creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

Abstract

By implicitly maximizing the gap between classes in the reproducing kernel Hilbert space (RKHS), a multiple kernel learning (MKL) is formulated as a linear programming in this paper.

For each sample, my method tries to enforce the distance between intra-class and inter-class samples in RKHS to be as distant as possible. In my method, each training sample imposes at most r constraints for the linear programming where r is the number of different kernel types. Unlike previous methods of multiple kernel learning, the proposed method does not need a large amount of computations. My method is compared with various methods of MKL to prove its efficiency of finding a good kernel mixture parameter.

Keywords: Multiple Kernel Learning, Reproducing Kernel Hilbert Space, Kernel Method, Kernel Alignment, Kernel Trick.

Student Number: 2014-24829

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The support vector machine(SVM) has posed itself as a powerful classifier tool for binary classification problems [2]. For samples of N independent and identically distributed samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i$ is the D-dimensional input vector and $y_i \in \{-1, +1\}$ is each sample's class label, SVM finds the optimal separating hyperplane with the maximum margin in the feature space induced by the mapping function $\Phi : \mathbb{R}^D \mapsto \mathbb{R}^S$. The optimal hyperplane function is $f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$. The classifier can be trained by solving the following quadratic optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{with respect to} && \mathbf{w} \in \mathbb{R}^S, \xi \in \mathbb{R}_+^N, b \in \mathbb{R} \\ &\text{subject to} && y_i(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned} \tag{1.1}$$

where $\mathbf{w}$ is the vector of weight coefficients, C represents trade-off parameter between model complexity and classification error, ξ is the vector composed of slack variables that relax the constraints of the canonical hyperplane equation, and b is the bias term of

the discriminant. Constrained minimization of $\frac{1}{2}\|\mathbf{w}\|_2^2$ can be solved by the Lagrangian dual formulation:

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \underbrace{\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle}_{k(\mathbf{x}_i,\mathbf{x}_j)} \\ &\text{with respect to} && \alpha \in \mathbb{R}_+^N \\ &\text{subject to} && \sum_{i=1}^{N}\alpha_i y_i = 0, C \geq \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{1.2}$$

where k is the *kernel function* that is $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ while α is the vector of lagrangian dual variables. Solving this, the optimal hyperplane becomes

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) + b. \tag{1.3}$$

Therefore, classification of an unknown example $\mathbf{x}$ is performed by computing the weighted sum of the kernel function with respect to the support vectors $\mathbf{x}_i$.

In the fields of pattern recognition and machine learning, kernel method comes as a very useful trick for modelling and classifying a given data because of its conceptual simplicity and good performance. From many kernel function types, the three major kernel functions commonly used in the literature are the linear kernel, the polynomial kernel, and the RBF kernel (or the Gaussian kernel), which are defined in respective order:

$$\begin{aligned} k_{LIN}(\mathbf{x}_i,\mathbf{x}) &= \langle\mathbf{x}_i,\mathbf{x}_j\rangle \\ k_{POL}(\mathbf{x}_i,\mathbf{x}) &= (\langle\mathbf{x}_i,\mathbf{x}_j\rangle + 1)^q, \quad q \in \mathbb{N} \\ k_{GAU}(\mathbf{x}_i,\mathbf{x}) &= \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/s^2), \quad s \in \mathbb{R}_{++} \end{aligned} \tag{1.4}$$

In the kernel trick, selecting a kernel function and its parameter(s) (for example, q in k_{POL} or s in k_{GAU}) is an important task, but it usually requires multiple unpromising

heuristic trials. Such selection of a kernel function and its parameters can be considered as a feature extraction process. However, in many real life applications, a single type of feature cannot provide enough information about the data and it thus should be considered more practical to combine multiple feature representations to catalyst the performance of a learning algorithm. For example, in image classification problems, various feature descriptors such as colors, shapes, and textures are combined for better classification performance.

Likewise, in kernel methods, recent developments have suggested to use multiple kernels instead of selecting one specific kernel function and its corresponding parameters as in [1] [3]. One can approach this problem by considering linear combinations of s kernels, as in

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{s} \mu_m k^{(m)}(\mathbf{x}_i, \mathbf{x}_j), \tag{1.5}$$

with $\mu \triangleq [\mu_1, \cdots, \mu_s]^T$ is the weight vector for s different kernel functions $k^{(m)}(\cdot,\cdot), m = 1, \cdots, s$, and $\mathbf{x}_i$ and $\mathbf{x}_j$ are input data samples. This idea of multiple kernel learning (MKL) consists of learning the weight μ_m for each base kernel, rather than selecting a single kernel and optimizing the corresponding kernel parameters.

MKL was first introduced by Lanckriet et al. [4] for binary classification problems where mainly two formulations are presented. In the first formulation, they try to optimize the weight vector μ in the process of training the support vector machine (SVM). Depending on the domain of μ, MKL algorithms can be categorized as L_1 regularized MKL ($\{\mu \in \mathbb{R}_+^s : ||\mu||_1 = \sum_{m=1}^{s} |\mu_m| \leq 1\}$) [5] and L_p regularized MKL ($\{\mu \in \mathbb{R}_+^s : ||\mu||_p = (\sum_{m=1}^{s} |\mu_m|^p)^{\frac{1}{p}} \leq 1\}$) [6].

The second formulation in [4] is based on maximizing the kernel alignment that mea-

sures similarity between the target kernel and the kernel to be optimized. This idea of kernel alignment was firstly introduced by Cristianini et al. [7] and Cortes et al. [8] used centered version of a kernel matrix in calculating the kernel alignment.

Chapter 2

Related Works

Many MKL algorithms have been previously proposed, and they differ each other in various aspects. These learning methods among them can be categorized based on whether it is heuristically approached or optimized through the solvers. Some of the algorithms are based on kernels that are linearly combined, others are non-linearly combined such as polynomial combination, and others learn the combination of both based on a given data. Some optimize for a target kernel, and some minimize the structural risk.

ABMKL, CABMKL and my method build the learning system, directly optimizing the final kernel matrix, while other algorithms try to minimize the sum of a regularization term and an error term. Later part in this paper, the proposed method proves its efficiency through experiments on classic classification data by outperforming the algorithms noted within this chapter.

2.1 Alignment-based MKL (ABMKL)

The idea of Kernel alignment was firstly introduced by Cristianini et al. [7] to which my method is closely related. Their framework is based on maximizing a similarity measure, *alignment*, between a target-kernel and a combination of multiple kernels. For the binary classification problems with n training samples, the target matrix is set as $\mathbf{K}_t = \mathbf{y}\mathbf{y}^T$ where $\mathbf{y} = [y_1, \cdots, y_n]^T \in \{\pm 1\}^n$ is the class vector. Then, the similarity alignment between the kernel matrix $\mathbf{K}$ and the target matrix $\mathbf{K}_t$ becomes

$$A(\mathbf{K}, \mathbf{K}_t) = \frac{\langle \mathbf{K}, \mathbf{K}_t \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{K}_t, \mathbf{K}_t \rangle_F}} = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^T \rangle_F}{\sqrt{n \langle \mathbf{K}, \mathbf{K} \rangle_F}}, \tag{2.1}$$

and the method of kernel alignment tries to maximize this similarity. Here, $\langle \cdot, \cdot \rangle_F$ denotes Frobenius inner product between Gram matrices, $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \text{trace}(\mathbf{K}_1^T \mathbf{K}_2) = \sum_{i,j=1}^{n} k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j)$.

