ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-354

A Bayesian Multiple Kernel Learning Framework for Single and Multiple Output Regression

Mehmet Gönen [1]

Abstract. Multiple kernel learning algorithms are proposed to combine kernels in order to obtain a better similarity measure or to integrate feature representations coming from different data sources. Most of the previous research on such methods is focused on classification formulations and there are few attempts for regression. We propose a fully conjugate Bayesian formulation and derive a deterministic variational approximation for single output regression. We then show that the proposed formulation can be extended to multiple output regression. We illustrate the effectiveness of our approach on a single output benchmark data set. Our framework outperforms previously reported results with better generalization performance on two image recognition data sets using both single and multiple output formulations.

1 INTRODUCTION

The main idea of kernel-based algorithms is to learn a linear decision function in the feature space where data points are implicitly mapped to using a kernel function [12]. Given a sample of N independent and identically distributed training instances $\{\boldsymbol{x}_i \in \mathcal{X}\}_{i=1}^N$, the decision function that is used to predict the target output of an unseen test instance $\boldsymbol{x}_\star$ can be written as

$$f(\boldsymbol{x}_\star) = \boldsymbol{a}^\top \boldsymbol{k}_\star + b \tag{1}$$

where the vector of weights assigned to each training data point and the bias are denoted by $\boldsymbol{a}$ and b, respectively, and $\boldsymbol{k}_\star = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_\star) & \dots & k(\boldsymbol{x}_N, \boldsymbol{x}_\star) \end{bmatrix}^\top$ where $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel function that calculates a similarity measure between two data points. Using the theory of structural risk minimization, the model parameters can be found by solving a quadratic programming problem, known as *support vector machine* (SVM) [12]. The model parameters can also be interpreted as random variables to obtain a Bayesian interpretation of the model, known as *relevance vector machine* (RVM) [10].

Kernel selection (i.e., choosing a functional form and its parameters) is the most important issue that affects the empirical performance of kernel-based algorithms and usually done using a cross-validation procedure. *Multiple kernel learning* (MKL) methods have been proposed to make use of multiple kernels simultaneously instead of selecting a single kernel (see a recent survey [7]). Such methods also provide a principled way of integrating feature representations coming from different data sources or modalities. Most of the previous research is focused on developing MKL algorithms for classification problems. There are few attempts to formulate MKL models for regression problems [4, 8, 6, 13, 11]. Nevertheless, existing Bayesian MKL methods require too much computation time due to their dependency on sampling methods or complex inference procedures. In this paper, we propose an efficient Bayesian MKL framework for single and multiple output regression problems by formulating the combination with a fully conjugate probabilistic model.

In Section 2, we give an overview of the related work by considering existing MKL regression algorithms. Section 3 explains the proposed fully conjugate Bayesian formulation and deterministic variational inference scheme for single output regression. In Section 4, we extend this formulation to multiple output regression. Section 5 tests our framework, called *Bayesian multiple kernel learning for regression* (BMKLR), on a single output benchmark data set and reports very promising results on two image recognition data sets, which are frequently used to compare MKL algorithms.

2 RELATED WORK

MKL algorithms basically replace the kernel in (1) with a combined kernel calculated as a function of the input kernels. The most common combination is using a weighted sum of P kernels $\{k_m \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}\}_{m=1}^P$:

$$f(\boldsymbol{x}_\star) = \boldsymbol{a}^\top \underbrace{\left(\sum_{m=1}^P e_m \boldsymbol{k}_{m,\star} \right)}_{\boldsymbol{k}_{e,\star}} + b$$

where $\boldsymbol{k}_{m,\star} = \begin{bmatrix} k_m(\boldsymbol{x}_1, \boldsymbol{x}_\star) & \dots & k_m(\boldsymbol{x}_N, \boldsymbol{x}_\star) \end{bmatrix}^\top$ and the vector of kernel weights is denoted by $\boldsymbol{e}$. Existing MKL algorithms with a weighted sum differ in the way that they formulate restrictions on the kernel weights: arbitrary weights (i.e., linear sum), nonnegative weights (i.e., conic sum), or weights on a simplex (i.e., convex sum).

[8] proposes an MKL binary classification algorithm that optimizes the kernel weights and the support vector coefficients jointly with an alternating optimization strategy. Their algorithm requires to solve an SVM problem and to update the kernel weights at each iteration. They also show that their method can be extended to regression with minor modifications. [6] presents a localized MKL regression algorithm that allows us to combine kernels in a data-dependent way using gating functions. This algorithm can learn complex fits using multiple copies of a simple kernel (e.g., linear kernel) due to nonlinearity in the gating model.

[4] presents Bayesian MKL algorithms for regression and binary classification using hierarchical models. The combined kernel is defined as a convex sum of the input kernels using a Dirichlet prior on the kernel weights. As a consequence of the nonconjugacy between Dirichlet and normal distributions, they choose to use an im-

[1] Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University School of Science, email: mehmet.gonen@aalto.fi

portance sampling scheme to update the kernel weights when deriving variational approximations. [13] proposes a fully Bayesian inference methodology for extending generalized linear models to kernelized models using a Markov chain Monte Carlo approach. The main issue with these approaches is that they depend on some sampling strategy and may not be trained in a reasonable time when the number of kernels is large. Recently, [11] proposes a multitask GP model that combines a common set of GP functions (i.e., information sharing between the tasks) defined over multiple covariances with task-dependent weights whose sparsity is tuned using the spike and slab prior. A variational approximation approach is derived for an efficient inference scheme.

3 BAYESIAN MULTIPLE KERNEL LEARNING FOR SINGLE OUTPUT REGRESSION

In this section, we formulate a fully conjugate probabilistic model and develop a deterministic variational approximation inference procedure for single output regression. The main novelty of our approach is to calculate intermediate outputs from each kernel using the same set of weight parameters and to combine these outputs using the kernel weights to estimate the target output. This idea is originally proposed for binary classification problems [5]. Figure 1 illustrates the proposed probabilistic model for single output regression with a graphical model.

