JMLR: Workshop and Conference Proceedings 20 (2011) 129–144 Asian Conference on Machine Learning

Unsupervised Multiple Kernel Learning

Jinfeng Zhuang ZHUA0016@NTU.EDU.SG
Jialei Wang JL.WANG@NTU.EDU.SG
Steven C. H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering, Nanyang Technological University, Singapore
Xiangyang Lan XYLAN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, USA

Editor: Chun-Nan Hsu and Wee Sun Lee

Abstract

Traditional multiple kernel learning (MKL) algorithms are essentially supervised learning in the sense that the kernel learning task requires the class labels of training data. However, class labels may not always be available prior to the kernel learning task in some real world scenarios, e.g., an early preprocessing step of a classification task or an unsupervised learning task such as dimension reduction. In this paper, we investigate a problem of Unsupervised Multiple Kernel Learning (UMKL), which does not require class labels of training data as needed in a conventional multiple kernel learning task. Since a kernel essentially defines pairwise similarity between any two examples, our unsupervised kernel learning method mainly follows two intuitive principles: (1) a good kernel should allow every example to be well reconstructed from its localized bases weighted by the kernel values; (2) a good kernel should induce kernel values that are coincided with the local geometry of the data. We formulate the unsupervised multiple kernel learning problem as an optimization task and propose an efficient alternating optimization algorithm to solve it. Empirical results on both classification and dimension reductions tasks validate the efficacy of the proposed UMKL algorithm.

Keywords: Kernel Methods, Multiple Kernel Learning, Unsupervised Learning, Dimension Reduction

1. Introduction

Kernel learning is an active research topic in machine learning. The family of kernel based machine learning algorithms has been extensively studied over the past decade (Shawe-Taylor and Cristianini, 2004). The well-known examples include Support Vector Machines (SVM) (Vapnik, 1998), Kernel Logistic Regression (Zhu and Hastie, 2001), and Kernel PCA for denoising (Mika et al., 1998), etc. These kernel methods have been successfully applied to a variety of real applications and often achieved promising performance.

The crux of kernel methods is *kernel*, which is in general a function that defines an inner product of any two samples in some induced Hilbert space (Shawe-Taylor and Cristianini, 2004; Hofmann et al., 2008). By mapping data from an input space to some Reproducing Kernel Hilbert space (RKHS) which can be potentially high-dimensional, traditional linear methods can be extended with reasonable effort to yield considerably better performance. Many empirical studies have shown that the choice of kernel often affects the resulting performance of a kernel method significantly. In fact,

inappropriate kernels usually result in sub-optimal or even poor performance when applying kernel methods to solve a real-world problem.

For many real-world situations, it is often not easy to choose an appropriate kernel, which usually requires some domain knowledge that may be difficult for non-expert users. To address such limitations, recent years have witnessed the active research on learning effective kernels automatically from data (Lanckriet et al., 2004; Sonnenburg et al., 2006; Hoi et al., 2007; Zhuang et al., 2011b). One popular technique for kernel learning is *Multiple Kernel Learning* (MKL) (Lanckriet et al., 2004; Sonnenburg et al., 2006), which aims at learning a linear (or convex) combination of a set of predefined kernels in order to identify a good target kernel for the applications. Comparing with traditional kernel methods using a single fixed kernel, MKL does exhibit its strength of automated kernel parameter tuning and capability of concatenating heterogeneous data. Over the past few years, MKL has been actively studied, in which a variety of algorithms have been proposed to resolve the efficiency of MKL (Sonnenburg et al., 2006; Xu et al., 2008), and a lot of extended MKL techniques have been proposed to improve the regular MKL method (Gehler and Nowozin, 2008; Varma and Babu, 2009; Corinna Cortes and Rostamizadeh, 2009b,a; Kloft et al., 2009; Jin et al., 2010).

Despite being studied extensively, existing MKL methods are essentially supervised learning which requires class labels of training data available for the kernel learning task. However, the class labels of training data may not always be available for some learning tasks. Examples include unsupervised learning tasks such as dimension reduction, clustering, or some early preprocessing step of a supervised classification task for choosing a kernel before obtaining the labeled data. By the above motivations, in this paper, we address a challenging problem of *Unsupervised Multiple Kernel Learning* (UMKL), which aims to determine a linear combination of multiple kernels by learning only from unlabeled data.

The UMKL task is more challenging than traditional MKL tasks since no class labels are available to the learning task. In this paper, we propose to attack the unsupervised multiple kernel learning problem by exploiting two intuitions: (1) a good kernel should allow every example to be well reconstructed from its localized bases weighted by the kernel values; (2) a good kernel should induce kernel values that are coincided with the local geometry of the data. We formulate the problem as an optimization task by combining the above two intuitions and propose an iterative algorithm to efficiently solve the optimization task.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries of MKL. Section 3 presents the problem formulation and the proposed UMKL algorithm. Section 4 gives our empirical evaluations on the proposed UMKL algorithm. Section 5 concludes this work.

2. Background Review

In this section, we introduce some preliminaries of traditional multiple kernel learning (MKL) in a supervised learning setting, followed by some discussion on related work.

2.1. Supervised Multiple Kernel Learning

We first introduce some common notations. We use bold upper case letters to denote matrices, bold lower case letters to denote vectors, and curlycue upper case letter to denote sets. The notation $[\cdot]$ with subscripts denotes the corresponding entries of a vector or matrix.

In a typical supervised multiple kernel learning task, we are given a collection of n training samples $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector and y_i is the class label of $\mathbf{x}_i$. The conventional multiple kernel learning (MKL) can be formulated into the following optimization task:

$$\min_{k\in\mathcal{K}} \min_{f\in\mathcal{H}_k} \quad \lambda\|f\|_{\mathcal{H}_k} + \sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)), \tag{1}$$

where $\ell(\cdot)$ denotes some loss function, e.g. the hinge loss $\ell(t) = \max(0, 1-t)$ used for SVM, $\mathcal{H}_k$ is the reproducing kernel Hilbert space associated with kernel k, $\mathcal{K}$ denotes the optimization domain of the candidate kernels, and λ is a regularization parameter.

