Efficient Bayesian Maximum Margin Multiple Kernel Learning

Changying Du[1,2(✉)], Changde Du[3], Guoping Long[1], Xin Jin[4], and Yucheng Li[1]

[1] Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
changying@iscas.ac.cn
[2] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
[3] Research Center for Brain-inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[4] Central Software Institute, Huawei Technologies Co. Ltd., Beijing 100085, China

Abstract. Multiple Kernel Learning (MKL) suffers from slow learning speed and poor generalization ability. Existing methods seldom address these problems well simultaneously. In this paper, by defining a multi-class (pseudo-) likelihood function that accounts for the margin loss for kernelized classification, we develop a robust Bayesian maximum margin MKL framework with Dirichlet and the three parameter Beta normal priors imposed on the kernel and sample combination weights respectively. For inference, we exploit the data augmentation idea and devise an efficient MCMC algorithm in the augmented variable space, employing the Riemann manifold Hamiltonian Monte Carlo technique to sample from the conditional posterior of kernel weights, and making use of local conjugacy for all other variables. Such geometry and conjugacy based posterior sampling leads to very fast mixing rate and scales linearly with the number of kernels used. Extensive experiments on classification tasks validate the superiority of the proposed method in both efficacy and efficiency.

1 Introduction

Kernel-based machine learning is a popular technique for dealing with nonlinearities in real prediction tasks. The performance of this kind of learning methods generally is determined by two orthogonal aspects, i.e., the selected kernel function and the learning principle. On one hand, a kernel function implicitly maps the input data points to an infinite-dimensional feature space and actually provides a similarity measure on it. Since the learning process is conducted in the feature space, the appropriateness of the chosen kernel usually is crucial for the final modelling quality. On the other hand, the learning principle, e.g., the maximum margin principle in Support Vector Machine (SVM), defines the searching strategy in the hypothesis space, and thus is responsible for model generalization ability.

P. Frasconi et al. (Eds.): ECML PKDD 2016, Part I, LNAI 9851, pp. 165–181, 2016.
DOI: 10.1007/978-3-319-46128-1_11

Though (single) kernel selection can be done via cross-validation on the training data, there are at least two reasons for studying Multiple Kernel Learning (MKL) [21], an active research topic that aims at learning a linear (or convex) combination of a set of predefined kernels in order to identify a good target kernel for the applications (see [17] for a survey). First, from the perspective of users without sufficient domain knowledge, it is desirable to design algorithms that can learn effective kernels automatically from data. Second and more important, to achieve superior performance it is necessary for many real applications to fully exploit the rich features underlying each sample. A promising way to achieve this is to define a large set of kernel mappings on all features and each individual feature, and then learn the optimal combination of them.

Through the past decade, there have been lots of MKL studies, most of which were focused on seeking (appropriately regularized) max-margin point model estimates [9,20,33,34]. Adopting the max-margin principle, these models essentially have advantage in yielding good generalization performance. However, their deterministic point estimate formulations make them less robust to noisy and small training data. Under the Bayesian framework, there already exist some MKL methods [10,14,15] that estimate the entire posterior distribution of model weights. Unfortunately, these methods either require matrix inversions to compute the posterior covariance of kernel weights or have to perform time-consuming importance sampling, and thus scale poorly with the number of kernels used. Moreover, since the max-margin hinge loss does not lend itself to a convenient description of a likelihood function, the combination of Bayesian MKL and max-margin principle has been deemed as intractable for a long time.

In this paper, by defining a multiclass (pseudo-) likelihood function that accounts for the margin loss for kernelized classification, we develop an efficient Bayesian maximum margin MKL framework. The Bayesian model averaging mechanism along with the max-margin principle allow us to make robust predictions with the guarantee of arguably good generalization performance. Moreover, imposing the sparsity-inducing Three Parameter Beta Normal (TPBN) prior [2] and the Dirichlet prior on the sample and kernel combination weights respectively, the resultant model has good interpretability and adaptivity. For inference, we exploit the data augmentation idea and devise an efficient Markov Chain Monte Carlo (MCMC) algorithm in the augmented variable space, employing the Riemann manifold Hamiltonian Monte Carlo (HMC) technique to sample from the conditional posterior of kernel weights, and making use of local conjugacy for all other variables. Such geometry and conjugacy based posterior sampling leads to very fast mixing rate and scales linearly with the number of kernels used.

Extensive experiments on both binary and multiclass classification data sets show that the proposed Bayesian max-margin MKL model not only outperforms a number of competitors consistently in terms of prediction performance but also requires substantially fewer training time when the number of kernels is large.

2 Related Work

Compared with traditional kernel methods using a single fixed kernel, MKL provides a natural way for the automated kernel parameter tuning, the integration of diverse nonlinear mappings, and the concatenation of heterogeneous data. It was originally formulated as a semi-definite programming (SDP) problem in [21], and then improved with the quadratically constrained quadratic programming (QCQP) [4], and the semi-infinite linear programming (SILP) [34].

Over the past decade, MKL has been actively studied, and a variety of algorithms have been proposed to address the efficiency of MKL, e.g., the adaptive 2-norm regularization formulation [32], the extended level method [36], the group lasso based methods [3,37], the proximal minimization method [35], the online-batch strongly convex two-stage method [29], the spectral projected gradient descent method [19], and the mean-field variational inference method [15]. Besides, a lot of extended MKL techniques have been proposed to improve the regular MKL method, e.g., the localized MKL [7,16,38] that achieve local assignments of kernel weights at the group level, the sample-adaptive MKL [24,28] that switches off kernels at the data sample level, the absent MKL [23] that handles the channel missing problem of individual samples, and the Bayesian MKL [14,15,22] that estimate the entire posterior distribution of model weights. Our method differs from existing efficient MKL algorithms in that, it employs the Riemann manifold HMC technique to sample from the conditional posterior of kernel weights, and makes use of local conjugacy for all other variables. Such geometry and conjugacy based posterior sampling leads to very fast mixing rate and scales linearly with the number of kernels used. Our method also differs from existing Bayesian MKL model since it is based on the max-margin (pseudo-) likelihood and data augmentation idea, and tends to has better generalization performance.

As sparsity-inducing approaches to kernel weight learning rarely outperform trivial baselines in practical applications [8,20], we choose the Dirichlet prior for kernel combination weights in our Bayesian MKL framework. Though it has been considered in [14], our expanded-mean parameterization of the Dirichlet is particularly suitable for HMC based methods, which is more efficient than importance sampling based variational approximation. Besides, our choice of the sparsity-inducing TPBN prior for sample combination weights differs from the Gaussian-inverse-Gamma prior used in [14,15]. As in kernelized SVM, the sparse sample weights is responsible for selecting support vectors actually needed in decision function, and thus good interpretability can be obtained.

We note that the GP-based Bayesian nonlinear SVM model proposed in [18] adopts a similar data augmentation idea for max-margin learning and infers its GP kernel parameters automatically with slice sampling. Its difference from ours is that it was designed for single kernel binary classification, and our model can be seen as an efficient multiclass multi-kernel extension of it. As detailed in the following and observed in the experiments, such extension is not trivial.

