Multiple Kernel Learning for Sparse Representation-based Classification

Ashish Shrivastava, *Student Member, IEEE,* Vishal M. Patel, *Member, IEEE,* and Rama Chellappa, *Fellow, IEEE*

***Abstract*—In this paper, we propose a Multiple Kernel Learning (MKL) algorithm that is based on the Sparse Representation-based Classification (SRC) method. Taking advantage of the non-linear kernel SRC in efficiently representing the non-linearities in the high-dimensional feature space, we propose an MKL method based on the kernel alignment criteria. Our method uses a two step training method to learn the kernel weights and the sparse codes. At each iteration, the sparse codes are updated first while fixing the kernel mixing coefficients, and then the kernel mixing coefficients are updated while fixing the sparse codes. These two steps are repeated until a stopping criteria is met. The effectiveness of the proposed method is demonstrated using several publicly available image classification databases and it is shown that this method is efficient and can perform significantly better than many competitive image classification algorithms.**

***Index Terms*—Sparse representation-based classification, multiple kernel learning, kernel sparse representation, object recognition.**

I. Introduction

In recent years, the field of sparse representation has undergone rapid development, both in theory and in algorithms [1], [2], [3], [4]. This is partly due to the fact that signals or images of interest, though high dimensional, can often be coded using a few representative atoms in some dictionary. Let $\mathbf{Y}$ be a redundant dictionary with N atoms in $\mathbb{R}^d$

$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{d\times N}.$$

The atoms have unit Euclidean norm i.e., $\|\mathbf{y}_i\| = 1 \;\; \forall i$. Given a signal $\mathbf{y}_t \in \mathbb{R}^d$, finding the sparsest representation of $\mathbf{y}_t$ in $\mathbf{Y}$ entails solving the following optimization problem

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{y}_t = \mathbf{Y}\mathbf{x}, \tag{1}$$

where the $\|\mathbf{x}\|_0 := \#\{j : x_j \neq 0\}$, which is a count for the number of nonzero elements in $\mathbf{x}$. Problem (1) is NP-hard and cannot be solved in a polynomial time. Hence, approximate solutions are usually sought [3], [5], [6], [7]. For instance, Basis Pursuit [5] offers the solution via ℓ_1-minimization as

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \text{ subject to } \mathbf{y}_t = \mathbf{Y}\mathbf{x}, \tag{2}$$

where $\|\cdot\|_p$ for $0 < p < \infty$ is the ℓ_p-norm defined as

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^{d} |x_j|^p\right)^{\frac{1}{p}}.$$

The sparsest recovery is possible provided that certain conditions are met [8], [4]. One can adapt the above framework to noisy setting, where the measurements are contaminated with an error $\mathbf{n}$ obeying $\|\mathbf{n}\|_2 < \epsilon$, that is

$$\mathbf{y}_t = \mathbf{Y}\mathbf{x} + \mathbf{n} \quad \text{for} \quad \|\mathbf{n}\|_2 < \epsilon. \tag{3}$$

A stable solution can be obtained by solving the following optimization problem [4]

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \text{ subject to } \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 < \epsilon. \tag{4}$$

It has been shown that sparse representation works well in many inverse problems where the original signal $\mathbf{y}_t$ needs to be reconstructed as accurately as possible, such as denoising, deconvolution and image inpainting. Sparse representation framework has also been used for signal classification tasks [9], [10], [2], [11], [12]. In particular, Sparse Representation-based Classification (SRC) algorithm [10] has gained a lot of attraction in recent years. This is mainly due to the fact that it is robust to noise and occlusion [10], [13].

The SRC method is based on finding a linear representation of the data. However, linear representations are almost always inadequate for representing non-linear structures of the data which arise in many practical applications. To deal with this problem, non-linear SRC methods have been proposed in the literature [14], [15]. These algorithms essentially map the non-linear data into high-dimensional feature space using the kernel trick so that data of the same distribution are easily grouped together and are linearly separable. This may also allow one to easily find the sparse representation of data and significantly reduce the reconstruction error [16], [17]. Kernel SRC methods have shown to produce better classification results than the traditional SRC.

Kernel SRC methods [14], [15] require the use of a pre-determined kernel function such as the polynomial kernel or the Gaussian kernel. Selection of the kernel function and its parameters is an important issue in training when kernel SRC methods are used for classification. In general, cross validation is used to choose the best kernel function among a set of kernel functions. Recently, Multiple Kernel Learning (MKL) methods that allow one to use multiple kernels instead of using a specific kernel function have been proposed in the literature [18].

Ashish Shrivastava is with the Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742 (e-mail:ashish@umiacs.umd.edu).

Vishal M. Patel is with the Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742 (e-mail: pvishalm@umd.edu).

Rama Chellappa is with the Department of Electrical and Computer Engineering and the Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742 (e-mail: rama@umiacs.umd.edu).

EDICS: SMR-REP, ARS-SRE

In this paper, we propose a kernel sparse representation-based classification method based on MKL where multiple kernel functions are combined to obtained a better solution. Our method uses a two step training method using the SRC as the base learner. At each iteration, first the combination function parameters are updated while fixing the base learner parameters, and then the base learner parameters are updated while fixing the combination function parameters. These two steps are repeated until convergence. Fig. 1 presents an overview of our method.

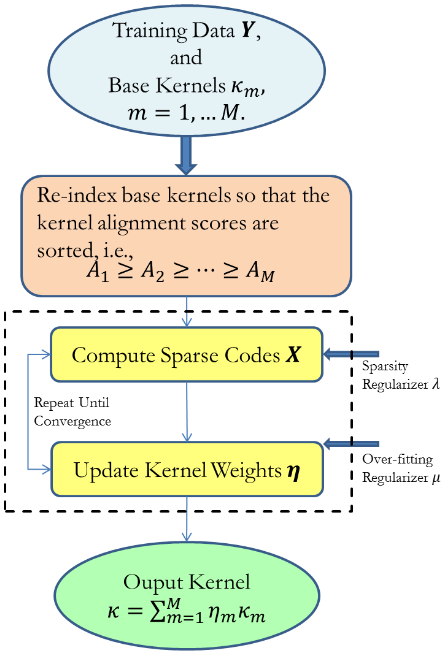

Training Data $\boldsymbol{Y}$,
and
Base Kernels κ_m,
$m = 1, \ldots M$.
Re-index base kernels so that the kernel alignment scores are sorted, i.e.,
$A_1 \geq A_2 \geq \cdots \geq A_M$
Compute Sparse Codes $\boldsymbol{X}$
Sparsity Regularizer λ
Repeat Until Convergence
Update Kernel Weights $\boldsymbol{\eta}$
Over-fitting Regularizer μ
Ouput Kernel
$\kappa = \sum_{m=1}^{M} \eta_m \kappa_m$

Fig. 1: Overview of the proposed method.

A. *Organization of the Paper*

This paper is organized as follows. In Section II, we review some related work on SRC, kernel SRC and MKL. Details of our MKL-based SRC method are given in Section III. Experimental results are presented in Section IV and Section V concludes the paper with a brief summary and discussion.

II. Background

In this section, we review some related work on SRC, kernel SRC and MKL.

A. *Sparse Representation-based Classification*

Suppose that we are given C distinct classes and a set of N_c training images per class. We identify an $l \times p$ grayscale image as a d-dimensional vector which can be obtained by stacking its columns. Let $\mathbf{Y}_c = [\mathbf{y}_1^c, \ldots, \mathbf{y}_{N_c}^c] \in \mathbb{R}^{d \times N_c}$ be the matrix of training images from the cth class. Define a new matrix, $\mathbf{Y}$, as the concatenation of training samples from all the classes as

$$\begin{aligned}\mathbf{Y} &= [\mathbf{Y}_1, \ldots, \mathbf{Y}_C] \in \mathbb{R}^{d \times N} \\ &= [\mathbf{y}_1^1, \ldots, \mathbf{y}_{N_1}^1 | \mathbf{y}_1^2, \ldots, \mathbf{y}_{N_2}^2 | \ldots\ldots | \mathbf{y}_1^C, \ldots, \mathbf{y}_{N_C}^C] \\ &\triangleq [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N],\end{aligned}$$

where $N = \sum_c N_c$. We consider an observation vector $\mathbf{y}_t \in \mathbb{R}^d$ of unknown class as a linear combination of the training vectors as

$$\mathbf{y}_t = \sum_{c=1}^{C} \sum_{i=1}^{N_c} x_i^c \mathbf{y}_i^c \tag{5}$$

with coefficients $x_i^c \in \mathbb{R}$. The above equation can be more compactly written as

$$\mathbf{y}_t = \mathbf{Y}\mathbf{x}, \tag{6}$$

where

$$\begin{aligned}\mathbf{x} &= [x_1^1, \ldots, x_{N_1}^1 | x_1^2, \ldots, x_{N_2}^2 | \ldots | x_1^C, \ldots, x_{N_C}^C]^T \\ &\triangleq [x_1, x_2, \ldots, x_N]^T\end{aligned} \tag{7}$$