The above optimization aims to simultaneously identify both the optimal kernel k from domain $\mathcal{K}$ and the optimal prediction function f from the kernel Hilbert space $\mathcal{H}_k$ induced by the optimal kernel k. By the representer theorem(Schölkopf et al., 2001), the decision function $f(\mathbf{x})$ is in form of a linear expansion of kernel evaluation on training sample $\mathbf{x}_i$'s, i.e.,

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i k(\mathbf{x}_i, \mathbf{x}),$$

where β_i's are the coefficients. In traditional MKL (Lanckriet et al., 2004), $\mathcal{K}$ is chosen to be a set of any convex combination of m predefined base kernels:

$$\mathcal{K}_{conv} = \left\{ k(\cdot,\cdot) = \sum_{t=1}^{m} \mu_t k_t(\cdot,\cdot) : \sum_{t=1}^{m} \mu_t = 1, \mu_t \geq 0 \right\},$$

where each candidate kernel k is some combination of the m base kernels $\{k_1, \ldots, k_m\}$, μ_t is the coefficient of the tth base kernel and $\mathbb{N}_m = \{1, \ldots, m\}$. Based on the above definition of $\mathcal{K}_{conv}$, the decision function of regular MKL can be written as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \sum_{t=1}^{m} \mu_t k_t(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{n} \sum_{t=1}^{m} \beta_i \mu_t k_t(\mathbf{x}_i, \mathbf{x}) \ . \tag{2}$$

2.2. Related Work

Although MKL in general can be formulated as a convex optimization task, such an optimization task is usually difficult to solve. In literature, researchers have spent extensive efforts in developing efficient solvers for supervised MKL. Some representative examples include (Gehler and Nowozin, 2008; Varma and Babu, 2009; Corinna Cortes and Rostamizadeh, 2009b,a; Kloft et al., 2009; Xu et al., 2010). In addition to the efficiency issue, some recent studies have attempted to overcome some limitations of regular MKL techniques (Cortes, 2009). For example, some studies have addressed the limitation of using linear combination of multiple kernels by exploring more flexible kernel combination methods (Gehler and Nowozin, 2008; Varma and Babu, 2009; Corinna Cortes and Rostamizadeh, 2009b,a; Zhuang et al., 2011a). Most of these methods essentially follow the same large margin learning framework of SVM. Very recently, Cortes et. al. proposed the two-stage kernel target alignment (Cortes et al., 2010), which isolates the kernel learning task from SVM. It exhibits the state-of-the-art empirical performance when comparing with other conventional kernel learning techniques.

In MKL, the objective function follows the regularization framework of SVM, i.e., *loss + regularization*. The norm of f in RKHS as a regularization term penalizes the complexity of SVM effectively as it upper bounds of the Rademacher complexity of the function class induced by a single kernel(Bartlett and Mendelson, 2002). When the kernel class varies during the learning, it always introduces more complexity(Lanckriet et al., 2004; Srebro and Ben-David, 2006; Ying and Campbell, 2009). Unlike the existing supervised multiple kernel learning algorithms, we study unsupervised multiple kernel learning that does not require the class labels of training data available in the learning task. Different from some existing unsupervised kernel learning studies that follow the large margin learning principle (Valizadegan and Jin, 2006; Zhao et al., 2009; M. Ehsan Abbasnejad and Mandava, 2010), our unsupervised multiple kernel learning method is inspired in part by some recent studies on the two-stage kernel learning (Cortes et al., 2010) and manifold-based semi-supervised learning (Yu et al., 2009).

3. Unsupervised Multiple Kernel Learning

In this section, we formally formulate the problem of unsupervised multiple kernel learning and then present an iterative algorithm to solve the optimization task.

3.1. Problem Formulation

Consider a collection of n training samples $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector and y_i is the unknown class label of $\mathbf{x}_i$, and a set of m predefined kernel functions $\{k_t(\cdot,\cdot), t = 1, \ldots, m\}$. The goal of an Unsupervised Multiple Kernel Learning (UMKL) task is to find an optimal linear combination of the m kernel functions, i.e., $k^*(\cdot,\cdot) \in \mathcal{K}_{conv}$, where $\mathcal{K}_{conv}$ is defined as follows:

$$\mathcal{K}_{conv} = \left\{ k(\cdot,\cdot) = \sum_{t=1}^{m} \mu_t k_t(\cdot,\cdot) : \sum_{t=1}^{m} \mu_t = 1, \mu_t \geq 0 \right\} \tag{3}$$

Unlike supervised MKL, the key challenge of UMKL is how to seek the optimal kernel $k^*(\cdot,\cdot)$ purely from the unlabeled training data. In other words, we need some principles/intuitions to guide the kernel learning task, which should be independent of class labels.

To attack the challenge, we propose to formulate the unsupervised multiple kernel learning task by exploiting the following two intuitive principles:

- A good kernel should enable each training instance to be well reconstructed from the localized bases weighted by the kernel values. In other words, for each $\mathbf{x}_i$ we expect the optimal kernel can minimize the approximation error $\|\mathbf{x}_i - \sum_j k_{ij}\mathbf{x}_j\|^2$;
- A good kernel should induce kernel values that are coincided with the local geometry of the training data. This is equivalent to finding the optimal kernel that minimizes the distortion over all training data $\sum_{ij} k_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|^2$, where $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Besides, we also exploit the locality preserving principle, which has been shown effective for many unsupervised dimension reduction and semi-supervised learning tasks. In particular, to infer such a local structure, we introduce a set of local bases for each $\mathbf{x}_i \in \mathcal{X}$, denoted $\mathcal{B}_i \subset \mathcal{X}$, which is

used to reconstruct sample $\mathbf{x}_i$ and compute the distortion as well. Combining the above principles, we formulate the optimization problem for UMKL as follows:

$$\min_{k\in\mathcal{K}_{conv},\mathcal{B}} \frac{1}{2}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \sum_{\mathbf{x}_j\in\mathcal{B}_i} k_{ij}\mathbf{x}_j\right\|^2 + \gamma_1\sum_{i=1}^{n}\sum_{\mathbf{x}_j\in\mathcal{B}_i} k_{ij}\|\mathbf{x}_i-\mathbf{x}_j\|^2 + \gamma_2\sum_i|\mathcal{B}_i| \tag{4}$$

where both the target kernel $k \in \mathcal{K}_{conv}$ and the local basis set $\mathcal{B}_i$ are unknown variables to be optimized, γ_1 controls the trade-off between the coding error and the locality distortion, γ_2 controls the size of local basis set.

To simplify the formulation, for each $\mathbf{x}_i$, we introduce a matrix $\mathbf{D} \in \{0,1\}^{n\times n}$ in which each column vector $\mathbf{d} \in \{0,1\}^n$ indicates its neighbors, i.e., $\mathcal{B}_i = \{\mathbf{x}_j : d_j \neq 0\}$. As a result, the optimization problem can be re-written in a matrix-based form:

$$\begin{aligned}\min_{\boldsymbol{\mu},\mathbf{D}}\ & \frac{1}{2}\left\|\mathbf{X}\big(\mathbf{I}-\mathbf{K}\circ\mathbf{D}\big)\right\|_F^2+\gamma_1\mathrm{tr}\ \mathbf{K}\circ\mathbf{D}\circ\mathbf{M}\big(\mathbf{1}\mathbf{1}^\top\big)+\gamma_2\|\mathbf{D}\|_{1,1} \\ \text{s.t.}\quad & [\mathbf{K}]_{ij}=\sum_{t=1}^{m}\mu_t k^t(\mathbf{x}_i,\mathbf{x}_j), 1\le i,j\le n, \boldsymbol{\mu}^\top\mathbf{1}=1, \boldsymbol{\mu}\ge 0, \mathbf{D}\in\{0,1\}^{n\times n},\end{aligned} \tag{5}$$

where the i-th column of $\mathbf{X}$ is the point $\mathbf{x}_i$, matrix $[\mathbf{M}]_{ij} := \mathbf{x}_i^\top\mathbf{x}_i+\mathbf{x}_j^\top\mathbf{x}_j-2\mathbf{x}_i^\top\mathbf{x}_j$ is the Euclidean distance matrix defined on $\mathbf{X}$. For the above notations, $\circ$ denotes an element-wise multiplication of two matrices, $\|\cdot\|_F^2$ denotes the Frobenius-norm of a matrix, and tr denotes the trace of a matrix.

