On Multiple Kernel Learning with Multiple Labels

Lei Tang
Department of CSE
Arizona State University
L.Tang@asu.edu

Jianhui Chen
Department of CSE
Arizona State University
Jianhui.Chen@asu.edu

Jieping Ye
Department of CSE
Arizona State University
Jieping.Ye@asu.edu

Abstract

For classification with multiple labels, a common approach is to learn a classifier for each label. With a kernel-based classifier, there are two options to set up kernels: select a specific kernel for each label or the same kernel for all labels. In this work, we present a unified framework for multi-label multiple kernel learning, in which the above two approaches can be considered as two extreme cases. Moreover, our framework allows the kernels shared partially among multiple labels, enabling flexible degrees of label commonality. We systematically study how the sharing of kernels among multiple labels affects the performance based on extensive experiments on various benchmark data including images and microarray data. Interesting findings concerning efficacy and efficiency are reported.

1 Introduction

With the proliferation of kernel-based methods like support vector machines (SVM), kernel learning has been attracting increasing attentions. As widely known, the kernel function or matrix plays an essential role in kernel methods. For practical learning problems, different kernels are usually prespecified to characterize the data. For instance, Gaussian kernel with different width parameters; data fusion with heterogeneous representations [Lanckriet *et al.*, 2004b]. Traditionally, an appropriate kernel can be estimated through cross-validation. Recent multiple kernel learning (MKL) methods [Lanckriet *et al.*, 2004a] manipulate the Gram (kernel) matrix directly by formulating it as a semi-definite program (SDP), or alternatively, search for an optimal convex combination of multiple user-specified kernels via quadratically constrained quadratic program (QCQP). Both SDP and QCQP formulations can only handle data of medium size and small number of kernels. To address large scale kernel learning, various methods are developed, including SMO-like algorithm [Bach *et al.*, 2004], semi-infinite linear program (SILP) [Sonnenburg *et al.*, 2007] and projected gradient method [Rakotomamonjy *et al.*, 2007]. It is noticed that most existing works on MKL focus on binary classifications. In this work, MKL (learning the weights for each base kernel) for classification with multiple labels is explored instead.

Classification with multiple labels refers to classification with more than 2 categories in the output space. Commonly, the problem is decomposed into multiple binary classification tasks, and the tasks are learned independently or jointly. Some works attempt to address the kernel learning problem with multiple labels. In [Jebara, 2004], all binary classification tasks share the same Bernoulli prior for each kernel, leading to a sparse kernel combination. [Zien, 2007] discusses the problem of kernel learning for multi-class SVM, and [Ji *et al.*, 2008] studies the case for multi-label classification. Both works above exploit the same kernel directly for all classes, yet *no empirical result is formally reported concerning whether the same kernel across labels performs better over a specific kernel for each label*.

The same-kernel-across-tasks setup seems reasonable at first glimpse but needs more investigation. Mostly, the multiple labels are within the same domain, and naturally the classification tasks share some commonality. One the other hand, the kernel is more informative for classification when it is aligned with the target label. Some tasks (say recognize *sunset* and *animal* in images) are quite distinct, so a specific kernel for each label should be encouraged. Given these considerations, two questions rises naturally:

- Which approach could be better, the same kernel for all labels or a specific kernel for each label? To our best knowledge, no work has formally studied this issue yet.
- A natural extension is to develop kernels that capture the similarity and difference among labels simultaneously. This matches the relationship among labels more reasonably, but could it be effective in practice?

The questions above motivate us to develop a novel framework to model task *similarity* and *difference* simultaneously when handling multiple related classification tasks. We show that the framework can be solved via QCQP with proper regularization on kernel difference. To be scalable, an SILP-like algorithm is provided. In this framework, selecting the same kernel for all labels or a specific kernel for each label are deemed as two extreme cases. Moreover, this framework allows various degree of kernel sharing with proper parameter setup, enabling us to study different strategies of kernel sharing systematically. Based on extensive experiments on benchmark data, we report some interesting findings and explanations concerning the aforementioned two questions.

