Beta Process Multiple Kernel Learning

Bingbing Ni
Advanced Digital Sciences Center
Singapore 138632
bingbing.ni@adsc.com.sg

Teng Li
Anhui University
Hefei 230039, P. R. China
tenglwy@gmail.com

Pierre Moulin
UIUC
IL 61820-5711 USA
moulin@ifp.uiuc.edu

Abstract

In kernel based learning, the kernel trick transforms the original representation of a feature instance into a vector of similarities with the training feature instances, known as kernel representation. However, feature instances are sometimes ambiguous and the kernel representation calculated based on them do not possess any discriminative information, which can eventually harm the trained classifier. To address this issue, we propose to automatically select good *feature instances when calculating the kernel representation in multiple kernel learning. Specifically, for the kernel representation calculated for each input feature instance, we multiply it element-wise with a latent binary vector named as instance selection variables, which targets at selecting good instances and attenuate the effect of ambiguous ones in the resulting new kernel representation. Beta process is employed for generating the prior distribution for the latent instance selection variables. We then propose a Bayesian graphical model which integrates both MKL learning and inference for the distribution of the latent instance selection variables. Variational inference is derived for model learning under a max-margin principle. Our method is called Beta process multiple kernel learning. Extensive experiments demonstrate the effectiveness of our method on instance selection and its high discriminative capability for various classification problems in vision.*

1. Introduction

In kernel based learning, a data sample $\mathbf{x}_\star$ is mapped from its original feature space onto a new space consisting of its similarities with all training samples: $\mathbf{k}_\star = [k(\mathbf{x}_1, \mathbf{x}_\star), \cdots, k(\mathbf{x}_N, \mathbf{x}_\star)]^T$, where $k(,)$ denotes the similarity (kernel) function and we assume N training samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. This new representation is used to predict the target output f of an unseen testing sample as $\mathbf{x}_\star$ by $f(\mathbf{x}_\star) = \langle \mathbf{a}, \mathbf{k}_\star \rangle + b$ ($\langle , \rangle$ denotes dot product). We call $\mathbf{k}_\star$ the *kernel representation* for sample $\mathbf{x}_\star$ and $\mathbf{a}$ is the N-dimensional model coefficient vector to learn.

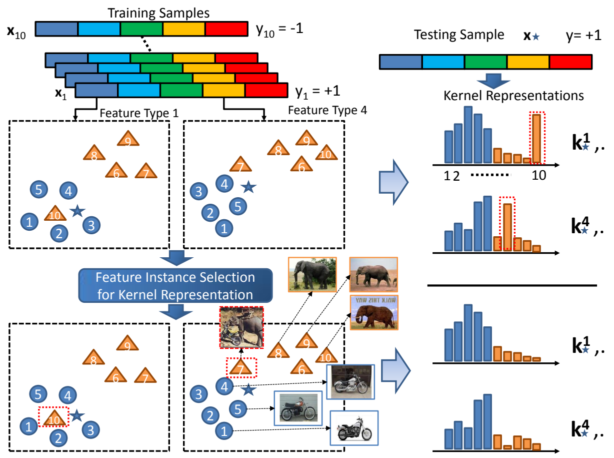

Figure 1. Motivation of the proposed Beta Process multiple kernel learning framework (BPMKL). In this toy example, there are 10 data samples with 5 positive (blue circle) and 5 negative (yellow triangle) samples. Each sample is described by 5 types of features and we calculate 5 kernel matrices based on them. Due to some ambiguous feature instances, *i.e.*, sample 10 on feature type 1 and sample 7 on feature type 4 (ambiguous visual feature with both elephant and motorcycle), the resulting kernel representations (for testing sample $\mathbf{x}_\star$) have strong similarity scores on these instances, which harms the trained classifier. In contrast, BPMKL can select *good* instances for calculating the kernel representations and therefore improves the classification performance.

In most real applications, a single type of feature is too *weak* to represent a data sample. *Weakness* can be attributed to the fact that a large portion of features coming from different classes share similar values and are thus ambiguous. To be more discriminative, we always use multiple types of features, *i.e.*, $\mathbf{x} = [\mathbf{x}^1, \cdots, \mathbf{x}^P]^T$, to represent a data sample $\mathbf{x}$. We assume P types of features and each $\mathbf{x}^m, m = 1, \cdots, P$ is called a feature instance. For example, we usually extract various types of features including color, shape, and texture etc., to describe an image. Multiple kernel learning (MKL) [1, 9, 19, 28, 21, 29, 8, 5] is proposed to combine different types of features to boost the classification performance. In particular, for an input data sample $\mathbf{x} = [\mathbf{x}_\star^1, \cdots, \mathbf{x}_\star^P]^T$, its kernel representations based on different feature types $\{\mathbf{k}_\star^1, \cdots, \mathbf{k}_\star^P\}$ (*i.e.*, $\mathbf{k}_\star^m =$

$[k(\mathbf{x}_1^m, \mathbf{x}_\star^m), \cdots, k(\mathbf{x}_N^m, \mathbf{x}_\star^m)]^T$) are linearly combined to predict the target value as: $\mathbf{k}_\star = \sum_{m=1}^P e_m \mathbf{k}_\star^m, \quad f(\mathbf{x}_\star) = \langle \mathbf{a}, \mathbf{k}_\star \rangle + b$, where $\mathbf{e} = [e_1, \cdots, e_P]^T$ is the kernel combination vector.

On top of multiple kernel combination, in this work we explore the idea whether we can perform **feature instance selection** in calculating kernel representation and how it could help to improve the trained classifier. Our intuitions are as follows. On the one hand, kernel representations based on the similarities measured with the ambiguous feature instances do not possess discriminative information and they can harm the trained classifier. If we can automatically discover these ambiguous feature instances and attenuate their effects during training, the obtained classifier can be more discriminative. On the other hand, de-selecting some feature instances in computing the kernel representation will not loose too much useful information and degrade the representation power since multiple feature types complement each other and the corresponding multiple kernel representations convey rich descriptive information. Our motivation is visualized in Figure 1. To the best of our knowledge, there exist few works in MKL literature that explores this idea.

This motivates us to propose a probabilistic MKL framework which can simultaneously perform *good* feature instance selection in kernel representation and classifier learning. Specifically, for the kernel representation calculated for each input feature instance, we multiply it element-wise with a latent binary vector named as instance selection variables, which targets at selecting good instances and attenuate the effect of ambiguous ones in the resulting new kernel representation. The objective is then to capture the distribution of these latent indicator variables during MKL training. To this end, Beta process is employed for generating the prior distribution for the introduced latent instance selection variables. We then integrates both MKL learning and inference for the distribution of the latent instance selection variables, within a Bayesian graphical model framework. Variational inference is derived for model learning under a max-margin principle. Our method is called Beta process multiple kernel learning (BPMKL). Qualitative and quantitative evaluations on a synthetic data, UCL toy datasets, two image classification benchmarks and an action recognition video benchmark demonstrate that 1) the inferred distribution of latent instance selection variables well select *unambiguous* feature instances in kernel representation by the proposed method, and 2) it possesses high discriminative capability for various classification problems in vision applications.

