Pattern Recognition 46 (2013) 795–807

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Localized algorithms for multiple kernel learning

Mehmet Gönen*, Ethem Alpaydın

Department of Computer Engineering, Boğaziçi University, TR-34342 Bebek, İstanbul, Turkey

ARTICLE INFO

Article history:
Received 20 September 2011
Received in revised form
28 March 2012
Accepted 2 September 2012
Available online 11 September 2012

Keywords:
Multiple kernel learning
Support vector machines
Support vector regression
Classification
Regression
Selective attention

ABSTRACT

Instead of selecting a single kernel, multiple kernel learning (MKL) uses a weighted sum of kernels where the weight of each kernel is optimized during training. Such methods assign the same weight to a kernel over the whole input space, and we discuss localized multiple kernel learning (LMKL) that is composed of a kernel-based learning algorithm and a parametric gating model to assign local weights to kernel functions. These two components are trained in a coupled manner using a two-step alternating optimization algorithm. Empirical results on benchmark classification and regression data sets validate the applicability of our approach. We see that LMKL achieves higher accuracy compared with canonical MKL on classification problems with different feature representations. LMKL can also identify the relevant parts of images using the gating model as a saliency detector in image recognition problems. In regression tasks, LMKL improves the performance significantly or reduces the model complexity by storing significantly fewer support vectors.

1. Introduction

Support vector machine (SVM) is a discriminative classifier based on the theory of structural risk minimization [33]. Given a sample of independent and identically distributed training instances $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $y_i \in \{-1,+1\}$ is its class label, SVM finds the linear discriminant with the maximum margin in the feature space induced by the mapping function $\Phi(\cdot)$. The discriminant function is

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$$

whose parameters can be learned by solving the following quadratic optimization problem:

$$\begin{aligned} \text{min.} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^N \xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w} \in \mathbb{R}^S,\ \boldsymbol{\xi} \in \mathbb{R}_+^N,\ b \in \mathbb{R} \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i) \rangle + b) \geq 1-\xi_i \quad \forall i \end{aligned}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, S is the dimensionality of the feature space obtained by $\Phi(\cdot)$, C is a predefined positive trade-off parameter between model simplicity and classification error, $\boldsymbol{\xi}$ is the vector of slack variables, and b is the bias term of the separating hyperplane. Instead of solving this optimization problem directly, the Lagrangian dual function enables us to obtain the following dual formulation:

$$\begin{aligned} \text{max.} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_i y_i y_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in [0,C]^N \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \end{aligned}$$

where $\boldsymbol{\alpha}$ is the vector of dual variables corresponding to each separation constraint and the obtained kernel matrix of $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle$ is positive semidefinite. Solving this, we get $\boldsymbol{w} = \sum_{i=1}^N \alpha_i y_i \Phi(\boldsymbol{x}_i)$ and the discriminant function can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^N \alpha_i y_i k(\boldsymbol{x}_i,\boldsymbol{x}) + b.$$

There are several kernel functions successfully used in the literature such as the linear kernel (k_L), the polynomial kernel (k_P), and the Gaussian kernel (k_G)

$$\begin{aligned} k_L(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \langle \boldsymbol{x}_i,\boldsymbol{x}_j \rangle \\ k_P(\boldsymbol{x}_i,\boldsymbol{x}_j) &= (\langle \boldsymbol{x}_i,\boldsymbol{x}_j \rangle + 1)^q \quad q \in \mathbb{N} \\ k_G(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \exp(-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2/s^2) \quad s \in \mathbb{R}_{++}. \end{aligned}$$

There are also kernel functions proposed for particular applications, such as natural language processing [24] and bioinformatics [31].

Selecting the kernel function $k(\cdot,\cdot)$ and its parameters (e.g., q or s) is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function

* Corresponding author.
E-mail addresses: gonen@boun.edu.tr (M. Gönen), alpaydin@boun.edu.tr (E. Alpaydın).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.09.002

among a set of kernel functions on a separate validation set different from the training set. In recent years, multiple kernel learning (MKL) methods are proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=f_\eta(\{k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\}_{m=1}^P) \tag{1}$$

where the combination function $f_\eta(\cdot)$ can be a linear or a nonlinear function of the input kernels. Kernel functions, $\{k_m(\cdot,\cdot)\}_{m=1}^P$, take P feature representations (not necessarily different) of data instances, where $\boldsymbol{x}_i=\{\boldsymbol{x}_i^m\}_{m=1}^P$, $\boldsymbol{x}_i^m\in\mathbb{R}^{D_m}$, and D_m is the dimensionality of the corresponding feature representation.

The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination. There can be two uses of MKL: (i) Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias, and in allowing a learner to choose among a set of kernels, a better solution can be found. (ii) Different kernels may be using inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, combining kernels is one possible way to combine multiple information sources.

Since their original conception, there is significant work on the theory and application of multiple kernel learning. Fixed rules use the combination function in (1) as a fixed function of the kernels, without any training. Once we calculate the combined kernel, we train a single kernel machine using this kernel. For example, we can obtain a valid kernel by taking the summation or multiplication of two kernels as follows [10]:

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=k_1(\boldsymbol{x}_i^1,\boldsymbol{x}_j^1)+k_2(\boldsymbol{x}_i^2,\boldsymbol{x}_j^2)$$
$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=k_1(\boldsymbol{x}_i^1,\boldsymbol{x}_j^1)k_2(\boldsymbol{x}_i^2,\boldsymbol{x}_j^2).$$

The summation rule is applied successfully in computational biology [27] and optical digit recognition [25] to combine two or more kernels obtained from different representations.

Instead of using a fixed combination function, we can have a function parameterized by a set of parameters $\boldsymbol{\Theta}$ and then we have a learning procedure to optimize $\boldsymbol{\Theta}$ as well. The simplest case is to parameterize the sum rule as a weighted sum

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j|\boldsymbol{\Theta}=\boldsymbol{\eta})=\sum_{m=1}^P \eta_m k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)$$

with $\eta_m\in\mathbb{R}$. Different versions of this approach differ in the way they put restrictions on the kernel weights [22,4,29,19]. For example, we can use arbitrary weights (i.e., linear combination), nonnegative kernel weights (i.e., conic combination), or weights on a simplex (i.e., convex combination). A linear combination may be restrictive and nonlinear combinations are also possible [23,13,8]; our proposed approach is of this type and we will discuss these in more detail later.

We can learn the kernel combination weights using a quality measure that gives performance estimates for the kernel matrices calculated on training data. This corresponds to a function that assigns weights to kernel functions

$$\boldsymbol{\eta}=g_\eta(\{k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\}_{m=1}^P).$$

The quality measure used for determining the kernel weights could be "kernel alignment" [21,22] or another similarity measure such as the Kullback–Leibler divergence [36]. Another possibility inspired from ensemble and boosting methods is to iteratively update the combined kernel by adding a new kernel as training continues [5,9]. In a trained combiner parameterized by $\boldsymbol{\Theta}$, if we assume $\boldsymbol{\Theta}$ to contain random variables with a prior, we can use a Bayesian approach. For the case of a weighted sum, we can, for example, have a prior on the kernel weights [11,12,28]. A recent survey of multiple kernel learning algorithms is given in [18].

This paper is organized as follows: We formulate our proposed nonlinear combination method localized MKL (LMKL) with detailed mathematical derivations in Section 2. We give our experimental results in Section 3 where we compare LMKL with MKL and single kernel SVM. In Section 4, we discuss the key properties of our proposed method together with related work in the literature. We conclude in Section 5.

2. Localized multiple kernel learning

Using a fixed unweighted or weighted sum assigns the same weight to a kernel over the whole input space. Assigning different weights to a kernel in different regions of the input space may produce a better classifier. If the data has underlying local structure, different similarity measures may be suited in different regions. We propose to divide the input space into regions using a gating function and assign combination weights to kernels in a data-dependent way [13]; in the neural network literature, a similar architecture is previously proposed under the name "mixture of experts" [20,3]. The discriminant function for binary classification is rewritten as

$$f(\boldsymbol{x})=\sum_{m=1}^P \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m,\Phi_m(\boldsymbol{x}^m)\rangle+b \tag{2}$$

where $\eta_m(\boldsymbol{x}|\mathbf{V})$ is a parametric gating model that assigns a weight to $\Phi_m(\boldsymbol{x}^m)$ as a function of $\boldsymbol{x}$ and $\mathbf{V}$ is the matrix of gating model parameters. Note that unlike in MKL, in LMKL, it is not obligatory to combine different feature spaces; we can also use multiple copies of the same feature space (i.e., kernel) in different regions of the input space and thereby obtain a more complex discriminant function. For example, as we will see shortly, we can combine multiple linear kernels to get a piecewise linear discriminant.

2.1. Gating models

In order to assign kernel weights in a data-dependent way, we use a gating model. Originally, we investigated the softmax gating model [13]

$$\eta_m(\boldsymbol{x}|\mathbf{V})=\frac{\exp(\langle \boldsymbol{v}_m,\boldsymbol{x}^{\mathcal{G}}\rangle+v_{m0})}{\sum_{h=1}^P \exp(\langle \boldsymbol{v}_h,\boldsymbol{x}^{\mathcal{G}}\rangle+v_{h0})}\quad \forall m \tag{3}$$

where $\boldsymbol{x}^{\mathcal{G}}\in\mathbb{R}^{D_{\mathcal{G}}}$ is the representation of the input instance in the feature space in which we learn the gating model and $\mathbf{V}\in\mathbb{R}^{P\times(D_{\mathcal{G}}+1)}$ contains the gating model parameters $\{\boldsymbol{v}_m,v_{m0}\}_{m=1}^P$. The softmax gating model uses kernels in a competitive manner and generally a single kernel is active for each input.

It is possible to use other gating models and below, we discuss two new ones, namely sigmoid and Gaussian. The gating model defines the shape of the region of expertise of the kernels. The sigmoid function allows multiple kernels to be used in a cooperative manner

$$\eta_m(\boldsymbol{x}|\mathbf{V})=1/(1+\exp(-\langle \boldsymbol{v}_m,\boldsymbol{x}^{\mathcal{G}}\rangle-v_{m0}))\quad \forall m. \tag{4}$$

Instead of parameterizing the boundaries of the local regions for kernels, we can also parameterize their centers and spreads using Gaussian gating

$$\eta_m(\boldsymbol{x}|\mathbf{V})=\frac{\exp(-\|\boldsymbol{x}^{\mathcal{G}}-\boldsymbol{\mu}_m\|_2^2/\sigma_m^2)}{\sum_{h=1}^P \exp(-\|\boldsymbol{x}^{\mathcal{G}}-\boldsymbol{\mu}_h\|_2^2/\sigma_h^2)}\quad \forall m \tag{5}$$

where $\mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)}$ contains the means, $\{\boldsymbol{\mu}_m\}_{m=1}^P$, and the spreads, $\{\sigma_m\}_{m=1}^P$; we do not experiment any further with this in this current work.

