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 pre-specified 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 $\{f^i\}_{i=1}^k$ respectively, based on $n$ training samples $\{(x_i, y_i)\}_{i=1}^n$, where $i$ is the index of a specific label. Let $\mathcal{H}_K^i$ be the feature space, and $\phi^i_K$ be the mapping function defined as $\phi^i_K: \mathcal{H}_K^i \rightarrow \mathcal{H}_K^i$, for a kernel function $K^i$. Let $\mathcal{G}^i$ be the kernel (Gram) matrix for the $t$-th task, namely $\mathcal{G}^i_{ij} = K^i(x_i, x_j) = \langle \phi^i_K(x_i), \phi^i_K(x_j) \rangle$. Under the setting of learning multiple labels $\{f^i\}_{i=1}^k$ using SVM, each label $f^i$ can be seen as learning a linear function in the feature space $\mathcal{H}_K^i$, such that $f^i(x) = \langle w^i, \phi^i_K(x) \rangle + b^i$ where $w^i$ is the feature weight and $b^i$ is the bias term.

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

$$D(\alpha^i, \mathcal{G}^i) = [\alpha^i]^T e - \frac{1}{2} [\alpha^i]^T (\mathcal{G}^i \circ y^i [y^i]^T) \alpha^i$$

(1)

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

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

(2)

Here, $C$ is the penalty parameter for allowing the misclassification. Given $\{\mathcal{G}^i\}_{i=1}^k$, optimal $\{\alpha^i\}_{i=1}^k$ 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 $k$ kernels and $p$ base kernels $G_1, G_2, \cdots, G_p$ can be formulated as:

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

(3)\hspace{1cm}(4)\hspace{1cm}(5)

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

Clearly, if $\lambda$ is set to $0$, the objective goal is decoupled into $k$ sub-problems, with each selecting a kernel independently (Independent Model); When $\lambda$ 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 $\lambda$ 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 $\theta^i$ essentially represents the kernel associated with the $t$-th label. We decouple the kernel weights of each label into two non-negative parts:

$$\theta_i^t = \zeta_i + \eta_i^t, \ \zeta_i, \eta_i^t \geq 0$$

(6)

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

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

(7)

For presentation convenience, we denote

$$G_i^t(\alpha) = [\alpha^i]^T (G_i \circ y^i [y^i]^T) \alpha^i$$

(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):

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

The kernel weights of each label ($\zeta_i, \eta_i^t$) 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 $\lambda$ is equivalent to the following problem with a proper $\beta$ and other constraints specified in (3):

$$\begin{align*}
\min_{\{\mathcal{G}^i\}_{i=1}^k} \ & \sum_{i=1}^k \left\{[\alpha^i]^T e - \frac{1}{2} [\alpha^i]^T (\mathcal{G}^i \circ y^i [y^i]^T) \alpha^i\right\} \\
\text{s.t.} \ & \Omega(\{\mathcal{G}^i\}_{i=1}^k) \leq \beta
\end{align*}$$

(9)\hspace{1cm}(10)

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

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

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

with $S(t) = \{\alpha^t | 0 \leq \alpha^t \leq C, [\alpha^t]^T y^t = 0\}$ (13)

Note that the max operation with respect to $\alpha^t$ is transformed into a constraint for all the possible $\alpha^t$ 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 $\alpha^t$ iteratively. In the $J$-th iteration, we perform the following:

1. Given $\theta_{i}^t$ and $g^t$ from previous iteration, find out new $\alpha_{i}^t$ in the set (13) which violates the constraints (12) most for each label. Essentially, we need to find out $\alpha^t$ 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:

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

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 $\alpha_{i}^t$ obtained in Step 1, add $k$ linear constraints:

$$\theta_{i}^t D(\alpha_{i}^t, G_i) \leq g^t, \quad t = 1, \ldots, k$$

and find out new $\theta^t$ and $g^t$ via the problem below:

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

$$\text{s.t.} \sum_{i=1}^{p} \theta_{i}^t D(\alpha_{i}^t, G_i) \leq g^t, \quad j = 1, \ldots, J$$

$$\theta^t \geq 0, \quad \sum_{i=1}^{p} \theta_{i}^t = 1, \quad t = 1, \ldots, k$$

$$\frac{1}{2} \sum_{t=1}^{k} \sum_{i=1}^{p} \gamma_{i}^t \leq \beta, \quad \theta^t = \zeta_i + \gamma_{i}^t, \quad \zeta_i, \gamma_{i}^t \geq 0$$