2. Related Works

Bach et al. [1] formulated the multiple kernel learning problem as a second-order cone programming (SCOP) problem. Lanckriet et al. [9] formulated it as a semi-definite programming (SDP) problem, and Gehler and Nowozin [4] proposed a boosting-type MKL algorithm. To cope with medium and large-scale problem, MKL is re-formulated into different types of optimization problems including semi-infinite linear programming (SILP) by Sonnenburg et al. [19], sub-gradient descent (SD) by Rakotomamonjy et al. [17], extension of the level method by Xu et al. [28]. Instead of sparse kernel selection, Varma and Babu [21] proposed a generalized MKL algorithm that can use any differentiable and continuous regularization term on the kernel weights. Xu et al. [29], Kloft et al. [8] and Vishwanathan et al. [22] presented efficient MKL algorithms with the ℓ_p-norm on the kernel weights. Gönen [5] recently formulated a very efficient MKL method based on fully factorized variational approximation. However, none of these previous works on MKL have explored the idea of feature instance selection for a more discriminative kernel representation. Wang et al. demonstrated that multiple features can be integrated by embedding sample relationships in graphs [26] [27]. Wang et al. [23] proposed to learning image similarities using fast kernel machines with few training samples.

Beta process prior [7] has been successfully applied in the problem of factor analysis [15]. Beta process was also employed as a prior for learning the dictionary in sparse image representation, which was applied in denoising, inpainting and compressive sensing (CS) [31]. Recently, Mittelman et al. [14] developed a extension restricted Boltzmann machine (RBM) by incorporating a Beta-Bernoulli process prior for image mid-level feature learning. However, Beta process has not been explored in the context of multiple kernel learning. There exist a large number of works on feature selection. However, instead of selecting good features, the proposed work selects good instances for calculating kernel representation.

3. Methodology

We introduce the notations used throughout the paper as follows. Assume we have N training samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ and each sample j is described by P types of features, *i.e.,* $\mathbf{x}_j = [\mathbf{x}_j^1, \cdots, \mathbf{x}_j^P]^T$ (each $\mathbf{x}_j^m$ is called a feature instance). The $N \times N$ kernel matrix based on the m-th feature type is denoted as $\mathbf{K}^m = [k_{j,i}^m]_{j,i=1,\cdots,N}$, where $k_{j,i}^m = k(\mathbf{x}_j^m, \mathbf{x}_i^m)$ and $\mathbf{K} = \{\mathbf{K}^1, \cdots, \mathbf{K}^P\}$. Note that we use j, i, and m to index training sample (column in kernel matrix), reference training sample (same as training sample, row in kernel matrix) and feature type. The kernel representation for the m-th feature instance of the j-th sample corresponds to the j-th column of $\mathbf{K}^m$, which is denoted as $\mathbf{k}_{j,\cdot}^m$. The associated class labels for all training samples are represented as a N-dimensional vector $\mathbf{y}$, where each element $y_j \in \{-1, +1\}, j = 1, \cdots, N$.

3.1. Generative Process and Max-Margin Learning

As shown in Figure 1, the kernel representation for the feature instance $\mathbf{x}_\star$ consists of its similarities with respect to all training instances. However, we can note that some training instances from positive and negative classes are ambiguous, namely, 1) some instances from both classes are very close in feature space (ambiguity, e.g., the image with both *elephant* and *motorcycle* is visually similar to both classes); and 2) some instances locate within the cluster of instances from the other class (outlier). It directly follows that the kernel representation based on these *bad* feature instances will harm the trained classifier. Our idea is thus to select *good* feature instances for enhance the discriminative power of the kernel representation.

To this end, for each kernel representation $\mathbf{k}^m_{j,.}, j = 1, \cdots, N; m = 1, \cdots, P$, we introduce a N-dimensional vector of binary multiplier $\mathbf{z}^m_{j,.} = [z^m_{j,1}, \cdots, z^m_{j,N}]^T$. For calculating the kernel representation for the m-th instance of data sample j, element-wise product between $\mathbf{k}^m_{j,.}$ and $\mathbf{z}^m_{j,.}$ (i.e., $\mathbf{k}^m_{j,.} \circ \mathbf{z}^m_{j,.}$, where $\circ$ denotes element-wise multiplication) can select (if $z^m_{j,i} = 1$ for some i) or de-select (if $z^m_{j,i} = 0$ for some i) the corresponding reference instance. We denote $\mathbf{Z} = \{\mathbf{Z}^1, \cdots, \mathbf{Z}^P\}$, $\mathbf{Z}^m = [\mathbf{z}^m_{1,.}, \cdots, \mathbf{z}^m_{N,.}]$ the set of **instance selection variables**. We note that $\mathbf{Z}$ are latent variables and we assume that the prior distribution of binary vectors $\mathbf{z}^m_{j,.}, j = 1, \cdots, N$ is sampled from a Beta process (BP) [7, 31]. The BP with parameters $a_{\pi0}$, $b_{\pi0}$, and base measure H_0, is represented as $H \sim \mathrm{BP(a_{\pi0}, b_{\pi0}, H_0)}$. Using BP, for each feature type m, we can draw the vector of prior probabilities $\boldsymbol{\pi}^m = [\pi^m_1, \cdots, \pi^m_i, \cdots, \pi^m_N]^T$ (and we denote $\mathbf{\Pi} = \{\boldsymbol{\pi}^1, \cdots, \boldsymbol{\pi}^P\}$) for selecting instance i ($i = 1, \cdots, N$) in the kernel representation as:

$$\begin{aligned} H(\mathbf{x}^m) &= \sum_{1}^{N} \pi_i \delta_{\mathbf{x}^m}(\mathbf{x}^m_i), \\ \pi^m_i &\sim \mathrm{Beta(a_{\pi0}/N, b_{\pi0}(N-1)/N)}, \end{aligned} \tag{1}$$

where $\delta_{\mathbf{x}}()$ is a Dirac-Delta function with non-zero value at point $\mathbf{x}$. Note that H is a valid measure as $N \to \infty$. For our problem, this value is truncated to the number of training samples. We assume that training feature instances $\{\mathbf{x}^m_1, \cdots, \mathbf{x}^m_N\}$ are generated from the base distribution H_0. Each $z^m_{j,i}, j = 1, \cdots, N$ is then sampled from the Bernoulli distribution $\mathrm{Bernoulli(\pi^m_i)}$. We then use the new kernel representation vector after instance selection (i.e., $\mathbf{k}^m_{j,.} \circ \mathbf{z}^m_{j,.}$) as input to the MKL learning framework. The class prediction function can therefore be expressed as:

$$f_j = \mathbf{a}^T(\sum_{m=1}^{P} e_m \mathbf{k}^m_{j,.} \circ \mathbf{z}^m_{j,.}) + b, \forall j = 1, \cdots, N. \tag{2}$$