In [9], Qiu and Lane propose a simple heuristic way of selecting kernel weights, μ_m, using kernel alignment:

$$\mu_m = \frac{A\left(\mathbf{K}_m, \mathbf{y}\mathbf{y}^T\right)}{\sum_{p=1}^{s} A\left(\mathbf{K}_p, \mathbf{y}\mathbf{y}^T\right)} \tag{2.2}$$

Kernel Alignment method was later developed into a more efficient optimization setting by Lankriet et al. [4]. They developed Cristianini's idea into a QCQP by implementing multiple kernel combination into alignment maximization problem. Adding the constraints that $\mathbf{K}$ is a linear combination of fixed kernel matrices and the vector of

weighting coefficients to be $\mu \geq 0$, the learning problem in the paper yields

$$\begin{aligned} \underset{\mu}{\text{maximize}} \quad & \left\langle \sum_{m=1}^{s} \mu_m \mathbf{K}^{(m)}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} \quad & \left\langle \sum_{m=1}^{s} \mu_m \mathbf{K}^{(m)}, \sum_{p=1}^{s} \mu_p \mathbf{K}^{(p)} \right\rangle_F \leq 1, \\ & \mu \geq 0. \end{aligned} \tag{2.3}$$

Then, the followings can be defined:

$$\begin{aligned} \left\langle \sum_{m=1}^{s} \mu_m \mathbf{K}^{(m)}, \mathbf{y}\mathbf{y}^T \right\rangle_F &= \sum_{m=1}^{s} \mu_m \langle \mathbf{K}^{(m)}, \mathbf{y}\mathbf{y}^T \rangle_F \\ &= \mu^T \mathbf{q}, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \left\langle \sum_{m=1}^{s} \mu_m \mathbf{K}^{(m)}, \sum_{p=1}^{s} \mu_p \mathbf{K}^{(p)} \right\rangle_F &= \sum_{m,p=1}^{s} \mu_m \mu_p \langle \mathbf{K}^{(m)}, \mathbf{K}^{(p)} \rangle_F, \\ &= \mu^T \Lambda \mu \end{aligned} \tag{2.5}$$

where $\mathbf{q} = [q_1, \cdots, q_s]$ with $q_i = \left\langle \mathbf{K}_i, \mathbf{y}\mathbf{y}^T \right\rangle_F$, the numerator of (2.1), and $\Lambda = [\lambda_{ij}]_{i,j \in \{1,\cdots,s\}}$ with $\lambda_{ij} = \left\langle \mathbf{K}_i, \mathbf{K}_j \right\rangle_F$, the denominator of (2.1), in order to reduce the number of constraints. Thus, the alignment maximization with multiple kernel combination in the paper becomes

$$\begin{aligned} \max_{\mu} \quad & \mu^T \mathbf{q} \\ \text{subject to} \quad & \mu^T \Lambda \mu \leq 1. \end{aligned} \tag{2.6}$$

Instead of maximizing the alignment measure, He et al [10] choose to optimize the distance between the combination of multiple kernels and the target ideal kernel. The

optimization setting proposed in their paper is

$$\begin{aligned} \text{minimize} \quad & \langle \mathbf{K} - \mathbf{y}\mathbf{y}^T, \mathbf{K} - \mathbf{y}\mathbf{y}^T \rangle_F^2 \\ \text{subject to} \quad & \mathbf{K} = \sum_{m=1}^{s} \mu_m \mathbf{K}^{(m)}, \\ & \sum_{m=1}^{s} \mu_m = 1, \\ & \mu \geq 0. \end{aligned} \tag{2.7}$$

2.2 Centered-alignment-based MKL (CABMKL)

For the issues of the unbalanced datasets, Cortes et al. [11] define a new concept of alignment, *centered-kernel-alignment*. The centered-alignment of two kernels, $\mathbf{K}_1$ and $\mathbf{K}_2$, is defined as follows:

$$CA(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1^c, \mathbf{K}_2^c \rangle_F}{\sqrt{\langle \mathbf{K}_1^c, \mathbf{K}_1^c \rangle_F \langle \mathbf{K}_2^c, \mathbf{K}_2^c \rangle_F}}, \tag{2.8}$$

where $\mathbf{K}^c$ is the centered version of $\mathbf{K}$ and can be calculated as

$$\mathbf{K}^c = \mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{1}^T + \frac{1}{N^2}(\mathbf{1}^T\mathbf{K}\mathbf{1})\mathbf{1}\mathbf{1}^T \tag{2.9}$$

where $\mathbf{1}$ is the vector of ones with proper dimension. The optimization in the paper is set as:

$$\begin{aligned} \text{maximize} \quad & CA(\mathbf{K}_m, \mathbf{y}\mathbf{y}^T) \\ \text{subject to} \quad & ||\mu||_2 = 1 \end{aligned} \tag{2.10}$$

This setting can be expanded further and have an analytical solution :

$$\mu = \frac{\mathbf{M}^{-1}\mathbf{a}}{||\mathbf{M}^{-1}\mathbf{a}||_2} \tag{2.11}$$

where $\mathbf{M} = \{\langle \mathbf{K}_m^c, \mathbf{K}_h^c \rangle_F\}_{m,h=1}^P$ and $\mathbf{a} = \{\langle \mathbf{K}_m^c, \mathbf{y}\mathbf{y}^T \rangle_F\}_{m=1}^P$

Cortes et al. [11] add the constraints on the kernel weights to be nonnegative, and change the setting as:

$$\begin{aligned} &\text{minimize} \quad \mathbf{v}^T\mathbf{M}\mathbf{v} - 2\mathbf{v}^T\mathbf{a} \\ &\text{subject to} \quad \mathbf{v} \in \mathbb{R}_+^P \end{aligned} \tag{2.12}$$

with the kernel weights defined as $\mu = \mathbf{v}/||\mathbf{v}||_2$.

Since the linearly learning method in the feature space is not directly related with centering the kernel values, their definition of centered-alignment aims to consider the correlation between the random variables of training kernel matrix and the ideal kernel matrix.

2.3 MKL, Simple MKL (SimpleMKL), Generalized MKL (GMKL)

Some studies optimize the kernel weights by following the structural risk minimization framework through linear approaches as in MKL and SIMPLEMKL or through nonlinear approach as in GMKL with kernel weights.

Bach et al. in [12] propose the modified version of SVM primal formulation.

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\Big(\sum_{m=1}^{s} d_m||\mathbf{w}_m||_2\Big)^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} \quad \mathbf{w}_m \in \mathbb{R}^{S_m}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R} \\ &\qquad\qquad y_i\Big(\sum_{m=1}^{s} d_m\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\Big) \geq 1 - \xi_i \quad \forall i, \end{aligned} \tag{2.13}$$

defining the feature space constructed by $\Phi_m(\cdot)$ has the dimensionality of S_m and the weight d_m Considering this as a second-order cone programming (SOCP), the dual for-

mulation can be obtained as follows:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\gamma^2 - \sum_{m=1}^{N} \alpha_i \\
\text{subject to} \quad & \gamma \in \mathbb{R}, \alpha \in \mathbb{R}_+^N \\
& \gamma^2 d_m^2 \geq \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k^{(m)}(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m \\
& \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i.
\end{aligned} \tag{2.14}$$

The benefits of this formulation is that Bach et al. in [12] look for the SMO-like algorithm through implementation of Moreau-Yoshida regularization term, $1/2\sum_{m=1}^{s} a_m^2 ||\mathbf{w}_m||_2^2$. The algorithmic solutions similar to sequential minimization optimization (SMO), such as this formulation, suffice for larger problems with large number of kernels and data points.

Rakotomamonjy et al. in [13] and [14] set up a different primal formulation of SVM for MKL:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{s} \frac{1}{\mu_m} ||\mathbf{w}_m||_2^2 + C\sum_{i=1}^{N} \xi_i \\
\text{subject to} \quad & \mathbf{w}_m \in \mathbb{R}^{S_m}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R}, \mu \in \mathbb{R}_+^P \\
& y_i\Big(\sum_{m=1}^{s} \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\Big) \geq 1 - \xi_i \quad \forall i, \\
& \sum_{m=1}^{s} \mu_m = 1
\end{aligned} \tag{2.15}$$

and they define the SVM objective function $J(\mu)$ given μ to be

$$\begin{aligned} \text{minimize} \quad & J(\mu) = \frac{1}{2}\sum_{m=1}^{s}\frac{1}{\mu_m}||\mathbf{w}_m||_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & \mathbf{w}_m \in \mathbb{R}^{S_m}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R}, \mu \in \mathbb{R}_+^P \\ & y_i\Big(\sum_{m=1}^{s}\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\Big) \geq 1 - \xi_i \quad \forall i. \end{aligned} \tag{2.16}$$

Their optimization problem, called SIMPLEMKL, can be solved by the projected gradient method in a two-steps procedure. For the First step, the SVM optimization problem with the given μ is solved. Next, with the α found in the previous step, μ is updated using the corresponding gradient. The gradient update procedure is considered with the nonnegativity and normalization properties of μ.

Varma and Babu in [15] propose GMKL and take the weights of the hyperplane into account in their objective function by devising a loss function. Implementing a loss function such as a hinge loss for classification or ε-loss for regression, the proposed primal formulation for a binary classification problem becomes:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||\mathbf{w}_\mu||_2^2 + C\sum_{i=1}^{N}\xi_i + r(\mu) \\ \text{subject to} \quad & \mathbf{w}_\mu \in \mathbb{R}^{S_\mu}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R}, \mu \in \mathbb{R}_+^P \\ & y_i(\langle \mathbf{w}_\mu, \Phi_\mu(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned} \tag{2.17}$$

where $\Phi_\mu(\cdot)$ is the feature space that is defined by the combined kernel function $k_\mu(\cdot,\cdot)$ and $\mathbf{w}_\mu$ is the vector of weight coefficients, μ_m. With two additional regularization terms and a loss function, the formulation is not convex anymore. However, its solution strategy is the same as SIMPLEMKL's, which is a projected gradient method.