If we combine the same feature representation with different kernels (i.e., $\boldsymbol{x} = \boldsymbol{x}^1 = \boldsymbol{x}^2 = \ldots = \boldsymbol{x}^P$), we can simply use it also in the gating model (i.e., $\boldsymbol{x}^{\mathcal{G}} = \boldsymbol{x}$) [13]. If we combine different feature representations with the same kernel, the gating model representation $\boldsymbol{x}^{\mathcal{G}}$ can be one of the representations $\{\boldsymbol{x}^m\}_{m=1}^P$, a concatenation of a subset of them, or a completely different representation. In some application areas such as bioinformatics where data instances may appear in a non-vectorial format such as sequences, trees, and graphs, where we can calculate kernel matrices but cannot represent the data instances as $\boldsymbol{x}$ vectors directly, we may use an empirical kernel map [31, Chapter 2], which corresponds to using the kernel values between $\boldsymbol{x}$ and training points as the feature vector for $\boldsymbol{x}$, and define $\boldsymbol{x}^{\mathcal{G}}$ in terms of the kernel values [15]

$$\boldsymbol{x}^{\mathcal{G}} = [k_{\mathcal{G}}(\boldsymbol{x}_1,\boldsymbol{x}) \; k_{\mathcal{G}}(\boldsymbol{x}_2,\boldsymbol{x}) \; \cdots \; k_{\mathcal{G}}(\boldsymbol{x}_N,\boldsymbol{x})]^\top$$

where the gating kernel, $k_{\mathcal{G}}(\cdot,\cdot)$, can be one of the combined kernels, $\{k_m(\cdot,\cdot)\}_{m=1}^P$, a combination of them, or a completely different kernel used only for determining the gating boundaries.

2.2. Mathematical model

Using the discriminant function in (2) and regularizing the discriminant coefficients of all the feature spaces together, LMKL obtains the following optimization problem:

$$\begin{aligned}
&\text{min.} \quad \frac{1}{2}\sum_{m=1}^P \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^N \xi_i \\
&\text{w.r.t.} \quad \boldsymbol{w}_m \in \mathbb{R}^{S_m},\ \boldsymbol{\xi} \in \mathbb{R}_+^N,\ \mathbf{V} \in \mathbb{R}^{P\times(D_{\mathcal{G}}+1)},\ b \in \mathbb{R} \\
&\text{s.t.} \quad y_i f(\boldsymbol{x}_i) \geq 1-\xi_i \quad \forall i
\end{aligned} \tag{6}$$

where nonconvexity is introduced to the model due to the nonlinearity formed using the gating model outputs in the separation constraints. Instead of trying to solve (6) directly, we can use a two-step alternating optimization algorithm [13], also used for choosing kernel parameters [6] and obtaining η_m parameters of MKL [29]. This procedure consists of two basic steps: (i) solving the model with a fixed gating model, and, (ii) updating the gating model parameters with the gradients calculated from the current solution.

Note that if we fix the gating model parameters, the optimization problem (6) becomes convex and we can find the corresponding dual optimization problem using duality. For a fixed $\mathbf{V}$, we obtain the Lagrangian dual of the primal problem (6) as follows:

$$L_D(\mathbf{V}) = \frac{1}{2}\sum_{m=1}^P \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^N \xi_i - \sum_{i=1}^N \beta_i\xi_i - \sum_{i=1}^N \alpha_i\left(y_i f(\boldsymbol{x}_i) - 1 + \xi_i\right)$$

and taking the derivatives of $L_D(\mathbf{V})$ with respect to the primal variables gives

$$\begin{aligned}
&\frac{\partial L_D(\mathbf{V})}{\partial \boldsymbol{w}_m} = 0 \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^N \alpha_i y_i \eta_m(\boldsymbol{x}_i|\mathbf{V})\Phi_m(\boldsymbol{x}_i^m) \quad \forall m \\
&\frac{\partial L_D(\mathbf{V})}{\partial b} = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0 \\
&\frac{\partial L_D(\mathbf{V})}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \beta_i \quad \forall i.
\end{aligned} \tag{7}$$

From $L_D(\mathbf{V})$ and (7), the dual formulation is obtained as

$$\begin{aligned}
&\text{max.} \quad J(\mathbf{V}) = \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_i y_i y_j k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j) \\
&\text{w.r.t.} \quad \boldsymbol{\alpha} \in [0,C]^N \\
&\text{s.t.} \quad \sum_{i=1}^N \alpha_i y_i = 0
\end{aligned} \tag{8}$$

where the *locally combined kernel function* is defined as

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{m=1}^P \eta_m(\boldsymbol{x}_i|\mathbf{V}) k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\eta_m(\boldsymbol{x}_j|\mathbf{V}).$$

Note that if the input kernel matrices are positive semidefinite, the combined kernel matrix is also positive semidefinite by construction. Locally combined kernel matrix is the summation of P matrices obtained by pre- and post-multiplying each kernel matrix by the vector that contains gating model outputs for this kernel. Using the support vector coefficients obtained from (8) and the gating model parameters, we obtain the following discriminant function:

$$f(\boldsymbol{x}) = \sum_{i=1}^N \alpha_i y_i k_\eta(\boldsymbol{x}_i,\boldsymbol{x}) + b. \tag{9}$$

For a given $\mathbf{V}$, the gradients of the objective function in (8) are equal to the gradients of the objective function in (6) due to strong duality, which guarantees that, for a convex quadratic optimization, the dual problem has the same optimum value as its primal problem. These gradients are used to update the gating model parameters at each step.

2.3. Training with alternating optimization

We can find the gradients of $J(\mathbf{V})$ with respect to the parameters of all three gating models. The gradients of (8) with respect to the parameters of the softmax gating model (3) are

$$\begin{aligned}
\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{v}_m} &= -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\sum_{h=1}^P \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V})k_h(\boldsymbol{x}_i^h,\boldsymbol{x}_j^h)\eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times(\boldsymbol{x}_i^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \boldsymbol{x}_j^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V}))) \\
\frac{\partial J(\mathbf{V})}{\partial v_{m0}} &= -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\sum_{h=1}^P \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V})k_h(\boldsymbol{x}_i^h,\boldsymbol{x}_j^h)\eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V}) + \delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V}))
\end{aligned}$$

where $\Upsilon_i^j = \alpha_i\alpha_j y_i y_j$, and δ_m^h is 1 if $m=h$ and 0 otherwise. The same gradients with respect to the parameters of the sigmoid gating model (4) are

$$\begin{aligned}
\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{v}_m} &= -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \Upsilon_i^j \eta_m(\boldsymbol{x}_i|\mathbf{V})k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\eta_m(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times(\boldsymbol{x}_i^{\mathcal{G}}(1 - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \boldsymbol{x}_j^{\mathcal{G}}(1 - \eta_m(\boldsymbol{x}_j|\mathbf{V}))) \\
\frac{\partial J(\mathbf{V})}{\partial v_{m0}} &= -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \Upsilon_i^j \eta_m(\boldsymbol{x}_i|\mathbf{V})k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\eta_m(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times(1 - \eta_m(\boldsymbol{x}_i|\mathbf{V}) + 1 - \eta_m(\boldsymbol{x}_j|\mathbf{V}))
\end{aligned}$$

where the gating model parameters for a kernel function are updated independently. We can also find the gradients with respect to the means and the spreads of the Gaussian gating model (5) are

$$\begin{aligned}
\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{\mu}_m} &= -\sum_{i=1}^N\sum_{j=1}^N\sum_{h=1}^P \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V})k_h(\boldsymbol{x}_i^h,\boldsymbol{x}_j^h)\eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times((\boldsymbol{x}_i^{\mathcal{G}} - \boldsymbol{\mu}_m)(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + (\boldsymbol{x}_j^{\mathcal{G}} - \boldsymbol{\mu}_m)(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V})))/\sigma_m^2 \\
\frac{\partial J(\mathbf{V})}{\partial \sigma_m} &= -\sum_{i=1}^N\sum_{j=1}^N\sum_{h=1}^P \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V})k_h(\boldsymbol{x}_i^h,\boldsymbol{x}_j^h)\eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad \times(\|\boldsymbol{x}_i^{\mathcal{G}} - \boldsymbol{\mu}_m\|_2^2(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \|\boldsymbol{x}_j^{\mathcal{G}} - \boldsymbol{\mu}_m\|_2^2(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V})))/\sigma_m^3.
\end{aligned}$$

The complete algorithm of our proposed LMKL is summarized in Algorithm 1. Previously, we used to perform a predetermined number

of iterations [13]; now, we calculate a step size at each iteration using a line search method and catch the convergence of the algorithm by observing the change in the objective function value of (8). This allows converging to a better solution and hence a better learner. Our algorithm is guaranteed to converge in a finite number of iterations. At each iteration, we pick the step size using a line search method and there is no chance of increasing the objective function value. After a finite number of iterations, our algorithm converges to one of local optima due to nonconvexity of the primal problem in (6).

Algorithm 1. Localized Multiple Kernel Learning (LMKL).

1: Initialize $\mathbf{V}^{(0)}$ randomly
2: **repeat**
3: Calculate $\mathbf{K}_\eta^{(t)}=\{k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)\}_{i,j=1}^N$ using $\mathbf{V}^{(t)}$
4: Solve kernel machine with $\mathbf{K}_\eta^{(t)}$
5: Calculate descent direction $\frac{\partial J(\mathbf{V})}{\partial \mathbf{V}}$
6: Determine step size, $\Delta^{(t)}$, using a line search method
7: $\mathbf{V}^{(t+1)} \Leftarrow \mathbf{V}^{(t)}-\Delta^{(t)}\frac{\partial J(\mathbf{V})}{\partial \mathbf{V}}$
8: **until** convergence

2.4. Extensions to other algorithms

We extend our proposed LMKL framework for two-class classification [13] to other kernel-based algorithms, namely support vector regression (SVR) [16], multiclass SVM (MCSVM), and one-class SVM (OCSVM). Note that any kernel machine that has a hyperplane-based decision function can be localized by replacing $\langle \boldsymbol{w},\Phi(\boldsymbol{x})\rangle$ with $\sum_{m=1}^P \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m,\Phi_m(\boldsymbol{x}^m)\rangle$ and deriving the corresponding update rules.