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 $\alpha$ 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>
<thead>
<tr>
<th>Data</th>
<th>#samples</th>
<th>#labels</th>
<th>#kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USPS</td>
<td>1000</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Letter</td>
<td>1300</td>
<td>26</td>
<td>10</td>
</tr>
<tr>
<td>Yaleface</td>
<td>1000</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20news</td>
<td>2000</td>
<td>20</td>
<td>62</td>
</tr>
<tr>
<td>Segment</td>
<td>2310</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Multi-class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Letter</td>
<td>1300</td>
<td>26</td>
<td>10</td>
</tr>
<tr>
<td>Yaleface</td>
<td>1000</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20news</td>
<td>2000</td>
<td>20</td>
<td>62</td>
</tr>
<tr>
<td>Segment</td>
<td>2310</td>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 1: Data Description

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 $\beta$ 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
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 represent 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 $O(n^3)$ with $\eta \in [1,2,3]$ [Platt, 1999]. After that, a LP with $O(pk)$ 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 $O(1kp^2n^2) + O(1kn^3)$ 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 2: Kernel Weights of Different Models**

<table>
<thead>
<tr>
<th>Model</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
<th>K4</th>
<th>K5</th>
<th>K6</th>
<th>K7</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.25</td>
<td>.43</td>
<td>.32</td>
</tr>
<tr>
<td>C</td>
<td>.04</td>
<td>.01</td>
<td>0</td>
<td>0</td>
<td>.03</td>
<td>.92</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.11</td>
<td>.59</td>
<td>.30</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.34</td>
<td>.53</td>
<td>.12</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.10</td>
<td>.42</td>
<td>.42</td>
<td>.06</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.02</td>
<td>.41</td>
<td>.46</td>
<td>.10</td>
</tr>
<tr>
<td>C</td>
<td>.03</td>
<td>0</td>
<td>0</td>
<td>.39</td>
<td>.49</td>
<td>.09</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.00</td>
<td>.03</td>
<td>.20</td>
<td>.16</td>
<td>.54</td>
<td>.06</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.14</td>
<td>.39</td>
<td>.39</td>
<td>.08</td>
</tr>
<tr>
<td>C</td>
<td>.10</td>
<td>.02</td>
<td>.02</td>
<td>.24</td>
<td>.15</td>
<td>.47</td>
<td>.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
<th>K4</th>
<th>K5</th>
<th>K6</th>
<th>K7</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>0</td>
<td>0</td>
<td>.19</td>
<td>.60</td>
<td>.19</td>
</tr>
<tr>
<td>C</td>
<td>.03</td>
<td>.01</td>
<td>0</td>
<td>0</td>
<td>.05</td>
<td>.91</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>0</td>
<td>0</td>
<td>.09</td>
<td>.70</td>
<td>.18</td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>0</td>
<td>0</td>
<td>.22</td>
<td>.67</td>
<td>.09</td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>.07</td>
<td>.23</td>
<td>.65</td>
<td>.03</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>.03</td>
<td>.23</td>
<td>.67</td>
<td>.05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.02</td>
<td>.04</td>
<td>.19</td>
<td>.67</td>
<td>.05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>.03</td>
<td>.08</td>
<td>.12</td>
<td>.68</td>
<td>.06</td>
</tr>
<tr>
<td>C</td>
<td>.02</td>
<td>.01</td>
<td>0</td>
<td>.08</td>
<td>.22</td>
<td>.63</td>
<td>.05</td>
</tr>
<tr>
<td>C</td>
<td>.08</td>
<td>.02</td>
<td>0</td>
<td>.14</td>
<td>.11</td>
<td>.65</td>
<td>.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>S</th>
<th>Tr=60%</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>–</td>
<td>.03</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>.07</td>
</tr>
<tr>
<td>C</td>
<td>.93</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 3: Ligand Result