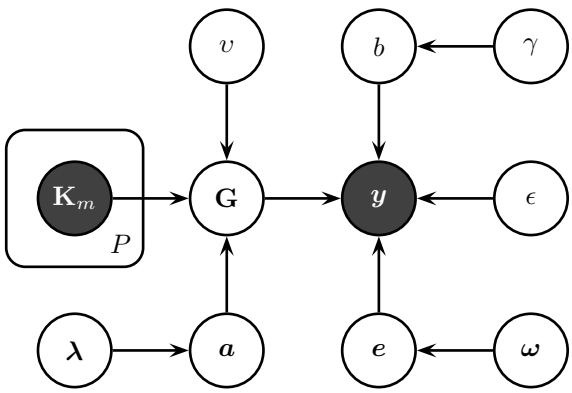

Figure 1. Bayesian multiple kernel learning for single output regression.

The notation we use for this model is as follows: N and P represent the numbers of training instances and input kernels, respectively. The $N \times N$ kernel matrices are denoted by $\mathbf{K}_m$, where the columns of $\mathbf{K}_m$ by $\boldsymbol{k}_{m,i}$ and the rows of $\mathbf{K}_m$ by $\boldsymbol{k}_m^i$. The $N \times 1$ vectors of weight parameters a_i and their precision priors λ_i are denoted by $\boldsymbol{a}$ and $\boldsymbol{\lambda}$, respectively. The $P \times N$ matrix of intermediate outputs g_i^m is represented as $\mathbf{G}$, where the columns of $\mathbf{G}$ as $\boldsymbol{g}_i$ and the rows of $\mathbf{G}$ as $\boldsymbol{g}^m$. The precision prior for intermediate outputs is denoted by υ. The bias parameter and its precision prior are denoted by b and γ, respectively. The $P \times 1$ vectors of kernel weights e_m and their precision priors ω_m are denoted by $\boldsymbol{e}$ and $\boldsymbol{\omega}$, respectively. The $N \times 1$ vector of target outputs y_i is represented as $\boldsymbol{y}$. The precision prior for target outputs is denoted by ϵ. As short-hand notations, all priors in the model are denoted by $\boldsymbol{\Xi} = \{\epsilon, \gamma, \boldsymbol{\lambda}, \boldsymbol{\omega}, \upsilon\}$, where the remaining variables by $\boldsymbol{\Theta} = \{\boldsymbol{a}, b, \boldsymbol{e}, \mathbf{G}\}$ and the hyper-parameters by $\boldsymbol{\zeta} = \{\alpha_\epsilon, \beta_\epsilon, \alpha_\gamma, \beta_\gamma, \alpha_\lambda, \beta_\lambda, \alpha_\omega, \beta_\omega, \alpha_\upsilon, \beta_\upsilon\}$. Dependence on $\boldsymbol{\zeta}$ is omitted for clarity throughout the rest.

The distributional assumptions of our probabilistic model are

$$
\begin{aligned}
\lambda_i &\sim \mathcal{G}(\lambda_i; \alpha_\lambda, \beta_\lambda) && \forall i \\
a_i|\lambda_i &\sim \mathcal{N}(a_i; 0, \lambda_i^{-1}) && \forall i \\
\upsilon &\sim \mathcal{G}(\upsilon; \alpha_\upsilon, \beta_\upsilon) \\
g_i^m|\boldsymbol{a}, \boldsymbol{k}_{m,i}, \upsilon &\sim \mathcal{N}(g_i^m; \boldsymbol{a}^\top \boldsymbol{k}_{m,i}, \upsilon^{-1}) && \forall (m,i) \\
\gamma &\sim \mathcal{G}(\gamma; \alpha_\gamma, \beta_\gamma) \\
b|\gamma &\sim \mathcal{N}(b; 0, \gamma^{-1}) \\
\omega_m &\sim \mathcal{G}(\omega_m; \alpha_\omega, \beta_\omega) && \forall m \\
e_m|\omega_m &\sim \mathcal{N}(e_m; 0, \omega_m^{-1}) && \forall m \\
\epsilon &\sim \mathcal{G}(\epsilon; \alpha_\epsilon, \beta_\epsilon) \\
y_i|b, \boldsymbol{e}, \boldsymbol{g}_i, \epsilon &\sim \mathcal{N}(y_i; \boldsymbol{e}^\top \boldsymbol{g}_i + b, \epsilon^{-1}) && \forall i.
\end{aligned}
$$

$\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the normal distribution with the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. $\mathcal{G}(\cdot; \alpha, \beta)$ denotes the gamma distribution with the shape parameter α and the scale parameter β. Sample-level sparsity can be tuned by assigning suitable values to the hyper-parameters $(\alpha_\lambda, \beta_\lambda)$ as in RVMs [10]. Kernel-level sparsity can also be tuned by changing the hyper-parameters $(\alpha_\omega, \beta_\omega)$. Sparsity-inducing gamma priors, e.g., (0.001, 1000), can simulate the ℓ_1-norm on the kernel weights, whereas uninformative priors, e.g., (1, 1), simulate the ℓ_2-norm.