The generative process of the proposed probabilistic model for integrating MKL parameters and latent feature instance selection variables is as follows:

$$\begin{aligned} &\lambda_i \sim \mathcal{G}(\lambda_i; \alpha_{\lambda0}, \beta_{\lambda0}), \quad a_i \sim \mathcal{N}(a_i; 0, \lambda_i^{-1}), \quad \forall i, \\ &\omega_m \sim \mathcal{G}(\omega_m; \alpha_{\omega0}, \beta_{\omega0}), \quad e_m \sim \mathcal{N}(e_m; 0, \omega_m^{-1}) \quad \forall m, \\ &\pi^m_i \sim \mathrm{Beta(\pi^m_i; \alpha_{\pi0}, \beta_{\pi0})}, \quad \forall \mathrm{i} \quad \forall \mathrm{m}, \\ &z^m_{j,i} \sim \mathrm{Bernoulli(z^m_{j,i}; \pi^m_i)}, \quad \forall \mathrm{j}, \mathrm{i} \quad \forall \mathrm{m}. \end{aligned} \tag{3}$$

Here $\mathcal{G}(.)$, $\mathcal{N}()$, $\mathrm{Beta}()$, and $\mathrm{Bernoulli}()$ denote Gamma, Gaussian, Beta and Bernoulli distributions, respectively. $\alpha_{\pi0} = a_{\pi0}/N$ and $\beta_{\pi0} = b_{\pi0}(N-1)/N$. $(\alpha_{\lambda0}, \beta_{\lambda0}, \alpha_{\pi0}, \beta_{\pi0}, \alpha_{\omega0}, \beta_{\omega0})$ are the hyper-parameters.

The rationale underlying our model learning is: by minimizing an integrated objective function, we aim to find the underlying distributions for both latent instance selection variables $\mathbf{Z}$ and the multiple kernel parameters $\mathbf{a}$ and $\mathbf{e}$. On the one hand, we can predict accurately on unseen data with a sufficient margin, and on the other hand, we can explain the data well (i.e., capable of automatically discovering *good* and *bad* feature instances). To this end, we propose a fully factorized variational distribution $q(\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{\Pi}, \mathbf{e}, \mathbf{a}, \mathbf{Z})$ to approximate the joint distribution of proposed graphical model, *i.e.*, $p(\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{\Pi}, \mathbf{e}, \mathbf{a}, \mathbf{Z}|\mathbf{K}, \mathbf{y}, b)$. (We omit hyper-parameters for notational simplicity). We further assume the approximate distribution q can be fully factorized as:

$$q(\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{\Pi}, \mathbf{e}, \mathbf{a}, \mathbf{Z}) = q(\boldsymbol{\lambda})q(\boldsymbol{\omega})q(\mathbf{\Pi})q(\mathbf{e})q(\mathbf{a})q(\mathbf{Z}), \tag{4}$$

with the factors given by:

$$\begin{aligned} q(\boldsymbol{\lambda}) &= \prod_{i=1}^{N} q(\lambda_i) = \prod_{i=1}^{N} \mathcal{G}(\lambda_i; \alpha_{\lambda,i}, \beta_{\lambda,i}), \\ q(\mathbf{a}) &= \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}_a, \boldsymbol{\sigma}_a) = \prod_{i=1}^{N} \mathcal{N}(a_i; \mu_{a,i}, \sigma_{a,i}), \\ q(\boldsymbol{\omega}) &= \prod_{m=1}^{P} q(\omega_m) = \prod_{m=1}^{P} \mathcal{G}(\omega_m; \alpha_{\omega,m}, \beta_{\omega,m}), \\ q(\mathbf{e}) &= \mathcal{N}(\mathbf{e}; \boldsymbol{\mu}_e, \boldsymbol{\sigma}_e) = \prod_{m=1}^{P} \mathcal{N}(e_m; \mu_{e,m}, \sigma_{e,m}), \\ q(\mathbf{\Pi}) &= \prod_{i,m} q(\pi^m_i) = \prod_{i,m} \mathrm{Beta(\pi^m_i; \alpha^m_{\pi,i}, \beta^m_{\pi,i})}, \\ q(\mathbf{Z}) &= \prod_{j,i,m} q(z^m_{j,i}) = \prod_{j,i,m} \mathrm{Bernoulli(z^m_{j,i}; \phi^m_{j,i})}. \end{aligned} \tag{5}$$

The objective of variational approximation is to minimize the KL-divergence $\mathcal{KL}(q \parallel p)$, which is equivalent to maximize an upper bound $L(q) = E_q(\log p) - E_q(\log q)$. Thus, similar as in [32], the integrated learning problem using the max-margin principle be can expressed as:

$$\begin{aligned}\min_{q,\boldsymbol{\xi},b} & \; -E_q(\log p) + E_q(\log q) + \mathcal{C}\sum_j \xi_j \\ s.t. & \;\; y_j E_q(f_j) \geq 1 - \xi_j, \quad \xi_j \geq 0, \quad \forall j, \\ & \; f_j = \mathbf{a}^T(\sum_{m=1}^{P} e_m \mathbf{k}_{j,.}^m \circ \mathbf{z}_{j,.}^m) + b, \end{aligned} \tag{6}$$

where $\mathcal{C}$ is the penalty factor and we set $\mathcal{C} = 1000$ in this work. Note that like in [32], the constraints of Eqn. (6) are in the forms of expectation under the variational distribution q. The optimization task it to estimate the variational parameters $\boldsymbol{\alpha}_\lambda = (\alpha_{\lambda,1}, \cdots, \alpha_{\lambda,N})^T$, $\boldsymbol{\beta}_\lambda = (\beta_{\lambda,1}, \cdots, \beta_{\lambda,N})^T$, $\boldsymbol{\mu}_a = (\mu_{a,1}, \cdots, \mu_{a,N})^T$, $\boldsymbol{\sigma}_a = (\sigma_{a,1}, \cdots, \sigma_{a,N})^T$, $\boldsymbol{\alpha}_\omega = (\alpha_{\omega,1}, \cdots, \alpha_{\omega,P})^T$, $\boldsymbol{\beta}_\omega = (\beta_{\omega,1}, \cdots, \beta_{\omega,P})^T$, $\boldsymbol{\mu}_e = (\mu_{e,1}, \cdots, \mu_{e,P})^T$, $\boldsymbol{\sigma}_e = (\sigma_{e,1}, \cdots, \sigma_{e,P})^T$, $\boldsymbol{\alpha}_\pi = [\alpha_{\pi,i}^m]_{i=1,\cdots,N}^{m=1,\cdots,P}$, $\boldsymbol{\beta}_\pi = [\beta_{\pi,i}^m]_{i=1,\cdots,N}^{m=1,\cdots,P}$, $\boldsymbol{\Phi} = [\phi_{j,i}^m]_{j,i=1,\cdots,N}^{m=1,\cdots,P}$. To estimate these variational parameters, we develop an EM-like algorithm, which iterates the following two steps. In E-step, we infer the posterior distributions of the latent instance selection variables ($q(\mathbf{Z})$ and $q(\boldsymbol{\Pi})$); In M-step, we infer the posterior distributions of the MKL model parameters ($q(\boldsymbol{\lambda})$, $q(\mathbf{a})$, $q(\boldsymbol{\omega})$, and $q(\mathbf{e})$).