2.4 The Group Lasso-based MKL (GLMKL)

There can exist a group structure in a combination of kernels. When a group of kernels are computed on the same set of features and even if a non-zero weight to one of them is assigned, the features must be extracted in the testing stage. When kernels are composed of such a group structure, it is reasonable to pick all or none of them in the kernel combination [1]. Saketha Nath et al. in [16] suggest an MKL method that takes the group structure into account between the kernels. The paper proposes a formulation that enforces the L_∞-norm at the group level and the L_1-norm within each group, so that every group feed to the final learning stage, promoting sparsity among kernels in each group. MKL formulation from [17] by Kloft et al. and [18] by Xu et al. optimizes the generalization for arbitrary L_p-norms with $p \geq 1$. Both papers set an alternating optimization method for SVM solution at each iteration and update the kerenel weights as follows:

$$\mu_m = \frac{||\mathbf{w}_m||_2^{\frac{2}{p+1}}}{\left(\sum_{p=1}^{s} ||\mathbf{w}_p||_2^{\frac{2p}{p+1}}\right)^{\frac{1}{p}}} \tag{2.18}$$

where $||\mathbf{w}_m||_2^2 = \mu_m^2 \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k^{(m)}(\mathbf{x}_i^m, \mathbf{x}_j^m)$ from the dual formulation.

2.5 Non-linear MKL (NLMKL)

Cortes et al. in [19] proposes a polynomial combination of kernels that is nonlinear method based on KRR. Their definition of a kernel combination is :

$$k_\mu(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathbf{q}\in\mathcal{R}} \mu_1^{q_1}\mu_2^{q_2}\ldots\mu_P^{q_P} k_1(\mathbf{x}_i^1, \mathbf{x}_j^1)^{q_1} k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)^{q_2}\ldots k_P(\mathbf{x}_i^P, \mathbf{x}_j^P)^{q_P}. \tag{2.19}$$

$\mathcal{R}$ is defined as a space which is constructed by $\{\mathbf{q} : \mathbf{q} \in \mathbb{Z}_+^P, \sum_{m=1}^{s} q_m = 1\}$, and $\mu \in \mathbb{R}^P$. For the case when $d = 2$, the weight coefficients of this formulation can be optimized with the following problem :

$$\underset{\mu \in \mathcal{M}}{\text{minimize}} \quad \underset{\alpha \in \mathbb{R}^N}{\text{maximize}} \qquad -\alpha^T (\mathbf{K}_\mu + \lambda \mathbf{I})\alpha + 2\mathbf{y}^T \alpha \tag{2.20}$$

where two possible sets for $\mathcal{M}$ are L_1-norm and L_2-norm bounded sets defined as :

$$\begin{aligned} \mathcal{M}_1 &= \{\mu : \mu \in \mathbb{R}_+^P, ||\mu - \mu_0||_1 \leq \Lambda\} \\ \mathcal{M}_2 &= \{\mu : \mu \in \mathbb{R}_+^P, ||\mu - \mu_0||_2 \leq \Lambda\} \end{aligned} \tag{2.21}$$

with μ_0 and Λ are two model parameters. During iterations, kernel ridge regression is solved to acquire α, then updates μ with the gradients with the α obtained.

2.6 Localized MKL (LMKL)

For the pursuit of finding data-dependent algorithms, an addition of a generative probabilistic model is required according to Gönen and Alpadyn in [20]. They propose a data-dependent learning system called localized multiple kernel learning that calculates kernel weights based on a gating model. Their primal formulation is as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{s} ||\mathbf{w}_m||_2^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & \mathbf{w}_m \in \mathbb{R}^{S_m}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R}, \mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)} \\ & y_i \Big(\sum_{m=1}^{s} \mu_m(\mathbf{x}_i|\mathbf{V}) \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m) \rangle + b \Big) \geq 1 - \xi_i \quad \forall i, \end{aligned} \tag{2.22}$$

where the gating model is labeled as $\mu_m(\mathbf{x}_i|\mathbf{V})$, parametrized by $\mathbf{V}$. The gating model assigns the weighting coefficients. The combination of kernels for their proposal is

defined as :

$$k_\mu(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{s} \mu_m\big(\mathbf{x}_i|\mathbf{V}\big) k^{(m)}(\mathbf{x}_i^m, \mathbf{x}_j^m) \mu_m\big(\mathbf{x}_j|\mathbf{V}\big) \quad \forall m. \tag{2.23}$$

There are two gating models proposed in their paper. When the gating model is expressed using softmax function, then

$$\mu_m\big(\mathbf{x}|\mathbf{V}\big) = \frac{\Big(\exp(\langle \mathbf{v}_m, \mathbf{x}^{\mathcal{G}})\rangle + v_{m0}\Big)}{\sum_{p=1}^{s} \exp\Big(\langle \mathbf{v}_p, \mathbf{x}^{\mathcal{G}})\rangle + v_{p0}\Big)} \quad \forall m \tag{2.24}$$

with $\mathbf{V} = \{\mathbf{v}, v_{m0}\}_{m=1}^{s}$, and $\mathbf{x}^{\mathcal{G}} \in \mathbb{R}^{D_{\mathcal{G}}}$ represents the input samples in the feature space. There must be $P \times (D_{\mathcal{G}} + 1)$ parameters where $D_{\mathcal{G}}$ represents the gating feature space dimensionality. The softmax function tends to optimize to a single kernel active, while the sigmoid function optimizes to multiple kernels that share weights flexibly. The sigmoid function for the gating model is :

$$\mu_m\big(\mathbf{x}|\mathbf{V}\big) = \frac{1}{\exp\Big(\langle \mathbf{v}_m, \mathbf{x}^{\mathcal{G}})\rangle + v_{m0}\Big)} \quad \forall m. \tag{2.25}$$

The parameters for the gating models are updated through iterations with the gradient-descent method.

Most methods of MKL use convex optimization techniques such as quadratically constrained quadratic programming (QCQP) [4], second-order cone programming (SOCP) [12] [21], and quadratic programming (QP) [8]. To cope with larger problems, MKL is reformulated into different types of optimization problems such as semi-infinite programming (SIP) [5] [22], sub-gradient descent (SD) [14], and level method [23]. There are many other algorithms for MKL and extensive reviews of various approaches can be found in [1] and [3].

Besides, there are attempts to integrate dictionary learning with MKL to perform discriminative embedding in reproducing kernel Hilbert space (RKHS) induced by multiple kernels [24] [25]. Also, Ni et. al [26] try to select instances for a more discriminative kernel representation.

This paper formulates an MKL in a different way so that the distance between intra-class and inter-class instances in RKHS is maximized. My method is closely related to the kernel alignment approach but different from it in that the target kernel incorporates distance information in RKHS. The resulting target kernel has lots of *'don't care'* elements, thus the optimization is made to be easier. My method's formulation of MKL is a linear programming (LP) with at most sn constraints where s and n are the number of different kernel types and the number of training samples, respectively. Compared to previous methods of MKL that try to solve more complexed problems such as QCQP, SOCP, SIP, and so on, my method is faster because an LP is involved in.

Chapter 3

MKL by Gap-Maximization (MKL-GM)

3.1 Motivation: Kernel Target Alignment

The kernel target alignment methods can be considered as making the kernel as similar as possible with the target kernel. Assuming that the training data instances are sorted by their class labels and the binary class labels are $0/1$ instead of ± 1, the target kernel becomes a block diagonal matrix whose non-zero blocks are made up of all-one matrices (as shown in the left side of Fig. 3.1). In this case, each n^2 element k_{ij} of the output kernel matrix by MKL has a contribution to the objective function, i.e., kernel alignment tries to make $k_{ij} = \phi(x_i)^T \phi(x_j)$ large for the sample pairs with the same class label while keeping it small for the pairs with the different class labels. However, this objective is hard to meet because even in the high dimensional RKHS, it is difficult to place all the samples in a class close together while keeping the distance of the samples with

different class labels far away. Therefore, my method relax the optimization problem by focusing on the local connectivity of the samples as shown in the right side of Fig. 3.1. For each sample, I try to place some of the samples in the same class close together while at the same time, a large portion of the samples in different classes are located far apart. Furthermore, the proposed method tries to enhance the computational complexity of MKL by formulating MKL as an LP rather than QCQP as in (2.6).

$$\left[\begin{array}{ccccc|ccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \quad \left[\begin{array}{ccccc|ccccc} 1 & \cdot & \cdot & \cdot & \cdot & 0 & 0 & 0 & 0 & \cdot \\ 1 & 1 & \cdot & \cdot & \cdot & 0 & 0 & \cdot & \cdot & 0 \\ \cdot & 1 & 1 & \cdot & \cdot & 0 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \cdot & 0 & 1 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 0 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & 0 & 0 & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & 0 & \cdot & 0 & \cdot & \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & 0 & \cdot & 0 & \cdot & \cdot & \cdot & 1 & 1 \end{array}\right]$$

Figure 3.1: Motivation of the proposed method. Left: Examplary target kernel for the kernel alignment. Right: relaxed target kernel with lots of *don't care* positions deonted by $\cdot$. Note that only local connectivity is considered in the right while global connectivity is considered in the left.