3 Bayesian Max-Margin MKL

Suppose we have a set of labeled data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $y_i \in \{1, 2, ..., C\}$ and $\mathbf{x}_i \in \mathcal{X}$, a d-dimensional Euclidean space. MKL algorithms typically use a weighted sum of P kernels $\{\mathfrak{K}_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\}_{m=1}^P$ to measure data similarity. For multiclass leaning, we adopt the one-versus-all strategy, and learn a shared kernel combination weights vector $\mathbf{w}$ for all binary sub-problems, which corresponds to sharing the similarity measure when jointly learn all sub-problems [7,15,33]. Note that, this sharing not only is essential for efficient computation when the number of classes is large, but also is important to alleviate overfitting when we only have very few training data. Specifically, we define the following decision function for the c-th binary sub-problem:

$$f(\mathbf{x}_i; \Theta_c) \;=\; \mathbf{a}_c^\top \left(\sum\nolimits_{m=1}^{P} w_m \mathbf{K}_{m,\cdot i} \right) + b_c \tag{1}$$

where $\mathbf{K}_{m,\cdot i} = [\mathfrak{K}_m(\mathbf{x}_i, \mathbf{x}_1), ..., \mathfrak{K}_m(\mathbf{x}_i, \mathbf{x}_N)]^\top$ contains the similarities between $\mathbf{x}_i$ and each training example under the kernel feature mapping $\mathfrak{K}_m$; $\mathbf{a}_c$ and b_c are the sample weights and bias for the c-th binary sub-problem, respectively; $\mathbf{w} = [w_1, ..., w_P]^\top$ is the kernel weights shared by all sub-problems; $\Theta_c = \{\mathbf{a}_c, b_c, \mathbf{w}\}$.

3.1 Max-Margin Pseudo-likelihood

To account for the training error on $(\mathbf{x}_i, y_i)$, we further define the following multiclass max-margin pseudo-likelihood function:

$$L(y_i|\Theta) = \exp\Big\{ -2 \sum\nolimits_{c=1}^{C} \max(\zeta_{ci}, 0) \Big\}, \tag{2}$$

where $\zeta_{ci} = 1 - \delta_{y_i,c} f(\mathbf{x}_i; \Theta_c)$, $\delta_{y_i,c} = 1$ if $y_i = c$ and -1 otherwise. This pseudo-likelihood plays the similar role as that in the Bayesian SVM model [31], which addressed linear binary classification only. Intuitively, the negative margin losses of each binary multiple kernel classifier are summed up and passed through an exponential transformation. The larger the loss, the smaller the likelihood is. Despite its importance in our Bayesian MKL modeling, (2) makes direct posterior inference intractable due to the max function in $L(\Theta)$. Fortunately, the following identity holds [1]:

$$\exp\{-|\zeta|\} = \int_0^\infty \frac{\exp\{\frac{-\zeta^2}{2\varpi} - \frac{\varpi}{2}\}}{\sqrt{2\pi\varpi}} d\varpi. \tag{3}$$

Multiplying through (3) by $\exp\{-\zeta\}$ and noting $\max(\zeta, 0) = \frac{1}{2}(|\zeta| + \zeta)$, we have:

$$L(y_i|\Theta) = \prod_{c=1}^{C} \int_0^\infty \frac{\exp\{\frac{-1}{2\lambda_{ci}}(\lambda_{ci} + \zeta_{ci})^2\}}{\sqrt{2\pi\lambda_{ci}}} d\lambda_{ci}, \tag{4}$$

which allows us to introduce auxiliary variables to the original inference problem. Thus, by regarding the original posterior as the marginal of a higher dimensional distribution that involves the augmented variables $\boldsymbol{\lambda}$, we can bypass the calculation of the max function. Consequently, efficient algorithms can be designed.

3.2 Priors on Model Parameters

Symmetric Dirichlet Prior on Kernel Weights. Under the Bayesian framework, we can impose either Dirichlet [14] or Gaussian [15] prior on $\mathbf{w}$. While Gaussian prior is convenient for inference, it cannot ensure the positivity of each kernel weight, which sometimes is difficult for interpretation and even degrades performance. Thus, we consider the symmetric Dirichlet prior $\mathbf{w} \sim \mathrm{Dir}(\eta)$ for kernel weights, where $\eta > 0$. An asymmetric prior can be used if the user has a rough idea about kernel importance.

Note that, the Dirichlet prior on kernel weights leads to non-conjugacy even if we can re-express our pseudo-likelihood as the product of C location-scale mixtures of normals. Such a non-conjugacy issue generally complicates posterior inference. To address it, many strategies have been used in literature, ranging from variational approximations to Metropolis-Hastings methods. Unlike the inefficient importance sampling method in [14], we will explore the recently developed Riemann manifold HMC [13] approach. Exploiting the Riemannian geometry of the parameter space, RHMC can efficiently samples from a continuous distribution with its unnormalized probability density.

Sparsity-inducing Prior on Sample Weights. Similar as in kernelized SVM, the sample weights $\mathbf{a}$ is often expected to be sparse for selecting support vectors actually needed in decision function. In this paper, we choose the Three Parameter Beta Normal (TPBN) [2] as the sparsity-inducing prior due to its better mixing properties than priors such as the spike-and-slab, the Student's-t prior, and the double exponential prior. The TPBN prior can be expressed as scale mixtures of normals and favors strong shrinkage of small signals while having heavy tails to avoid over-shrinkage of the larger signals. If $a_{ci} \sim \mathrm{TPBN}(\alpha_a, \beta_a, \kappa)$, $c = 1, ..., C$, $i = 1, ..., N$, then:

$$a_{ci} \sim \mathcal{N}(0, \nu_{ci}), \quad \nu_{ci} \sim \Gamma(\alpha_a, \varsigma_{ci}), \quad \varsigma_{ci} \sim \Gamma(\beta_a, \kappa),$$

where $\mathcal{N}(\cdot)$ and $\Gamma(\cdot)$ denote the Gaussian and Gamma (shape-rate parameterization) distribution respectively. One advantage of this hierarchical shrinkage prior is the full local conjugacy that allows posterior inference easily implemented. For fixed values of α_a and β_a, decreasing the parameter κ encourages stronger shrinkage.

Normal Prior on Biases. Finally, an isotropic normal prior is imposed on the bias vector $\mathbf{b}$, i.e. $\mathbf{b} \sim \mathcal{N}(\mathbf{0}, \tau \mathbf{I}_C)$, where $\mathbf{I}_C$ is a C-dimensional identity matrix.