Exact inference for our probabilistic model is intractable and using a Gibbs sampling approach is computationally expensive [3]. We instead formulate a deterministic variational approximation, which is more efficient in terms of computation time. The variational methods use a lower bound on the marginal likelihood using an ensemble of factored posteriors to find the joint parameter distribution [1]. We can write the factorable approximation of the required posterior as

$$
\begin{aligned}
p(\boldsymbol{\Theta}, \boldsymbol{\Xi}|\{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y}) \approx q(\boldsymbol{\Theta}, \boldsymbol{\Xi}) = \\
q(\boldsymbol{\lambda}) q(\boldsymbol{a}) q(\upsilon) q(\mathbf{G}) q(\gamma) q(\boldsymbol{\omega}) q(b, \boldsymbol{e}) q(\epsilon)
\end{aligned}
$$

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

$$
\begin{aligned}
q(\boldsymbol{\lambda}) &= \prod_{i=1}^N \mathcal{G}(\lambda_i; \alpha(\lambda_i), \beta(\lambda_i)) \\
q(\boldsymbol{a}) &= \mathcal{N}(\boldsymbol{a}; \mu(\boldsymbol{a}), \Sigma(\boldsymbol{a})) \\
q(\upsilon) &= \mathcal{G}(\upsilon; \alpha(\upsilon), \beta(\upsilon)) \\
q(\mathbf{G}) &= \prod_{i=1}^N \mathcal{N}(\boldsymbol{g}_i; \mu(\boldsymbol{g}_i), \Sigma(\boldsymbol{g}_i)) \\
q(\gamma) &= \mathcal{G}(\gamma; \alpha(\gamma), \beta(\gamma)) \\
q(\boldsymbol{\omega}) &= \prod_{m=1}^P \mathcal{G}(\omega_m; \alpha(\omega_m), \beta(\omega_m)) \\
q(b, \boldsymbol{e}) &= \mathcal{N}\left(\begin{bmatrix} b \\ \boldsymbol{e} \end{bmatrix}; \mu(b, \boldsymbol{e}), \Sigma(b, \boldsymbol{e}) \right) \\
q(\epsilon) &= \mathcal{G}(\epsilon; \alpha(\epsilon), \beta(\epsilon))
\end{aligned}
$$

where $\alpha(\cdot)$, $\beta(\cdot)$, $\mu(\cdot)$, and $\Sigma(\cdot)$ denote the shape parameter, the scale parameter, the mean vector, and the covariance matrix for their arguments, respectively.

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

$$\log p(\boldsymbol{y}|\{\mathbf{K}_m\}_{m=1}^P) \geq \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log p(\boldsymbol{y},\boldsymbol{\Theta},\boldsymbol{\Xi}|\{\mathbf{K}_m\}_{m=1}^P)] - \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log q(\boldsymbol{\Theta},\boldsymbol{\Xi})] \quad (2)$$

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

$$q(\boldsymbol{\tau}) \propto \exp(\mathrm{E}_{q(\{\boldsymbol{\Theta},\boldsymbol{\Xi}\}\setminus\boldsymbol{\tau})}[\log p(\boldsymbol{y},\boldsymbol{\Theta},\boldsymbol{\Xi}|\{\mathbf{K}_m\}_{m=1}^P)]).$$

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 of the precision priors for the weight parameters can be found as

$$q(\boldsymbol{\lambda}) = \prod_{i=1}^N \mathcal{G}\left(\lambda_i; \alpha_\lambda + \frac{1}{2}, \left(\frac{1}{\beta_\lambda} + \frac{\widetilde{a_i^2}}{2}\right)^{-1}\right)$$

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 approximate posterior distribution of the weight parameters is a multivariate normal distribution:

$$q(\boldsymbol{a}) = \mathcal{N}\left(\boldsymbol{a}; \Sigma(\boldsymbol{a})\left(\widetilde{\upsilon}\sum_{m=1}^P \mathbf{K}_m \widetilde{\boldsymbol{g}^m}\right), \left(\mathrm{diag}(\widetilde{\boldsymbol{\lambda}}) + \widetilde{\upsilon}\sum_{m=1}^P \mathbf{K}_m\mathbf{K}_m^\top\right)^{-1}\right). \quad (3)$$

The approximate posterior distribution of the precision prior for the intermediate outputs can be found as

$$q(\upsilon) = \mathcal{G}\left(\upsilon; \alpha_\upsilon + \frac{PN}{2}, \left(\frac{1}{\beta_\upsilon} + \frac{\sum_{m=1}^P\sum_{i=1}^N \widetilde{(r_i^m)^2}}{2}\right)^{-1}\right)$$

where $r_i^m = g_i^m - \boldsymbol{a}^\top \boldsymbol{k}_{m,i}$. The approximate posterior distribution of the intermediate outputs can be found as a product of multivariate normal distributions:

$$q(\mathbf{G}) = \prod_{i=1}^N \mathcal{N}\left(\boldsymbol{g}_i; \Sigma(\boldsymbol{g}_i)\left(\widetilde{\upsilon}\begin{bmatrix}\boldsymbol{k}_1^i \\ \vdots \\ \boldsymbol{k}_P^i\end{bmatrix}\widetilde{\boldsymbol{a}} + \widetilde{\epsilon}(y_i\widetilde{\boldsymbol{e}} - \widetilde{b\boldsymbol{e}})\right), \left(\widetilde{\upsilon}\mathbf{I} + \widetilde{\epsilon\boldsymbol{e}\boldsymbol{e}^\top}\right)^{-1}\right). \quad (4)$$

The approximate posterior distributions of the precision priors for the bias and the kernel weights can be found as

$$q(\gamma) = \mathcal{G}\left(\gamma; \alpha_\gamma + \frac{1}{2}, \left(\frac{1}{\beta_\gamma} + \frac{\widetilde{b^2}}{2}\right)^{-1}\right)$$

$$q(\boldsymbol{\omega}) = \prod_{m=1}^P \mathcal{G}\left(\omega_m; \alpha_\omega + \frac{1}{2}, \left(\frac{1}{\beta_\omega} + \frac{\widetilde{e_m^2}}{2}\right)^{-1}\right).$$

The approximate posterior distribution of the bias and the kernel weights can be formulated as a multivariate normal distribution:

$$q(b,\boldsymbol{e}) = \mathcal{N}\left(\begin{bmatrix}b \\ \boldsymbol{e}\end{bmatrix}; \Sigma(b,\boldsymbol{e})\begin{bmatrix}\widetilde{\epsilon}\mathbf{1}^\top\boldsymbol{y} \\ \widetilde{\epsilon}\widetilde{\mathbf{G}}\boldsymbol{y}\end{bmatrix}, \begin{bmatrix}\widetilde{\gamma} + \widetilde{\epsilon}N & \widetilde{\epsilon}\mathbf{1}^\top\widetilde{\mathbf{G}^\top} \\ \widetilde{\epsilon}\widetilde{\mathbf{G}}\mathbf{1} & \mathrm{diag}(\widetilde{\boldsymbol{\omega}}) + \widetilde{\epsilon\mathbf{G}\mathbf{G}^\top}\end{bmatrix}^{-1}\right) \quad (5)$$

where we allow kernel weights to take negative values.