Our method initiates from considering the similarity among feature vectors for each sample. Considering that each element of a kernel matrix measures the pairwise similarity between samples, it is natural to find kernel parameters μ that maximizes the minimum gap between intra-class similarity and the inter-class similarity.

Then the kernel alignment problem can be translated as

$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & -b \\ \text{subject to} \quad & \mathrm{sm}_{ji}(\mu) - \mathrm{sm}_{li}(\mu) \geq b, \forall i \in \{1, \cdots, n\}, \\ & \forall j \in \{j | y_j = y_i\}, \forall l \in \{l | y_l \neq y_i\}, \\ & \mathbf{1}^T \mu = 1, \mu \geq 0. \end{aligned} \tag{3.1}$$

Here, $\mathrm{sm}_{ab}(\mu)$ denotes similarity measure between two training samples x_a and x_b and $\geq$ is element-wise inequality. As a similarity measure, various measures can used. The kernel itself, i.e., $\mathrm{sm}_{ij} = k_{ij}$, or the distance-based measure can be the examples. In the second case, I can use negative squared distance (nsd) in RKHS, e.g., $\mathrm{sm}_{ij} = \mathrm{nsd}_{ij} \triangleq -\frac{1}{2}||\phi(x_i) - \phi(x_j)||^2$. Consider the samples x_i, x_j and x_l where x_i and x_j belongs to the same class while x_l does not. Then, the squared distance gap between x_j and x_l with respect to x_i can be calculated as

$$\begin{aligned} &\frac{1}{2}(||\phi(x_i) - \phi(x_l)||^2 - ||\phi(x_i) - \phi(x_j)||^2) \\ &\qquad = k_{ji} - k_{li} + \frac{1}{2}(k_{ll} - k_{jj}), \end{aligned} \tag{3.2}$$

and this can replace $\mathrm{sm}_{ji} - \mathrm{sm}_{li}$ in (3.1). Note that for a distance-based kernel such as radial basis function (RBF) kernels (Gaussian, Laplacian, ANOVA) similarity gap using nsd_{ij} is identical to that using k_{ij}. Note also that (3.1) is not restricted to binary classification problems and can equally apply to multi-class problems.

3.2 MKL-GM for the same type of kernels

Our formulation (3.1) is a simple LP with primal variable μ, but it has $n_p n_n n$ constraints where n_p and n_n are the number of samples for class 1 and -1 respectively that

satisfies $n = n_p + n_n$. Because of the huge number of constraints, for now, my method restricts our attention only on the RBF-type kernels such as Gaussian and Laplacian kernels where the distance ordering information within the original space is conserved in the kernel space. The proposed method is especially good for selecting kernel parameters among the same type of kernels with different kernel parameters.

In case of the Gaussian kernel

$$k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2}{\sigma^2}\right), \tag{3.3}$$

if I use k_{ij} (or equivalently nsd_{ij}) as a similarity measure, it depends on $\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2$ with σ^2 influencing only as an exponential scaling factor. Therefore the distance ordering information between samples in RKHS does not change with parameter σ^2.

Consider binary classification problems with training data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and its class labels $y_i \in \{-1, 1\}$. For this dataset, I first calculate the squared Euclidean distance matrix:

$$\mathbf{D} = \left[(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)\right]_{\forall i,j} = [d_{ij}]_{\forall i,j}, \tag{3.4}$$

where d_{ij} is the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. This distance matrix acts as a base similarity measure in the input space in my algorithm.

For each column i of distance matrix $\mathbf{D}$ which represents squared distances in the original space from $\mathbf{x}_i$ to others, I find the index of a sample $\mathbf{x}_j$ with maximum distance from $\mathbf{x}_i$ among the ones that share the same class labels with $\mathbf{x}_i$, i.e., $y_j = y_i$, and the index of a sample $\mathbf{x}_l$ whose distance is minimum among the ones with different class labels than $\mathbf{x}_i$, i.e., $y_l \neq y_i$. These indices j and l are stored in the i-th column of the integer-valued index matrix $\mathbf{M} \in \mathbb{I}^{2\times n}$.

Then this method tries to find the kernel mixture parameters μ in (1.5) that maximizes the difference between the elements of the mixed kernel located in the indices m_{1i} and m_{2i} for all $i \in \{1, \cdots, n\}$.

Such approach reduces a significant amount of computational burdens of MKL methods. To build a kernel matrix, it usually requires calculations in the order of n^2 along with d-vector operations during the training process. The computations increase even more up to pdn^2 for multiple kernel learning with p kernels. However, because it does not need to compute for all kernels, the proposed algorithm requires less computional expense. The construction of $\mathbf{D}$ needs dn^2 operations for d-vectors. After $O(n^2)$ for findinig minimum and maximum values for all n columns is calculated, the last computation is computing kernel values for p kernels with $2 \times n$ values selected from $\mathbf{D}$ in the order of $2pn$. The whole computation for my method then requires $O(dn^2 + n^2 + pn)$ before the optimization process. More specifically, it is defined that a reduced distance matrix $\mathbf{D}' \in \mathbb{R}^{2 \times n}$ which contains the elements of $\mathbf{D}$ in the location specified by $\mathbf{M}$, then a reduced kernel matrix is considered:

$$\mathbf{K}'^{(m)} = \exp\left(-\frac{\mathbf{D}'}{\sigma_m^2}\right) \tag{3.5}$$

where the $\exp(\cdot)$ is Hadamard (element-wise) exponential. Thus it requires only $2ns$ operations in computing $\mathbf{K}'^{(m)}, m = 1 \cdots, s$.

This approach reduces a significant amount of computational burdens of MKL methods. To build a kernel matrix, it usually requires calculations in the order of n^2 along with d-vector operations during the training process. The computations increase even more up to pdn^2 for multiple kernel learning with p kernels. However, because it does not need to compute for all kernels, my method requires less computional expense. The

construction of $\mathbf{D}$ needs dn^2 operations for d-vectors. After $O(n^2)$ for findinig minimum and maximum values for all n columns is calculated, the last computation is computing kernel values for p kernels with $2 \times n$ values selected from $\mathbf{D}$ in the order of $2pn$. The whole computation for my method then requires $O(dn^2 + n^2 + pn)$ before the optimization process.

Let the final reduced kernel be:

$$\mathbf{K}' = \sum_{m=1}^{s} \mu_m \mathbf{K}'^{(m)} \in \mathbb{R}^{2 \times n}. \tag{3.6}$$

Then, my primal problem becomes:

$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & -b \\ \text{subject to} \quad & k'_{1i} - k'_{2i} \geq b, \forall i \in \{1, \cdots, n\}, \\ & \mathbf{1}^T \mu = 1, \mu \geq 0. \end{aligned} \tag{3.7}$$

Here, k'_{ij} is the (i, j) element of $\mathbf{K}'$. Defining the overall gap and the kernel-specific gap for i-th sample as $g_i \triangleq k'_{1i} - k'_{2i}$ and $g_i^{(m)} \triangleq k_{1i}^{\prime(m)} - k_{2i}^{\prime(m)}$ respectively, it becomes $g_i = \sum_{i=1}^{m} \mu_m g_i^{(m)}$. Using this notation, the first constraint can be written in a clear LP form with respect to μ as $\mu^T \mathbf{g}_i \geq b$, where $\mathbf{g}_i = [g_i^{(1)}, \cdots, g_i^{(s)}]^T$. Note that in (3.7), the indices j and l in (3.1) have already been incorporated in $\mathbf{K}'$.