and $.^T$ denotes the transposition operation. One can make an assumption that given sufficient training samples of the cth class, $\mathbf{Y}_c$, any new test image $\mathbf{y}_t \in \mathbb{R}^d$ that belongs to the same class will approximately lie in the linear span of the training samples from the class c. This implies that most of the coefficients not associated with class c in (7) will be close to zero. As a result, assuming that observations are noisy, one can recover this sparse vector by solving the following optimization problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad \text{subject to } \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 \leq \epsilon \tag{8}$$

or equivalently the following formulation,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 + \lambda\|\mathbf{x}\|_1, \tag{9}$$

where λ is a parameter. The sparse code $\mathbf{x}_t$ can then be used to determine the class of $\mathbf{y}_t$ by computing the following error for each class,

$$e_c = \|\mathbf{y}_t - \mathbf{Y}_c\mathbf{x}_t^c\|_2, \tag{10}$$

where, $\mathbf{x}_t^c$ is the part of coefficient vector $\mathbf{x}_t$ that corresponds to $\mathbf{Y}_c$. Finally, the class c^* that is associated to the test sample $\mathbf{y}_t$, can be declared as the one that produces the smallest approximation error

$$c^* = \text{class of } \mathbf{y}_t = \arg\min_c e_c. \tag{11}$$

The SRC method was originally proposed for face biometric in [10]. It was then extended for cancelable iris biometric in [13] and for automatic target recognition in [19].

B. *Kernel SRC*

Many types of descriptors in computer vision such as the spatial pyramid descriptor and the region covariance descriptor have intrinsic nonlinear similarity measure functions. This has motivated researchers to develop non-linear kernel sparse representations for object representation and classification [14], [15], [16], [20], [21], [22], [23], [24].

In kernel SRC, essentially the idea is to map data in the high dimensional feature space and solve (9) using the kernel trick [25]. Let $\mathbf{\Phi} : \mathbb{R}^d \rightarrow G$ be a non-linear mapping from d-dimensional space into a dot product space G. A non-linear SRC can be performed by solving the following optimization problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1, \qquad (12)$$

where

$$\mathbf{\Phi}(\mathbf{Y}) \triangleq [\mathbf{\Phi}(\mathbf{y}_1^1), \cdots, \mathbf{\Phi}(\mathbf{y}_{N_1}^1)| \cdots |\mathbf{\Phi}(\mathbf{y}_1^C), \cdots, \mathbf{\Phi}(\mathbf{y}_{N_C}^C)].$$

Denote the first term of (12) by $\mathcal{E}_\kappa$ as follows

$$\begin{aligned}\mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) &= \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 \\ &= \mathbf{\Phi}(\mathbf{y}_t)^T\mathbf{\Phi}(\mathbf{y}_t) + \mathbf{x}^T\mathbf{\Phi}(\mathbf{Y})^T\mathbf{\Phi}(\mathbf{Y})\mathbf{x} \\ &\quad - 2\mathbf{\Phi}(\mathbf{y}_t)^T\mathbf{\Phi}(\mathbf{Y})\mathbf{x} \\ &= \boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{y}_t) + \mathbf{x}^T\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\mathbf{x} \\ &\quad - 2\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y})\mathbf{x}, \qquad (13)\end{aligned}$$

where $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y}) \in \mathbb{R}^{N\times N}$ is a positive semidefinite kernel Gram matrix whose elements are computed as

$$\begin{aligned}[\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})]_{i,j} &= [\langle\mathbf{\Phi}(\mathbf{Y}), \mathbf{\Phi}(\mathbf{Y})\rangle]_{i,j} \\ &= \mathbf{\Phi}(\mathbf{y}_i)^T\mathbf{\Phi}(\mathbf{y}_j) = \kappa(\mathbf{y}_i, \mathbf{y}_j),\end{aligned}$$

$\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{y}_t) = \kappa(\mathbf{y}_t, \mathbf{y}_t)$, and

$$\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y}) \triangleq [\kappa(\mathbf{y}_t, \mathbf{y}_1), \kappa(\mathbf{y}_t, \mathbf{y}_2), \cdots, \kappa(\mathbf{y}_t, \mathbf{y}_N)] \in \mathbb{R}^{1\times N}.$$

Here, $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the kernel function.

It is apparent that the objective function is feasible since it only involves a matrix of finite dimension. Furthermore, the computation of $\boldsymbol{\mathcal{K}}$ only requires dot products. Therefore, we are able to employ Mercer kernel functions to compute these dot products without carrying out the mapping $\mathbf{\Phi}$. Some commonly used kernels include polynomial kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle(\mathbf{x}, \mathbf{y}\rangle + a)^b$$

and Gaussian kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right),$$

where a, b and c are the parameters. Note that κ in the subscript of $\mathcal{E}_\kappa$ stresses on the fact that the error term depends on the choice of the kernel function. With the above definitions, the kernel version of the SRC optimization problem in (9) can be written as,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) + \lambda\|\mathbf{x}\|_1. \qquad (14)$$

One can solve the optimization problem (14) by modifying the LARS algorithm [26]. In the case, when ℓ_1-norm is replaced by the ℓ_0-norm in (14), kernel orthogonal matching pursuit algorithm can be used to solve the resulting optimization problem [20], [16], [27].

C. Multiple Kernel Learning

In order to achieve good performance in object classification tasks, it is important to combine inputs from various image features. The large margin classifiers as well as many other classifiers in computer vision are constructed based on similarity measures between samples (or kernels). Finding appropriate feature combinations entails designing good kernel functions among a set of candidate kernels. One way to achieve this is by finding positive mixtures of predetermined base kernels. MKL is a theoretically and technically very attractive way of determining the mixing weights of multiple kernels [18], [28], [29], [30], [31], [32]. This method have also been used to combine multiple features as explored by [33] and object detection [34]. MKL learns the kernel weights and the classifier simultaneously.

Let $\kappa_1, \cdots, \kappa_M$ be a set of base kernel functions that would be used to compute kernel matrices. In the MKL framework, linear combinations of the base kernels are considered

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \eta_m\kappa_m(\mathbf{y}_i, \mathbf{y}_j),$$

and the mixing coefficients η_m are learned together with the model parameters, so as to maximize the classification ability [18]. Various MKL algorithms have been proposed in the literature that essentially differ in the training method, the base learner, the functional form or the learning method [18].

For example, one can obtain a valid kernel by taking the summation or multiplication of M valid kernels

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \kappa_m(\mathbf{y}_i, \mathbf{y}_j)$$

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \prod_{m=1}^{M} \kappa_m(\mathbf{y}_i, \mathbf{y}_j).$$

One can also select the kernel weights based on the performance of each kernel. The following rule is proposed in [35] for the selection of kernel weights

$$\eta_m = \frac{\xi_m - \delta}{\sum_{l=1}^{M}(\xi_l - \delta)},$$

where δ is the threshold that should be less than or equal to the minimum of the accuracies obtained from single-kernel learners and ξ_m is the accuracy obtained using only $\boldsymbol{\mathcal{K}}_m$.

The notion of of kernel alignment which is a measure of similarity between two kernel functions or between a kernel and a target function was introduced in [36]. Let $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ be the Gram matrices of kernel functions κ_1 and κ_2 for a set $\{\mathbf{y}_i\}_{i=1}^N$ of the inputs. Then, the alignment score A between the kernel matrices $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ is defined,

$$A(\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2) = \frac{\langle\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2\rangle_F}{\sqrt{\langle\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_1\rangle_F\langle\boldsymbol{\mathcal{K}}_2, \boldsymbol{\mathcal{K}}_2\rangle_F}}, \qquad (15)$$

where,

$$\langle\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2\rangle_F = \sum_{i=1}^{N}\sum_{j=1}^{N} \kappa_1(\mathbf{y}_i, \mathbf{y}_j)\kappa_2(\mathbf{y}_i, \mathbf{y}_j).$$

One can view kernel alignment as the cosine of the angle between $\mathcal{K}_1$ and $\mathcal{K}_2$. Kernel alignment can be used to select the kernel weights. In [37], the following approach is used for selecting the kernel weights

$$\eta_m = \frac{A(\mathcal{K}_m, \boldsymbol{\mathcal{K}d})}{\sum_{l=1}^{M} A(\mathcal{K}_l, \boldsymbol{\mathcal{K}d})} \quad \forall m,$$

where $\boldsymbol{\mathcal{K}d} \in \mathbb{R}^{N \times N}$ is the ideal kernel matrix whose elements are defined as follows

$$[\boldsymbol{\mathcal{K}d}]_{(i,j)} = \begin{cases} 1, & \text{if } \mathbf{y}_i \in \text{class } c \text{ and } \mathbf{y}_j \in \text{class } c \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

In other words, $\boldsymbol{\mathcal{K}d}$ is a block diagonal matrix which has 1's where rows and columns correspond to the same class and 0's everywhere else. Suppose that we are given 3 classes and 2 samples per class, e.g., $\mathbf{Y}_{ex} = [\mathbf{y}_1^1, \mathbf{y}_2^1, \mathbf{y}_1^2, \mathbf{y}_2^2, \mathbf{y}_1^3, \mathbf{y}_2^3]$, then the resulting ideal matrix is the following block diagonal matrix

$$\boldsymbol{\mathcal{K}d}(\mathbf{Y}_{ex}, \mathbf{Y}_{ex}) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{17}$$

A method for updating Gram matrices based on optimizing the alignment $A(\mathcal{K}, \boldsymbol{\mathcal{K}d})$ was proposed in [36], where the definition of the ideal kernel was slightly different. Many other methods have been proposed in the literature that use kernel alignment or a variation of kernel alignment for learning the kernel weights. See [18] for an excellent survey of different MKL algorithms.