By further constraining the size of each local base to some fixed constant $B \in \mathbb{N}_+$, we can rewrite the formulation into the final formulation as follows:

$$\begin{aligned}\min_{\boldsymbol{\mu}\in\boldsymbol{\Delta},\mathbf{D}}\ & \frac{1}{2}\left\|\mathbf{X}\big(\mathbf{I}-\mathbf{K}\circ\mathbf{D}\big)\right\|_F^2+\gamma\mathrm{tr}\ \mathbf{K}\circ\mathbf{D}\circ\mathbf{M}\big(\mathbf{1}\mathbf{1}^\top\big) \\ \text{s.t.}\quad & \mathbf{D}\in\{0,1\}^{n\times n}, \|\mathbf{d}_i\|_1=B, i=1,\ldots,n,\end{aligned} \tag{6}$$

where $\boldsymbol{\Delta} = \{\boldsymbol{\mu} : \boldsymbol{\mu}^\top\mathbf{1} = 1, \boldsymbol{\mu} \ge 0\}$ is the domain of a simplex, the parameter $B \ll n$ is used to control the size of $\mathcal{B}_i$ for each $\mathbf{x}_i$, the kernel $\mathbf{K}$ is defined by $[\mathbf{K}]_{ij} = \sum_{t=1}^{m}\mu_t k^t(\mathbf{x}_i,\mathbf{x}_j), 1 \le i,j \le n$, and the previous parameter γ_1 is abbreviated as γ.

The above formulation essentially treats the kernel evaluation $\boldsymbol{\kappa}_i$ as a local coding coordinate of $\mathbf{x}_i$. With the additional constraint $\boldsymbol{\kappa}^\top\mathbf{1} = 1$, such a local coding system allows a linear approximation of a target function as shown by (Yu et al., 2009):

Theorem 1 *Let $(\boldsymbol{\kappa},\mathcal{B})$ be an arbitrary coordinate coding on $\mathbb{R}^d$. Let f be an (α,β,p)-Lipschitz smooth function, i.e., $|f(\mathbf{x})-f(\mathbf{x}')| \le \alpha\|\mathbf{x}-\mathbf{x}'\|$ and $|f(\mathbf{x}')-f(\mathbf{x})-\nabla f(\mathbf{x})^\top(\mathbf{x}'-\mathbf{x})| \le \beta\|\mathbf{x}-\mathbf{x}'\|^{1+p}$, $\alpha,\beta>0, p\in(0,1]$. For all $\mathbf{x}\in\mathbb{R}^d$, we have:*

$$\left|f(\mathbf{x})-\sum_{\mathbf{x}_j\in\mathcal{B}}\kappa(\mathbf{x},\mathbf{x}_j)f(\mathbf{x}_j)\right| \le \alpha\|\mathbf{x}-\sum_j\kappa(\mathbf{x},\mathbf{x}_j)\mathbf{x}_j\|+\beta\sum_{\mathbf{x}_j\in\mathcal{B}}|\kappa(\mathbf{x},\mathbf{x}_j)|\|\mathbf{x}_j-\sum_j\kappa(\mathbf{x},\mathbf{x}_j)\mathbf{x}_j\|^{1+p}.$$

For practical purpose, we exclude the constraint $\boldsymbol{\kappa}^\top\mathbf{1} = 1$. Instead, we deem the coding as kernel evaluation result, which is sampled from a predefined set $\mathcal{K}_{conv}$. Moreover, for each data point $\mathbf{x}_i$, we learn a set of local bases, rather than a global basis set. This allows to further exploit locality characterization of the data. Such local bases are also useful for discovering the data topology, which is essential for machine learning and data mining tasks such as data embedding.

3.2. Alternating Optimization Algorithm

It is not difficult to see that the optimization task (6) is essentially a mixed integer programming task, which is difficult to solve directly. In our approach, we propose an alternating optimization algorithm by solving $\boldsymbol{\mu}$ and $\mathbf{D}$ alternatively (Bezdek and Hathaway, 2003). The similar idea is commonly applied in many machine learning studies (Tan et al., 2010; Kang et al., 2011).

3.2.1. SOLVING μ BY FIXING $\mathbf{D}$

We first fix the variable $\mathbf{D}$ and then attempt to solve $\boldsymbol{\mu}$. Since $\mathbf{D}$ is fixed, all the original constraints on the variable $\mathbf{D}$ can be ignored. Further, it is not difficult to see that the objective function can be rewritten as:

$$J(\boldsymbol{\mu}) = \boldsymbol{\mu}^\top \left(\sum_{t=1}^{m} \sum_{i=1}^{n} \boldsymbol{\kappa}_{t,i} \boldsymbol{\kappa}_{t,i}^\top \circ \mathbf{d}_i \mathbf{d}_i^\top \circ \mathbf{P} \right) \top \boldsymbol{\mu} + \mathbf{z}^\top \boldsymbol{\mu}, \tag{7}$$

where $[\mathbf{z}]_t := \sum_{i=1}^{n} (2\gamma \mathbf{v} \circ \mathbf{d}_i - 2\mathbf{p}_i \circ \mathbf{d}_i)^\top \boldsymbol{\kappa}_{t,i}$, $\mathbf{P} = \mathbf{X}^\top \mathbf{X}$, $\boldsymbol{\kappa}_{t,i} := [k^t(\mathbf{x}_i, \mathbf{x}_1), \ldots, k^t(\mathbf{x}_i, \mathbf{x}_n)]^\top$ is the i-th column of the t-th kernel matrix, $\mathbf{p}$ and $\mathbf{v}$ are the columns of $\mathbf{P}$ and $\mathbf{M}$ corresponding to $\mathbf{x}_i$, respectively.

As a result, the original optimization problem (6) w.r.t. $\boldsymbol{\mu}$ essentially is reduced to a Quadratic Program (QP) (Boyd and Vandenberghe, 2004), which can be solved efficiently by an off-the-shelf convex optimization toolkit.