The approximate posterior distribution of the precision prior for the target outputs can be found as

$$q(\epsilon) = \mathcal{G}\left(\epsilon; \alpha_\epsilon + \frac{N}{2}, \left(\frac{1}{\beta_\epsilon} + \frac{\sum_{i=1}^N \widetilde{s_i^2}}{2}\right)^{-1}\right)$$

where $s_i = y_i - \boldsymbol{e}^\top\boldsymbol{g}_i - b$.

$\sum_{m=1}^P \mathbf{K}_m\mathbf{K}_m^\top$ in (3) should be cached before starting inference to reduce the computational complexity. (3) requires inverting an $N \times N$ matrix for the covariance calculation, whereas (4) and (5) require inverting $P \times P$ and $(P+1) \times (P+1)$ matrices, respectively. One of these two update types will dominate the running time depending on whether $N > P$.

The inference mechanism sequentially updates the approximate posterior distributions of the model parameters and the latent variables until convergence, which can be checked by monitoring the lower bound in (2). 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.

In order to obtain the predictive distribution of the intermediate outputs $\boldsymbol{g}_\star$ for a new data point, we can replace $p(\boldsymbol{a}|\{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y})$ and $p(\upsilon|\{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y})$ with their approximate posterior distributions $q(\boldsymbol{a})$ and $q(\upsilon)$:

$$p(\boldsymbol{g}_\star|\{\boldsymbol{k}_{m,\star}, \mathbf{K}_m\}_{m=1}^P, \boldsymbol{y}) = \prod_{m=1}^P \mathcal{N}\left(g_\star^m; \mu(\boldsymbol{a})^\top\boldsymbol{k}_{m,\star}, \frac{1}{\widetilde{\upsilon}} + \boldsymbol{k}_{m,\star}^\top\Sigma(\boldsymbol{a})\boldsymbol{k}_{m,\star}\right).$$

The predictive distribution of the target output $y_\star$ can also be found by replacing $p(b,\boldsymbol{e}|\{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y})$ and $p(\epsilon|\{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y})$ with their approximate posterior distributions $q(b,\boldsymbol{e})$ and $q(\epsilon)$:

$$p(y_\star|\boldsymbol{g}_\star, \{\mathbf{K}_m\}_{m=1}^P, \boldsymbol{y}) = \mathcal{N}\left(y_\star; \mu(b,\boldsymbol{e})^\top\begin{bmatrix}1 \\ \boldsymbol{g}_\star\end{bmatrix}, \frac{1}{\widetilde{\epsilon}} + \begin{bmatrix}1 & \boldsymbol{g}_\star\end{bmatrix}\Sigma(b,\boldsymbol{e})\begin{bmatrix}1 \\ \boldsymbol{g}_\star\end{bmatrix}\right).$$

4 BAYESIAN MULTIPLE KERNEL LEARNING FOR MULTIPLE OUTPUT REGRESSION

The multiple output regression problems are generally considered as independent single output regression problems. In this approach, we can not make use of correlation between the outputs. Instead, we propose to use a common similarity measure by sharing the kernel weights between the outputs. We extend the probabilistic model of the previous section to handle multiple output regression. Figure 2 illustrates the modified probabilistic model for multiple output regression with a graphical model.

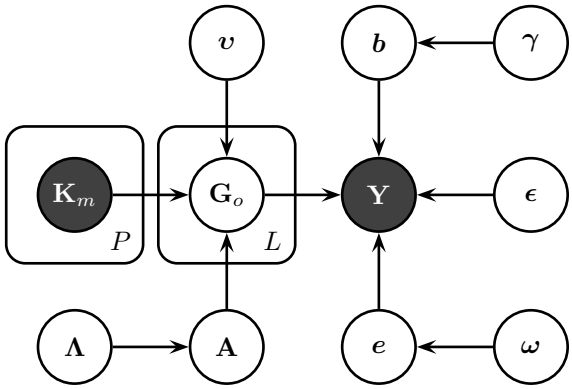

Figure 2. Bayesian multiple kernel learning for multiple output regression.