III. Multiple Kernel Learning for SRC

In this section, we first present our formulation for Multiple Kernel Learning for SRC (MKL-SRC). We then present the details of the optimization algorithm.

A. Problem Formulation

If we use the training matrix $\mathbf{Y}$ to predict the class of a training sample, then the sparse code will always be all zeros but a single 1 at the location corresponding to the training sample under consideration. This sparse code will always correctly classify all the training samples and, hence, will not help in computing the optimal kernel. In order to avoid this degenerate case, we set the corresponding column of $\mathbf{Y}$ to $\mathbf{0}$ before computing the sparse code. This can be done as follows

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \mathcal{E}_\kappa(\mathbf{x}_i; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) + \lambda \|\mathbf{x}_i\|_1, \tag{18}$$

where,

$$\tilde{\mathbf{Y}}_i = [\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathbf{0}, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_N], \tag{19}$$

and $\mathbf{0}$ is a d-dimensional vector with zeros as its entries. We stack up all the sparse vectors $\mathbf{x}_i$ in columns of a matrix $\mathbf{X}$, i.e., $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times N}$. Now, in order to learn the optimal kernel κ, we write it as linear combination of M base kernels as follows

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \eta_m \kappa_m(\mathbf{y}_i, \mathbf{y}_j), \tag{20}$$

where η_m is the weight of the mth base kernel and $\sum_{m=1}^{M} \eta_m = 1$. Using (20), $\mathcal{E}_\kappa$ can be written as,

$$\begin{aligned} \mathcal{E}(\mathbf{x}, \boldsymbol{\eta}; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) &= \sum_m \eta_m \mathcal{K}_m(\mathbf{y}_i, \mathbf{y}_i) \\ &+ \mathbf{x}^T \Big(\sum_m \eta_m \mathcal{K}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}) \Big) \mathbf{x} - 2 \sum_m \eta_m \mathcal{K}_m(\mathbf{y}_i, \tilde{\mathbf{Y}}) \mathbf{x} \\ &= \sum_m \eta_m \Big(\mathcal{K}_m(\mathbf{y}_i, \mathbf{y}_i) + \mathbf{x}^T \mathcal{K}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}) \mathbf{x} \\ &- 2 \mathcal{K}_m(\mathbf{y}_i, \tilde{\mathbf{Y}}) \mathbf{x} \Big), \end{aligned}$$

where $\mathcal{K}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}})$ can be computed by setting ith row and ith column of $\mathcal{K}(\mathbf{Y}, \mathbf{Y})$ to zeros, and $\mathcal{K}_m(\mathbf{y}_i, \tilde{\mathbf{Y}})$ by setting ith column of $\mathcal{K}(\mathbf{y}_i, \mathbf{Y})$ to zero. Let the kernel mixing coefficients vector be denoted by $\boldsymbol{\eta}$, i.e., $\boldsymbol{\eta} := \left[\eta_1, \cdots, \eta_M\right]^T$. Note that we have dropped the subscript κ and added the variable $\boldsymbol{\eta}$ to stress the dependency of the cost on the coefficients $\boldsymbol{\eta}$. In order to jointly learn the optimal sparse codes $\hat{\mathbf{X}}$ and the kernel function coefficients $\hat{\boldsymbol{\eta}}$, the following MKL optimization problem needs to be solved,

$$\begin{aligned} \hat{\mathbf{X}}, \hat{\boldsymbol{\eta}} &= \arg\min_{\mathbf{X}, \boldsymbol{\eta}} \sum_{i=1}^{N} \mathcal{E}(\mathbf{x}, \boldsymbol{\eta}; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) + \lambda \|\mathbf{x}_i\|_1 \\ \text{subject to} & \quad \sum_{m=1}^{M} \eta_m = 1, \text{and } \boldsymbol{\eta} \geq 0. \end{aligned} \tag{21}$$

To optimize (21), one can alternate between solving for $\mathbf{X}$ with fixed $\boldsymbol{\eta}$ and, then, solving for $\boldsymbol{\eta}$ while keeping $\mathbf{X}$ fixed. With fixed $\boldsymbol{\eta}$, the optimization problem reduces to the standard kernel SRC which can be solved by using LARS [26] type of algorithm. However, while solving for $\boldsymbol{\eta}$ (with fixed $\mathbf{X}$), the problem reduces to a linear programming (LP) problem and has the following two shortcomings:

1) The solution of the optimization problem finds the kernel that reduces the reconstruction error of each sample but does not necessarily classify them in correct classes.
2) The LP finds a solution at the vertex, which, in our problem, lies on the axes. As a result, the optimization chooses just one kernel at each iteration and this choice of kernel keeps changing over iterations. This makes the algorithm very unstable.

In order to avoid these issues, we propose a kernel alignment-based algorithm for kernel learning that focuses on classification error of the training samples. To this end, our goal at the kernel learning stage is to learn the optimal kernel function κ that results in the maximum training classification accuracy while avoiding over-fitting. We first explain how this is done to avoid over-fitting. Then, we describe our algorithm for computing the weights $\boldsymbol{\eta}$.

B. Ordered Kernel Alignment Scores

We rank each base kernel based on how close the corresponding kernel matrix of the training data is to the ideal kernel matrix $\boldsymbol{\mathcal{K}d} \in \mathbb{R}^{N\times N}$ that we defined in (16). To avoid over-fitting, we give preference to a kernel matrix $\boldsymbol{\mathcal{K}}$ that is "closer" to the ideal matrix.

The notion of closeness between two kernel matrices is defined in terms of kernel alignment criterion in (15). Kernel alignment score between a base kernel matrix $\boldsymbol{\mathcal{K}}_m$ and the ideal kernel matrix can be computed as

$$A_m(\boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}d}) = \frac{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}d}\rangle_F}{N\sqrt{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}}_m\rangle_F}}. \tag{22}$$

A kernel function κ_m whose corresponding kernel matrix $\boldsymbol{\mathcal{K}}_m$ gives higher alignment score with the ideal kernel matrix, is ranked higher. Without loss of generality (w.l.o.g.) we assume that the alignment scores of the base kernel functions $\kappa_1, \ldots, \kappa_M$ are sorted as follows,

$$A_1 \geq A_2 \geq \cdots \geq A_M. \tag{23}$$

The assumption is true w.l.o.g. because if the alignment scores are not sorted, we can re-index the base kernels so that they become sorted. Next, we explain how we compute the weights $\eta_m, m = 1, \ldots, M$, for all the base kernels based on the classification accuracy. Furthermore, we show how the ordering of kernels based on alignment scores helps to avoid over-fitting.

C. Computing Kernel Function Weights $\boldsymbol{\eta}$

Our MKL-SRC method alternates between learning sparse coefficients $\mathbf{X}$ and kernel function weights $\boldsymbol{\eta}$. Given sparse codes, we predict the labels of all the training samples. Let h_i be the predicted label of $\mathbf{y}_i$ using the current kernel function. To determine the prediction accuracy on the training samples, we define boolean variables $z_i \in \{0,1\}, i = 1, \ldots, N$, that is set to 1 if the predicted labels of $\mathbf{y}_i$ is correct and 0 otherwise. That is,

$$z_i = \begin{cases} 1, & \text{if } h_i = l_i \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

We update the kernel weights $\boldsymbol{\eta}$ by adding a kernel that can help to classify the samples correctly where $z_i = 0$. We pre-compute the predicted labels of all the training samples for each base kernel. Let g_i^m be the predicted label of the ith sample with the mth base kernel and let,

$$z_i^m = \begin{cases} 1, & \text{if } g_i^m = l_i \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

We choose the base kernel m if its prediction error is the smallest among all the base kernel functions. At the same time, to avoid over-fitting, we want this chosen kernel to have high alignment score. This ensures that we do not choose a kernel just based on its training classification accuracy. In other words, taking the alignment score of the kernel functions into consideration ensures generalization capability of the classifier and, thus, avoids over-fitting. For the miss-classified samples, let the accuracy for the mth base kernels be defined as

$$c_m = \frac{\sum_{\{i:z_i=0\}} z_i^m}{\sum_{i=1}^N (1 - z_i)}. \tag{26}$$

We choose a kernel κ_{m^*}, if it gives the best accuracy among all the kernels that have lower alignment score than κ_{m^*}, and its accuracy c_{m^*} is at least better by μ than the accuracy of any kernel function that has higher alignment scores than that of κ_{m^*}. Formally,

$$\text{choose } k_{m^*} \text{ such that} \begin{cases} c_{m^*} \geq c_m, & \text{if } m \leq m^* \\ c_{m^*} \geq c_m + \mu, & \text{if } m > m^*. \end{cases} \tag{27}$$