2 A Unified Framework

To systematically study the effect of kernel sharing among multiple labels, we present a unified framework to allow flexible degree of kernel sharing. We focus on the well-known kernel-based algorithm SVM, for learning k binary classification tasks $\{f^t\}_{t=1}^k$ respectively, based on n training samples $\{(x_i, y_i^t)\}_{i=1}^n$, where t is the index of a specific label. Let $\mathcal{H}_K^t$ be the feature space, and ϕ_K^t be the mapping function defined as $\phi_K^t : \phi_K^t(x) \rightarrow \mathcal{H}_K^t$, for a kernel function K^t. Let $\mathcal{G}^t$ be the kernel (Gram) matrix for the t-th task, namely $\mathcal{G}_{ij}^t = K^t(x_i, x_j) = \langle \phi_K^t(x_i) \cdot \phi_K^t(x_j) \rangle$. Under the setting of learning multiple labels $\{f^t\}_{t=1}^k$ using SVM, each label f^t can be seen as learning a linear function in the feature space $\mathcal{H}_K^t$, such that $f^t(x) = \text{sign}(\langle w^t, \phi_K^t(x) \rangle + b^t)$ where w^t is the feature weight and b^t is the bias term.

Typically, the dual formulation of SVM is considered. Let $D(\alpha^t, \mathcal{G}^t)$ denote the dual objective of the t-th task given kernel matrix $\mathcal{G}^t$:

$$D(\alpha^t, \mathcal{G}^t) = [\alpha^t]^T \mathbf{e} - \frac{1}{2}[\alpha^t]^T \left(\mathcal{G}^t \circ \mathbf{y}^t[\mathbf{y}^t]^T\right) \alpha^t \tag{1}$$

where for the task f^t, $\mathcal{G}^t \in S_+^n$ denotes the kernel matrix and S_+^n is the set of semi-positive definite matrices; $\circ$ denotes element-wise matrix multiplication; $\alpha^t \in \mathbb{R}^n$ denotes the dual variable vector. Mathematically, multi-label learning with k labels can be formulated in the dual form as:

$$\begin{aligned} \max_{\{\alpha^t\}_{t=1}^k} \quad & \sum_{t=1}^k D(\alpha^t, \mathcal{G}^t) \qquad (2) \\ \text{s.t.} \quad & [\alpha^t]^T \mathbf{y}^t = 0, \ \ 0 \leq \alpha^t \leq C, \ \ t = 1, \cdots, k \end{aligned}$$

Here, C is the penalty parameter for allowing the misclassification. Given $\{\mathcal{G}^t\}_{t=1}^k$, optimal $\{\alpha^t\}_{t=1}^k$ in Eq. (2) can be found by solving a convex problem.

Note that the dual objective is the same as the primal objective of SVM due to its convexity (equal to the empirical classification loss plus model complexity). Following [Lanckriet *et al.*, 2004a], multiple kernel learning with k labels and p base kernels $G_1, G_2, \cdots, G_p$ can be formulated as:

$$\begin{aligned} \min_{\{\mathcal{G}^t\}_{t=1}^k} \quad & \lambda \cdot \Omega(\{\mathcal{G}^t\}_{t=1}^k) + \max_{\{\alpha^t\}_{t=1}^k} \sum_{t=1}^k D(\alpha^t, \mathcal{G}^t) \qquad (3) \\ \text{s.t.} \quad & [\alpha^t]^T \mathbf{y}^t = 0, \ \ 0 \leq \alpha^t \leq C, \ \ t = 1, \cdots, k \\ & \mathcal{G}^t = \sum_{i=1}^p \theta_i^t G_i, \ \ t = 1, \cdots, k \qquad (4) \\ & \sum_{i=1}^p \theta_i^t = 1, \ \ \theta^t \geq 0, \ \ t = 1, \cdots, k \qquad (5) \end{aligned}$$

where $\Omega(\{\mathcal{G}^t\}_{t=1}^k)$ is a regularization term to represent the cost associated with kernel differences among labels. To capture the commonality among labels, Ω should be a *monotonic* increasing function of kernel difference. λ is the trade-off parameter between kernel difference and classification loss.

Clearly, if λ is set to 0, the objective goal is decoupled into k sub-problems, with each selecting a kernel independently (Independent Model); When λ is sufficiently large, the regularization term dominates and forces all labels to select the same kernel (Same Model). In between, there are infinite number of Partial Model which control the degrees of kernel difference among tasks. The larger λ is, the more similar the kernels of each label are.