There are slight modifications to the notation but we explain them in detail for completeness. N, P, and L represent the numbers of training instances, input kernels, and outputs, respectively. The $N \times N$ kernel matrices are denoted by $\mathbf{K}_m$, where the columns of $\mathbf{K}_m$ by $\boldsymbol{k}_{m,i}$ and the rows of $\mathbf{K}_m$ by $\boldsymbol{k}_m^i$. The $N \times L$ matrices of weight parameters a_c^i and their priors λ_c^i are denoted by $\mathbf{A}$ and $\boldsymbol{\Lambda}$, respectively, where the columns of $\mathbf{A}$ and $\boldsymbol{\Lambda}$ by $\boldsymbol{a}_c$ and $\boldsymbol{\lambda}_c$. The $P \times N$ matrices of intermediate outputs for each output $g_{o,i}^m$ are represented as $\mathbf{G}_o$, where the columns of $\mathbf{G}_o$ as $\boldsymbol{g}_{o,i}$ and the rows of $\mathbf{G}_o$ as $\boldsymbol{g}_o^m$. The $L \times 1$ vector of precision priors υ_o for intermediate outputs is denoted by $\boldsymbol{\upsilon}$. The vectors of bias parameters b_o and their priors γ_o are denoted by $\boldsymbol{b}$ and $\boldsymbol{\gamma}$, respectively. The $P \times 1$ vectors of kernel weights e_m and their priors ω_m are denoted by $\boldsymbol{e}$ and $\boldsymbol{\omega}$, respectively. The $L \times N$ matrix of target outputs y_i^o is represented as $\mathbf{Y}$, where the columns of $\mathbf{Y}$ as $\boldsymbol{y}_i$ and the rows of $\mathbf{Y}$ as $\boldsymbol{y}^o$. The $L \times 1$ vector of precision priors ϵ_o for target outputs is denoted by $\boldsymbol{\epsilon}$. As short-hand notations, all priors in the model are denoted by $\boldsymbol{\Xi} = \{\boldsymbol{\epsilon}, \boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{\upsilon}\}$, where the remaining variables by $\boldsymbol{\Theta} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{e}, \mathbf{G}\}$ and the hyper-parameters by $\boldsymbol{\zeta} = \{\alpha_\epsilon, \beta_\epsilon, \alpha_\gamma, \beta_\gamma, \alpha_\lambda, \beta_\lambda, \alpha_\omega, \beta_\omega, \alpha_\upsilon, \beta_\upsilon\}$. Dependence on $\boldsymbol{\zeta}$ is omitted for clarity throughout the rest.

The distributional assumptions of our modified probabilistic model are defined as

$$
\begin{aligned}
\lambda_o^i &\sim \mathcal{G}(\lambda_o^i; \alpha_\lambda, \beta_\lambda) && \forall (i,o) \\
a_o^i | \lambda_o^i &\sim \mathcal{N}(a_o^i; 0, (\lambda_o^i)^{-1}) && \forall (i,o) \\
\upsilon_o &\sim \mathcal{G}(\upsilon_o; \alpha_\upsilon, \beta_\upsilon) && \forall o \\
g_{o,i}^m | \boldsymbol{a}_o, \boldsymbol{k}_{m,i}, \upsilon_o &\sim \mathcal{N}(g_i^m; \boldsymbol{a}_o^\top \boldsymbol{k}_{m,i}, \upsilon_o^{-1}) && \forall (o,m,i) \\
\gamma_o &\sim \mathcal{G}(\gamma_o; \alpha_\gamma, \beta_\gamma) && \forall o \\
b_o | \gamma_o &\sim \mathcal{N}(b_o; 0, \gamma_o^{-1}) && \forall o \\
\omega_m &\sim \mathcal{G}(\omega_m; \alpha_\omega, \beta_\omega) && \forall m \\
e_m | \omega_m &\sim \mathcal{N}(e_m; 0, \omega_m^{-1}) && \forall m \\
\epsilon_o &\sim \mathcal{G}(\epsilon_o; \alpha_\epsilon, \beta_\epsilon) && \forall o \\
y_i^o | b_o, \boldsymbol{e}, \boldsymbol{g}_{o,i}, \epsilon_o &\sim \mathcal{N}(y_i^o; \boldsymbol{e}^\top \boldsymbol{g}_{o,i} + b_o, \epsilon_o^{-1}) && \forall (o,i)
\end{aligned}
$$

where we use the same set of kernel weights for all outputs and this strategy enables us to capture the dependency between outputs.

We write the factorable approximation of the required posterior as

$$
\begin{aligned}
p(\boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_m\}_{m=1}^P, \mathbf{Y}) &\approx q(\boldsymbol{\Theta}, \boldsymbol{\Xi}) = \\
& q(\boldsymbol{\Lambda}) q(\mathbf{A}) q(\boldsymbol{\upsilon}) q(\{\mathbf{G}_o\}_{o=1}^L) q(\boldsymbol{\gamma}) q(\boldsymbol{\omega}) q(\boldsymbol{b}, \boldsymbol{e}) q(\boldsymbol{\epsilon})
\end{aligned}
$$

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

$$
\begin{aligned}
q(\boldsymbol{\Lambda}) &= \prod_{i=1}^N \prod_{o=1}^L \mathcal{G}(\lambda_o^i; \alpha(\lambda_o^i), \beta(\lambda_o^i)) \\
q(\mathbf{A}) &= \prod_{o=1}^L \mathcal{N}(\boldsymbol{a}_o; \mu(\boldsymbol{a}_o), \Sigma(\boldsymbol{a}_o)) \\
q(\boldsymbol{\upsilon}) &= \prod_{o=1}^L \mathcal{G}(\upsilon_o; \alpha(\upsilon_o), \beta(\upsilon_o)) \\
q(\{\mathbf{G}_o\}_{o=1}^L) &= \prod_{o=1}^L \prod_{i=1}^N \mathcal{N}(\boldsymbol{g}_{o,i}; \mu(\boldsymbol{g}_{o,i}), \Sigma(\boldsymbol{g}_{o,i})) \\
q(\boldsymbol{\gamma}) &= \prod_{o=1}^L \mathcal{G}(\gamma_o; \alpha(\gamma_o), \beta(\gamma_o)) \\
q(\boldsymbol{\omega}) &= \prod_{m=1}^P \mathcal{G}(\omega_m; \alpha(\omega_m), \beta(\omega_m)) \\
q(\boldsymbol{b}, \boldsymbol{e}) &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{e} \end{bmatrix}; \mu(\boldsymbol{b}, \boldsymbol{e}), \Sigma(\boldsymbol{b}, \boldsymbol{e}) \right) \\
q(\boldsymbol{\epsilon}) &= \prod_{o=1}^L \mathcal{G}(\epsilon_o; \alpha(\epsilon_o), \beta(\epsilon_o)).
\end{aligned}
$$