The parameter μ controls the over-fitting by favoring a kernel function that has higher kernel alignment score. Note that the above choice of m^* in (27) gives the preference to the kernels with higher alignment score because they are assumed to be sorted in decreasing order. After choosing the kernel κ_{m^*}, we adjust the weights of the kernel functions in proportion to their respective accuracies. In order to compute the weights of the kernel functions, we consider only those samples whose labels are incorrectly predicted by either the current kernel or the chosen kernel. By the current kernel we mean the linear combination of all the kernel functions in the previous iteration. Hence, the weight of the new kernel is given by,

$$w_{newKernel} = \frac{\sum_{i=1}^N (z_i^{m^*}) \wedge (1 - z_i)}{\sum_{i=1}^N (1 - z_i) \vee (1 - z_i^{m^*})}, \tag{28}$$

where $\wedge$ is a logical 'AND' operator and $\vee$ is a logical 'OR' operator. The numerator in (28) counts the number of samples where the new kernel predicts correct label while the current kernel does not. Similarly, the denominator counts the number of samples where either the current kernel or the new kernel does not predict the correct label. Likewise, the weight for the current kernel is computed as,

$$w_{currKernel} = \frac{\sum_{i=1}^N (1 - z_i^{m^*}) \wedge (z_i)}{\sum_{i=1}^N (1 - z_i) \vee (1 - z_i^{m^*})} \tag{29}$$

where, the numerator counts the number of samples whose labels are correctly predicted by the current kernel but not by the new kernel. The current kernel is the linear combination of the kernels in the previous iteration. Let $\boldsymbol{\eta}^t = [\eta_1^t, \ldots, \eta_M^t]$ be the kernel weights at the tth iteration. Then, the weights for the $(t+1)$th iteration are computed as follows

$$\eta_m^{t+1} = \begin{cases} w_{newKernel} & \text{if } m = m^* \\ \eta_m^t * w_{currKernel} & \text{otherwise.} \end{cases} \tag{30}$$

The kernel weights is initialized all the weight given to the kernel with highest alignment score, i.e. $\eta_1^0 = 1$ Finally, we divide each kernel weight η_m by the sum of all the weights

$$\eta_m \leftarrow \frac{\eta_m}{s}, \quad \text{where} \quad s = \sum_{m=1}^M \eta_m. \tag{31}$$

As an example, consider an illustrative example of 10 samples as shown in Fig. 2. The current kernel predicts

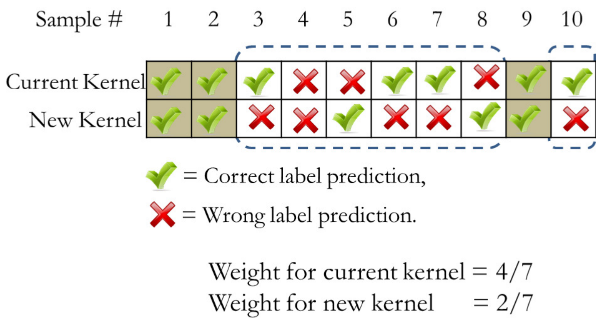

Sample #
1
2
3
4
5
6
7
8
9
10
Current Kernel ✓ ✓ ✓ ✗ ✗ ✓ ✓ ✗ ✓ ✓
New Kernel ✓ ✓ ✗ ✗ ✓ ✗ ✗ ✓ ✓ ✗
✓ = Correct label prediction,
✗ = Wrong label prediction.
Weight for current kernel = 4/7
Weight for new kernel = 2/7

Fig. 2: Updating kernel weights in each iteration.

correct labels of samples $\{1,2,3,6,7,9,10\}$, while the chosen kernel κ_{m^*} predicts correct class of the samples $\{1,2,5,8,9\}$. Since samples $\{1,2,9\}$ are predicted correctly by both the kernels, we consider samples $\{3,4,5,6,7,8,10\}$. Out of these 7 samples, current kernel predicts 4 correctly while, the new kernel predicts 2 correctly. Hence, $w_{currKernel} = 4/7$ and $w_{newKernel} = 2/7$.

Our approach for learning kernel weights is summarized in Algorithm 1.

```
Algorithm 1: Multiple Kernel Learning for SRC
Input: Data samples $\mathbf{Y}$, labels $\mathbf{l}$, kernel functions $\kappa_m$,
       parameters $\lambda, \mu$, maximum iteration count $T$, $\epsilon_0$.
Output: Kernel function weights $\boldsymbol{\eta}$.
For each kernel function $\kappa_m$ and sample $\mathbf{y}_i$ compute the
predicted label $g_i^m$, by computing the sparse code using
(18).
Initialize $\epsilon_1 \leftarrow \epsilon_0 + 1, t \leftarrow 0$, and compute kernel
matrices $\boldsymbol{\mathcal{K}}_m(\mathbf{Y},\mathbf{Y})$.
while $t \leq T$ and $\epsilon_1 \geq \epsilon_0$ do
    for $i = 1, \dots N$ do
        Compute $\boldsymbol{\mathcal{K}}_m(\tilde{\mathbf{Y}}_i, \tilde{\mathbf{Y}}_i)$ by setting the $i$th row and
        the $i$th column of $\boldsymbol{\mathcal{K}}_m(\mathbf{Y},\mathbf{Y})$ to 0.
        Compute the sparse code $\mathbf{x}_i$ using (18).
        Compute the predicted label $h_i$ using $\mathbf{x}_i$.
    end
    Update $\eta_m^t, \quad \forall m = 1, \dots, M$ using (30).
    Compute the sum of all weights $s = \sum_{m=1}^{M} \eta_m$.
    $\eta_m \leftarrow \eta_m / s, \quad \forall m = 1, \dots, M$.
    Set $\epsilon_1 \leftarrow \|\boldsymbol{\eta}^{t-1} - \boldsymbol{\eta}^t\|_2$.
    $t \leftarrow t + 1$
end
return $\boldsymbol{\eta}$
```

D. Classification

In order to predict the class of a test sample $\mathbf{y}_t$, we compute the sparse code by optimizing the following problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) \tag{32}$$

where, the learned linear weights $\boldsymbol{\eta}$ are used to compute the kernel function κ. Then, the error for each class is computed as

$$e_c^\kappa = \mathcal{E}_\kappa(\mathbf{x}_t^c; \mathbf{Y}, \mathbf{y}_t). \tag{33}$$

Finally, the class of the sample $\mathbf{y}_t$ is the one that results in the minimum error

$$c_t^* = \text{class of } \mathbf{y}_t = \arg\min_c e_c^\kappa. \tag{34}$$

IV. Experimental Results

In this section, we present several experimental results demonstrating the effectiveness of the proposed MKL-SRC method for classification tasks on both synthetic and real datasets. In particular, we present classification results on the Caltech101 object dataset [38], University of Washington RGB-D dataset [39] and gender recognition on the AR Face dataset [40]. We compare the results of our method with that of SVM, linear SRC [10], kernel SRC [14], [15], and a multiple kernel learning algorithm based on SVM (SVM-MKL) [41].

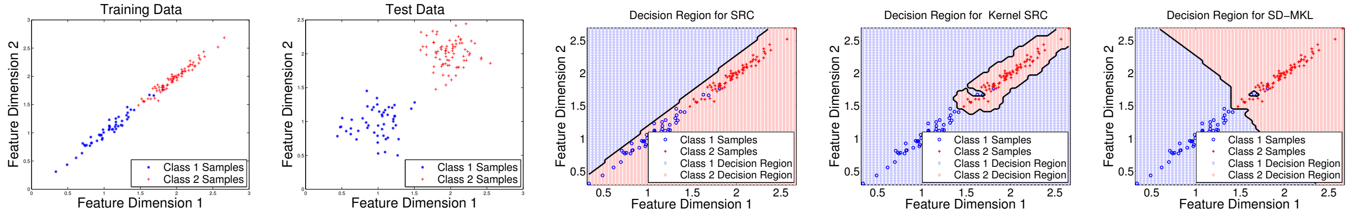
For all the experiments, we use a total of 50 base kernels $\kappa_m(\mathbf{y}_i, \mathbf{y}_j)$ which are described below

1) Two linear kernels, $\mathbf{y}_i^T\mathbf{y}_j$ and $(1 + \mathbf{y}_i^T\mathbf{y}_j)$.
2) Fifteen polynomial kernels, $(a + \mathbf{y}_i^T\mathbf{y}_j)^b$ of degree $b = 2$, 3, 4, and constant $a = 0.5, 1.0, 1.5, 2.0, 2.5$.
3) Ten tangent hyperbolic kernels, $\tanh(c1 + c2 * (\mathbf{y}_i^T\mathbf{y}_j))$ with $(c1, c2) = (0.1, 1)$, $(0.2, 1)$, $(0.3, 1)$, $(0.4, 1)$, $(0.5, 1)$, $(0.5, 0.2)$, $(0.5, 0.4)$, $(0.5, 0.6)$, $(0.5, 0.8)$, $(0.5, 1)$.
4) One histogram intersection kernel.
5) Remaining 22 Gaussian kernels, $\exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$, with c from 0.1 to 2.2 in the steps of 0.1.