3 Regularization on Kernel Difference

Here, we develop one regularization scheme such that formula (3) can be solved via convex programming. Since the optimal kernel for each label is expressed as a convex combination of multiple base kernels as in eq. (4) and (5), each θ^t essentially represents the kernel associated with the t-th label. We decouple the kernel weights of each label into two non-negative parts:

$$\theta_i^t = \zeta_i + \gamma_i^t, \ \ \zeta_i, \gamma_i^t \geq 0 \tag{6}$$

where ζ_i denotes the shared kernel across labels, and γ_i^t is the label-specific part. So the kernel difference can be defined as:

$$\Omega\left(\{\mathcal{G}^t\}_{t=1}^k\right) = \frac{1}{2} \sum_{t=1}^k \sum_{i=1}^p \gamma_i^t \tag{7}$$

For presentation convenience, we denote

$$G_i^t(\alpha) = [\alpha^t]^T \left(G_i^t \circ \mathbf{y}^t[\mathbf{y}]^T\right) \alpha^t \tag{8}$$

It follows that the MKL problem can be solved via QCQP[1].

Theorem 3.1. *Given regularization as presented in* (6) *and* (7)*, the problem in* (3) *is equivalent to a Quadratically Constrained Quadratic Program (QCQP):*

$$\begin{aligned} \max \quad & \sum_{t=1}^k [\alpha^t]^T e - \frac{1}{2} s \\ \textit{s.t.} \quad & s \geq s_0, \ \ s \geq \sum_{t=1}^k s_t - k\lambda \\ & s_0 \geq \sum_{t=1}^k G_i^t(\alpha), \quad i = 1, \cdots, p \\ & s_t \geq G_i^t(\alpha), \quad i = 1, \cdots, p, \ \ t = 1, \cdots, k \\ & [\alpha^t]^T \mathbf{y}^t = 0, \ \ 0 \leq \alpha^t \leq C, \quad t = 1, \cdots, k \end{aligned}$$

The kernel weights of each label (ζ_i, γ_i^t) can be obtained via the dual variables of the constraints.

The QCQP formulation involves $nk+2$ variables, $(k+1)p$ quadratic constraints and $O(nk)$ linear constraints. Though this QCQP can be solved efficiently by general optimization software, the quadratic constraints might exhaust memory resources if k or p is large. Next, we'll show a more scalable algorithm that solves the problem efficiently.

4 Algorithm

The objective in (3) given λ is equivalent to the following problem with a proper β and other constraints specified in (3):

$$\min_{\{\mathcal{G}^t\}} \max_{\{\alpha^t\}} \sum_{t=1}^k \left\{ [\alpha^t]^T \mathbf{e} - \frac{1}{2}[\alpha^t]^T \left(\mathcal{G}^t \circ \mathbf{y}^t[\mathbf{y}^t]^T\right) \alpha^t \right\} \tag{9}$$

$$\text{s.t.} \quad \Omega(\{\mathcal{G}^t\}_{t=1}^k) \leq \beta \tag{10}$$

[1] Please refer to the appendix for the proof.

Compared with λ, β has an explicit meaning: the maximum difference among kernels of each label. Since $\sum_{i=1}^{p} \theta_i^t = 1$, the min-max problem in (9), akin to [Sonnenburg *et al.*, 2007], can be expressed as:

$$\min \quad \sum_{t=1}^{k} g^t \tag{11}$$

$$\text{s.t.} \quad \sum_{i=1}^{p} \theta_i^t D(\alpha^t, G_i) \le g^t, \ \forall \alpha^t \in S(t) \tag{12}$$

$$\text{with} \quad S(t) = \left\{ \alpha^t | 0 \le \alpha^t \le C, \ [\alpha^t]^T \mathbf{y}^t = 0 \right\} \tag{13}$$

Note that the max operation with respect to α^t is transformed into a constraint for all the possible α^t in the set $S(t)$ defined in (13). An algorithm similar to cutting-plane could be utilized to solve the problem, which essentially adds constraints in terms of α^t iteratively. In the J-th iteration, we perform the following:

1). Given θ_i^t and g^t from previous iteration, find out new α_J^t in the set (13) which violates the constraints (12) most for each label. Essentially, we need to find out α^t such that $\sum_{i=1}^{p} \theta_i^t D(\alpha^t, G_i)$ is maximized, which boils down to an SVM problem with fixed kernel for each label:

$$\max_{\alpha^t} \quad [\alpha^t]^T \mathbf{e} - \frac{1}{2} [\alpha^t]^T \left(\sum_{i=1}^{p} \theta_i^t G_i^t \circ \mathbf{y}^t [\mathbf{y}^t]^T \right) \alpha^t$$

Here, each label's SVM problem can be solved independently and typical SVM acceleration techniques and existing SVM implementations can be used directly.