2.4.1. Support vector regression

We can also apply the localized kernel idea to ϵ-tube SVR [16]. The decision function is rewritten as

$$f(\boldsymbol{x})=\sum_{m=1}^P \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m,\Phi_m(\boldsymbol{x}^m)\rangle + b$$

and the modified primal optimization problem is

$$\begin{aligned}
\text{min.}\quad & \frac{1}{2}\sum_{m=1}^P \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^N(\xi_i^+ + \xi_i^-)\\
\text{w.r.t.}\quad & \boldsymbol{w}_m\in\mathbb{R}^{S_m},\ \boldsymbol{\xi}^+\in\mathbb{R}_+^N,\ \boldsymbol{\xi}^-\in\mathbb{R}_+^N,\ \mathbf{V}\in\mathbb{R}^{P\times(D_\mathcal{G}+1)},\ b\in\mathbb{R}\\
\text{s.t.}\quad & \epsilon+\xi_i^+\geq y_i-f(\boldsymbol{x}_i)\quad \forall i\\
& \epsilon+\xi_i^-\geq f(\boldsymbol{x}_i)-y_i\quad \forall i
\end{aligned}$$

where $\{\boldsymbol{\xi}^+,\boldsymbol{\xi}^-\}$ are the vectors of slack variables and ϵ is the width of the regression tube. For a given $\mathbf{V}$, the corresponding dual formulation is

$$\begin{aligned}
\text{max.}\quad & J(\mathbf{V})=\sum_{i=1}^N y_i(\alpha_i^+-\alpha_i^-)-\epsilon\sum_{i=1}^N(\alpha_i^++\alpha_i^-)\\
& -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N(\alpha_i^+-\alpha_i^-)(\alpha_j^+-\alpha_j^-)k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)\\
\text{w.r.t.}\quad & \boldsymbol{\alpha}^+\in[0,C]^N,\ \boldsymbol{\alpha}^-\in[0,C]^N\\
\text{s.t.}\quad & \sum_{i=1}^N(\alpha_i^+-\alpha_i^-)=0
\end{aligned}$$

and the resulting decision function is

$$f(\boldsymbol{x})=\sum_{i=1}^N(\alpha_i^+-\alpha_i^-)k_\eta(\boldsymbol{x}_i,\boldsymbol{x})+b.$$

The same learning algorithm given for two-class classification problems can be applied to regression problems by simply replacing Υ_i^j in gradient-descent of the gating model (see Section 2.3) with $(\alpha_i^+-\alpha_i^-)(\alpha_j^+-\alpha_j^-)$.

2.4.2. Multiclass support vector machine

In a multiclass classification problem, a data instance can belong to one of K classes and the class label is given as $y_i\in\{1,2,\ldots,K\}$. There are two basic approaches in the literature to solve multiclass problems. In the multimachine approach, the original multiclass problem is converted to a number of independent, uncoupled two-class problems. In the single-machine approach, the constraints due to having multiple classes are coupled in a single formulation [33].

We can easily apply LMKL to the multimachine approach by solving (8) for each two-class problem separately. In such a case, we obtain different gating models parameters and hence, different kernel weighing strategies for each of the problems. Another possibility is to solve these uncoupled problems separately but learn a common gating model; a similar approach is used for obtaining common kernel weights in MKL for multiclass problems [29].

For the single-machine approach, for class l, we write the discriminant function as follows:

$$f^l(\boldsymbol{x})=\sum_{m=1}^P \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m^l,\Phi_m(\boldsymbol{x}^m)\rangle + b^l.$$

The modified primal optimization problem is

$$\begin{aligned}
\text{min.}\quad & \frac{1}{2}\sum_{m=1}^P\sum_{l=1}^K \|\boldsymbol{w}_m^l\|_2^2 + C\sum_{i=1}^N\sum_{l=1}^K \xi_i^l\\
\text{w.r.t.}\quad & \boldsymbol{w}_m^l\in\mathbb{R}^{S_m},\ \boldsymbol{\xi}^l\in\mathbb{R}_+^N,\ \mathbf{V}\in\mathbb{R}^{P\times(D_\mathcal{G}+1)},\ b^l\in\mathbb{R}\\
\text{s.t.}\quad & f^{y_i}(\boldsymbol{x}_i)-f^l(\boldsymbol{x}_i)\geq 2-\xi_i^l\quad \forall(i,l\neq y_i)\\
& \xi_i^{y_i}=0\quad \forall i.
\end{aligned}$$

We can obtain the dual formulation for a given $\mathbf{V}$ by following the same derivation steps:

$$\begin{aligned}
\text{max.}\quad & J(\mathbf{V})=2\sum_{i=1}^N\sum_{l=1}^K\alpha_i^l-\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\delta_{y_i}^{y_j}A_iA_jk_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)\\
& +\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\sum_{l=1}^K\alpha_i^l(2\alpha_j^{y_i}-\alpha_j^l)k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)\\
\text{w.r.t.}\quad & \boldsymbol{\alpha}^l\in\mathbb{R}_+^N\\
\text{s.t.}\quad & \sum_{i=1}^N\alpha_i^l-\sum_{i=1}^N\delta_{y_i}^lA_i=0\quad \forall l\\
& (1-\delta_{y_i}^l)C\geq\alpha_i^l\geq 0\quad \forall(i,l)
\end{aligned}$$

where $A_i=\sum_{l=1}^K\alpha_i^l$. The resulting discriminant functions that use the locally combined kernel function are given as

$$f^l(\boldsymbol{x})=\sum_{i=1}^N(\delta_{y_i}^lA_i-\alpha_i^l)k_\eta(\boldsymbol{x}_i,\boldsymbol{x})+b^l.$$

Υ_i^j should be replaced with $(\delta_{y_i}^{y_j}A_iA_j-\sum_{l=1}^K\alpha_i^l(2\alpha_j^{y_i}-\alpha_j^l))$ in learning the gating model parameters for multiclass classification problems.

2.4.3. One-class support vector machine

OCSVM is a discriminative method proposed for novelty detection problems [30]. The task is to learn the smoothest hyperplane that puts most of the training instances to one side of the hyperplane while allowing other instances remaining on the other side with a cost. In the localized version, we rewrite the discriminant function as

$$f(\boldsymbol{x})=\sum_{m=1}^P \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m,\Phi_m(\boldsymbol{x}^m)\rangle + b,$$

and the modified primal optimization problem is

$$\text{min.} \quad \frac{1}{2}\sum_{m=1}^{P}\|\boldsymbol{w}_m\|_2^2+C\sum_{i=1}^{N}\xi_i+b$$

$$\text{w.r.t.} \quad \boldsymbol{w}_m\in\mathbb{R}^{S_m},\ \boldsymbol{\xi}\in\mathbb{R}_+^N,\ \mathbf{V}\in\mathbb{R}^{P\times(D_\varphi+1)},\ b\in\mathbb{R}$$

$$\text{s.t.} \quad f(\boldsymbol{x}_i)+\xi_i\geq 0 \quad \forall i.$$

For a given $\mathbf{V}$, we obtain the following dual optimization problem:

$$\text{max.} \quad J(\mathbf{V})=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

$$\text{w.r.t.} \quad \boldsymbol{\alpha}\in[0,C]^N$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i=1$$

and the resulting discriminant function is

$$f(\boldsymbol{x})=\sum_{i=1}^{N}\alpha_i k_\eta(\boldsymbol{x}_i,\boldsymbol{x})+b.$$

In the learning algorithm, Υ_i^j should be replaced with $\alpha_i\alpha_j$ when calculating the gradients with respect to the gating model parameters.

3. Experiments

In this section, we report empirical performance of LMKL for classification and regression problems on several data sets and compare LMKL with SVM, SVR, and MKL (using the linear formulation of [4]). We use our own implementations[1] of SVM, SVR, MKL, and LMKL written in MATLAB and the resulting optimization problems for all these methods are solved using the MOSEK optimization software [26].

Except otherwise stated, our experimental methodology is as follows: A random one-third of the data set is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate ten training and validation sets, with stratification (i.e., preserving class ratios) for classification problems. The validation sets of all folds are used to optimize C by trying values {0.01, 0.1, 1, 10, 100} and for regression problems, ϵ, the width of the error tube. The best configuration (measured as the highest average classification accuracy or the lowest mean square error (MSE) for regression problems) on the validation folds is used to train the final classifiers/regressors on the training folds and their performance is measured over the test set. We have 10 test set results, and we report their averages and standard deviations, as well as the percentage of instances stored as support vectors and the total training time (in seconds) including the cross-validation. We use the 5×2 cv paired F test for comparison [2]. In the experiments, we normalize the kernel matrices to unit diagonal before training.

3.1. Classification experiments

3.1.1. Illustrative classification problem

In order to illustrate our proposed algorithm, we use the toy data set Gauss4 [13] consisting of 1200 data instances generated from four Gaussian components (two for each class) with the following prior probabilities, mean vectors and covariance matrices:

$$p_{11}=0.25 \quad \mu_{11}=\begin{pmatrix}-3.0\\+1.0\end{pmatrix} \quad \Sigma_{11}=\begin{pmatrix}0.8 & 0.0\\0.0 & 2.0\end{pmatrix}$$

$$p_{12}=0.25 \quad \mu_{12}=\begin{pmatrix}+1.0\\+1.0\end{pmatrix} \quad \Sigma_{12}=\begin{pmatrix}0.8 & 0.0\\0.0 & 2.0\end{pmatrix}$$

$$p_{21}=0.25 \quad \mu_{21}=\begin{pmatrix}-1.0\\-2.2\end{pmatrix} \quad \Sigma_{21}=\begin{pmatrix}0.8 & 0.0\\0.0 & 4.0\end{pmatrix}$$

$$p_{22}=0.25 \quad \mu_{22}=\begin{pmatrix}+3.0\\-2.2\end{pmatrix} \quad \Sigma_{22}=\begin{pmatrix}0.8 & 0.0\\0.0 & 4.0\end{pmatrix}$$

where data instances from the first two components are labeled as positive and others are labeled as negative.

First, we train both MKL and LMKL with softmax gating to combine a linear kernel, k_L, and a second-degree polynomial kernel, k_P ($q=2$). Fig. 1(b) shows the classification boundaries calculated and the support vectors stored on one of the training folds by MKL that assigns combination weights 0.32 and 0.68 to k_L and k_P, respectively. We see that using the kernel matrix obtained by combining k_L and k_P with these weights, we do not achieve a good approximation to the optimal Bayes' boundary. As we see in Fig. 1(c), LMKL divides the input space into two regions and uses the polynomial kernel to separate one component from two others quadratically in one region and the linear kernel for the other component in the other region. We see that we get a very good approximation of the optimal Bayes' boundary. The softmax function in the gating model achieves a smooth transition between the two kernels. The superiority of the localized approach is also apparent in the smoothness of the fit that uses fewer support vectors: MKL achieves 90.95 ± 0.61 per cent average test accuracy by storing 38.23 ± 2.34 per cent of training instances as support vectors, whereas LMKL achieves 91.83 ± 0.24 per cent average test accuracy by storing 25.13 ± 0.91 per cent support vectors.