4 Inference via Posterior Sampling

4.1 Augmenting the Posterior

As stated above, (2) makes direct posterior inference intractable due to the max function in $L(\Theta)$. However, it is easy to verify that the posterior of our model is the marginal of

$$q(\Theta, \boldsymbol{\lambda}|\mathcal{D}) = p_0(\Theta) \prod_{i=1}^{N} L(y_i, \boldsymbol{\lambda}_{\cdot i}|\Theta)/Z(\mathcal{D}), \tag{5}$$

where $p_0(\Theta)$ is the model prior, $\boldsymbol{\lambda}_{\cdot i}$ denotes a vector of C augmented variables (each for one class) for $\mathbf{x}_i$, and

$$L(y_i, \boldsymbol{\lambda}_{\cdot i}|\Theta) = \prod_{c=1}^{C} \frac{\exp\{\frac{-1}{2\lambda_{ci}}(\lambda_{ci} + \zeta_{ci})^2\}}{\sqrt{2\pi\lambda_{ci}}}. \tag{6}$$

The above property indicates that we can bypass the calculation of the max function through sampling from the augmented posterior (5), and the interested information about the original posterior can be recovered by discarding $\boldsymbol{\lambda}$. Though (5) still is intractable to compute analytically due to the normalization constant, it is not difficult to develop MCMC algorithms by making use of local conjugacy and the Riemann HMC.

4.2 Efficient Geometry-Based MCMC

In the following, we devise a Gibbs sampling algorithm that generates a sample from the posterior distribution of each variable in turn, conditional on the current values of the other variables. It can be shown that the sequence of samples constitutes a Markov chain, and the stationary distribution of that Markov chain is just the joint posterior.

Given $\boldsymbol{\lambda}$, $\mathbf{a}$ and $\mathbf{b}$, the conditional (augmented) posterior distribution of $\mathbf{w}$ is

$$q(\mathbf{w}|\boldsymbol{\lambda}, \mathbf{a}, \mathbf{b}, \eta, \mathcal{D}) \propto \mathrm{Dir}(\mathbf{w}; \eta) \cdot \prod_{i=1}^{N} L(y_i, \boldsymbol{\lambda}_{\cdot i}|\Theta),$$

where $L(y_i, \boldsymbol{\lambda}_{\cdot i}|\Theta)$ can be transformed into a Gaussian density of $\mathbf{w}$. Since the Dirichlet prior is not conjugate to the Gaussian distribution, it is hard to get the analytical form of the above distribution. To generate a sample from $q(\mathbf{w}|\boldsymbol{\lambda}, \mathbf{a}, \mathbf{b}, \eta, \mathcal{D})$ with its unnormalized density, we appeal to the Riemann Hamiltonian Monte Carlo (RHMC) [13] approach. As in HMC [27], which simulates the Hamiltonian dynamics, RHMC proposes samples with auxiliary momentum variables $\mathbf{r}$ in a Metropolis-Hastings (MH) framework. The difference from ordinary HMC is that, RHMC explores the underlying geometry of the target distribution to accelerate mixing.

In doing so, the problem is that the Dirichlet prior represents our belief that $\mathbf{w}$ should lie on the probability simplex $\{(w_1, ..., w_P) : \ w_m \geq 0, \ \sum_m w_m = 1\} \subset \mathbb{R}^P$, which is compact and has boundaries that has to be accounted for when an update proposes a step that brings the vector outside the simplex. There are several possible ways to simplify boundary considerations via parameterizing the probability simplex, and the performance of RHMC depends strongly on the choice of parameterization. As studied in [30], the expanded-mean parameterization yields higher effective sample size and more efficient computation, so it is adopted here. Specifically, we introduce a P-dimensional unnormalized parameter $\mathbf{e}$ with a product of P independent Gamma distributions, i.e., $p(\mathbf{e}) \propto \prod_{m=1}^{P} e_m^{\eta-1} \exp(-e_m)$. Setting $w_m = e_m / \sum_{m=1}^{P} e_m$ for $m = 1, ..., P$, the prior on $\mathbf{w}$ is still $\mathrm{Dir}(\eta)$, while the conditional posterior of $\mathbf{e}$ is

$$q(\mathbf{e}|\boldsymbol{\lambda},\mathbf{a},\mathbf{b},\eta,\mathcal{D}) \propto \prod_{m=1}^{P} e_m^{\eta-1}\exp(-e_m)\cdot\prod_{c=1}^{C}\prod_{i=1}^{N}\frac{\exp\{\frac{-1}{2\lambda_{ci}}(\lambda_{ci}+\zeta_{ci})^2\}}{\sqrt{2\pi\lambda_{ci}}},$$

where $\zeta_{ci} = 1-\delta_{y_i,c}(\mathbf{a}_c^\top\mathbf{h}^i+b_c)$, $\mathbf{h}^i = \sum_{m=1}^{P}\left(\frac{e_m}{\sum_m e_m}\right)\cdot\mathbf{K}_{m,\cdot i}$, $\delta_{y_i,c}=1$ if $y_i=c$ and -1 otherwise.

Then we consider the Hamiltonian $H(\mathbf{e},\mathbf{r}) = -\log q(\mathbf{e}|\boldsymbol{\lambda},\mathbf{a},\mathbf{b},\eta,\mathcal{D}) + \frac{1}{2}\mathbf{r}^\top\mathbf{r}$, and use the following transition rule to generate proposals:

$$\mathbf{r}^* = \mathbf{r} + \epsilon\mathbf{G}(\mathbf{e})^{-\frac{1}{2}}\nabla\log q(\mathbf{e}|\boldsymbol{\lambda},\mathbf{a},\mathbf{b},\eta,\mathcal{D}) + \epsilon\nabla\mathbf{G}(\mathbf{e})^{-\frac{1}{2}} - \epsilon\mathbf{G}(\mathbf{e})^{-1}\mathbf{r} + \xi, \quad (7)$$

$$\mathbf{e}^* = |\mathbf{e} + \epsilon\mathbf{G}(\mathbf{e})^{-\frac{1}{2}}\mathbf{r}^*|, \quad (8)$$

where ϵ is the step size, $\xi\sim\mathcal{N}(\mathbf{0},2\epsilon\mathbf{G}(\mathbf{e})^{-1})$ is the added Gaussian noise, and $\mathbf{G}(\mathbf{e}) = \mathrm{diag}(\mathbf{e})^{-1}$ is the Riemann manifold used to precondition the dynamics in a locally adaptive manner. Note that, the boundary reflection idea (by taking the absolute value of the proposed new $\mathbf{e}$) is used as in [30] to ensure the positivity.

The other posterior conditional distributions can be derived analytically as follows using the local conjugacy properties.

For $\boldsymbol{\lambda}$: The conditional distribution of λ_{ci} is a generalized inverse Gaussian (GIG) distribution:

$$q(\lambda_{ci}|\Theta,\mathcal{D}) = \mathcal{GIG}\Big(\frac{1}{2},\ 1,\ \zeta_{ci}^2\Big). \quad (9)$$

See [11] for generating random variates from the GIG distribution.