2). Given α_J^t obtained in Step 1, add k linear constraints:

$$\theta_i^t D(\alpha_J^t, G_i) \le g^t, \ \ t = 1, \cdots, k$$

and find out new θ^t and g^t via the problem below:

$$\begin{aligned}
\min \quad & \sum_{t=1}^{k} g^t \\
\text{s.t.} \quad & \sum_{i=1}^{p} \theta_i^t D(\alpha_j^t, G_i) \le g^t, \quad j = 1, \cdots, J \\
& \theta^t \ge 0, \ \sum_{i=1}^{p} \theta_i^t = 1, \quad t = 1, \cdots, k \\
& \frac{1}{2} \sum_{t=1}^{k} \sum_{i=1}^{p} \gamma_i^t \le \beta, \ \theta_i^t = \zeta_i + \gamma_i^t, \ \zeta_i, \gamma_i^t \ge 0
\end{aligned}$$

Note that both the constraints and the objective are linear, so the problem can be solved efficiently by general optimization package.

3). $J = J + 1$. Repeat the above procedure until no α is found to violate the constraints in Step 1.

So in each iteration, we interchangeably solve k SVM problems and a linear program of size $O(kp)$.

5 Experimental Study

5.1 Experiment Setup

4 multi-label data sets are selected as in Table 1. We also include 5 multi-class data sets as they are special cases of multi-label classification and one-vs-all approach performs reasonably well [Rifkin and Klautau, 2004]. We report average AUC and accuracy for multi-label and multi-class data, respectively. A portion of data are sampled from *USPS*, *Letter* and *Yaleface*, as they are too large to handle directly. Various type of base kernels are generated. We generate 15 diffusion kernels with parameter varying from 0.1 to 6 for *Ligand* [Tsuda and Noble, 2004]; The 8 kernels of *Bio* are generated following [Lanckriet *et al.*, 2004b]; *20news* uses diverse text representations [Kolda, 1997] leading to 62 different kernels; For other data, Gaussian kernels with different widths are constructed. The trade-off parameter C of SVM is set to a sensible value based on cross validation. We vary the number of samples for training and randomly sample 30 different subsets in each setup. The average performance are recorded.

Table 1: Data Description

	Data	#samples	#labels	#kernels
Multi-label	Ligand	742	36	15
	Bio	3588	13	8
	Scene	2407	6	15
	Yeast	2417	14	15
Multi-class	USPS	1000	10	10
	Letter	1300	26	10
	Yaleface	1000	10	10
	20news	2000	20	62
	Segment	2310	7	15

5.2 Experiment Results

Due to space limit, we can only report some representative results in Table 3-5. The details are presented in an extended technical report [Tang *et al.*, 2009]. *Tr Ratio* in the first row denotes the training ratio, the percentage of samples used for training. The first column denotes the portion of kernels shared among labels via varying the parameter β in (10), with Independent and Same model being the extreme. The last row (*Diff*) represents the performance difference between the best model and the worst one. Bold face denotes the best in each column unless there is no significant difference. Below, we seek to address the problems raised in the introduction.

Does kernel regularization yield any effect?

The maximal difference of various models on all data sets are plotted in Figure 1. The x-axis denotes increasing training data and y-axis denotes maximal performance difference. Clearly, when the training ratio is small, there's a difference between various models, especially for *USPS*, *Yaleface* and *Ligand*. For instance, the difference could be as large as 9% when only 2% *USPS* data is used for training. This kind of classification with rare samples are common for applications like object recognition. Data *Bio* and *Letter* demonstrate medium difference (between $1-2\%$). But for other data sets like *Yeast*, the difference ($< 1\%$) is negligible.

The difference diminishes as training data increases. This is common for all data sets. When training samples are moderately large, kernel regularization actually has no much effect. It works only when the training samples are few.

Which model excels?

Here, we study which model (Independent, Partial or Same) excels if there's a difference. In Table 3-5, the entries in bold