3.2. Inference for $q(\boldsymbol{\lambda})$, $q(\mathbf{a})$, $q(\boldsymbol{\omega})$ and $q(\mathbf{e})$

To infer $q(\boldsymbol{\lambda})$, we expand the objective function of Eqn. (6) by explicitly expressing $\{\alpha_{\lambda,i}\}$ and $\{\beta_{\lambda,i}\}$ as:

$$\begin{aligned}\mathcal{Q}_\lambda : \; = & \; (\alpha_{\lambda,i} - \alpha_{\lambda,0})[\psi(\alpha_{\lambda,i}) - \log(\beta_{\lambda,i})] \\ & (\beta_{\lambda,i} - \beta_{\lambda,0} - \frac{1}{2}\widetilde{a_i^2})\frac{\alpha_{\lambda,i}}{\beta_{\lambda,i}} + \text{Const.}, \end{aligned} \tag{7}$$

where $\psi()$ denotes the digamma function. Note that the inference for $q(\boldsymbol{\lambda})$ does not involve the constraints of Eqn. (6). By setting $\frac{\partial \mathcal{Q}_\lambda}{\partial \alpha_{\lambda,i}} = 0$ and $\frac{\partial \mathcal{Q}_\lambda}{\partial \beta_{\lambda,i}} = 0$ and solving for optimal $\{\alpha_{\lambda,i}\}$ and $\{\beta_{\lambda,i}\}$, we obtain the updating rules as:

$$\alpha_{\lambda,i} = \alpha_{\lambda 0} + \frac{1}{2}, \quad \beta_{\lambda,i} = \beta_{\lambda 0} + \frac{\widetilde{a_i^2}}{2}, \quad \forall i, \tag{8}$$

where $\widetilde{a_i^2}$ denotes the expected value of a_i^2 with respect to the variational distribution $q(\mathbf{a})$, namely $\widetilde{a_i^2} = E_{q(\mathbf{a})}(a_i^2) = \mu_{a,i}^2 + \sigma_{a,i}^2$.

To infer $q(\mathbf{a})$, we expand Eqn. (6) by explicitly expressing $\{\mu_{a,i}\}$, $\{\sigma_{a,i}\}$ and derive the Lagrange of Eqn. (6) as:

$$\begin{aligned}\mathcal{L}_a \; := & \; \frac{1}{2}\sum_{i=1}^{N} \widetilde{\lambda_i}(\mu_{a,i}^2 + \sigma_{a,i}^2) - \frac{1}{2}\sum_{i=1}^{N} \log(2\pi e \sigma_{a,i}^2) \\ - & \; \sum_{j=1}^{N} \upsilon_j^\star \{y_j[\boldsymbol{\mu}_a^T(\sum_{m=1}^{P} \mu_{e,m}\mathbf{k}_{j,.}^m \odot \widetilde{\mathbf{z}_{j,.}^m}) + b] - 1 + \xi_j\} \\ + & \; \mathcal{C}\sum_j \xi_j - \sum_{j=1}^{N} \upsilon_j \xi_j + \text{Const.}, \end{aligned} \tag{9}$$

where we can calculate $\widetilde{\mathbf{z}_{j,.}^m} = E_{q(\mathbf{z}_{j,.}^m)}(\mathbf{z}_{j,.}^m) = \boldsymbol{\phi}_{j,.}^m$ and $\widetilde{\lambda_i} = E_{q(\lambda_i)}(\lambda_i) = \alpha_{\lambda,i}/\beta_{\lambda,i}$. Here $\boldsymbol{\upsilon}^\star = [\upsilon_1^\star, \cdots, \upsilon_N^\star]^T$ is the vector of dual variables. By setting $\frac{\partial \mathcal{L}_a}{\partial \sigma_{a,i}^2} = 0$ we obtain the updating rules for $\{\sigma_{a,i}\}$ as:

$$\sigma_{a,i}^2 = \frac{1}{\widetilde{\lambda_i}}, \forall i. \tag{10}$$

By setting $\frac{\partial \mathcal{L}_a}{\partial \mu_{a,i}} = 0$ we can obtain the updating rules:

$$\mu_{a,i} = (\sum_{j=1}^{N} \upsilon_j^\star y_j \sum_{m=1}^{P} k_{j,i}^m \widetilde{z_{j,i}^m}\widetilde{e_m})/\widetilde{\lambda_i}, \quad \forall i. \tag{11}$$

Here $\widetilde{e_m} = E_{q(e_m)}(e_m) = \mu_{e,m}$. By setting $\frac{\partial \mathcal{L}_a}{\partial \xi_j} = 0$ and $\frac{\partial \mathcal{L}_a}{\partial b} = 0$ and substitute $\mu_{a,i}$ with Eqn. (11), we can derive the dual problem as:

$$\begin{aligned}\mathrm{D_a}: \quad & \min_{\boldsymbol{\upsilon}^\star} \; \mathbf{1}^T\boldsymbol{\upsilon}^\star - \frac{1}{2}\boldsymbol{\upsilon}^{\star,T} A_1 \boldsymbol{\upsilon}^\star, \\ & s.t. \;\; 0 \leq \upsilon_j^\star \leq \mathcal{C}, \quad \forall j, \\ & \sum_{j=1}^{N} y_j \upsilon_j^\star = 0, \end{aligned} \tag{12}$$

where $\mathbf{1}$ is an all-one vector and A_1 is defined as:

$$A_1 = \mathbf{y}\mathbf{y}^T \circ [\sum_{i=1}^{N} \frac{1}{\widetilde{\lambda_i}}(\sum_{m=1}^{P} \widetilde{e_m}\mathbf{k}_{,i}^m \circ \widetilde{\mathbf{z}_{.,i}^m})(\sum_{m=1}^{P} \widetilde{e_m}\mathbf{k}_{,i}^m \circ \widetilde{\mathbf{z}_{.,i}^m})^T]. \tag{13}$$

Note that the dual problem has the same form of the support vector machine (SVM), therefore we use the SMO algorithm [16] to efficiently solve for the optimal values of dual variables.

