run a single replication using the provided train/test splits and report the Hamming loss as the performance measure.

Table 1 reports the classification results on multilabel data sets. The figures for comparison algorithms are taken from [48] and the best result for each data set is marked in boldface. `KBMF` obtains the best results on 10 out of 14 data sets and the second best results on the remaining four. In

TABLE 1
Classification Performances (i.e., Hamming Loss Values) on the Multilabel Classification Data Sets

Data Set	$N_{\texttt{train}}$	$N_{\texttt{test}}$	D	L	KBMF	Zhang's	ML-KNN	RML	Tang's	RankSVM
Emotions	391	202	72	6	**0.176**	0.195	0.202	0.241	0.240	0.234
Scene	1211	1196	294	6	**0.086**	0.089	0.099	0.109	0.130	0.127
Yeast	1500	917	103	14	**0.189**	0.196	0.195	0.204	0.190	0.201
Arts	2000	3000	462	26	**0.057**	**0.057**	0.061	0.058	0.094	0.063
Business	2000	3000	681	33	**0.025**	0.026	0.027	0.032	0.092	0.027
Computers	2000	3000	640	21	**0.036**	**0.036**	0.041	0.037	0.097	0.042
Education	2000	3000	606	22	0.039	**0.038**	0.039	0.050	**0.038**	0.048
Entertainment	2000	3000	743	40	**0.046**	0.055	0.063	0.059	0.053	0.062
Health	2000	3000	636	27	**0.036**	0.037	0.047	0.041	0.222	0.042
Recreation	2000	3000	438	30	**0.044**	0.057	0.062	0.057	0.057	0.064
Reference	2000	3000	550	33	0.027	**0.025**	0.032	0.027	0.087	0.034
Science	2000	3000	612	32	0.032	**0.031**	0.033	0.051	0.057	0.038
Social	2000	3000	793	33	0.022	**0.021**	0.022	0.101	0.072	0.027
Society	2000	3000	1047	39	**0.038**	0.052	0.054	0.096	0.056	0.060
				Average Rank	**1.536**	1.964	3.750	4.464	4.607	4.679