For $\mathbf{a}$, $\mathbf{b}$: The conditional distribution of $\mathbf{a}_c$ and b_c is

$$q(\mathbf{a}_c,b_c|\boldsymbol{\lambda},\mathbf{e},\boldsymbol{\nu},\tau,\mathcal{D}) \propto \exp\Big(-\mathbf{a}_c^\top\Lambda_{\boldsymbol{\nu}_{c\cdot}}\mathbf{a}_c - \tau b_c^2 - \sum_i\frac{(\lambda_{ci}+\zeta_{ci})^2}{2\lambda_{ci}}\Big),$$

a multivariate Gaussian with covariance and mean:

$$\Sigma_{(\mathbf{a},b)} = \left(\begin{bmatrix}\Lambda_{\boldsymbol{\nu}_{c\cdot}}^{-1}, & 0\\ 0, & \tau^{-1}\end{bmatrix} + \sum_i\frac{1}{\lambda_{ci}}\begin{bmatrix}\mathbf{h}^i\\ 1\end{bmatrix}\begin{bmatrix}\mathbf{h}^i\\ 1\end{bmatrix}^\top\right)^{-1}, \quad (10)$$

$$\mu_{(\mathbf{a},b)} = \Sigma_{(\mathbf{a},b)}\sum_i\left(\delta_{y_i,c}+\frac{\delta_{y_i,c}}{\lambda_{ci}}\right)\begin{bmatrix}\mathbf{h}^i\\ 1\end{bmatrix}, \quad (11)$$

where $\Lambda_{\boldsymbol{\nu}_{c\cdot}} = \mathrm{diag}(\boldsymbol{\nu}_{c\cdot})$.

For TPBN shrinkage: The conditional distribution of $\boldsymbol{\nu}$ and $\boldsymbol{\varsigma}$ are

$$\nu_{ci}|\mathbf{a},\boldsymbol{\varsigma} \sim \mathcal{GIG}(\alpha_a-\frac{1}{2},\ 2\varsigma_{ci},\ a_{ci}^2), \quad (12)$$

$$\varsigma_{ci}|\boldsymbol{\nu},\kappa \sim \Gamma(\alpha_a+\beta_a,\ \nu_{ci}+\kappa). \quad (13)$$

4.3 Prediction

For an instance $\mathbf{x}^{new}$ that is unseen during the above sampling process, the posterior predictive probability of its label y^{new} can be estimated as follows:

$$P(y^{new} = c|\mathbf{x}^{new}) = \frac{1}{T}\sum\nolimits_{t=1}^{T} P(y^{new} = c|\mathbf{x}^{new}, \Theta^{(t)}),$$

$$P(y^{new} = c|\mathbf{x}^{new}, \Theta^{(t)}) = \frac{\exp\{f(\mathbf{x}^{new}; \Theta_c^{(t)})\}}{\sum_{c=1}^{C} \exp\{f(\mathbf{x}^{new}; \Theta_c^{(t)})\}},$$

where $c \in \{1, 2, ..., C\}$ is the class label, T is the number of post-convergence samples $\Theta^{(t)}$ obtained from MCMC.

We finally predict the class label of $\mathbf{x}^{new}$ as

$$y^{new} = \arg\max_{c} P(y^{new} = c|\mathbf{x}^{new}).$$

5 Further Analysis and Efficient Implementation

5.1 More Informative Prior for $\boldsymbol{\lambda}$

With (2) and (4), an improper flat prior distribution on $[0, \infty)$ is implicitly imposed on λ. Alternatively, we can impose an exponential prior $\lambda \sim \text{Exp}(\gamma_0)$ to restrict λ from taking too large values, which is beneficial to discourage $\zeta \ll 0$ for correct classifications [18]. Though not so desirable in linear case [31], such a property is important for robust kernel weights learning in our MKL model, and generally improves mixing. The corresponding new likelihood is

$$\begin{aligned} L'(y_i|\Theta) &= \prod_{c=1}^{C} \int_0^{\infty} \frac{\gamma_0 \exp\{-\gamma_0 \lambda_{ci}\}}{\sqrt{2\pi\lambda_{ci}}} \exp\left\{\frac{(\lambda_{ci} + \zeta_{ci})^2}{-2\lambda_{ci}}\right\} d\lambda_{ci} \\ &= \prod_{c=1}^{C} \frac{\gamma_0}{s} \begin{cases} \exp\{-(s+1)\zeta_{ci}\}, & \text{if } \zeta_{ci} \geq 0 \\ \exp\{(s-1)\zeta_{ci}\}, & \text{if } \zeta_{ci} < 0, \end{cases} \end{aligned}$$

where $s = \sqrt{1 + 2\gamma_0} > 1$. Note that, the new likelihood always decays faster when the training samples don't satisfy the max-margin criterion $\zeta_{ci} \geq 0$ for each binary classification sub-problem, while it also discourages $\zeta \ll 0$ for correctly classified samples. When $\gamma_0 \to 0$ (hence $s \to 1$), the hinge loss based pseudo-likelihood (2) can be recovered. When $\gamma_0 \gg 0$ (hence $s \gg 1$), $L'(y_i|\Theta)$ will behave more like the mechanism behind proximal SVM [26]. For moderate γ_0, this general likelihood is expected to benefit from both sides, thus will be adopted in our implementation.

For posterior inference, all conditionals remain the same as above, except that the conditional posterior of λ should be modified as

$$q(\lambda_{ci}|\Theta) = \mathcal{GIG}\Big(\frac{1}{2},\ 1 + 2\gamma_0,\ \zeta_{ci}^2\Big). \tag{14}$$

5.2 Correctness of the Sampler

Here we show the correctness of (7) and (8) using the recent theoretical framework in [25], where a general recipe for constructing MCMC samplers based on continuous Markov processes was developed. The recipe involves defining a (stochastic) system parameterized by two matrices: a positive semidefinite diffusion matrix, $\mathbf{D}(\mathbf{z})$, and a skew-symmetric curl matrix, $\mathbf{Q}(\mathbf{z})$, where $\mathbf{z} = (\mathbf{e}, \mathbf{r})$ with $\mathbf{e}$ model parameters of interest and $\mathbf{r}$ a set of auxiliary variables. The dynamics are then written explicitly in terms of the target stationary distribution and these two matrices. It can be verified that (7) and (8) fall within their framework when

$$\mathbf{D}(\mathbf{e}, \mathbf{r}) = \begin{bmatrix} 0, & 0 \\ 0, & \mathbf{G}(\mathbf{e})^{-1} \end{bmatrix}, \qquad \mathbf{Q}(\mathbf{e}, \mathbf{r}) = \begin{bmatrix} 0, & -\mathbf{G}(\mathbf{e})^{-1/2} \\ \mathbf{G}(\mathbf{e})^{-1/2}, & 0 \end{bmatrix}.$$

5.3 Efficient Implementation

Omitting MH correction in Riemann HMC. Inspired by the stochastic gradient HMC method [5] and the general stochastic gradient MCMC framework in [25], we can simplify the leapfrog procedure in RHMC and omit the Metropolis-Hastings correction step by using decreasing step sizes, which guarantees the sampler to yield the correct invariant distribution. However, having to decrease ϵ to zero comes at the cost of increasingly small updates. We can also use a finite, small step size in practice, resulting in a biased (but faster) sampler [25].