As described earlier, if (3.1) is used, there will be $n_p n_n n$ constraints. However, if the combination of RBF kernels are used, where distance ordering information does not depend on parameter values, $\mathbf{D}$ matrix allows the pre-understanding of indices of where maximum and minimum are present in each column. This allows the reduction of constraints required to only 1 for each instance. Therefore, the optimization (3.7) requires n constraints rather than $n_n n_p n$.

The optimization problem (3.1) and (3.7) can be thought of as approximations for kernel target alignment in that it tries to maximize the gap between worst intra-class similarity and the best inter-class similarity. However, as described in Section 2, it is difficult to meet this objective of placing all the samples in a class close together while keeping considerable distance between different classes.

Therefore, my method relax the constraint by not taking all the samples but a small portion of samples into account. For each column of $\mathbf{D}$, instead of finding the indices for minimum intra-class similarity and maximum inter-class similarity, I take indices of αn_w-th most similar intra-class sample and βn_b-th least similar inter-class sample. Here, n_w and n_b are the number of samples that belongs to the same or different classes with the sample in question respectively, and $0 \leq \alpha, \beta \leq 1$. Finally, I construct $\mathbf{K}'^{(m)}$ based on the parameters α and β, and solve the LP in (3.7) for MKL. The overall procedure of MKL-GM for the same type of kernels is described in **Algorithm 1**.

In MKL-GM, the parameters α and β control the number of *don't care* elements in the kernel matrix. The two extremes are $(\alpha, \beta) = (1, 1)$ and $(\alpha, \beta) = (\varepsilon_1, \varepsilon_2)$ which correspond to the original target alignment problem that does not have any *don't care* elements and almost all the elements are not considered in the optimization, respectively. Here, ε_1 and ε_2 are the smallest possible numbers. Although the optimal choice of these parameters is problem dependent, I empirically have observed that MKL-GM is not highly sensitive to these parameters in Section 4.

Although I derived the proposed method using RBF-type kernels, the technique can be used for MKL consisting of any combination of the same type of kernels with different kernel parameters. For example, for polynomial kernels $k_{ij} = (\gamma(\mathbf{x}_i^T \mathbf{x}_j + 1))^d$ and

Algorithm 1 MKL-GM for the same type of kernels.

Input : Training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, α, β, set of kernel parameters $\Theta = [\theta_1, \cdots, \theta_s]$.

Output : Weights of each kernel μ.

1: Construct base similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ (e.g., $\mathbf{S} = -\mathbf{D}$ which is computed as in (3.4)).

2: **for** $i = 1, \cdots, n$ **do**

3: $\quad m_{1i} \leftarrow \arg(\alpha n_w)^{th} \max_{\{j|y_j=y_i\}} s_{ji}$

4: $\quad m_{2i} \leftarrow \arg(\beta n_b)^{th} \min_{\{l|y_l \neq y_i\}} s_{li}$

5: **end for**

6: **for** $m = 1, \cdots, s$ **do**

7: $\quad$ Construct reduced kernel matrix $\mathbf{K}'^{(m)}$ using $\mathbf{M}$ and θ_m (e.g., as in (3.5)).

8: **end for**

9: Find $\mu = [\mu_1, \cdots, \mu_s]$ by solving (3.7).

hyperbolic tangent kernels $k_{ij} = \tanh(\gamma(\mathbf{x}_i^T \mathbf{x}_j + 1))$, I can firstly compute a base similarity matrix $\mathbf{S}$ consisting of elements $s_{ij} = \mathbf{x}_i^T \mathbf{x}_j + 1$. Then, this matrix can be used to construct the index matrix $\mathbf{M}$ and the reduced kernel matrices $\mathbf{K}'^{(m)}$ in **Algorithm 1**.

3.3 MKL-GM for different types of kernels

Until now, MKL-GM is proposed for the MKL problems composed of the same type of kernels. When different types of kernels are involved, it is difficult to pinpoint the $(\alpha n_w)^{th}$ maximum intra-class similarity and the $(\beta n_b)^{th}$ minimum inter-class similarity elements. This is because when different types of kernels are used, these indices change with different mixing weight μ. For example, when RBF and hyper-tangent kernels are

Algorithm 2 MKL-GM for different types of kernels.

Input : Training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, α, β, r different kernel types.

Output : Weights of each kernel μ.

Construct $\mathbf{S}^{(t)} \in \mathbb{R}^{n \times n}$, $t = 1, \cdots, r$.

for $i = 1, \cdots, n$ **do**

  for $t = 1, \cdots, r$ **do**

    $m_{2t-1,i} \leftarrow \arg(\alpha n_w)^{th} \max_{\{j|y_j=y_i\}} s_{ji}^{(t)}$

    $m_{2t,i} \leftarrow \arg(\beta n_b)^{th} \min_{\{l|y_l \neq y_i\}} s_{li}^{(t)}$

  end for

end for

for $m = 1, \cdots, s$ **do**

  Construct reduced kernel matrix $\mathbf{K}'^{(m)}$ using $\mathbf{M}$.

end for

Find $\mu = [\mu_1, \cdots, \mu_s]$ by solving (3.8).

mixed, because the base similarity matrices in the input space are different for these two kernel types, the final similarity measure in the RKHS is affected by both base measures and the index matrix $\mathbf{M}$ cannot be determined.

However, although different base similarity measures are complementary, they are somewhat related. Consider, for example, the distance based similarity measure (e.g., negative squared distance) and correlation based similarity measure (e.g., inner product). Although they are not perfectly matched, the high value of inner product between two samples generally give rise to a high value of nsd as described in (3.2). Therefore, I can expect that the order of similarity in RKHS does not change dramatically as different μ

is used. Also, the proposed MKL-GM is not much sensitive to the parameters α and β as supported by the forthcoming experimental results in Section 4.

Hence, for each type of base similarity measure, I compute two indices corresponding to the samples that have $(\alpha n_w)^{th}$ maximum intra-class and the $(\beta n_b)^{th}$ minimum inter-class similarity with $\mathbf{x}_i$ and store these in $\mathbf{M}$. Now, if I have $r(\leq s)$ different types of kernels, the size of $\mathbf{M}$ becomes $2r \times n$. Then, as before, $\mathbf{K}'^{(m)} \in \mathbb{R}^{2r \times n}$ is computed using $\mathbf{M}$ for $m = 1, \cdots, s$. For each sample i the gap matrix $\mathbf{G}_i \in \mathbb{R}^{s \times r}$ which consists of elements $g_{i,mt} = k'^{(m)}_{2t-1,i} - k'^{(m)}_{2t,i}$ for $m = 1, \cdots, s$ and $t = 1, \cdots, r$ is computed for the following optimization problem.

$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & -b \\ \text{subject to} \quad & \mu^T \mathbf{G}_i \geq b, \forall i \in \{1, \cdots, n\} \\ & \mathbf{1}^T \mu = 1, \mu \geq 0, \end{aligned} \tag{3.8}$$

Note that the number of constraint except the trivial ones $\mathbf{1}^T\mu = 1, \mu \geq 0$ are increased to rn. If all the base kernels are in different types, then this number becomes sn. The detailed MKL-GM algorithm for different types of kernels is described in **Algorithm 2**. Note that the base similarity matrix $\mathbf{S}^{(t)}$ can be replaced with base kernel $\mathbf{K}^{(m)}$.

The formulation (3.8) looks close to the formulation of LP-β algorithm of [27] but it has mainly two differences. Firstly, while LP-β consists of two step optimization 1) firstly find weak learner $f_m(x)$ by finding optimal SVM parameters for each base kernel, 2) then kernel mixing weight μ is optimized using a similar form of (3.8), MKL-GM does not have the SVM optimization step. Secondly, unlike LP-β, MKL-GM does not maximize the minimum gap but relax this constraint to maximize an appropriate gap of

intra-class and inter-class samples in RKHS.

Chapter 4

Experiments

In this part, MKL-GM is applied to various problems from simple toy example (two-spiral) to a image recognition problem (Caltech101 dataset). For all the experiments, CVX toolbox [28] is used as an optimization solver. Since support vector machine (SVM) algorithm [2] is widely used, the experiments are also proceeded with SVM as the classifier. For all the experiments, the base kernel value itself was used as a base similarity measure.