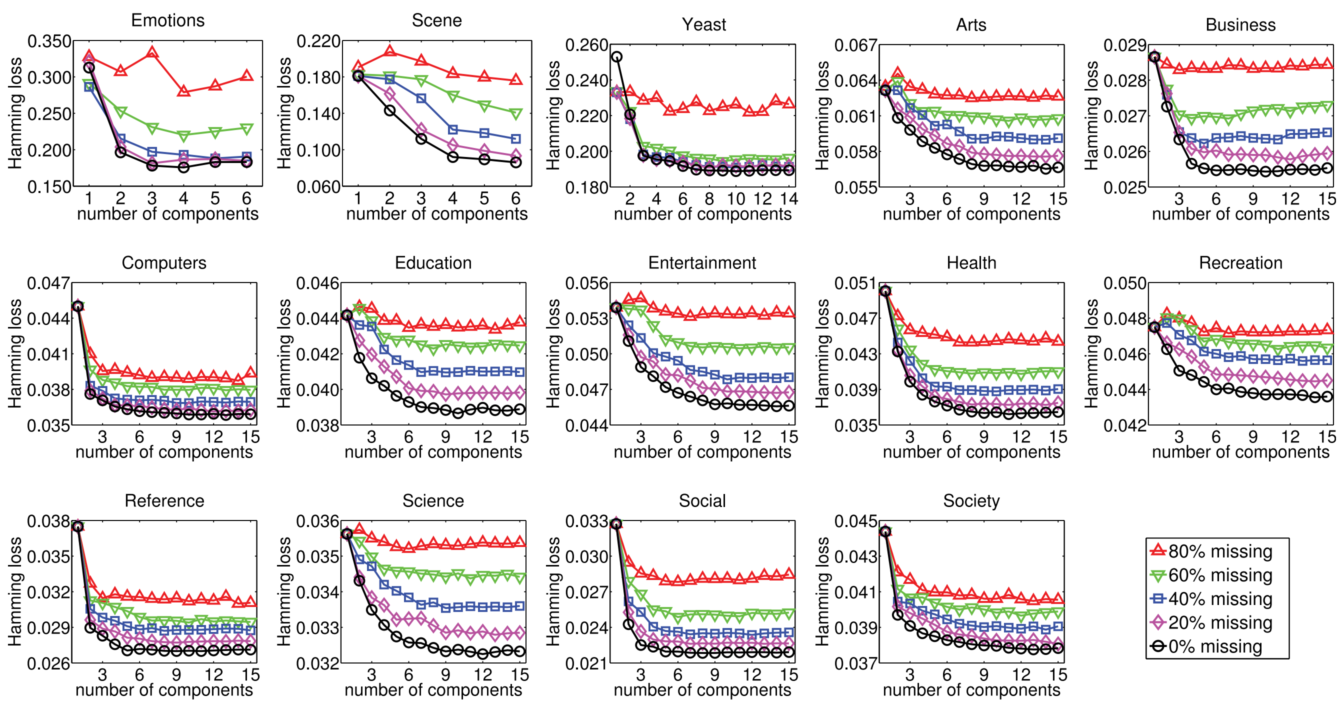

Fig. 10. Semi-supervised classification performances of KBMF on the multilabel classification data sets.

terms of average rank over 14 data sets, Zhang's method is better than other comparison algorithms, whereas KBMF has the best result. These results validate the suitability of our framework to multilabel classification.

We also run semi-supervised classification experiments on multilabel data sets with KBMF to show its capability of learning with incomplete target label matrices (details in Section 6). From each data set, we create four different semi-supervised data sets by randomly removing 20, 40, 60, and 80 percent of the target label matrix entries of training samples. We follow the same experimental setup as in the previous set of experiments.

Fig. 10 shows the semi-supervised classification performances of KBMF with different numbers of components. We see that, on all data sets, Hamming loss values decrease as we decrease the ratio of missing labels. As an important observation, KBMF does not lose much from its performance with 20 and 40 percent missing labels compared to the fully supervised case. This is the case in particular for the Emotions, Yeast, and Computers data sets.

7.4 Yeast Cell Cycle Data Set

We perform multiple output regression experiments on a biological data set, which consists of mRNA levels of 800 yeast genes within the eukaryotic cell cycle, provided by [49]. The data set includes mRNA levels measured every 7 minutes for 119 minutes, with a total of 18 time points covering two cell cycles. These 800 genes are represented using *chromatin immunoprecipitation* (ChIP) data, which contains binding information for a total of 106 transcription factors, provided by [50]. We use a subset of the original data set studied by [51] after removing the genes with missing mRNA levels or binding information, resulting in a data set with 524 genes.

Predicting mRNA levels of test genes using ChIP data can be cast into our formulation as follows: Genes and time points are assumed to be from domains $\mathcal{X}$ and $\mathcal{Z}$, respectively. The mRNA level measurements correspond to the target output matrix $\mathbf{Y}$ in our model. After learning the model parameters, we can do predictions for a test gene by doing out-of-matrix prediction for a complete row in the target output matrix.

We compare one baseline and two matrix factorization methods: (i) Baseline simply calculates the target output averages over each time point as the predictions, (ii) KPMF method of [20] with real-valued outputs, and (iii) KBMF method with real-valued outputs.

The experimental methodology is as follows: The similarities between genes are measured with the Gaussian kernel whose width is selected as the square root of the dimensionality of the data points, whereas the similarity between time points is selected as the correlation between mRNA level measurements of training genes. For KPMF and KBMF, the standard deviation σ_y is set to one. For KBMF, we use uninformative priors for the projection matrices and the kernel weights, i.e., $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$, and the standard deviations (σ_g, σ_h) are set to $(0.1, 0.1)$. We perform simulations with 15 different numbers of components, i.e., $R \in \{1, 2, \dots, 15\}$. We run five replications of five-fold cross validation over genes (i.e., leaving 20 percent of genes out at each fold and making out-of-matrix prediction for them) and report the average *mean squared error* (MSE) over the 25 results as the performance measure. We set $R = 15$ because our results showed that the performance stabilizes before 15 components with the original algorithm. Our variant with ARD selects the model order as $R = 6$, which obtains almost the same average MSE with 15 components.

The results in Fig. 11 reveal that both KPMF and KBMF are clearly better than the baseline even with one component.

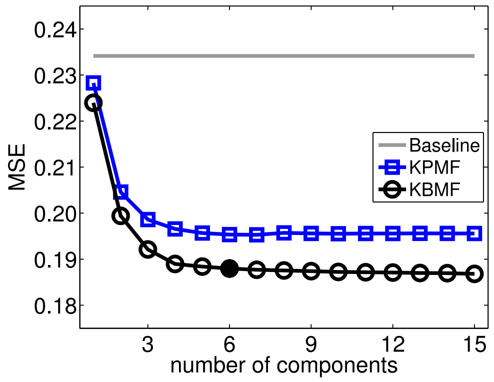

Fig. 11. Average prediction performances (mean squared error) on the yeast cell cycle data set of [49]. Filled point shows the model order selected by our ARD variant.