3.2.2. SOLVING $\mathbf{D}$ BY FIXING THE KERNEL $\mathbf{K}$

Second, we attempt to solve $\mathbf{D}$ by fixing the values of $\boldsymbol{\mu}$ and the kernel $\mathbf{K}$.

Notice that the columns of $\mathbf{D}$ are independent of each others. That is, the solution of a column $\mathbf{d}_i$ for a specific sample $\mathbf{x}_i$ would not affect the solutions to the bases of the other samples. Therefore, we can solve each column $\mathbf{d}_i$ of $\mathbf{D}$ separately.

In our approach, we propose a greedy algorithm to tackle the original mixed integer programming problem after fixing $\boldsymbol{\mu}$. To this end, we re-formulate (6) w.r.t. $\mathbf{d}$ as

$$J(\mathbf{d}) = \mathbf{d}^\top \big(\boldsymbol{\kappa}\boldsymbol{\kappa}^\top \circ \mathbf{P}\big)\mathbf{d} + \big(2\gamma \boldsymbol{\kappa} \circ \mathbf{v} - 2\boldsymbol{\kappa} \circ \mathbf{p}\big)^\top \mathbf{d}, \tag{8}$$

where the common subscript of $\mathbf{d}, \boldsymbol{\kappa}, \mathbf{p}, \mathbf{v}$ are omitted. Let $\mathbf{Q}$ abbreviate $\boldsymbol{\kappa}\boldsymbol{\kappa}^\top \circ \mathbf{P}$, and $\mathbf{c}$ abbreviate $2\gamma\boldsymbol{\kappa} \circ \mathbf{v} - 2\boldsymbol{\kappa} \circ \mathbf{p}$, we can simplify the objective function as $J(\mathbf{d}) = \mathbf{d}^\top \mathbf{Q}\mathbf{d} + \mathbf{c}^\top \mathbf{d}$. Next we propose a greedy algorithm to solve $\mathbf{d}$ based on the submodular functions (Nemhauser et al., 1978).

First of all, it is not difficult to show that $J(\mathbf{d})$ is a submodular function, i.e., $J(\mathbf{d}^1) + J(\mathbf{d}^2) \geq J(\mathbf{d}^1 | \mathbf{d}^2) + J(\mathbf{d}^1 \& \mathbf{d}^2)$, where $|$ and $\&$ are the bit-wise "or" and "and" operations, respectively. Following the principle of submodular functions, we initialize all the entries of $\mathbf{d}$ to zero (i.e., the base set $\mathcal{B}_i = \emptyset$); then at each step, we greedily add one sample $\mathbf{x}^*$ into the base set $\mathcal{B}_i$, which implies to set one component of $\mathbf{d}$ to 1. We choose the sample $\mathbf{x}^*$ as the one that minimizes the increase of $J(\mathbf{d})$, i.e.,

$$\mathbf{x}^* = \underset{\mathbf{x}_j \in \mathcal{X} - \mathcal{B}_i}{\arg\min}\, 2 \sum_{t:\mathbf{x}_t \in \mathcal{B}_i} Q_{tj} + Q_{jj} + c_j$$

This process is repeated until the number of nonzero entries of $\mathbf{d}$ is reached B.

The following theorem (Nemhauser et al., 1978) guarantees the performance of the above greedy approach of optimizing $\mathcal{B}_i$ (i.e., the solution of $\mathbf{d}$).

Theorem 2 *Let* $\mathbf{d}^*$ *denote the optimal solution of (8), and* $\hat{\mathbf{d}}$ *denote the approximation solution found by the greedy algorithm. We have*

$$J(\hat{\mathbf{d}}) \geq J(\mathbf{d}^*)(1 - 1/e),$$

if $J(\mathbf{d})$ *satisfies the following conditions: 1)* $J(\mathbf{d}^1) \leq J(\mathbf{d}^2)$ *if* $\mathbf{d}^1 \subset \mathbf{d}^2$*; 2)* $J(\mathbf{d})$ *is a submodular function, and 3)* $J(\mathbf{0}) = 0$.

Finally, Algorithm 1 summarizes the details of the proposed UMKL algorithm. After learning the kernel, it can be used in many machine learning tasks, including supervised, semi-supervised, and unsupervised learning tasks (e.g., classification, and dimension reduction, etc).

Algorithm 1 UMKL: Unsupervised Multiple Kernel Learning

Input: Unlabeled data $\mathbf{X}$, base kernels $\mathcal{K}_{base} = \{k_1, \ldots, k_m\}$, γ, B;
Output: Kernel weight $\boldsymbol{\mu}$, bases indicator matrix $\mathbf{D}$.

Initialize $[\mathbf{M}]_{ij} = \mathbf{x}_i^\top \mathbf{x}_i + \mathbf{x}_j^\top \mathbf{x}_j - 2\mathbf{x}_i^\top \mathbf{x}_j$, $\boldsymbol{\mu} = \mathbf{1}/m$, $\mathbf{P} = \mathbf{X}^\top \mathbf{X}$;
repeat
  $\mathbf{W} = \sum_{t=1}^{m} \sum_{i=1}^{n} \boldsymbol{\kappa}_{t,i} \boldsymbol{\kappa}_{t,i}^\top \circ \mathbf{d}_i \mathbf{d}_i^\top \circ \mathbf{P}$, $[\mathbf{z}]_t := \sum_{i=1}^{n} (2\gamma \mathbf{v} \circ \mathbf{d}_i - 2\mathbf{p}_i \circ \mathbf{d}_i)^\top \boldsymbol{\kappa}_{t,i}$;
  $\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu}} \boldsymbol{\mu}^\top \mathbf{W} \boldsymbol{\mu} + \mathbf{z}^\top \boldsymbol{\mu}$;
  for each $\mathbf{x}_i$ **do**
    Set $\mathcal{B}_i = \emptyset$; $\mathbf{Q} = \boldsymbol{\kappa} \boldsymbol{\kappa}^\top \circ \mathbf{P}$; $\mathbf{c} = \boldsymbol{\kappa}(2\gamma \mathbf{v} - 2\mathbf{p})$;
    for $|\mathcal{B}_i| < B$ **do**
      $\mathbf{x}^* = \arg\min_{\mathbf{x}_j \in \mathcal{X} - \mathcal{B}_i} 2 \sum_{t:\mathbf{x}_t \in \mathcal{B}_i} Q_{tj} + Q_{jj} + c_j$;
      $\mathcal{B}_i = \mathcal{B}_i \cup \{\mathbf{x}^*\}$;
    end for
  end for
until convergence

4. Experiments

In order to examine the efficacy of the proposed UMKL algorithm, we apply the proposed UMKL algorithm to two scenarios: (i) a pre-processing tool to find an appropriate kernel for classification task, and (ii) a nonlinear dimension reduction tool for machine learning and data mining tasks.