With LMKL, we can also combine multiple copies of the same kernel, as shown in Fig. 1(d), which shows the classification and gating model boundaries of LMKL using three linear kernels and approximates the optimal Bayes' boundary in a piecewise linear manner. For this configuration, LMKL achieves 91.78 ± 0.55 per cent average test accuracy by storing 23.83 ± 1.20 per cent support vectors. Instead of using complex kernels such as polynomial kernels of high-degree or the Gaussian kernel, local combination of simple kernels (e.g., linear or low-degree polynomial kernels) can produce accurate classifiers and avoid overfitting. Fig. 2 shows the average test accuracies, support vector percentages, and training times with one standard deviation for LMKL with different number of linear kernels. We see that even if we provide more kernels than needed, LMKL uses only as many support vectors as required and does not overfit. LMKL obtains nearly the same average test accuracies and support vector percentages with three or more linear kernels. We also see that the training time of LMKL is increasing linearly with increasing number of kernels.

3.1.2. Combining multiple feature representations of benchmark data sets

We compare SVM, MKL, and LMKL in terms of classification performance, model complexity (i.e., stored support vector percentage), and training time. We train SVMs with linear kernels calculated on each feature representation separately. We also train an SVM with a linear kernel calculated on the concatenation of all feature representations, which is referred to as All. MKL and LMKL combine linear kernels calculated on each feature representation. LMKL uses a single feature representation or the concatenation of all feature representations in the gating model. We use both softmax and sigmoid gating models in our experiments.

We perform experiments on the Multiple Features (MultiFeat) digit recognition data set[2] from the UCI Machine Learning Repository,

[1] Available at http://www.cmpe.boun.edu.tr/~gonen/lmkl.

[2] Available at http://archive.ics.uci.edu/ml/datasets/Multiple+Features.

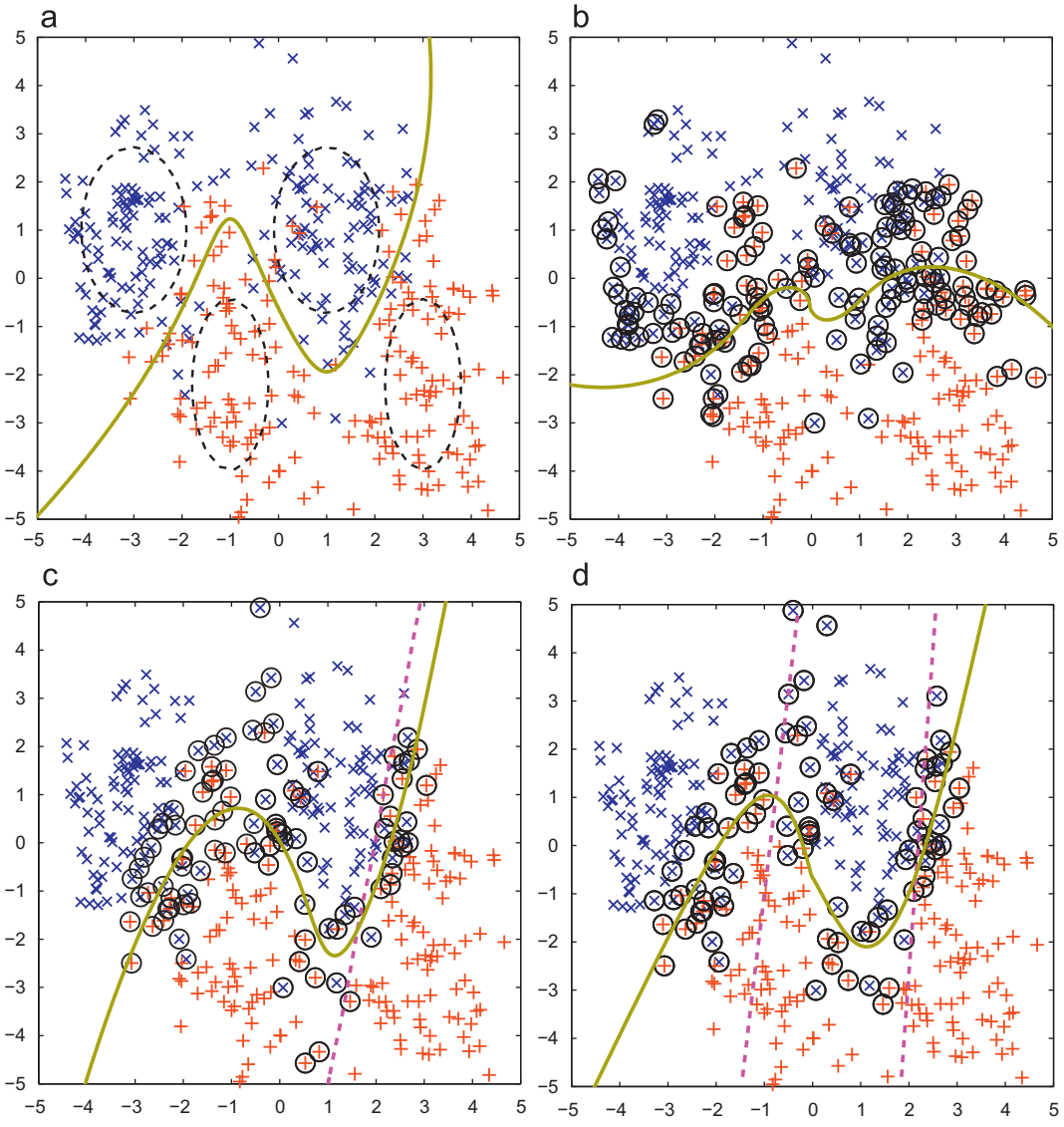

Fig. 1. MKL and LMKL solutions on the GAUSS4 data set. (a) The dashed ellipses show the Gaussians from which data are sampled and the solid line shows the optimal Bayes' discriminant. (b)–(d) The solid lines show the discriminants learned. The circled data points represent the support vectors stored. For LMKL solutions, the dashed lines shows the gating boundaries, where the gating model outputs of neighboring kernels are equal. (a) GAUSS4 data set. (b) MKL with (k_L-k_P). (c) LMKL with (k_L-k_P). (d) LMKL with (k_L-k_L-k_L).

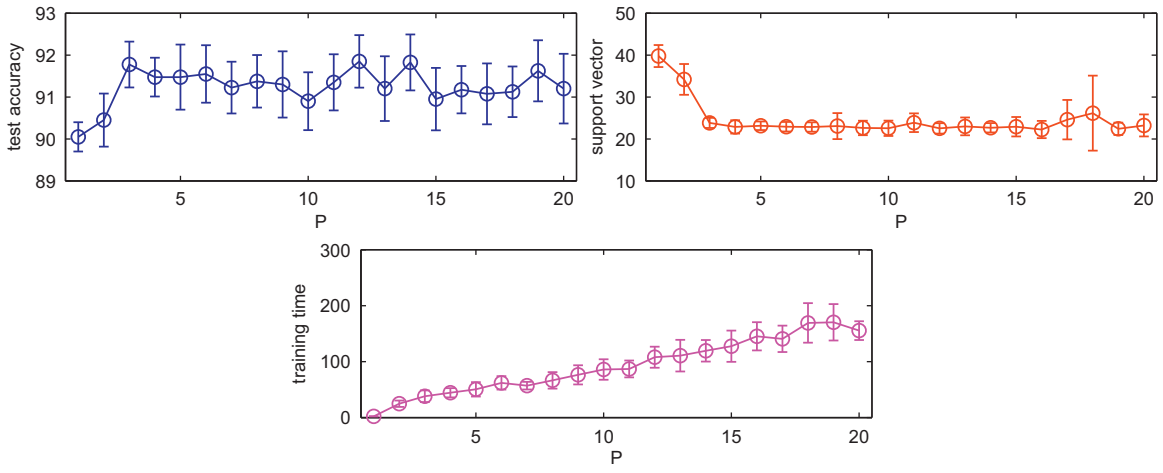

Fig. 2. The average test accuracies, support vector percentages, and training times on the GAUSS4 data set obtained by LMKL with multiple copies of linear kernels and softmax gating.

composed of six different data representations for 2000 handwritten numerals. The properties of these feature representations are summarized in Table 1. A binary classification problem is generated from the MultiFeat data set to separate small ('0'–'4') digits from large ('5'–'9') digits. We use the concatenation of all feature representations in the gating model for this data set.

Table 2 lists the classification results on the MultiFeat data set obtained by SVM, MKL, and LMKL. We see that SVM (All) is significantly more accurate than the best SVM with single feature representation, namely SVM (Fac), but with a significant increase in the number of support vectors. MKL is as accurate as SVM (All) but stores significantly more support vectors. LMKL with softmax gating is as accurate as SVM (All) using significantly fewer support vectors. LMKL with sigmoid gating is significantly more accurate than MKL, SVM (All), and single kernel SVMs. It stores significantly fewer support vectors than MKL and SVM (All), and ties with SVM (Fac). For the MultiFeat data set, the average kernel weights and the average number of active kernels (whose gating values are nonzero) calculated on the test set are given in Table 3. We see that both LMKL with softmax gating and LMKL with sigmoid gating use fewer kernels than MKL in the decision function. MKL uses all kernels with the same weight for all inputs; LMKL uses a different smaller subset for each input. By storing significantly fewer support vectors and using fewer active kernels, LMKL is significantly faster than MKL in the testing phase.

MKL and LMKL are iterative methods and need to solve SVM problems at each iteration. LMKL also needs to update the gating parameters and that is why it requires significantly longer training times than MKL when the dimensionality of the gating model representation is high (649 in this set of experiments)—LMKL needs to calculate the gradients of (8) with respect to the parameters of the gating model and to perform a line search using these gradients. Learning with sigmoid gating is faster than softmax gating because with the sigmoid during the gradient-update only a single value is used and updating takes $\mathcal{O}(P)$ time, whereas with the softmax, all gating outputs are used and updating is $\mathcal{O}(P^2)$. When learning time is critical, the time complexity of this step can be reduced by decreasing the dimensionality of the gating model representation using an unsupervised dimensionality reduction method. Note also that both the output calculations and the gradients in separate kernels can be efficiently parallelized when parallel hardware is available.

Instead of combining different feature representations, we can combine multiple copies of the same feature representation with LMKL. We combine multiple copies of linear kernels on the single best Fac representation using the sigmoid gating model on the same representation (see Fig. 3). Even if we increase accuracy (not significantly) by increasing the number of copies of the kernels compared to SVM (Fac), we could not achieve the performance obtained by combining different representations with sigmoid gating.

Table 1
Multiple feature representations in the MultiFeat data set.

Name	Dimension	Data source
Fac	216	Profile correlations
Fou	76	Fourier coefficients of the shapes
Kar	64	Karhunen–Loève coefficients
Mor	6	Morphological features
Pix	240	Pixel averages in 2×3 windows
Zer	47	Zernike moments

Table 2
Classification results on the MultiFeat data set.