As for the parameter selection, we have only two parameters: sparsity regularizer λ and over-fitting regularizer μ. In all of our experiments, we set $\lambda = 0.01$ and $\mu = 0.05$.

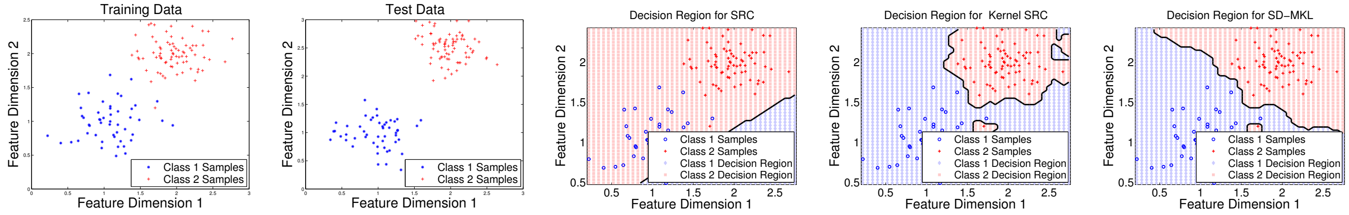
A. Analysis on Synthetic Data

We thoroughly evaluate the basic behavior of the proposed MKL-SRC method on two two-dimensional synthetic data sets. In both of the synthetic experiments, we generate two classes of two dimensional data from the Gaussian distributions, with different means but the same covariance matrices. In the first experiment, the mean of class 1 is set to $[1, 1]^T$ and that of class 2 to $[2, 2]^T$. The covariance matrix for both of the classes is $\begin{bmatrix} 0.51 & 0.049 \\ 0.049 & 0.51 \end{bmatrix}$. This generates the samples which are approximately co-linear as shown in Fig. 3(a). To generate the test data in this experiment, we change the covariance matrix to $\begin{bmatrix} 0.51 & 0.01 \\ 0.01 & 0.51 \end{bmatrix}$ which results in the data as shown in Fig. 3(b). It is known that, with co-linear data, SRC algorithm does not work well because both classes can be represented almost equally well by the training samples of either class [14]. This results in a biased decision boundary as shown in Fig. 3(c) and gives very poor accuracy of only 62.40%. In order to remove the co-linearity of the data, one can represent the samples in a high dimensional feature space and perform classification using the kernel SRC. However, choosing a kernel based on classification accuracy of the training data alone, can result in over fitting and non-optimal choice of kernel, as shown in Fig. 3(d). The kernel is non optimal because it does not take into the consideration of



(a) (b) (c) (d) (e)

Fig. 3: First synthetic dataset. (a) Training Data. (b) Test Data. Decision boundary with (c) SRC, (d) kernel SRC and (e) MKL-SRC.



(a) (b) (c) (d) (e)

Fig. 4: Second synthetic dataset. (a) Training Data, (b) Test Data. Decision boundary with (c) SRC, (d) kernel SRC, (e) MKL-SRC.

the fact that test data might be slightly different from the training data, and hence results in over-fitting. On the other hand, the proposed method uses the alignment score of the kernel matrices and computes the best composite kernel that gives good alignment score as well as good training accuracy. The result is a classifier that can generalize well on slightly different test data as shown in Fig. 3(e) and results in 100% accuracy.

In the second synthetic experiment, we generate class 1 data from a Gaussian distribution with mean $[1,1]^T$ and covariance matrix $\begin{bmatrix} 0.07 & 0.00 \\ 0.00 & 0.07 \end{bmatrix}$ and the second class using a Gaussian distribution with mean $[2,2]^T$ and the same covariance matrix. The test data for class 1 is generated from the same Gaussian distribution as the training data, however, we slightly change the mean of class 2 for test data to $[2,2.5]^T$. The training and the test data for this experiment are shown in Fig. 4(a) and (b), respectively. We show the decision boundary for SRC, kernel SRC and the proposed method in Fig. 4(c), (d) and (e), respectively. This experiment shows that learning a kernel that avoids over-fitting can result in better decision boundaries and hence better classification accuracy. Classification results on the synthetic data are summarized in Table I.

	SRC	Kernel SRC	MKL-SRC
Synthetic data 1	62.40	80.00	**100.0**
Synthetic data 2	63.20	69.60	**99.20**

TABLE I: Accuracy (%) on the synthetic data in Fig. 3 and 4.

B. *Object Recognition*

We perform the first set of object recognition experiments on the Caltech-101 database [38]. The Caltech101 dataset contains 102 categories including one background class. Each category has about 40 to 80 images and most of the categories have about 50 images. The images have been downloaded from the internet using Google search engine (www.google.com). The database contains a diverse and challenging set of images from buildings, musical instruments, animals and natural scenes, etc.

To show the appropriateness of sparsity in our application, we plot sparse coefficients when a test sample is represented as a sparse linear combination of training samples in the feature space. In particular, we randomly select five classes from the Caltech101 dataset to form a training matrix $\mathbf{Y}$ with fifteen samples from each class. Then, given a test sample $\mathbf{y}_t$ corresponding to one of the five classes from the Caltech101 dataset, we solve the following problem in the feature space using the polynomial kernel of degree two

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \qquad (35)$$

We repeat this procedure fifteen times with different test samples corresponding to each class and take the average of sparse codes. We plot these spare representations in Figure 5. From this figure, we see that most of the coefficients are clustered around the class corresponding to the training samples. Furthermore, we ran multiple experiments with different kernel functions and obtained similar results. This essentially shows that on average the data used in our application does have a sparse representation in the feature space.

Following the common experimental set up on this dataset [42], we train on j images, where $j \in \{5,10,15,20,25,30\}$, and test on the rest. For fair comparison, we use the same spatial pyramid features as used in [42]. Table II shows the comparison of our classification accuracy with the state-of-the-art. Note that our method performs significantly better than the linear SRC. Furthermore, it is interesting that our method

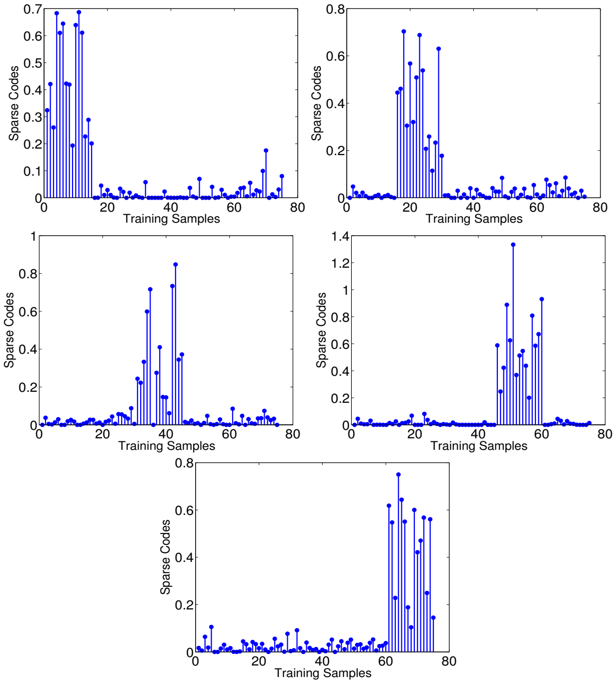


Fig. 5: Sparse coefficients corresponding to five classes from the Caltech101 dataset in the feature space.

Number of training samples	5	10	15	20	25	30
Malik [43]	46.6	55.8	59.1	62.0	–	66.20
Lazebnik [44]	–	–	56.4	–	–	64.6
Griffin [45]	44.2	54.5	59.0	63.3	65.8	67.60
Irani [46]	–	–	65.0	–	–	70.40
Grauman [47]	–	–	61.0	–	–	69.10
Venkatesh [48]	–	–	42.0	–	–	–
Gemert [49]	–	–	–	–	–	64.16
Yang [50]	–	–	67.0	–	–	73.20
Wang [51]	51.15	59.77	65.43	67.74	70.16	73.44
K-SVD [52]	49.8	59.8	65.2	68.7	71.0	73.2
D-KSVD [53]	49.6	59.5	65.1	68.6	71.1	73.0
LC-KSVD1 [42]	53.5	61.9	66.8	70.3	72.1	73.40
LC-KSVD2 [42]	54.0	63.1	67.1	70.5	72.3	73.40
SVM-MKL [41]	51.2	62.4	67.1	69.8	72.7	74.6
Kernel SRC	50.4	60.8	66.5	69.2	72.0	74.1
SRC [10]	48.8	60.1	64.9	67.7	69.2	70.7
MKL-SRC	**54.6**	**64.9**	**69.3**	**72.0**	**74.2**	**75.7**

TABLE II: Accuracy (%) on the Caltech 101 object recognition dataset.

outperforms the other discriminative approaches such as LC-KSVD and D-KSVD.