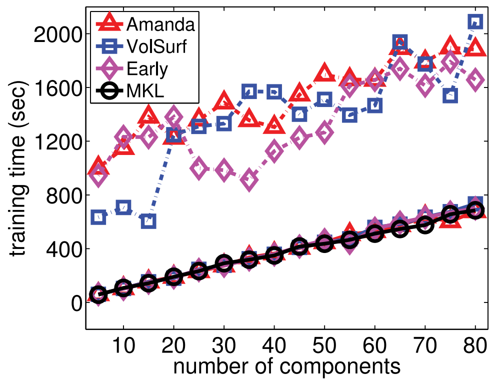

Fig. 12. Average training times on the drug-protein data set of [41]. Solid lines: KBMF with binary outputs; dash-dotted lines: KPMF.

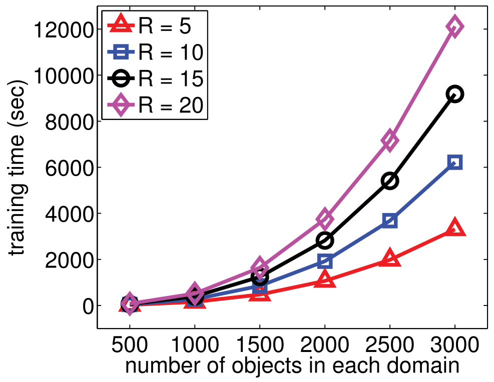

Fig. 13. Average training times of KBMF on randomly generated interaction data sets.

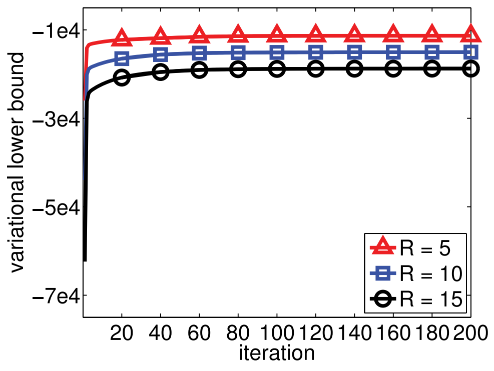

Fig. 14. Convergence behavior of KBMF with different numbers of components on the yeast data set of [49].

The ordering of the results of the two matrix factorization methods is the same as in the drug-protein interaction case studies. KBMF is consistently better than KPMF for all numbers of components.

7.5 Training Time and Convergence

We compare the training times of KPMF and KBMF on the drug-protein interaction data set of [41]. Fig. 12 shows the average training times over 25 replications. We can clearly see the advantage of KBMF in terms of training time. As detailed in Section 4, there is a clear linear trend in the training times of KBMF with respect to the number of components. We observe similar behavior for all other data sets used in this study (not shown). We also perform an additional set of experiments on randomly generated interaction data sets containing from 500 to 3,000 objects in each domain. We report the average training times of KBMF on these data sets as a function of the number of components in Fig. 13, which validates the computational complexity analysis of Section 4 (i.e., cubic scaling with respect to the number of objects).

In order to illustrate the convergence behavior of our algorithm, we report the variational lower bound of KBMF on the yeast cell cycle data set throughout 200 iterations in Fig. 14. We see that KBMF is able to quickly converge under three different settings before 50 iterations. We observe the same behavior for all data sets used in this study (not shown).

8 Discussion

We introduce a kernelized Bayesian matrix factorization method that can make use of multiple side information sources about the objects (both rows and columns) and be applied in various scenarios including recommender systems, interaction network modeling, multilabel classification, and multiple output regression. Our two main contributions are: (i) formulating an efficient variational approximation scheme for inference with the help of a novel fully conjugate probabilistic model and (ii) coupling matrix factorization with multiple kernel learning to integrate multiple side information sources into the model. In contrast to the earlier kernelized probabilistic matrix factorization method of [20], for our probabilistic model, it is possible to derive a computationally feasible fully Bayesian treatment. We also extend our model towards Bayesian model selection using automatic relevance determination and towards semi-supervised learning setup for partially observed output case. We illustrate the usefulness of the method on one toy data set, two molecular biological interaction data sets, 14 multilabel classification data sets, and one yeast cell cycle data set.