Method	Test accuracy	Support vector	Training time (s)
SVM (Fac)	94.97 ± 0.87	17.93 ± 0.91	5.42 ± 0.20
SVM (Fou)	90.54 ± 1.11	28.90 ± 1.69	5.47 ± 0.26
SVM (Kar)	88.13 ± 0.73	33.62 ± 1.31	5.58 ± 0.56
SVM (Mor)	69.61 ± 0.14	61.90 ± 0.49	6.50 ± 0.65
SVM (Pix)	89.42 ± 0.65	46.35 ± 1.64	5.71 ± 0.62
SVM (Zer)	89.12 ± 0.63	26.27 ± 1.67	5.28 ± 0.35
SVM (All)	97.69 ± 0.44	23.36 ± 1.15	3.65 ± 0.32
MKL	97.40 ± 0.37	32.59 ± 0.82	38.90 ± 3.83
LMKL (softmax)	97.69 ± 0.44	15.06 ± 1.03	5568.51 ± 1590.39
LMKL (sigmoid)	98.58 ± 0.41	15.27 ± 0.92	1639.03 ± 278.17
LMKL (6 Fac and sigmoid)	97.03 ± 0.67	18.52 ± 2.47	1190.74 ± 562.49

Table 3
Average kernel weights and number of active kernels on the MultiFeat data set.

Method	Fac	Fou	Kar	Mor	Pix	Zer
MKL	0.2466	0.2968	0.0886	0.1464	0.1494	0.0722
LMKL (softmax)	0.1908	0.1333	0.1492	0.3335	0.1134	0.0797
LMKL (sigmoid)	0.5115	0.5192	0.5429	0.5566	0.5274	0.5132

The average numbers of active kernels are 6.00, 2.43, and 5.36, respectively.

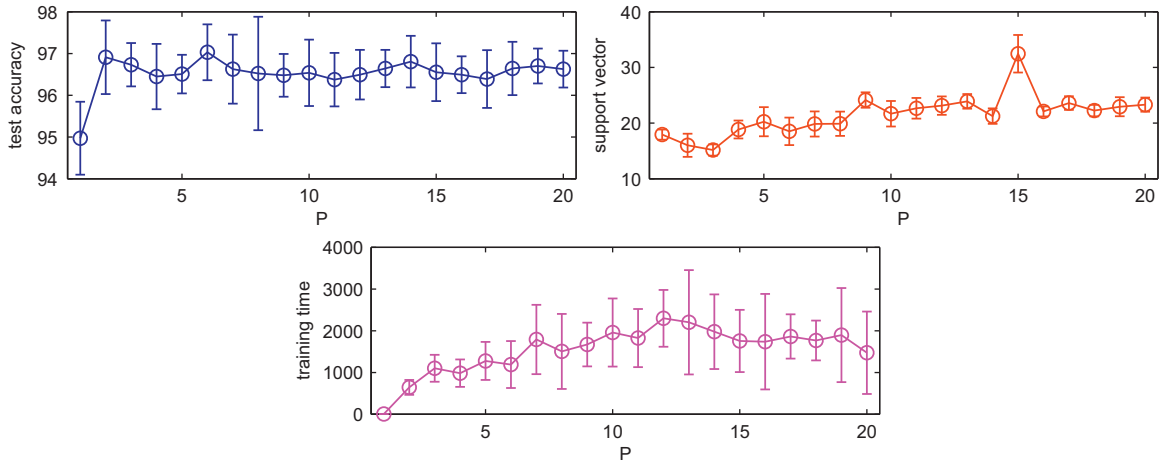

Fig. 3. The average test accuracies, support vector percentages, and training times on the MultiFeat data set obtained by LMKL with multiple copies of linear kernels and sigmoid gating on the Fac representation.

Table 4
Multiple feature representations in the Advert data set.

Name	Dimension	Data source
URL	457	Phrases occurring in the URL
OrigURL	495	Phrases occurring in the URL of the image
AncURL	472	Phrases occurring in the anchor text
Alt	111	Phrases occurring in the alternative text
Caption	19	Phrases occurring in the caption terms

Table 5
Classification results on the Advert data set.

Method	Test accuracy	Support vector	Training time (s)
SVM (URL)	94.67 ± 0.24	83.32 ± 1.89	26.42 ± 1.27
SVM (OrigURL)	92.04 ± 0.26	96.16 ± 0.51	23.34 ± 0.96
SVM (AncURL)	95.45 ± 0.31	64.90 ± 5.41	26.65 ± 1.42
SVM (Alt)	89.64 ± 0.38	87.73 ± 1.17	22.52 ± 0.77
SVM (Caption)	86.60 ± 0.09	96.65 ± 0.42	20.02 ± 0.75
SVM (All)	96.43 ± 0.24	41.99 ± 1.76	31.90 ± 2.59
MKL	96.32 ± 0.50	35.82 ± 4.35	101.42 ± 4.89
LMKL (softmax)	95.78 ± 0.46	41.72 ± 11.59	914.42 ± 162.14
LMKL (sigmoid)	96.72 ± 0.46	34.40 ± 1.51	819.61 ± 111.22
LMKL (5 AncURL and sigmoid)	95.66 ± 0.29	10.87 ± 1.07	1287.67 ± 405.55

For example, LMKL with sigmoid gating and kernels over six different feature representations is better than LMKL with sigmoid gating and six copies of the kernel over the Fac representation in terms of both classification accuracy (though not significantly) and the number of support vectors stored (significantly) (see Table 2). We also see that the training time of LMKL is increasing (though not monotonically) with increasing number of kernels.

We also perform experiments on the Internet Advertisements (Advert) data set[3] from the UCI Machine Learning Repository, composed of five different feature representations (different bags of words) with some additional geometry information of the images, which is ignored in our experiments due to missing values. The properties of these feature representations are summarized in Table 4. The classification task is to predict whether an image is an advertisement or not. We use the Caption representation in the gating model due to its lower dimensionality compared to the other representations.

Table 5 gives the classification results on the Advert data set obtained by SVM, MKL, and LMKL. We see that SVM (All) is significantly more accurate than the best SVM with single feature representation, namely SVM (AncURL), and uses significantly fewer support vectors. MKL has comparable classification accuracy to SVM (All) and the difference between the number of support vectors is not significant. LMKL with softmax/sigmoid gating has comparable accuracy to MKL and SVM (All). LMKL with sigmoid gating stores significantly fewer support vectors than SVM (All). The average kernel weights and the average number of active kernels on the Advert data set are given in Table 6. The difference between the running times of MKL and LMKL is not as significant as on the MultiFeat data set because the gating model representation (Caption) has only 19 dimensions. Different from the MultiFeat data set, LMKL uses approximately the same number of or more kernels compared to MKL on this data set. (On one of the ten folds, MKL chooses five and on the remaining nine folds, it chooses four kernels, leading to an average of 4.1.)

Table 6
Average kernel weights and number of active kernels on the Advert data set.

Method	URL	OrigURL	AncURL	Alt	Caption
MKL	0.3073	0.1600	0.3497	0.1828	0.0003
LMKL (softmax)	0.3316	0.0160	0.6292	0.0172	0.0060
LMKL (sigmoid)	0.9918	0.9820	0.9900	0.9913	0.4027

The average numbers of active kernels are 4.10, 4.04, and 4.96, respectively.

When we combine multiple copies of linear kernels on the AncURL representation with LMKL using the sigmoid gating model on the same representation (see Fig. 4), we see that LMKL stores much fewer support vectors compared to the single kernel SVM (AncURL) without sacrificing from accuracy. But, as before on the MultiFeat data set, we could not achieve the classification accuracy obtained by combining different representations with sigmoid gating. For example, LMKL with sigmoid gating and kernels over five different feature representations is significantly better than LMKL with sigmoid gating and five copies of the kernel over the AncURL representation in terms of classification accuracy but the latter stores significantly fewer support vectors (see Table 5). We again see that the training time of LMKL is increasing linearly with increasing number of kernels.

3.1.3. Combining multiple input patches for image recognition problems

For image recognition problems, only some parts of the images contain meaningful information and it is not necessary to examine the whole image in detail. Instead of defining kernels over the whole input image, we can divide the image into non-overlapping patches and use separate kernels in these patches. The kernels calculated on the parts with relevant information take nonzero weights and the kernels over the non-relevant patches are ignored. We use a low-resolution (simpler) version of the image as input to the gating model, which selects a subset of the high-resolution localized kernels. In such a case, it is not a good idea to use softmax gating in LMKL because softmax gating would choose one or very few patches and a patch by itself does not carry enough discriminative information.

We train SVMs with linear kernels calculated on the whole image in different resolutions. MKL and LMKL combine linear kernels calculated on each image patch. LMKL uses the whole image with different resolutions in the gating model [14].

We perform experiments on the Olivetti data set,[4] which consists of 10 different 64×64 grayscale images of 40 subjects. We construct a two-class data set by combining male subjects (36 subjects) into one class versus female subjects (four subjects) in another class. Our experimental methodology for this data set is slightly different: We select two images of each subject randomly and reserve these total 80 images as the test set. Then, we apply 8-fold cross-validation on the remaining 320 images by putting one image of each subject to the validation set at each fold. MKL and LMKL combine 16 linear kernels calculated on image patches of size 16×16.

Table 7 shows the results of MKL and LMKL combining kernels calculated over non-overlapping patches of face images. MKL achieves significantly higher classification accuracy than all single kernel SVMs except in 32×32 resolution. LMKL with softmax gating has comparable classification accuracy to MKL and stores significantly fewer support vectors when 4×4 or 16×16 images

[3] Available at http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements.

[4] Available at http://cs.nyu.edu/~roweis/data.html.

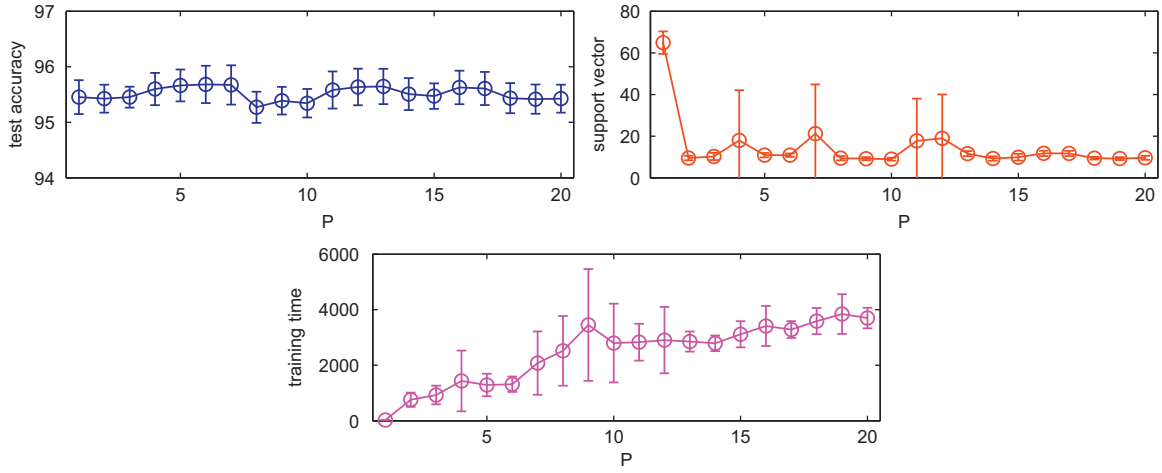

Fig. 4. The average test accuracies, support vector percentages, and training times on the ADVERT data set obtained by LMKL with multiple copies of linear kernels and sigmoid gating on the ANCURL representation.