In Fig. 6, we plot the learned kernel weights for the experiment when 30 samples per class are used for training. As can be seen from this plot, our method is able to learn the optimal combinations of base kernels directly from data.

To further analyze our results, we plot the confusion matrix in Fig. 7(a) and per class accuracy in Fig. 7(b) in the case when 30 samples per class are used for training. There are in total 11 classes that result in 100% accuracy. These images are shown in Fig. 8.

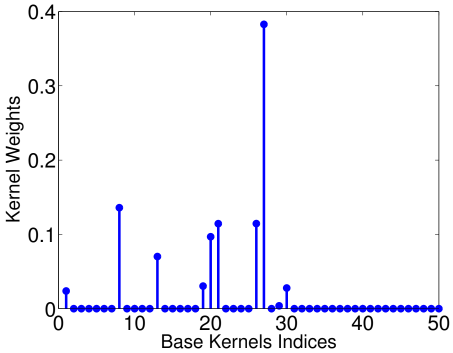


Fig. 6: Learned kernel weights for the Caltech101 (for 30 training samples).

C. Object Recognition using Intensity and Depth Data

Recently, there has been a growing interest in using both the intensity and the depth data for computer vision algorithms. For example, with Microsoft's Kinect camera one can capture videos of both color as well as corresponding depth data. The purpose of this experiment is to demonstrate that our algorithm can naturally be extended to the multi-modal features. We use the same set of base kernels for intensity as well as depth images to compute the kernel matrices. This can be viewed as a single modality but with twice as many number of base kernels.

In this experiment, we use the RGB-D dataset of University of Washington [39] which consists of 51 categories. Few examples of pairs of color and depth images from this dataset are shown in Fig. 9. Most of the depth images are noisy as shown in the second row of the figure. Hence, we apply a recursive median filter to remove the missing values. Processed images are shown in the third row of Fig. 9.

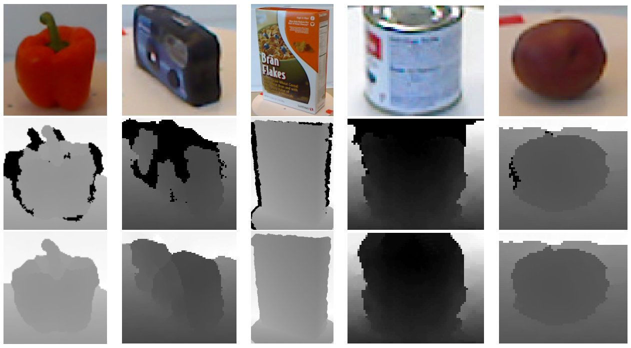
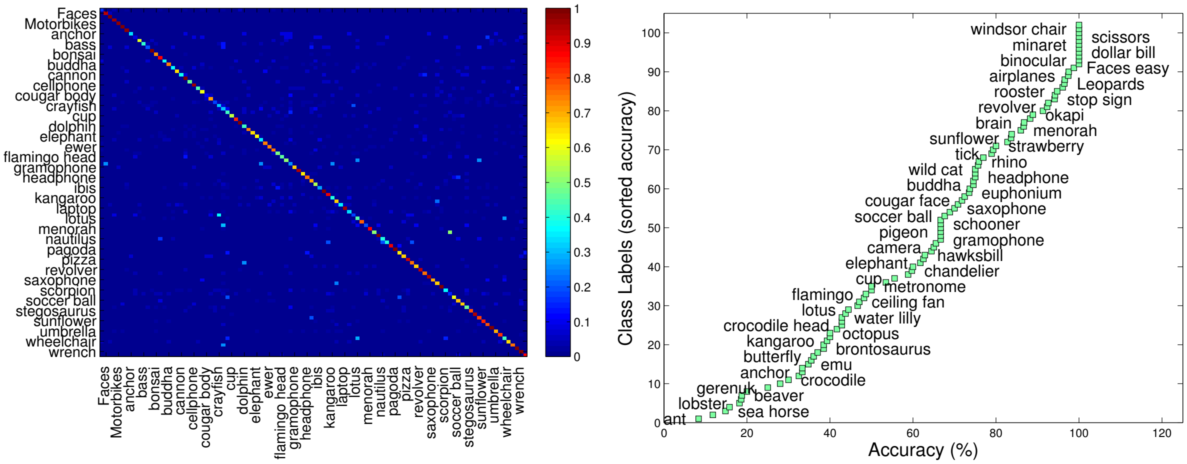


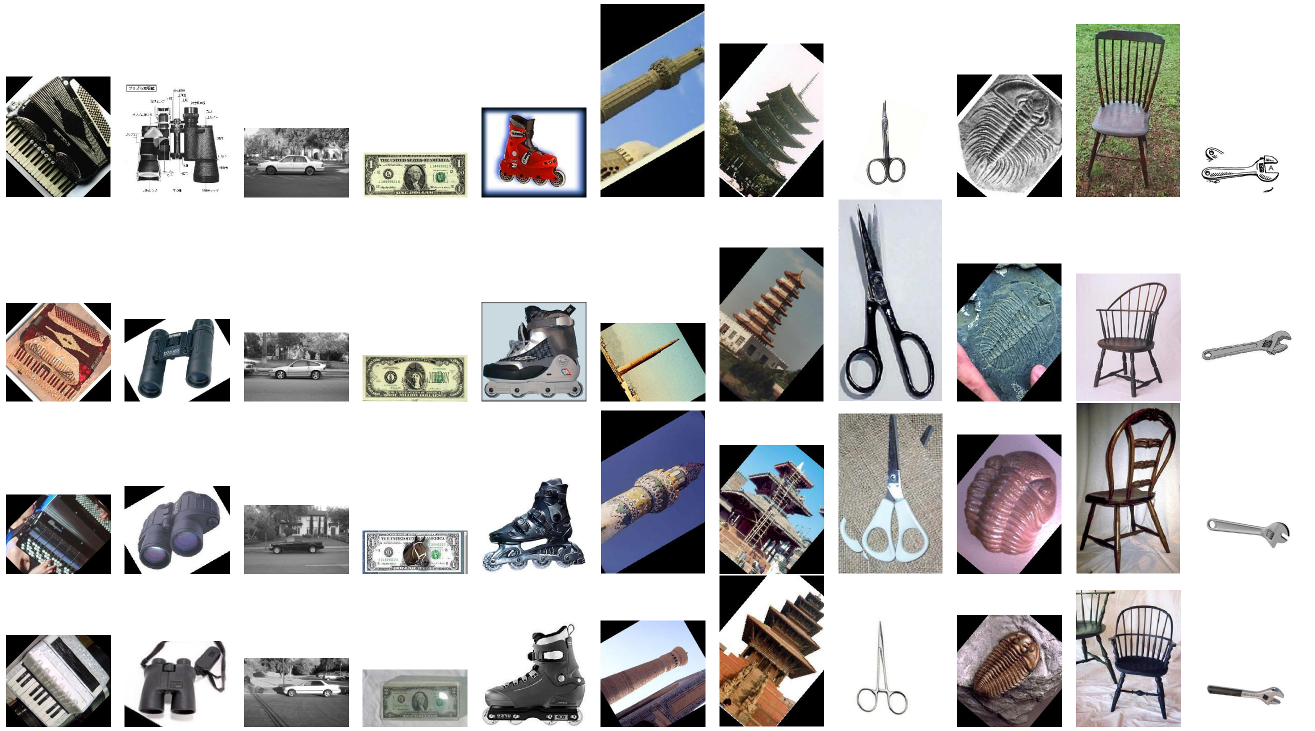
Fig. 9: Example images of the RGBD dataset. First row shows the intensity images, second row displays the corresponding depth images, and the third row is the denoised version of the second row after applying the recursive median filter.

We test our algorithm on the subset of the dataset by randomly selecting 10 images for training and 10 images for testing from each category. As is common in object recognition, we use bag-of-words (BoW) features for intensity as well as depth images. For the intensity images, we use BoW of 128 dimensional SIFT features [54] and 1000 clusters using the k-means algorithm for computing the bags. For the depth



(a) (b)

Fig. 7: Results on the Caltech 101 object dataset. (a) Confusion matrix. (b) Per class accuracy.