Acknowledgments

This work was financially supported by the Integrative Cancer Biology Program of the National Cancer Institute (grant no 1U54CA149237) and the Academy of Finland (grant no 140057 and Finnish Centre of Excellence in Computational Inference Research COIN, grant no 251170). Most of this

work has been done while the first author was working at the Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University.

REFERENCES

[1] N. Srebro, "Learning with matrix factorizations," Ph.D. dissertation, Massachusetts Inst. Tech., Cambridge, MA, USA, 2004..

[2] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[3] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 880–887.

[4] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 931–940.

[5] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 19–28.

[6] H. Shan and A. Banerjee, "Generalized probabilistic matrix factorizations for collaborative filtering," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 1025–1030.

[7] D. Agarwal and B.-C. Chen, "fLDA: Matrix factorization through latent Dirichlet allocation," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 91–100.

[8] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 448–456.

[9] J. Yoo and S. Choi, "Bayesian matrix co-factorization: Variational algorithm and Cramér-Rao bound," in *Proc. Eur. Conf. Mach. Learn. Principles Practice Knowl. Discovery Databases, Part III*, 2011, pp. 537–552.

[10] S. Park, Y.-D. Kim, and S. Choi, "Hierarchical Bayesian matrix factorization with side information," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1593–1599.

[11] K. T. Miller, T. L. Griffiths, and M. I. Jordan, "Nonparametric latent feature models for link prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1276–1284.

[12] A. K. Menon and C. Elkan, "Link prediction via matrix factorization," in *Proc. Eur. Conf. Mach. Learn. Principles Practice Knowl. Discovery Databases, Part II*, 2011, pp. 437–452.

[13] L. Zhang, D. Agarwal, and B.-C. Chen, "Generalizing matrix factorization through flexible regression priors," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 13–20.

[14] W. Chu, V. Sindhwani, Z. Ghahramani, and S. S. Keerthi, "Relational learning with Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 289–296.

[15] J. Luttinen and A. Ilin, "Variational Gaussian-process factor analysis for modeling spatio-temporal data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1177–1185.

[16] M. N. Schmidt and H. Laurberg, "Nonnegative matrix factorization with Gaussian process priors," *Comput. Intell. Neurosci.*, vol. 2008, pp. 1–10, 2008.

[17] M. N. Schmidt, "Function factorization using warped Gaussian processes," in *Proc. 26th Int. Conf. Mach. Learn.*, 2009, pp. 921–928.

[18] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with Gaussian processes," in *Proc. 26th Int. Conf. Mach. Learn.*, 2009, pp. 601–608.

[19] R. Adams, G. Dahl, and I. Murray, "Incorporating side information in probabilistic matrix factorization with Gaussian processes," in *Proc. 26th Conf. Annu Conf. Uncertainity Artif. Intell.*, 2010, pp. 1–9.

[20] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro, "Kernelized probabilistic matrix factorization: Exploiting graphs and side information," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 403–414.

[21] Y. Yamanishi, "Supervised bipartite graph inference," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1841–1848.

[22] A. Ben-Hur and W. S. Noble, "Kernel methods for predicting protein–protein interactions," *Bioinformatics*, vol. 21, no. suppl. 1, pp. i38–i46, 2005.

[23] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kaneisha, "Prediction of drug-target interaction networks from the integration of chemical and genomic spaces," *Bioinformatics*, vol. 24, pp. i232–i240, 2008.

[24] Y. Yamanishi, M. Kotera, M. Kanesiha, and S. Goto, "Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework," *Bioinformatics*, vol. 26, pp. i246–i254, 2010.

[25] *Kernel Methods in Computational Biology*, B. Schölkopf, K. Tsuda, and J.-P. Vert, eds., Cambridge, MA, USA: MIT Press, 2004.

[26] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, 2011.

[27] M. Gönen, S. A. Khan, and S. Kaski, "Kernelized Bayesian matrix factorization," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 864–872.

[28] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[29] M. Gönen, "Predicting drug-target interactions from chemical and genomic kernels using Bayesian matrix factorization," *Bioinformatics*, vol. 28, no. 18, pp. 2304–2310, 2012.

[30] J. H. Albert and S. Chib, "Bayesian analysis of binary and polychotomous response data," *J. Amer. Statist. Assoc.*, vol. 88, no. 422, pp. 669–679, 1993.