4.1. Experimental Testbed and Setup

We evaluate the performance of the proposed UMKL algorithms for binary classification tasks on a testbed with a number of publicly available data sets as shown in Table 1[1][2].

Following the settings of previous MKL studies (Xu et al., 2008), for each data set, we create the set of base kernels $\mathcal{K}$ as follows: (1) Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^6\}$) on all features and on each single feature; (2) polynomial kernels of degree 1 to 3 on all features and on each single feature. Each base kernel matrix is normalized to unit trace. The training instances are normalized to be of zero mean and unit variance, and the test instances are also normalized using the same mean and variance of the training data. To get stable results, for each data set, we repeat each algorithm 20 times and compute the average results of the 20 runs.

1. http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
2. http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark

Table 1: The statistics of the 12 binary-class data sets used in our experiments.

Data Set	Breast	Diabetes	Waveform	Sonar	Liver	German
# instances	683	768	400	208	345	1,000
# dimensions	10	8	21	60	6	24
Data Set	Australian	Thyroid	Heart	Banana	Titanic	FlareSolar
# instances	690	140	270	400	150	666
# dimensions	14	5	13	2	3	9

4.2. Experiment I: Unsupervised MKL versus Supervised MKL

The first experiment is to examine whether the proposed UMKL algorithm can produce good results yb comparing it with traditional supervised kernel learning (SKL) algorithms on the same set of training data. Specifically, we have compared the following algorithms:

AvgKernel: The average combination of multiple kernels;

MKL$^{\text{Level}}$: The convex multiple kernel learning algorithm, that is, the target kernel class is $\mathcal{K}_{conv}$ defined in (3). We use the extended level method (Xu et al., 2008) to learn the kernel;

LpMKL: The MKL algorithm with L_p norm regularization over the kernel weight (Kloft et al., 2009). We adopt their cutting plane algorithm with second order Taylor approximation of L_p;

GMKL: The Generalized MKL algorithm in (Varma and Babu, 2009). The target kernel class is the Hadamard product of single Gaussian kernel defined on each dimension;

KTA: The Two-stage Kernel Learning by Kernel Target Alignment in (Cortes et al., 2010);

UMKL: The proposed Unsupervised MKL algorithm with the greedy base set selection approach. We first learn the kernel by UMKL and then apply the kernel for learning an SVM classifier.

For parameter settings, the regularization parameter C in SVM for all the compared methods is determined by 5-fold cross validation on the training data over the range of $\{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$. For a fair comparison, the same set of base kernels was adopted by MKL$^{\text{Level}}$, LpMKL, KTA, and UMKL. For LpMKL, we examine $p = 2, 3, 4$ and report the best result. For the proposed UMKL algorithm, the parameters γ and B were also chosen by cross validation on the training data.

For each data set, we randomly sample 50% of all instances as training data, and use the rest as test data. The classification accuracy is shown in Table 2, from which we can draw two observations.

First, UMKL is comparable with supervised kernel learning algorithms. Among the evaluated 12 data sets, UMKL reports 6 best results, which is the largest among all the evaluated methods. None of the compared 4 algorithms, i.e., MKL, LpMKL, GMKL, and KTA, can outperform the proposed UMKL algorithm significantly. The average rank of UMKL is comparable with KTA. These three algorithms are significantly better than MKL and GMKL. This result verifies the efficacy of UMKL clearly. When the half of the data have labels for training, SKL and UMKL, which does not use the label information, produce similar classification accuracy.

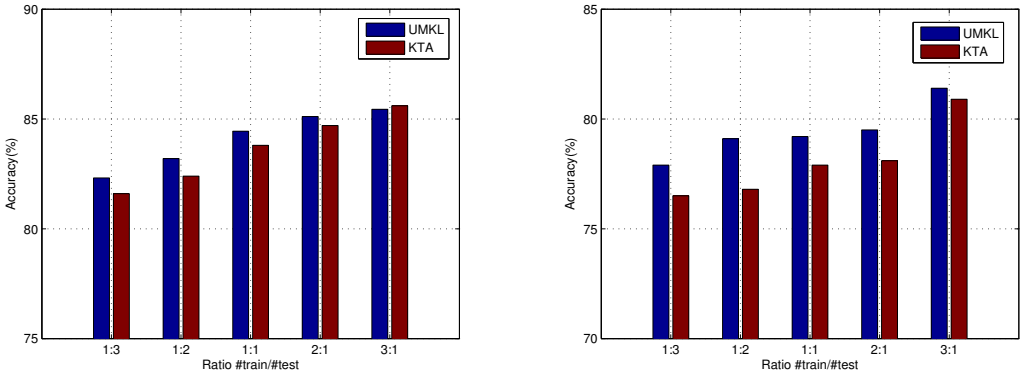
Second, among the SKL algorithms, KTA yield the best performance. The average rank of KTA among SKL algorithms is the best. Traditional MKL cannot result in the best accuracy on any evaluated data sets. The results of KTA on *thyroid* are statistically significant better than other SKL

Table 2: The evaluation of classification performance of a number of different algorithms. Each element in the table shows the mean and standard deviation of classification accuracy (%). The **bold** element indicates the best performance.

Data Set	AvgKernel	$\text{MKL}^{\text{level}}$	LpMKL	GMKL	KTA	UMKL
Breast	95.5±1.0	96.5±0.8	96.2±0.7	**97.0±1.0**	96.5±0.8	**97.0±0.6**
Diabetes	65.4±2.0	75.8±2.5	72.6±2.5	66.4±2.5	75.2±3.3	**77.1±2.3**
Titanic	76.2±4.2	77.1±2.9	77.0±3.0	76.7±3.1	78.9±1.2	**79.2±2.5**
Heart	75.6±6.0	83.0±2.9	76.7±3.8	77.0±3.6	82.0±1.9	**84.7±2.6**
German	69.5±1.5	71.4±2.8	**74.3±1.4**	70.4±1.6	72.1±1.1	**74.8±1.8**
Australian	66.7±5.6	85.0±1.5	84.5±1.6	80.0±2.3	**87.2±0**	86.3±1.3
Banana	86.1±2.3	90.2±2.0	87.5±2.6	83.4±2.7	**92.1±1.7**	90.2±1.9
Thyroid	82.6±4.4	92.9±2.9	93.1±2.2	94.6±2.1	**97.9±1.3**	91.2±2.7
FlareSolar	64.8±1.6	**67.6±2.0**	64.8±1.8	65.3±1.8	66.7±2.3	**67.8±1.8**
Waveform	73.5±7.1	88.2±1.6	**88.9±2.0**	88.2±1.8	88.5±2.5	88.4±2.5
Sonar	59.1±9.7	78.3±3.5	**84.8±3.2**	78.8±4.6	81.3±2.9	81.0±2.9
Liver	57.6±2.3	62.3±4.5	**69.4±2.9**	63.6±2.6	68.7±1.5	68.3±4.3

methods. Figure 1 illustrates the performance comparison of both UMKL and KTA algorithms by varying the fraction of training data instances on two datasets. For most situtations, the performance of UMKL is consistently better or comparable to that of the KTA algorithm, which again validates the efficacy of the unsupervised multiple kernel learning algorithm.