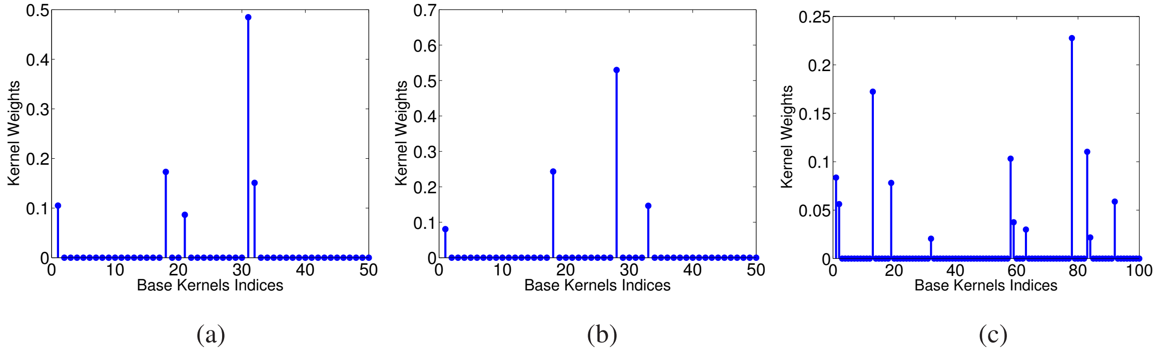


accordion, binocular, car_side, dollar_bill, inline_skate, minaret, pagoda, scissors, trilobite, windsor_chair, wrench

Fig. 8: Example images from 11 categories of the Caltech101 dataset that achieve 100% accuracy.

features, we compute the dense spin image [55] features on the depth data at each 3D point. We use the radius of 0.1 to compute neighbors at each point, and bin size of 16 to compute the spin images. Finally, BoW features are computed for each depth image with 1000 bags computed using the k-means algorithm.

Comparison of all the methods using the intensity features alone are shown in the second column of Table III. As can be seen from this column, our method performs more than 4% better than linear SRC. In this experiment, kernel SRC with non-optimal choice of the kernel performs a little worse than linear SRC. Next, we perform the same experiment on the depth data alone. Results of this experiment are summarized in the third column of Table III. Our method performs slightly better than the other methods on the depth data as well. Next, we demonstrate that multiple features can improve the performance of the classifier. To this purpose, we compute BoW on HOG features and HSV color histograms of intensity images and evaluate our algorithm against SRC and kernel SRC. Again, our algorithm can naturally be extended to incorporate multiple features by computing kernel matrices for each of these features. With 3 features and 50 base kernels for



(a) (b) (c)

Fig. 10: Learned kernel weights for RGBD dataset: (a) Intensity data. (b) Depth data. (c) Intensity and depth data.

Algorithms	Intensity features	Depth features	Intensity and depth features
SVM	71.96	74.90	84.11
NN	69.80	75.88	86.08
SVM-MKL [41]	72.75	78.24	86.07
SRC	70.00(72.16)	79.80	86.27
Kernel SRC	69.80(72.54)	80.00	86.60
MKL-SRC	**74.12(75.88)**	**81.37**	**87.65**

TABLE III: Accuracy (%) on the RGB-depth object dataset. In parentheses, we show the accuracy for multiple features (see text for details).

each of them, we have total 150 kernel matrices and learned $\boldsymbol{\eta}$ is 150 dimensional weight vector. The accuracy with these features combined have been shown in the parentheses in Table III.

Finally, we demonstrate how our algorithm can be extended when both the intensity and the depth features are available. For the non MKL based methods, we concatenate the intensity and the depth features for classification. This is equivalent to giving equal weights to both of the features. On the other hand, when we look at the kernel matrices from both the modalities, we can view them as twice as many base kernel matrices. Since in our case we use 50 base kernels, combined intensity and depth features results in 100 base kernel matrices. Learning weights for each kernel matrix automatically learns the optimal weights for the modalities. To further elaborate this point, we show the kernel weights in Fig. 10(a)-(c) for the intensity, depth, and their combination, respectively. Fig. 10(c) has 100 base kernel indices, out of which first 50 correspond to the intensity feature base kernels while the last 50 are for the depth feature base kernels. As can be seen from this figure, the weight given to the depth feature base kernels is more than the weights for the intensity features. This can be explained by the observation that the depth feature alone results in higher accuracy than the intensity feature. Results of the combined features are shown in the fourth column of Table III. Again, our method performs better than the other methods on this combined dataset.

D. Gender Recognition

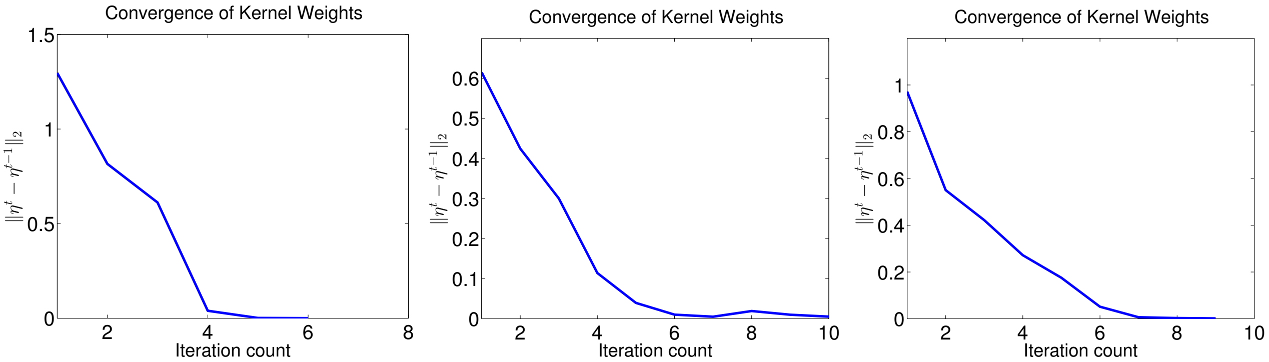
In the final set of experiments, we evaluate our algorithm on the gender recognition task. Towards this purpose, we use the AR Face dataset [40] that consists of 126 individuals with frontal faces captured in two sessions with different illuminations, expressions and occlusions. We choose 50 male subjects and 50 female subjects and 14 faces per subject from both sessions. Next, we train our algorithm, with first 25 males subjects and 25 female subjects and test our method with the remaining 25 male and 25 female subjects. The feature dimension was reduced to 300 using the principle component analysis. Comparison of our method with that of different methods is summarized in Table IV. Note that our method not only outperforms linear and non-linear SRC but it perform better than the state-of-the-art discriminative dictionary learning method such as [53].

SVM	NN	SVM-MKL [41]	DKSVD [53]	Kernel SRC	SRC	MKL-SRC
92.4	90.7	93.1	86.1	94.1	93.0	**95.4**

TABLE IV: Accuracy (%) on the gender recognition task using the AR face dataset.

E. On the Convergence of the Proposed Method

The proposed method iterates until the kernel weights converge or the maximum number of iterations are reached. So, a natural question about the convergence of the algorithm arises. Do the kernel weights converge and whether the classification accuracy actually improves over iterations? To answer these questions, let us closely look at what happens at each iteration. At every iteration, a new kernel is added only if it can correctly predict the labels of the subset of those samples that are incorrectly predicted using current kernel. In case of multiple choices, we pick the kernel with maximum accuracy. Intuitively, adding this new kernel should complement the current kernel and improve the accuracy. However, it is likely that the new kernel might wrongly predict the labels of those samples which are correctly predicted by the current kernel. Hence, we adjust the weight of the current kernel and the new kernel in proportion to the number of correctly predicted samples as explained in section III and illustrated in Fig. 2. Although, there is no theoretical guarantee that the combination of two kernels should improve upon the individual kernels, we empirically observe that combining the



(a) (b) (c)

Fig. 11: Convergence of kernel weights ($\|\boldsymbol{\eta}^t - \boldsymbol{\eta}^{t-1}\|_2$): (a) Caltech 101 dataset. (b) RGBD dataset. (c) Gender recognition on AR dataset.

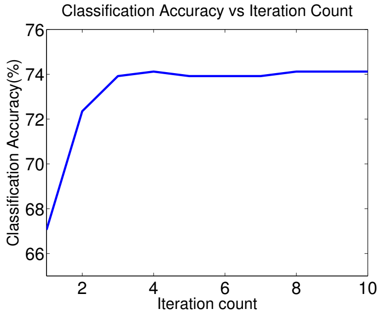


Fig. 12: Improvement of classification accuracy over iterations (RGBD dataset using intensity features).

kernel as proposed improves the overall classification accuracy at each iteration, as shown in Fig. 12 for RGB-D dataset.

Note that the kernel coefficients are updated at each iteration only if a new kernel complements the current kernel by correctly predicting the labels of the those samples where current kernel fails. As we add more kernels, the number of correctly predicted samples increases which, intuitively, results in reduced scope of further gain in later iterations. We can imagine that eventually, no kernel can correctly predict any significant subset of training data which is not already done by current kernel. This intuition is corroborated with experimental evaluation on all the three datasets. In Fig. 11 we plot $\|\boldsymbol{\eta}^t - \boldsymbol{\eta}^{t+1}\|_2$ and observe the quick convergence of kernel weights.