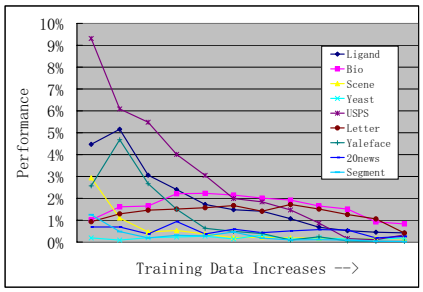

Figure 1: Performance Difference

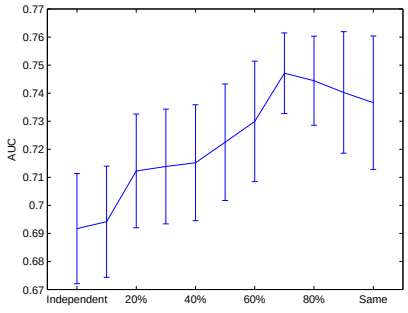

Figure 2: Performance on *Ligand*

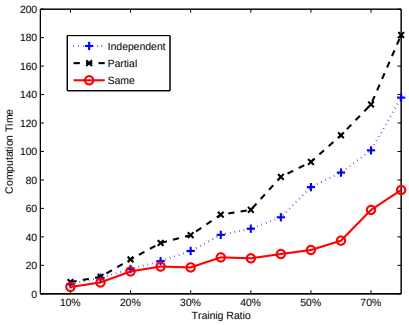

Figure 3: Efficiency Comparison

denote the best one in each setting. It is noticed that Same Model tends to be the winner or a close runner-up most of the time. This trend is observed for almost all the data. Figure 2 shows the average performance and standard deviation of different models when 10% of Ligand data are used for training. Clearly, a general trend is that sharing the same kernel is likely to be more robust compared with Independent Model. Note that the variance is large because of the small sample for training. Actually, Same Model performs best or close in 25 out of 30 trials.

So it is a wiser choice to share the same kernel even if binary classification tasks are quite different. Independent Model is more likely to overfit the data. Partial model, on the other hand, takes the winner only if it is close to the same model (sharing 90% kernel as in Table 4). Mostly, its performance stays between Independent and Same Model.

Why is Same Model better?

As have demonstrated, Same Model outperforms Partial and Independent Model. In Table 2, we show the average kernel weights of 30 runs when 2% of USPS data is employed for training. Each column stands for a kernel. The weights for kernel 8-10 are not presented as they are all 0. The first 2 blocks represents the kernel weights of each class obtained via Independent and Partial Model sharing 50% kernel, respectively. The row follows are the weights produced by Same Model. All the models, no matter which class, prefer to choose $K5$ and $K6$. However, Same Model assigns a very large weight to the 6-th kernel. In the last row, we also present the weight obtained when 60% of data is used for training, in which case Independent Model and Same Model tend to select almost the same kernels. Compared with others, the weights of Same Model obtained using 2% data is closer to the solution obtained with 60% data. In other words, forcing all the binary classification tasks to share the same kernel is tantamount to increasing data samples for training, resulting in a more robust kernel.

Which model is more scalable?

Regularization on kernel difference seems to affect marginally when the samples are more than few. Thus, a method requiring less computational cost is favorable.

Our algorithm consists of multiple iterations. In each iteration, we need to solve k SVMs given fixed kernels. For each binary classification problem, combining the kernel matrix costs $O(pn^2)$. The time complexity of SVM is $O(n^\eta)$ with $\eta \in [1, 2.3]$ [Platt, 1999]. After that, a LP with

Table 2: Kernel Weights of Different Models

		K1	K2	K3	K4	K5	K6	K7
I	C_1	0	0	0	0	.25	.43	.32
	C_2	.04	.01	0	0	.03	.92	0
	C_3	0	0	0	0	.11	.59	.30
	C_4	0	0	0	0	.34	.53	.12
	C_5	0	0	0	.10	.42	.42	.06
	C_6	0	0	0	.02	.41	.46	.10
	C_7	0	.03	0	0	.39	.49	.09
	C_8	.00	0	.03	.20	.16	.54	.06
	C_9	0	0	0	.14	.39	.39	.08
	C_{10}	.10	.02	.02	.24	.15	.47	.00
P	C_1	.02	.01	0	0	.19	.60	.19
	C_2	.03	.01	0	0	.05	.91	0
	C_3	.02	.01	0	0	.09	.70	.18
	C_4	.02	.01	0	0	.22	.67	.09
	C_5	.02	.01	0	.07	.23	.65	.03
	C_6	.02	.01	0	.03	.23	.67	.05
	C_7	.02	.02	0	.04	.19	.67	.05
	C_8	.02	.01	.03	.08	.12	.68	.06
	C_9	.02	.01	0	.08	.22	.63	.05
	C_{10}	.08	.02	0	.14	.11	.65	.00
S	–	**.03**	**.01**	**0**	**0**	**.07**	**.88**	**.01**
Tr=60%	–	0	0	0	0	.07	.93	0

$O(pk)$ variables (the kernel weights) and increasing number of constraints needs to be solved. We notice that the algorithms terminates with dozens of iterations and SVM computation dominates the computation time in each iteration if $p << n$. Hence, the total time complexity is approximately $O(Ikpn^2) + O(Ikn^\eta)$ where I is the number of iterations.