4.1 Toy example: Two-spiral data

A two-spiral toy exmample data of 100 samples is constructed as shown in Fig. 4.1. Two types of kernels are considered : 1. Gaussian kernel of the form (3.3), 2. hyper-tangent kernel of the form $k_{ij} = \tanh(\gamma(\mathbf{x}_i^T \mathbf{x}_j + 1))$. For both types of kernels, ten base kernels were constructed with different kernel parameters σ^2 and γ varying in the log

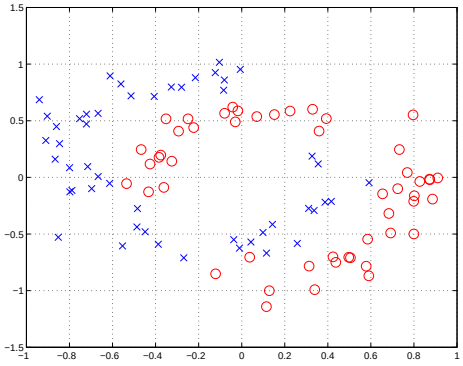

Figure 4.1: Original two-spiral data in the 2-D input space.

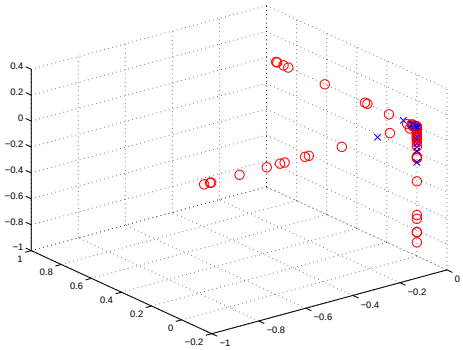

Figure 4.2: MKL-GM: Gaussian kernels. The data points from Fig. 4.1 projected to the RKHS by MKL-GM. The figure is obtained by KPCA.

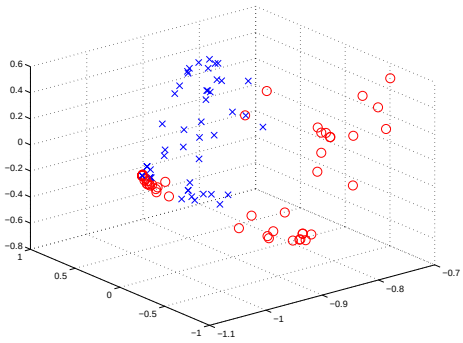

Figure 4.3: MKL-GM: tanh kernels. The data points from Fig. 4.1 projected to the RKHS by MKL-GM. The figure is obtained by KPCA.

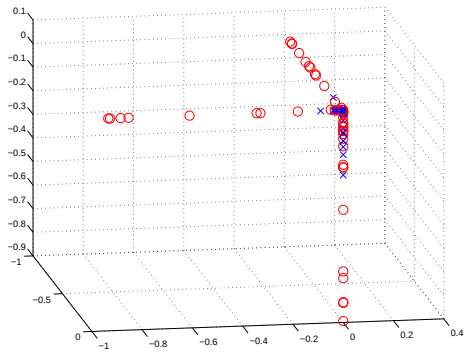

Figure 4.4: MKL-GM: Gaussian + tanh. The data points from Fig. 4.1 projected to the RKHS by MKL-GM. The figure is obtained by KPCA.

space such that $\sigma^2, \gamma \in [10^{-3}, 10^3]$. Fig. 4.2 - Fig. 4.4 show the typical data distribution in RKHS, which were obtained by applying KPCA on the combined kernel from MKL-GM and plotting three principal components for each of 100 samples.

To see the sensitivity of MKL-GM with respect to α and β, these parameters are varied such that $\alpha \in \{0.04, 0.1, 0.2, \cdots, 0.9, 1.0\}$ and $\beta \in \{0.1, 0.2, \cdots, 1.0\}$. Setting $\alpha = 0.04$ corresponds to setting j as the index of the nearest sample because $n_w = 50$ in this case. I randomly tested MKL-GM with 20 base kernels (10 Gaussian + 10 tanh) 100 times and the average classification accuracies for a separate test data using SVM are plotted in Fig. 4.5. In the figure, there is a large flat area of (α, β) where the classification accuracy has uniformly high values. From this, it can be seen that MKL-GM is not much sensitive to the selection of (α, β). The best average performance of 99.52% was obtained at (α, β) = (0.8, 0.2). I also tested MKL-GM using a single type of kernel (Gaussian or tanh) and the tendency is almost the same as Fig. 4.5. The best performance for Gaussian mixture and tanh mixture were 99.52% and 90.60%, respectively. The performance using original input data was 72.47%.

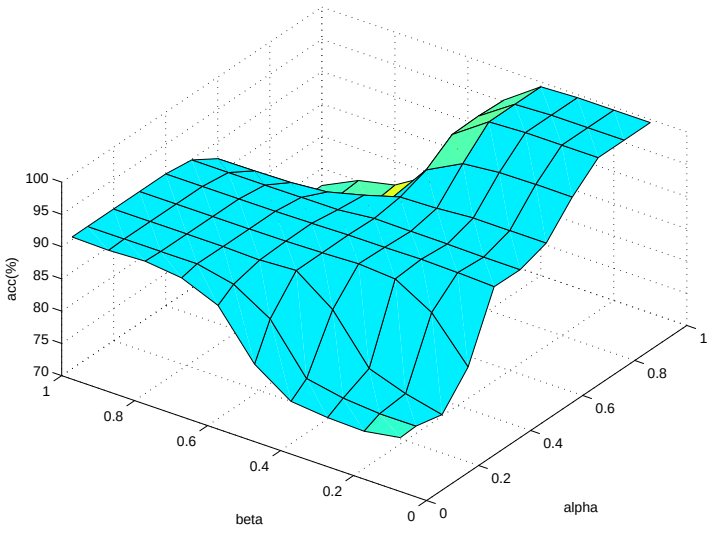

Figure 4.5: Average accuracy of MKL-GM with different values of (α, β)

To see which kinds of kernels are generated by MKL-GM, I set α and β equal and varied the values from 0.1 to 1.0 and showed the resulting kernels in Fig. 4.6. The samples are ordered by their class and the perfect kernel should look like a block diagonal structure as shown in the left side in Fig. 3.1. For this dataset, when α and β are too small (≤ 0.2) or too big (≥ 0.9) the resulting kernel matrix becomes too dense and the performance was not good (under 80%). On the other hand, when α and β are in the middle range from 0.3 to 0.8, as can be seen in the figure, the resulting kernel matrix becomes sparse and the classification performance becomes better (over 90%). Considering that the kernel alignment corresponds to $\alpha = \beta = 1$, this result support my motivation that some amount of '*don't care*' elements help constructing a better combined kernel.

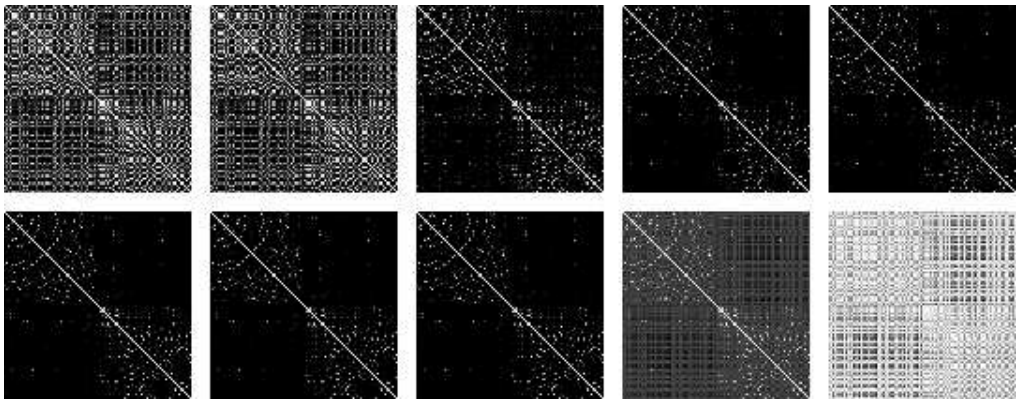

Figure 4.6: Final kernels with different values of (α, β). 1^{st} row: $\alpha = \beta = 0.1 \cdots, 0.5$. 2^{nd} row: $\alpha = \beta = 0.6 \cdots, 1.0$. (white = 1, black = 0)

Figure 4.7: Examples of face and non-face images.

4.2 FERET face database

I evaluated the proposed method on Face vs. Non-face data from the Color FERET face database [29], which consists 8,000 face and non-face instances in binary classes. The face images were cropped based on the centers of the right and left pupils and they were resized to 24×24 pixels. The non-face images were randomly cropped and resized to have the same size. Fig. 4.7 shows some of the examples. The 400 test data instances are selected from the whole dataset, and the rest was used as training data. I have experimented kernel target alignment method of Lankriet's alignment maximization [4] and MKL-GM (the proposed method) to compare the performance of each.