Third, to further examine why UMKL works well, we compare the performance of UMKL and KTA on two datasets: *heart* and titanic, by varying the ratios between training data and test data. Figure 1 shows the comparison results. When the number of training data is small, the unsupervised approach outperforms the supervised approach, which probably because the supervised multiple kernel learning algorithms suffer from over-fitting when training data is insufficient. We can see that as the training data becomes larger, the performance of KTA grows quickly and the difference between UMKL and KTA becomes smaller.



(*a*) heart (*b*) titanic

Figure 1: Classification accuracy of UMKL and KTA by varying #train/#test.

4.3. Experiment II: UMKL for Dimensionality Reduction

Our second experiment is to examine if the proposed UKML algorithm is effective for an unsupervised dimension reduction task using the kernel learning techniques. To this purpose, we apply the UMKL algorithm as a tool to learn an appropriate kernel function for Kernel PCA, which is a classical nonlinear dimensionality reduction technique that chooses the dominating principle component in the feature space. Typically, it is often non-trivial to select the appropriate kernel for kernel PCA based on given data.

Specifically, in this experiment, we compare the following different approaches of choosing kernels for kernel PCA towards dimension reduction tasks:

Gaussian: We consider a single Gaussian kernel, where the band-width parameter σ was fixed to 1 in our experiments;

Average: The average kernel by combining the set of multiple kernels via a uniformly linear combination approach, where the set of multiple kernels is the same as the previous experiment;

MMUKL: The maximum margin based unsupervised kernel learning algorithm (Valizadegan and Jin, 2006), which was inspired by Maximum Margin Clustering for learning a linear combination of multiple kernels towards clustering;

UMKL: The proposed UMKL algorithm to identify the appropriate linear combination of multiple kernels on the same set of multiple kernels, in which the parameter γ was fixed to 100, and B was fixed to 10.

Table 3: The classification accuracy of a 5-NN classifier on the projected data by KPCA with three different kinds of kerenls, i.e., a gaussian kernel with fixed sigma=1, the average combination of multiple kernels, and the kernel learned by UMKL on 9 benchmark data sets. For each dataset, the **bold** element indicates the best performance.

Data Set	Gaussian Kernel	Average Kernel	MMUKL	UMKL
Breast	95.1±0.8	92.7±2.7	**95.7±0.9**	95.2±1.3
Diabetes	64.6±2.4	65.1±2.3	63.8±2.6	**66.3±2.5**
Heart	56.2±3.5	61.5±5.8	57.4±4.3	**72.4±5.9**
German	63.9±1.7	64.3±1.9	64.1±1.5	**64.9±2.2**
Australian	57.7±2.3	55.9±2.6	67.1±8.5	**72.9±3.8**
FlareSolar	61.9±3.4	62.5±3.6	62.4±3.8	**63.0±3.5**
Waveform	59.4±3.5	63.2±4.1	66.5±8.7	**76.9±3.9**
Sonar	50.7±5.0	56.7±5.8	52.6±7.3	**57.0±7.2**
Liver	52.9±4.5	52.0±3.3	**53.6±2.6**	53.2±3.3

In this experiment, for each of the above approaches, we apply it to reduce data from the original dimensionality to 2 dimensions, and then adopt a 5-nearest neighbor classification scheme to evaluate the classification performance on the reduced data. For each data set, we randomly sample 50%

of all instances as training data, and use the rest as test data. To obtain stable results, for each data set, we repeat the randomized sampling process 20 times, run each algorithm on these randomly sampled training/test data, and finally compute the average results of each algorithm over these 20 runs. We adopt 9 benchmark datasets as used in Experiment I [3].

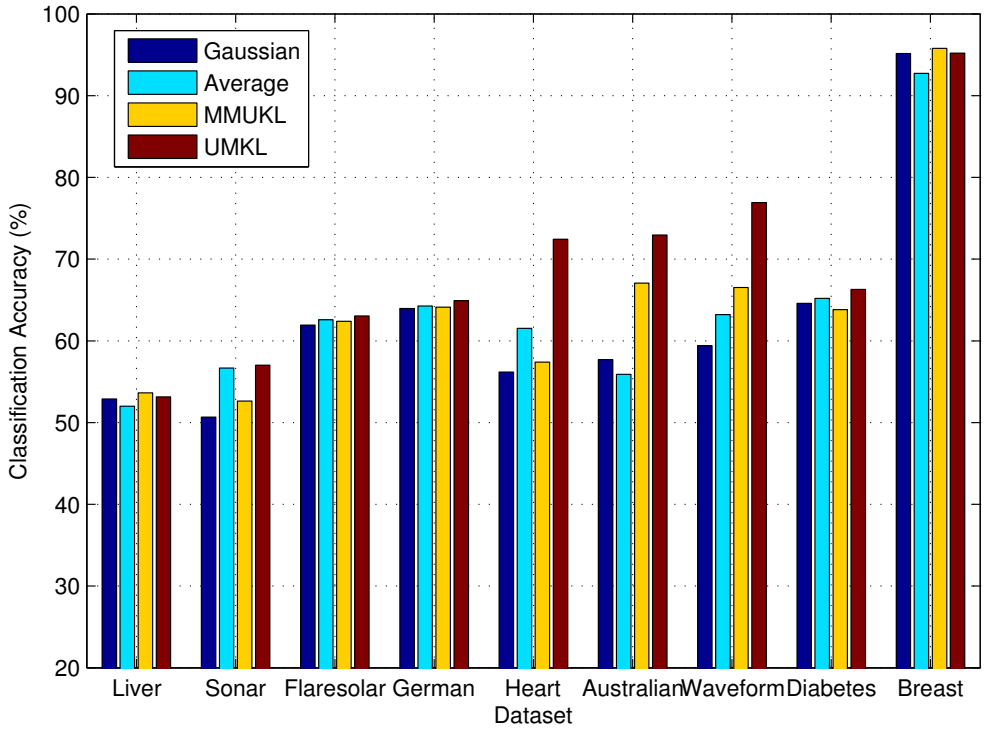


Figure 2: The classification accuracy of 5-NN after KPCA with Gaussian kernel of sigma=1,the average combination of multiple kernels, the kernel learned by UMKL on 9 benchmark data sets. The parameter B of UMKL is fixed to 10.

Table 3 and Figure 2 summarize the experimental results of the kernel PCA by three kinds of different kernels for supervised classification tasks. From the results, we could see that the proposed UMKL algorithm is able to learn a better kernel that is considerably more effective than either a single gaussian kernel or an average kernel via a uniformly linear combination for most cases. The results are particularly impressive on several datasets including heart, australian, waveform, where UMKL significantly surpasses the other two approaches.