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

$$
\begin{aligned}
&\log p(\mathbf{Y} | \{\mathbf{K}_m\}_{m=1}^P) \geq \\
&\quad \mathrm{E}_{q(\boldsymbol{\Theta}, \boldsymbol{\Xi})}[\log p(\mathbf{Y}, \boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_m\}_{m=1}^P)] - \mathrm{E}_{q(\boldsymbol{\Theta}, \boldsymbol{\Xi})}[\log q(\boldsymbol{\Theta}, \boldsymbol{\Xi})]
\end{aligned}
$$

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

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

The approximate posterior distribution of the precision priors for the weight parameters can be found as

$$
q(\boldsymbol{\Lambda}) = \prod_{i=1}^N \prod_{o=1}^L \mathcal{G}\left(\lambda_o^i; \alpha_\lambda + \frac{1}{2}, \left(\frac{1}{\beta_\lambda} + \frac{\widetilde{(a_o^i)^2}}{2} \right)^{-1} \right).
$$

The approximate posterior distribution of the weight parameters is a multivariate normal distribution:

$$
\begin{aligned}
q(\mathbf{A}) = \prod_{o=1}^L \mathcal{N}\Bigg(\boldsymbol{a}_o; \Sigma(\boldsymbol{a}_o) \Bigg(\widetilde{\upsilon_o} \sum_{m=1}^P \mathbf{K}_m \widetilde{\boldsymbol{g}_o^{m\top}} \Bigg), \\
\left(\mathrm{diag}(\widetilde{\boldsymbol{\lambda}_o}) + \widetilde{\upsilon_o} \sum_{m=1}^P \mathbf{K}_m \mathbf{K}_m^\top \right)^{-1} \Bigg).
\end{aligned}
$$

The approximate posterior distribution of the precision priors for the intermediate outputs can be found as

$$
q(\boldsymbol{\upsilon}) = \prod_{o=1}^L \mathcal{G}\left(\upsilon_o; \alpha_\upsilon + \frac{PN}{2}, \left(\frac{1}{\beta_\upsilon} + \frac{\sum_{m=1}^P \sum_{i=1}^N \widetilde{(r_{o,i}^m)^2}}{2} \right)^{-1} \right)
$$

where $r_{o,i}^m = g_{o,i}^m - \boldsymbol{a}_o^\top \boldsymbol{k}_{m,i}$. The approximate posterior distribution of the intermediate outputs can be found as a product of multivariate normal distributions:

$$q(\{\mathbf{G}_o\}_{o=1}^L) = \prod_{o=1}^{L}\prod_{i=1}^{N}\mathcal{N}\left(\boldsymbol{g}_{o,i};\Sigma(\boldsymbol{g}_{o,i})\right.$$
$$\left.\left(\widetilde{\upsilon_o}\begin{bmatrix}\boldsymbol{k}_1^i\\ \vdots \\ \boldsymbol{k}_P^i\end{bmatrix}\widetilde{\boldsymbol{a}_o} + \widetilde{\epsilon_o}(y_i\widetilde{\boldsymbol{e}} - \widetilde{b_o}\boldsymbol{e})\right), \left(\widetilde{\upsilon_o}\mathbf{I} + \widetilde{\epsilon_o}\widetilde{\boldsymbol{e}\boldsymbol{e}^\top}\right)^{-1}\right).$$

The approximate posterior distributions of the precision priors for the bias parameters and the kernel weights can be found as

$$q(\boldsymbol{\gamma}) = \prod_{o=1}^{L}\mathcal{G}\left(\gamma_o;\alpha_\gamma+\frac{1}{2},\left(\frac{1}{\beta_\gamma}+\frac{\widetilde{b_o^2}}{2}\right)^{-1}\right)$$
$$q(\boldsymbol{\omega}) = \prod_{m=1}^{P}\mathcal{G}\left(\omega_m;\alpha_\omega+\frac{1}{2},\left(\frac{1}{\beta_\omega}+\frac{\widetilde{e_m^2}}{2}\right)^{-1}\right).$$

The approximate posterior distribution of the bias parameters and the kernel weights can be formulated as a multivariate normal distribution:

$$q(\boldsymbol{b},\boldsymbol{e}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{e}\end{bmatrix};\Sigma(\boldsymbol{b},\boldsymbol{e})\begin{bmatrix}\widetilde{\epsilon_1}\boldsymbol{y}^1\mathbf{1}\\ \vdots\\ \widetilde{\epsilon_L}\boldsymbol{y}^L\mathbf{1}\\ \sum_{o=1}^{L}\widetilde{\epsilon_o}\widetilde{\mathbf{G}_o}\boldsymbol{y}^{o\top}\end{bmatrix},\right.$$
$$\left.\begin{bmatrix} & & \widetilde{\epsilon_1}\mathbf{1}^\top\widetilde{\mathbf{G}_1^\top}\\ \mathrm{diag}(\widetilde{\boldsymbol{\gamma}})+\mathrm{diag}(\widetilde{\boldsymbol{\epsilon}})N & & \vdots\\ & & \widetilde{\epsilon_L}\mathbf{1}^\top\widetilde{\mathbf{G}_L^\top}\\ \widetilde{\epsilon_1}\widetilde{\mathbf{G}_1}\mathbf{1} & \dots & \widetilde{\epsilon_L}\widetilde{\mathbf{G}_L}\mathbf{1} \quad \mathrm{diag}(\widetilde{\boldsymbol{\omega}})+\sum_{o=1}^{L}\widetilde{\epsilon_o}\widetilde{\mathbf{G}_o\mathbf{G}_o^\top}\end{bmatrix}^{-1}\right).$$

The approximate posterior distribution of the precision priors for the target outputs can be found as

$$q(\boldsymbol{\epsilon}) = \prod_{o=1}^{L}\mathcal{G}\left(\epsilon_o;\alpha_\epsilon+\frac{N}{2},\left(\frac{1}{\beta_\epsilon}+\frac{\sum_{i=1}^{N}\widetilde{(s_i^o)^2}}{2}\right)^{-1}\right)$$

where $s_i^o = y_i^o - \boldsymbol{e}^\top\boldsymbol{g}_{o,i} - b_o$.