Table 4.1: Experiment result on Face vs. Non-face Dataset

	Alignment	MKL-GM
Recognition Rate	97.25%	97.92%
Activated Sigma Value(s)	$10^{2.4}, 10^{2.5}$	$10^{3.1}$
Non-zero μ Value(s)	0.1748, 0.8252	1.0000

As base kernels, I used Gaussian kernels with different σ^2 values that increments in log space by $10^{0.1}$ from 10^{-3} to 10^{6}. Therefore, total 91 base kernels were used for both algorithms. For this experiment, the setting was $(\alpha, \beta) = (0.1, 0.1)$. MKL-GM and kernel alignment resulted in different combination of σ^2 values. As the result, kernel target alignment is optimized to have combination of two dominating σ^2 values, $10^{2.4}$ and $10^{2.5}$ with weighting coefficients, $\mu_{10^{2.4}} = 0.1748$ and $\mu_{10^{2.5}} = 0.8252$, respectively. Our method is optimized to have one dominant σ^2 value activated, $10^{3.1}$ with a weighting coefficient $\mu_{10^{3.1}} = 1$. This is summarized in Table 4.3.

The difference in recognition performance is not large, but my method is obtained by far lower amount of computations than kernel target alignment is. While my method only computes the ones that are αn_w-th and βn_b-th indices for minimum and maximum values from the matrix $\mathbf{D}$, target alignment method takes all the datasets into accounts in order to achieve its optimized solution. In addition, my method is much faster because my formulation is LP while that of kernel alignment is QCQP which requires much more computation than LP.

4.3 Protein Fold Prediction

Taking Gonen [1] into account, I performed experiments on the Protein Fold (PROTEIN) prediction data set from the MKL Repository in order to compare my method against other MKL algorithms. The dataset is composed by 12 different feature representations and two kernels (linear[1] and Gaussian) for 694 instances (311 training and 383 for testing). For the fair comparison, I tried to strictly follow the methods of experiments that were performed in the paper [1]. We reconstructed the dataset into a binary classification problem by combining the major structural classes α, β[2] into one class and $\alpha/\beta, \alpha+\beta$ into another class. Our experiments were proceeded with 30-fold cross-validation for each of three different subsets of kernels: Subset (COM-SEC-HYD-VOL-POL-PLZ) denoted by (COM-PLZ), subset (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30) denoted by (COM-L30) and subset (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30-BLO-PAM) denoted by (COM-PAM) are used. Therefore, for each problem, I have 6, 10, and 12 base kernels, respectively.

By firstly setting $\beta = 0.5$, I heuristically experimented with five different intra-class samples that are distant in the range from 1% to 5% far ($\alpha = 0.01, 0.02, \cdots, 0.05$). Then, I took the one with the best result.

Table 4.2 lists the performance results of various algorithms on (COM-PLZ), (COM-L30) and (COM-PAM). The results of all the algorithms were performed with linear base kernels. The last two rows represent the performance of the MKL-GM with linear base kernels and Gaussian base kernels respectively. Labels for each algorithm is described in

[1] $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j$.

[2] Here, α and β are different from the ones used throughout the paper.

Gonen (2011) [1] and the performance of other algorithms are also from the same paper.

Among the compared methods, all the methods except RBMKL (rule-based MKL) and NLMKL (nonlinear MKL) [11] result in a linear combination of base kernels, thus these are referred to as linear MKL methods. On the other hand, NLMKL finds an optimal product of base kernels as an output kernel, and RBMKL(mean) and RBMKL(product) output mean of base kernels and product of base kernels, thus these can be regarded as nonlinear MKL algorithms.

In order to compare against algorithms with linear MKL algorithms such as ABMKL and CABMKL, I first calculated linear kernels for each representation, and then optimized the weighting coefficients by maximizing the gap between the $(\alpha n_w)^{th}$ maximum intra-class kernel value and the $(\beta n_b)^{th}$ minimum inter-class kernel value for each column of the kernel matrix. For the second kernel function, the proposed algorithm was proceeded with the Gaussian kernel function in two steps. First, each representation is optimized to learn the optimal σ^2 values from the combination of Gaussian kernels. We looked for the optimal σ^2 value from 91 σ^2 values that increments by $10^{0.1}$ from 10^{-3} to 10^6. The weighting coefficients for kernels of each representation with previously learned parameters are then optimized for the data instances to be segregated apart according to their class labels. Therefore, one can easily notice which representation is critical in order for the data to be well-discriminated.

The test accuracy of MKL-GM(linear) outperforms other linear MKL algorithms for the (COM-PLZ) set and it shows a similar performance to the NLMKL which is a nonlinear method. Although this dataset is the most difficult classification problem from [1], MKL-GM(gaussian) performs far better than other methods including the nonlinear one

Table 4.2: Performances of single-kernel SVM, representative MKL algorithms with linear kernel and MKL-GM on three different subsets of kernels of PROTEIN data set [1].

	Test Accuracy		
Algorithm	(COM - PLZ)	(COM - L30)	(COM - PAM)
SVM(best)	72.06±0.74	72.15±0.68	78.37±1.08
SVM(all)	79.13±0.45	79.63±0.74	82.01±0.76
RBMKL(mean)	78.01±0.63	81.35±0.74	83.57±0.59
RBMKL(product)	72.35±0.95	53.04±0.21	53.04±0.21
ABMKL(conic)	79.03±0.92	80.45±0.68	83.52±0.94
ABMKL(convex)	76.90±1.17	77.47±0.62	83.76±1.02
ABMKL(ratio)	78.06±0.62	76.22±1.14	85.65±0.67
CABMKL(linear)	79.51±0.78	77.15±0.63	83.48±0.92
CABMKL(conic)	79.28±0.97	81.02±0.67	83.43±0.95
MKL	76.38±1.19	79.74±1.02	83.55±1.25
simpleMKL	74.96±0.50	74.53±0.90	83.96±1.20
GMKL	74.96±0.50	74.68±0.68	85.67±0.91
GLMKL($p = 1$)	77.71±0.96	79.77±0.86	85.96±0.96
GLMKL($p = 2$)	77.20±0.42	78.00±0.43	85.02±1.20
NLMKL($p = 1$)	83.49±0.76	85.38±0.70	87.00±0.66
NLMKL($p = 2$)	82.30±0.62	85.40±0.69	87.28±0.65
LMKL(softmax)	80.24±1.37	81.11±1.82	83.72±1.35
LMKL(sigmoid)	81.91±0.92	81.90±2.01	85.06±0.83
MKL-GM(linear)	83.03±0.79	83.03±0.54	83.60±0.86
MKL-GM(gaussian)	**88.74±0.55**	**88.61±0.39**	**88.46±0.44**

(NLMKL) by more than 5%. For other problems of (COM-L30) and (COM-PAM), although the performance of MKL-GM(linear) does not improve much with more base kernels (10 and 12, respectively), it still ranks top among the linear MKL algorithms for (COM-L30) problem and shows compatible performance with other linear MKL algorithms for (COM-PAM) problem. For all the problems, MKL-GM(gaussian) performs the best.

4.4 Pendigits Digit Recognition

Pendigits (PENDIGITS) digit recognition data set from the MKL Repository consists of 4 different feature representations DYN, STA4, STA8 and STA16. The detailed information on the features can be found in [1]. This dataset is considered large because each representation is composed with 10,992 instances, and the data is trained and tested through 10-fold cross validation. From the dataset, two different binary classification problems are generated as in [1]. PENDIGITS-EO is composed of data that are even digits and odd digits, and PENDIGITS-SL is composed of data that are small digits ('0'-'4') and large digits ('5'-'9'). All other experimental methodologies for this data is the same as the ones for PROTEIN data.

Name	Dim.	Data Source
DYN	16	8 successive pen points on $\mathbb{R}^2$ coordinate system
STA4	16	4×4 image bitmap representation
STA8	64	8×8 image bitmap representation
STA16	256	16×16 image bitmap representation

Table 4.3: Multiple feature representations in the PENDIGITS data set [1].

Table 4.4: Performances of single-kernel SVM, representative MKL algorithms with linear kernel and MKL-GM on three different subsets of kernels of PENDIGITS data set [1].