Inference for $q(\boldsymbol{\omega})$ is similar to the inference of $q(\boldsymbol{\lambda})$ and the updating rules for $\{\alpha_{\omega,m}\}$ and $\{\beta_{\omega,m}\}$ are given by:

$$\widetilde{\alpha_{\omega,m}} = \alpha_{\omega 0} + \frac{1}{2}, \quad \widetilde{\beta_{\omega,m}} = \beta_{\omega 0} + \frac{\widetilde{e_m^2}}{2}, \quad \forall m, \tag{14}$$

where $\widetilde{e_m^2} = E_{q(e_m)}(e_m) = \mu_{e,m}^2 + \sigma_{e,m}^2$. Similarly, to infer $q(\mathbf{e})$, we expand Eqn. (6) by explicitly expressing $\{\mu_{e,m}\}$

and $\{\sigma_{e,m}\}$ and derive the Lagrange as:

$$\begin{aligned}\mathcal{L}_e \;:=\;& \frac{1}{2}\sum_{m=1}^{P}\widetilde{\omega_m}(\mu_{e,m}^2+\sigma_{e,m}^2)-\frac{1}{2}\sum_{m=1}^{P}\log(2\pi e\sigma_{e,m}^2)\\ -\;& \sum_{j=1}^{N}\upsilon_j^\star\{y_j[\widetilde{\mathbf{a}^T}(\sum_{m=1}^{P}\mu_{e,m}\mathbf{k}_{j,\cdot}^m\circ\widetilde{\mathbf{z}_{j,\cdot}^m})+b]-1+\xi_j\}\\ +\;& \mathcal{C}\sum_{j=1}^{N}\xi_j-\sum_{j=1}^{N}\upsilon_j\xi_j+\text{Const.} \qquad (15)\end{aligned}$$

By setting $\frac{\partial\mathcal{L}_e}{\partial\sigma_{e,m}^2}=0$ and $\frac{\partial\mathcal{L}_e}{\partial\mu_{e,m}}=0$ we can have the updating rules for $\{\mu_{e,m}\}$ and $\{\sigma_{e,m}\}$ as:

$$\begin{aligned}\sigma_{e,m}^2 &= \frac{1}{\widetilde{\omega_m}},\forall m, & (16)\\ \mu_{e,m} &= [\sum_{j=1}^{N}\upsilon_j^\star y_j\widetilde{\mathbf{a}^T}(k_{j,\cdot}^m\circ\widetilde{\mathbf{z}_{j,\cdot}^m})]/\widetilde{\omega_m},\quad \forall m, & (17)\end{aligned}$$

where $\widetilde{\mathbf{a}}=E_{q(\mathbf{a})}(\mathbf{a})=\boldsymbol{\mu}_a$ and $\widetilde{\omega_m}=E_{q(\omega_m)}(\omega_m)=\alpha_{\omega,m}/\beta_{\omega,m}$. By setting $\frac{\partial\mathcal{L}_e}{\partial\xi_j}=0$ and $\frac{\partial\mathcal{L}_e}{\partial b}=0$ and substitute $\mu_{e,m}$ with Eqn. (17) we can obtain the dual problem as:

$$\begin{aligned}\mathrm{D_e}: \quad & \min_{\boldsymbol{\upsilon}^\star}\ \mathbf{1}^T\boldsymbol{\upsilon}^\star-\frac{1}{2}\boldsymbol{\upsilon}^{\star,T}A_2\boldsymbol{\upsilon}^\star,\\ & s.t.\ \ 0\le\upsilon_j^\star\le\mathcal{C},\quad\forall j, \qquad (18)\\ & \sum_{j=1}^{N}y_j\upsilon_j^\star=0, \qquad (19)\end{aligned}$$

where A_2 is defined as:

$$A_2=\mathbf{y}\mathbf{y}^T\circ[\sum_{m=1}^{P}\frac{1}{\widetilde{\omega_m}}(\mathbf{K}^m\circ\widetilde{\mathbf{Z}^m})^T\widetilde{\mathbf{a}\mathbf{a}^T}(\mathbf{K}^m\circ\widetilde{\mathbf{Z}^m})],\ (20)$$

and we apply SMO algorithm to obtain the optimal values for the dual variables.

3.3. Inference for $q(\mathbf{Z})$ and $q(\mathbf{\Pi})$

We note that the inference of $q(\mathbf{\Pi})$ does not involve the constraints, and we expand the objective in Eqn. (6) by explicitly expressing $\{\alpha_{\pi,i}^m\}$ and $\{\beta_{\pi,i}^m\}$ as:

$$\begin{aligned}\mathcal{Q}_\pi &:= (\alpha_{\pi,i}^m-\alpha_{\pi0}-\sum_{j=1}^{N}\widetilde{z_{j,i}^m})(\psi(\alpha_{\pi,i}^m)-\psi(\alpha_{\pi,i}^m+\beta_{\pi,i}^m))\\ &+(\beta_{\pi,i}^m-\beta_{\pi0}-\sum_{j=1}^{N}(1-\widetilde{z_{j,i}^m}))(\psi(\beta_{\pi,i}^m)-\psi(\alpha_{\pi,i}^m+\beta_{\pi,i}^m)). \qquad (21)\end{aligned}$$

By setting $\frac{\partial\mathcal{Q}_\pi}{\partial\alpha_{\pi,i}^m}=0$ and $\frac{\partial\mathcal{Q}_\pi}{\partial\beta_{\pi,i}^m}=0$ we can obtain the updating rules as:

$$\alpha_{\pi,i}^m=\alpha_{\pi,0}+\sum_{j=1}^{N}\widetilde{z_{j,i}^m},\quad \beta_{\pi,i}^m=\beta_{\pi,0}+\sum_{j=1}^{N}(1-\widetilde{z_{j,i}^m}),\ (22)$$

where $\widetilde{z_{j,i}^m}=E_{q(z_{j,i}^m)}(z_{j,i}^m)=\phi_{j,i}^m,\forall j,i,m.$

For the inference of $q(\mathbf{Z})$, we expand Eqn. (6) by explicitly expressing $\{\phi_{j,i}^m\}$ and derive the Lagrange as:

$$\begin{aligned}\mathcal{L}_z \;:=\;& -\phi_{j,i}^m\widetilde{\log(\pi_i^m)}-(1-\phi_{j,i}^m)\widetilde{\log(1-\pi_i^m)}\\ +\;& \phi_{j,i}^m\log(\phi_{j,i}^m)+(1-\phi_{j,i}^m)\log(1-\phi_{j,i}^m)\\ -\;& \upsilon_j^\star\{y_j[\mathbf{a}^T(\sum_{m=1}^{P}\widetilde{e_m}\mathbf{k}_{,j}^m\circ\boldsymbol{\phi}_{,j}^m)+b]-1+\xi_j\}\\ +\;& \mathcal{C}\xi_j-\upsilon_j^\star\xi_j+\text{Const.} \qquad (23)\end{aligned}$$