Efficiently Sampling. $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$: Though directly generating random variates from the GIG distribution is straightforward according to [11], it can be inefficient when the data set is very large. In fact, we can make use of the relationship[1] between GIG and inverse Gaussian (IG). For $\boldsymbol{\lambda}$, it is easy to see λ_{ci}^{-1} follows the following inverse Gaussian distribution:

$$q(\lambda_{ci}^{-1}|\Theta) = \mathcal{IG}\left(\frac{\sqrt{1+2\gamma_0}}{|\zeta_{ci}|},\ 1+2\gamma_0\right). \tag{15}$$

For $\boldsymbol{\nu}$, consider the case $\alpha_a = 1$, which leads to

$$\nu_{ci}|\mathbf{a}, \boldsymbol{\varsigma} \sim \mathcal{GIG}\left(\frac{1}{2},\ 2\varsigma_{ci},\ a_{ci}^2\right). \tag{16}$$

Consequently, we have

$$\nu_{ci}^{-1}|\mathbf{a}, \boldsymbol{\varsigma} \sim \mathcal{IG}\left(\frac{\sqrt{2\varsigma_{ci}}}{|a_{ci}|},\ \ 2\varsigma_{ci}\right). \tag{17}$$

Thus, we can sample from the corresponding IG distribution instead, which is more efficient and adopted in our implementation.

[1] Generally, we have [6]: if $\lambda \sim \mathcal{GIG}(1/2, \varrho, \chi)$, then $\lambda^{-1} \sim \mathcal{IG}(\vartheta, \varrho)$, where $\chi = \varrho/\vartheta^2$.

6 Experiments

We conduct extensive experiments on both binary and multiclass classification data sets. For each data set, we run 200 MCMC iterations of the proposed BM^{3}KL model and use the samples collected in the last 20 iterations for prediction. The code for BM^{3}KL was written purely in Matlab and all experiments were performed on a desktop with 3.2 GHz CPU and 12 GB memory.

6.1 Compared Algorithms and Parameter Settings

We compare BM^{3}KL with the following state-of-the-art kernel-based algorithms:

- A data augmentation based Bayesian nonlinear SVM model (BSVM) [18];
- Bayesian efficient MKL [15] with sparse and non-sparse kernel weights (sBEMKL and BEMKL, respectively);
- Maximum Margin Multiple Kernel (M^{3}K) learning [9] for multiclass classification;
- Efficient and accurate ℓ_p-norm MKL [20], a general max-margin MKL framework with ℓ_p-norm constraint on kernel weights. We consider $p = 1, 2, 4, \infty$ and use its Shogun C++ implementation;
- Online-batch strongly convex MKL (OBSCURE) [29], an efficient two-stage method which learns an online model for batch initialization;
- SimpleMKL [33], a well-known MKL baseline with max-margin principle.

We perform 5-fold cross-validation on training data sets to select the regularization parameter $C \in \{10^{-1}, 10^0, \ldots, 10^4\}$ for SimpleMKL and ℓ_p-norm MKL, and $C \in \{1, 10, 100, 1000\}$ and $p \in \{1.01, 1.05, 1.10, 1.25, 1.50, 1.75, 2\}$ for OBSCURE, following the instructions in their original papers. The recommended setting in the publicly available code for BEMKL and sBEMKL is used. The hyper-parameters of BM^{3}KL are fixed to $\eta = 1$, $\alpha_a = 1$, $\kappa = 10^{-10}$, $\tau = 10^{-4}$ in all experiments.

6.2 Binary Classification on Benchmark Data

In this subsection, we evaluate the performance of the proposed BM^{3}KL on a number of binary classification tasks as shown in Table 1, where five binary classification tasks were constructed from the TRECVID 2003 data set, which has five classes of 165-dimensional manually labeled video shots.

Comparison with single kernel machines. Before we systematically compare BM^{3}KL with various MKL algorithms, we first briefly demonstrate the performance improvements of our multi-kernel method over BSVM [18], which adopts a similar data augmentation idea and infers its GP kernel parameters with slice sampling. Following [18], the data sets were normalized to have zero mean and unit variance, and then randomly split into 10 folds of which one at a time was used as test set to evaluate models trained on the remaining nine folds. For MKL, we predefine a pool of 16 kernel functions on each data set, including

Table 1. The number of instances and the dimensionality for each binary classification data set.

Data set	Ionosphere	Sonar	Wisconsin	Crabs	Bupaliver	Trecvid03				
						1vs3	1vs4	3vs4	3vs5	4vs5
#Insance	351	208	683	200	345	393	340	317	303	250
#Dimension	34	60	9	7	6	165	165	165	165	165

13 Gaussian kernels with widths in $\{2^{-6}, 2^{-5}, \ldots, 2^{6}\}$ and 3 polynomial kernels with degrees in $\{1, 2, 3\}$. All kernel matrices were normalized to have unit diagonal entries (i.e., spherical normalization). Table 2 shows mean accuracy for the methods under consideration, where the results of BSVM, SVM and Gaussian Process Classification (GPC) are directly cited from [18], and the results of a latest Bayesian MKL model (BEMKL) are also listed for reference. For the proposed BM^{3}KL, we fix $\gamma_0 = 100$ and $\beta_a = 0.1$ for all data sets. The RHMC step size ϵ is set to 0.1 for Sonar, and 0.01 for the others. It is clear that BM^{3}KL can consistently outperform all other competitors, even without bothersome tuning of (hyper-) parameters.

Table 2. Comparison with single kernel learning. Listed results are mean test accuracies (%) from 10-fold cross validation.

Data set	N	BSVM	SVM	GPC	sBEMKL	BEMKL	BM^{3}KL
Ionosphere	315	94.02	94.29	92.59	93.06	92.64	**96.11**
Sonar	187	88.94	88.46	87.50	85.95	86.53	**90.68**
Wisconsin	614	97.07	96.93	97.36	97.78	97.79	**97.91**
Crabs	180	98.50	98.00	97.50	99.00	98.75	**99.50**

Comparison with various MKL algorithms. We then compare the proposed BM^{3}KL with state-of-the-art MKL algorithms. Following the experimental settings in [15], we construct Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$) and polynomial kernels with 3 different degrees ($\{1, 2, 3\}$) on all features and on each single feature. For the Bupaliver and Sonar data, we randomly select 70 % of each data set as the training set and use the remaining as the test set as in [15]. For the binary classification tasks constructed from Trecvid 2003 (see Sect. 6.3 for its details), the ratio of training/testing split is 20 % vs. 80 % because we have thousands of kernels on them and large training ratios lead to out of memory. All data sets were normalized to have zero mean and unit variance, and all base kernel matrices were normalized to have unit diagonal entries and precomputed before running the algorithm. For the proposed BM^{3}KL, we set[2] $\gamma_0 = 500$, $\epsilon = 0.1/t$ and select $\beta_a \in \{1, 2, 3\}$ via

[2] To get decreasing step sizes, we use t to denote the t-th MCMC iteration.