As for Same Model, the same kernel is used for all the binary classification problems and thus requires less time for kernel combination. Moreover, compared with Partial Model, only $O(p)$, instead of $O(pk)$ variables (kernel weights), need to be determined, resulting less time to solve the LP. With Independent Model, the total time for SVM training and kernel combination remains almost the same as Partial. Rather than one LP with $O(pk)$ variables, Independent needs to solve k LP with only $O(p)$ variables in each iteration, potentially saving some computation time. One advantage of Independent Model is that, it decomposes the problem into multiple independent kernel selection problem, which can be paralleled seamlessly with a multi-core CPU or clusters.

In Figure 3, we plot the average computation time of various models on *Ligand* data on a PC with Intel P4 2.8G CPU and 1.5G memory. We only plot Partial model sharing 50% kernel to make the figure legible. All the models yield simi-

Table 3: Ligand Result

Tr Ratio	10%	15%	20%	25%	30%	35%	40%	45%	50%	60%	70%	80%
Independent	69.17	77.30	79.22	81.01	80.92	82.73	82.85	83.95	83.83	85.42	86.67	85.76
20%	71.23	77.43	79.33	81.07	81.01	82.80	82.92	84.03	83.90	85.47	86.70	85.80
40%	71.52	77.88	80.34	81.17	81.55	82.90	83.01	84.18	84.01	85.53	86.80	85.92
60%	72.99	79.71	81.39	82.09	82.28	83.30	83.69	84.49	84.29	85.71	86.94	86.12
80%	**74.44**	**80.65**	81.75	82.83	82.86	**83.66**	**84.35**	**84.78**	84.45	**85.88**	**86.99**	**86.30**
Same	73.66	**80.65**	**81.95**	**82.90**	**82.90**	**83.64**	**84.34**	**84.79**	**84.52**	**85.83**	**86.98**	**86.29**
Diff	5.54	3.44	2.73	1.93	1.97	0.93	1.52	0.84	0.69	0.46	0.33	0.54

Table 4: Bio Result

Tr Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%	20%	30%
Independent	60.13	63.84	66.31	67.51	69.18	71.42	72.24	73.18	73.69	74.95	79.81	81.95
20%	60.24	64.38	66.90	68.57	70.19	72.33	73.09	73.87	74.23	75.50	80.08	82.15
40%	60.37	64.86	67.30	69.10	70.67	72.86	73.59	74.35	74.66	75.88	80.33	82.40
60%	60.71	65.21	67.77	69.47	71.06	73.27	73.95	74.73	75.02	76.21	80.56	82.61
80%	60.92	65.40	**67.96**	69.68	71.38	73.52	74.22	75.03	75.27	76.42	80.72	82.76
90%	**60.99**	**65.45**	67.94	**69.72**	**71.41**	**73.57**	**74.25**	**75.10**	**75.34**	**76.46**	**80.73**	**82.78**
Same	59.98	65.21	67.51	69.52	71.37	73.43	74.13	75.04	**75.34**	76.44	80.70	82.73
Diff	1.01	1.61	1.65	2.21	2.23	2.15	2.01	1.92	1.65	1.51	0.92	0.83

Table 5: USPS Result

Tr Ratio	2%	3%	4%	5%	6%	7%	8%	9%	10%	20%	40%	60%
Independent	49.09	60.54	64.57	69.28	72.44	75.08	77.24	78.84	80.69	86.35	90.12	91.96
20%	51.50	61.39	65.14	70.19	72.96	75.44	77.53	79.10	80.90	86.47	90.18	92.04
40%	53.27	62.48	65.86	71.19	73.63	75.84	77.82	79.47	81.06	86.49	90.20	92.20
60%	54.64	63.71	67.22	72.01	74.33	76.29	78.27	79.85	81.24	86.51	90.23	92.20
80%	56.39	65.18	68.47	72.61	74.93	76.70	78.63	80.06	81.45	86.50	90.22	92.29
Same	**58.40**	**66.63**	**70.05**	**73.29**	**75.49**	**77.07**	**79.08**	**80.31**	**81.56**	86.46	90.21	92.28
Diff(%)	9.31	6.09	5.48	4.01	3.05	1.99	1.84	1.47	0.87	0.16	0.11	0.33

lar magnitude with respect to number of samples, as we have analyzed. But Same Model takes less time to arrive at a solution. Similar trend is observed for other data sets as well.