In theory, we can do the optimization to get the optimal solution of $\{\phi_{j,i}^m\}$ and the corresponding optimal Lagrange multipliers $\{\upsilon_j^\star\}$. But the full optimization would be expensive. Therefore we follow the optimization strategy proposed in [32], which is to perform a single step update of $\phi_{j,i}^m$, rather than a full optimization. Note that this one-step approximation could lead to a slight increase of the objective function during the iterations. Our empirical studies show that this increase is usually within an acceptable range. More specifically, we fix $\boldsymbol{\xi}$ and the lagrange multipliers $\boldsymbol{\upsilon}^\star$ at the optimum solution of the previous step. Then, by setting $\frac{\partial\mathcal{L}_z}{\partial\phi_{j,i}^m}=0$ together with the constraint $0\le\phi_{j,i}^m\le1$, we obtain the updating rules for $\phi_{j,i}^m$ as:

$$\begin{aligned}\phi_{j,i}^m &= \frac{\exp\big(\widetilde{\log(\pi_i^m)}-\widetilde{\log(1-\pi_i^m)}+\Delta_{j,i}^m\big)}{\exp\big(\widetilde{\log(\pi_i^m)}-\widetilde{\log(1-\pi_i^m)}+\Delta_{j,i}^m\big)+1},\\ \Delta_{j,i}^m &= \upsilon_j^\star y_j\widetilde{a_i}\widetilde{e_m}k_{j,i}^m,\quad\forall i,j,m. \qquad (24)\end{aligned}$$

Here $\widetilde{\log(\pi_i^m)}=\psi(\alpha_{\pi,i}^m)-\psi(\alpha_{\pi,i}^m+\beta_{\pi,i}^m)$ and $\widetilde{\log(1-\pi_i^m)}=\psi(\beta_{\pi,i}^m)-\psi(\alpha_{\pi,i}^m+\beta_{\pi,i}^m)$. The derived updating rules for $q(\mathbf{Z})$ have intuitive explanations: 1) $\upsilon_j^\star=0$ means the corresponding samples (instances) are far from the decision boundary and are easily classified, therefore the derived updating rule can solely depend on the prior $q(\pi_i^m)$ and we don't need to change the value of $\phi_{j,i}^m$ according to the observed data $z_{j,i}^m$, *i.e.*, $\Delta_{j,i}^m=0$; 2) $\upsilon_j>0$ means the corresponding samples are around the decision boundary and are more likely to be *ambiguous*. We assume $\widetilde{e_m}>0$. If $\widetilde{a_i}y_jk_{j,i}^m<0$, it means that $k_{j,i}^m$ can contribute in the correct direction (sign) to the prediction value f_j. It is therefore less ambiguous if we use the i-th reference instance $\mathbf{x}_i^m$ in calculating the kernel representation for feature instance $\mathbf{x}_j^m$. In this case we should rely more on the similarity $k_{j,i}^m$ and thus $\phi_{j,i}^m$ should be increased, which follows that $\Delta_{j,i}^m>0$. And vice versa.

3.4. Convergence, Complexity and Prediction

The stopping criterion is set that change of parameter values between consecutive iterations do not exceed 10^{-4}. The SMO algorithm in both E and M steps decreases the objective function value. Despite of the slight increase of the cost function during inference of $q(\mathbf{Z})$, our empirical studies show that this increase is usually within an acceptable range and does not influence the trend of the optimization procedure. Our experiments also show convergence can usually be attained within 20 iterations. Besides the computational cost for SMO, the complexity of the computation steps for $\boldsymbol{\mu}_a$, A_1, $\boldsymbol{\mu}_e$, A_2, $\mathbf{\Pi}$ and $\mathbf{\Phi}$ are just $\mathcal{O}(N^2P)$ (we ignore other less computational steps). In the meantime, as our algorithm does not involve inversion of the kernel matrix (which is costly), it can scale well with large scale problem (*i.e.*, large N).

We can use the obtained posterior estimation of the variational distribution q to derive the predictive function for an unseen sample (out-of-sample) $\mathbf{x}_\star = [\mathbf{x}_\star^1, \cdots, \mathbf{x}_\star^P]^T$. This can be done by calculating the expectation the predictive function over q as:

$$\begin{aligned} f_j &= E_q\{\mathbf{a}^T(\sum_{m=1}^{P} e_m \mathbf{k}_{\star,\cdot}^m \circ \mathbf{z}_{\star,\cdot}^m)\} + b, \\ &\doteq \boldsymbol{u}_a^T[\sum_{m=1}^{P} \mu_{e,m} \mathbf{k}_{\star,\cdot}^m \circ (\boldsymbol{\alpha}_\pi^m./\boldsymbol{\beta}_\pi^m)] + b, \end{aligned} \quad (25)$$

where we use the fact $E_q(\mathbf{z}_{\star,\cdot}) = E_{q(\boldsymbol{\phi}_{\star,\cdot}^m)}\boldsymbol{\phi}_{\star,\cdot}^m \doteq (\boldsymbol{\alpha}_\pi^m./\boldsymbol{\beta}_\pi^m)$ and $./$ denotes element-wise division. For multi-class learning, we use the one-versus-all scheme to directly extend the proposed model.

4. Experiments

4.1. Synthetic Data

We have 100 positive data samples and 100 negative data samples. Each sample is composed of 50 types of features and each type of feature is two-dimensional. For each feature type (i.e., two-dimensional vector), positive and negative instances are distributed as two Gaussians. We randomly select 20% instances and add Gaussian noises in order to make the positive instances confuse with negative ones (i.e., distributed together and not easily separable. See the *diamond* shaped instances in Figure 2). We then generate 50 Gaussian kernels (each kernel corresponds to a feature type) and run BPMKL training. The feature instances and the corresponding inferred confidence values $\{\phi_{ji}^k = E_q(z_{ji}^k)\}$ from two feature types are visualized in Figure 2. We rearrange $\{\phi_{ji}^k\}$ and the ones corresponding to the 20% corrupted instances are on top. We note that BPMKL can automatically select good feature instances (brighter regions on the map) for kernel representation.