In addition to the fixed number of reduced dimensions, we also try to examine how the compared algorithms work when applying KPCA to obtain projected data of a varied numbers of dimensions. Figure 3 shows the experimental results of evaluating the classification performance on the data by applying KPCA to obtain low-dimensional data of varied numbers of dimensions. From the results,

3. Note that we do not use three of the datasets (i.e., Banana, Titanic, and THyroid) because of their original dimensionality is already very slow (e.g., 2,3,5 respectively)

it is clear that UMKL performs consistently better than both the single gaussian kernel and the average kernel on all the cases of varied numbers of dimensions.

Moreover, we also evaluate the classification performance by examining the effects of varied numbers of nearest neighbors, i.e., k, used in the k-NN classifiers. Figure 4 shows the detailed results of classification evaluation by varying the number of nearest neighbors on the *waveform* dataset. The proposed UMKL algorithm consistently surpasses the other two baselines. Figure 5 gives more results on the other datasets. For most of the datasets, the proposed UMKL algorithm achieves the best or close to the best performance among all compared schemes.

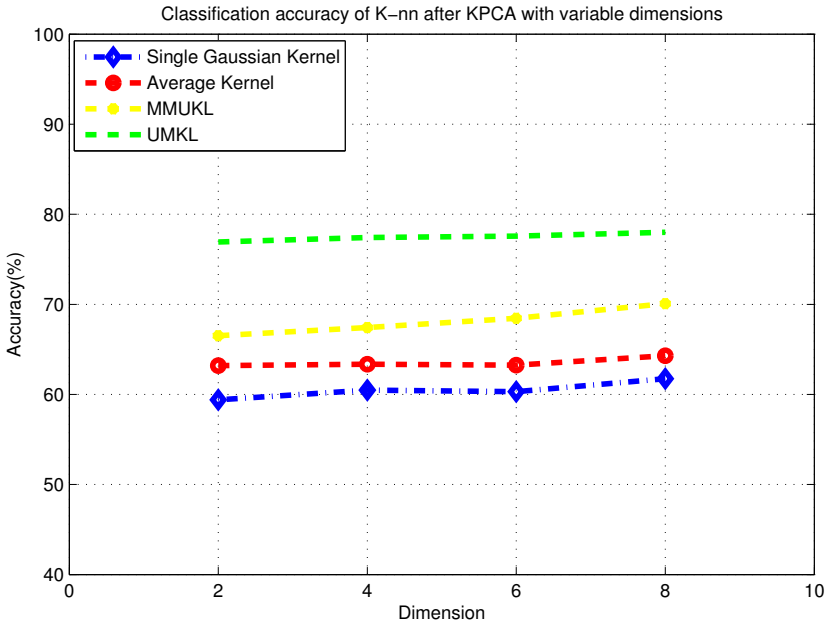


Figure 3: The classification accuracy of 5-NN on the embedded data of the *waveform* dataset with varied number of dimensions by applying KPCA with four different kernels: (i) a gaussian kernel of sigma=1, (ii) an average combination of multiple kernels, (iii) the kernel learned by MMUKL, and (iv) the kernel learned by UMKL.

Finally, we examine the time efficiency of the two different unsupervised kernel learning algorithms for learning an optimal linear combination of multiple kernels. Table 4 shows the evaluation of average running time costs taken by the two algorithms on different datasets. It is clear to see that the proposed UMKL algorithm is significantly more efficient than the MMUKL algorithm. This is because, unlike the proposed simple algorithm, MMUKL has to resolve a semi-definite program, which is often highly computationally intensive, making it inefficient and non-scalable for large applications.

Table 4: The average running time cost of kernel learning by MMUKL and UMKL (seconds).

Data Set	heart	waveform	sonar	liver	flaresolar	breast	diabetes	australian	german
MMUKL	5.59	15.22	1.43	11.80	94.76	128.57	89.34	124.92	233.42
UMKL	0.61	3.27	0.37	2.00	9.68	11.07	23.97	9.22	31.39

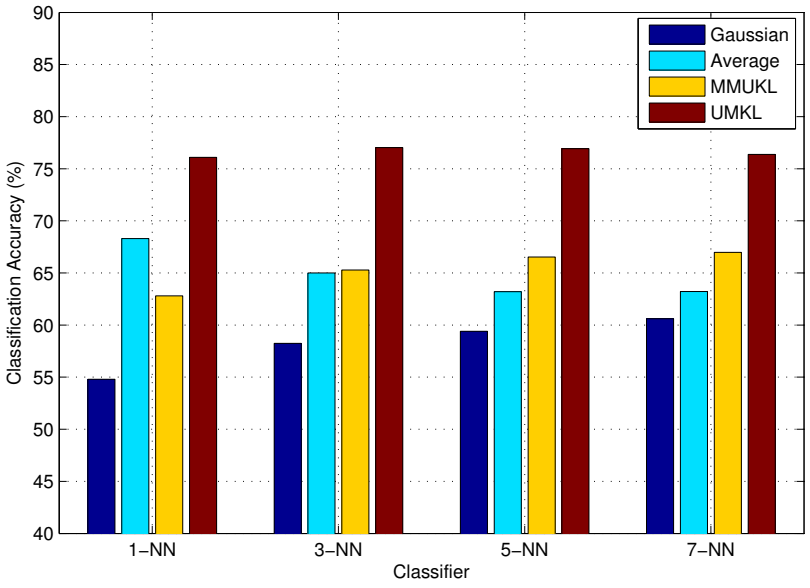


Figure 4: The classification accuracy of k-NN with varied numbers of nearest neighbors on the embedded data of the *waveform* dataset by applying KPCA with four different kernels: (i) a gaussian kernel of sigma=1, (ii) an average combination of multiple kernels, (iii) the kernel learned by MMUKL, (iv) the kernel learned by UMKL.

5. Conclusion

In this paper we propose a novel unsupervised multiple kernel learning (UMKL) method, which is able to identify an appropriate linear combination of multiple kernels purely from unlabeled data. In particular, the proposed UKML approach deems the kernel evaluation result of a data instance as its local coding, and adopts an efficient iterative algorithm to learn the kernel and the local set of basis simultaneously. We apply UKML for two machine learning tasks: (i) kernel learning for choosing appropriate kernels in a classification task, and (ii) kernel selection in a kernel-based dimension reduction task based on the well-known Kernel PCA technique. Empirical results show that the classifier using the kernel learned by UMKL is comparable with the state-of-the-art supervised MKL algorithms, and the KPCA scheme using the kernel learned by UKML considerably outperforms the conventional approaches. Future research will address the theoretical analysis of the Unsupervised Multiple Kernel Learning algorithm and apply the proposed technique to more applications.