In order to obtain the predictive distribution of the intermediate outputs $\mathbf{G}_{\cdot,\star}$ for a new data point, we can replace $p(\mathbf{A}|\{\mathbf{K}_m\}_{m=1}^P,\mathbf{Y})$ and $p(\boldsymbol{\upsilon}|\{\mathbf{K}_m\}_{m=1}^P,\mathbf{Y})$ with their approximate posterior distributions $q(\mathbf{A})$ and $q(\boldsymbol{\upsilon})$:

$$p(\mathbf{G}_{\cdot,\star}|\{\boldsymbol{k}_{m,\star},\mathbf{K}_m\}_{m=1}^P,\mathbf{Y}) =$$
$$\prod_{o=1}^{L}\prod_{m=1}^{P}\mathcal{N}\left(g_{o,\star}^m;\mu(\boldsymbol{a}_o)^\top\boldsymbol{k}_{m,\star},\frac{1}{\widetilde{\upsilon_o}}+\boldsymbol{k}_{m,\star}^\top\Sigma(\boldsymbol{a}_o)\boldsymbol{k}_{m,\star}\right).$$

The predictive distribution of the target output $\boldsymbol{y}_\star$ can also be found by replacing $p(\boldsymbol{b},\boldsymbol{e}|\{\mathbf{K}_m\}_{m=1}^P,\mathbf{Y})$ and $p(\boldsymbol{\epsilon}|\{\mathbf{K}_m\}_{m=1}^P,\mathbf{Y})$ with their approximate posterior distributions $q(\boldsymbol{b},\boldsymbol{e})$ and $q(\boldsymbol{\epsilon})$:

$$p(\boldsymbol{y}_\star|\mathbf{G}_{\cdot,\star},\{\mathbf{K}_m\}_{m=1}^P,\mathbf{Y}) =$$
$$\prod_{o=1}^{L}\mathcal{N}\left(y_\star^o;\mu(b_o,\boldsymbol{e})^\top\begin{bmatrix}1\\ \boldsymbol{g}_{o,\star}\end{bmatrix},\frac{1}{\widetilde{\epsilon_o}}+\begin{bmatrix}1 & \boldsymbol{g}_{o,\star}\end{bmatrix}\Sigma(b_o,\boldsymbol{e})\begin{bmatrix}1\\ \boldsymbol{g}_{o,\star}\end{bmatrix}\right).$$

5 EXPERIMENTS

We first test our new framework BMKLR on a single output benchmark data set to show its effectiveness. We then illustrate its generalization performance comparing it with previously reported MKL results on two image recognition data sets. We implement the proposed variational approximations for BMKLR in Matlab and our implementations are available at `http://users.ics.aalto.fi/gonen/bmklr/`.

5.1 Motorcycle data set

We show the effectiveness of our framework on `Motorcycle` data set discussed in [9]. The data set is normalized to have zero mean and unit standard deviation. We construct Gaussian kernels with 21 different widths ($\{2^{-10},2^{-9},\dots,2^{+10}\}$). We force sparsity at both sample-level and kernel-level using sparsity inducing Gamma priors for the sample and kernel weights. The hyper-parameter values are selected as $(\alpha_\epsilon,\beta_\epsilon) = (\alpha_\gamma,\beta_\gamma) = (\alpha_\upsilon,\beta_\upsilon) = (1,1)$ and $(\alpha_\lambda,\beta_\lambda) = (\alpha_\omega,\beta_\omega) = (10^{-10},10^{+10})$.

Figure 3 gives the kernel weights obtained by BMKLR on `Motorcycle` data set. We see that most of the kernels are eliminated from the combination with zero weights. Gaussian kernels with widths $\{2^{-2},2^{-1},2^{+1}\}$ are enough to get a good fit for this data set as we will see next.

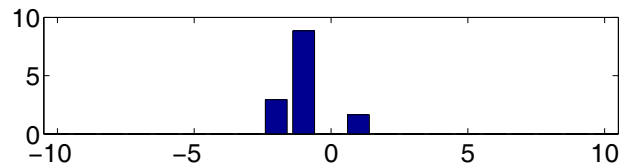

Figure 3. Kernel weights on `Motorcycle` data set obtained by BMKLR.

Figure 4 shows the fitted curve superimposed with the training samples obtained by BMKLR on `Motorcycle` data set. Only three samples shown as filled points have nonzero weights. BMKLR is able to learn a very good fit to the data using only three training samples and three out of 21 input kernels in the final decision function.

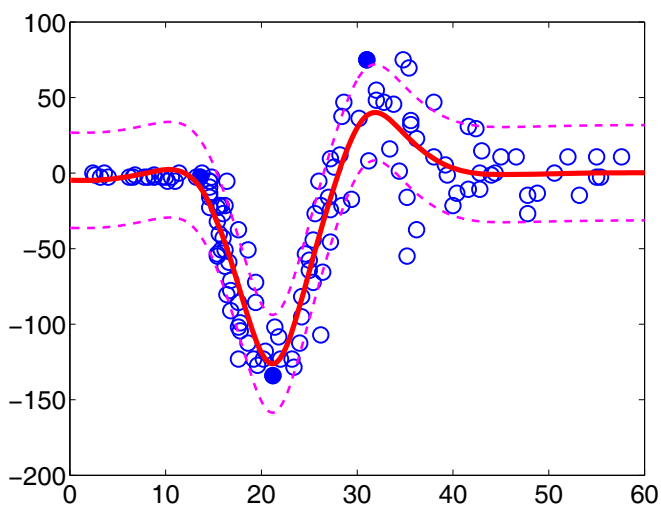

Figure 4. Fit on `Motorcycle` data set obtained by BMKLR. Training samples with nonzero weights are shown as filled points.