[31] N. D. Lawrence and M. I. Jordan, "Semi-supervised learning via Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 753–760.

[32] M. J. Beal, "Variational algorithms for approximate Bayesian inference," Ph.D. dissertation, The Gatsby Comput. Neurosci. Unit, Univ. College London, London, U.K., 2003.

[33] M. Gönen, "Bayesian efficient multiple kernel learning," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1–8.

[34] M. Girolami and S. Rogers, "Hierarchic Bayesian models for kernel learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 241–248.

[35] T. Damoulas and M. A. Girolami, "Probabilistic multi-class multikernel learning: On protein fold recognition and remote homology detection," *Bioinformatics*, vol. 24, no. 10, pp. 1264–1270, 2008.

[36] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 514–520.

[37] I. Murray and R. P. Adams, "Slice sampling covariance hyperparameters of latent Gaussian models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1732–1740.

[38] M. Filippone, M. Zhong, and M. Girolami, "A comparative evaluation of stochastic-based inference methods for Gaussian process models," *Mach. Learn.*, vol. 93, no. 1, pp. 93–114, 2013.

[39] R. M. Neal, *Bayesian Learning for Neural Networks*, New York, NY, USA: Springer, 1996.

[40] V. Y. F. Tan and C. Févotte, "Automatic relevance determination in nonnegative matrix factorization with the β-divergence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1592–1605, Jul. 2013.

[41] S. A. Khan, A. Faisal, J. P. Mpindi, J. A. Parkkinen, T. Kalliokoski, A. Poso, O. P. Kallioniemi, K. Wennerberg, and S. Kaski, "Comprehensive data-driven analysis of the impact of chemoinformatic structure on the genome-wide biological response profiles of cancer cells to 1159 drugs," *BMC Bioinformat.*, vol. 13, p. 112, 2012.

[42] A. Duran, G. C. Martinez, and M. Pastor, "Development and validation of AMANDA, a new algorithm for selecting highly relevant regions in molecular interaction fields," *J. Chem. Inf. Model.*, vol. 48, no. 9, pp. 1813–1823, 2008.

[43] G. Cruciani, M. Pastor, and W. Guba, "VolSurf: A new tool for the pharmacokinetic optimization of lead compounds," *Eur. J. Pharmaceutical Sci.*, vol. 11, no. suppl. 2, pp. S29–S39, 2000.

[44] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 681–687.

[45] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recog.*, vol. 40, no. 7, pp. 2038–2048, 2007.

[46] L. Tang, J. Chen, and J. Ye, "On multiple kernel learning with multiple labels," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, 2009, pp. 1255–1260.

[47] J. Petterson and T. Caetano, "Reverse multi-label learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1912–1920.

[48] W. Zhang, X. Xue, J. Fan, X. Huang, B. Wu, and M. Liu, “Multi-kernel multi-label learning with max-margin concept network,” in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2012, pp. 1615–1620.

[49] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botsein, and B. Futcher, “Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization,” *Mol. Biol. Cell*, vol. 9, no. 12, pp. 3273–3297, 1998.

[50] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R.A. Young, “Transcriptional regulatory networks in Saccharomyces cerevisiae,” *Science*, vol. 298, no. 5594, pp. 799–804, 2002.

[51] H. Chun and S. Keleş, “Sparse partial least squares regression for simultaneous dimension reduction and variable selection,” *J. Royal Statist. Soc.: Ser. B (Statist. Methodol.)*, vol. 72, pp. 3–25, 2010.

Mehmet Gönen received the BSc degree in industrial engineering, the MSc and the PhD degrees in computer engineering from Boğaziçi University, İstanbul, Turkey, in 2003, 2005, and 2010, respectively. He did his postdoctoral work at the Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University, Espoo, Finland. He is currently a senior research scientist at Sage Bionetworks, Seattle, Washington. His research interests include support vector machines, kernel methods, Bayesian methods, optimization for machine learning, dimensionality reduction, information retrieval, and computational biology applications.

Samuel Kaski received the DSc (PhD) degree in computer science from the Helsinki University of Technology, Finland, in 1997. He is currently a professor at Aalto University and the director of Helsinki Institute for Information Technology HIIT, Aalto University and the University of Helsinki, Finland. His research interests include statistical machine learning, computational biology and medicine, information visualization and proactive interfaces. He is a senior member of the IEEE.