Same Model, with more strict constraints, is indeed more efficient than Independent and Partial model if parallel computation is not considered. So in terms of both classification performance and efficiency, Same model should be preferred. Partial Model, seemingly more reasonable to match the relationship between labels, should not be considered given its marginal improvement and additional computational cost. This conclusion, as we believe, would be helpful and suggestive for other practitioners.

A special case: Average Kernel

Here we examine one special case of Same Model: average of base kernels. The first block of Table 6 shows the performance of MKL with Same Model compared with average kernel on Ligand over 30 runs. Clearly, Same Model is almost always the winner. It should be emphasized that simple average actually performs reasonably well, especially when base kernels are good. The effect is mostly observable when samples are few (say only 10% training data). Interestingly, as training samples increases to 60%-80%, the performance with Average Kernel decreases. However, Same Model's performance improves consistently. This is because Same Model can learn an optimal kernel while Average does not consider the increasing label information for kernel combination.

A key difference between Same Model and Average Kernel is that the solution of the former is sparse. For instance, Same Model on Ligand data picks 2-3 base kernels for the final solution while Average has to consider all the 15 base kernels. For data like microarray, graphs, and structures, some specialized kernels are computationally expensive. This is even worse if we have hundreds of base kernels. Thus, it is desirable to select those relevant kernels for prediction. Another potential disadvantage with Average Kernel is robustness. To verify this, we add an additional linear kernel with random noise. The corresponding performance is presented in the 2nd block of Table 6. The performance of Average Kernel deteriorates whereas Same Model's performance remains nearly unchanged. This implies that Average Kernel can be affected by noisy base kernels whereas Same Model is capable of picking the right ones.

6 Conclusions

Kernel learning for multi-label or multi-class classification problem is important in terms of kernel parameter tuning or heterogeneous data fusion, whereas it is not clear whether a specific kernel or the same kernel should be employed in practice. In this work, we systematically study the effects of different kernel sharing strategies. We present a unified framework with kernel regularization such that flexible degree of kernel sharing is viable. Under this framework, three different models are compared: Independent, Partial and Same Model. It turns out that the same kernel is preferred for classification even if the labels are quite different.

When samples are few, Same Model tends to yield a more robust kernel. Independent Model, on the contrary, is likely to learn a 'bad' kernel due to over-fitting. Partial Model, occasionally better, lies in between most of the time. However, the difference of these models vanishes quickly with increas-

Table 6: Same Model compared with Average Kernel on Ligand Data. The 1st block is the performance when all the kernels are reasonably good; The 2nd block is the performance when a noisy kernel is included in the base kernel set.

	Tr_ratio	10%	15%	20%	25%	30%	35%	40%	45%	50%	60%	70%	80%
Good Kernels	Same	73.66	**80.65**	**81.95**	**82.90**	**82.90**	**83.64**	**84.34**	**84.79**	**84.52**	**85.83**	**86.98**	**86.29**
	Average	**77.12**	79.67	80.69	81.72	82.01	82.42	82.52	82.19	81.83	80.76	78.44	76.17
Noisy Kernels	Same	**73.69**	**80.64**	**81.92**	**82.92**	**82.92**	**83.63**	**84.36**	**84.72**	**84.53**	**85.84**	**86.94**	**86.32**
	Average	73.32	78.41	79.08	79.81	78.98	80.29	79.72	79.81	79.12	77.80	76.11	71.67

ing training samples. All the models yield similar classification performance when samples are large. In this case, Independent and Same are more efficient. Same Model, a little bit surprising, is the most efficient method. Partial Model, though, asymptotically bears the same time complexity, often needs more computation time.

It is observed that for some data, simply using the average kernel (which is a special case of Same Model) with a proper parameter tuning for SVM occasionally gives reasonable good performance. This also confirms our conclusion that selecting the same kernel for all labels is more robust in reality. However, this average kernel is not sparse and can be sensitive to noisy kernels. In this work, we only consider kernels in the input space. It could be interesting to explore the construction of kernels in the output space as well.

Acknowledgments

This work was supported by NSF IIS-0612069, IIS-0812551, CCF-0811790, NIH R01-HG002516, and NGA HM1582-08-1-0016. We thank Dr. Rong Jin for helpful suggestions.