5.2 Oxford flower data sets

We use two data sets, namely, `Flowers17` and `Flowers102`, that are previously used to compare MKL algorithms and have kernel matrices available for direct evaluation. The proposed framework has been designed for regression problems, but it can also be used approximately to solve classification problems using output values to identify class membership. Both data sets consider multiclass classification problems and we set the corresponding output to $+1$ for the target class, whereas the outputs for the others are set to -1. We have small numbers of kernels available and do not force any sparsity on the sample and kernel weights. The hyper-parameter values are selected as $(\alpha_\epsilon, \beta_\epsilon) = (\alpha_\gamma, \beta_\gamma) = (\alpha_\lambda, \beta_\lambda) = (\alpha_\omega, \beta_\omega) = (\alpha_\upsilon, \beta_\upsilon) = (1, 1)$. We report both single and multiple output results.

`Flowers17` data set contains flower images from 17 different types with 80 images per class and it is available at http://www.robots.ox.ac.uk/~vgg/data/flowers/17/. It also provides three predefined splits with 60 images for training and 20 images for testing from each class. There are seven precomputed distance matrices over different feature representations. These matrices are converted into kernels as $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_j)/s)$ where s is the mean distance between training point pairs. The classification results on `Flowers17` data set are shown in Table 1. BMKLR achieves higher average test accuracy with smaller standard deviation across splits than the boosting-type MKL algorithm of [2]. On this data set, single output approach is better than multiple output approach in terms of classification accuracy.

Table 1. Performance comparison on `Flowers17` data set.

Method	Test Accuracy
[2]	85.5±3.0
BMKLR (single output)	86.2±1.6
BMKLR (multiple output)	86.0±1.6

`Flowers102` data set contains flower images from 102 different types with more than 40 images per class and it is available at http://www.robots.ox.ac.uk/~vgg/data/flowers/102/. There is a predefined split consisting of 2040 training and 6149 testing images. There are four precomputed distance matrices over different feature representations. These matrices are converted into kernels using the same procedure on `Flowers17` data set. Table 2 shows the classification results on `Flowers102` data set. We report averages of *area under ROC curve* (AUC) and *equal error rate* (EER) values calculated for each class in addition to multiclass accuracy. We see that BMKLR outperforms the GP-based method of [11] in all metrics on this challenging task. Different from `Flowers17` data set, multiple output approach is better than single output approach in terms of classification accuracy. This shows that sharing the same kernel function between the outputs improves the generalization performance.

Table 2. Performance comparison on `Flowers102` data set.

Method	AUC	EER	Accuracy
[11]	0.952	0.107	40.0
BMKLR (single output)	0.977	0.060	68.8
BMKLR (multiple output)	0.977	0.062	69.3

Note that the results used for comparison may not be the best results reported on these data sets but we use the exact same kernel matrices that produce these results for our algorithm to have comparable performance measures.

6 CONCLUSIONS

In this paper, we introduce a Bayesian multiple kernel learning framework for single and multiple output regression. This framework allows us to develop fully conjugate probabilistic models and to derive very efficient deterministic variational approximations for inference. We give detailed derivations of the inference procedures for single and multiple output regression scenarios. Our algorithm for single output regression can be interpreted as multiple kernel variant of RVM [10].

Experimental results on a single output benchmark data set shows the effectiveness of our method in terms of kernel learning capability. We also report very promising results on two image recognition data sets, which contain multiclass classification problems, compared to previously reported results.

When extending our single output formulation to multiple output regression, we assume that the outputs share the same kernel weights. This strategy may not be suitable for the problems where the outputs depend on different feature subsets. This setup requires to have different kernel functions for different outputs and we can choose to share the same sample weights instead of the kernel weights. In such a case, the model uses the same set of training points to calculate a single set of intermediate outputs and combines them with different sets of kernel weights for each output. However, we leave this extension for future research.

ACKNOWLEDGEMENTS

This work was financially supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, grant no 251170).

REFERENCES

[1] M. J. Beal, *Variational Algorithms for Approximate Bayesian Inference*, Ph.D. dissertation, The Gatsby Computational Neuroscience Unit, University College London, 2003.

[2] P. Gehler and S. Nowozin, 'On feature combination for multiclass object classification', in *Proceedings of the 12th International Conference on Computer Vision*, (2009).

[3] A. E. Gelfand and A. F. M. Smith, 'Sampling-based approaches to calculating marginal densities', *Journal of the American Statistical Association*, **85**, 398–409, (1990).

[4] M. Girolami and S. Rogers, 'Hierarchic Bayesian models for kernel learning', in *Proceedings of the 22nd International Conference on Machine Learning*, (2005).

[5] M. Gönen, 'Bayesian efficient multiple kernel learning', in *Proceedings of the 29th International Conference on Machine Learning*, (2012).

[6] M. Gönen and E. Alpaydın, 'Localized multiple kernel regression', in *Proceedings of the 20th International Conference on Pattern Recognition*, (2010).

[7] M. Gönen and E. Alpaydın, 'Multiple kernel learning algorithms', *Journal of Machine Learning Research*, **12**, 2211–2268, (2011).

[8] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, 'SimpleMKL', *Journal of Machine Learning Research*, **9**, 2491–2521, (2008).

[9] B. W. Silverman, 'Some aspects of the spline smoothing approach to non-parametric regression curve fitting', *Journal of the Royal Statistical Society: Series B*, **47**, 1–52, (1985).

[10] M. E. Tipping, 'Sparse Bayesian learning and the relevance vector machine', *Journal of Machine Learning Research*, **1**, 211–244, (2001).

[11] M. K. Titsias and M Lázaro-Gredilla, 'Spike and slab variational inference for multi-task and multiple kernel learning', in *Advances in Neural Information Processing Systems 24*, (2011).

[12] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, 1998.

[13] Z. Zhang, G. Dai, and M. I. Jordan, 'Bayesian generalized kernel mixed models', *Journal of Machine Learning Research*, **12**, 111–139, (2011).