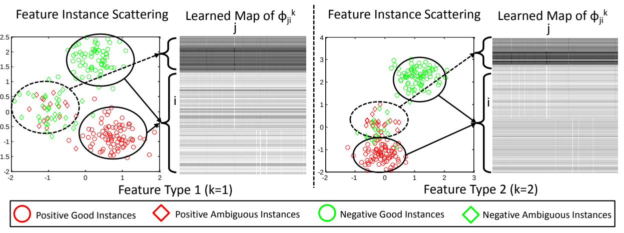

Figure 2. Synthetic experiment result. Left column: $2D$ feature instance scattering plot. Right column: the corresponding ϕ_{ji}^k map learned by BPMKL. Brighter pixels represent high ϕ_{ji}^k values. For better view, please zoom in the original pdf.

4.2. UCL Machine Learning Benchmarks

Classification accuracies and running times of BPMKL and state-of-the-art MKL algorithms are evaluated on five data sets from the UCI repository: bupaliver, heart, ionosphere, pima, and sonar. Following the experimental settings in [5], we randomly select 70% samples for training and the rest for testing. We construct Gaussian and polynomial kernels with the same bandwidth and degree parameter settings as in [5]. BPMKL is compared to three state-of-the-art MKL algorithms including: 1) SILP [19], 2) SimpleMKL [17] and 3) BEMKL [5]. The comparison results are given in Table 1 in terms of mean value (e.g., accuracy, running time) and standard deviation out of 20 runs. Table 1 shows: 1) BPMKL algorithm outperforms all comparing MKL algorithms in terms of classification accuracy. The reason is that BPMKL selects good reference instances to compute kernel representation while others do not possess this capability and the learned classifier is degraded by noisy feature instances; 2) the time complexity of BPMKL is similar with that of SimpleMKL since both algorithms are based on iterative SMO steps. Method based on kernel matrix inversion (e.g., SILP) becomes inefficient when the kernel size (N^2) is large (e.g., pima) and the number of kernels (P) to inverse is large (e.g., sonar).

To evaluate the algorithmic robustness, for bupaliver and ionosphere, we randomly corrupt some features of the training data. We randomly select 10%, 20%, 30% feature instances (e.g., elements of feature vectors) from the training samples and contaminate them by adding zero mean and unit variance noises. We then train MKL classifiers and visualize the classification accuracies on the testing data in Figure 3. We note that BPMKL is robust to noisy features since it can *select* good feature instances for kernel representation. In contrast, other MKL methods degrade significantly with the increasing of noise level.

4.3. Image Classification Benchmarks

We perform image classification experiments using the proposed BPMKL algorithm on the Caltech101 [11] and Caltech256 [6] image benchmarks. For both datasets, we randomly select 30 images for training and report the classi-

Dataset	bupaliver N=241 P=91		heart N=189 P=377		ionosphere N=245 P=442		pima N=537 P=117		sonar N=144 P=793	
	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)
SILP	65.9 ± 2.6	21.6 ± 4.6	78.3 ± 2.1	53.9 ± 2.8	91.7 ± 2.5	108 ± 21.2	76.5 ± 2.3	118 ± 19.9	80.5 ± 5.1	1201 ± 385
SimpleMKL	65.9 ± 2.3	9.7 ± 2.5	78.6 ± 3.2	25.8 ± 3.9	91.5 ± 2.5	62.4 ± 10.9	76.5 ± 2.6	41.5 ± 5.6	80.6 ± 5.1	85 ± 46.2
BEMKL	68.4 ± 4.8	4.5 ± 0.5	80.8 ± 4.5	15.7 ± 1.6	92.0 ± 1.7	24.1 ± 3.1	75.0 ± 2.3	22.5 ± 2.4	76.9 ± 4.1	55.8 ± 1.6
BPMKL	$\mathbf{71.1 \pm 3.9}$	6.3 ± 1.4	$\mathbf{83.1 \pm 2.8}$	22.6 ± 2.2	$\mathbf{96.2 \pm 2.2}$	30.3 ± 5.9	$\mathbf{77.3 \pm 2.7}$	38.9 ± 4.1	$\mathbf{84.2 \pm 3.7}$	32.8 ± 4.7

Table 1. Accuracy and running time comparisons on UCL machine learning benchmarks. The computing platform is Intel Xeon(R) duo-core@2.8GHz CPU with 6 GB memory. We denote by N and P the number of samples and kernels, respectively.

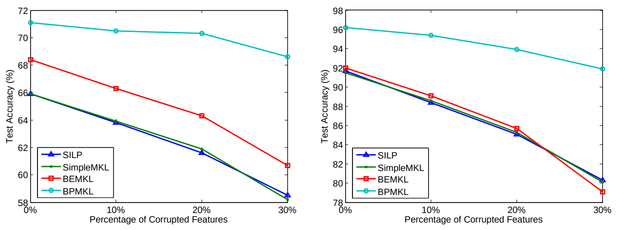

Figure 3. Test accuracies under different noise levels on bupaliver (left) and ionosphere (right).

fication accuracies averaged over all categories. We extract SIFT [12] features from densely located patches centered at every 4 pixels on the images and the size of the patches is fixed as 16×16 pixels. We construct a visual word dictionary containing K words ($K = 4096$) from the training samples via K-means clustering. Each SIFT feature vector is encoded into a K-dimensional code vector by locality-constrained linear coding (LLC) [25].

Multiple kernels are constructed using the following scheme. Images are hierarchically partitioned into 1×1, 2×2 and 4×4 blocks on as in SPM [10]. We also vertically partition the images into 2×1 and 3×1. We therefore has 26 spatial blocks (feature channels). Within each block, features are pooled to form a K-dimensional representation vector. For each feature channel, we calculate the kernel matrix using linear, Gaussian, χ^2, and histogram intersection kernels. These yields $26 \times 4 = 104$ kernel matrices. The bandwidth parameters for χ^2 and Gaussian kernels are set as the average of the squared distances (χ^2 and Euclidean, respectively) of the training sample pairs. In addition, following [4], we also include 8 PHOG shape based kernels, 3 local binary pattern based kernels and one $V1S+$ (thresholded Gabor filter responses) based kernel. The total number of kernels is therefore 116.

In Table 2, mean accuracies and standard deviations out of 20 runs (e.g., random split) are reported for various image classification methods. The methods for comparison include: 1) the spatial pyramid matching kernel method (KSPM) in [10]; 2) the sparse coding + spatial pyramid matching method (ScSPM) [30]; 3) the locally linear encoding method (LLC) in [25]; 4) the smooth sparse coding method (SSC) in [2]; 5) the Laplacian sparse coding method (LScSPM) in [3]; 6) the multiple kernel object feature combination method (LP-β-MKL) in [4]; 7) the dense feature pooling method (BSPR) in [18]. Note that we directly report the published results for these state-of-the-art methods as all methods follow the same experimental settings. We note that BPMKL achieves higher average test accuracy than state-of-the-art MKL algorithms. BPMKL also outperforms state-of-the-art image classification methods. To further demonstrate the instance selection capability of BPMKL, we illustrate in Figure 4 several image blocks that correspond to top (left column) and bottom (right column) confidence values (i.e., ϕ_{ji}^k) trained by BPMKL. We note that the image blocks which are selected for kernel representation (top ϕ_{ji}^k values) present discriminative appearances and vice versa.