Table 3. Comparison of various MKL methods on binary classification tasks. Each element in the table shows the mean and standard deviation of testing accuracies/training times obtained from 20 independent trials. All experiments were conducted in Matlab, but ℓ_p-MKL and OBSCURE called C++ routines. Thus the results of training time for them may be over optimistic.

Algorithm	Metric	Bupaliver N=241, P=91	Sonar N=144, P=793	Trecvid03 1vs3 N=78, P=2158	Trecvid03 1vs4 N=68, P=2158	Trecvid03 3vs4 N=63, P=2158	Trecvid03 3vs5 N=60, P=2158	Trecvid03 4vs5 N=50, P=2158
SimpleMKL	Test-Acc (%)	70.3 ± 3.3	83.4 ± 4.7	78.8 ± 2.4	77.2 ± 2.8	70.4 ± 2.8	71.8 ± 4.0	71.6 ± 2.7
	Train-Time (s)	18.4 ± 20.1	35.3 ± 15.4	25.2 ± 17.6	22.6 ± 11.7	16.3 ± 10.5	20.8 ± 11.2	14.4 ± 8.9
sBEMKL	Test-Acc (%)	68.4 ± 4.3	76.9 ± 4.0	77.7 ± 2.3	75.6 ± 3.5	69.6 ± 4.7	69.0 ± 3.7	71.3 ± 3.9
	Train-Time (s)	2.8 ± 0.1	17.0 ± 0.3	131.1 ± 0.8	129.9 ± 0.4	129.2 ± 0.4	128.7 ± 0.3	127.1 ± 0.3
BEMKL	Test-Acc (%)	70.7 ± 3.8	82.8 ± 4.1	79.4 ± 2.3	78.5 ± 2.9	71.8 ± 2.6	73.8 ± 3.1	73.6 ± 4.5
	Train-Time (s)	2.9 ± 0.1	16.9 ± 0.2	130.8 ± 0.7	129.7 ± 0.5	129.2 ± 0.4	128.8 ± 0.4	126.9 ± 0.2
ℓ_1-MKL	Test-Acc (%)	68.8 ± 4.2	82.7 ± 3.4	78.8 ± 3.0	77.0 ± 1.6	71.1 ± 4.1	71.8 ± 3.1	70.4 ± 2.8
	Train-Time (s)	2.0 ± 0.7	22.1 ± 3.4	34.7 ± 14.1	30.4 ± 5.0	25.4 ± 6.0	30.3 ± 14.9	14.5 ± 1.8
ℓ_2-MKL	Test-Acc (%)	70.6 ± 4.3	83.3 ± 3.9	79.5 ± 3.4	79.6 ± 1.6	72.1 ± 3.8	74.5 ± 1.5	71.0 ± 3.5
	Train-Time (s)	**0.6 ± 0.0**	6.2 ± 0.7	7.5 ± 2.5	6.2 ± 0.4	6.6 ± 0.4	6.2 ± 0.2	3.8 ± 0.1
ℓ_4-MKL	Test-Acc (%)	71.0 ± 4.0	83.6 ± 3.5	81.1 ± 3.3	80.1 ± 2.1	72.9 ± 3.7	74.1 ± 1.5	71.3 ± 3.0
	Train-Time (s)	1.2 ± 0.2	8.0 ± 0.7	5.6 ± 0.7	4.4 ± 0.6	5.2 ± 0.7	4.5 ± 0.2	2.6 ± 0.1
ℓ_∞-MKL	Test-Acc (%)	70.5 ± 3.7	83.5 ± 3.1	80.8 ± 3.7	80.4 ± 1.9	73.7 ± 3.4	74.9 ± 1.8	70.5 ± 2.9
	Train-Time (s)	1.1 ± 0.1	6.4 ± 0.5	5.6 ± 1.0	4.2 ± 0.9	4.9 ± 0.6	4.3 ± 0.2	2.8 ± 0.1
OBSCURE	Test-Acc (%)	69.5 ± 4.7	83.4 ± 3.3	81.0 ± 2.2	77.8 ± 2.4	71.0 ± 3.8	72.4 ± 3.0	69.6 ± 2.8
	Train-Time (s)	3.3 ± 0.2	13.5 ± 0.6	4.9 ± 0.3	4.6 ± 0.2	4.4 ± 0.3	3.9 ± 0.1	2.9 ± 0.0
BM^{3}KL	Test-Acc (%)	**71.6 ± 3.5**	**84.3 ± 3.8**	**81.5 ± 1.7**	**81.1 ± 2.8**	**74.0 ± 2.4**	**75.7 ± 2.5**	**75.2 ± 2.9**
	Train-Time (s)	3.1 ± 0.1	**5.6 ± 0.1**	**3.4 ± 0.0**	**2.6 ± 0.0**	**2.4 ± 0.0**	**2.2 ± 0.0**	**1.5 ± 0.0**

5-fold cross-validation for each data set. To get stable results, we independently repeat the random split of each data set, and then run each algorithm on it, for 20 times. The mean and standard deviation are reported in Table 3 in terms of testing accuracy and training time.

The superiority of BM^{3}KL over the competitors is evident. When comparing BM^{3}KL with BEMKL, it is easy to see significant improvements of the former in terms of prediction performance. We attribute this to the max-margin principle underlying our model and its more accurate MCMC-based inference rather than variational approximation with mean-field assumption. As to the computational complexity, BM^{3}KL and BEMKL scale linearly and cubically with the number of kernels P respectively, while they both scale cubically with the number of training samples N. Consequently, as we observed, BEMKL needs considerable more training time when P is large (e.g., on Trecvid03).

Another thing worth to mention is that, with a simple and fixed Dirichlet prior on kernel weights, BM^{3}KL can consistently outperform ℓ_p-MKL and OBSCURE. This indicates that the superiority of our method is not brought in by solely regularizing the weights. We believe this is owing to the inherent advantages of our Bayesian max-margin modeling, and the Riemann manifold based HMC method.

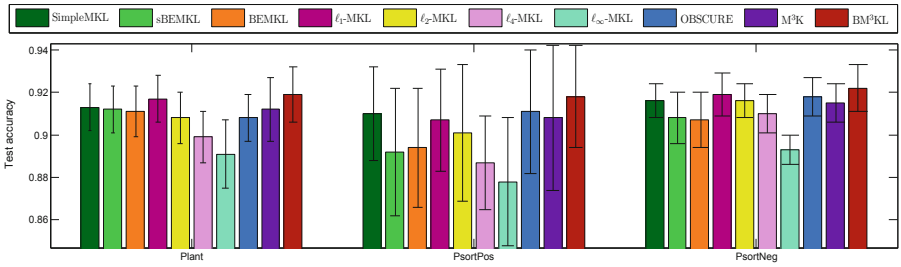


Fig. 1. Multiclass classification performance comparison on protein subcellular localization data sets. All results are averaged over 10 independent trials.