Table 7
Classification results on the OLIVETTI data set.

Method	Test accuracy	Support vector	Training time (s)
SVM ($\boldsymbol{x}=4\times4$)	93.28 ± 0.65	21.70 ± 0.93	1.67 ± 0.19
SVM ($\boldsymbol{x}=8\times8$)	97.50 ± 1.16	20.13 ± 1.04	1.21 ± 0.22
SVM ($\boldsymbol{x}=16\times16$)	97.03 ± 0.93	19.91 ± 1.01	1.24 ± 0.12
SVM ($\boldsymbol{x}=32\times32$)	97.97 ± 1.48	23.71 ± 1.39	2.05 ± 0.22
SVM ($\boldsymbol{x}=64\times64$)	97.66 ± 1.41	25.94 ± 1.01	2.10 ± 0.18
MKL	99.06 ± 0.88	22.19 ± 1.00	11.53 ± 0.82
LMKL (softmax and $\boldsymbol{x}^G=4\times4$)	97.19 ± 2.81	16.65 ± 3.34	83.13 ± 32.09
LMKL (softmax and $\boldsymbol{x}^G=8\times8$)	97.19 ± 2.48	22.54 ± 4.56	139.73 ± 49.78
LMKL (softmax and $\boldsymbol{x}^G=16\times16$)	99.22 ± 1.33	16.38 ± 1.50	405.61 ± 153.25
LMKL (sigmoid and $\boldsymbol{x}^G=4\times4$)	99.22 ± 0.93	22.72 ± 1.83	45.39 ± 5.62
LMKL (sigmoid and $\boldsymbol{x}^G=8\times8$)	99.84 ± 0.44	26.88 ± 2.24	64.55 ± 14.99
LMKL (sigmoid and $\boldsymbol{x}^G=16\times16$)	99.38 ± 1.34	21.65 ± 1.44	81.03 ± 36.59

are used in the gating model. This is mainly due to the normalization property of softmax gating that generally activates a single patch and ignores the others; this uses fewer support vectors but is not as accurate. LMKL with sigmoid gating significantly improves the classification accuracy over MKL by looking at the 8×8 images in the gating model and choosing a subset of the high-resolution patches. We see that the training time of LMKL is monotonically increasing with the dimensionality of the gating model representation.

Fig. 5 illustrates the example uses of MKL and LMKL with softmax and sigmoid gating. Fig. 5(b)–(c) show the combination weights found by MKL and sample face images stored as support vectors weighted with those. MKL uses the same weights over the whole input space and thereby the parts whose weights are nonzero are used in the decision process for all subjects. When we look at the results of LMKL, we see that the gating model activates important parts of each face image and these parts are used in the classifier with nonzero weights, whereas the parts whose gating model outputs are zero are not considered. That is, looking at the output of the gating model, we can skip processing the high-resolution versions of these parts. This can be considered similar to a selective attention mechanism whereby the gating model defines a saliency measure and drives a high-resolution "fovea"/"eye" to consider only regions of high saliency [1]. For example, if we use LMKL with softmax gating (see Fig. 5(d)–(f)), the gating model generally activates a single patch containing a part of eyes or eyebrows depending on the subject. This may not be enough for good discrimination and using sigmoid gating is more appropriate. When we use LMKL with sigmoid gating (see Fig. 5(g)–(i)), multiple patches are given nonzero weights in a data-dependent way.

Fig. 6 gives the average kernel weights on the test set for MKL, LMKL with softmax gating, and LMKL with sigmoid gating. We see that MKL and LMKL with softmax gating use fewer high-resolution patches than LMKL with sigmoid gating.

We can generalize this idea even further: Let us say that we have a number of information sources that are costly to extract or process, and a relatively simpler one. In such a case, we can feed the simple representation to the gating model and feed the costly representations to the actual kernels and train LMKL. The gating model then chooses a costly representation only when it is needed and chooses only a subset of the costly representations. Note that the representation used by the gating model does not need to be very precise, because it does not do the actual decision, but only chooses the representation(s) that do the actual decision.

3.2. Regression experiments

3.2.1. Illustrative regression problem

We illustrate the applicability of LMKL to regression problems on the MOTORCYCLE data set discussed in [32]. We train LMKL with three linear kernels and softmax gating ($C=1000$ and $\epsilon=16$) using 10-fold cross-validation. Fig. 7 shows the average of global and local fits obtained for these 10 folds. We learn a piecewise linear fit through three local models that are obtained using linear kernels in each region and we combine them using the softmax gating model (shown by dashed lines). The softmax gating model divides the input space between kernels, generally selects a single kernel to use, and also ensures a smooth transition between local fits.

3.2.2. Combining multiple kernels for benchmark data sets

We compare SVR and LMKL in terms of regression performance (i.e., mean square error), model complexity (i.e., stored support vector percentage), and training time. We train SVRs with different kernels, namely linear kernel and polynomial kernels up to fifth degree. LMKL combines these five kernels with both softmax and sigmoid gating models.

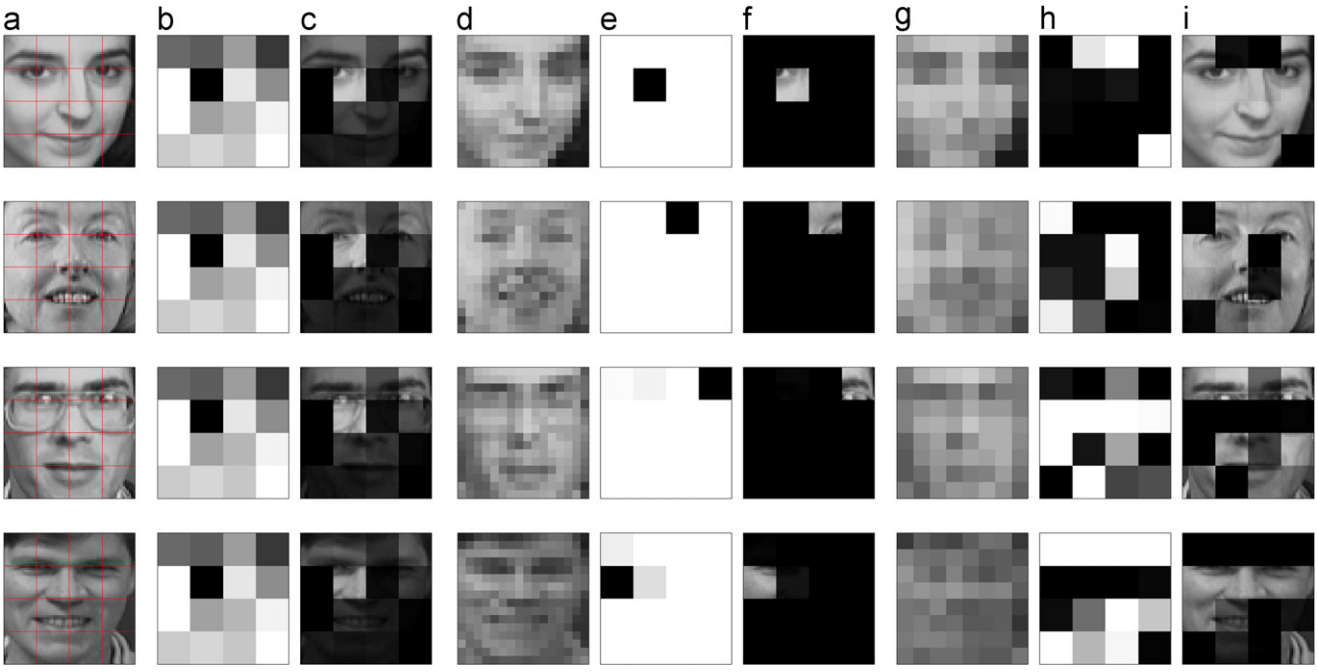

Fig. 5. Example uses of MKL and LMKL on the Olivetti data set. (a) $\Phi_m(\boldsymbol{x}^m)$: features fed into kernels, (b) η_m: combination weights, and (c) $\eta_m\Phi_m(\boldsymbol{x}^m)$: features weighted with combination weights, (d) $\boldsymbol{x}^g$: features fed into softmax gating model, (e) $\eta_m(\boldsymbol{x}|\mathbf{V})$: softmax gating model outputs, (f) $\eta_m(\boldsymbol{x}|\mathbf{V})\Phi_m(\boldsymbol{x}^m)$: features weighted with softmax gating model outputs, (g) $\boldsymbol{x}^g$: features fed into sigmoid gating model, (h) $\eta_m(\boldsymbol{x}|\mathbf{V})$: sigmoid gating model outputs, and (i) $\eta_m(\boldsymbol{x}|\mathbf{V})\Phi_m(\boldsymbol{x}^m)$: features weighted with sigmoid gating model outputs.

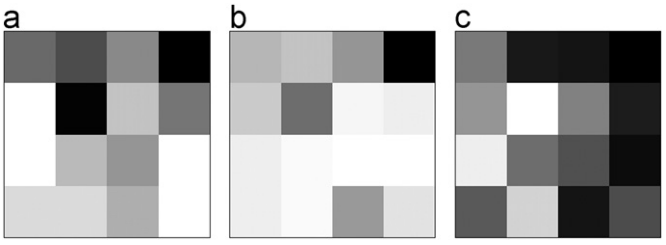

Fig. 6. Average kernel weights on the Olivetti data set. (a) MKL, (b) LMKL with softmax gating on 16×16 resolution, and (c) LMKL with sigmoid gating on 8×8 resolution.

We perform experiments on the Concrete Compressive Strength (Concrete) data set[5] and the Wine Quality (WhiteWine) data set[6] from the UCI Machine Learning Repository. ϵ is selected from {1, 2, 4, 8, 16} for the Concrete data set and {0.08, 0.16, 0.32, 0.64, 1.28} for the WhiteWine data set.

Table 8 lists the regression results on the Concrete data set obtained by SVR and LMKL. We see that both LMKL with softmax gating and LMKL with sigmoid gating are significantly more accurate than all of the single kernel SVRs. LMKL with softmax gating uses k_L, k_P ($q=4$), and k_P ($q=5$) with relatively higher weights but LMKL with sigmoid gating uses all of the kernels with significant weights (see Table 9). When we combine multiple copies of the linear kernel using the softmax gating model (shown in Fig. 8), we see that LMKL does not overfit and we get significantly lower error than the best single kernel SVR (k_P and $q=3$). For example, LMKL with five copies of k_L and softmax gating gets significantly lower error than SVR (k_P and $q=3$) and stores significantly fewer support vectors. Similar to the binary classification results, the training time of LMKL is increasing linearly with increasing number of kernels.

[5] Available at http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.

[6] Available at http://archive.ics.uci.edu/ml/datasets/Wine+Quality.