Table 2. Classification accuracy (%) comparison on Caltech101 and Caltech256 datasets.

Algorithm	Caltech101	Caltech256
KSPM [10]	64.6 ± 0.8	34.1
ScSPM [30]	73.2 ± 0.5	34.0 ± 0.4
LLC [25]	73.4	41.2
SSC [2]	81.0 ± 1.2	–
LScSPM [3]	–	35.7 ± 0.1
LP-β-MKL [4]	–	45.8
BSPR [18]	–	46.8 ± 0.2
SILP	78.3 ± 0.9	43.8 ± 0.5
SimpleMKL	78.7 ± 1.2	44.1 ± 0.6
BEMKL	79.1 ± 1.3	43.4 ± 1.1
BPMKL	$\mathbf{83.4 \pm 0.8}$	$\mathbf{48.8 \pm 0.7}$

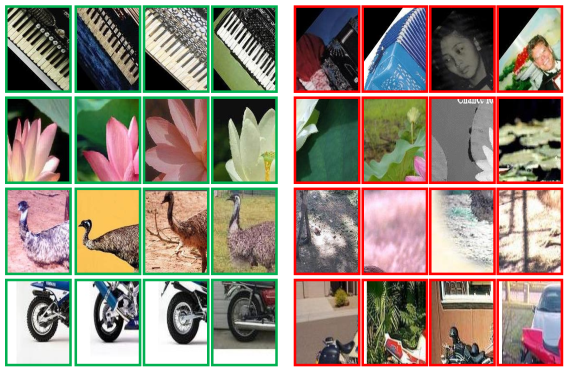

Figure 4. Sample image blocks from Caltech101 dataset. Left: with highest inferred ϕ_{ji}^k values. Right: with lowest inferred ϕ_{ji}^k values. Each row corresponds to patches from one category.

4.4. Action Recognition Benchmark

We also apply the BPMKL algorithm for action recognition on the challenging Hollywood2 action recognition video benchmark [13]. We follow the same settings as in [24] to extract dense trajectory features, i.e., length of

Table 3. Average precision per action class for the Hollywood2 dataset, using the dense trajectory method [24] and our BPMKL method.

Method	Dense Trajectory [24]	BPMKL
AnswerPhone	32.6	35.8
DriveCar	88.0	87.5
Eat	65.2	68.8
FightPerson	81.4	84.5
GetOutCar	52.7	51.9
HandShake	29.6	35.7
HugPerson	54.2	55.5
Kiss	65.8	67.9
Run	82.1	85.6
SitDown	62.5	66.7
SitUp	20.0	24.4
StandUp	65.2	68.1
mAP	58.3	**61.0**

trajectory is set to be 15. For each motion trajectory, as in [24], we extract trajectory shape descriptor (TSD), motion boundary histogram (MBH), histogram of oriented gradient (HoG) and histogram of optical flow (HoF) descriptors. We calculated bag-of-words (BOW) representations for each of the three types of feature descriptors. Dictionaries are trained by K-means algorithm and the size is set to be 2048. Multiple kernels are constructed as follows. We follow [20] and spatially and temporally partition the video volume into 24 sub-volumes and for each sub-volume we calculate BOW representation vectors for four types of features. For each feature channel, we calculate the χ^2 kernel matrix. Therefore the total number of kernels is $24 \times 4 = 96$. The bandwidth parameter for χ^2 kernel is set as the average of the squared distances of the training sample pairs. Per-class average precision (AP) comparisons with the state-of-the-art dense trajectory method [24] are given in Table 3. BPMKL outperforms the state-of-the-art on this challenging benchmark. The mAP values using other MKL algorithms are 58.6, 58.9, 58.0 for SILP, SimpleMKL and BEMKL, respectively.

5. Conclusions

A Beta process multiple kernel learning (BPMKL) method was proposed to perform instance selection for more discriminative multiple kernel representation. It was applied for various vision problems including image classification and action recognition with significant performance gain over the state-of-the-art methods.

Acknowledgment

This study is supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR). This work is also partially supported by the National Natural Science Foundation (NSF) of China (No. 61300056).

References

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

[2] K. Balasubramanian, K. Yu, and G. Lebanon. Smooth sparse coding via marginal regression for learning sparse representations. In *ICML*, 2013.

[3] S. Gao, I. W. Tsang, L.-T. Chia, and P. Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, 2010.

[4] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, pages 221–228, 2009.

[5] M. Gönen. Bayesian efficient multiple kernel learning. In *ICML*, 2012.

[6] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[7] N. L. Hjort. Nonparametric bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*.

[8] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. lp-norm multiple kernel learning. *JMLR*, 12:953–997, 2011.

[9] G. Lanckriet, N. Cristianini, P. Bartlett, and L. E. Ghaoui. Learning the kernel matrix with semi-definite programming. *JMLR*, 5:27–72, 2004.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[11] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR workshop*, 2004.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[13] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, pages 2929–2936, June.

[14] R. Mittelman, H. Lee, B. Kuipers, and S. Savarese. Weakly supervised learning of mid-level features with beta-bernoulli process restricted boltzmann machines. In *CVPR*, 2013.

[15] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *ICML*, 2009.

[16] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[17] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *JMLR*, 9:2491–2521, 2008.

[18] D. X. S. L. S. Yan, X. Xu and X. Li. Beyond spatial pyramids: a new feature extraction framework with dense spatial sampling for image classification. In *ECCV*, pages 473–487, 2012.

[19] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

[20] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, pages 2004–2011, 2009.

[21] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, pages 1065–1072, 2009.

[22] S. V. N. Vishwanathan, Z. Sun, N. Theera-ampornpunt, and M. Varma. Multiple kernel learning and the smo algorithm. In *NIPS*, 2010.

[23] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr groups using fast kernel machines. *TPAMI*, 34(11):2177–2188, 2012.

[24] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action recognition by dense trajectories. In *CVPR*, 2011.

[25] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[26] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song. Unified video annotation via multi-graph learning. *TCSVT*, 19(5):733–746, 2009.

[27] M. Wang, H. Li, D. Tao, K. Lu, and X. Wu. Multimodal graph-based reranking for web image search. *TIP*, 21(11):4649–4661, 2012.

[28] Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, 2008.

[29] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

[30] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[31] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric bayesian dictionary learning for sparse image representations. In *NIPS*, 2009.

[32] J. Zhu, A. Ahmed, and E. P. Xing. Medlda: Maximum margin supervised topic models for regression and classification. In *ICML*, 2009.