6.3 Multiclass Classification

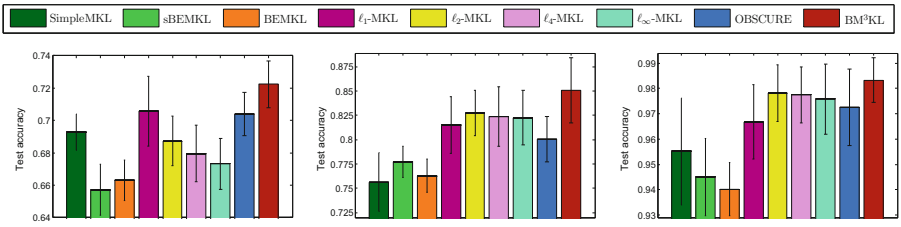
Protein Subcellular Localization. We first consider three biological data sets (Plant, PsortPos, and PsortNeg) that have been widely used to compare MKL algorithms. The numbers of data instances in Plant, PsortPos, and PsortNeg are 940, 541, and 1444, respectively, while the numbers of data classes are 4, 4, and 5, respectively. For each of these three data sets, 69 biologically motivated sequence kernels [40] are used and the corresponding base kernel matrices are provided online[3]. To replicate the experiments of a latest multiclass max-margin multiple kernel classification model (M^{3}K) [9], all kernel matrices were first centered and then normalized to have unit diagonal entries, and the training split fractions for Plant, PsortPos, and PsortNeg were set as 0.5, 0.8, and 0.65, respectively. As above, we set $\gamma_0 = 500$, $\epsilon = 0.1/t$, and select $\beta_a \in \{1, 2, 3\}$ via 5-fold cross-validation. Then we independently repeat the random split of each data set and run each algorithm on it, for 10 times. The mean testing accuracies and the standard deviations are shown in Fig. 1, where the results of M^{3}K are directly cited from [9]. From the results we can see, (1) BM^{3}KL achieves highest mean accuracy on each data set; (2) sometimes the performance differences are not so significant since the training sample ratios are large enough; (3) BM^{3}KL has advantage in terms of robustness.

Video shots classification. The TRECVID 2003 data set has 1078 manually labeled video shots which are categorized into 5 classes. Each of the video shots is represented by a 165-dimensional vector of HSV color histogram. To fully exploit the rich features underlying each sample, we define a large set of kernel mappings on all features and each individual feature, and then learn the optimal combination of them. Specifically, we construct Gaussian kernels with 13 different widths ($\{2^{-6}, 2^{-5}, \ldots, 2^{6}\}$) and polynomial kernels with 3 different degrees ($\{1, 2, 3\}$) on the 165-dimensional vectors, and construct Gaussian kernels with 7 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^{3}\}$) on each single feature. This gives us 1171 kernels in total (to avoid out of memory, we didn't consider even more kernels).

[3] http://raetschlab.org//suppl/protsubloc.

As in [12,39], the entire data set was evenly split into training and testing sets. We independently repeat the random split and run each algorithm on it, for 10 times. Here $\gamma_0 = 500$, $\epsilon = 0.1/t$, and $\beta_a = 0.5$. The mean testing accuracies and the standard deviations are shown in Fig. 2(a), from which we can observe significant advantage of BM^{3}KL. Note that, these results are also significantly better than those reported in [12,39], where max-margin supervised subspace learning was studied.

Face recognition. The ORL data set contains 10 different face images for each of 40 distinct subjects. For some subjects, the images were taken at different times with varying lighting and facial details (open/closed eyes, smiling/not smiling, glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position, and were manually aligned, cropped and resized to 32×32 pixels. We construct Gaussian kernels with 13 different widths ($\{2^{-6}, 2^{-5}, \ldots, 2^6\}$) and polynomial kernels with 3 different degrees ($\{1, 2, 3\}$) on the 1024-dimensional pixel vectors, and construct Gaussian kernels with 2 different widths ($\{2^1, 2^2\}$) and polynomial kernel with degree 2 on each single pixel. This gives us 3088 kernels in total (to avoid out of memory, we didn't consider even more kernels). For the training sample ratio, we consider two cases, i.e., 0.2 and 0.7 (accordingly, we have 80 and 280 training samples respectively). We independently repeat the random splits and run each algorithm on them, for 10 times. Here $\gamma_0 = 10000$, $\epsilon = 0.01$, and $\beta_a = 0.1$. The mean testing accuracies and the standard deviations are shown in Fig. 2(b) and (c), from which we see obvious advantages of BM^{3}KL again.

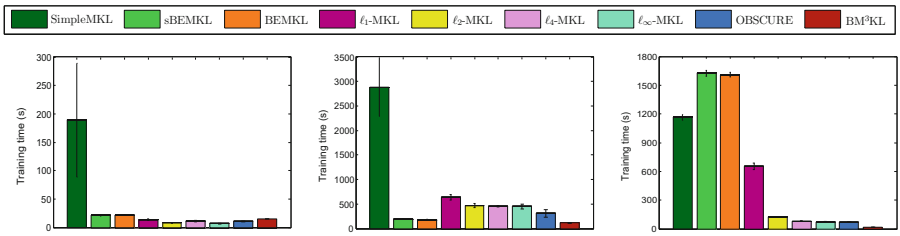


(a) TRECVID 2003 (b) ORL, $N = 80$ (c) ORL, $N = 280$

Fig. 2. Multiclass classification performance comparison on vision data sets. All results are averaged over 10 independent trials.

6.4 Time Efficiency and Convergence Rate

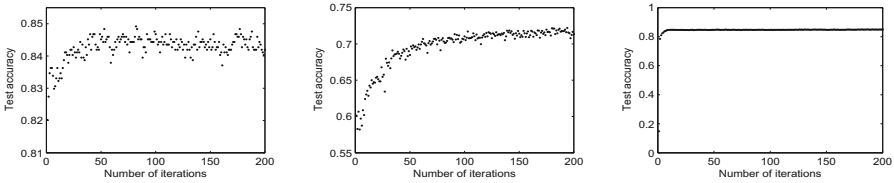
As shown in Table 3 and Fig. 3, the training time efficiency of the proposed BM^{3}KL is either significantly better than or comparable with the state-of-the-art MKL algorithms on all considered data sets. Note that here we couldn't directly compare with M^{3}K since its code is not publicly available. But as shown in [9] it generally needs several times more training time than OBSCURE.



(a) Plant (b) TRECVID 2003 (c) ORL, $N = 80$

Fig. 3. Training time efficiency comparison on multiclass classification data sets. All results are averaged over 10 independent trials conducted in Matlab, but ℓ_p-MKL and OBSCURE called C++ routines. Thus the results for them may be over optimistic.

Specifically, it is observed that BM^{3}KL generally is more efficient than BEMKL, a latest Bayesian MKL model focusing on efficient inference [15]. The reason is that BEMKL has to perform matrix inversion to compute the posterior covariance of kernel weights in each round of iteration while it is avoided in BM^{3}KL via employing Riemann HMC. Besides, enjoying the Bayesian modelling advantages, BM^{3}KL achieves even faster learning speed than the optimization-based point estimate methods. We attribute this to the fast convergence rate of our geometry and local conjugacy based approximate posterior sampling, which is depicted in Fig. 4.