<table>
<thead>
<tr>
<th>Tr Ratio</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>35%</th>
<th>40%</th>
<th>45%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>69.17</td>
<td>77.30</td>
<td>79.22</td>
<td>81.01</td>
<td>80.92</td>
<td>82.73</td>
<td>82.85</td>
<td>83.95</td>
<td>83.83</td>
<td>85.42</td>
<td>86.67</td>
<td>85.76</td>
</tr>
<tr>
<td>20%</td>
<td>71.23</td>
<td>77.43</td>
<td>79.33</td>
<td>81.07</td>
<td>81.01</td>
<td>82.80</td>
<td>82.92</td>
<td>84.03</td>
<td>83.90</td>
<td>85.47</td>
<td>86.70</td>
<td>85.80</td>
</tr>
<tr>
<td>40%</td>
<td>71.52</td>
<td>77.88</td>
<td>80.34</td>
<td>81.17</td>
<td>81.55</td>
<td>82.90</td>
<td>83.01</td>
<td>84.18</td>
<td>84.01</td>
<td>85.53</td>
<td>86.80</td>
<td>85.92</td>
</tr>
<tr>
<td>60%</td>
<td>72.99</td>
<td>79.71</td>
<td>81.39</td>
<td>82.09</td>
<td>82.28</td>
<td>83.30</td>
<td>83.69</td>
<td>84.49</td>
<td>84.29</td>
<td>85.71</td>
<td>86.94</td>
<td>86.12</td>
</tr>
<tr>
<td>80%</td>
<td><strong>74.44</strong></td>
<td><strong>80.65</strong></td>
<td>81.75</td>
<td>82.83</td>
<td>82.86</td>
<td>83.66</td>
<td>84.35</td>
<td>84.78</td>
<td>84.45</td>
<td><strong>85.88</strong></td>
<td><strong>86.99</strong></td>
<td><strong>86.30</strong></td>
</tr>
<tr>
<td>Same</td>
<td>73.66</td>
<td><strong>80.65</strong></td>
<td>81.95</td>
<td>82.90</td>
<td>82.90</td>
<td>83.64</td>
<td>84.34</td>
<td>84.79</td>
<td>84.52</td>
<td><strong>85.83</strong></td>
<td><strong>86.98</strong></td>
<td><strong>86.29</strong></td>
</tr>
<tr>
<td>Diff</td>
<td>5.54</td>
<td>3.44</td>
<td>2.73</td>
<td>1.93</td>
<td>1.97</td>
<td>0.93</td>
<td>1.52</td>
<td>0.84</td>
<td>0.69</td>
<td>0.46</td>
<td>0.33</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table 4: Bio Result

<table>
<thead>
<tr>
<th>Tr Ratio</th>
<th>1%</th>
<th>2%</th>
<th>3%</th>
<th>4%</th>
<th>5%</th>
<th>6%</th>
<th>7%</th>
<th>8%</th>
<th>9%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>60.13</td>
<td>63.84</td>
<td>66.31</td>
<td>67.51</td>
<td>69.18</td>
<td>71.42</td>
<td>72.24</td>
<td>73.18</td>
<td>73.69</td>
<td>74.95</td>
<td>79.81</td>
<td>81.95</td>
</tr>
<tr>
<td>20%</td>
<td>60.24</td>
<td>64.38</td>
<td>66.90</td>
<td>68.57</td>
<td>70.19</td>
<td>72.33</td>
<td>73.09</td>
<td>73.87</td>
<td>74.23</td>
<td>75.50</td>
<td>80.08</td>
<td>82.15</td>
</tr>
<tr>
<td>40%</td>
<td>60.37</td>
<td>64.86</td>
<td>67.30</td>
<td>69.10</td>
<td>70.67</td>
<td>72.86</td>
<td>73.59</td>
<td>74.35</td>
<td>74.66</td>
<td>75.88</td>
<td>80.33</td>
<td>82.40</td>
</tr>
<tr>
<td>60%</td>
<td>60.71</td>
<td>65.21</td>
<td>67.77</td>
<td>69.47</td>
<td>71.06</td>
<td>73.27</td>
<td>73.95</td>
<td>74.73</td>
<td>75.02</td>
<td>76.21</td>
<td>80.56</td>
<td>82.61</td>
</tr>
<tr>
<td>80%</td>
<td>60.92</td>
<td>65.40</td>
<td><strong>67.96</strong></td>
<td>69.68</td>
<td>71.38</td>
<td>73.52</td>
<td>74.22</td>
<td>75.03</td>
<td>75.27</td>
<td>76.42</td>
<td>80.72</td>
<td>82.76</td>
</tr>
<tr>
<td>90%</td>
<td><strong>60.99</strong></td>
<td><strong>65.45</strong></td>
<td>67.94</td>
<td><strong>69.72</strong></td>
<td><strong>71.41</strong></td>
<td><strong>73.57</strong></td>
<td><strong>74.25</strong></td>
<td><strong>75.10</strong></td>
<td><strong>75.34</strong></td>
<td><strong>76.46</strong></td>
<td><strong>80.73</strong></td>
<td><strong>82.78</strong></td>
</tr>
<tr>
<td>Same</td>
<td>59.98</td>
<td>65.21</td>
<td>67.51</td>
<td>69.52</td>
<td>71.37</td>
<td>73.43</td>
<td>74.13</td>
<td>75.04</td>
<td><strong>75.34</strong></td>
<td>76.44</td>
<td>80.70</td>
<td>82.73</td>
</tr>
<tr>
<td>Diff</td>
<td>1.01</td>
<td>1.61</td>
<td>1.65</td>
<td>2.21</td>
<td>2.25</td>
<td>2.15</td>
<td>2.01</td>
<td>1.92</td>
<td>1.65</td>
<td>1.31</td>
<td>0.92</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 5: USPS Result