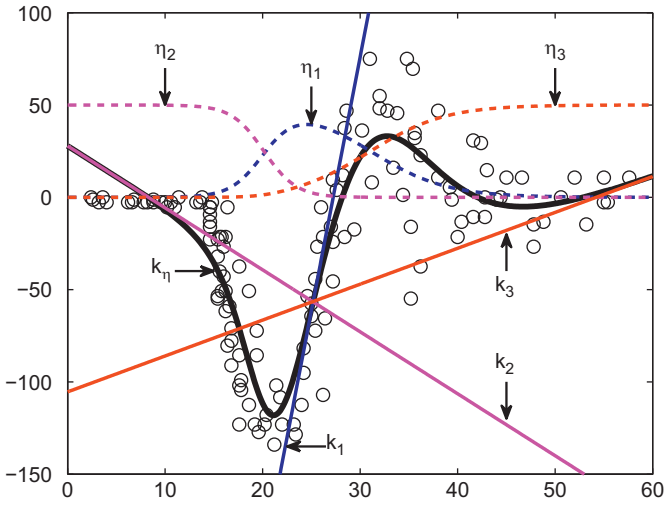

Fig. 7. Global and local fits (solid lines) obtained by LMKL with three linear kernels and softmax gating on the Motorcycle data set. The dashed lines show gating model outputs, which are multiplied by 50 for visual clarity.

Table 10 lists the regression results on the WhiteWine data set obtained by SVR and LMKL. We see that both LMKL with softmax gating and LMKL with sigmoid gating obtain significantly less error than SVR (k_L), SVR (k_P and $q=2$), and SVR (k_P and $q=3$), and have comparable error to SVR (k_P and $q=4$) and SVR (k_P and $q=5$) but store significantly fewer support vectors than all single kernel SVRs. Even if we do not decrease the error, we learn computationally simpler models by storing much fewer support vectors. We see from Table 11 that LMKL with softmax gating assigns relatively higher weights to k_L, k_P ($q=3$), and k_P ($q=5$), whereas LMKL with sigmoid gating uses the polynomial kernels nearly everywhere in the input space and the linear kernel for some of the test instances.

4. Discussion

We discuss the key properties of the proposed method and compare it with similar MKL methods in the literature.

4.1. Computational complexity

When we are training LMKL, we need to solve a canonical kernel machine problem with the combined kernel obtained with the current gating model parameters and calculate the gradients of $J(\mathbf{V})$ at each iteration. The gradients calculations are made using the support vectors of the current iteration. The gradient calculation step has lower time complexity compared to the kernel machine solver when the gating model representation is low-dimensional. If we have a high-dimensional gating model representation, we can apply an unsupervised dimensionality reduction method (e.g., principal component analysis) on this representation in order to decrease the training time. The computational complexity of LMKL also depends on the complexity of the canonical kernel machine solver used in the main loop, which can be reduced using a hot-start procedure (i.e., starting from the previous solution). The number of iterations before convergence clearly depends on the training data and the step size selection procedure. The key issue for faster convergence is to select good gradient-descent step sizes at each iteration. The step size of each iteration should be determined with a line search method (e.g., Armijo's rule whose search procedure allows backtracking and does not use any curve fitting method), which requires solving additional kernel machine problems. Clearly, the time complexity for each iteration increases but the algorithm converges in fewer iterations. In practice, we see convergence in 5–20 iterations.

One main advantage of LMKL is in reducing the time complexity for the testing phase as a result of localization. When calculating the locally combined kernel function, $k_\eta(\boldsymbol{x}_i,\boldsymbol{x})$, in (9), $k_m(\boldsymbol{x}_i^m,\boldsymbol{x}^m)$ needs to be evaluated or calculated only if both $\eta_m(\boldsymbol{x}_i)$ and $\eta_m(\boldsymbol{x})$ are active (i.e., nonzero).

4.2. Knowledge extraction

The kernel weights obtained by MKL can be used to extract knowledge about the relative contributions of kernel functions used in combination. Different kernels define different similarity mea-

Table 8
Regression results on the Concrete data set.

Method	MSE	Support vector	Training time (s)
SVR (k_L)	120.61 ± 2.15	44.31 ± 3.46	30.32 ± 3.10
SVR (k_P and $q=2$)	92.57 ± 4.19	36.22 ± 1.24	32.81 ± 8.18
SVR (k_P and $q=3$)	58.32 ± 3.66	73.16 ± 1.21	47.71 ± 3.64
SVR (k_P and $q=4$)	63.83 ± 9.58	52.52 ± 2.40	34.37 ± 5.19
SVR (k_P and $q=5$)	61.26 ± 5.31	52.17 ± 2.23	34.55 ± 3.72
LMKL (softmax)	44.80 ± 6.33	64.28 ± 4.02	818.32 ± 113.65
LMKL (sigmoid)	48.18 ± 5.22	49.20 ± 2.05	575.32 ± 81.75
LMKL (5 k_L and softmax)	53.14 ± 6.42	34.93 ± 8.45	1115.32 ± 137.15

Table 9
Average kernel weights and number of active kernels on the Concrete data set.

Method	k_L	k_P $q=2$	k_P $q=3$	k_P $q=4$	k_P $q=5$
LMKL (softmax)	0.1495	0.0091	0.0117	0.0951	0.7346
LMKL (sigmoid)	0.6675	0.8176	0.9962	0.9721	0.9989

The average numbers of active kernels are 4.52 and 4.68, respectively.

Table 10
Regression results on the WhiteWine data set.

Method	MSE	Support vector	Training time (s)
SVR (k_L)	0.59 ± 0.00	66.83 ± 0.57	1870.13 ± 148.78
SVR (k_P and $q=2$)	0.54 ± 0.01	66.22 ± 0.67	1864.24 ± 166.48
SVR (k_P and $q=3$)	0.54 ± 0.00	66.14 ± 1.13	3622.55 ± 297.24
SVR (k_P and $q=4$)	0.52 ± 0.01	66.55 ± 1.03	4223.70 ± 328.03
SVR (k_P and $q=5$)	0.52 ± 0.01	66.27 ± 1.24	2469.31 ± 221.74
LMKL (softmax)	0.52 ± 0.01	18.66 ± 13.41	60632.51 ± 6905.62
LMKL (sigmoid)	0.51 ± 0.00	38.29 ± 2.34	76249.60 ± 12724.65

Table 11
Average kernel weights and number of active kernels on the WhiteWine data set.

Method	k_L	k_P $q=2$	k_P $q=3$	k_P $q=4$	k_P $q=5$
LMKL (softmax)	0.2238	0.0302	0.1430	0.0296	0.5733
LMKL (sigmoid)	0.5956	0.9698	0.9978	0.9849	0.9929

The average numbers of active kernels are 1.05 and 4.58, respectively.

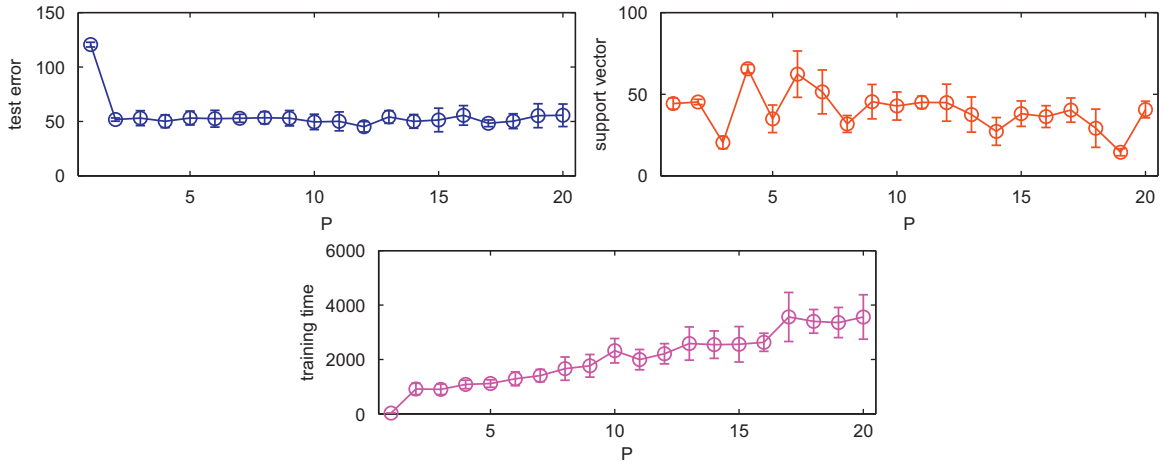

Fig. 8. The average test mean square errors, support vector percentages, and training times on the Concrete data set obtained by LMKL with multiple copies of linear kernels and softmax gating.

sures and we can deduce which similarity measures are appropriate for the task at hand. If kernel functions are evaluated over different feature subsets or feature representations, the important ones have higher combination weights. With our LMKL framework, we can extract similar information for different regions of the input space. This enables us to extract information about kernels (similarity measures), feature subsets, and/or feature representations in a data-dependent manner.

4.3. Regularization

Canonical kernel machines learn sparse models as a result of regularization on the weight vector but the underlying complexity of the kernel function is the main factor for determining the model complexity. The main advantage of LMKL in terms of regularization over canonical kernel machines is the inherent regularization effect on the gating model. When we regularize the sum of the hyperplane weight vectors in (6), because these weight vectors are written in terms of the gating model as in (7), we also regularize the gating model as a side effect. MKL can combine only different kernel functions and more complex kernels are favored over the simpler ones in order to get better performance. However, LMKL can also combine multiple copies of the same kernel and it can dynamically construct a more complex locally combined kernel using the kernels in a data-dependent way. LMKL eliminates some of the kernels by assigning zero weights to the corresponding gating outputs in order to get a more regularized solution. Figs. 2–4 and 8 give empirical support to this regularization effect, where we see that LMKL does not overfit even if we increase the number of kernels up to 20.

4.4. Dimensionality reduction

The localized kernel idea can also be combined with dimensionality reduction. If the training instances have a local structure (i.e., lie on low-dimensional manifolds locally), we can learn low-dimensional local projections in each region, which we can also use for visualization. Previously, it had been proposed to integrate a projection matrix into the discriminant function [6] and we extended this idea to project data instances into different feature spaces using local projection matrices combined with a gating model, and calculate the combined kernel function with the dot product in the combined feature space [17]. The local projection matrices can be learned with the other parameters, as before, using a two-step alternating optimization algorithm.

4.5. Related work

LMKL finds a nonlinear combination of kernel functions with the help of the gating model. The idea of learning a nonlinear combination is also discussed in different studies. For example, a latent variable generative model using the maximum entropy discrimination to learn data-dependent kernel combination weights is proposed in [23]. This method combines a generative probabilistic model with a discriminative large margin method using a log-ratio of Gaussian mixtures as the classifier.