A Proof for Theorem 3.1

Proof. Based on Eq. (4), (5) and (6), we can assume

$$\sum_{i=1}^{p}\zeta_i = c_1,\ \ \sum_{i=1}^{p}\gamma_i^t = c_2,\ \ c_1+c_2=1,\ \ c_1,c_2\geq 0.$$

Let $\mathcal{G}^t(\alpha) = [\alpha^t]^T\left(\mathcal{G}^t\circ \mathbf{y}^t[\mathbf{y}]^T\right)\alpha^t$ and $G_i^t(\alpha)$ as in Eq. (8). Then Eq. (3) can be reformulated as:

$$\begin{aligned}&\max_{\alpha^t}\min_{\{\mathcal{G}^t\}}\sum_{t=1}^{k}[\alpha^t]^T\mathbf{e}-\frac{1}{2}\sum_{t=1}^{k}\left\{-\lambda c_2+\mathcal{G}^t(\alpha)\right\}\\ =\ &\max_{\alpha^t}\sum_{t=1}^{k}[\alpha^t]^T\mathbf{e}-\frac{1}{2}\max_{\zeta_i,\gamma_i^t}\sum_{t=1}^{k}\left\{-\lambda c_2+\sum_{i=1}^{p}(\zeta_i+\gamma_i^t)G_i^t(\alpha)\right\}\end{aligned}$$

It follows that the 2nd term can be further reformulated as

$$\begin{aligned}&\max_{c_1,c_2,\zeta_i,\gamma_i^t}\sum_{t=1}^{k}\left\{\sum_i\zeta_i G_i^t(\alpha)+\sum_{i=1}^{p}\gamma_i^t G_i^t(\alpha)-\lambda c_2\right\}\\ =\ &\max_{c_1,c_2}\left\{\max_{\sum_i\zeta_i=c_1}\sum_{i=1}^{p}\zeta_i\sum_{t=1}^{k}G_i^t(\alpha)\right.\\ &\qquad\left.+\sum_{t=1}^{k}\max_{\sum_i\gamma_i^t=c_2}\sum_{i=1}^{p}\gamma_i^t G_i^t(\alpha)-k\lambda c_2\right\}\\ =\ &\max_{c_1,c_2}\left\{c_1\max_i\sum_{t=1}^{k}G_i^t(\alpha)+\sum_{t=1}^{k}c_2\max_i G_i^t(\alpha)-k\lambda c_2\right\}\\ =\ &\max_{c_1+c_2=1}\left\{c_1\max_i\sum_{t=1}^{k}G_i^t(\alpha)+c_2\left[\sum_{t=1}^{k}\max_i G_i^t(\alpha)-k\lambda\right]\right\}\\ =\ &\max\left\{\max_i\sum_{t=1}^{k}G_i^t(\alpha),\ \ \left[\sum_{t=1}^{k}\max_i G_i^t(\alpha)-k\lambda\right]\right\}\end{aligned}$$

By adding constraints as

$$\begin{aligned}&s\geq s_0, s\geq\sum_{t=1}^{k}s_t-k\lambda\\ &s_0\geq\sum_{t=1}^{k}G_i^t(\alpha),\quad i=1,\cdots,p\\ &s_t\geq G_i^t(\alpha),\quad i=1,\cdots,p,\ \ t=1,\cdots,k\end{aligned}$$

We thus prove the Theorem. □

References

[Bach *et al.*, 2004] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

[Jebara, 2004] Tony Jebara. Multi-task feature and kernel selection for svms. In *ICML*, 2004.

[Ji *et al.*, 2008] S. Ji, L. Sun, R. Jin, and J. Ye. Multi-label multiple kernel learning. In *NIPS*, 2008.

[Kolda, 1997] Tamara G. Kolda. *Limited-memory matrix methods with applications*. PhD thesis, 1997.

[Lanckriet *et al.*, 2004a] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5, 2004.

[Lanckriet *et al.*, 2004b] Gert R. G. Lanckriet, et al. Kernel-based data fusion and its application to protein function prediction in yeast. In *PSB*, 2004.

[Platt, 1999] John C. Platt. Fast training of support vector machines using sequential minimal optimization. 1999.

[Rakotomamonjy *et al.*, 2007] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. More efficiency in kernel learning. In *ICML*, 2007.

[Rifkin and Klautau, 2004] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *JMLR*, 5, 2004.

[Sonnenburg *et al.*, 2007] Sren Sonnenburg, Gunnar Rtsch, Christin Schfer, and Bernhard Schlkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2007.

[Tang *et al.*, 2009] Lei Tang, Jianhui Chen, and Jieping Ye. On multiple kernel learning with multiple labels. Technical report, Arizona State University, 2009.

[Tsuda and Noble, 2004] Koji Tsuda and William Stafford Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20:326–333, 2004.

[Zien, 2007] Alexander Zien. Multiclass multiple kernel learning. In *ICML*, 2007.