V. Conclusion

The SRC method works by computing the sparse coefficients of a test sample directly from the training data and does not require any prior training. However, for most of the applications, training can be useful provided that there is no over-fitting. In this paper, we have introduced a training stage to SRC that can learn the optimal kernel and significantly enhance the classification performance of SRC. The resulting algorithm alternates between learning sparse codes and kernel function weights. Various experiments on popular object recognition data sets have shown that our method is efficient and can perform significantly better than SRC, kernel SRC and SVM-based MKL methods.

Even though, in this paper, we used a linear SRC as a base learner for MKL, it is possible to learn discriminative SRC and kernel weights simultaneously by adapting a discriminative SRC in our formulation. One can also adapt dictionary learning methods in our MKL formulation. It remains an interesting topic for future work to develop and analyze the accuracy of a discriminative dictionary learning-based MKL algorithm for classification.

Acknowledgment

This work was partially supported by an ONR grant N00014-12-1-0124.

References

[1] R. Rubinstein, A. Bruckstein, and M. Elad, “Dictionaries for sparse representation modeling,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045 –1057, june 2010.

[2] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, “Sparse representation for computer vision and pattern recognition,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031 –1044, june 2010.

[3] M. Elad, M. Figueiredo, and Y. Ma, “On the role of sparse and redundant representations in image processing,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972 –982, june 2010.

[4] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[5] S. Chen, D. Donoho, and M. Saunders, “Atomic decomposition by basis pursuit,” *SIAM J. Sci. Comp.*, vol. 20, no. 1, pp. 33–61, 1998.

[6] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition,” *1993 Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

[7] J. A. Tropp, “Greed is good: Algorithmic results for sparse approximation,” *IEEE Trans. Info. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[8] A. M. Bruckstein, D. L. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images,” *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[9] K. Etemand and R. Chellappa, “Separability-based multiscale basis selection and feature extraction for signal and image classification,” *IEEE Transactions on Image Processing*, vol. 7, no. 10, pp. 1453–1465, Oct. 1998.

[10] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210 –227, feb. 2009.

[11] V. M. Patel and R. Chellappa, "Sparse representations, compressive sensing and dictionaries for pattern recognition," in *Asian Conference on Pattern Recognition(ACPR)*, 2010.

[12] V. M. Patel, R. Chellappa, and M. Tistarelli, "Sparse representations and random projections for robust and cancelable biometrics," *International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, Guangzhou, China, Dec. 2010.

[13] J. K. Pillai, V. M. Patel, R. Chellappa, and N. Ratha, "Secure and robust iris recognition using random projections and sparse representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1877–1893, Sept. 2011.

[14] L. Zhang, W.-D. Zhou, P.-C. Chang, J. Liu, Z. Yan, T. Wang, and F.-Z. Li, "Kernel sparse representation-based classifier," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1684 –1695, april 2012.

[15] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Sparse representation with kernels," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 423 –434, feb. 2013.

[16] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, 2012.

[17] H. Qi and S. Hughes, "Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements," in *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, may 2011, pp. 3940 –3943.

[18] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, July 2011.

[19] V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Sparsity-motivated automatic target recognition," *Applied Optics*, vol. 50, no. 10, April 2011.

[20] H. Li, Y. Gao, and J. Sun, "Fast kernel sparse representation," in *International Conference on Digital Image Computing Techniques and Applications*, Dec. 2011, pp. 72 –77.

[21] X.-T. Yuan and S. Yan, "Visual classification with multi-task joint sparse representation," in *Computer Vision and Pattern Recognition*, 2010.

[22] S. Gao, I. W. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *European Conference on Computer Vision (ECCV)*, vol. 6314, 2010.

[23] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Sparse embedding: a framework for sparsity promoting dimensionality reduction," in *European Conference on Computer Vision (ECCV)*, 2012.

[24] A. Shrivastava, H. V. Nguyen, V. M. Patel, and R. Chellappa, "Design of non-linear discriminative dictionaries for image classification," in *Asian Conference on Computer Vision (ACCV)*, 2012.

[25] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[27] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 217 –231, jan. 2013.

[28] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[29] C. A. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Mach. Learn. Res.*, vol. 6, pp. 1099–1125, Dec. 2005.

[30] C. S. Ong, A. J. Smola, and R. C. Williamson, "Learning the kernel with hyperkernels," *J. Mach. Learn. Res.*, vol. 6, pp. 1043–1071, Dec. 2005.

[31] A. Zien and C. S. Ong, "Multiclass multiple kernel learning," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07, 2007, pp. 1191–1198.

[32] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *International Conference on Machine Learning*, 2010, pp. 1175–1182.

[33] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *International Conference on Computer Vision (ICCV)*, 2007.

[34] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *International Conference on Computer Vision (ICCV)*, 2009.

[35] H. Tanabe, T. B. Ho, C. H. Nguyen, and S. Kawasaki, "Simple but effective methods for combining kernels in computational biology," in *IEEE International Conference on Research, Innovation and Vision for the Future*, july 2008, pp. 71 –78.

[36] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Neural Information Processing Systems*, 2001, pp. 367–373.

[37] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 6, no. 2, pp. 190–199, Apr. 2009.

[38] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," 2004.

[39] K. Lai, L. Bo, X. Ren, and D. Fox, "Sparse Distance Learning for Object Recognition Combining RGB and Depth Information," in *IEEE International Conference on on Robotics and Automation (ICRA)*, 2011.

[40] A. Martinez and R. Benavente, "The ar face database," *CVC Technical Report No. 24,*, June 1998.

[41] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research*, 2008.

[42] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*. IEEE, 2011.

[43] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[44] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[45] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," in *CIT Technical Report 7694*, 2007.

[46] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[47] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[48] D.-S. Pham and S. Venkatesh, "Joint learning and dictionary construction for pattern recognition," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[49] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in *European Conference on Computer Vision (ECCV)*, 2008.

[50] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[51] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[52] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: Design of dictionaries for sparse representation," in *IN: PROCEEDINGS OF SPARS'05*, 2005, pp. 9–12.

[53] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[54] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[55] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433 – 449, May 1999.

PLACE
PHOTO
HERE

Ashish Shrivastava received B.E. (Hons.) degree in electrical and electronics engineering (EEE) from Birla Institute of Technology and Science (BITS), Pilani, India in 2005.He is a PhD candidate in electrical and computer engineering at the University of Maryland, College Park. Before joining PhD he worked with Texas Instruments, Bangalore from 2005 to 2008. He is a student member of IEEE. His research interests are in Computer Vision and Machine Learning.

PLACE
PHOTO
HERE

Vishal M. Patel is a member of the research faculty at the University of Maryland Institute for Advanced Computer Studies (UMIACS). He received the B.S. degrees in Electrical Engineering and Applied Mathematics (with honors) and the M.S. degree in Applied Mathematics from North Carolina State University, Raleigh, NC, in 2004 and 2005, respectively. He received his Ph.D. from the University of Maryland, College Park, MD, in Electrical Engineering in 2010. He received an ORAU postdoctoral fellowship in 2010. His research interests are in signal processing, computer vision and pattern recognition with applications to biometrics and imaging. He is a member of Eta Kappa Nu, Pi Mu Epsilon and Phi Beta Kappa.

PLACE
PHOTO
HERE

Rama Chellappa received the B.E. (Hons.) degree in Electronics and Communication Engineering from the University of Madras, India and the M.E. (with Distinction) degree from the Indian Institute of Science, Bangalore, India. He received the M.S.E.E. and Ph.D. Degrees in Electrical Engineering from Purdue University, West Lafayette, IN. During 1981-1991, he was a faculty member in the department of EE-Systems at University of Southern California (USC). Since 1991, he has been a Professor of Electrical and Computer Engineering (ECE) and an affiliate Professor of Computer Science at University of Maryland (UMD), College Park. He is also affiliated with the Center for Automation Research and the Institute for Advanced Computer Studies (Permanent Member) and is serving as the Chair of the ECE department. In 2005, he was named a Minta Martin Professor of Engineering. His current research interests span many areas in image processing, computer vision and pattern recognition. Prof. Chellappa is a recipient of an NSF Presidential Young Investigator Award and four IBM Faculty Development Awards. He received two paper awards and the K.S. Fu Prize from the International Association of Pattern Recognition (IAPR). He is a recipient of the Society, Technical Achievement and Meritorious Service Awards from the IEEE Signal Processing Society. He also received the Technical Achievement and Meritorious Service Awards from the IEEE Computer Society. He is a recipient of Excellence in teaching award from the School of Engineering at USC. At UMD, he received college and university level recognitions for research, teaching, innovation and mentoring undergraduate students. In 2010, he was recognized as an Outstanding ECE by Purdue University. Prof. Chellappa served as the Editor-in-Chief of IEEE Transactions on Pattern Analysis and Machine Intelligence and as the General and Technical Program Chair/Co-Chair for several IEEE international and national conferences and workshops. He is a Golden Core Member of the IEEE Computer Society, served as a Distinguished Lecturer of the IEEE Signal Processing Society and as the President of IEEE Biometrics Council. He is a Fellow of IEEE, IAPR, OSA, AAAS and ACM and holds four patents