In a more recent work, a nonlinear kernel combination method based on kernel ridge regression and polynomial combination of kernels is proposed [8]

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=\sum_{\boldsymbol{q}\in\mathcal{Q}}\eta_1^{q_1}\dots\eta_P^{q_P}k_1(\boldsymbol{x}_i^1,\boldsymbol{x}_j^1)^{q_1}\dots k_P(\boldsymbol{x}_i^P,\boldsymbol{x}_j^P)^{q_P}$$

where $\mathcal{Q}=\{\boldsymbol{q}:\boldsymbol{q}\in\mathbb{Z}_+^P,\sum_{m=1}^P q_m=d\}$ and the kernel weights are optimized over a positive, bounded, and convex set using a projection-based gradient-descent algorithm.

Similar to LMKL, a Bayesian approach is developed for combining different feature representations in a data-dependent way under the Gaussian process framework [7]. A common covariance function is obtained by combining the covariances of feature representations in a nonlinear manner. This formulation can identify the noisy data instances for each feature representation and prevent them from being used. Classification is performed using the standard Gaussian processes approach with the common covariance function.

Inspired from LMKL, two methods that learn a data-dependent kernel function are used for image recognition applications [34,35]; they differ in their gating models that are constants rather than functions of the input. In [34], the training set is divided into clusters as a preprocessing step and then cluster-specific kernel weights are learned using an alternating optimization method. The combined kernel function can be written as

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=\sum_{m=1}^P\eta_{c_i}^m k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\eta_{c_j}^m$$

where $\eta_{c_i}^m$ corresponds to the weight of kernel $k_m(\cdot,\cdot)$ in the cluster $\boldsymbol{x}_i$ belongs to. The kernel weights of the cluster which a test instance is assigned to are used in the testing phase. In [35], instance-specific kernel weights are used instead of cluster-specific weights. The corresponding combined kernel function is

$$k_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)=\sum_{m=1}^P\eta_i^m k_m(\boldsymbol{x}_i^m,\boldsymbol{x}_j^m)\eta_j^m$$

where η_i^m corresponds to the weight of kernel $k_m(\cdot,\cdot)$ for $\boldsymbol{x}_i$ and instance-specific weights are optimized using an alternating optimization problem for the training set. But, in the testing phase, the kernel weights for a test instance are all taken to be equal.

5. Conclusions

This work introduces a localized multiple kernel learning framework for kernel-based algorithms. The proposed algorithm has two main ingredients: (i) a gating model that assigns weights to kernels for a data instance, (ii) a kernel-based learning algorithm with the locally combined kernel. The training of these two components is coupled and the parameters of both components are optimized together using a two-step alternating optimization procedure. We derive the learning algorithm for three different gating models (softmax, sigmoid, and Gaussian) and apply the localized multiple kernel learning framework to four different machine learning problems (two-class classification, regression, multiclass classification, and one-class classification).

We perform experiments for several two-class classification and regression problems. We compare the empirical performance of LMKL with single kernel SVM and SVR as well as MKL. For classification problems defined on different feature representations, LMKL is able to construct better classifiers than MKL by combining the kernels on these representations locally. In our experiments, LMKL achieves higher average test accuracies and stores fewer support vectors compared to MKL. If the combined feature representations are complementary and do not contain redundant information, the sigmoid gating model should be selected instead of softmax gating, in order to have the possibility of using more than one representation. We also see that, as expected, combining heterogeneous feature representations is more advantageous than combining multiple copies of the same representation. For image recognition problems, LMKL identifies the relevant parts of each input image separately using the gating model as a saliency detector on the kernels on the image patches, and we see that LMKL obtains better classification results than

MKL for a gender recognition task using face images. For regression problems, LMKL improves the performance by reducing the mean square error significantly on one of the data sets and storing significantly fewer support vectors on the other data set. Different from MKL methods that use global kernel weights, LMKL can combine multiple copies of the same kernel. We show that even if we provide more kernels than needed, LMKL uses only as many support vectors as required and does not overfit.

Acknowledgments

This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, the Boğaziçi University Scientific Research Project 07HA101, and the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK.

References

[1] E. Alpaydın, Selective attention for handwritten digit recognition, in: Advances in Neural Information Processing Systems 8, 1996.
[2] E. Alpaydın, Combined 5×2 cv F test for comparing supervised classification learning algorithms, Neural Computation 11 (8) (1999) 1885–1892.
[3] E. Alpaydın, M.I. Jordan, Local linear perceptrons for classification, IEEE Transactions on Neural Networks 7 (3) (1996) 788–792.
[4] F.R. Bach, G.R.G. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in: Proceedings of the 21st International Conference on Machine Learning, 2004.
[5] K.P. Bennett, M. Momma, M.J. Embrechts, MARK: a boosting algorithm for heterogeneous kernel models, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.
[6] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1–3) (2002) 131–159.
[7] M. Christoudias, R. Urtasun, T. Darrell, Bayesian Localized Multiple Kernel Learning, Technical Report. UCB/EECS-2009-96, University of California at Berkeley, 2009.
[8] C. Cortes, M. Mohri, A. Rostamizadeh, Learning non-linear combinations of kernels, in: Advances in Neural Information Processing Systems 22, 2010.
[9] K. Crammer, J. Keshet, Y. Singer, Kernel design using boosting, in: Advances in Neural Information Processing Systems 15, 2003.
[10] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, 2000.
[11] M. Girolami, S. Rogers, Hierarchic Bayesian models for kernel learning, in: Proceedings of the 22nd International Conference on Machine Learning, 2005.
[12] M. Girolami, M. Zhong, Data integration for classification problems employing Gaussian process priors, in: Advances in Neural Processing Systems 19, 2007.
[13] M. Gönen, E. Alpaydın, Localized multiple kernel learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008.
[14] M. Gönen, E. Alpaydın, Localized multiple kernel learning for image recognition, in: NIPS Workshop on Understanding Multiple Kernel Learning Methods, 2009.
[15] M. Gönen, E. Alpaydın, Multiple kernel machines using localized kernels, in: Supplementary Proceedings of the Fourth IAPR International Conference on Pattern Recognition in Bioinformatics, 2009.
[16] M. Gönen, E. Alpaydın, Localized multiple kernel regression, in: Proceedings of the 20th International Conference on Pattern Recognition, 2010.
[17] M. Gönen, E. Alpaydın, Supervised learning of local projection kernels, Neurocomputing 73 (10–12) (2010) 1694–1703.
[18] M. Gönen, E. Alpaydın, Multiple kernel learning algorithms, Journal of Machine Learning Research 12 (2011) 2211–2268.
[19] M. Hu, Y. Chen, J.T.-Y. Kwok, Building sparse multiple-kernel SVM classifiers, IEEE Transactions on Neural Networks 20 (5) (2009) 827–839.
[20] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79–87.
[21] J. Kandola, J. Shawe-Taylor, N. Cristianini, Optimizing kernel alignment over combinations of kernels, in: Proceedings of the 19th International Conference on Machine Learning, 2002.
[22] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.
[23] D.P. Lewis, T. Jebara, W.S. Noble, Nonstationary kernel combination, in: Proceedings of the 23rd International Conference on Machine Learning, 2006.
[24] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, Text classification using string kernels, Journal of Machine Learning Research 2 (2002) 419–444.
[25] J.M. Moguerza, A. Muñoz, I.M. de Diego, Improving support vector classification via the combination of multiple sources of information, in: Proceedings of Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, 2004, pp. 592–600.
[26] Mosek, The MOSEK Optimization Tools Manual Version 6.0 (Revision 119). MOSEK ApS, Denmark, 2011.
[27] P. Pavlidis, J. Weston, J. Cai, W.N. Grundy, Gene functional classification from heterogeneous data, in: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology, 2001, pp. 242–248.
[28] I. Psorakis, T. Damoulas, M.A. Girolami, Multiclass relevance vector machines: sparsity and accuracy, IEEE Transactions on Neural Networks 21 (10) (2010) 1588–1598.
[29] A. Rakotomamonjy, F.R. Bach, S. Canu, Y. Grandvalet, SimpleMKL, Journal of Machine Learning Research 9 (2008) 2491–2521.
[30] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, Cambridge, MA, 2002.
[31] B. Schölkopf, K. Tsuda, J.P. Vert (Eds.), Kernel Methods in Computational Biology, The MIT Press, Cambridge, MA, 2004.
[32] B.W. Silverman, Some aspects of the spline smoothing approach to non-parametric regression curve fitting, Journal of the Royal Statistical Society: Series B 47 (1985) 1–52.
[33] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
[34] J. Yang, Y. Li, Y. Tian, L. Duan, W. Gao, Group-sensitive multiple kernel learning for object categorization, in: Proceedings of the 12th IEEE International Conference on Computer Vision, 2009.
[35] J. Yang, Y. Li, Y. Tian, L. Duan, W. Gao, Per-sample multiple kernel approach for visual concept learning, EURASIP Journal on Image and Video Processing (2010), http://dx.doi.org/10.1155/2010/461450.
[36] Y. Ying, K. Huang, C. Campbell, Enhanced protein fold recognition through a novel data integration approach, BMC Bioinformatics 10 (1) (2009) 267.

Mehmet Gönen received the B.Sc. degree in industrial engineering, the M.Sc. and the Ph.D. degrees in computer engineering from Boğaziçi University, İstanbul, Turkey, in 2003, 2005, and 2010, respectively.

He was a Teaching Assistant at the Department of Computer Engineering, Boğaziçi University. He is currently doing his postdoctoral work at the Department of Information and Computer Science, Aalto University School of Science, Espoo, Finland. His research interests include support vector machines, kernel methods, Bayesian methods, optimization for machine learning, dimensionality reduction, information retrieval, and computational biology applications.

Ethem Alpaydın received his B.Sc. from the Department of Computer Engineering of Boğaziçi University in 1987 and the degree of Docteur es Sciences from Ecole Polytechnique Fédérale de Lausanne in 1990.

He did his postdoctoral work at the International Computer Science Institute, Berkeley, in 1991 and afterwards was appointed as Assistant Professor at the Department of Computer Engineering of Boğaziçi University. He was promoted to Associate Professor in 1996 and Professor in 2002 in the same department. As visiting researcher, he worked at the Department of Brain and Cognitive Sciences of MIT in 1994, the International Computer Science Institute, Berkeley, in 1997 and IDIAP, Switzerland, in 1998. He was awarded a Fulbright Senior scholarship in 1997 and received the Research Excellence Award from the Boğaziçi University Foundation in 1998 (junior level) and in 2008 (senior level), the Young Scientist Award from the Turkish Academy of Sciences in 2001 and the Scientific Encouragement Award from the Scientific and Technological Research Council of Turkey in 2002. His book Introduction to Machine Learning was published by The MIT Press in October 2004. Its German edition was published in 2008, its Chinese edition in 2009, its second edition in 2010, and its Turkish edition in 2011. He is a senior member of the IEEE, an editorial board member of The Computer Journal (Oxford University Press) and an associate editor of Pattern Recognition (Elsevier).