<table>
<thead>
<tr>
<th>Tr Ratio</th>
<th>1%</th>
<th>2%</th>
<th>3%</th>
<th>4%</th>
<th>5%</th>
<th>6%</th>
<th>7%</th>
<th>8%</th>
<th>9%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>49.09</td>
<td>60.54</td>
<td>64.57</td>
<td>65.28</td>
<td>69.28</td>
<td>72.44</td>
<td>74.08</td>
<td>77.24</td>
<td>78.84</td>
<td>80.69</td>
<td>86.35</td>
<td>90.12</td>
</tr>
<tr>
<td>20%</td>
<td>51.50</td>
<td>61.39</td>
<td>65.14</td>
<td>70.19</td>
<td>72.96</td>
<td>75.44</td>
<td>77.53</td>
<td>79.10</td>
<td>80.90</td>
<td>86.47</td>
<td>90.18</td>
<td>92.04</td>
</tr>
<tr>
<td>40%</td>
<td>53.27</td>
<td>62.48</td>
<td>65.86</td>
<td>71.19</td>
<td>73.63</td>
<td>75.84</td>
<td>77.82</td>
<td>79.47</td>
<td>81.06</td>
<td>86.49</td>
<td>90.20</td>
<td>92.20</td>
</tr>
<tr>
<td>60%</td>
<td>54.64</td>
<td>63.71</td>
<td>67.22</td>
<td>72.01</td>
<td>74.33</td>
<td>76.29</td>
<td>78.27</td>
<td>79.85</td>
<td>81.24</td>
<td>86.51</td>
<td>90.22</td>
<td>92.29</td>
</tr>
<tr>
<td>80%</td>
<td>56.39</td>
<td>65.18</td>
<td>68.47</td>
<td>72.61</td>
<td>74.93</td>
<td>76.70</td>
<td>78.63</td>
<td>80.06</td>
<td>81.45</td>
<td>86.50</td>
<td>90.22</td>
<td>92.29</td>
</tr>
<tr>
<td>Same</td>
<td><strong>58.40</strong></td>
<td><strong>66.63</strong></td>
<td><strong>70.05</strong></td>
<td><strong>73.29</strong></td>
<td><strong>75.49</strong></td>
<td><strong>77.07</strong></td>
<td><strong>79.08</strong></td>
<td><strong>80.31</strong></td>
<td><strong>81.56</strong></td>
<td><strong>86.46</strong></td>
<td><strong>90.21</strong></td>
<td><strong>92.28</strong></td>
</tr>
<tr>
<td>Diff(%)</td>
<td>9.31</td>
<td>6.09</td>
<td>5.48</td>
<td>4.01</td>
<td>3.05</td>
<td>1.99</td>
<td>1.84</td>
<td>1.47</td>
<td>0.87</td>
<td>0.16</td>
<td>0.11</td>
<td>0.33</td>
</tr>
</tbody>
</table>

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.

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.

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 increases.
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.

<table>
<thead>
<tr>
<th></th>
<th>Same</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>73.66</td>
<td>80.65</td>
</tr>
<tr>
<td>Kernels</td>
<td></td>
<td>81.95</td>
</tr>
<tr>
<td></td>
<td>82.90</td>
<td>82.90</td>
</tr>
<tr>
<td></td>
<td>83.64</td>
<td>84.34</td>
</tr>
<tr>
<td></td>
<td>84.79</td>
<td>84.52</td>
</tr>
<tr>
<td></td>
<td>85.83</td>
<td>86.98</td>
</tr>
<tr>
<td></td>
<td>86.29</td>
<td></td>
</tr>
<tr>
<td>Noisy</td>
<td>73.69</td>
<td>80.64</td>
</tr>
<tr>
<td>Kernels</td>
<td></td>
<td>81.92</td>
</tr>
<tr>
<td></td>
<td>82.92</td>
<td>82.92</td>
</tr>
<tr>
<td></td>
<td>83.63</td>
<td>84.36</td>
</tr>
<tr>
<td></td>
<td>84.72</td>
<td>84.53</td>
</tr>
<tr>
<td></td>
<td>85.84</td>
<td>86.94</td>
</tr>
<tr>
<td></td>
<td>86.32</td>
<td></td>
</tr>
</tbody>
</table>

By adding constraints as
\[ s \geq s_0, s \geq \sum_{t=1}^{k} s_t - k\lambda \]
\[ s_0 \geq \sum_{t=1}^{k} G_t^i(\alpha), \quad i = 1, \cdots, p \]
\[ s_t \geq G_t^i(\alpha), \quad i = 1, \cdots, p, \quad t = 1, \cdots, k \]

We thus prove the Theorem.

References