Acknowledgements

The authors would like to thank Ivor Tsang for his comments on some of the optimization issues. This work was supported by Singapore MOE ARC tier-2 research grant (T208B2203) and Microsoft Research grant.

References

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

James C. Bezdek and Richard J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11:351–368, December 2003. ISSN 1061-5369.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Mehryar Mohri Corinna Cortes and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *NIPS*, 2009a.

Mehryar Mohri Corinna Cortes and Afshin Rostamizadeh. L2 regularization for learning kernels. In *UAI*, 2009b.

Corinna Cortes. Invited talk: Can learning kernels help performance? In *ICML*, page 161, 2009.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *ICML*, pages 239–246, 2010.

Peter Vincent Gehler and Sebastian Nowozin. Infinite kernel learning. In *TECHNICAL REPORT NO. TR-178,Max Planck Institute for Biological Cybernetics*, 2008.

Thomas Hofmann, Bernhard Scholkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

Steven C.H. Hoi, Rong Jin, and Michael R. Lyu. Learning non-parametric kernel matrices from pairwise constraints. In *Proceedings of the 24th International Conference on Machine Learning (ICML2007)*, OR, US, June 20–24 2007.

Rong Jin, Steven C.H. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *21st Intl. Conf. on Algorithmic Learning Theory (ALT'10)*, pages 390–404, 2010.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proc. 28th International Conference on Machine Learning (ICML'11)*, pages 521–528, Bellevue, Washington, USA, June 2011.

Marius Kloft, Ulf Brefeld, Soren Sonnenburg, Pavel Laskov, Klaus-Robert Muller, and Alexander Zien. Efficient and accurate l_p-norm multiple kernel learning. In *NIPS*, 2009.

Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Dhanesh Ramachandram M. Ehsan Abbasnejad and Rajeswari Mandava. An unsupervised approach to learn the kernel functions: from global influence to local similarity. *Neural Computing & Applications*, 2010.

Sebastian Mika, Bernhard Schölkopf, Alex J. Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *NIPS*, pages 536–542, 1998.

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

Nathan Srebro and Shai Ben-David. Learning bounds for support vector machines with learned kernels. In *COLT*, pages 169–183, 2006.

Mingkui Tan, Li Wang, and Ivor W. Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, pages 1047–1054, 2010.

Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *NIPS*, pages 1417–1424, 2006.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, page 134, 2009.

Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2008.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, 2010.

Yiming Ying and Colin Campbell. Generalization bounds for learning the kernel. In *COLT*, 2009.

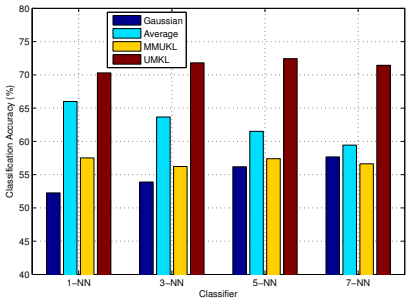
Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *NIPS*, 2009.

Bin Zhao, James T. Kwok, and Changshui Zhang. Multiple kernel clustering. In *SDM*, pages 638–649, 2009.

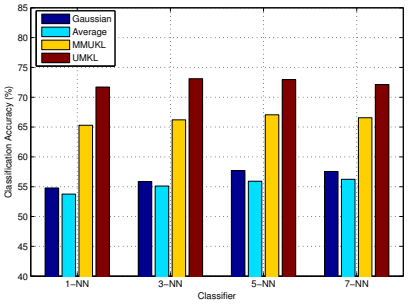
Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2001.

Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. Two-layer multiple kernel learning. In *JMLR Workshop and Conference Proceedings (AISTATS-2010)*, volume 9, pages 988–995, 2011a.

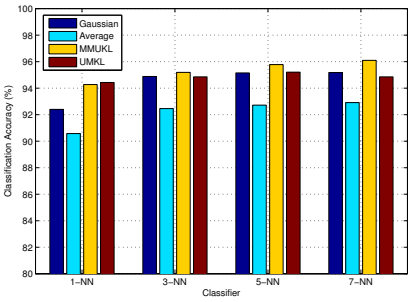
Jinfeng Zhuang, Ivor W. Tsang, and Steven C.H. Hoi. A family of simple non-parametric kernel learning algorithms. *Journal of Machine Learning Research*, 12:1313–1347, 2011b.



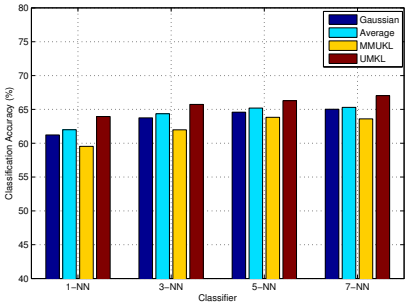
(*a*) Heart



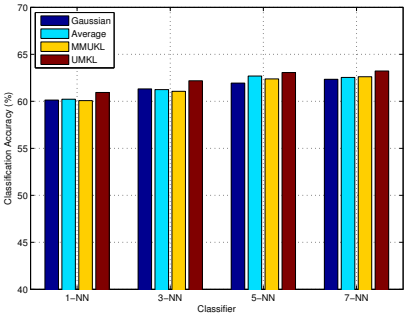
(*b*) Australian



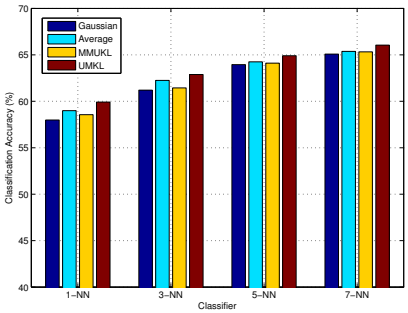
(*c*) Breast



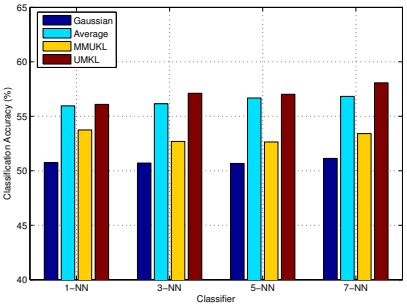
(*d*) Diabetes



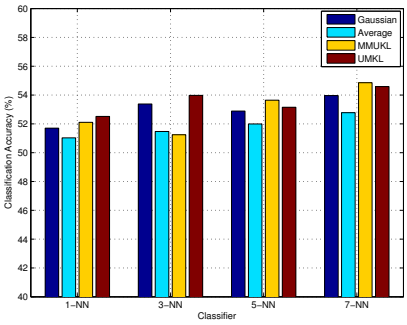
(*e*) Flaresolar



(*f*) German



(*g*) Sonar



(*h*) Liver

Figure 5: The classification accuracy of k-NN on the embedded data by applying KPCA with four different kernels: (i) a gaussian kernel of sigma=1, (ii) uniform combination of multiple kernels, (iii) the kernel learned by MMUKL, (iv) the kernel learned by UMKL.