Algorithm	Test Accuracy for (PENDIGITS-EO)	Test Accuracy for (PENDIGITS-SL)
SVM(best)	88.93±0.28	84.44±0.49
SVM(all)	92.12±0.42	89.48±0.67
RBMKL(mean)	93.34±0.28	91.11±0.34
RBMKL(product)	98.46±0.16	98.37±0.11
ABMKL(conic)	93.40±0.15	90.97±0.49
ABMKL(convex)	93.53±0.26	90.85±0.51
ABMKL(ratio)	93.35±0.20	91.12±0.32
CABMKL(linear)	93.42±0.16	91.02±0.47
CABMKL(conic)	93.42±0.16	91.02±0.47
MKL	93.28±0.29	90.85±0.45
simpleMKL	93.29±0.27	90.84±0.50
GMKL	93.28±0.26	90.85±0.47
GMKL($p = 1$)	93.34±0.27	90.90±0.46
GMKL($p = 2$)	93.32±0.25	91.12±0.44
NLMKL($p = 1$)	99.36±0.08	99.11±0.10
NLMKL($p = 2$)	99.38±0.07	99.07±0.12
LMKL(softmax)	97.14±0.39	97.77±0.54
LMKL(sigmoid)	97.80±0.20	97.13±0.40
MKL-GM(linear)	93.05±0.73	96.83±0.03
MKL-GM(gaussian)	**99.46±0.21**	**99.73±0.14**

Table 4.4 gives the classification accuracy of all algorithms on both PENDIGITS-EO and PENDIGITS-SL datasets. For PENDIGITS-EO, while MKLGM(linear) performs no greater than other algorithms, MKLGM(gaussian) achieves average test accuracy rate of 99.46% which is the best of all. The other algorithms that show similar good results are the nonlinear ones NLMKL(p = 1) and NLMKL(p = 2). For PENDIGITS-SL, MKLGM(linear) outperforms other linear MKL algorithms. MKLGM(gaussian) not only reaches top 3 test accuracy, but also achieves the best accuracy of all, higher by more than 0.6% than the second best on average.

4.5 Clatech-101 dataset

In this part, I use the Caltech-101 dataset [30] which is a benchmark dataset for object classification. It contains 9,144 images from 102 classes (i.e. 101 object classes and a 'background'). We follow the experimental setup proposed by the designers of the datasets. Performance is measured as the mean prediction rate per class, thus balancing the influence of categories with a large number of test examples. I report results using all 102 classes of the Caltech-101 dataset averaged over three splits. In the experiment, I downloaded pre-computed base kernel matrices with 15 training examples per class from the webpage of [31] and applied MKL-GM directly to the base kernels. There are total 10 base kernels which were obtained from three types of raw features: dense SIFT, self-similarity, and geometric blur features. The details on how the base kernels are constructed can be found in the webpage. Using the 10 base kernels, I obtained 70.8% of accuracy on $(\alpha, \beta) = (0.05, 0.99)$. Table 4.5 shows the comparison of classification accuracy of different methods. In the table, I also showed the number of kernels if multiple

Table 4.5: Comparison of classification accuracy on Caltech-101 dataset using 15 samples per class.

Method	Accuracy (%)	No. of kernels (features)
Zhang [32]	59.1±0.6	N/A
Jiang [33]	67.7	N/A
Shrivastava [34]	69.3	N/A
Vedaldi [31]	71.1±0.6	10
Nguyen [35]	72.5	39
Gehler (LP-β) [27]	70.4±0.8	39
Gehler2 (LP-β) [27]	74.6±1.0	48
Thiagarajan [25]	75.7	48
MKL-GM	70.8±0.7	10

kernels are used. Considering different number of kernels is used for each method, my method is not directly comparable with other methods. However, MKL-GM which uses 10 features shows similar performance to LP-β with 39 features. Considering that the same algorithm (Gehler2 (LP-β)) with 9 more features of Vedaldi (whose kernels are used in my method), performed 74.6% accuracy, I can expect that my method with these 48 features will hopefully reach the top accuracy over 75%. Our method can be directly compared to Vedaldi [31] where exactly the same base kernels are used with MKL algorithm of Varma et. al. [21]. Although MKL-GM is 0.3% inferior, the performance of the two are almost the same. This strongly suggests that MKL-GM is a competitive method of MKL.

Chapter 5

Conclusion and Future Work

In this paper, I propose a new kernel target learning method called, MKL-GM, that try to maximize the gap between intra and inter-class samples in RKHS. Motivated by kernel target alignment, the proposed method was originally developed for MKL consisting of the same type of kernels, then the method was extended to deal with different types of kernels. Especially, when only a single type of kernels are used, MKL-GM has an effect of selecting appropriate kernel parameters. Since computing the right kernel parameters has been usually achieved by heuristic approaches, my algorithm would make experiments more practical in terms of both performance and efficiency. For MKL consisting of different types of kernels, my method shows competitive performance to the other MKL methods. The proposed method has a merit that it does not solve a complicated convex optimization problems such as QCQP, SOCP, or SIP but solves a simple LP with reduced set of constraints.

Further study of this method would primarily be selecting appropriate parameters and kernel functions. When figuring the right α and β, or the indices selected from $\mathbf{D}$ to

be fed to kernel functions, the process include heuristic aspects, although the domain of search is quiet predictable. Thus, there needs additional investigations on finding the right αs and βs in order to polish the quality of the proposed algorithm because the parameters significantly influence the classification performance after all. Secondly, because the proposed method does not directly use target kernel nor involves in SVM optimization, I hope to find a way of extending MKL-GM for unsupervised or semi-supervised kernel optimization.

Bibliography

[1] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Machine Learning Research, JMLR*, vol. 12, pp. 2211–2268, 2011.

[2] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[3] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1354–1369, 2014.

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

[5] S. Sonnenburg, G. Rätsch, and C. Schäfer, "A general and efficient multiple kernel learning algorithm," in *Advances in Neural Information Processing Systems, NIPS*, vol. 18, 2005, pp. 1275–1282.

[6] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm multiple kernel learning," *The Journal of Machine Learning Research*, vol. 12, pp. 953–997, 2011.

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

[8] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 795–828, 2012.

[9] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 6, no. 2, pp. 190–199, 2009.

[10] J. He, S.-F. Chang, and L. Xie, "Fast kernel learning for spatial pyramid matching," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–7.

[11] C. Cortes, M. Mohri, and A. Rostamizadeh, "Two-stage learning kernel algorithms," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 239–246.

[12] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 6.

[13] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 775–782.

[14] R. Alain, R. Francis, C. Phane, and S. Yves, "Simple mkl," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[15] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1065–1072.

[16] S. N. Jagarlapudi, G. Dinesh, S. Raman, C. Bhattacharyya, A. Ben-Tal, and R. Kr, "On the algorithmics and applications of a mixed-norm based kernel learning formulation," in *Advances in neural information processing systems*, 2009, pp. 844–852.

[17] M. Kloft, U. Brefeld, P. Laskov, K.-R. Müller, A. Zien, and S. Sonnenburg, "Efficient and accurate lp-norm multiple kernel learning," in *Advances in neural information processing systems*, 2009, pp. 997–1005.

[18] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 1175–1182.

[19] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Advances in neural information processing systems*, 2009, pp. 396–404.

[20] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 352–359.

[21] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[22] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *The Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[23] Z. Xu, R. Jin, I. King, and M. Lyu, "An extended level method for efficient multiple kernel learning," in *Advances in neural information processing systems*, 2009, pp. 1825–1832.

[24] Y. Y. Lin, T. L. Liu, and C. S. Fuh, "Multiple kernel learning for dimensionality reduction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 6, pp. 1147–1160, 2011.

[25] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multiple kernel sparse representations for supervised and unsupervised learning," *IEEE Trans. on Image Processing*, vol. 23, no. 7, pp. 2905 – 2915, July 2014.

[26] B. Ni, V. R. Paramathayalan, and P. Moulin, "Multiple granularity analysis for fine-grained action detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[27] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 221–228.

[28] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, 2008, pp. 95–110.

[29] P. J. Phillips, H. Moon, S. Rizvi, P. J. Rauss *et al.*, "The feret evaluation methodology for face-recognition algorithms," *IEEE Tran. Pattern Analysis and Machine Intelligence, TPAMI*, vol. 22, no. 10, pp. 1090–1104, 2000.

[30] "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *CVPRW*, 2004, p. 178.

[31] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 606–613.

[32] H. Z. A. C. B. Michael and M. J. Malik, "Svmknn: Discriminative nearest neighbor classification for visual category recognition," *Computer Science Division, EECS Department Univ. of California, Berkeley, CA*, vol. 94720, 2007.

[33] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1697–1704.

[34] A. Shrivastava, V. M. Patel, and R. Chellappa, "Multiple kernel learning for sparse representation-based classification," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 23, no. 7, pp. 3013–3024, July 2014.

[35] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *Image Processing, IEEE Transactions on*, vol. 22, no. 12, pp. 5123–5135, 2013.