(a) Sonar (b) TRECVID 2003 (c) ORL, $N = 80$

Fig. 4. Convergence rate of the approximate posterior sampling of BM^{3}KL on Sonar, TRECVID 2003 and ORL ($N = 80$). All settings are the same as above except that the single sample of model weights obtained in each MCMC iteration is used for prediction on each data set.

7 Conclusion and Future Work

By defining a multiclass (pseudo-) likelihood function that accounts for the margin loss for kernelized classification, we have developed a robust Bayesian max-margin MKL framework with Dirichlet and TPBN priors imposed on the kernel and sample weights respectively. Employing Riemann manifold HMC to sample

from the conditional posterior of kernel weights, and making use of local conjugacy for all other variables, an efficient MCMC algorithm in the augmented variable space is devised. Extensive experiments on both binary and multiclass data sets show that the proposed classification model not only outperforms a number of competitors consistently but also requires substantially fewer training time when the number of kernels is large. In future, we plan to apply our framework to multi-kernel regression analysis.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (No. 61473273, 61573335, 91546122, 61303059), Guangdong provincial science and technology plan projects (No. 2015B010109005), and the Science and Technology Funds of Guiyang (No. 201410012).

References

1. Andrews, D.F., Mallows, C.L.: Scale mixtures of normal distributions. J. Roy. Stat. Soc. B (Methodol.) **36**(1), 99–102 (1974)
2. Armagan, A., Clyde, M., Dunson, D.B.: Generalized beta mixtures of gaussians. In: NIPS (2011)
3. Bach, F.R.: Consistency of the group lasso and multiple kernel learning. JMLR **9**, 1179–1225 (2008)
4. Bach, F.R., Lanckriet, G.R., Jordan, M.I.: Multiple kernel learning, conic duality, and the smo algorithm. In: ICML (2004)
5. Chen, T., Fox, E., Guestrin, C.: Stochastic gradient Hamiltonian Monte Carlo. In: ICML (2014)
6. Chhikara, R.: The Inverse Gaussian Distribution: Theory, Methodology, and Applications, vol. 95. CRC Press, Boca Raton (1988)
7. Christoudias, M., Urtasun, R., Darrell, T.: Bayesian localized multiple kernel learning. Univ. California Berkeley, Berkeley (2009)
8. Cortes, C.: Invited talk: can learning kernels help performance? In: ICML (2009)
9. Cortes, C., Mohri, M., Rostamizadeh, A.: Multi-class classification with maximum margin multiple kernel. In: ICML, pp. 46–54 (2013)
10. Damoulas, T., Girolami, M.A.: Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection. Bioinformatics **24**(10), 1264–1270 (2008)
11. Devroye, L.: Random variate generation for the generalized inverse gaussian distribution. Stat. Comput. **24**(2), 239–246 (2014)
12. Du, C., Zhe, S., Zhuang, F., Qi, Y., He, Q., Shi, Z.: Bayesian maximum margin principal component analysis. In: AAAI (2015)
13. Girolami, M., Calderhead, B.: Riemann manifold langevin and hamiltonian monte carlo methods. J. Roy. Stat. Soc. B (Stat. Methodol.) **73**(2), 123–214 (2011)
14. Girolami, M., Rogers, S.: Hierarchic bayesian models for kernel learning. In: ICML (2005)
15. Gönen, M.: Bayesian efficient multiple kernel learning. In: ICML, pp. 1–8 (2012)
16. Gönen, M., Alpaydin, E.: Localized multiple kernel learning. In: ICML, pp. 352–359 (2008)
17. Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. JMLR **12**, 2211–2268 (2011)

18. Henao, R., Yuan, X., Carin, L.: Bayesian nonlinear support vector machines and discriminative factor modeling. In: NIPS, pp. 1754–1762 (2014)
19. Jain, A., Vishwanathan, S.V., Varma, M.: Spg-gmkl: generalized multiple kernel learning with a million kernels. In: SIGKDD (2012)
20. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: Lp-norm multiple kernel learning. JMLR **12**, 953–997 (2011)
21. Lanckriet, G.R., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. JMLR **5**, 27–72 (2004)
22. Lázaro-gredilla, M., Titsias, M.K.: Spike and slab variational inference for multi-task and multiple kernel learning. In: NIPS, pp. 2339–2347 (2011)
23. Liu, X., Wang, L., Yin, J., Dou, Y., Zhang, J.: Absent multiple kernel learning. In: AAAI (2015)
24. Liu, X., Wang, L., Zhang, J., Yin, J.: Sample-adaptive multiple kernel learning. In: AAAI (2014)
25. Ma, Y.A., Chen, T., Fox, E.: A complete recipe for stochastic gradient mcmc. In: NIPS (2015)
26. Mangasarian, O.L., Wild, E.W.: Proximal support vector machine classifiers. In: SIGKDD (2001)
27. Neal, R.M.: Mcmc using hamiltonian dynamics. Handbook of Markov Chain Monte Carlo, vol. 2 (2011)
28. Ni, B., Li, T., Moulin, P.: Beta process multiple kernel learning. In: CVPR (2014)
29. Orabona, F., Jie, L., Caputo, B.: Multi kernel learning with online-batch optimization. JMLR **13**(1), 227–253 (2012)
30. Patterson, S., Teh, Y.W.: Stochastic gradient riemannian langevin dynamics on the probability simplex. In: NIPS, pp. 3102–3110 (2013)
31. Polson, N.G., Scott, S.L.: Data augmentation for support vector machines. Bayesian Analysis **6**(1), 1–23 (2011)
32. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More efficiency in multiple kernel learning. In: ICML, pp. 775–782 (2007)
33. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: Simplemkl. JMLR **9**, 2491–2521 (2008)
34. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. JMLR **7**, 1531–1565 (2006)
35. Suzuki, T., Tomioka, R.: Spicymkl: a fast algorithm for multiple kernel learning with thousands of kernels. Machine learning **85**(1–2), 77–108 (2011)
36. Xu, Z., Jin, R., King, I., Lyu, M.: An extended level method for efficient multiple kernel learning. In: NIPS, pp. 1825–1832 (2009)
37. Xu, Z., Jin, R., Yang, H., King, I., Lyu, M.R.: Simple and efficient multiple kernel learning by group lasso. In: ICML (2010)
38. Yang, J., Li, Y., Tian, Y., Duan, L., Gao, W.: Group-sensitive multiple kernel learning for object categorization. In: CVPR (2009)
39. Zhu, J., Chen, N., Xing, E.P.: Bayesian inference with posterior regularization and applications to infinite latent svms. JMLR **15**, 1799–1847 (2014)
40. Zien, A., Ong, C.S.: Multiclass multiple kernel learning. In: ICML, pp. 1191–1198 (2007)